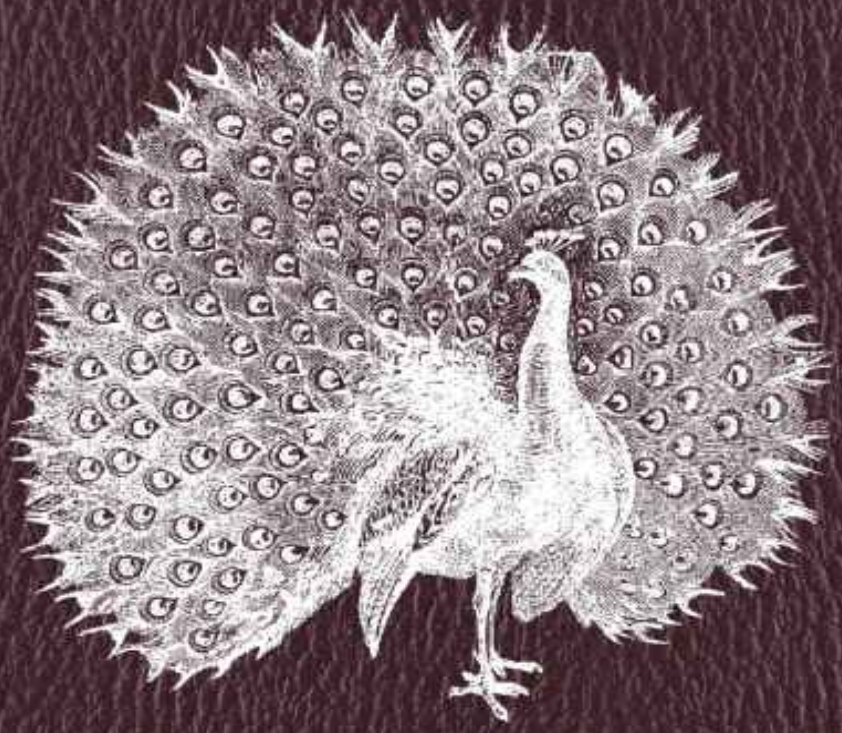
软·件·工·程·师·典·藏·版

开发效率提升10倍

Java 程序开发范例宝典

■ 赛奎春 郭鑫 宋禹蒙 编著



人民邮电出版社
POSTS & TELECOM PRESS

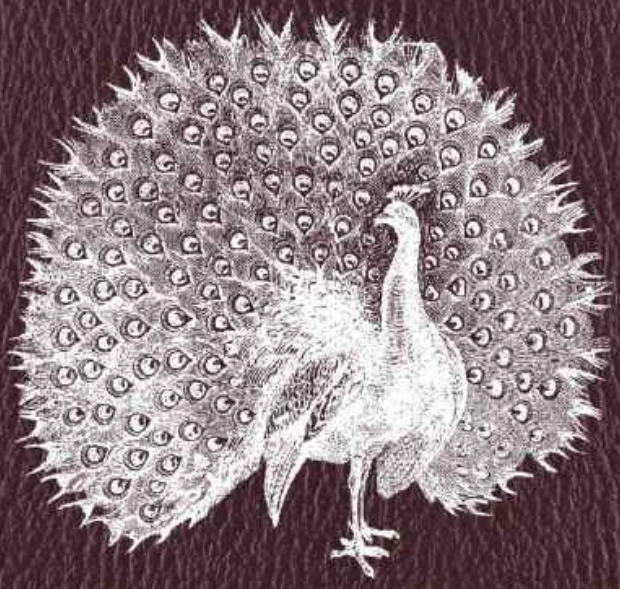
DVD

软·件·工·程·师·典·藏·版

开发效率提升10倍

Java 程序开发 范例宝典

■ 赛奎春 郭鑫 宋禹蒙 编著



人民邮电出版社
POSTS & TELECOM PRESS

DVD

目　录

Java程序开发范例宝典

■赛奎春　郭鑫　宋禹蒙　编著

人民邮电出版社

北京

前言

前些年，笔者参加了一个项目的开发工作，项目要求时间很紧，开发团队几乎是挑灯夜战。当时基于Windows的开发资料很少，网络也不发达，常常为了解决一个问题，大家连续奋战几天、十几天，甚至几十天。之后，笔者又参加了多个项目的开发工作。在开发过程中深刻地感觉到：编程是一项创造性较强的活动，因其涉及面广，开发者往往需要学习、研究各方面的技术和问题；编程水平的提高与开发时间成正比，需要长时间的经验积累和磨炼；编程是一项需要相互学习、相互交流的工作，在交流过程中，不但可分享他人的编程经验、体会，更会产生新的灵感，达到事半功倍的效果。

总之，项目开发从来不是一件容易的事，即使是非常有经验的开发人员，也经常会遇到一些技术难题。要成为一名合格的程序员，就必须不断吸取和借鉴其他开发者的成功经验。阅读别人编写的程序，从中吸取编程思想的精华，这也是学习程序设计最好的方法。

本书内容

本书精选了470个典型实例，所选实例覆盖了开发中的热点问题和关键问题。全书按实际应用进行分类，可以使读者在短时间内掌握更多有用的技术，快速提高编程水平。所选内容均来源于实际项目的开发，有的实例是作者开发实践的积累，有的实例来源于公司的开发项目，还有的来自读者的问题。通过对这些实例进行详细分析和讲解，可以让读者迅速掌握Java项目的开发经验和编程技巧，迅速提高程序设计的综合水平。

全书分为18章，涵盖了窗体与界面设计、控件应用、Commons组件应用、数据库技术、SQL查询相关技术、数据库高级应用、图形图像技术、动画、文件操作技术、操作办公文档、JFreeChart图表、报表打印、操作PDF、解析XML文件、网络技术、邮件收发技术、Java安全、游戏开发等方面内容。

在实例讲解上，全书采用了统一的编排方式，每个实例都包括“实例说明”“技术要点”“实现过程”和“举一反三”4个部分。在“实例说明”中，以图文结合的方式给出了实例的功能说明及运行效果；在“技术要点”中给出了实例的重点、难点技术和相关编程技巧；在“实现过程”中介绍了该实例的设计过程和主要程序代码；在“举一反三”中给出了相关实例的扩展应用。

本书特色

● 所有实例内容都以解决开发人员在编程中遇到的实际问题和开发中应该掌握的技术为中心，每个实例都可以解决某一方面的问题。有的可以解决工作中的难题，有的可以提高工作效率，有的可以提升工作价值。

● 所选实例都具有极强的扩展性，能够给读者以启发，使读者举一反三，开发出非常实用的软件。

● 所选实例都具有代表性，所有实例都提供了源代码，方便读者使用。

本书的约定

书中每个实例的标题栏都给出了程序的特色和实例在光盘中的路径，读者可根据需要学习和使用。

书中涉及数据库的实例，在实例对应文件夹中均提供了数据库文件或数据库文件路径。

书中可能多个实例用到了同一主要技术，为节省篇幅，相关技术的讲解只在一个实例中介绍。

因篇幅限制，书中实例只给出了关键代码，其他代码参见光盘中实例的源程序。

使用本书实例光盘前，请仔细阅读光盘中的“光盘使用说明”。

本书的服务

本书由明日科技组织编写，参加编写的有王小科、王国辉、王占龙、周佳星、张鑫、辛洪郁、赛奎春、高春艳、杨丽、刘佳、刘丽艳、刘红艳、孙雨婷等。由于作者水平有限，错漏之处在所难免，请广大读者批评指正。

如果读者在使用本书时遇到问题，可以访问明日科技网站，我们将通过该网站为读者提供全面的网上服务和支持。对于读者使用本书时遇到的问题，我们将在5个工作日内回复。

服务网站：www.mingribook.com

服务信箱：mingrisoft@mingrisoft.com

服务电话：4006751066

编者

2014年11月

第1章 窗体与界面设计

设置窗体位置
设置窗体大小
设置窗体的标题栏
设置窗体的背景
窗体形状及应用
对话框
MDI窗体的使用
为窗体设置特效

1.1 设置窗体位置

实例001 控制窗体加载时的位置

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\01\Ex01_01

实例说明

第一次运行Windows窗体应用程序时，窗体一般都有一个默认的显示位置，如在桌面上居中显示、在桌面上的任意位置显示等。本实例将通过Java代码控制窗体加载时在桌面上居中显示，实例运行效果如图1.1所示。

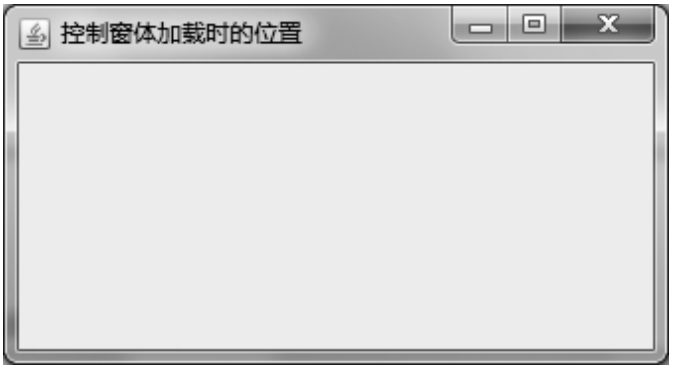


图1.1 控制窗体加载时的位置

技术要点

本实例在控制窗体加载时的位置时主要用到了窗体的setLocationRelativeTo()方法，下面对其进行详细讲解。

setLocationRelativeTo()方法用于设置窗口相对于指定控件的位置。如果控件当前未显示，或者参数 c为null，则此窗口将置于屏幕的中央。该方法的语法格式如下：

```
public void setLocationRelativeTo(Component c)
```

c：确定窗口位置涉及的控件。

实现过程

（1）在项目中创建窗体类LoadPosition。设置窗体的标题文本，为窗体添加WindowListener事件监听器。

（2）编写窗体打开的事件处理方法，在该方法中调用setLocationRelativeTo()设置窗体相对位置。

```
protected void do_this_windowOpened(WindowEvent e) {
  setLocationRelativeTo(null);
        //设置窗体居中
}
```

举一反三

根据本实例，读者可以开发以下程序。

在桌面右下角显示窗体。

使窗体默认居中显示。

实例002 设置窗体在屏幕中的位置

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\01\Ex01_02

实例说明

窗体可以在设置与控件的相对位置时使用空参数使窗体居中显示，但是根据需求，窗体位置的设置应该更加灵活地显示在屏幕上任意位置。本实例将接收用户指定的屏幕坐标，来控制窗体显示的位置，实现窗体位置自定义。实例运行效果如图1.2所示。

图1.2 设置窗体在屏幕中的位置

技术要点

本实例设置窗体在屏幕中的位置时，主要通过JFrame类的setLocation()方法实现，该方法的声明格式如下：

```
public void setLocation(int x, int y)
```

该方法将窗体设置到新位置。通过x和y参数来指定新位置的左上角。

- x：新位置左上角的x 坐标。
- y：新位置左上角的y 坐标。

实现过程

（1）在项目中新建窗体类SetLocation。在窗体中添加两个文本框和一个“设置”按钮。

（2）编写“设置”按钮的事件处理方法，在该方法中获取用户在文本框中输入的数值，并根据该数值来设置窗体在屏幕中的位置。关键代码如下：

```
protected void do_button_actionPerformed(ActionEvent e) {
    Object value=
leftField.getValue();                          //获取左边距
坐标文本
    Object value2=
topField.getValue();                           //获取上边距
坐标文本
```

```
    if (value == null || value2 == null)
      return;
    int left= ((Number)
value).intValue();                          //提取左边距坐标
值
    int top= ((Number)
value2).intValue();                          //提取上边距坐标
值
    setLocation(left,
top);                                       //用左边距和上边距
坐标值设置窗体位置
  }
```

举一反三

根据本实例，读者可以实现以下功能。

根据分辨率的变化动态设置窗体位置。

获取屏幕中心坐标。

实例003 从上次关闭位置启动窗体

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\01\Ex01_03

实例说明

实际开发中，有很多软件都有一个通用的功能，即从上次关闭位置启动窗体，那么可不可以用Java语言实现这样的功能？答案是肯定的。本实例将使用Java语言的Preferences首选项类实现从上次关闭位置启动窗体的功能，实例运行效果如图1.3所示。

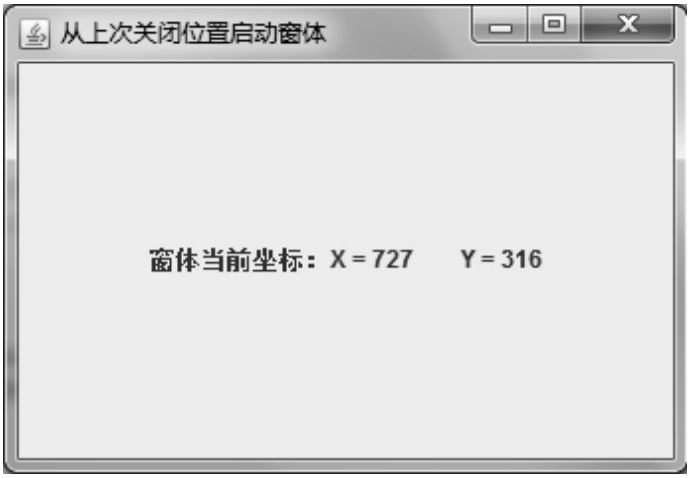


图1.3 从上次关闭位置启动窗体

技术要点

Preferences首选项类的应用，该类的实例对象可以保存程序的各种参数与设置。下面分别介绍该类对象的相关操作。

□ 获取用户的根首选项节点

```
public static Preferences userRoot()
```

□ 向首选项保存整型数值

```
public abstract void putInt(String key, int value)
```

参数说明

● key：要与字符串形式的value 相关联的键。

● value：要与key 相关联的字符串形式的值。

□ 获取首选项中的整数数值

```
public abstract int getInt(String key, int def)
```

参数说明

● key：要作为int 返回其关联值的键。

● def：此首选项节点不具有与key 相关联的值或者无法将该关联值解释为int 或者内部存储不可访问时要返回的默认值。

实现过程

（1）在项目中创建窗体类StartFormByLClosePosition。在窗体中添加一个标签控件用于显示当前窗体坐标。

（2）编写窗体移动的事件处理方法，在该方法中控制标签控件显示当前窗体的位置，只要窗体移动就会立刻更新标签控件的信息。关键代码如下：

```
protected void do_this_componentMoved(ComponentEvent e) {
  Point
location=getLocation();
  //获取窗体坐标
  int x = location.x;
  int y = location.y;
  //把窗体当前坐标显示在标签控件中
  label.setText("窗体当前坐标：X= "+x+ "  Y= "+y);
}
```

（3）编写窗体关闭的事件处理方法，在窗体进行关闭的过程中，这个方法会读取当前窗体的坐标信息并保存到首选项对象中。关键代码如下：

```
protected void do_this_windowClosing(WindowEvent e) {
  Preferences
root=Preferences.userRoot();
  //获取用户首选项
  Point
location=getLocation();
  //获取窗体位置
  root.putInt("locationX",
location.x);                                    //保存窗体X
坐标
```

```
    root.putInt("locationY",
location.y);                                    //保存窗体Y
坐标
  }
```

（4）编写窗体打开的事件处理方法，该方法在窗体打开时被调用，方法中首先获取首选项对象中的坐标信息，然后利用该坐标重新为窗体定位。关键代码如下：

```
  protected void do_this_windowOpened(WindowEvent e) {
    Preferences
root=Preferences.userRoot();
  //获取用户首选项
    int x=
root.getInt("locationX",100);
      //提取窗体X坐标
    int y=
root.getInt("locationY",100);
      //提取窗体Y坐标
    setLocation(x,
y);                                             //恢复
窗体坐标
  }
```

举一反三

根据本实例，读者可以实现以下功能。

将上次关闭位置写入注册表。

使用Preferences首选项保存数据。

实例004 始终在桌面最顶层显示的窗体

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\01\Ex01_04

实例说明

Windows桌面上允许多个窗体同时显示，但是只有一个窗体能够得到焦点，当一个窗体得到焦点后，在其上面的窗体会被得到焦点的窗体遮挡，得到焦点的窗体会显示在最上层，这样被覆盖的窗体就不能完全地显示给用户，也有某些窗体中具有实时性和比较重要的信息需要随时置顶的特殊情况。本实例将实现此功能，运行本实例后，主窗体会始终显示在桌面的最上面。实例运行效果如图1.4所示。

图1.4 始终在桌面最顶层显示的窗体

技术要点

在其他开发语言中实现窗体始终在最顶层比较复杂，但在Java中实现非常简单，只要调用窗体的setAlwaysOnTop()方法即可。下面对

该方法进行介绍。

setAlwaysOnTop()方法可以设置窗体是否置顶显示，也就是说是否使窗体始终显示在其他窗体之上。该方法的语法格式如下：

public final void setAlwaysOnTop(boolean alwaysOnTop)throws SecurityException

alwaysOnTop：如果该属性为true，则窗体保持置顶显示。

实现过程

（1）在项目中新建窗口类AlwaysActiveWindows。在窗体上添加标签控件。

（2）编写窗体界面设计代码，设置窗体标题及添加内容面板与标签控件。最主要的代码在于setAlwaysOnTop()方法设置窗体置顶。关键代码如下：

```
public AlwaysActiveWindows() {
  setTitle("始终在桌面最顶层显示的窗体");                    //设置窗体标题
  setAlwaysOnTop(true);
//设置窗体显示在最顶端（本实例的核心代码）
  setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  setBounds(100,100,319,206);
//设置窗体位置
  contentPane=new JPanel();                                  //创建内容面板
  contentPane.setLayout(new BorderLayout(0, 0));
  setContentPane(contentPane);
  //设置内容面板
  JLabel label=new JLabel("我就在上面不下去了，咋滴。");
  label.setHorizontalAlignment(SwingConstants.CENTER);
```

```
        contentPane.add(label,BorderLayout.CENTER);
        //添加标签控件
    }
```

举一反三

根据本实例，读者可以开发以下程序。

可以将设为最上层的窗体设置成为一个电子表，以便观看时间。

可以将设为最上层的窗体设置成为一个工作计划表，以便随时提醒自己。

1.2 设置窗体大小

实例005 根据桌面大小调整窗体大小

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\01\Ex01_05

实例说明

窗体与桌面的大小比例是软件运行时用户经常会注意到的一个问题。例如，在分辨率为1024×768的桌面上，如果放置一个很大（如1280×1024）或者很小（如10×10）的正方形窗体，会显得非常不协调，正是基于以上这种情况，所以大部分软件的窗体界面都是根据桌面的大小进行自动调整的，本实例就实现这样的功能。实例运行效果如图1.5所示。

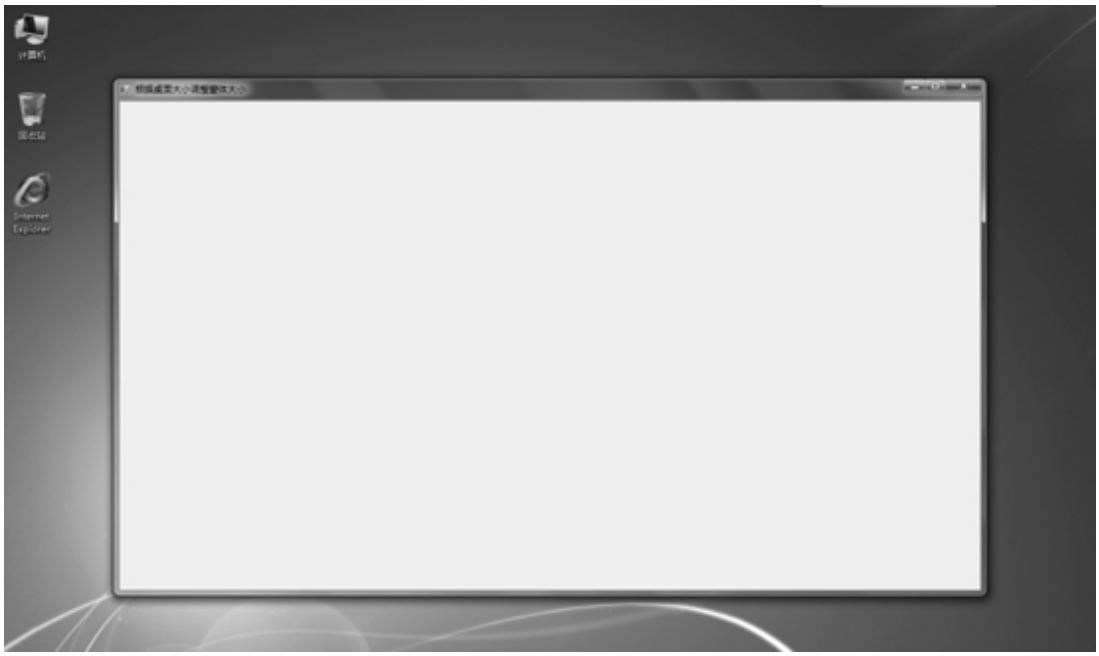


图1.5 根据桌面大小调整窗体大小

技术要点

本实例实现的重点是如何获取桌面的大小，而获取桌面大小时，主要用到窗体的工具包Toolkit类，下面对本实例中用到的关键技术进行详细介绍。

□ 获取窗体工具包

每个窗体类都提供了getToolkit()方法来获取窗体的工具包对象。在窗体内部已经封装了这个工具包，随时可以获取。该方法的声明如下：

```
public Toolkit getToolkit()
```

□ 获取桌面屏幕大小

窗体的工具包提供了方法来获取当前屏幕的大小，该方法的声明如下：

```
public abstract Dimension getScreenSize() throws
HeadlessException
```

实现过程

（1）在项目中创建窗体类SetFormSizeByDeskSize。

（2）编写窗体的打开事件处理方法，该方法在窗体打开时被执行，在方法中，首先获取窗体工具包对象，然后通过工具包对象的getScreenSize()方法获取屏幕的大小，最后把窗体设置为屏幕大小的80%。关键代码如下：

```
protected void do_this_windowOpened(WindowEvent e) {
    Toolkit
toolkit=getToolkit();
        //获得窗体工具包
    Dimension screenSize=
toolkit.getScreenSize();                                    //
获取屏幕大小
```

```
    intwidth=(int)
(screenSize.width*0.8);
  //计算窗体新宽度
    int height=(int)
(screenSize.height*0.8);
    //计算窗体新高度
    setSize(width,height);
                    //设置窗体大小
  }
```

举一反三

根据本实例，读者可以实现以下功能。

根据显示器的分辨率信息设置窗体大小及位置。

根据显示器的分辨率信息调整窗体界面。

实例006 自定义最大化、最小化和关闭按钮

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\01\Ex01_06

实例说明

在制作应用程序时，为了使用户界面更加美观，一般都自己设计窗体的外观，以及窗体的最大化、最小化和关闭按钮。本实例实现设计窗体的外观及最大化、最小化和关闭按钮，再通过鼠标来实现窗体移动效果。实例运行效果如图1.6所示。

图1.6 自定义最大化、最小化和关闭按钮

技术要点

本实例使用的关键技术较多，其中包括取消窗体修饰、按钮外观设置、改变窗体状态等。下面将介绍本实例应用到的这些关键技术。

□ 取消窗体修饰

JFrame窗体默认采用本地系统的窗体修饰，这样会使窗体有标题栏以及标题栏上的所有按钮。但是有些情况需要开发人员根据需求自己定义窗体外观，这时就要禁止JFrame继承本地系统的窗体外观修饰，这可以通过setUndecorated()方法实现。该方法的声明如下：

public void setUndecorated(boolean undecorated)

undecorated：用于指定是否禁止采用本地系统对窗体的修饰，默认值为false，如果该参数为true，窗体将没有任何标题栏内容及窗体边框，它看上去像一块灰色的布料贴在屏幕上。

□ 设置按钮外观

按钮的外观一般需要设置其图标属性，这包括按钮按下与抬起的图标、鼠标经过的图标等。但设置图标无法达到预期效果，因为按钮原有外观与边框会显得不自然，所以要对按钮进行特殊设置。下面介绍有关按钮的关键技术。

● 设置鼠标经过图标

除了setIcon()方法可以为鼠标设置普通状态图标之外，还可以设置按钮的其他状态图标，如设置鼠标经过按钮时显示的图标。这需要

调用按钮的 setRolloverIcon()方法，其方法声明如下：

public void setRolloverIcon(Icon rolloverIcon)

rolloverIcon：鼠标经过按钮时显示的图标对象。

● 取消鼠标外观

要定义鼠标新的外观就必须取消原有外观的绘制，下面介绍关键方法。

```
button.setFocusPainted(false);
    //取消焦点绘制
button.setBorderPainted(false);
  //取消边框绘制
button.setContentAreaFilled(false);
    //取消内容绘制
```

这3个方法分别取消按钮的焦点绘制、边框绘制及内容绘制，这样按钮就没有外观和任何效果了，就像窗体取消修饰效果一样。

□ 改变窗体状态

实例中自定义的最大化、最小化按钮都需要控制窗体的状态，这需要通过 JFrame 类的setExtendedState()方法来实现，其方法声明如下：

public void setExtendedState(int state)

state：该参数是位于JFrame类中的窗体状态常量，其可选值如表1.1所示。

表1.1 窗体状态常量说明

枚 举 值	描 述
ICONIFIED	最小化的窗口
NORMAL	默认大小的窗口
MAXIMIZED_HORIZ	水平方向最大化的窗口
MAXIMIZED_VERT	垂直方向最大化的窗口
MAXIMIZED_BOTH	水平与垂直方向都最大化的窗口

实现过程

（1）在项目中新建窗体类 ControlFormStatus。为窗体添加背景图片，在窗体右上角放置 3个按钮，分别是最小化、最大化和关闭按钮。然后设置窗体的Undecorated属性为true来阻止窗体采用本机系统的修饰，这样窗体就没有标题栏和边框了。

（2）编写最小化按钮的事件处理方法，在该方法中改变窗体的状态值为ICONIFIED最小化常量。关键代码如下：

```
protected void do_button_itemStateChanged(ActionEvent e)
{
  setExtendedState(JFrame.ICONIFIED);
            //窗体最小化
}
```

（3）编写关闭按钮的事件处理方法，在该方法中调用销毁窗体的方法，如果窗体是当前仅剩的唯一窗体，那么程序就会自动退出；如果存在执行业务处理的线程，那么会等待线程结束而关闭虚拟机。关键代码如下：

```
protected void do_button_2_actionPerformed(ActionEvent e)
{
  dispose();
        //销毁窗体
}
```

（4）编写最大化按钮的事件处理方法，该按钮是JToggleButton按钮类的实例对象，所以它有选择与取消选择两种状态，在按钮处于选择状态时，应设置窗体最大化，而当按钮被取消选择时，恢复窗体原有大小。关键代码如下：

```
protected void do_button_1_itemStateChanged(ItemEvent e)
{
```

```
    if (e.getStateChange() == ItemEvent.SELECTED) {
      setExtendedState(JFrame.MAXIMIZED_BOTH);
              //最大化窗体
    } else {
      setExtendedState(JFrame.NORMAL);
              //恢复普通窗体状态
    }
  }
```

（5）编写自定义窗体标题栏面板的鼠标事件处理方法，当用户拖动自定义窗体标题栏时，应该实现窗体移动的效果。关键代码如下：

```
  protected void do_topPanel_mousePressed(MouseEvent e) {
    pressedPoint=
e.getPoint();                                          //记
录鼠标坐标
  }
  protected void do_topPanel_mouseDragged(MouseEvent e) {
    Point point=
e.getPoint();                                           //
获取当前坐标
    Point
locationPoint=getLocation();
      //获取窗体坐标
    int x= locationPoint.x+point.x -
pressedPoint.x;                              //计算移动后的新坐
标
    int y = locationPoint.y + point.y - pressedPoint.y;
```

```
    setLocation(x,y);
            //改变窗体位置
}
```

举一反三

根据本实例，读者可以实现以下功能。

实现隐藏窗体功能。

游戏界面中自定义标题栏。

实例007 禁止改变窗体的大小

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\01\Ex_07

实例说明

本实例主要实现禁止改变窗体大小的功能，运行本实例，默认可以通过鼠标拖曳的方式改变窗体大小，但是当用户单击“禁止改变窗体大小”按钮后，窗体将会以一种对话框的方式进行显示，这时就不可以再用鼠标拖曳的方式改变窗体的大小。实例运行结果如图 1.7所示。

图1.7 禁止改变窗体的大小

技术要点

本实例在实现禁止改变窗体的大小功能时，主要是通过将窗体的resizable属性设置为false实现的。下面介绍设置该属性的方法。

public void setResizable(boolean resizable)

参数说明

resizable：如果此窗体是可调整大小的，则为 true；否则为 false。

实现过程

（1）在项目中新建窗体类LimitChangeFormSize。在该窗体中添加一个按钮控件，用来执行禁止改变窗体大小功能。

（2）编写“禁止改变窗体大小”按钮的事件处理方法，在该方法中设置窗体的resizable属性为false。关键代码如下：

```
protected void do_button_actionPerformed(ActionEvent e) {
  setResizable(false);
            //禁止改变窗体大小
}
```

举一反三

根据本实例，读者可以实现以下功能。

制作屏幕保护程序。

将窗体以对话框形式显示。

1.3 设置窗体的标题栏

实例008 指定窗体标题栏图标

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\01\Ex01_08

实例说明

窗体的标题栏图标也称为窗体的图标，当窗体显示时在左上角标题栏位置会显示这个图标与窗体标题信息，这用于区分不同窗体或标注窗体用途，本实例通过Java语言实现窗体图标的设置，其效果如图1.8 所示，单击任意一个按钮，就会把窗体图标更换为按钮的图标。另外，本实例在Windows 7 系统中开发，程序运行后，除窗体之外，在任务栏也会显示窗体的图标，效果如图1.9所示。

图1.8 设置窗体标题栏图标

图1.9 Windows 7 任务栏图标

技术要点

本实例主要通过设置JFrame窗体类的iconImage属性来实现窗体图标的设定。下面将介绍如何改变该属性的值。

setIconImage()方法是JFrame类提供的，它用于设置窗体标题栏中的图标图片，该方法的声明格式如下：

public void setIconImage(Image image)

参数说明

image：要设置为窗体标题栏图标的图片对象。

实现过程

（1）在项目中新建窗体类FrameIcon。在窗体中设置背景，并添加4个更改窗体图标的按钮。

（2）编写所有按钮的事件处理方法，在用户单击不同按钮时，该方法可以判断事件源并获取相对图标文件的 URL路径，然后通过该URL路径创建图标对象，并指定给窗体的 iconImage属性。关键代码如下：

```
protected void do_button_actionPerformed(ActionEvent e) {
  String resource=
"";                                                    //定
义图标文件名称变量
    if
(e.getSource()==button1)
            //确定用户单击的按钮
```

```
        resource=
    "icon1.png";
      //确定按钮对应的图标文件
      if (e.getSource() == button2)
        resource ="icon2.png";
      if (e.getSource() == button3)
        resource ="icon3.png";
      if (e.getSource() == button4)
        resource ="icon4.png";
      URLurl=getClass().getResource(resource);
                          //获取图标文件路径
      setIconImage(Toolkit.getDefaultToolkit().getImage(url))
;                                 //设置窗体的图标
    }
```

举一反三

根据本实例，读者可以实现以下功能。

将标题栏图标设置为明日科技的公司Logo。

为窗体标题栏设置动态的图标。

实例009 拖动没有标题栏的窗体

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\01\Ex01_09

实例说明

在开发程序的过程中，图形化的窗体（即自定义窗体，一般没有标题栏）固然很吸引用户的眼球，但往往会因为这种窗体无法拖动而

使用户放弃设计这种窗体的可能性，本实例实现拖动没有标题栏的窗体。运行程序，用户可以随意拖动窗体，其结果如图1.10所示。

图1.10 拖动没有标题栏的窗体程序界面

技术要点

本实例的关键技术完全在于鼠标事件的处理方法，通过控件的鼠标拖曳事件来实现窗体的移动，要注意拖曳时鼠标坐标是不停变换的，所以要在鼠标按键按下的事件中记录鼠标坐标，然后在拖曳事件中计算窗体位置与拖曳的控件位置的差距，然后改变窗体位置，这就需要对鼠标事件监听器有一定的了解，事件监听器中的方法可以处理各种事件。下面介绍本实例使用的事件监听器中的关键事件处理方法。

□ MouseListener 事件监听器

该事件监听器用于监听鼠标按键按下与抬起、鼠标单击、鼠标进入与离开控件区域的事件监听，并在事件监听器接口中定义了相应的事件处理方法，把具体事件委托给指定方法去实现事件处理。而本实例中主要应用了该事件监听器中的 mousePressed()实现鼠标按键按下时的事件处理。该方法的声明如下：

```
public void mousePressed(MouseEvent event) {
  //事件处理代码
}
```

参数说明

event：鼠标事件对象，该对象可以获取事件源与鼠标当前坐标。

□ MouseMotionListener 事件监听器

该监听器可以监听鼠标的移动与拖曳动作，其中本实例实现的是拖曳动作的时间静态方法，该方法的声明如下：

```
public void mouseDragged(MouseEvent e) {
  //拖曳事件处理
}
```

参数说明

event：鼠标事件对象，该对象可以获取事件源与鼠标当前坐标。

实现过程

（1）在项目中新建窗体类DropCustomFrame。在窗体中添加一个“关闭”按钮。

（2）编写“关闭”按钮的事件处理方法，由于没有窗体标题栏，无法通过GUI关闭窗体，所以提供该按钮，并在按钮的事件处理方法中实现程序退出功能。关键代码如下：

```
protected void do_button_actionPerformed(ActionEvent e) {
  dispose();
        //销毁窗体
}
```

（3）编写内容面板的鼠标按下事件处理方法。在该方法中要记录鼠标按下按键时的鼠标坐标位置。关键代码如下：

```
protected void do_backgroundPanel_mousePressed(MouseEvent
e) {
```

```
    pressedPoint=
e.getPoint();                                          //记
录鼠标坐标
  }
```

（4）编写鼠标拖曳事件处理方法，在该方法中计算鼠标位置与窗体位置的差距，然后再计算窗体应该移动多少像素，并更改窗体位置。关键代码如下：

```
    protected void do_backgroundPanel_mouseDragged(MouseEvent
e) {
      Point point=
e.getPoint();                                          //
获取当前坐标
      Point
locationPoint=getLocation();
        //获取窗体坐标
      int x= locationPoint.x+point.x -
pressedPoint.x;                              //计算移动后的新坐
标
      int y = locationPoint.y + point.y - pressedPoint.y;
      setLocation(x,
y);                                                    //改变
窗体位置
  }
```

举一反三

根据本实例，读者可以实现以下功能。

制作禁止改变窗体大小的窗体。

制作一个在固定位置打开的窗体。

实例010 取消窗体标题栏与边框

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\01\Ex01_10

实例说明

普通窗体都有标题栏与边框，标题栏可以显示窗体图标、标题和窗体控制按钮。但是随着客户的需求不同，会有各式各样的窗体需要自定义，例如，本实例就是根据客户需求把关于信息的窗体标题栏与窗体边框都去掉了。实例运行效果如图1.11所示。

图1.11 没有窗体标题栏与边框的窗体

技术要点

本实例使用的关键技术请参见实例006。

实现过程

（1）在项目中新建窗体类CancelFrameTitleBorder。

（2）在窗体中添加关于信息与“关闭”按钮，并对窗体所有组件进行布局，最关键的是取消本地系统对窗体的修饰效果。关键代码如下：

```
public CancelFrameTitleBorder() {
  //设置背景色
  getContentPane().setBackground(new Color(240, 255,
255));
```

```
    setUndecorated(true);
            //取消窗体修饰效果
    setTitle("关于进销存管理系
统");                                          //设置标题栏
    getContentPane().setLayout(null);
    setBounds(100, 100, 354, 206);
    setLocationRelativeTo(null);
                //窗体居中
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    final JLabel label=new
JLabel();                                       //用标签显示
logo
    label.setIcon(new
ImageIcon(getClass().getResource("logo.png")));
    label.setBounds(10, 27, 112, 98);
    getContentPane().add(label);
    textArea=new
JTextArea();                                        //用
文本域显示系统信息
    textArea.setOpaque(false);
              //控件透明
    textArea.setText("系统：\n Microsoft Windows Server
2003\n"+" Standard Editon\n Service Pack 2\n\n\n"+"软件：进
销存管理系统\n版权：明日科技");
    textArea.setBounds(154, 6, 187, 154);
    getContentPane().add(textArea);
              //添加控件到窗体
```

```
    JButton button=new
JButton("\u5173\u95ED");                              //创建
“关闭”按钮
    button.addActionListener(newActionListener()
{                                  //添加按钮的事件监听器
      public void actionPerformed(ActionEvent e) {
        do_button_actionPerformed(e);
              //调用按钮事件处理方法
      }
    });
    button.setBounds(230, 172, 90, 30);
    getContentPane().add(button);
              //添加按钮到窗体
  }
```

（3）编写“关闭”按钮的事件处理方法，该方法在窗体没有标题栏和控制按钮的情况下，实现程序退出功能。关键代码如下：

```
  protected void do_button_actionPerformed(ActionEvent e) {
    dispose();
      //销毁窗体
  }
```

举一反三

根据本实例，读者可以实现以下功能。

制作一个带透明控件的窗体。

具有提示功能的窗体。

实例011 设置闪烁的标题栏

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\01\Ex01_11

实例说明

在大型项目中常出现多个窗口同时处理并显示业务数据的情况，每个窗口和窗口中的数据分类与重要性都不相同，有的窗口用于显示实时信息，必须时刻保持醒目位置，但是有的信息重要性非常高，必须第一时间让用户注意到。本实例就实现窗体标题栏的闪烁效果，这将以动态的明显的方式突出某窗体的信息的重要性。实例闪烁过程如图1.12所示。

图1.12 闪烁中的窗体标题栏

技术要点

本实例的关键技术在于Timer控件的使用，窗体标题闪烁效果就是依靠该控件不断地产生ActionEvent事件，并在事件处理中实现的。下面介绍该控件的使用。

□ 创建Timer 对象

本实例所使用的是Java.swing包中的Timer对象，而非Java.util包的。创建这个控件的构造方法如下：

```
public Timer(int delay, ActionListener listener)
```

参数说明

- delay：触发事件的时间间隔，单位为毫秒。
- listener：初始事件监听器，用于获取控件的action 事件。

□ 启动Timer 对象

Timer控件的start()方法将启动Timer，使它开始向其侦听器发送动作事件。该方法的声明如下：

public void start()

实现过程

（1）在项目中新建窗体类FlashTitleBar。在窗体中添加标签控件并显示重要信息。

（2）编写窗体打开的事件处理方法，在该方法中创建Timer控件，并在控件内部实现窗体闪烁效果，而且该效果一直循环，每秒闪烁一次。关键代码如下：

```
protected void do_this_windowOpened(WindowEvent e) {
  Timer timer=newTimer(500,newActionListener()
{                    //创建Timer控件
    String
title=getTitle();                                          //获
取窗体标题
    @Override
    publicvoid actionPerformed(ActionEvent e)
{                        //实现窗体闪烁
      if (getTitle().isEmpty())
  {                                    //如果标题为空
        setTitle(title);
        //恢复窗体标题
      } else {
        setTitle("");
      //如果窗体标题不为空，则清空窗体标题
      }
    }
```

```
    });
    timer.start();
        //启动Timer控件
}
```

举一反三

根据本实例，读者可以实现以下功能。

运行时设置控件的位置。

启动和关闭Timer计时器。

1.4 设置窗体的背景

实例012 设置窗体背景颜色为淡蓝色

这是一个可以用来提高基础性能的实例

实例位置：光盘\mingrisoft\01\Ex01_12

实例说明

开发程序时，Windows窗体的背景色默认为灰色，为了能够使窗体看起来更加美观，可以通过编码为Windows窗体设置更好看的颜色。本实例将 Windows 窗体的背景颜色设置成了淡蓝色，实例运行效果如图1.13所示。

图1.13 设置窗体背景颜色为淡蓝色

技术要点

本实例在设置窗体的背景颜色时，主要用到了窗体的background属性，下面对其进行详细讲解。

setBackground()方法用于设置控件的背景色。背景色仅在组件是不透明时才使用，并且只能由 JComponent 或 ComponentUI 实现的子

类使用。JComponent 的直接子类必须重写paintComponent以遵守此属性。setBackground()方法的声明如下：

public void setBackground(Color bg)

参数说明

bg：所需的背景Color对象。

实现过程

（1）在项目中新建窗体类SetFormBackColor。

（2）编码设置窗体中内容面板的背景色。

```
public SetFormBackColor() {
  setTitle("设置窗体背景颜色为淡蓝
色");                                    //设置窗体标题栏
  setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  setBounds(100,100,308,238);
          //设置窗体位置
  contentPane=new
JPanel();                                          //创建内
容面板
  //设置内容面板的背景色
  contentPane.setBackground(new Color(102, 204, 255));
  setContentPane(contentPane);
            //设置窗体内容面板
}
```

举一反三

根据本实例，读者可以实现以下功能。

通过“属性”更快的设置窗体的背景色。

改变进度条的颜色。

实例013 实现带背景图片的窗体

这是一个可以提高基础性能的实例

实例位置：光盘\mingrisoft\01\Ex01_13

实例说明

开发桌面窗体应用程序时，界面的美观是程序的一个重要组成部分，一般的应用程序界面背景都是非常漂亮或者代表实际意义的图片，那么如何为窗体设置背景图片呢？本实例将通过Java代码重写JPanel面板来实现窗体背景图片的设置，实例运行效果如图1.14所示。

图1.14 设置窗体背景为指定图片

技术要点

本实例在设置窗体的背景图片时，继承JPanel自定义了自己的面板组件，并重写了面板绘制方法，还为自己绘制了背景图片。面板绘制方法的声明格式如下：

```
protected void paintComponent(Graphics graphics)
```

参数说明

graphics：控件中的绘图对象。

实现过程

（1）在项目中新建窗体类SetFormBackImage。在窗体中添加自定义的BackgroundPanel面板。

（2）在窗体类的构造方法中设置窗体标题，并为添加的BackgroundPanel 自定义面板设置背景图片。关键代码如下：

```
public SetFormBackImage() {
    setTitle("实现带背景图片的窗
体");                                          //设置窗体标题
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(100,100,450,300);
            //设置窗体位置
    contentPane=new
JPanel();                                          //创建内
容面板
    setContentPane(contentPane);
            //设置窗体内容面板
    contentPane.setLayout(new BorderLayout(0, 0));
    BackgroundPanelbackgroundPanel=newBackgroundPanel();
            //创建背景面板
    backgroundPanel.setImage(getToolkit().getImage(getClass
().getResource("Penguins.jpg")));                  //
设置面板背景图片
    contentPane.add(backgroundPanel);
            //把背景面板添加到窗体内容面板
}
```

（3）继承JPanel类编写自己的面板。自定义面板类名定义为BackgroundPanel，重写JPanel类的paintComponent()方法，在该方法中实现绘制面板背景图片的代码。关键代码如下：

```
protected void paintComponent(Graphics g)
{                                          //重写绘制组件外观
```

```
    if (image != null) {
      g.drawImage(image,0,0,
this);                                                    //绘制图
片与组件大小相同
    }
    super.paintComponent(g);
                    //执行超类方法
}
```

举一反三

根据本实例，读者可以实现以下功能。

为企业管理系统主窗体设置背景图片。

为用户登录窗体设置背景图片。

实例014 使背景图片自动适应窗体的大小

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\01\Ex01_14

实例说明

开发人员在开发桌面应用程序时，有时为一个窗体设置了背景图片，但是由于图片的大小与窗体的大小并不一定相同，所以就可能导致图片显示不全，那么如何来避免这种情况的发生呢？本实例将通过自定义面板控件来实现背景图片自动适应窗体的大小的功能，实例运行效果如图1.15所示。

图1.15 背景图片自动适应窗体的大小

技术要点

本实例在实现使背景图片自动适应窗体的大小功能时，继承JPanel自定义面板控件并重写了paintComponent()方法绘制背景图片，但使用的是绘图对象的drawImage()方法的一种重载格式，它支持图片大小的指定。该方法的声明如下：

```
public abstract boolean drawImage(Image img, int x, int y, int width, int height, ImageObserver observer)
```

方法中的参数说明如表1.2所示。

表1.2 方法中的参数说明

参 数 名	说 明	参 数 名	说 明
x	绘制图片的起始位置的 x 坐标	height	指定绘制图片的高度
y	绘制图片的起始位置的 y 坐标	observer	转换了更多图像时要通知的对象
width	指定绘制图片的宽度		

实现过程

（1）在项目中创建窗体类 AutoImageSizeByForm。在窗体中添加自定义的面板BackgroundPanel控件，并为其设置背景图片。关键代码如下：

```
public AutoImageSizeByForm() {
    setTitle("使背景图片自动适应窗体的大小");                    //设置窗体标题
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
        setBounds(100,100,450,300);
                        //设置窗体位置
        contentPane=new
    JPanel();                                                    //创
    建内容面板
        contentPane.setLayout(new BorderLayout(0, 0));
        BackgroundPanel backgroundPanel = new
    BackgroundPanel();
        backgroundPanel.setImage(getToolkit().getImage(getClass
    ().getResource("Penguins.jpg")));
        //设置背景面板的图片
        setContentPane(contentPane);
        //创建自定义背景面板
        contentPane.add(backgroundPanel,BorderLayout.CENTER);
                            //添加背景面板到内容面板
      }
```

（2）继承JPanel编写自己的面板类BackgroundPanel，重写paintComponent()方法，在该方法中获取面板的宽度与高度，然后将图片以相同的大小绘制在控件上。关键代码如下：

```
    protected void paintComponent(Graphics g)
{                                          //重写绘制组件外观
      if (image != null) {
        intwidth=getWidth();
                  //获取组件大小
        int height = getHeight();
        g.drawImage(image,0,0,width,height,
    this);                                  //绘制图片与组件大小
```

```
相同
  }
  super.paintComponent(g);
                    //执行超类方法
}
```

举一反三

根据本实例，读者可以实现以下功能。

使窗体的背景图片居中显示。

为添加用户窗体设置背景图片。

实例015 背景为渐变色的主界面

这是一个可以启发思维的实例

实例位置：光盘\mingrisoft\01\Ex01_15

实例说明

窗体背景颜色可以通过属性进行设置，但是通过属性设置的窗体背景颜色都是单一的颜色。在以往程序安装界面中，背景色都是上下渐变的蓝色背景，看上去不伤眼睛，而且没有强烈的疲劳感，一时成为安装界面的流行背景。本实例将实现这个背景颜色渐变的效果，使窗体更加美观。实例运行效果如图1.16所示。

图1.16 背景为渐变色的主界面

技术要点

本实例在实现背景色渐变时涉及两个关键技术，这两个关键技术分别是设置填充模式与绘制矩形，下面分别对这两个关键技术进行介绍。

□ 设置渐变填充模式

● 创建渐变填充模式对象

设置填充模式首先要创建填充模式对象，本实例要实现渐变效果，所以创建的是GradientPaint 类的实例对象，创建该对象的构造方法的参数包括填充起点的坐标与颜色、填充终点的坐标与颜色，其方法声明如下：

public GradientPaint(float x1, float y1, Color color1, float x2, float y2, Color color2)

方法中的参数说明如表1.3所示。

表1.3 方法中的参数说明

参 数 名	说 明	参 数 名	说 明
x1	起始位置的 x 坐标	x2	终止位置的 X 坐标
y1	起始位置的 y 坐标	y2	终止位置的 Y 坐标
color1	起始渐变点的颜色	color2	终止渐变点的颜色

● 设置绘图对象填充模式

创建渐变填充模式对象以后需要设置Graphics2D绘图上下文对象的填充属性，然后由此绘图对象绘制的所有图形都使用这个新的填充模式。设置绘图上下文填充模式的方法的声明如下：

public abstract void setPaint(Paint paint)

参数说明

paint：填充模式对象。

□ 绘制矩形图形

设置绘图上下文对象使用渐变填充模式以后，还要以自定义控件相同的大小来绘制控件界面，绘制内容是一个矩形图形，这样可以均匀地遮盖整个控件界面。绘制矩形图形的方法的声明如下：

```
public abstract void fillRect(int x, int y, int width, int height)
```

方法中的参数说明如表1.4所示。

表1.4 方法中的参数说明

参 数 名	说 明
x	绘制矩形起始位置的 x 坐标
y	绘制矩形起始位置的 y 坐标
width	指定绘制矩形的宽度
height	指定绘制矩形的高度

实现过程

（1）在项目中新建窗体类ImageInFormCenter。设置窗体类的标题，在窗体中添加自定义的渐变背景面板。关键代码如下：

```
public ShadeBackgroundImage() {
    setTitle("背景为渐变色的主界面");                    //设置窗体标题
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(100, 100, 450, 300);
    contentPane=new JPanel();                          //创建内容面板
    contentPane.setLayout(new BorderLayout(0, 0));
    setContentPane(contentPane);
    ShadePanel shadePanel=newShadePanel();             //创建渐变背景面板
```

```
        contentPane.add(shadePanel,BorderLayout.CENTER);
                        //添加面板到窗体内容面板
    }
```

（2）继承 JPanel类编写自己的渐变面板控件，重写paintComponent()方法，在该方法中创建GradientPaint填充类的实例对象，然后把它设置为当前绘图对象的填充模式，再使用新的填充模式绘制一个与控件相同大小的矩形。关键代码如下：

```
    protected void paintComponent(Graphics g1)
{                                       //重写绘制组件外观
      Graphics2D g = (Graphics2D) g1;
      super.paintComponent(g);
              //执行超类方法
      intwidth=getWidth();
              //获取组件大小
      int height = getHeight();
      //创建填充模式对象
      GradientPaint paint = new GradientPaint(0, 0,
  Color.CYAN, 0, height, Color.MAGENTA);
      g.setPaint(paint);
                //设置绘图对象的填充模式
      g.fillRect(0,0,width,height);
                //绘制矩形填充控件界面
    }
```

举一反三

根据本实例，读者可以实现以下功能。

将进销存管理系统的背景色设置为从黄到红的渐变色填充。

制作一个渐变色的系统菜单。

实例016 随机更换窗体背景

本实例可以提高工作效率

实例位置：光盘\mingrisoft\01\Ex01_16

实例说明

在一些管理软件中实现随机更换主界面背景可以增加软件的人性化程度，使用户心情愉悦。本实例实现随机更换主界面的功能，每次窗体恢复显示时主界面背景图片都会随机更换。程序运行效果如图1.17所示。

图1.17 随机更换主界面背景

技术要点

Java util 包中的Random 类提供了常用的伪随机数生成方法，类型包括boolean、int、long、double等。下面介绍本实例如何使用该

类的对象生成整型随机数字。

□ 创建随机数对象

Random 的对象可以生成各种类型的随机数字，但在此之前必须先创建它，本实例使用了默认构造方法直接传递的实例对象，这非常简单，无须介绍。所以这里介绍一个接收参数的构造方法，其方法声明如下：

public Random(long seed)

参数说明

seed：创建随机数对象的种子。

□ 生成指定范围的随机整数

Random 的对象可以调用多个方法来生成不同数据类型的随机数，本实例需要随机指定数组下标索引来获取数组中的某个背景图片对象，而数组下标只能是整数，并且不能超出数组范围。所以下面介绍如何生成指定范围的随机整数，其方法声明如下：

public int nextInt(int num)

参数说明

num：要返回的随机数的范围，必须为正数。

实现过程

（1）在项目中新建窗体类RandomBackgroundImage。在窗体中添加自定义的支持背景的面板控件。

（2）编写初始化背景图片数组的initPhotoArray()方法，在该方法中创建图片数组，然后通过for循环初始化数组中的元素，每个元素赋值为一个图片对象。关键代码如下：

```
private void initPhotoArray() {
  images=new
Image[6];                                          //初始化背
景图片数组
```

```
    String photoPath ="";
    for (int i=0; i< images.length; i++)
{                                    //遍历数组并初始化所有元素
      photoPath= "/com/img/photo"+ (i+1)+
  ".jpg";                        //生成文件名
      images[i] = getToolkit()
        .getImage(getClass().getResource(photoPath));
          //初始化数组元素
    }
  }
```

（3）编写窗体激活事件的处理方法，当窗体刚刚被显示，或者从最小化恢复到显示状态时，都会触发这个窗体激活事件。在事件处理方法中，首先生成随机数，然后用这个随机数作为图片数组的下标索引获取一个图片，设置给支持背景的面板控件，最后重新绘制窗体界面。关键代码如下：

```
  protected void do_this_windowActivated(WindowEvent arg0)
{
    Random
  random=newRandom();                                    //创建随
  机数对象
    int num=
  random.nextInt(6);                                      //生
  成随机数
    panel.setImage(images[num]);
        //设置面板背景图片
    repaint();
      //重绘窗体界面
```

}

举一反三

根据本实例，读者可以实现以下功能。

每天更换主程序背景的主界面。

随机更换菜单栏、工具栏图标的主程序。

1.5 窗体形状及应用

实例017 椭圆形窗体界面

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\01\Ex01_17

实例说明

个性的窗体形状可以增加程序的趣味性，可以使程序更具吸引力。见惯了方方正正的矩形窗体，椭圆形的窗体更会使用户眼前一亮。本实例设计一个椭圆形的窗体，运行程序，窗体为椭圆形，如图1.18所示。单击窗体，即可退出程序。

图1.18 椭圆形窗体

技术要点

本实例的关键技术在于椭圆类Ellipse2D的应用与设置窗体形状的API。下面分别对其进行介绍。

□ 创建椭圆对象

Java 的椭圆对象由 Ellipse2D 类定义，该类是一个抽象类，不能实例化，但是在其内部包含了两个静态内部实现类，分别为Double与Float，它们接收double与float类型的参数定义椭圆大小，本实例使用的是Float实现类。下面是该类构造方法的声明：

public Ellipse2D.Float(float x, float y, float w, float h)

方法中的参数说明如表1.5所示。

表1.5 方法中的参数说明

设 置 值	描 述
x	椭圆对应矩形左上角的 x 坐标
y	椭圆对应矩形左上角的 y 坐标
w	椭圆对应矩形的宽度
h	椭圆对应矩形的高度

□ 设置窗体形状

在JDK 1.6 中提供了设置窗体形状的API，通过这个API 可以设置窗体为指定图形。所用的方法的声明如下：

public static void setWindowShape(Window window, Shape shape)

参数说明

● window：窗体对象。

● shape：图形接口的实现。

实现过程

（1）在项目中创建窗体类EllipseFrame，去掉窗体修饰。关键代码如下：

```
public EllipseFrame() {
  setUndecorated(true);
    //去掉窗体修饰
  //省略其他代码
}
```

（2）编写窗体打开时的事件处理方法，在该方法中创建椭圆对象，并把这个椭圆设置为窗体的形状。关键代码如下：

```
protected void do_this_windowOpened(WindowEvent e) {
```

```
        //创建椭圆对象
        Ellipse2D.Float ellipse = new Ellipse2D.Float(0f, 10f,
    400f, 130f);
        AWTUtilities.setWindowShape(this,
    ellipse);                              //设置窗体椭圆形状
      }
```

（3）编写窗体的鼠标单击事件处理方法，在该方法中销毁窗体对象，由于这是程序唯一的一个窗体与线程，所以销毁窗体后，程序会自动退出。关键代码如下：

```
    protected void do_this_mouseClicked(MouseEvent e) {
      dispose();
        //销毁窗体
    }
```

举一反三

根据本实例，读者可以实现以下功能。

将窗体制作成各种卡通图形。

将窗体制作成各种几何图形。

实例018 钻石形窗体

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\01\Ex01_18

实例说明

设置个性形状的窗体可以增加程序的趣味性，如定义奇特形状的登录窗体，可以让用户在进入系统之前对程序有好奇心并急于了解。本实例实现自定义钻石图形的窗体，运行程序后显示窗体界面，程序运行效果如图1.19所示。

图1.19 钻石形窗体

技术要点

Java创建钻石图形需要依靠Polygon类创建多边形来实现，所以Polygon类是本实例的关键技术。多边形可以实现任意图形，只要有合理的顶点位置即可。下面来看一下如何通过Polygon类创建多边形实现钻石图形。

Polygon 类有一个接收坐标点参数的构造方法，通过这个构造方法可以直接创建想要的图形。该方法的声明如下：

public Polygon(int[] xpoints, int[] ypoints, int npoints)

参数说明

- xpoints：顶点X 坐标的数组。
- ypoints：顶点Y 坐标的数组。
- npoints：多边形中顶点的数量。

实现过程

（1）在项目中新建窗体类DiamondFrame，去掉窗体修饰。关键代码如下：

```
public DiamondFrame() {
```

```
    setUndecorated(true);
                    //去掉窗体修饰
    //省略其他代码
  }
```

（2）编写窗体打开时的事件处理方法，在该方法中创建多边形组成的钻石图形对象，并把这个图形设置为窗体的形状。关键代码如下：

```
  protected void do_this_windowOpened(WindowEvent e) {
    int[]xPoints=
{0,50,350,400,200,0};
  //定义各顶点的X坐标
    int[]yPoints=
{200,100,100,200,400,200};
    //定义各顶点的Y坐标
    Polygon
polygon=newPolygon(xPoints,yPoints,6);
          //创建多边形
    AWTUtilities.setWindowShape(this,polygon);
                    //设置窗体形状
  }
```

举一反三

根据本实例，读者可以实现以下功能。

制作菱形窗体。

制作三角形窗体。

实例019 创建透明窗体

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\01\Ex01_19

实例说明

有些实时信息、事物提醒和各类助手程序要保持窗体置顶状态，即始终显示在所有窗体之上。这样可以保持信息的实时显示、提高助手类程序操作的方便性等。但是程序保持置顶就会遮盖窗体下方的其他窗口或者桌面上的图标，被遮盖的位置也许只包含部分信息。例如，Eclipse的代码编辑窗口，如果为程序提供窗体透明功能，窗体就不会成为屏幕上的补丁似的障碍物，反而会更受欢迎。本实例通过Java技术实现窗体透明效果，并可以控制透明度。实例运行结果如图1.20所示，透过窗体可以看见底部的Eclipse代码编辑器中的代码。

图1.20 透明窗体

技术要点

实例中使用到了 AWTUtilities 类的 setWindowOpacity()方法，它是实例中的关键技术，用于设置窗体的透明度。该方法的声明如下：

public static void setWindowOpacity (Window window, float alpha)

参数说明

● window：窗体对象。

● alpha：窗体透明度，取值范围是在0～1 之间的小数。

实现过程

（1）在项目中创建窗体类TransparencyFrame，设置窗体置顶，在窗体中添加一个滑块控件。

（2）编写滑块控件的事件处理方法，在改变滑块值时这个方法获取滑块当前值，并把它转换为百分比来改变窗体的透明度。关键代码如下：

```
protected void do_slider_stateChanged(ChangeEvent e) {
  int value=
slider.getValue();
//获取滑块当前值
  AWTUtilities.setWindowOpacity(this,value/100f);
                //使用滑块值改变窗体透明度
}
```

举一反三

根据本实例，读者可以实现以下功能。

实现桌面日历的设置透明度功能。

制作鼠标穿透窗体。

1.6 对话框

实例020 模态对话框与非模态对话框

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\01\Ex01_20

实例说明

在程序设计中，对话框的显示可以分为两种，即模态显示和非模态显示。每种显示方式对应于不同的程序应用场景。下面就通过一个程序来看一下什么是模态显示，什么是非模态显示。程序运行效果如图1.21所示，单击界面上的“模态显示对话框”与“非模态显示对话框”两个按钮可以查看不同的效果，注意它们对父窗体的限制。

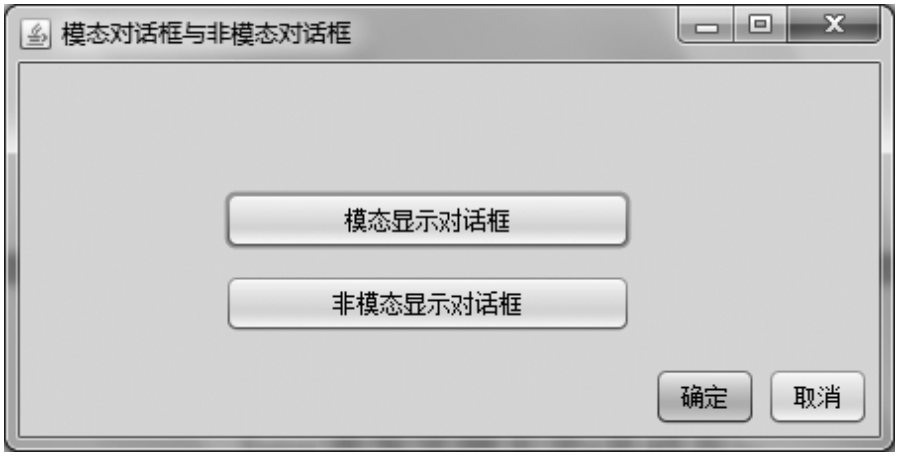


图1.21 程序运行效果

技术要点

本实例中使用到了JDialog对话框类的setModal()方法，它是实例中的关键技术，用于设置对话框的显示形态，如果以模态显示对话框，那么在对话框关闭之前，其父窗体是无法使用与获取焦点的，父

窗体的所有事件都被对话框拦截，而以非模态显示对话框则完全相反。setModal()方法的声明如下：

public void setModal(boolean modal)

参数说明

modal：是否以模态显示对话框。

实现过程

（1）在项目中创建窗体类 ModalDialog。在窗体中添加“模态显示对话框”与“非模态显示对话框”两个按钮。

（2）编写“模态显示对话框”按钮的事件处理方法。该方法将创建当前窗体的对话框，并设置对话框以模态显示。关键代码如下：

```
protected void do_button_actionPerformed(ActionEvent e) {
  JDialog dialog=new
JDialog(this);                                    //创建当
前窗体的对话框
  dialog.setModal(true);
            //设置对话框为模态
  dialog.setSize(300,200);
            //设置对话框大小
  dialog.setLocationByPlatform(true);
              //由系统平台布置窗体位置
  dialog.setTitle("模态对话
框");                                             //对话框标题
  dialog.setVisible(true);
            //显示对话框
}
```

（3）编写“非模态显示对话框”按钮的事件处理方法。该方法同样创建一个当前窗体的对话框，但是要设置对话框以非模态显示。关

键代码如下：

```
    protected void do_button_1_actionPerformed(ActionEvent e)
{
        JDialog dialog=new
    JDialog(this);                                          //创建当
    前窗体的对话框
        dialog.setModal(false);
                    //设置对话框为非模态
        dialog.setSize(300,200);
                    //设置对话框大小
        dialog.setLocationByPlatform(true);
                      //由系统平台布置窗体位置
        dialog.setTitle("非模态对话
    框");                                                   //对话框标题
        dialog.setVisible(true);
                      //显示对话框
    }
```

举一反三

根据本实例，读者可以实现以下功能。

无法改变大小的对话框。

选择商品的模态对话框。

实例021 信息提示对话框

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\01\Ex01_21

实例说明

信息提示对话框是程序开发中经常用到的功能，它可以为用户显示警告、错误、提示等信息内容，可以根据不同的信息级别选择不同类型的对话框。本实例实现了每个类型对话框的信息提示效果，程序运行效果如图1.22（a）所示，显示一个信息提示对话框的效果如图1.22（b）所示。

(a)　　(b)

图1.22 程序运行界面与信息对话框

技术要点

Java 语言中提供了 JDialog 类实现对话框效果，程序开发人员可以继承该类编写自定义的对话框。但是这相对来说比较复杂，所以Java提供了一个工具类JOptionPane来负责常见对话框的创建与使用，本实例调用了JOptionPane类的静态方法showMessageDialog()来显示信息提示对话框，该方法有多种重载格式，下面介绍本实例使用的重载方法的方法声明：

```
public static void showMessageDialog(Component
parentComponent,Object message,String title,int messageType)
throws HeadlessException
```

方法中的参数说明如表1.6所示。

表1.6 方法中的参数说明

设 置 值	描 述
parentComponent	对话框的父窗体
message	对话框显示的信息字符串
title	对话框的标题字符串
messageType	对话框类型，这个类型值确定对话框的信息图标样式

实现过程

（1）在项目中创建窗体类MessageDialog。在窗体中添加接收对话框标题和内容文本的文本框及文本域控件，添加选择信息类别的5个单选按钮和一个“弹出对话框”按钮。

（2）编写“弹出对话框”按钮的事件处理方法，在该方法中接收用户输入的对话框标题与内容文本，根据用户选择的对话框类型来显示对话框。关键代码如下：

```
protected void do_button_actionPerformed(ActionEvent e) {
  String title=
titleField.getText();
//获取标题文本
  Stringmessage=messageArea.getText();
            //获取内容文本
  String
command=bg.getSelection().getActionCommand();
    //获取选中的单选按钮
  intmessageType=
JOptionPane.INFORMATION_MESSAGE;                    //创建信
息类型
  if (command.equals("普
通"))                                              //根据用户选择
确定对话框类型
    messageType = JOptionPane.PLAIN_MESSAGE;
```

```
    if (command.equals("疑问"))
        messageType = JOptionPane.QUESTION_MESSAGE;
      if (command.equals("警告"))
        messageType = JOptionPane.WARNING_MESSAGE;
      if (command.equals("错误"))
        messageType = JOptionPane.ERROR_MESSAGE;
      //显示对话框
      JOptionPane.showMessageDialog(this, message, title,
  messageType);
    }
```

举一反三

根据本实例，读者可以实现以下功能。

带选择按钮的信息对话框。

删除成功提示框。

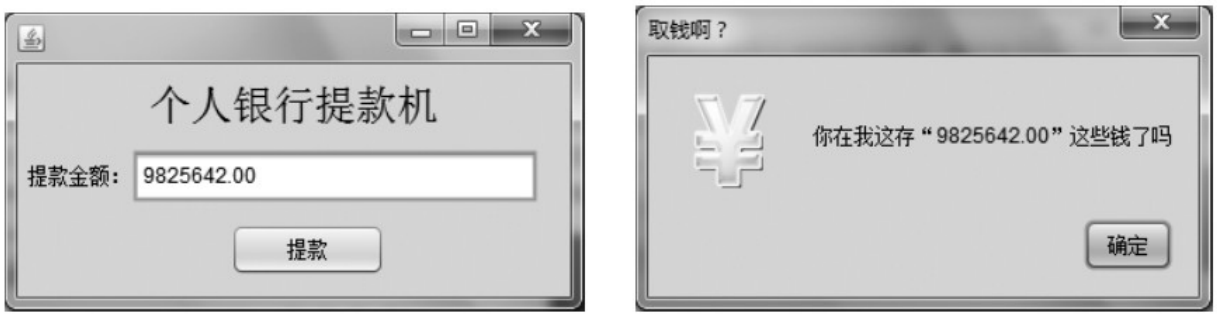
实例022 设置信息提示对话框的图标

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\01\Ex01_22

实例说明

信息提示对话框可以作为向用户说明程序当前情况与问题的工具。虽然 JOptionPane 类已经提供了多种类型的信息提示，但是这些类型仅限于普通消息、警告、错误、疑问等。在实际项目应用中，涉及的信息提示类型会很多，而且需要有针对性。本实例实现自定义对话框提示图标，以满足不同程序的不同类型的信息提示需求。程序主界面如图1.23（a）所示。当单击“提款”按钮时将弹出带有自定义图标的信息提示对话框，如图1.23（b）所示。



(a)　　(b)

图1.23 主程序界面与自定义图标的信息提示对话框

技术要点

本实例也使用到了JOptionPane类的showMessageDialog()方法，但使用的是该方法的最终重载格式，也就是参数最多的且支持自定义对话框图标的方法。下面介绍本实例使用的重载方法的方法声明：

```
public static void showMessageDialog(Component
parentComponent,Object message,String title,int
messageType,Icon icon) throws HeadlessException
```

方法中的参数说明如表1.7所示。

表1.7 方法中的参数说明

设　置　值	描　　述
parentComponent	对话框的父窗体
message	对话框显示的信息字符串
title	对话框的标题字符串
messageType	对话框类型，这个类型值确定对话框的信息图标样式
icon	对话框的图标对象

实现过程

（1）在项目中创建窗体类MessageDialogIcon。在窗体中添加文本框和“提款”按钮。

（2）编写“提款”按钮的事件处理方法，在该方法中获取用户输入，并将输入作为信息提示内容的一部分。然后获取图片资源文件的路径，并用该路径创建图标对象，最后显示对话框。关键代码如下：

```
    protected void do_button_actionPerformed(ActionEvent
arg0) {
      String text=
  textField.getText();
        //获取文本框输入
      URL
  resource=getClass().getResource("money.png");
                  //获取资源文件路径
      ImageIcon icon=new
  ImageIcon(resource);                                    //
  创建图标对象
      //显示带有自定义图标的信息提示对话框
      JOptionPane.showMessageDialog(this, "你在我这存""+ text
  +""这些钱了吗", "取钱啊？", JOptionPane.QUESTION_MESSAGE,
  icon);
    }
```

举一反三

根据本实例，读者可以开发以下程序。

实现多种图标的提示对话框。

设置警告提示对话框。

实例023 文件选择对话框指定数据库备份文件

这是一个可以启发思维的实例

实例位置：光盘\mingrisoft\01\Ex01_23

实例说明

文件对话框是桌面应用程序最常用的，其中很多数据非常重要，但在内存中是无法长久保存的，程序退出或者异常关闭，都会导致内存数据丢失。最有效的数据长久保存的办法是把数据保存到磁盘文件或者数据库中。但是最终任何形式的数据都将以磁盘文件的形式保存。本实例实现文件选择对话框为程序设置备份文件的功能。实例运行效果如图1.24所示。单击“浏览”按钮弹出的文件选择对话框如图1.25所示。

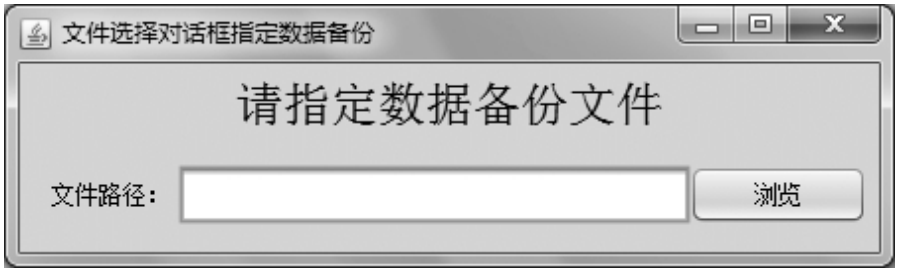


图1.24 实例运行效果

图1.25 文件选择对话框

技术要点

本实例用到了Swing的JFileChooser类，该类可以创建文件打开与保存的对话框，本实例使用的是文件打开对话框。显示文件打开对话

框的方法声明如下：

public int showOpenDialog(Component parent) throws HeadlessException

参数说明

● parent：父窗体对象。

● 返回值：用户在文件打开对话框中进行的操作对应的int 型常量。

实现过程

（1）在项目中创建窗体类FileSelectDialog。在窗体中添加文本框与“浏览”按钮。

（2）编写“浏览”按钮的事件处理方法，在该方法中创建文件选择器，然后调用其方法显示文件打开对话框，并获取用户选择的文件，再把文件路径与名称显示到文本框中。关键代码如下：

```
protected void do_button_actionPerformed(ActionEvent e) {
  JFileChooser chooser=new
JFileChooser();                          //创建文件选择器
  int option=
chooser.showOpenDialog(this);                          //显示
文件打开对话框
  if (option== JFileChooser.APPROVE_OPTION)
{                    //判断用户是否选定文件
    File file=
chooser.getSelectedFile();                              //
获取用户选择文件
    textField.setText(file.getAbsolutePath());
              //把选择的文件路径显示在文本框中
  }
```

}

举一反三

根据本实例，读者可以实现以下功能。

验证用户选择文件是否为NULL。

按指定格式选择文件。

实例024 指定打开对话框的文件类型

这是一个可以美化界面的实例

实例位置：光盘\mingrisoft\01\Ex01_24

实例说明

Java中的文件选择器的默认格式就可以显示文件的打开、保存以及各种自定义的文件选择对话框，而且对话框默认显示多种类型的文件，如果浏览某个文件夹中的必要类型的文件过多，用户就不得不从中一一挑选，找到自己要的文件类型，再确定文件名称，这会给用户带来不便。本实例将介绍如何给文件选择器添加一个过滤器，使文件选择对话框只显示需要的文件类型。实例运行结果如图1.26所示。当单击“打开图片文件”按钮时，将弹出只能选择jpg、png、gif三种格式图片文件的对话框，如图1.27所示。

技术要点

本实例为文件选择器设置了一个文件类型过滤器，这个过滤器是FileFilter抽象类的一个子类FileNameExtensionFilter。下面介绍如何创建这个过滤器以及如何为文件选择器设置这个过滤器。

□ 创建文件类型过滤器

FileNameExtensionFilter类的构造方法中可以指定文件类型的描述与支持的图片扩展名称，其中扩展名称可以是多个。下面是该类的构造方法的声明：

public FileNameExtensionFilter(String description,String... extensions)

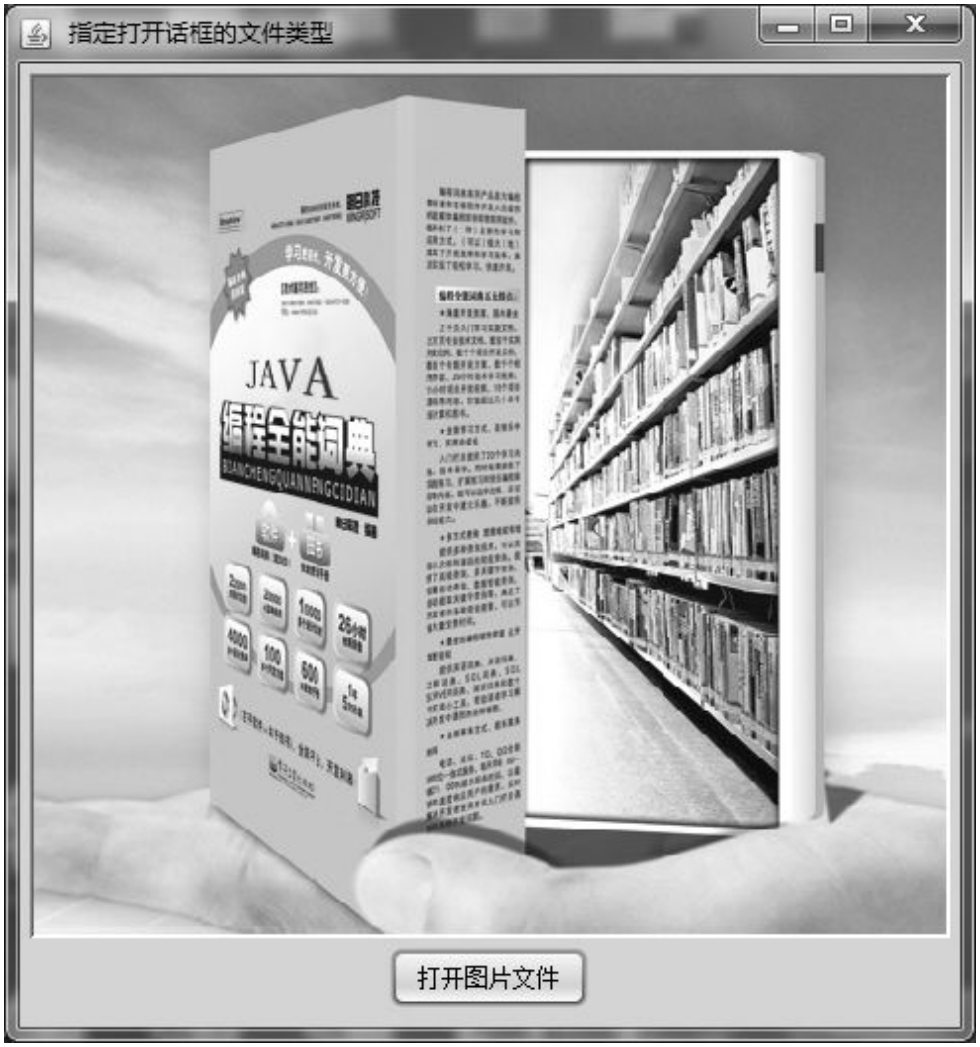


图1.26 程序主窗体

图1.27 文件选择对话框

参数说明

● description：文件类型的描述信息，它将显示在文件选择对话框的“文件类型”下拉列表框中。

● extensions：文件支持的扩展名称，可以是多个字符串参数，它们使用逗号分割。

□ 设置文件选择器的过滤器

JFileChooser文件选择器可以设置文件过滤器，这个过滤器可以是FileFilter抽象类的各种实现类。本实例使用了一个文件类型过滤器，这是Java提供的实现类。下面介绍文件选择器如何设置这个过滤器。

设置文件选择器的过滤器要通过 JFileChooser 类的setFileFilter()方法实现，该方法的声明如下：

public void setFileFilter(FileFilter filter)

参数说明

filter：FileFilter抽象类的各种实现类的对象。

实现过程

（1）在项目中创建窗体类CustomSelectFileType。在窗体中添加背景面板与“打开图片文件”按钮。

（2）编写“打开图片文件”按钮的事件处理方法，在该方法中创建文件选择器，创建FileNameExtensionFilter过滤器的实例对象，这个过滤器只接收jpg、gif和png类型的图片，然后调用文件选择器的方法显示文件打开对话框，并获取用户选择的文件，最后把图片文件内容显示在界面中。关键代码如下：

```
protected void do_button_actionPerformed(ActionEvent e) {
  JFileChooser chooser=new
JFileChooser();                                    //创建文件
选择器
```

```
    FileNameExtensionFilter
filter=newFileNameExtensionFilter("图片文件","jpg", "gif",
"png", "jpeg");                                          //创
建文件类型过滤器
    chooser.setFileFilter(filter);
                          //设置选择器的过滤器
    int option=
chooser.showOpenDialog(this);
    //显示打开对话框
    if(option==JFileChooser.APPROVE_OPTION){
      File file=
  chooser.getSelectedFile();
          //获取用户选择文件
      try {
        //加载图片文件
        ImageIcon image=new
    ImageIcon(file.toURI().toURL());
        backgroundPanel.setImage(image.getImage());
                      //显示图片文件
      } catch (MalformedURLException e1) {
        e1.printStackTrace();
      }
    }
  }
```

举一反三

根据本实例，读者可以实现以下功能。

打开其他类型的文件。

未输入文件名提示对话框。

实例025 文件的保存对话框

这是一个可以美化界面的实例

实例位置：光盘\mingrisoft\01\Ex01_25

实例说明

文件选择对话框包括文件的打开与保存和自定义几种类别。其中文件保存对话框常用于各类编辑器模块中，如系统自带的记事本程序的文件保存对话框、画图程序的文件保存对话框以及Photoshop程序的文件保存对话框等。本实例将通过Java代码实现文件保存对话框的显示，读者可以把它应用到自己的项目中。实例运行效果如图1.28所示。在其中输入编辑文本，然后选择“文件”/“保存”命令，弹出“保存”对话框，如图1.29所示。

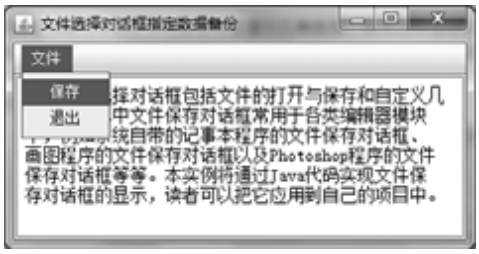


图1.28 实例运行效果

图1.29 文件保存对话框

技术要点

本实例同样使用 JFileChooser 类的方法来打开文件对话框，但本实例打开的是文件保存对话框而不是文件打开对话框，请注意对话框中的标题与按钮的名称。实例中用到的显示文件保存对话框的方法声明如下：

public int showSaveDialog(Component parent)throws HeadlessException

参数说明

● parent：父窗体对象。

● 返回值：用户在文件打开对话框中进行的操作对应的int 型常量。

实现过程

（1）在项目中创建窗体类FileSaveDialog。在窗体中添加文本域与菜单栏，然后在菜单栏中添加“保存”与“退出”菜单项。

（2）编写“保存”菜单项的事件处理方法，在该方法中创建文件选择器，然后调用其方法显示文件打开对话框，并获取用户选择的文件，然后把文本域中的文本保存到用户选择的文件中。关键代码如下：

```
protected void do_menuItem_actionPerformed(ActionEvent e)
{
    String text=
  textArea.getText();                                    //
  获取用户输入
    if (text.isEmpty())
  {                                                      //过滤空文
  本的保存操作
      JOptionPane.showMessageDialog(this, "没有需要保存的文
    本");
```

```
      return;
    }
    JFileChooser chooser=new
JFileChooser();                                    //创建文件选择
器
    int option=
chooser.showSaveDialog(this);
//打开文件保存对话框
    if (option== JFileChooser.APPROVE_OPTION)
{                                  //处理文件保存操作
      File file=
  chooser.getSelectedFile();
    //获取用户选择的文件
      try {
        FileOutputStream
    fout=newFileOutputStream(file);                    //创建该
    文件的输出流
        fout.write(text.getBytes());
                  //把文本保存到文件
      } catch (IOException e1) {
        e1.printStackTrace();
      }
    }
  }
```

举一反三

根据本实例，读者可以实现以下功能。

调用系统的保存路径对话框。

退出前提示保存对话框。

实例026 为保存对话框设置默认文件名

这是一个可以美化界面的实例

实例位置：光盘\mingrisoft\01\Ex01_26

实例说明

文件保存对话框常用于各类编辑器，在一些编辑软件中常进行电子文档的编写，要把这些文档保存下来就要使用文件。随着电脑的普及、计算机编程的日益成熟，人们对应用程序的要求也会越来越高，不可能要求用户自己去查询文件夹路径，再根据这个路径和要保存的文件名定义出资源文件的字符串。这在早期的DOS操作系统时代是有可能的，但目前Windows操作系统以及GUI应用程序能够为用户提供更便利的操作。文件保存对话框可以让用户在窗体中浏览文件夹的同时确认保存文件的路径，而本实例将在程序中动态指定保存的文件名称，这样就可以省略用户输入文件名的步骤。运行实例，选择“文件”/“新建”命令，将弹出输入新建文档名称的对话框，如图1.30所示，输入文档名称后单击“确定”按钮。对文档内容进行编辑后，单击“保存”按钮将弹出图1.31所示的“保存”对话框，在该对话框中已经指定了文件名称，用户可以修改或直接使用这个名称。

图1.30 程序运行效果

图1.31 “保存”对话框

技术要点

本实例使用 JFileChooser 类的方法来打开文件保存对话框，但是本实例在“保存”对话框中已经指定了文件名称。这是通过设置文件选择器的selectedFile属性实现的，设置该属性的方法声明如下：

```
public void setSelectedFile(File file)
```

参数说明

file：指定选中文件。

实现过程

（1）在项目中创建窗体类CurstomNameSave。在窗体中添加菜单栏、文本域控件与“保存”按钮。

（2）编写“新建”菜单项的事件处理方法，在该方法中接收用户输入的新文档名称，然后显示在标签控件中，并激活文本域为可用状态。关键代码如下：

```
protected void do_menuItem_actionPerformed(ActionEvent e)
{
  //接收用户输入
  String string = JOptionPane.showInputDialog("请输入新建
文档名称");
  if (string==null)
    return;
  label.setText(string);
       //用标签控件显示用户输入的稳定名称
  textArea.setEnabled(true);
       //激活文本域控件
}
```

（3）编写“保存”按钮的事件处理方法，在该方法中创建文件选择器，然后把用户输入的文档名称创建成文件对象，并设置为文件选择器的selectedFile属性，最后显示文件保存对话框。关键代码如下：

```
protected void do_button_actionPerformed(ActionEvent e) {
  String text=
label.getText();                                    //获取
```

标签控件保存的文档名称

```
    JFileChooser chooser=new
JFileChooser();                              //创建文件选择器
    File file=newFile(text+
".txt");                                      //用文档名称创建文
件对象
    chooser.setSelectedFile(file);
            //设置文件选择器的选择文件
    chooser.showSaveDialog(this);
        //显示保存对话框
    File selectedFile = chooser.getSelectedFile();
    JOptionPane.showMessageDialog(this, "文件保存路径:
\n"+selectedFile);
  }
```

举一反三

根据本实例，读者可以开发以下程序。

设置默认保存路径。

未输入文件名则用默认文件名。

实例027 支持图片预览的文件选择对话框

这是一个可以提高分析能力的实例

实例位置:光盘\mingrisoft\01\Ex01_27

实例说明

Windows系统中，在选择文件时，通常只能以列表的形式显示要选择的文件名，本实例将在选择图片文件后，在文件选择对话框的右侧

显示所选图片的缩略图，从而使用户确认选择的是不是自己需要的图片。实例运行结果如图1.32所示。

技术要点

实例中的关键技术在于设置文件选择器的 Accessory 控件属性，这个属性可以把一个控件作为文件选择器的辅助控件，本实例利用这个辅助控件显示了当前被选中图片文件的预览。设置Accessory控件属性的方法声明如下：

public void setAccessory(JComponent newAccessory)

参数说明

newAccessory：作为文件选择器的辅助控件。

图1.32 支持图片预览的文件选择对话框

实现过程

（1）在项目中创建窗体类 PicPreviewFileSelectDialog。把文件选择器作为窗体的控件进行添加，并设置过滤器与图片预览控件。

关键代码如下：

```
    JFileChooser fileChooser=new
JFileChooser();                                    //创建文件选择器
    contentPane.add(fileChooser,BorderLayout.CENTER);
                    //添加到窗体
    paint=newPaintPanel();
              //创建图片预览面板
    paint.setBorder(newBevelBorder(BevelBorder.LOWERED,null,n
ull,null,null));
          //设置面板的边框
    paint.setPreferredSize(newDimension(150,300));
                      //设置预览面板的大小
    fileChooser.setAccessory(paint);
                    //把面板设置为文件选择器控件
    //添加选择器的属性事件监听器
    fileChooser.addPropertyChangeListener(new
PropertyChangeListener() {
      public void propertyChange(PropertyChangeEvent arg0) {
        do_this_propertyChange(arg0);
      }
    });
    //设置文件选择器的过滤器
    fileChooser.setFileFilter(new FileNameExtensionFilter("图
片文件", "jpg","png", "gif"));
```

（2）编写文件选择器的属性改变事件处理方法。当改变选定文件时，这个方法会把图片文件加载到程序中，并设置图片预览面板的属性进行显示。关键代码如下：

```
    protected void do_this_propertyChange(PropertyChangeEvent
e) {
      //处理改变选定文件的属性事件
      if (JFileChooser.SELECTED_FILE_CHANGED_PROPERTY ==
  e.getPropertyName()) {
        Filepicfile= (File)
    e.getNewValue();                                    //获
    取选定的文件
        if (picfile!=null&&picfile.isFile()) {
          try {
            //从文件加载图片
            Image
        image=getToolkit().getImage(picfile.toURI().toURL());
            paint.setImage(image);
                      //设置预览面板的图片
            paint.repaint();
                      //刷新预览面板的界面
          } catch (MalformedURLException e1) {
            e1.printStackTrace();
          }
        }
      }
    }
```

（3）继承JPanel编写图片预览面板PaintPanel类，重写paintComponent()方法，在该方法中把图片对象绘制到面板上。关键代码如下：

```
    protected void paintComponent(Graphics g)
{                                          //重写绘制组件外观
      if (image != null) {
        g.drawImage(image,0,0,getWidth(),getHeight(),
    this);                          //绘制图片与组件大小相同
      }
      super.paintComponent(g);
              //执行超类方法
    }
```

举一反三

根据本实例，读者可以开发以下程序。

控制文件选择对话框中可以进行多选。

支持文件预览的文件选择对话框。

实例028 颜色选择对话框

这是一个可以启发思维的实例

实例位置：光盘\mingrisoft\01\Ex01_28

实例说明

颜色也是系统资源之一，它和文件同样重要。在设置颜色值时，不像文件路径那样可以通过字符串来表示，颜色值大多使用对话框进行选择，总的来说，颜色选择对话框就是让用户通过视觉来确定颜色值，而不是通过文本来确定。本实例实现了颜色选择框的应用，程序运行效果如图 1.33 所示。当单击任意一个“选择”按钮时都会弹出“选择颜色”对话框，如图 1.34所示。

图1.33 程序运行效果

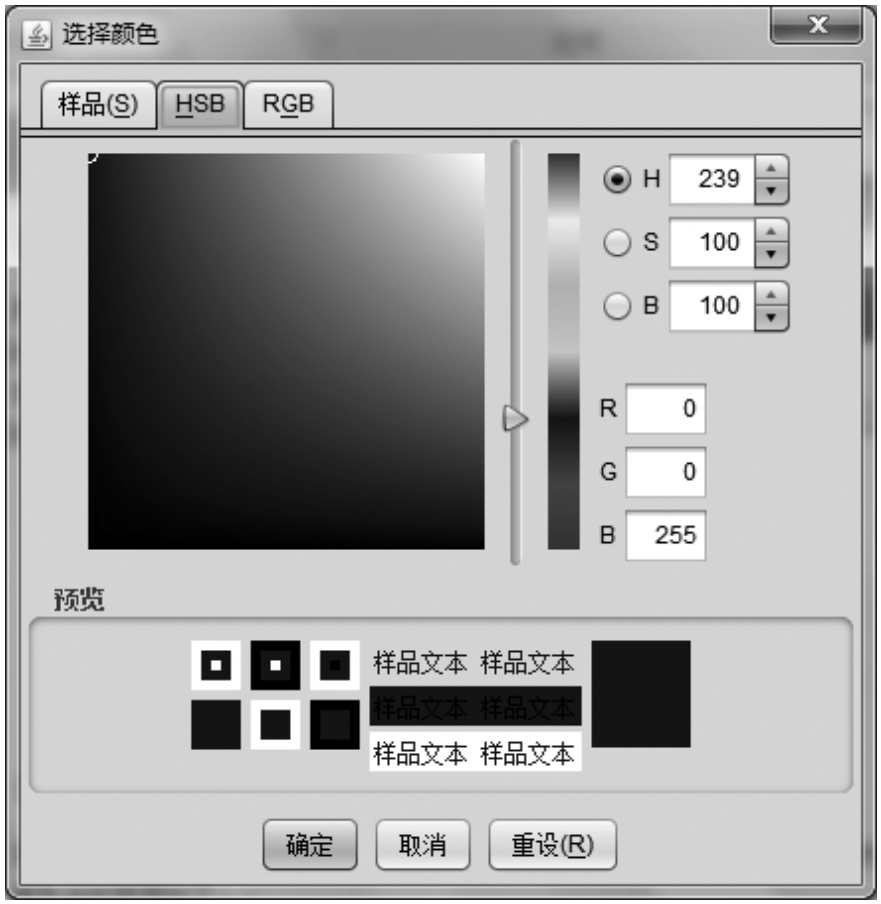


图1.34 “选择颜色”对话框

技术要点

本实例中使用到了JColorChooser类的showDialog()方法，它是实例中的关键技术，用于显示颜色选择对话框，并返回用户选择的颜色值对象。该方法的声明如下：

public static Color showDialog(Component component,String title,Color initialColor) throws HeadlessException

参数说明

- component：对话框的父级（上级）控件或窗体。
- title：对话框的标题。
- initialColor：对话框初始颜色对象。

实现过程

（1）在项目中创建窗体类 ColorChooser。在窗体中创建多个标签控件与多个“选择”按钮控件。

（2）编写“选择”按钮的事件处理方法。在方法体中调用setColor()方法为窗体上的标签指定背景颜色值。关键代码如下：

```
protected void do_button1_actionPerformed(ActionEvent e)
{
  setColor(label1);
        //指定标签的颜色设置
}
```

（3）编写setColor()方法。在方法体中，首先获取标签控件原来的颜色，然后用这个颜色作为默认值打开颜色选择对话框，最后把用户在对话框中选择的颜色值设置为标签控件的背景色。关键代码如下：

```
private void setColor(JLabel label) {
  Color color=
label.getBackground();                              //获取
原来的颜色对象
  //显示颜色选择对话框
  Color newColor= JColorChooser.showDialog(this, "选择颜
色", color);
  label.setBackground(newColor);
        //把获取的颜色设置为标签的背景色
```

}

举一反三

根据本实例，读者可以实现以下程序。

限制可选颜色。

调整颜色的明亮度。

实例029 信息输入对话框

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\01\Ex01_30

实例说明

GUI应用程序是与客户交互最灵活的一种应用程序，普通程序窗体中的控件能够与用户完成大部分交互，在特殊情况下还可以弹出各类对话框与用户进行信息交互，本实例利用输入对话框，接收用户输入添加联系人的姓名。实例运行结果如图1.35所示。利用对话框将极大地扩展程序的交互能力。

图1.35 “输入”对话框

技术要点

本实例中使用到了 JOptionPane 类的静态方法实现“输入”对话框的创建与显示，该方法的声明如下：

public static String showInputDialog(Object message,Object initialSelectionValue)

参数说明

- message：在对话框中的提示信息。
- initialSelectionValue：对话框的初始值。

实现过程

（1）在项目中创建窗体类InfoInputDialog。在窗体中添加列表控件、文本域控件和“添加”、“删除”两个按钮。

（2）编写“添加”按钮的事件处理方法。该方法将创建信息输入对话框，用于接收用户输入的姓名，并且对话框设置了默认值为“经理”。关键代码如下：

```
protected void do_button_actionPerformed(ActionEvent e) {
  //显示输入对话框
  String name = JOptionPane.showInputDialog("请输入要添加联系人的姓名：", "经理");
  DefaultListModelmodel= (DefaultListModel) list.getModel();                    //获取JList控件模型
  model.addElement(name);
                //向模型中添加新输入内容
}
```

（3）编写“删除”按钮的事件处理方法。该方法会根据列表控件的选择，来删除对应的选项。关键代码如下：

```
protected void do_button_1_actionPerformed(ActionEvent e) {
  int index= list.getSelectedIndex();
  //获取列表控件的选择项索引
  DefaultListModelmodel= (DefaultListModel) list.getModel();                    //获取列表的数据模型
```

```
    model.removeElementAt(index);
            //从模型中删除该索引指定的选项
}
```

举一反三

根据本实例，读者可以实现以下程序。

对用户输入内容的字数限制。

将用户输入的内容自动进行格式转换。

实例030 定制信息对话框

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\01\Ex01_30

实例说明

在程序设计中经常需要使用对话框显示提示信息。Java内置了信息、错误、警告、询问和普通类型的对话框，这些对话框可以满足大部分程序开发需求，但是对于特殊场景下的程序应用，需要在对话框中显示特定的信息以及特定的按钮。这就需要定制信息对话框的外观，本实例将介绍JOptionPane类的另一个更加灵活的定制对话框的使用。实例运行效果如图1.36所示。

(a) (b)

图1.36 实例运行效果

技术要点

本实例的关键技术在于对JOptionPane类的静态方法showOptionDialog()的应用，这个方法通过参数就可以定义对话框的信息、标题、图标、操作项和初始值等对话框属性。该方法的声明如下：

```
public static int showOptionDialog(Component parentComponent,Object message,String title,int optionType,int messageType,Icon icon,Object[] options,Object initialValue)throws HeadlessException
```

方法中的参数说明如表1.8所示。

表1.8 方法中的参数说明

枚 举 值	描 述
parentComponent	对话框的父窗体
message	对话框中显示的信息文本
title	对话框的标题
optionType	对话框底部显示的按钮选项

续表

枚 举 值	描 述
messageType	信息类型（这将确定对话框的图标）
icon	对话框的图标
options	对话框中操作按钮的名称（该数组决定了对话框中按钮的数量）
initialValue	对话框的初始值（与初始值同名的按钮将处于焦点状态）

实现过程

（1）在项目中创建窗体类CustomDialog。在窗体中使用标签控件显示问题文本，并添加一个“进入系统”按钮。

（2）编写“进入系统”按钮的事件处理方法。该方法将完善自定义对话框所需要的所有参数，并通过JOptionPane类的静态方法显示自定义信息的对话框。关键代码如下：

```
protected void do_button_actionPerformed(ActionEvent arg0) {
```

```
    //对话框操作项的名称
    String[]options=newString[] { "7月1日", "8月1日", "5月1日", "10月1日" };
    Stringmessage= "我国的建军节是每年的几月几日？";                    //对话框中的信息
    int num= JOptionPane.showOptionDialog(this,message, "基础考试",JOptionPane.YES_NO_OPTION,
JOptionPane.INFORMATION_MESSAGE,null,options, "8月1日");                    //显示自定义对话框
    if (options[num].equals("8月1日")) {
      JOptionPane.showMessageDialog(this, "恭喜您回答正确。");                    //回答正确的提示
    } else {
      JOptionPane.showMessageDialog(this, "回答错误，再见。");                    //回答错误的提示
    }
}
```

举一反三

根据本实例，读者可以开发以下程序。

自定义对话框的图标。

自定义对话框的信息文本。

1.7 MDI窗体的使用

实例031 创建内部子窗体

这是一个可以提高分析能力的实例

实例位置：光盘\mingrisoft\01\Ex01_31

实例说明

MDI（Multiple-Document Interface）窗体即多文档窗体，它主要用于同时显示多个文档，每个文档显示在各自的窗口中。MDI窗体的应用非常广泛，例如，如果某公司的库存管理系统需要实现自动化，则需要使用窗体来输入客户和货品的数据、发出订单以及跟踪订单等，而且这些窗体必须链接或者从属于一个窗体界面，并且必须能够同时处理多个文件，这时就需要建立MDI窗体来满足这些需求。本实例将带领读者一起来学习如何将一个窗体设置为父窗体，实例运行效果如图1.37所示。

图1.37 设置窗体为父窗体

技术要点

本实例主要用到了JDesktopPane桌面面板，之所以称为桌面面板，是因为它可以像系统桌面一样，包含很多内部的子窗体，本实例创建了多个 JInternalFrame 内部子窗体的实例，并将其添加到桌面面板中。下面分别介绍如何创建内部子窗体和如何添加内部窗体到桌面面板中。

□ 创建内部子窗体

内部子窗体只能应用于桌面面板中，它是 JInternalFrame 类的实例对象，创建该窗体时可以初始化窗体的标题名称、是否显示窗体控制按钮等。创建内部窗体完整的构造方法声明如下：

public JInternalFrame(String title,boolean resizable,boolean closable,boolean maximizable,boolean iconifiable)

方法中的参数说明如表1.9所示。

表1.9 方法中的参数说明

枚 举 值	描 述
title	内部窗体的标题名
resizable	是否允许调整内部窗体的大小
closable	是否显示内部窗体的关闭控制按钮
maximizable	是否显示内部窗体的最大化控制按钮
iconifiable	是否显示内部窗体的最小化控制按钮

□ 添加内部窗体到桌面面板

内部窗体和外部窗体一样都要设置窗体位置与大小，并通过visible属性控制窗体显示。但是在此之前需要把内部窗体添加到桌面面板中，否则无处显示。向桌面面板中添加内部窗体的方法声明如下：

public Component add(Component comp)

参数说明

comp：要添加到容器中的控件，对于桌面面板来说，就是内部窗体的实例对象，当然也可以添加其他控件。

实现过程

（1）在项目中创建窗体类 SetMDIForm。在窗体中添加一个桌面面板，然后在桌面面板中添加一个“打开”按钮。

（2）编写“打开”按钮的事件处理方法。在该方法中创建内部窗体的实例对象，并添加到桌面面板中，同时还要设置内部窗体的位置与大小。关键代码如下：

```
protected void do_button_actionPerformed(ActionEvent e) {
  frameCount++;
            //窗体计数器累加
  //创建内部子窗体
  JInternalFrame jif=new JInternalFrame("子窗体"+
frameCount, true, true,true, true);
  jif.setBounds(frameCount * 20, frameCount *
20,200,200);                          //设置窗体位置与大小
  desktopPane.add(jif);
                //添加子窗体到桌面面板
  jif.setVisible(true);
                  //显示子窗体
  desktopPane.setComponentZOrder(button,0);
                  //把按钮置顶
}
```

举一反三

根据本实例，读者可以开发以下程序。

设置内部子窗体的初始显示位置。

添加内部子窗体。

实例032 使子窗体最大化显示

这是一个可以启发思维的实例

实例位置：光盘\mingrisoft\01\Ex01_32

实例说明

在MDI窗体应用程序中打开子窗体时，一般都是以默认大小打开的，但在实际应用中，经常需要使子窗体在第一次打开时就以最大化方式打开，那么如何来实现这样的功能呢？本实例使用Java语言实现了这样的功能，实例运行效果如图1.38所示。

图1.38 使子窗体最大化显示

技术要点

本实例调用了内部子窗体的setMaximum()方法来改变窗体状态，这个方法只是尝试性地使窗体状态改变，在失败时会跑出异常，所以在调用该方法时，需要进行异常捕获，当出现不可执行的情况时做相应的处理。该方法的声明如下：

public void setMaximum(boolean b)throws PropertyVetoException

参数说明

b：当参数值为true时，控制窗体最大化，否则恢复状态。

实现过程

（1）在项目中创建一个窗体类MaxChildForm。在窗体中添加桌面面板和一个“最大化打开”按钮。

（2）编写“最大化打开”按钮的事件处理方法。在该方法中创建内部窗体，并设置其状态为最大化。关键代码如下：

```
protected void do_button_actionPerformed(ActionEvent e) {
  //创建内部子窗体
  JInternalFrame jif = new JInternalFrame("子窗体", true,
true, true, true);
  jif.setSize(200,200);
            //设置窗体大小
  desktopPane.add(jif);
            //添加子窗体到桌面面板
  jif.setVisible(true);
              //显示子窗体
  try {
    jif.setMaximum(true);
            //设置内部子窗体最大化状态
  } catch (PropertyVetoException e1) {
    e1.printStackTrace();
  }
  desktopPane.setComponentZOrder(button,0);
              //把按钮置顶
}
```

举一反三

根据本实例，读者可以开发以下程序。

获取父窗体的边界信息。

获取子窗体的最大化状态。

实例033 对子窗体进行平铺排列

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\01\Ex01_33

实例说明

在一个多文档窗体应用程序中添加完多个子窗体之后，如果一个MDI窗体中有多个子窗体同时打开，界面会显得非常混乱，而且不容易浏览，这时可以通过使用Java中的相应方法对多个子窗体进行排列，以便使其看起来更加有序。运行本实例，单击“加载子窗体”按钮，在MDI父窗体中打开新的子窗体，效果如图1.39所示；单击“窗体平铺”按钮，所有打开的子窗体都会在父窗体中平铺显示，效果如图1.40所示。

图1.39 加载子窗体

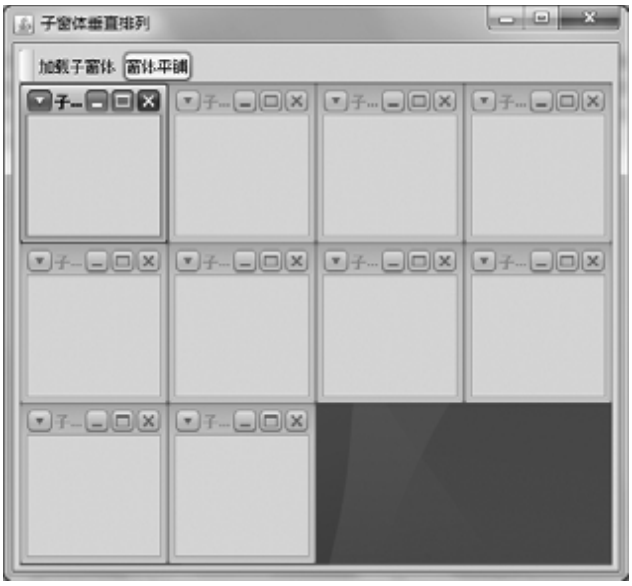


图1.40 对子窗体进行平铺排列

技术要点

本实例主要是通过容器的GridLayout布局管理器实现对子窗体的布局管理，由于子窗体对于桌面面板来说就是一个控件，所以布局管理器可以对其进行布局控制。

□ 设置布局管理器

设置桌面面板的布局管理器的方法声明如下：

```
public void setLayout(LayoutManager mgr)
```

参数说明

mgr：指定的布局管理器。

□ 创建网格布局管理器

在实现窗体的平铺过程中，本实例利用了GridLayout布局管理器的特性，自动把桌面面板中的所有控件平铺排列。创建网格布局管理器的方法声明如下：

```
public GridLayout(int rows, int cols)
```

参数说明

- rows：网格的行数。
- cols：网格的列数。

实现过程

（1）在项目中创建窗体类MDITileSort。在窗体中添加桌面面板和工具栏，并在工具栏中添加“加载子窗体”和“窗体平铺”两个按钮。

（2）编写“加载子窗体”按钮的事件处理方法。在该方法中创建子窗体，并设置随机大小与位置，然后添加到桌面面板中。关键代码如下：

```
protected void do_button_actionPerformed(ActionEvent e) {
  JInternalFrame jif=new JInternalFrame("子窗体"+
fCount++, true, true,true,
true);                                                    /
/创建内部窗体
    jif.setSize(random.nextInt(100) + 100,
random.nextInt(100) + 100);
    jif.setLocation(random.nextInt(getWidth() - 100),
random
      .nextInt(getHeight() -
100));                                                    //随机定位
内部窗体
  desktopPane.add(jif);
                  //添加内部窗体到桌面面板
  jif.setVisible(true);
                    //显示内部窗体
}
```

（3）编写“窗体平铺”按钮的事件处理方法。在该方法中为桌面面板设置网格布局管理器，然后重新布局，最后再取消布局管理器。关键代码如下：

```
    protected void do_button_1_actionPerformed(ActionEvent e)
{
      //设置桌面面板使用网格布局管理器
      desktopPane.setLayout(new GridLayout((int)
  Math.sqrt(fCount), 0));
      desktopPane.doLayout();
                  //布局桌面面板中的所有控件
      desktopPane.setLayout(null);
                        //取消桌面面板的布局管理器
    }
```

举一反三

根据本实例，读者可以开发以下程序。

对窗体进行平铺排列。

按照窗体的打开顺序排列。

实例034 禁用MDI窗体控制栏中的“最大化”按钮

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\01\Ex01_34

实例说明

在程序设计时，有时不希望窗体最大化，这样就要设置窗体控制栏中的“最大化”按钮不可用。本实例实现禁用MDI窗体控制栏中的“最大化”按钮，程序运行效果如图1.41所示。

图1.41 禁用MDI 窗体控制栏中的“最大化”按钮

技术要点

本实例主要是通过桌面面板与内部窗体的一些方法来设置当前内部窗体的最大化按钮的可用状态，下面分别介绍实例中的关键方法。

□ 获取当前选择的内部窗体

当前处于选择状态的内部窗体可以通过桌面面板的方法来获取，该方法的声明如下：

```
public JInternalFrame getSelectedFrame()
```

□ 禁用或激活窗体最大化按钮

内部窗体的最大化、最小化和关闭按钮都可以通过相应的方法来禁用或激活，下面介绍本实例禁用最大化按钮的方法，该方法的声明如下：

```
public void setMaximizable(boolean b)
```

参数说明

b：窗体“最大化”按钮的状态参数，如果该值为true，则窗体“最大化”按钮被激活；否则“最大化”按钮被禁用。

实现过程

（1）在项目中创建窗体类MDICtrlMaxButton。在窗体中添加桌面面板和工具栏，并在工具栏中添加“加载子窗体”和“禁止窗体的最大化”两个按钮。

（2）编写“加载子窗体”按钮的事件处理方法。在该方法中创建子窗体，并设置随机大小与位置，然后添加到桌面面板中。关键代码如下：

```
protected void do_button_actionPerformed(ActionEvent e) {
  final JInternalFrame jif=new JInternalFrame("子窗体"+
fCount++, true,true, true,
true);                                          //创建内部窗体
  jif.setSize(random.nextInt(100) + 100,
random.nextInt(100) + 100);
  jif.setLocation(random.nextInt(getWidth() - 100),
random
    .nextInt(getHeight() -
100));                                  //随机定位内部窗体
  desktopPane.add(jif);
    //添加内部窗体到桌面面板
  jif.setVisible(true);
      //显示内部窗体
  //为内部窗体添加事件监听器
  jif.addInternalFrameListener(new InternalFrameAdapter()
{
    @Override
    public void internalFrameActivated(InternalFrameEvent
e) {
      ctrlButton.setSelected(jif.isMaximizable());
            //改变“禁止窗体的最大化”按钮状态
    }
  });
```

```
    }
```

（3）编写“禁止窗体的最大化”按钮的事件处理方法。在该方法中获取桌面面板当前选择的窗体，并设置其窗体“最大化”按钮的可用状态。关键代码如下：

```
    protected void do_ctrlButton_itemStateChanged(ItemEvent
e) {
      JInternalFrame
  jif=desktopPane.getSelectedFrame();                    //获取
  选择的内部窗体
      if (jif != null) {
        jif.setMaximizable(!ctrlButton.isSelected());
                  //激活或禁用内部窗体“最大化”按钮
      }
    }
```

举一反三

根据本实例，读者可以实现以下功能。

创建大小固定的子窗体。

激活窗体的“最大化”按钮。

1.8 为窗体设置特效

实例035 右下角弹出信息窗体

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\01\Ex01_35

实例说明

在浏览网页时，有些网站会在网页右下角添加弹出信息，提示网站的各种即时信息。这种对用户的提醒方式，在桌面应用程序中也是常用的，如各种杀毒软件会以此方式显示拦截信息与查毒信息。本实例模拟Java编程词典软件在屏幕右下角显示产品升级信息的弹出信息窗体。实例运行结果如图1.42所示。

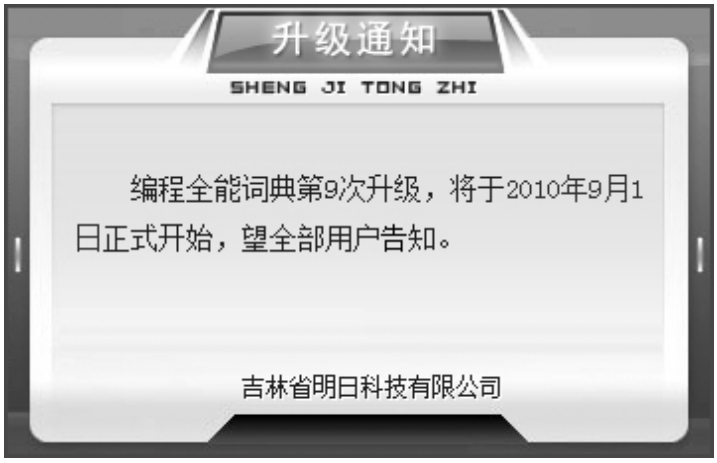


图1.42 在屏幕右下角弹出的窗体

技术要点

本实例的关键技术在于获取屏幕分辨率的大小，只有知道了屏幕分辨率，才能确定信息窗体显示的位置和移动的范围。下面介绍获取屏幕分辨率的方法。

□ 获取窗体工具包

窗体工具包是每个窗体都包含的一个对象，其中提供了多种操作的 API。获取窗体工具包的方法声明如下：

```
public Toolkit getToolkit()
```

□ 获取屏幕分辨率

窗体工具包的getScreenSize()方法用于获取当前系统屏幕的分辨率，该方法的声明如下：

```
public abstract Dimension getScreenSize()throws
HeadlessException
```

实现过程

（1）在项目中创建窗体类InfoWindow。设置窗体的大小和位置等属性，然后为窗体添加一个支持背景图片的面板控件，再为面板设置一个图片，这样就构成了弹出信息窗体的外观。关键代码如下：

```
public InfoWindow() {
  addMouseListener(newMouseAdapter()
{                                  //添加鼠标事件监听器
    @Override
    public void mousePressed(MouseEvent e) {
      do_this_mousePressed(e);
        //调用鼠标事件处理方法
    }
  });
  setBounds(100,100,359,228);
      //设置窗体大小
  BGPanel
panel=newBGPanel();                                    //创建
背景面板
```

```
        //设置背景图片
        panel.setImage(Toolkit.getDefaultToolkit().getImage(Inf
    oWindow.class.getResource("/com/lzw/panel/back.jpg")));
        getContentPane().add(panel,BorderLayout.CENTER);
    }
    protected void do_this_mousePressed(MouseEvent e)
{                          //鼠标事件处理方法
        dispose();
            //鼠标单击，则销毁这个窗体
    }
```

（2）在项目中创建主窗体类InfoDemoFrame。初始化窗体的标题、大小和位置，然后在窗体中添加一个按钮控件。关键代码如下：

```
    public InfoDemoFrame() {
        setTitle("右下角弹出信息窗
    体");                                          //设置窗体标题
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100,100,337,190);
                //窗体大小
        contentPane=new
    JPanel();                                          //创建内
    容面板
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);
                    //取消布局管理器
        JButton button=new JButton("获取即时信
    息");                          //创建按钮
```

```
    button.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        do_button_actionPerformed(e);
              //调用按钮事件处理方法
      }
    });
    button.setBounds(97, 59, 122, 30);
    contentPane.add(button);
  }
```

（3）编写按钮控件的事件处理方法。在该方法中创建Timer控件，实现动态调整信息窗体位置的渐变控制。关键代码如下：

```
  protected void do_button_actionPerformed(ActionEvent e) {
    //创建Timer控件
    timer = new Timer(1, new ActionListener() {
      @Override
      public void actionPerformed(ActionEvent e) {
        location.y -
    =1;                                          //提升信息窗体
    垂直坐标
        //在信息窗体显示而且没有达到上升位置之前持续移动窗
    体
        if (window.isShowing()&& location.y>
    screenSize.height -windowSize.height)
          window.setLocation(location);
        else
    {                                                //窗体未
    显示或超出移动范围时停止
```

```
            Timer source = (Timer) e.getSource();
            source.stop();
        }
      }
    });
    screenSize=getToolkit().getScreenSize();
            //获取屏幕大小
    window.setVisible(true);
        //显示信息窗体
    window.setAlwaysOnTop(true);
        //把信息窗体置顶
    windowSize=window.getSize();
        //获取信息窗体大小
    location=newPoint();
      //创建位置对象
    location.x= screenSize.width -
windowSize.width;                          //初始化窗体位置
    location.y = screenSize.height;
    timer.start();
        //启动Timer控件
  }
```

举一反三

根据本实例，读者可以实现以下功能。

在屏幕下弹出透明的信息窗体。

在信息窗体上添加关闭按钮。

实例036 淡入淡出的窗体

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\01\Ex01_36

实例说明

在浏览图片时由于图片过多，当前图片与下一张图片之间瞬间切换显示很容易造成视觉疲劳。很多智能的图片浏览软件添加了图片之间淡入淡出的切换效果，有效地缓解和防止了视觉疲劳。而本实例利用这个特性为窗体实现了同样的效果，实例运行过程如图1.43所示。

技术要点

本实例的关键技术在于设置窗体的透明度，然后通过不断改变这个值，来实现淡入淡出的效果。设置窗体的透明度的方法声明如下：

public static void setWindowOpacity (Window window, float alpha)

参数说明

- window：窗体对象。
- alpha：窗体透明度，取值范围是在0～1 之间的小数。

图1.43 淡入淡出过程中的窗体（左）与完全激活的窗体（右）

实现过程

（1）在项目中创建窗体类ShadeFrame。设置窗体的标题、大小和位置等属性。

（2）编写窗体激活事件处理方法。在这个方法中初始化窗体为完全透明状态，然后利用Timer控件主键改变窗体透明效果，使窗体转变

为完全不透明。关键代码如下：

```
protected void do_this_windowActivated(WindowEvent e) {
  AWTUtilities.setWindowOpacity(this,0f);
              //初始化窗体为完全透明
  ActionListener listener = new ActionListener() {
    float
alpha=0;                                                    /
/创建透明度控制变量
    @Override
    public void actionPerformed(ActionEvent e) {
      if (alpha<0.9)
  {                                                  //如果不透明
  度没有达到100%
        //不断累加透明度控制变量
        AWTUtilities
          .setWindowOpacity(ShadeFrame.this, alpha +=
      0.1);
      } else {
        //如果控制变量累加到完全不透明
        AWTUtilities.setWindowOpacity(ShadeFrame.this,
    1);
        Timer source = (Timer) e.getSource();
        source.stop();
            //停止Timer控件
      }
    }
  };
```

```
    newTimer(50,
listener).start();                                          /
/启动Timer控件
  }
```

举一反三

根据本实例，读者可以开发以下程序。

设置窗体的透明度。

窗体切换时实现其他窗体的预览。

实例037 窗体顶层的进度条

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\01\Ex01_37

实例说明

登录窗体是所有管理软件首先展现给用户的界面，用户只有输入合法的身份信息才被允许进入管理系统的主界面。由于管理系统大多与数据库相连，而且启动时可能要加载很多数据，导致登录界面消失后，很长时间才出现主窗体界面。为缓解这种现象，本实例在登录面板上显示进度条提示用户正在登录，避免用户以为程序运行出现问题。实例运行效果如图1.44所示。

图1.44 实例运行效果

技术要点

本实例的关键技术在于GlassPane面板的应用，该面板是每个JFrame窗体都包含的一个隐藏的窗体，它位于所有控件之上。默认情况下该面板是隐藏的，也就是Visible为false。可以通过设置GlassPane属性来设置窗体的玻璃面板。该方法的声明如下：

```
public void setGlassPane(Component glassPane)
```

参数说明

glassPane：窗体的玻璃面板。

实现过程

（1）在项目中创建面板类ProgressPanel。初始化面板并为其添加一个滚动条，重写paint()方法，在方法中绘制半透明的面板。关键代码如下：

```
public void paint(Graphics g) {
  Graphics2Dg2= (Graphics2D)
g.create();                                    //转换为2D绘图上下文
    g2.setComposite(AlphaComposite.SrcOver.derive(0.5f));
                    //设置透明合成规则
    g2.setPaint(Color.GREEN);
            //使用绿色前景色
    g2.fillRect(0, 0, getWidth(),
getHeight());                                  //绘制半透明矩形
    g2.dispose();
    super.paint(g);
            //执行超类绘图方法
```

```
}
```

（2）编写主窗体类LoginFrame。在窗体中添加自定义的ProgressPanel面板，并设置该面板为主窗体的GlassPane玻璃面板。关键代码如下：

```
//创建登录进度面板
panel = new ProgressPanel();
//把登录进度面板设置为窗体顶层
setGlassPane(panel);
```

（3）在窗体类中添加登录信息需要的各种控件，然后为“登录”按钮添加事件监听处理方法，在该方法中显示GlassPane登录面板。关键代码如下：

```
private final class LoginActionListener implements
ActionListener {
  public void actionPerformed(ActionEvent e) {
    //显示窗体的登录进度面板
    getGlassPane().setVisible(true);
  }
}
```

举一反三

根据本实例，读者可以开发以下程序。

带有动画的进度条。

用百分比显示加载的进度。

实例038 设置窗体的鼠标光标

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\01\Ex01_38

实例说明

鼠标是计算机操作中的主要设备之一，它在计算机屏幕中以光标图形来显示其位置和工作状态，如以箭头代表鼠标指针。当程序执行某项费时操作时，鼠标以忙碌的沙漏来通知用户暂时不可操作，但有些特殊程序需要更加复杂的鼠标光标外观。本实例演示采用设计的图片来实现鼠标光标的外观图形，实例运行效果如图1.45所示。

图1.45 实例运行效果

技术要点

本实例的关键技术在于图片资源的获取与鼠标光标对象的创建。但这些都需要利用窗体工具包来实现，所以关键技术还包括获取窗体的工具包对象。下面分别进行介绍。

□ 获取窗体工具包

获取窗体工具包的方法声明如下：

```
public Toolkit getToolkit()
```

□ 获取图片资源

获取图片资源的方法声明如下：

```
public abstract Image getImage(URL url)
```

参数说明

url：图片资源的路径。

□ 创建鼠标光标对象

创建鼠标光标对象的代码如下：

public Cursor createCustomCursor(Image cursor, Point hotSpot, String name) throws IndexOutOfBoundsException,HeadlessException

参数说明

- cursor：激活光标时要显示的图像。
- hotSpot：指定图像上的鼠标热点坐标。
- name：光标的本地化描述，用于 Java Accessibility。

实现过程

在项目中创建窗体类MouseCursorFrame。设置窗体的标题、大小和位置等属性。在构造方法中获取窗体工具包对象，然后通过该对象获取图片资源，并以此创建鼠标光标对象，最后设置该窗体。关键代码如下：

```
public MouseCursorFrame() {
  setTitle("设置窗体的鼠标光
标");                                          //设置窗体标
题
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(100,100,318,205);
                //设置窗体位置
    contentPane=new
JPanel();                                          //创
建内容面板
    Toolkit
toolkit=getToolkit();
        //获取窗体工具包
    //创建鼠标光标图片对象
```

```
        Image image =
    toolkit.getImage(getClass().getResource("1.png"));
        //通过图片创建光标对象
        Cursor cursor=
    toolkit.createCustomCursor(image,newPoint(0,0), "lzw");
        contentPane.setCursor(cursor);
                            //设置内容面板的鼠标光标
        contentPane.setBorder(newEmptyBorder(5,5,5,5));
                            //设置内容面板的边框
        contentPane.setLayout(newBorderLayout(0,
    0));                                    //设置内容面板的布局
        setContentPane(contentPane);
      }
```

举一反三

根据本实例，读者可以开发以下程序。

修改其他控件的鼠标光标。

修改光标显示的图像。

实例039 窗体抖动

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\01\Ex01_39

实例说明

QQ 是现在流行的网络通信工具，大部分网民都使用 QQ 进行网络聊天，该软件在聊天窗体中加入了一个窗体抖动的功能，以此来提醒聊天对方的注意。本实例模拟QQ的窗体抖动效果，在 Java 语言的窗

体中加入抖动效果，如图 1.46 所示，单击“窗体抖动”按钮，将使窗体发生抖动。

图1.46 实例运行效果

技术要点

本实例的关键技术在于窗体位置的控制。主要通过JFrame类的setLocation()方法实现，下面介绍该方法的声明格式：

```
public void setLocation(int x, int y)
```

该方法将窗体设置到新位置，通过x和y参数来指定新位置的左上角坐标。

参数说明

- x：新位置左上角的X 坐标。
- y：新位置左上角的Y 坐标。

实现过程

（1）在项目中创建窗体类ZoomFrameContent。设置窗体的标题、大小和位置等属性。

（2）在窗体中添加“窗体抖动”按钮，编写该按钮的事件处理方法，在方法体中获取当前窗体的位置，并通过双层for循环控制窗体的抖动效果。关键代码如下：

```
protected void do_button_actionPerformed(ActionEvent e) {
  int
num=15;
```

```
        //抖动次数
    Point
point=getLocation();
        //窗体位置
    for (int i=20; i>0; i--)
{                                                    //抖动大小
      for (int j = num; j > 0; j--) {
        point.y += i;
        setLocation(point);
                  //窗体向下移动
        point.x += i;
        setLocation(point);
                  //窗体向右移动
        point.y -= i;
        setLocation(point);
                  //窗体向上移动
        point.x -= i;
        setLocation(point);
                  //窗体向左移动
      }
    }
  }
```

举一反三

根据本实例，读者可以开发以下程序。

设置窗体的抖动时间。

设置窗体的抖动方向。

实例040 窗体标题显示计时器

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\01\Ex01_40

实例说明

窗体是操作系统中的程序单元，每个应用程序的每个功能都需要包含在窗体中，用户通过窗体区分不同的程序，所以窗体是记录工作时间最好的平台。本实例利用窗体的标题栏来显示窗体已经运行的时间，以秒为单位。实例运行效果如图1.47所示。

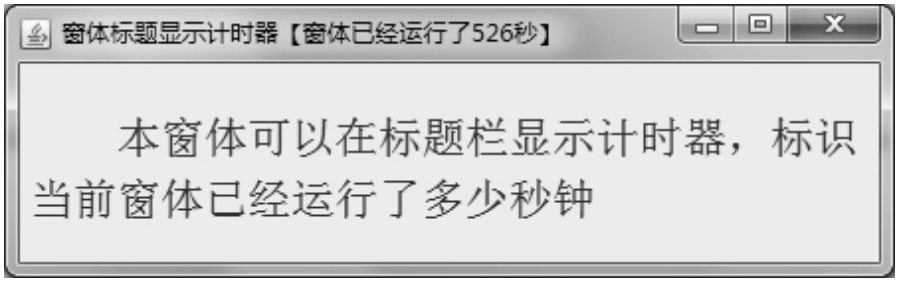


图1.47 实例运行效果

技术要点

本实例的关键技术在于 Timer 控件的应用。该控件能够在指定的时间间隔内重复执行Action。控件的ActionListener监听器将不断地捕获和处理该时间。创建一个Timer控件的构造方法的声明如下：

```
public Timer(int delay, ActionListener listener)
```

参数说明

- delay：初始延迟和动作事件间延迟的毫秒数。
- listener：初始侦听器，可以为null。

实现过程

（1）在项目中创建窗体类CalculagraphFrame。设置窗体的标题、大小和位置等属性。

（2）编写窗体打开事件的处理方法。在方法中调用System类的currentTimeMillis()方法获取当前时间的long值，然后创建Timer控

件每间隔1秒钟就获取新的当前时间的long值与原有值进行运算，把计算后的描述显示在窗体标题中。关键代码如下：

```
    protected void do_this_windowOpened(WindowEvent e)
{                                   //窗体打开事件处理方法
      sourTime=System.currentTimeMillis();
                        //记录窗体打开的初始时间
      //创建Timer控件
      Timer timer = new Timer(1000, new ActionListener() {
        String
    title=getTitle();
        //提取原始标题文本
        @Override
        public void actionPerformed(ActionEvent e) {
          //技术消耗时间
          long smTime = System.currentTimeMillis() -
      sourTime;
          //显示计时信息到标题栏
          setTitle(title +"【窗体已经运行了"+ (smTime / 1000)
      +"秒】");
        }
      });
      timer.start();
                //启动Timer控件
    }
```

举一反三

根据本实例，读者可以实现以下功能。

窗体标题栏显示系统的当前时间。

显示打开次数。

实例041 动态展开窗体

这是一个可以启发思维的实例

实例位置：光盘\mingrisoft\01\Ex01_41

实例说明

一款软件在费尽心思设计好符合人性化操作的GUI界面后，开发人员会对自己设计的程序感到信心十足。在实现业务的过程中往往会另外添加一些修饰效果，让程序更加吸引人。本实例就在登录界面中实现了窗体动态展开的效果，也就是说窗体刚出现时是一个竖条形状，然后逐渐拉伸为正常大小。实例运行效果如图1.48所示。

图1.48 窗体正在展开（左）与展开后界面（右）

技术要点

本实例的关键技术在于线程的休眠，如果实例没有设置线程休眠的时间，线程就会不停地工作，由于计算机速度非常快，用户可能根本就看不到窗体展开的动作。所以用休眠时间来延长线程改变窗体大小的时间来解决这个问题。线程休眠方法的声明如下：

```
public static void sleep(long millis)throws
InterruptedException
```

参数说明

millis：线程休眠的时间，单位为毫秒。

实现过程

（1）在项目中创建窗体类ExpandFrame。设置窗体的标题、大小和位置等属性。

（2）编写窗体打开事件的处理方法。在该方法中创建匿名的线程对象来控制窗体的拉伸效果，线程不断改变窗体的位置与大小形成展开的动作效果。关键代码如下：

```
    protected void do_this_windowOpened(WindowEvent e) {
      final int
  height=getHeight();
        //记录窗体高度
      newThread()
  {                                                          //
  创建新线程
        public void run() {
          Rectangle rec = getBounds();
          for (int i=0; i< frameWidth; i+=10)
      {                                  //循环拉伸窗体
            setBounds(rec.x - i / 2, rec.y,
          i,height);                                  //不断设置窗体
          大小与位置
              try {
                Thread.sleep(10);
                      //线程休眠10毫秒
              } catch (InterruptedException e1) {
                e1.printStackTrace();
              }
```

```
            }
        }
    }.start();
                    //启动线程
}
```

举一反三

根据本实例，读者可以开发以下程序。

打开窗体前显示动画。

用文字提示用户。

实例042 仿QQ隐藏窗体

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\01\Ex01_42

实例说明

目前大家非常熟悉网络流行的聊天软件 QQ，它有很多功能值得开发人员模仿并学习，如窗体抖动效果、窗体在屏幕边界隐藏等。本实例模拟QQ窗体隐藏的效果，如图1.49所示，当把窗体拖曳到屏幕顶端时，窗体会自动隐藏。

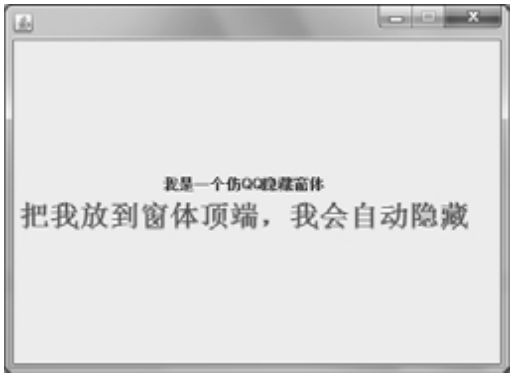


图1.49 实例运行效果

技术要点

本实例的关键技术请参见实例039。

实现过程

（1）在项目中创建窗体类QQFrame。设置窗体的标题、大小和位置等属性。

（2）编写窗体移动事件的处理方法。在方法体中判断窗体移动的位置，如果位于屏幕顶端10个像素以内，则隐藏该窗体。关键代码如下：

```
    protected void do_this_componentMoved(ComponentEvent e)
{         //窗体移动事件处理方法
      if
  (over)                                                    //如果
  鼠标在窗体中，就不做窗体隐藏操作
        return;
      Point
  point=getLocation();                                      //获取
  窗体位置
      if (point.y<10)
  {                                                  //如果窗体靠近屏幕
  顶端
        collection=
  true;                                             //确定隐藏窗体标
  识
      Dimension
  size=getSize();                                     //获取窗体大小
      setLocation(point.x, -
  size.height+5);                              //隐藏窗体
    } else {
```

```
        collection=
    false;                                          //如果窗体没有
    靠近屏幕顶端则取消隐藏标识
      }
    }
```

（3）编写窗体的鼠标进入时的事件处理方法。在该方法中判断窗体是否被隐藏在屏幕上方，然后把窗体设置到贴近屏幕顶端的位置。关键代码如下：

```
    protected void do_this_mouseEntered(MouseEvent e)
{                       //鼠标进入窗体的事件处理方法
      Point
  point=getLocation();                                    //
  获取窗体位置
      if
  (point.y>0)                                              //
  如果窗体没有被隐藏不做任何操作
        return;
      setLocation(point.x,8);
          //设置窗体显示
      over=
  true;                                                   //标识
  鼠标在窗体内部
      try {
        Thread.sleep(1000);
        //给窗体1秒钟时间让鼠标就位
      } catch (InterruptedException e1) {
        e1.printStackTrace();
```

```
    }
  }
```

（4）编写窗体的鼠标离开时的事件处理方法。在方法体中执行鼠标拖曳事件相同的处理方法。关键代码如下：

```
  protected void do_this_mouseExited(MouseEvent e)
{                       //鼠标离开窗体的事件处理方法
    if (over)
  {                                                    //如果鼠
  标标识在窗体内部
      over=
  false;                                               //取消
  鼠标位置的标识
      do_this_componentMoved(null);
        //隐藏窗体
    }
  }
```

举一反三

根据本实例，读者可以开发以下程序。

当鼠标处于窗体的隐藏位置时显示窗体。

当窗体处于屏幕边界时自动隐藏。

实例043 窗体百叶窗登场特效

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\01\Ex01_43

实例说明

百叶窗特效常用于图片浏览软件中的过渡效果，其目的是缓解视觉疲劳，避免闪屏对眼睛的刺激。本实例在窗体首次打开时也采用了这个效果来显示窗体界面。如图1.50所示，窗体打开后，界面是被蓝色矩形遮盖的，随后蓝色矩形以百叶窗的效果逐渐消失，最后会显示出原有的窗体界面。

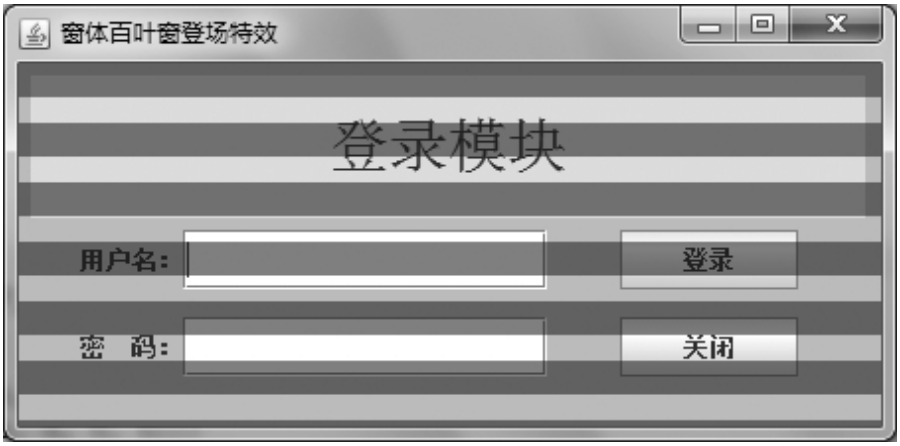


图1.50 百叶窗窗体

技术要点

本实例的关键技术请参见实例037。

实现过程

（1）在项目中创建自定义面板类 JalousiePanel。在构造方法中初始化面板为透明状态，并初始化窗体的玻璃面板，同时创建Timer控件来控制玻璃面板的显示与百叶窗效果中的参数变更。再为自定义面板添加控件事件监听器，当面板显示和调整大小事件发生时，启动Timer控件执行百叶窗特效。关键代码如下：

```
public JalousiePanel() {
  setOpaque(false);
        //面板透明
  finalComponent
oldPanel=getGlassPane();                                    //保
存原有玻璃面板
```

```
final boolean visible = oldPanel.isVisible();
setGlassPane(JalousiePanel.this);
            //把当前面板设置为窗体玻璃面板
getGlassPane().setVisible(true);
            //显示玻璃面板
//初始化Timer控件
timer = new Timer(30, new ActionListener() {
  @Override
  public void actionPerformed(ActionEvent e) {
    setGlassPane(JalousiePanel.this);
            //设置当前面板为窗体玻璃面板
    getGlassPane().setVisible(true);
              //显示玻璃面板
    if (hei-->0)
{                                                    //递减百叶
条渐变高度
      repaint();
        //重绘界面
    } else
{                                                        //如
果百叶条高度渐变小于0
      timer.stop();
        //停止Timer控件
      setGlassPane(oldPanel);
          //恢复原有玻璃面板
      hei=
  step;                                                  //初
```

```
  始化百叶条高度
      getGlassPane().setVisible(visible);
            //恢复玻璃面板显示状态
    }
  }
});
//添加控件的事件监听器
addComponentListener(new ComponentAdapter() {
  @Override
  public void componentShown(ComponentEvent e) {
    fillJalousie();
          //控件显示时调用的方法
  }
  private void fillJalousie() {
    Dimension
size=getSize();                                    //获取
窗体控件大小
    recNum= (size.height - 1) /
step+1;                                 //计算百叶条数量
    timer.start();
          //启动Timer控件
  }
  @Override
  public void componentResized(ComponentEvent e) {
    fillJalousie();
          //控件调整大小时调用的方法
  }
```

```
    });
  }
```

（2）重写自定义面板的paintComponent()方法。在该方法中绘制百叶窗中的每个条形画面，其中百叶条要实现半透明效果，所以设置绘图合成规则为半透明。关键代码如下：

```
  protected void paintComponent(Graphics g1) {
    Graphics2Dg= (Graphics2D)
g1;                                          //获取2D绘图对象
    g.setColor(Color.BLUE);
            //设置绘图前景色
    //设置绘图透明度
    g.setComposite(AlphaComposite.SrcOver.derive(0.5f));
    for (int i = 0; i < recNum; i++) {
      //绘制所有百叶条
      g.fillRect(0, i * step, getWidth(), hei);
    }
    super.paintComponent(g);
  }
```

举一反三

根据本实例，读者可以开发以下程序。

淡入淡出效果的窗体。

飞入式效果的窗体。

实例044 关闭窗体打开网址

这是一个可以提高分析能力的实例

实例位置：光盘\mingrisoft\01\Ex01_44

实例说明

桌面应用程序可以脱离网络，不需要像Web程序那样依赖浏览器，但是桌面应用程序在公司与网站宣传方面不及Web程序，所以大多桌面应用程序常调用本地浏览器来显示网站的部分页面，以此加强用户对软件与公司网站的了解。本实例在用户关闭窗体时将调用本地浏览器打开本公司的网站。实例运行结果如图1.51所示。

图1.51 程序运行界面（左）和窗体关闭时打开的网页（右）

技术要点

本实例的关键技术在于调用本地浏览器访问指定的网址。Java 6.0 在 Swing 新增的特性中包括一个Desktop控件，它可以执行本地应用程序的调用，其中包括调用本地浏览器访问指定的网址。下面介绍如何使用该类实现本实例的功能。

☐ 获取Desktop 实例

```
public static Desktop getDesktop()
```

☐ 浏览指定资源

```
public void browse(URI uri)throws IOException
```

参数说明

uri：要浏览的资源路径。

实现过程

（1）在项目中创建窗体类AddressFrame。设置窗体的标题、大小和位置等属性。

（2）编写窗体关闭事件的处理方法。在该方法中获取Desktop实例，然后通过它的browse()方法来访问指定的网址。关键代码如下：

```
protected void do_this_windowClosing(WindowEvent e)
{                              //窗体关闭事件处理方法
    Desktop
desktop=Desktop.getDesktop();
  //获取桌面程序管理器
    try {
      desktop.browse(newURI("http://www.mrbccd.com"));
                  //浏览指定网址
    } catch (IOException e1) {
      e1.printStackTrace();
    } catch (URISyntaxException e1) {
      e1.printStackTrace();
    }
  }
```

举一反三

根据本实例，读者可以开发以下程序。

Desktop浏览本地资源。

Desktop执行文件打开、编辑、打印操作。

第2章 控件应用

顶层容器的应用
输入控件的应用
选择控件的应用
菜单控件的应用
列表的应用
表格的应用
树控件的应用
JTextPane控件的应用
JEditorPane控件的应用
进度指示器的应用
微调控件
自定义控件
控件渲染
为控件添加动态效果

2.1 顶层容器的应用

实例045 设置框架容器的背景图片

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\02\Ex02_45

实例说明

普通框架容器是没有背景图片的。为了让应用程序更加美观，可以为其添加背景图片。添加背景图片的方法很多，本实例采用为层级窗格指定背景图片的方式，其优点是还可以在背景图片上增加其他控件。实例运行效果如图2.1所示。

图2.1 实例运行效果

技术要点

Swing中共有3个顶层容器，分别是JApplet、JFrame和JDialog。其他Swing控件都直接或间接包含在这几个顶层容器中。对于应用程序而言，通常使用JFrame作为其顶层容器。JFrame被分成了不同的层次以便实现不同的功能。Swing的常用方法如表2.1所示。

表2.1 Swing的常用方法

方 法 名	作 用
add(Component comp,int index)	在 index 位置增加控件 comp
getContentPane()	获得框架容器的内容窗格对象
getLayeredPane()	获得框架容器的层级窗格对象
setBounds(int x,int y,int width,int height)	设置控件的宽为 width，高为 height，左上角坐标是(x,y)
setDefaultCloseOperation(int operation)	设置框架在关闭时的动作
setLayout(LayoutManager mgr)	设置容器的布局管理器为 mgr
setLocationRelativeTo(Component c)	设置窗体与控件 c 的相对位置，如果 c 为空则居中显示
setOpaque(boolean isOpaque)	设置控件是否透明，true 为不透明
setTitle(String title)	设置框架的标题为 title
setVisible(boolean b)	设置窗体是否可见

实现过程

编写BackgroundImage类，它继承了JFrame类。利用给层级窗格增加标签的方法给框架设置背景图片，在内容窗格上增加一个按钮来测试可以在背景图片上增加其他控件。代码如下：

```
public class BackgroundImage extends JFrame {
  private static final long serialVersionUID=
-7734031908388740823L;                    //定义序列化标识
  public BackgroundImage() {
    ImageIcon background=new
ImageIcon("src/image/mingri.jpg");                    //创建
图标
    JLabel label=new
JLabel(background);                                   //利
用给定的图片创建标签
    //将标签的大小设置成图标的大小
    label.setBounds(0,0,background.getIconWidth(),backgro
und.getIconHeight());
```

```
    JPanel panel= (JPanel)
getContentPane();                              //将内容窗格
转型成面板
    panel.setOpaque(false);
            //将面板设置成透明的
    panel.setLayout(newFlowLayout());
  //将内容窗格的布局设置为流式布局
    panel.add(new JButton("编程词
典"));                                  //创建一个按钮对象作
为测试
    getLayeredPane().add(label,new
Integer(Integer.MIN_VALUE));              //给层级窗格增加
标签
    setBounds(0,0,background.getIconWidth(),background.ge
tIconHeight());
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            //设置单击关闭图标时框架为关闭
    setLocationRelativeTo(null);
              //将框架居中显示
    setTitle("框架容器的背景图
片");                              //设置框架的标题为“框架容器
的背景图片”
  }
  public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {
      public void run() {
```

```
            BackgroundImage
        image=newBackgroundImage();                    //在事件调度线
        程时运行程序
            image.setVisible(true);
                //设置框架为可见
          }
        });
      }
    }
```

举一反三

根据本实例，读者可以开发以下程序。

为容器添加背景图片。

在图片上增加其他控件。

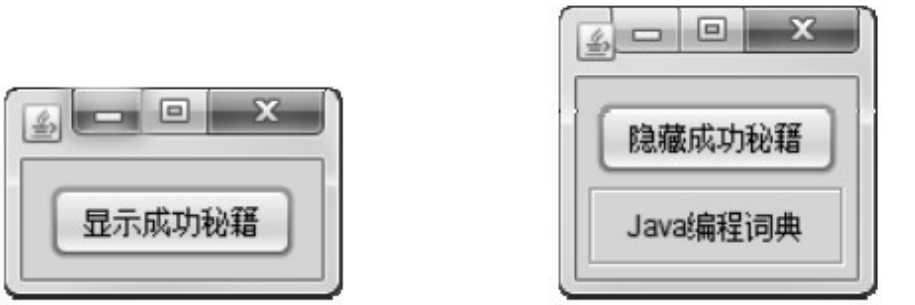
实例046 更多选项的框架容器

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\02\Ex02_46

实例说明

在用户注册网站时，除了用户名、密码必须填写外，还可以增加一些其他信息让用户选填。通常为了节约空间可以将选填项隐藏，如果用户想填写，可再显示。本实例在一个框架中实现了这个效果。实例运行效果如图2.2所示。



(a) 单击按钮前　　(b) 单击按钮后

图2.2 实例运行效果

技术要点

按钮是图形用户界面中最常见也是最简单的控件之一。在使用按钮时可以为其增加图片、设置快捷键等。为了让按钮对用户操作产生响应，通常对其增加动作监听。ActionListener是一个监听器接口，它定义了一个名为actionPerformed()的方法，用来实现对用户单击按钮的响应。该方法的声明如下：

```
void actionPerformed(ActionEvent e)
```

参数说明

e：动作事件对象。

实现过程

（1）编写MoreChoices类，该类继承了JFrame。在框架中增加一个按钮“显示成功秘籍”。

（2）本实例一共实现了两个动作监听器对象，即 more和 less，由于其代码相似，取 more进行讲解。该监听器实现了在内容窗格上增加面板控件hiddenPanel、修改按钮的文本信息、删除按钮上more的监听器并增加按钮上less的监听器的功能。代码如下：

```
ActionListener more = new ActionListener() {
  @Override
  public void actionPerformed(ActionEvent e) {
```

```
        getContentPane().add(hiddenPanel);                    /
/在内容窗格上增加面板控件hiddenPanel
        pack();                                              //重新
绘制窗体以使其刚好包含全部控件
        button.setText("隐藏成功秘籍");                        //
修改按钮的文本信息
        button.removeActionListener(more);                    /
/删除按钮上more的监听器
        button.addActionListener(less);
//增加按钮上less的监听器
    }
};
```

举一反三

根据本实例，读者可以实现以下功能。

在必填项添加红色标记。

验证必填项。

实例047 拦截事件的玻璃窗格

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\02\Ex02_47

实例说明

在软件进行比较耗时的操作时，可以使用玻璃窗格将软件界面暂时锁定，即拦截所有用户的输入事件。本实例模拟一个下载工具，当用户选择一个文件进行下载时，将暂时锁定界面并提示“正在下载”。此时用户无法再选择其他文件或者单击按钮。实例运行效果如图2.3所示。

图2.3 实例运行效果

技术要点

如果想锁定窗体，则可以使用玻璃窗格。通常需要根据需求自定义玻璃窗格的对象，然后再将其设置为框架的玻璃窗格。这可以使用setGlassPane()方法实现，该方法的声明如下：

```
public void setGlassPane(Component glassPane)
```

参数说明

glassPane：用户自定义的玻璃窗格。

实现过程

（1）编写GlassPane类，它继承了JComponent类。在其构造方法中，首先屏蔽了鼠标事件、键盘事件等。在paintComponent()方法中简单地在控件上绘制了一个红色字符串。代码如下：

```
public class GlassPane extends JComponent {
  private static final long serialVersionUID =
9060636159598343142L;
  public GlassPane() {
    addMouseListener(newMouseAdapter()
{                    //屏蔽鼠标事件
    });
    addMouseMotionListener(newMouseMotionAdapter()
{       //屏蔽鼠标拖拽事件
```

```
    });
    addKeyListener(newKeyAdapter()
{                          //屏蔽键盘事件
    });
    setFont(newFont("Default",Font.BOLD,16));
  //设置控件的字体
  }
  @Override
  protected void paintComponent(Graphics g) {
    g.setColor(Color.RED);
  //将画笔换成红色
    g.drawString("正在下
载",190,130);                          //在坐标(190,130)处绘
制字符串“正在下载”
  }
}
```

（2）编写 DownloadSoft 类，该类继承了 JFrame。在框架中，主要包括一个表格和一个按钮“开始下载”。表格用来模拟可以下载的资源。编写 do_button_actionPerformed()方法来监听单击按钮的事件，该方法用来显示玻璃窗格。代码如下：

```
protected void do_button_actionPerformed(ActionEvent e) {
  getGlassPane().setVisible(true);
              //显示玻璃窗格
}
```

举一反三

根据本实例，读者可以实现以下功能。

添加功能停止锁定。

添加功能暂时停止锁定。

实例048 简单的每日提示信息

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\02\Ex02_48

实例说明

对于一些功能比较复杂的软件，可以在软件启动时弹出一个对话框来显示一些提示信息，如软件的快捷键、软件的使用技巧、软件公司的简介等。本实例使用 JDialog 实现了一个每日提示对话框。实例运行效果如图2.4所示。

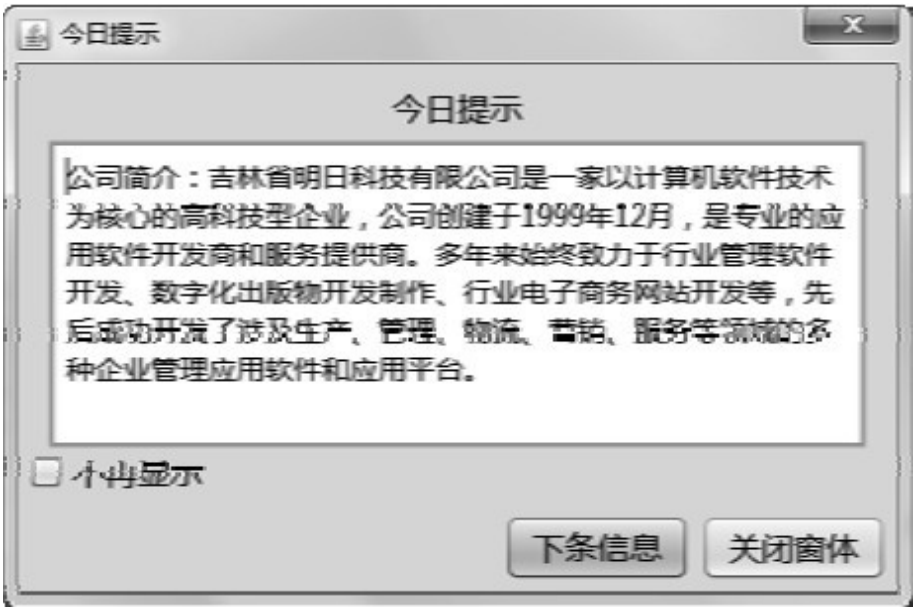


图2.4 实例运行效果

技术要点

本实例涉及的关键技术请参见实例021。

实现过程

编写TipOfDay类，该类继承了JDialog，实现了显示提示信息的功能。对话框包含一个标签、一个文本域、一个复选框和两个按钮。核心代码如下：

```
  public TipOfDay() {
    setTitle("\u4ECA\u65E5\u63D0\u793A");
                //设置对话框的标题
    setBounds(100,100,450,300);
              //设置对话框的大小和位置
    getContentPane().setLayout(newBorderLayout());
                    //设置对话框的布局为边框布局
    contentPanel.setBorder(newEmptyBorder(5,5,5,5));
                  //设置边框为空边框，宽度为5
    getContentPane().add(contentPanel,BorderLayout.CENTER);
                    //在中央增加面板contentPanel
    contentPanel.setLayout(newBorderLayout(0,
0));                                  //设置中央面板中的空白大小为0
    {
      JPanel panel=new
  JPanel();                                          //创建新的
  panel面板
      contentPanel.add(panel,BorderLayout.NORTH);
                  //在contentPanel中增加panel
      {
        JLabel label=new
    JLabel("\u4ECA\u65E5\u63D0\u793A");          //创建标签
        panel.add(label);
            //在panel中增加标签
      }
    }
    {
```

```
    JPanel panel=new
JPanel();                                          //创建新的
panel面板
    contentPanel.add(panel,BorderLayout.SOUTH);
                //在南面增加一个选择框
    panel.setLayout(new BorderLayout(0, 0));
    {
      JCheckBox checkBox=new
  JCheckBox("\u4E0D\u518D\u663E\u793A");
      panel.add(checkBox);
    }
  }
  {
    JPanel panel=new
JPanel();                                          //创建新的
panel面板
    contentPanel.add(panel,BorderLayout.WEST);
                //在西方增加一个空面板占位
  }
  {
    JPanel panel=new
JPanel();                                          //创建新的
panel面板
    contentPanel.add(panel,BorderLayout.EAST);
                  //在东方增加一个空面板占位
  }
  {
```

```
    JScrollPanescrollPane=newJScrollPane();
    contentPanel.add(scrollPane, BorderLayout.CENTER);
    {
      JTextArea textArea=new
  JTextArea();                              //利用文本域来显
  示主要的信息
      //省略文本信息代码
      scrollPane.setViewportView(textArea);
    }
  }
  {
    JPanel buttonPane=new
JPanel();                                   //创建新的
panel面板
    buttonPane.setLayout(new
FlowLayout(FlowLayout.RIGHT));
    getContentPane().add(buttonPane,BorderLayout.SOUTH);
                //增加按钮面板buttonPane
    {
      JButton okButton=new
  JButton("\u4E0B\u6761\u4FE1\u606F");
      okButton.setActionCommand("OK");
      buttonPane.add(okButton);
            //增加“下条信息”按钮
      getRootPane().setDefaultButton(okButton);
    }
    {
```

```
        JButton cancelButton=new
    JButton("\u5173\u95ED\u7A97\u4F53");
        cancelButton.setActionCommand("Cancel");
        buttonPane.add(cancelButton);
                //增加“关闭窗体”按钮
      }
    }
}
```

举一反三

根据本实例，读者可以开发以下程序。

设置提醒时间过时自动隐藏。

实现淡入淡出的提醒效果。

实例049 震动效果的提示信息

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\02\Ex02_49

实例说明

在软件的使用过程中，如果能增加一些动态效果是很有益的。例如，在QQ的2010版中就有窗体震动的效果。Java的Swing也能做成这种效果吗？答案是肯定的。本实例将实现一个震动效果的对话框。实例运行效果如图2.5所示。

图2.5 实例运行效果

技术要点

Timer类可用于在指定时间间隔触发一个或多个事件。本实例使用的方法如表2.2所示。

表2.2 Timer类的常用方法

方 法 名	作 用
Timer(int delay, ActionListener listener)	Timer 的构造方法，用于每隔 delay 毫秒触发事件 listener
start()	启动 Timer，使它开始向其侦听器发送动作事件
stop()	停止 Timer，使它停止向其侦听器发送动作事件

实现过程

编写 ShakeDialog 类，该类定义了 4 个方法：构造方法用来获得对话框对象；startShake()方法用来实现震动效果，震动时间是 1 秒钟；stopShake()方法用来关闭震动效果并将对话框恢复到原来的位置；main()方法用来进行测试。代码如下：

```
    public class ShakeDialog {
      private JDialog dialog;
      privatePoint
  start;                                          //保存对
  话框的初始位置
      private Timer shakeTimer;
      publicShakeDialog(JDialog dialog)
  {                                    //在构造方法中获得对话框对象
```

```
        this.dialog = dialog;
    }
    publicvoid startShake()
{                                          //开始震动方法
        final long
    startTime=System.currentTimeMillis();                  //获
    得程序运行的起始时间
        start=dialog.getLocation();
            //获得对话框的初始位置
        shakeTimer=newTimer(10,newActionListener()
    {                  //每隔10毫秒启动改变对话框坐标事件
          @Override
          publicvoid actionPerformed(ActionEvent e) {
            long elapsed=System.currentTimeMillis() -
        startTime;       //获得程序运行的时间
            Random
        random=newRandom(elapsed);                    //以运行时
        间为种子创建随机数对象
            int change=
        random.nextInt(50);                          //获得一个小
        于50的随机数整数
            dialog.setLocation(start.x+ change, start.y+
        change);         //随机改变坐标
            if (elapsed>=1000)
        {                                  //如果程序运行时间大于
        1秒钟则停止
              stopShake();
```

```
            }
        }
    });
    shakeTimer.start();//启动Timer
  }
  publicvoid stopShake()
{                                          //停止震动方法
    shakeTimer.stop();
      //停止Timer
    dialog.setLocation(start);
          //恢复对话框的坐标
    dialog.repaint();
      //重新绘制对话框
  }
  public static void main(String[] args)
{                              //测试方法
    JOptionPane pane=new JOptionPane("Java编程词典真好
  用！", JOptionPane.WARNING_MESSAGE);
    JDialog d=pane.createDialog(null, "震动效果的对话
  框");          //获得对话框对象
    ShakeDialog sd = new ShakeDialog(d);
    d.pack();
    //按对话框内的控件来绘制对话框
    d.setModal(false);
      //关闭模态
    d.setVisible(true);
      //设为可见
```

```
        //开始震动sd.startShake();
    }
}
```

举一反三

根据本实例，读者可以实现以下功能。

实现震动提示效果。

实现声音提示效果。

2.2 输入控件的应用

实例050 只能输入整数的文本域

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\02\Ex02_50

实例说明

在实际编程中，如果需要用户输入信息，则一般需要对其进行合法性判断。通常有前台和后台两种校验方式。Swing 中的后台校验基本是基于字符串校验的，本实例演示如何使用格式化文本域控件实现前台校验。实例运行效果如图2.6所示。

图2.6 实例运行效果

技术要点

在Java SE 1.4 版中，新增了格式化文本域类，它继承自文本域，主要用于规范输入的格式。除了常见的数字型输入和日期型输入，还可以使用该类实现更加专业的输入，如网址。在创建了格式化文本域对象之后，需要使用setValue()方法来设置使用哪种格式化的方式，该方法的声明如下：

```
public void setValue(Object value)
```

参数说明

value：要显示的当前值。

为了在文本域中显示输入的值，需要使用getValue()方法获得刚刚输入的值，该方法的声明如下：

public Object getValue()

实现过程

（1）编写类 IntegerOnlyTextField，该类继承了 JFrame。在框架中包含了一个格式化文本域控件，用于获得用户的输入；一个文本域控件，用于显示用户的输出；一个“显示结果”按钮。

（2）编写方法do_this_windowActivated()，用来监听窗体激活事件。在该方法中，设置了格式化文本域对象校验类型是整数。核心代码如下：

```
protected void do_this_windowActivated(WindowEvent e) {
  formattedTextField.setValue(new
Integer(0));                                    //设置格式化文本
域的初始值
  }
```

（3）编写方法 do_button_actionPerformed()，用来监听单击“显示结果”按钮事件。在该方法中，将格式化文本域中的值显示在文本域中。核心代码如下：

```
protected void do_button_actionPerformed(ActionEvent e) {
  textField.setText(formattedTextField.getValue().toStrin
g());                          //显示用户输入的值
  }
```

举一反三

根据本实例，读者可以实现以下功能。

对输入的内容进行校验。

输入非数字进行错误提示。

实例051 可以打开网页的标签

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\02\Ex02_51

实例说明

标签是Swing中最常用的控件之一。使用标签可以显示字符串或（和）图片。Swing默认的标签是不能对用户的操作进行响应的。本实例继承JLabel自定义了一个可以响应用户操作的标签。当用户单击网址时可以用本机默认的浏览器打开该网址。实例运行效果如图2.7所示。

图2.7 实例运行效果

技术要点

Desktop类允许Java应用程序启动本地系统默认的软件来打开文件和URI、打印文件、发送邮件等。本实例使用的方法如表2.3所示。

表2.3 Desktop类的常用方法

方 法 名	作 用
isDesktopSupported()	测试当前平台是否支持 Desktop 类
getDesktop()	获得 Desktop 类的实例
browse(URI uri)	使用系统默认的浏览器打开指定的 uri

实现过程

编写JHyperlinkLabel类，该类继承了JLabel类。本实例实现了一个可以打开指定网页的标签。代码如下：

```
public class JHyperlinkLabel extends JLabel {
```

```
    private static final long serialVersionUID
=-863116705726089148L;
    private String label;
    public JHyperlinkLabel(String label)
{                                    //在初始化时指明要显示的
字符串
      super(label);
      this.label = label;
      setForeground(Color.BLUE.darker());
                //将字符串的颜色设置成深蓝色
      setCursor(newCursor(Cursor.HAND_CURSOR));
        //将字符串上的鼠标设置成手形
      addMouseListener(newHyperlinkLabelMouseAdapter());
            //增加单击的监听事件
    }
    @Override
    protected void paintComponent(Graphics g) {
      super.paintComponent(g);
      g.setColor(getForeground());
          //将画笔的颜色设置成字符串的颜色
      Insets
  insets=getInsets();                                    //获
  得标签的边框
      int left = insets.left;
      if (getIcon() !=null)
  {                                          //如果有图片则重新
  计算左下角X坐标
```

```
        left+=getIcon().getIconWidth()+getIconTextGap();
      }
      g.drawLine(left,getHeight() - 1 - insets.bottom,
  (int) getWidth() - insets.right, getHeight() - 1 -
  insets.bottom);
      //绘制下划线
    }
    private class HyperlinkLabelMouseAdapter extends
MouseAdapter {
      @Override
      public void mouseClicked(MouseEvent e) {
        try {
          URI uri=newURI(label);                    //根
      据创建标签时使用的字符串来创建URI对象
          Desktop desktop = null;
          if (Desktop.isDesktopSupported())
      {                  //如果Desktop可用则获得其对象
            desktop = Desktop.getDesktop();
          }
          if (desktop != null) {
            desktop.browse(uri);                    //用
        浏览器打开uri
          }
        } catch (IOException ioe) {
          ioe.printStackTrace();
        } catch (URISyntaxException use) {
          use.printStackTrace();
```

```
        }
      }
    }
}
```

举一反三

根据本实例，读者可以实现以下功能。

可以打开网页的图片。

可以打开网页的文字。

实例052 密码域控件的简单应用

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\02\Ex02_52

实例说明

在用户注册网站的会员时，通常需要输入用户名、密码、邮箱等。密码需要输入两次以防止用户第一次输入错误。如果两次的密码相同则可以在数据库中创建用户，否则需要进行修改。本实例演示密码域控件的应用。实例运行效果如图2.8所示。

图2.8 实例运行效果

技术要点

密码域继承于文本域，但是修改了其显示方式。所有用户的输入并不能直接看到，而是用一些回显符号替代。典型的回显符号是“*”，用户可以根据需求自行设置。在使用密码域时，最关心的还是如何获得密码域中的文本信息，使用getPassword()方法可以实现。该方法的声明如下：

public char[] getPassword()

为了安全起见，在使用之后应该重置字符数组中的内容。

实现过程

（1）编写类JPasswordFieldTest，该类继承了JFrame。在框架中，包括了两个文本域和一个“提交”按钮。

（2）编写方法do_submitButton_actionPerformed()。该方法实现了对按钮单击事件的监听功能。如果用户输入的密码长度小于6则发出警告；如果用户两次输入的密码不一致，则发出警告；如果用户两次输入的密码一致则提示密码相同。核心代码如下：

```
protected void
do_submitButton_actionPerformed(ActionEvent e) {
      char[]password1=passwordField1.getPassword();
            //获得第一个密码域中的内容
      char[]password2=passwordField2.getPassword();
            //获得第二个密码域中的内容
      if (password1.length<6)
  {                                          //如果密码的长度小于
  6则发出警告信息
        JOptionPane.showMessageDialog(this, "密码长度小于6
    位", "",JOptionPane.WARNING_MESSAGE);
    } else if (!Arrays.equals(password1,password2))
  {                    //如果密码的长度不同则发出警告信息
```

```
        JOptionPane.showMessageDialog(this, "两次密码不同",
"",JOptionPane.WARNING_MESSAGE);
    } else {
        JOptionPane.showMessageDialog(this, "两次密码相同",
"",JOptionPane.INFORMATION_MESSAGE);
    }
}
```

举一反三

根据本实例，读者可以实现以下功能。

应用密码域来获得用户密码。

实现对用户输入的密码可见与隐藏功能。

实例053 给文本域设置背景图片

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\02\Ex02_53

实例说明

在软件的美化过程中，比较常用的方式之一是使用背景图片。在Swing中，除了可以给框架增加背景图片外，还可以给文本域设置背景图片。本实例将演示如何实现该操作。实例运行效果如图2.9所示。

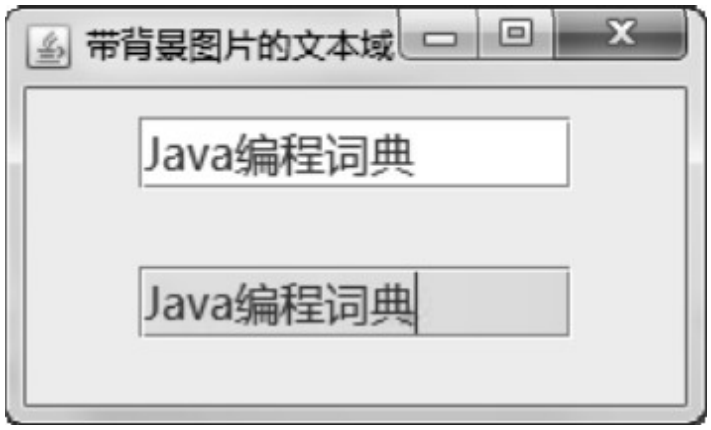


图2.9 实例运行效果

技术要点

ImageIO 类提供了很多读写图片的静态方法，还可以对图片进行简单的编码和解码。本实例使用该类中的read()方法从本地读取图片，该方法的声明如下：

public static BufferedImage read(File input)throws IOException

参数说明

input：被读入的文件。

TexturePaint 类提供一种用被指定为 BufferedImage 的纹理填充 Shape 的方式。因为BufferedImage数据由TexturePaint对象复制，所以BufferedImage对象的大小应该小一些。其构造方法声明如下：

public TexturePaint(BufferedImage txtr,Rectangle2D anchor)

参数说明

- txtr：具有用于绘制纹理的 BufferedImage 对象。
- anchor：用户空间中用于定位和复制纹理的Rectangle2D。

实现过程

（1）编写BackgroundJTextFieldTest类，该类继承了JFrame。在框架中包括了两个文本域。

（2）编写BackgroundJTextField类，该类继承了JTextField。在该类的构造方法中利用传递的 File 参数获得一个缓冲图像。以此来作为文本框的背景图片。在 paintComponent()方法中，将此背景图片绘制到文本域中。代码如下：

```
public class BackgroundJTextField extends JTextField {
  private static final long serialVersionUID =
5810044732894008630L;
```

```
  private TexturePaint paint;
  public BackgroundJTextField(File file) {
    super();
    try {
      BufferedImage image=
  ImageIO.read(file);                              //获得缓
  冲图片
      Rectangle rectangle=newRectangle(0, 0,
  image.getWidth(), image.getHeight());
      paint=newTexturePaint(image,
  rectangle);                                  //创建
  TexturePaint对象
      setOpaque(false);
                //将文本域设置成透明的
    } catch (IOException e) {
      e.printStackTrace();
    }
  }
  @Override
  protected void paintComponent(Graphics g) {
    Graphics2Dg2= (Graphics2D)
g;                                             //将g转型为
Graphics2D
    g2.setPaint(paint);
                //设置新的颜色模式
    g.fillRect(0, 0, getWidth(),
getHeight());                                  //让图片充
```

```
满整个区域
        super.paintComponent(g);
                        //调用父类的同名方法
    }
}
```

举一反三

根据本实例，读者可以实现以下功能。

为文本域设置背景色。

为文本域添加五色背景。

实例054 给文本区设置背景图片

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\02\Ex02_54

实例说明

在软件的美化过程中，比较常用的方式之一是使用背景图片。在Swing 中，除了可以给框架增加背景图片外，还可以给文本区设置背景图片。本实例将演示如何实现该操作。实例运行效果如图2.10所示。

图2.10 实例运行效果

技术要点

ImageIcon类是Icon接口的实现，它根据Image绘制Icon。在其构造方法中，提供了根据字符串路径创建图标的方式，该方法的声明如下：

public ImageIcon(String filename)

参数说明

filename：指定文件名或路径的字符串。

在获得图标之后，需要获得构成该图标的图片，使用 getImage()方法即可。该方法的声明如下：

public Image getImage()

实现过程

（1）编写BackgroundJTextAreaTest类，该类继承了JFrame。在框架中包括一个自定义文本区。

（2）编写BackgroundJTextArea类，该类继承了JTextArea。在其构造方法中利用给定的路径获得图片，并将文本区设置成透明的。在paint()方法中，将图片绘制到文本区中。代码如下：

```
public class BackgroundJTextArea extends JTextArea {
  private static final long serialVersionUID
=-41577822716327619 73L;
  private Image image;
  public BackgroundJTextArea(String path) {
    ImageIcon imageIcon=new
ImageIcon(path);                                    //获得图片
图标
    image=
imageIcon.getImage();
    //获得图片
    setOpaque(false);
              //将文本区设置成透明的
  }
  @Override
  public void paint(Graphics g) {
    g.drawImage(image,0,0,
this);                                              //绘制图片
    super.paint(g);
  }
}
```

举一反三

根据本实例，读者可以实现以下功能。

应用3种不同方式设置控件背景图片。

给文本区设置背景色。

实例055 简单的字符统计工具

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\02\Ex02_55

实例说明

在使用文本编辑软件，如Word 2007 时，会在软件界面中提示总字符等信息，方便用户掌握文档编写的进度。本实例将模拟 Word 的功能并进行增强，可以实时显示光标所在的位置和用户选择的文本所包含的字符数量。实例运行效果如图2.11所示。

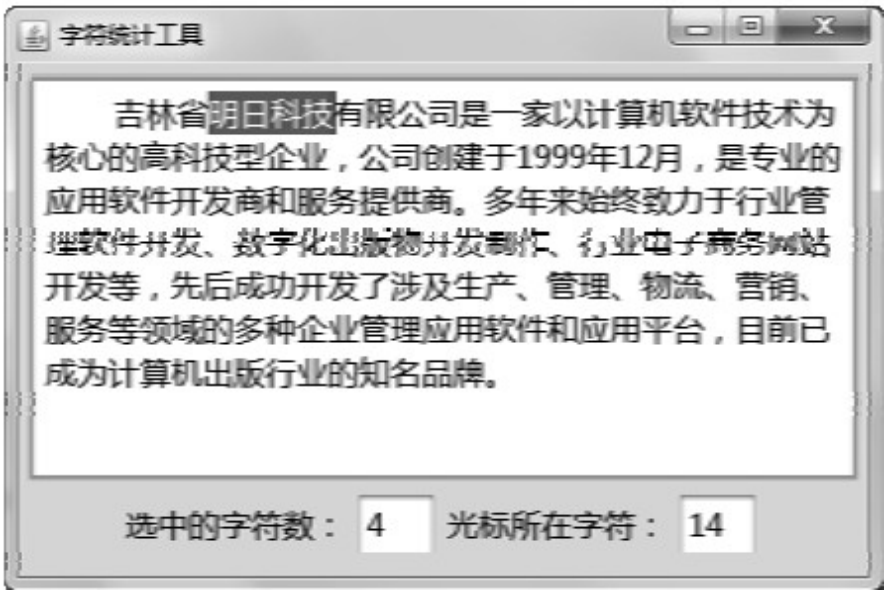


图2.11 实例运行效果

技术要点

CaretListener 接口用于侦听文本控件插入符的位置更改的侦听器。该接口定义了一个caretUpdate()方法，该方法在插入符的位置被更新时调用，其声明如下；

void caretUpdate(CaretEvent e)

参数说明

e：插入符事件。

CaretEvent 用于通知感兴趣的参与者事件源中的文本插入符已发生更改。该类定义了两个抽象方法，其声明和说明如下：

public abstract int getDot()

获得插入符的位置。

public abstract int getMark()

获得逻辑选择的另一端的位置。如果没有进行选择，则此位置将与 dot 相同。

实现过程

（1）编写CharCount类，该类继承了JFrame。在框架中包括一个文本区和两个文本域。

（2）编写方法do_textArea_caretUpdate()，用来显示光标的变化信息。该方法是由IDE自动生成的。核心代码如下：

```
protected void do_textArea_caretUpdate(CaretEvent e) {
  int dot=
e.getDot();                                    //获得光标所在的位置
  intmark=
e.getMark();                                   //获得使用鼠标选择时光标的起点位置
  textField1.setText(Math.abs(dot -mark)+
"");                    //计算用户选择的文本长度并在文本域中显示
  textField2.setText(dot+
"");                                    //显示光标所在的位置
  }
```

举一反三

根据本实例，读者可以实现以下功能。

当用户光标的长度超过某个特定数值时弹出一个对话框对用户进行提示。

增加一个标签来显示还有多少字符可以输入。

2.3 选择控件的应用

实例056 能预览图片的复选框

这是一个可以用来提高基础性能的实例

实例位置：光盘\mingrisoft\02\Ex02_56

实例说明

使用文本域、文本区等控件与用户交互时，很难解决的一个问题是如何保证用户输入的合法性。对于一些有确定范围的信息，可以使用复选框来让用户选择。复选框只有两种状态，即选中和未选中，这样就可以省略校验的代码。本实例根据用户的选择来显示不同的图片。实例运行效果如图2.12所示。

图2.12 实例运行效果

技术要点

JCheckBox 类是复选框的实现，复选框是一个可以被选中和取消选中的项，它将其状态显示给用户。按照惯例，可以选中组中任意数量的复选框。复选框常用方法如表2.4所示。

表2.4 复选框的常用方法

方 法 名	作 用
JCheckBox(String text)	创建一个带文本的、最初未被选中的复选框
addActionListener(ActionListener l)	将一个 ActionListener 添加到复选框中
isSelected()	判断复选框是否被选中
setMnemonic(int mnemonic)	设置当前模型上的键盘助记符
setSelected(boolean b)	设置复选框被选中

实现过程

（1）编写JCheckBoxTest类，该类继承了JFrame。在框架中包括4个复选框和一个用来显示图片的标签。

（2）编写do_checkBox1_actionPerformed()方法。该方法是IDE自动生成的，用于实现对选择复选框事件的监听。该方法实现了设置标签图片的功能。核心代码如下：

```
    protected void do_checkBox1_actionPerformed(ActionEvent
e) {
      if(checkBox1.isSelected())
  {                                    //如果复选框被选中
        ImageIcon icon=new
    ImageIcon("src/images/1.png");          //创建图片图标
        label.setIcon(icon);
      //设置图标
      }
    }
```

举一反三

根据本实例，读者可以实现以下功能。

设计投票复选框。

判断复选框是否被选中。

实例057 简单的投票计数软件

这是一个可以提高基础性能的实例

实例位置：光盘\mingrisoft\02\Ex02_57

实例说明

日常生活中，经常听到少数服从多数这句话。那么怎么知道哪个是少数，哪个是多数呢？通常是通过投票完成的。本实例实现一个简单的投票计算软件。实例运行效果如图2.13所示。

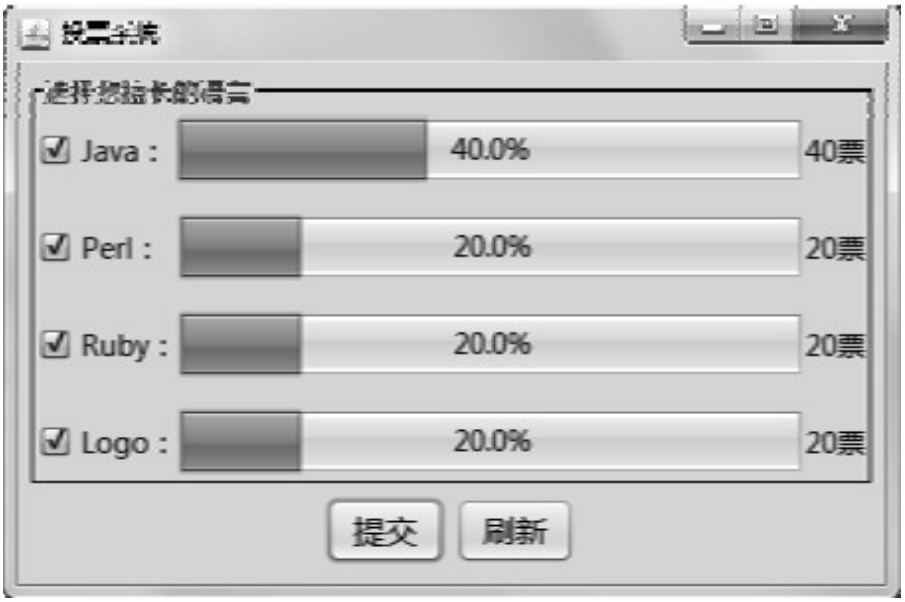


图2.13 实例运行效果

技术要点

本实例应用到的关键技术请参见实例056。

实现过程

（1）编写VoteSystem类，该类继承了JFrame。在框架中，共包括4个复选框、4个进度条、4个标签和两个按钮。“提交”按钮用于重新计算投票的结果。“刷新”按钮用于重置复选框。

（2）编写do_submitButton_actionPerformed()方法。该方法首先获得历史投票数，然后根据用户在复选框的选择结果重新计算投票结果，并在进度条和标签中显示。核心代码如下：

```
    protected void
do_submitButton_actionPerformed(ActionEvent e) {
        String text1=
    label1.getText();                                    //获得标签中
    的文本
        int number1= Integer.parseInt(text1.substring(0,
    text1.length() - 1));  //获得票数
        String text2=
    label2.getText();                                    //获得标签中
    的文本
        int number2= Integer.parseInt(text2.substring(0,
    text2.length() - 1));  //获得票数
        String text3=
    label3.getText();                                    //获得标签中
    的文本
        int number3= Integer.parseInt(text3.substring(0,
    text3.length() - 1));  //获得票数
        String text4=
    label4.getText();                                    //获得标签中
    的文本
        int number4= Integer.parseInt(text4.substring(0,
    text4.length() - 1));  //获得票数
        if (checkBox1.isSelected())
    {                                  //如果复选框被选中
          number1++;                                          //
    票数加1
```

```
        label1.setText(number1+
    "票");                          //更新标签
      }
      if (checkBox2.isSelected())
  {                                 //如果复选框被选中
        number2++;                                          //
    票数加1
        label2.setText(number2+
    "票");                          //更新标签
      }
      if (checkBox3.isSelected())
  {                                 //如果复选框被选中
        number3++;                                          //
    票数加1
        label3.setText(number3+
    "票");                          //更新标签
      }
      if (checkBox4.isSelected())
  {                                 //如果复选框被选中
        number4++;                                          //
    票数加1
        label4.setText(number4+
    "票");                          //更新标签
      }
      double
  total=number1+number2+number3+number4;          //计算总共
  的票数
```

```
    progressBar1.setString(number1 * 100 / total+
"%");                //在进度条上显示所占比例的文本信息
    progressBar1.setValue(number1);
    //在进度条上显示票数
    progressBar2.setString(number2 * 100 / total+
"%");                //在进度条上显示所占比例的文本信息
    progressBar2.setValue(number2);
    //在进度条上显示票数
    progressBar3.setString(number3 * 100 / total+
"%");                //在进度条上显示所占比例的文本信息
    progressBar3.setValue(number3);
    //在进度条上显示票数
    progressBar4.setString(number4 * 100 / total+
"%");                //在进度条上显示所占比例的文本信息
    progressBar4.setValue(number4);
    //在进度条上显示票数
  }
```

举一反三

根据本实例，读者可以实现以下功能。

显示排名第一的投票对象和其票数、显示总共的投票人数。

显示每人可以投的票数、保存投票的结果。

实例058 单选按钮的简单应用

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_58

实例说明

在使用复选框时，用户可以选择任意多个选项，但有时只希望用户选择选项组中的一个。最典型的例子就是性别，通常希望用户在男女之间选择一个。当用户选择一项之后，前次选择会自动取消。在Swing中，通常使用单选按钮来实现该功能。本实例利用单选按钮来浏览图片。实例运行效果如图2.14所示。

图2.14 实例运行效果

技术要点

JRadioButton 用于实现一个单选按钮，该单选按钮可被选中或取消选中，并可为用户显示其状态。与 ButtonGroup 对象配合使用可创建一组按钮，一次只能选择其中的一个按钮（创建一个ButtonGroup对象并用其add方法将JRadioButton对象包含在此组中）。单选按钮的常用方法如表2.5所示。

表2.5 单选按钮的常用方法

方 法 名	作 用
JRadioButton(String text)	创建一个具有指定文本的状态为未选中的单选按钮
addActionListener(ActionListener l)	将一个 ActionListener 添加到单选按钮中
isSelected()	判断单选按钮是否被选中
setMnemonic(int mnemonic)	设置当前模型上的键盘助记符
setSelected(boolean b)	设置单选按钮被选中

实现过程

（1）编写JRadioButtonTest类，该类继承了JFrame。在框架中包括3个单选按钮和一个用来显示图片的标签。

（2）编写do_radioButton1_actionPerformed()方法。该方法是IDE自动生成的，用于实现对选中单选按钮事件的监听。该方法实现了设置标签图片的功能。核心代码如下：

```
    protected void
do_radioButton1_actionPerformed(ActionEvent e) {
      if (radioButton1.isSelected())
  {                                        //如果单选按钮被选中
        ImageIcon icon=new
    ImageIcon("src/images/1.png");                  //创建图片图
    表
        label.setIcon(icon);
            //设置图标
      }
    }
```

举一反三

根据本实例，读者可以实现以下功能。

将单选按钮放入分组控件中。

判断单选按钮是否被选中。

实例059 能显示图片的组合框

这是一个可以启发思维的实例

实例位置：光盘\mingrisoft\02\Ex02_59

实例说明

在屏幕空间有限的情况下，使用单选按钮并不合适，因为需要列出所有的选项。此时可以考虑使用组合框。组合框类包括两部分，即一个文本域和一个下拉列表。对于普通的组合框使用非常简单，将用

户的选项组成一个数组或向量传递给组合框的构造方法即可。本实例用来实现在组合框中显示图片，这样可以使界面更加美观。实例运行效果如图2.15所示。

图2.15 实例运行效果

技术要点

JComboBox 是将按钮或可编辑字段与下拉列表组合的控件。用户可以从下拉列表中选择值，下拉列表在用户请求时显示。如果使组合框处于可编辑状态，则组合框将包括用户可在其中输入值的可编辑字段。JComboBox的常用方法如表2.6所示。

表2.6 JComboBox的常用方法

方 法 名	作 用
JComboBox(Object[] items)	创建包含指定数组中的元素的 JComboBox
addActionListener(ActionListener l)	添加 ActionListener
addItem(Object anObject)	为组合框添加项
getItemCount()	返回组合框中的项数
getRenderer()	返回用于显示 JComboBox 字段中所选项的渲染器
getSelectedIndex()	返回列表中与给定项匹配的第一个选项
getSelectedItem()	返回当前所选项
isEditable()	如果 JComboBox 可编辑，则返回 true
removeAllItems()	从列表项中移除所有项
removeItem(Object anObject)	从列表项中移除项
removeItemAt(int anIndex)	移除 anIndex 处的项
setEditable(boolean aFlag)	确定 JComboBox 字段是否可编辑
setMaximumRowCount(int count)	设置 JComboBox 显示的最大行数
setRenderer(ListCellRenderer aRenderer)	设置渲染器，该渲染器用于绘制列表项和选择的项
setSelectedIndex(int anIndex)	选择索引 anIndex 处的项
setSelectedItem(Object anObject)	将组合框显示区域中的所选项设置为参数中的对象

本实例还使用了ListCellRenderer 接口，它标识可用作“橡皮图章”以绘制 JList 中单元格的控件。该接口定义了一个getListCellRendererComponent()方法，该方法返回一个配置好的控件来显示特定值。该方法的声明如下：

getListCellRendererComponent(JList list, Object value, int index, boolean isSelected, boolean cellHasFocus)

该方法的返回值是Component，各个参数的说明如表2.7所示。

表2.7 getListCellRenererComponent方法的参数说明

参 数 名	作 用
list	正在绘制的 JList
value	由 list.getModel().getElementAt(index)返回的值
index	单元格索引
isSelected	如果选择了指定的单元格，则为 true
cellHasFocus	如果指定的单元格拥有焦点，则为 true

实现过程

（1）编写ComboBoxRenderer类，该类继承了JLabel并且实现了ListCellRenderer。该类用于生成组合框中的各个选项。代码如下：

```
    public class ComboBoxRenderer extends JLabel implements
ListCellRenderer {
      private static final long serialVersionUID
  =-3189390364606561O4L;
      privateMap<String, ImageIcon>
  content;                              //保存图片和其说明
      public ComboBoxRenderer(Map<String, ImageIcon> content)
  {
        this.content = content;
        setOpaque(true);
        //设置标签为不透明
        setHorizontalAlignment(CENTER);
          //水平方向居中对齐
        setVerticalAlignment(CENTER);
          //垂直方向居中对齐
      }
      @Override
      publicComponent getListCellRendererComponent(JList
  list,Object value, int index,boolean
  isSelected,booleancellHasFocus) {
        String key=
    (String)value;                                    //将组合
    框的一个值转换成字符串
        if (isSelected)
    {                                                 //根据是否处于
    选择状态而更改外观
          setBackground(list.getSelectionBackground());
```

```
            setForeground(list.getSelectionForeground());
        } else {
            setBackground(list.getBackground());
            setForeground(list.getForeground());
        }
        setText(key);
        //设置标签的文本
        setIcon(content.get(key));
            //设置标签的图标
        setFont(list.getFont());
              //设置标签的字体
        return this;
      }
    }
```

（2）编写JComboBoxTest类，该类继承了JFrame。在框架中显示一个组合框，其构造方法中增加了构造组合框的方法。核心代码如下：

```
    public JComboBoxTest() {
      setTitle("\u663E\u793A\u56FE\u7247\u7684\u7EC4\u5408\u6
  846");       //设置框架的标题
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            //设置框架在关闭时退出
      setBounds(100,100,200,150);
        //设置显示位置和大小
      contentPane=new
  JPanel();                                          //创建面板对象
```

```
    contentPane.setBorder(newEmptyBorder(5,5,5,5));
          //设置面板边距
    contentPane.setLayout(newBorderLayout(0,
0));                          //设置面板布局
    setContentPane(contentPane);
    Map<String, ImageIcon> content=newLinkedHashMap<String,
ImageIcon>();
    content.put("图片1",new
ImageIcon("src/images/1.png"));              //增加由图标说
明和图标组成的映射
    content.put("图片2",new
ImageIcon("src/images/2.png"));              //增加由图标说
明和图标组成的映射
    content.put("图片3",new
ImageIcon("src/images/3.png"));              //增加由图标说
明和图标组成的映射
    JComboBox comboBox=new
JComboBox(content.keySet().toArray());    //利用键值构造组
合框
    ComboBoxRenderer
renderer=newComboBoxRenderer(content);        //创建渲染器
    comboBox.setRenderer(renderer);
        //设置渲染器
    comboBox.setMaximumRowCount(3);
    //设置组合框最多显示3行可选项
    comboBox.setFont(newFont("微软雅
黑",Font.PLAIN,16));              //设置组合框字体
```

```
        contentPane.add(comboBox,BorderLayout.CENTER);
                //将组合框布局在框架中央
    }
```

举一反三

根据本实例，读者可以实现以下功能。

用渲染器设置列表项。

自定义选项列表。

实例060 使用滑块来选择日期

本实例可以提高工作效率

实例位置：光盘\mingrisoft\02\Ex02_60

实例说明

当可以选择的选项很多时，使用单选按钮并不理想，因为需要创建大量的按钮。此时可以考虑使用滑块。滑块可以让用户在一组离散值中进行选择。本实例将使用滑块来选择日期，实例运行效果如图2.16所示。

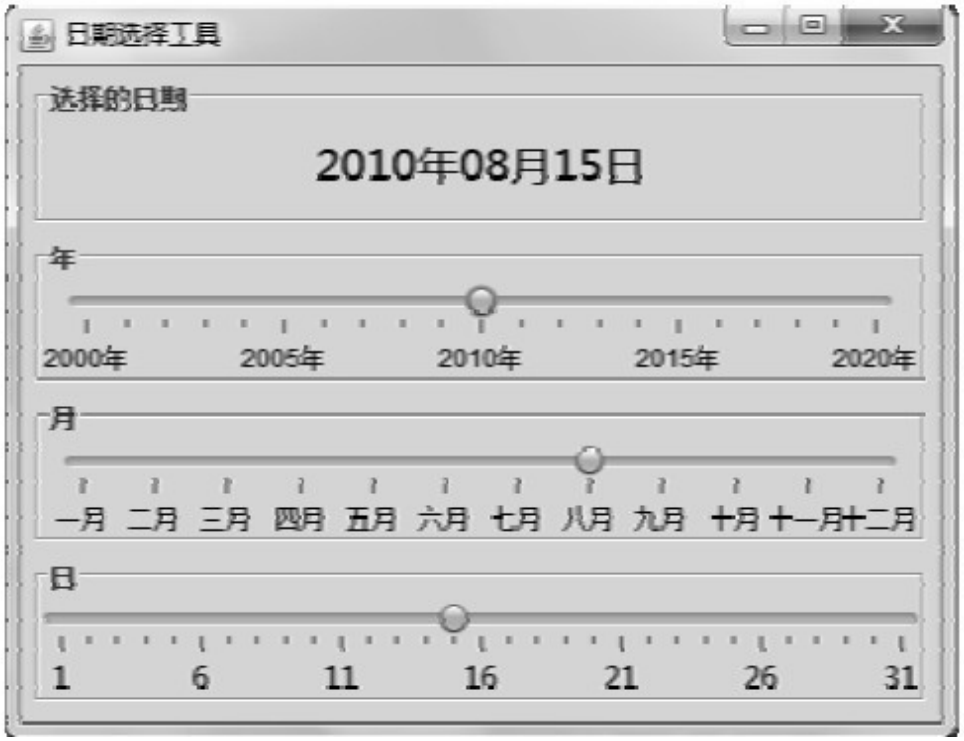


图2.16 实例运行效果

技术要点

JSlider是一个让用户以图形方式在有界区间内通过移动滑块来选择值的控件。当用户滑动滑块时，其值会在最大值与最小值之间变化。还可以给滑块增加标尺、标尺标签等进行修饰，其常用的方法如表2.8所示。

表2.8 滑块的常用方法

方 法 名	作 用
JSlider(int min, int max, int value)	用指定的最小值、最大值和初始值创建一个水平滑块
addChangeListener(ChangeListener l)	将一个 ChangeListener 添加到滑块
getValue()	从 BoundedRangeModel 返回滑块的当前值
setFont(Font font)	设置控件的字体
setInverted(boolean b)	指定为 true，则反转滑块显示的值范围
setMajorTickSpacing(int n)	此方法设置主刻度标记的间隔
setMaximum(int maximum)	将滑块的最大值设置为 maximum
setMinimum(int minimum)	将滑块的最小值设置为 minimum
setMinorTickSpacing(int n)	此方法设置次刻度标记的间隔
setPaintLabels(boolean b)	确定是否在滑块上绘制标签
setPaintTicks(boolean b)	确定是否在滑块上绘制刻度标记
setPaintTrack(boolean b)	确定是否在滑块上绘制滑道
setSnapToTicks(boolean b)	指定为 true，则滑块解析为最靠近用户放置滑块处的刻度标记的值
setValue(int n)	将滑块的当前值设置为 n

实现过程

（1）编写do_this_windowActivated()方法，该方法用于监听窗体活动事件。在该方法中，对滑块进行了基本设置，如最大值、最小值、起始值等。将时间设置成当前时间。核心代码如下：

```
protected void do_this_windowActivated(WindowEvent e) {
  yearSlider.setMaximum(2020);
      //将yearSlider滑块的最大值设置成2020
  yearSlider.setMinimum(2000);
      //将yearSlider滑块的最小值设置成2000
  yearSlider.setMajorTickSpacing(5);
          //将yearSlider滑块的主刻度设置成5
  yearSlider.setMinorTickSpacing(1);
          //将yearSlider滑块的副刻度设置成1
  yearSlider.setValue(calendar.get(Calendar.YEAR));
              //将yearSlider滑块的值设置成当前年
```

```
    Dictionary<Integer,Component>yearLabel=newHashtable<Integer,Component>();
    yearLabel.put(2000,new JLabel("2000年"));                    //为2000增加标签“2000年”
    yearLabel.put(2005,new JLabel("2005年"));                    //为2005增加标签“2005年”
    yearLabel.put(2010,new JLabel("2010年"));                    //为2010增加标签“2010年”
    yearLabel.put(2015,new JLabel("2015年"));                    //为2015增加标签“2015年”
    yearLabel.put(2020,new JLabel("2020年"));                    //为2020增加标签“2020年”
    yearSlider.setLabelTable(yearLabel);
            //为yearSlider增加标签
    yearSlider.addChangeListener(cl);
          //为yearSlider增加监听
    monthSlider.setMaximum(12);
      //将monthSlider滑块的最大值设置成12
    monthSlider.setMinimum(1);
        //将monthSlider滑块的最小值设置成1
    monthSlider.setMajorTickSpacing(1);
          //将monthSlider滑块的主刻度设置成1
    monthSlider.setValue(calendar.get(Calendar.MONTH)+1);
            //将monthSlider滑块的值设置成当月
    String[]months=(newDateFormatSymbols()).getShortMonths();          //获得本地月份字符串数组
```

```
    Dictionary<Integer,Component>monthLabel=newHashtable<In
teger,Component>(12);
    for (int i = 0; i < 12; i++) {
      monthLabel.put(i+1,new
  JLabel(months[i]));                          //为1~12增加标签
    }
    monthSlider.setLabelTable(monthLabel);
            //为monthSlider增加标签
    monthSlider.addChangeListener(cl);
          //为monthSlider增加监听
    daySlider.setMaximum(calendar.getMaximum(Calendar.DAY_O
F_MONTH)); //最大值设置成当月天数
    daySlider.setMinimum(1);
        //将daySlider滑块的最小值设置成1
    daySlider.setMajorTickSpacing(5);
          //将daySlider滑块的主刻度设置成5
    daySlider.setMinorTickSpacing(1);
          //将daySlider滑块的副刻度设置成1
    daySlider.setValue(calendar.get(Calendar.DATE));
              //将daySlider滑块的值设置成当前天
    daySlider.addChangeListener(cl);
          //为daySlider增加监听
    dateLabel.setText(dateFormat.format(newDate()));
              //用标签显示当前时间
  }
```

（2）编写内部类DateListener，该类继承了ChangeListener，用来监听滑块的变化事件。根据用户选择的不同日期来更新标签的内

容。代码如下：

```
private class DateListener implements ChangeListener {
  @Override
  public void stateChanged(ChangeEvent e) {
    calendar.set(yearSlider.getValue(),
monthSlider.getValue() - 1, 1);
    intmaxDays=
calendar.getActualMaximum(Calendar.DAY_OF_MONTH);    //获
得月最大天数
    if (daySlider.getMaximum() != maxDays) {
      daySlider.setValue(Math.min(daySlider.getValue(),ma
  xDays));          //设置滑块的值
      daySlider.setMaximum(maxDays);                //将滑
  块的最大值修改成当前月的最大天数
      daySlider.repaint();
            //重新绘制日期滑块
    }
    calendar.set(yearSlider.getValue(),monthSlider.getVal
ue() -
1,daySlider.getValue());
  //将日期设置成用户当前选择的日期
    dateLabel.setText(dateFormat.format(calendar.getTime(
)));                //更新标签的内容
  }
}
```

举一反三

根据本实例，读者可以实现以下功能。

使用其他方式，如修改框架的外观来美化滑块。
使用字符和图片作为标尺标签。

2.4 菜单控件的应用

实例061 模仿记事本的菜单栏

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_61

实例说明

在Windows操作系统中，自带了一款简单的文本编辑工具——记事本。记事本主要由菜单栏和文本区两部分组成。菜单栏实现了各种常用的功能，文本区用于让用户输入文本。本实例将实现一个类似记事本的菜单栏。实例运行效果如图2.17所示。

图2.17 实例运行效果

技术要点

在Swing中使用菜单的第一步是创建一个菜单栏保存各个菜单，并将菜单栏添加到框架上。代码如下：

```
JMenuBar menuBar = new JMenuBar();
setJMenuBar(menuBar);
```

第二步开始创建各个菜单及其菜单项，并将菜单项添加到菜单中。为了分类，可以使用分隔符将功能相近的菜单项分隔后添加到菜单中。代码如下：

```
JMenu fileMenu = new JMenu("\u6587\u4EF6(F)");
menuBar.add(fileMenu);
JMenuItem newMenuItem = new JMenuItem("\u65B0\u5EFA(N)");
fileMenu.add(newMenuItem);
```

实现过程

编写Notepad类，该类继承自JFrame。在其构造方法中，增加了一个菜单栏。在菜单栏中增加了Windows的记事本中的各个菜单项。核心代码如下：

```
public Notepad() {
  //省略设置框架属性的代码
  JMenuBar menuBar=new
JMenuBar();                                    //创建菜单栏
  setJMenuBar(menuBar);
        //在框架中增加菜单栏
  JMenu fileMenu=new
JMenu("\u6587\u4EF6(F)");                          //创建名为
“文件”的菜单
  fileMenu.setFont(newFont("微软雅
黑",Font.PLAIN,16));                    //设置菜单的字体
  menuBar.add(fileMenu);
          //将菜单添加到菜单栏中
  JMenuItem newMenuItem=new
JMenuItem("\u65B0\u5EFA(N)");              //创建新的菜单项
  newMenuItem.setFont(new Font("微软雅
黑",Font.PLAIN,16));                    //设置菜单的字体
  fileMenu.add(newMenuItem);
          //将菜单项添加到菜单中
```

```
    JMenuItemopenMenuItem=new
JMenuItem("\u6253\u5F00(O)...");              //创建新的菜单
项
    openMenuItem.setFont(newFont("微软雅
黑",Font.PLAIN,16));                  //设置菜单的字体
    fileMenu.add(openMenuItem);
            //将菜单项添加到菜单中
    JMenuItem saveMenuItem=new
JMenuItem("\u4FDD\u5B58(S)");              //创建新的菜单项
    saveMenuItem.setFont(newFont("微软雅
黑",Font.PLAIN,16));                  //设置菜单的字体
    fileMenu.add(saveMenuItem);
            //将菜单项添加到菜单中
    JMenuItem saveAsMenuItem=new
JMenuItem("\u53E6\u5B58\u4E3A(A)...");     //创建新的菜单项
    saveAsMenuItem.setFont(newFont("微软雅
黑",Font.PLAIN,16));                //设置菜单的字体
    fileMenu.add(saveAsMenuItem);
            //将菜单项添加到菜单中
    JSeparator separator1=new
JSeparator();                                  //创建分隔符
    fileMenu.add(separator1);
              //将分隔符添加到菜单中
    JMenuItempageSetMenuItem=new
JMenuItem("\u9875\u9762\u8BBE\u7F6E(U)...");
    pageSetMenuItem.setFont(newFont("微软雅
黑",Font.PLAIN,16));                //设置菜单的字体
```

```
    fileMenu.add(pageSetMenuItem);
                //将分隔符添加到菜单中
    JMenuItem printMenuItem=new
JMenuItem("\u6253\u5370(P)...");              //创建新的菜单
项
    printMenuItem.setFont(newFont("微软雅
黑",Font.PLAIN,16));                  //设置菜单的字体
    //将菜单项添加到菜单中fileMenu.add(printMenuItem);
    JSeparator separator2=new
JSeparator();                                  //创建分隔符
    fileMenu.add(separator2);
                //将分隔符添加到菜单中
    JMenuItem exitMenuItem=new
JMenuItem("\u9000\u51FA(X)");                 //创建新的菜单项
    exitMenuItem.setFont(newFont("微软雅
黑",Font.PLAIN,16));                    //设置菜单的字体
    fileMenu.add(exitMenuItem);
              //将菜单项添加到菜单中
    //省略其他菜单和文本区代码
  }
```

举一反三

根据本实例，读者可以实现以下功能。

增加一些助记符和快捷键，方便使用。

文本域中没有文本可以禁用保存菜单。

实例062 自定义纵向的菜单栏

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_62

实例说明

在使用软件时，其菜单栏通常是位于软件窗体顶部的。如果因为界面设计等方面的原因，需要将菜单栏放置在窗体左侧，则可以自定义一个纵向的菜单栏。本实例将实现这个功能。实例运行效果如图2.18所示。

技术要点

Java中菜单栏的主体是菜单，用JMenu对象表示。通过重写其setPopupMenuVisible()方法，可以设置菜单项弹出的位置，该方法的声明如下：

```
public void setPopupMenuVisible(boolean b)
```

图2.18 实例运行效果

参数说明

b：一个boolean值，true表示菜单可见，false表示隐藏。

为了让重写后的菜单更加好看，重写其getMinimumSize()方法，该方法用来设置控件的最小值，其声明如下：

```
public Dimension getMinimumSize()
```

实现过程

（1）编写HorizontalMenu类，该类继承了JMenu。在该类的构造方法中，设置了弹出菜单的布局是水平布局。重写其

getMinimumSize()方法使其最小值正好显示整个控件。重写其setPopupMenuVisible()方法，设置弹出菜单的显示位置。代码如下：

```
    public class HorizontalMenu extends JMenu {
      private static final long serialVersionUID =
  1943739671316999698L;
      public HorizontalMenu(String label) {
        super(label);
                  //调用父类的构造方法
        JPopupMenu
    popupMenu=getPopupMenu();                                    //
    获得菜单对象的弹出菜单
        popupMenu.setLayout(newBoxLayout(popupMenu,BoxLayout.
    LINE_AXIS));       //修改布局管理器
      }
      @Override
      public Dimension getMinimumSize() {
        return
    getPreferredSize();                                    //将控件
    的最小范围设置成显示控件的最佳范围
      }
      @Override
      public void setPopupMenuVisible(boolean b) {
        if (b != isPopupMenuVisible()) {
          if ((b== true)&& isShowing())
    {                                  //如果菜单处于显示状态
            if (getParent() instanceof JPopupMenu) {
```

```
                getPopupMenu().show(this,0,getHeight());
                        //修改弹出菜单的显示位置
              } else {
                getPopupMenu().show(this,getWidth(),0);
                    //修改弹出菜单的显示位置
              }
            } else {
              getPopupMenu().setVisible(false);
                    //设置弹出菜单不可见
            }
          }
        }
      }
```

（2）编写HorizontalMenuTest类，该类继承自JFrame。在其构造方法中，增加了菜单栏、菜单等，并且修改了内容窗格的布局。核心代码如下：

```
    public HorizontalMenuTest() {
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            //设置框架在退出时的状态
      setBounds(100,100,450,300);
        //设置框架的大小和显示的位置
      Container
  contentPane=getContentPane();                              //获
  得内容窗格
      contentPane.setBackground(Color.WHITE);
            //将内容窗格的背景颜色设置成白色
```

```
    JMenuBar menuBar=new
JMenuBar();                                    //建立菜单栏
    menuBar.setLayout(newBoxLayout(menuBar,BoxLayout.PAGE_A
XIS));    //修改菜单栏布局
    contentPane.add(menuBar,BorderLayout.WEST);
          //在内容窗格上增加菜单栏
    JMenu fileMenu=newHorizontalMenu("文件
(F)");                    //增加菜单项
    fileMenu.add("新建
(N)");                                          //增加菜单项
    fileMenu.add("打开
(O)...");                                        //增加菜单项
    fileMenu.add("保存
(S)");                                          //增加菜单项
    fileMenu.add("另存为
(A)...");                                       //增加菜单项
    fileMenu.add("页面设置
(U)...");                                       //增加菜单项
    //增加菜单项fileMenu.add("打印(P)...");
    fileMenu.add("退出(X)");//增加菜单项
    menuBar.add(fileMenu);//将菜单增加到菜单栏中
    //省略其他菜单
    }
```

举一反三

根据本实例，读者可以实现以下功能。

将菜单栏置于窗体右上方。

实现可拖曳的菜单栏。

实例063 复选框与单选按钮菜单

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_63

实例说明

复选框和单选按钮为用户的输入提供了便利，其实它们也可以用在菜单中。复选框菜单项和单选按钮菜单项分别对应于复选框和单选按钮，其使用的方式也是类似的。本实例将演示它们的用法。实例运行效果如图2.19所示。

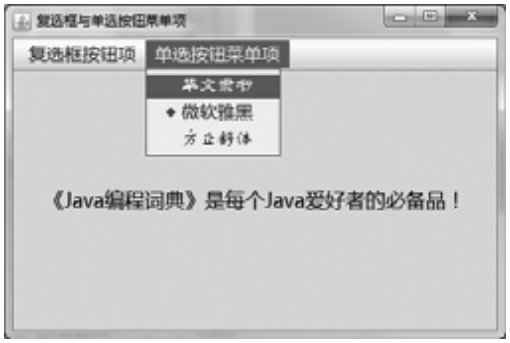


图2.19 实例运行效果

技术要点

JCheckBoxMenuItem 代表可以被选定或取消选定的菜单项。如果被选定，菜单项的旁边通常会出现一个复选标记。如果未被选定或被取消选定，菜单项的旁边就没有复选标记。像常规菜单项一样，复选框菜单项可以有与之关联的文本或图标，或者两者兼而有之。本实例使用以字符串为参数的构造方法创建复选框菜单项，该方法的声明如下：

JCheckBoxMenuItem(String text)

参数说明

text：CheckBoxMenuItem的文本。

JRadioButtonMenuItem是一个单选按钮菜单项的实现。JRadioButtonMenuItem属于一组菜单项中的一个菜单项，该组中只能

选择一个项。被选择的项显示其选择状态。选择此项的同时，其他任何以前被选择的项都切换到未选择状态。要控制一组单选按钮菜单项的选择状态，可使用 ButtonGroup 对象。本实例使用以字符串为参数的构造方法创建单选按钮菜单项，该方法的声明如下：

JRadioButtonMenuItem(String text)

参数说明

text：RadioButtonMenuItem 的文本。

实现过程

（1）编写类 FontChooser，该类继承了 JFrame。在框架的菜单栏中有两个菜单，分别用来演示复选框菜单项和单选按钮菜单项。在面板中显示带字符串的标签。

（2）使用匿名类创建listener对象，用来监听复选框菜单项被选择的事件用于实现是否让字体变粗和变斜体的功能。核心代码如下：

```
private ActionListener listener = new ActionListener() {
  @Override
  public void actionPerformed(ActionEvent e) {
    intmode=0;                                    //利用整
数保存字体的状态
    if (bold.isSelected()) {
      mode+=Font.BOLD;                           //如果加粗
复选框按钮项被选择，则让mode值发生变化
    }
    if (italic.isSelected()) {
      mode+=Font.ITALIC;                         //如果斜
体复选框按钮项被选择，则让mode值发生变化
    }
```

```
        Font font= label.getFont();                                    //
    获得标签正在使用的字体
        label.setFont(newFont(font.getName(),mode,
    font.getSize()));  //更新标签的字体
      }
    };
```

（3）对于不同的单选按钮菜单项，其实现的功能是类似的，在此选择显示为“华文隶书”的单选按钮进行讲解。编写方法do_radioButtonMenuItem1_actionPerformed()，用于监听单选按钮菜单项被选择的事件。核心代码如下：

```
    protected void
do_radioButtonMenuItem1_actionPerformed(ActionEvent e) {
      Font font=
  label.getFont();                                                 //获得标
  签正在使用的字体
      label.setFont(new Font("华文隶书", font.getStyle(),
  font.getSize()));    //更新标签的字体
  }
```

举一反三

根据本实例，读者可以实现以下功能。

使用复选框菜单项。

使用单选按钮菜单项。

实例064 包含图片的弹出菜单

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\02\Ex02_64

实例说明

除了固定的菜单栏，还有一种比较特殊的菜单，即弹出菜单。在Windows操作系统的桌面上，单击鼠标右键就会出现一个弹出式菜单。该菜单可以用来排列桌面图片、创建文件或文件夹等。本实例将演示如何在Swing框架中创建包含图片的弹出式菜单。实例的运行效果如图2.20所示。

图2.20 实例运行效果

技术要点

JPopupMenu 用于实现弹出菜单，弹出菜单是一个可弹出并显示一系列选项的小窗口。JPopupMenu 用于用户在菜单栏上选择菜单项时显示的菜单，还用于当用户选择菜单项并激活它时显示的“右拉式（pull-right）”菜单。最后，JPopupMenu 还可以在想让菜单显示的任何其他位置使用。例如，当用户在指定区域中右击时。创建弹出式菜单的语法如下：

```
JPopupMenu popup = new JPopupMenu();
```

在创建完弹出式菜单后，需要调用其所在控件的setComponentPopupMenu()方法来使用该菜单。该方法的声明如下：

```
public void setComponentPopupMenu(JPopupMenu popup)
```

参数说明

popup：分配给此控件的弹出菜单，可以为null。

实现过程

（1）编写PopupMenuTest类，该类继承自JFrame。在框架中只增加了一个标签，用于显示用户在弹出式菜单上选择的操作。代码如下：

```
public PopupMenuTest() {
    //省略设置框架属性的代码
    JPopupMenu popupMenu=new
JPopupMenu();                                    //创建弹出式菜
单
    contentPane.setComponentPopupMenu(popupMenu);
                    //为面板增加弹出式菜单
    JMenuItem cut=new
JMenuItem("\u526A\u5207");                                  //
创建新菜单项
    cut.setIcon(new
ImageIcon(PopupMenuTest.class.getResource("/images/cut.png"
)));
    cut.setFont(newFont("微软雅
黑",Font.PLAIN,16));                                //设置菜单
项字体
    cut.addActionListener(listener);
                        //增加监听
    popupMenu.add(cut);
                //增加菜单项
    //省略其他菜单项和标签的代码
}
```

（2）使用匿名类创建listener对象，用来监听弹出式菜单的菜单项被选择的事件，用于实现改变标签文本的功能。核心代码如下：

```
private ActionListener listener = new ActionListener() {
  @Override
  public void actionPerformed(ActionEvent e) {
    label.setText(e.getActionCommand());
          //设置标签的文本为用户选择的操作
  }
};
```

举一反三

根据本实例，读者可以实现以下功能。

使用弹出式菜单。

为弹出菜单增加保存功能。

实例065 工具栏的实现与应用

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_65

实例说明

除了标准菜单和弹出式菜单，还有一种让用户使用鼠标选择操作的方法。对于一些功能非常复杂的软件，可以将一些常用的操作放置在一个工具栏中方便用户使用。本实例将演示如何使用工具栏。实例运行效果如图2.21所示。

(a) (b)

图2.21 实例运行效果

技术要点

JToolBar 提供了一种快速访问程序常用功能的方法。工具栏的特殊之处在于可以随意地移动。工具栏的常用方法如表2.9所示。

表2.9 工具栏的常用方法

方 法 名	作 用
add(Component comp)	在工具栏中增加控件
addSeparator()	将默认大小的分隔符添加到工具栏的末尾
setToolTipText(String text)	注册要在工具提示中显示的文本

实现过程

编写ToolBarTest类，该类继承自JFrame。在其构造方法中，创建了工具栏并增加了4个按钮。构造方法的核心代码如下：

```
public ToolBarTest() {
  //省略设置框架属性的代码
  JToolBar toolBar=new
JToolBar();                                    //创建工具栏
  contentPane.add(toolBar,BorderLayout.NORTH);
                  //设置工具栏的布局
  JButton cutButton=new
JButton("");                                   //新建一个按
```

钮

```
    cutButton.setToolTipText("\u526A\u5207");
                    //为按钮增加提示信息
    cutButton.setIcon(new
ImageIcon(ToolBarTest.class.getResource("/images/cut.png"))
);
    toolBar.add(cutButton);
            //将按钮增加到工具栏上
    //省略其他按钮和标签的代码
  }
```

举一反三

根据本实例，读者可以实现以下功能。

使工具栏具有提示功能。

将常用的功能拖放到工具栏。

2.5 列表的应用

实例066 修改列表项显示方式

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_66

实例说明

当可供用户选择的选项比较多时，使用单选按钮或复选框会占用较多的空间。此时可以考虑使用列表。列表可以将若干选项组织起来，根据显示的列表项个数可以调节其占用的空间。本实例将演示列表布局的用法。实例运行效果如图2.22所示。

(a) 水平排列　　(b) 垂直排列

图2.22 实例运行效果

技术要点

JList类是显示对象列表并且允许用户选择一个或多个项的控件。在为列表增加列表项时，可以使用对象数组或列表模型。列表包含了很多方法，本实例使用的方法如表2.10所示。

表2.10 列表的常用方法

方 法 名	作 用
setLayoutOrientation(int layoutOrientation)	定义布置列表单元的方式
setListData(Object[] listData)	根据一个对象数组构造只读 ListModel，然后对此模型调用 setModel
setVisibleRowCount(int visibleRowCount)	根据不同的布局方式，设置可见的行或列数

列表有3种常见的布局方式，使用3个不同的域变量表示，其说明如表2.11所示。

表2.11 列表的布局方式

常 量 名	作 用
HORIZONTAL_WRAP	指示“报纸样式”布局，单元按先水平方向后垂直方向排列
VERTICAL	指示单个列中单元的垂直布局；默认布局
VERTICAL_WRAP	指示“报纸样式”布局，单元按先垂直方向后水平方向排列

实现过程

（1）编写类JListTest，该类继承自JFrame。在框架中包括了一个列表和一个按钮组。按钮组包含了3个按钮，用来表示列表的3种布局方式。

（2）编写方法do_this_windowActivated()，用来监听窗体激活事件。在该方法中，设置了列表的数据。核心代码如下：

```
protected void do_this_windowActivated(WindowEvent e) {
  String[]
listData=newString[12];                                    //创
建一个含有12个元素的数组
  for(int i=0;i<listData.length;i++) {
    listData[i]= "明日科技"+
(i+1);                                    //为数组中的各个元素
赋值
  }
  list.setListData(listData);
        //为列表增加列表项
```

```
    }
```

（3）编写方法 do_radioButton1_actionPerformed()，用来监听单选按钮 radioButton1 被选中事件，该方法用来修改列表的布局方式并更新界面。核心代码如下：

```
    protected void
do_radioButton1_actionPerformed(ActionEvent e) {
      list.setLayoutOrientation(JList.HORIZONTAL_WRAP);
            //修改列表的布局方式
      scrollPane.revalidate();
            //更新界面
    }
```

举一反三

根据本实例，读者可以开发以下程序。

先水平方向后垂直方向排列。

先垂直方向后水平方向排列。

实例067 修改列表项选择模式

这是一个可以启发思维的实例

实例位置：光盘\mingrisoft\02\Ex02_67

实例说明

默认情况下，列表项的选择个数和方式是没有限制的。用户既可以连续选择（使用 Shift键），又可以间隔选择（使用Ctrl键）。通过修改列表的选择模式，可以实现单选按钮或复选框的功能。本实例将演示列表的各种选择模式。实例运行效果如图2.23所示。

图2.23 实例运行效果

技术要点

setSelectionMode()方法可以设置列表的选择模式。方法是在选择模型上直接设置选择模式的覆盖方法。该方法的声明如下：

```
public void setSelectionMode(int selectionMode)
```

参数说明

selectionMode：列表支持的选择模式。

ListSelectionModel接口定义了列表支持的选择模式，其详细说明如表2.12所示。

表2.12 列表选择模式常量

常 量 名	作 用
MULTIPLE_INTERVAL_SELECTION	一次选择一个或多个连续的索引范围
SINGLE_INTERVAL_SELECTION	一次选择一个连续的索引范围
SINGLE_SELECTION	一次选择一个列表索引

实现过程

（1）编写类JListSelectModelTest，该类继承了JFrame。在框架中包含了3个列表，分别用来演示不同的选择模式。构造方法的核心代码如下：

```
  public JListSelectModelTest() {
    //省略框架相关属性的设置代码
    JScrollPane scrollPane1=new
JScrollPane();                                    //创建一个滚动
面板来保存列表
    panel.add(scrollPane1);
              //将滚动面板增加到面板中
    list1=new
JList();                                                    //
创建一个列表对象
    list1.setFont(newFont("微软雅
黑",Font.PLAIN,14));                          //设置列表的字体
    list1.setSelectionMode(ListSelectionModel.SINGLE_SELECT
ION);              //设置列表选择模式
    scrollPane1.setViewportView(list1);
                //将列表增加到滚动面板中
    label1=new
JLabel("\u5355\u9879\u9009\u62E9\u5217\u8868");
  //创建一个指定内容的标签
    label1.setFont(newFont("微软雅
黑",Font.PLAIN,14));                          //设置标签的字体
    label1.setHorizontalAlignment(SwingConstants.CENTER);
                  //设置标签文本的显示位置
    scrollPane1.setColumnHeaderView(label1);
                //将标签增加到滚动面板中
    //省略其他滚动面板相关代码
  }
```

（2）编写方法 do_this_windowActivated()，用来监听窗体激活事件。在该方法中设置了 3个列表的数据。核心代码如下：

```
protected void do_this_windowActivated(WindowEvent e) {
  String[]
listData=newString[12];                                    //创
建一个含有12个元素的数组
  for(int i=0;i<listData.length;i++) {
    listData[i]= "明日科技"+
(i+1);                                    //为数组中的各个元素
赋值
  }
  list1.setListData(listData);
          //为列表1增加列表项
  list2.setListData(listData);
          //为列表2增加列表项
  list3.setListData(listData);
          //为列表3增加列表项
}
```

举一反三

根据本实例，读者可以实现以下功能。

一次选择一个连续的索引范围。

一次选择一个列表索引。

实例068 列表项的全选与不选

这是一个可以美化界面的实例

实例位置：光盘\mingrisoft\02\Ex02_68

实例说明

对于支持多选的列表，如果用户需要选择全部的列表项，则可以首先选择第一个列表项，然后按住Shift键，并在最后一个列表项上单击即可。显然这种方式有些麻烦，因此本实例将使用按钮来实现列表项的全选和不选功能。实例运行效果如图2.24所示。

(a) 全选　　(b) 不选

图2.24 实例运行效果

技术要点

为了实现全选功能，需要知道列表中共有多少列表项。由于列表并不负责数据的存储，因此需要先获得列表模型，并使用其getSize()方法获得列表项的格式。代码如下：

```
list.getModel().getSize()
```

setSelectionInterval()方法可以用来设置列表的选择区域，该方法的声明如下：

```
public void setSelectionInterval(int anchor,int lead)
```

参数说明

- anchor：要选择的第一个索引。
- lead：要选择的最后一个索引。

clearSelection()方法用于取消列表中的选择，该方法的声明如下：

```
public void clearSelection()
```

实现过程

（1）编写类JListSelectionModelTest，该类继承了JFrame。在框架中包含了一个列表和“全选”、“不选”两个按钮。

（2）编写方法do_selectAllButton_actionperformed()，用来监听用户单击“全选”按钮事件。在该方法中，实现了对列表项全选的功能。核心代码如下：

```
    protected void
do_selectAllButton_actionPerformed(ActionEvent e) {
      int end= list.getModel().getSize() -
  1;                                  //获得最后一个列表项的索引
      if (end>=0)
  {                                                    //如果索引不
  小于0，则列表项个数至少为1
        list.setSelectionInterval(0,
    end);                                  //选择全部的列表项
      }
    }
```

（3）编写方法 do_selectNoneButton_actionperformed()，用来监听用户单击“不选”按钮事件。在该方法中，取消了对列表项的选择。核心代码如下：

```
    protected void
do_selectNoneButton_actionPerformed(ActionEvent e) {
      list.clearSelection();
            //取消选择
    }
```

举一反三

根据本实例，读者可以实现以下功能。

使用复选框实现全选。

使用复选框取消全选。

实例069 监听列表项单击事件

这是一个可以美化界面的实例

实例位置：光盘\mingrisoft\02\Ex02_69

实例说明

对于基本的Swing控件，当用户使用时通常会触发动作事件。列表采用了另一种不同的事件机制：当用户单击列表项时，会触发列表选择事件。可以对该事件进行监听以对用户的选择进行响应。本实例演示如何实现此类监听。实例运行效果如图2.25所示。

图2.25 实例运行效果

技术要点

对于列表选择事件，通常的处理流程是，首先使用addListSelectionListener()方法为列表增加一个列表选择事件监听器，该方法的声明如下：

```
public void
addListSelectionListener(ListSelectionListener listener)
```

参数说明

listener：要添加的ListSelectionListener。

其次使用监听器中的 valueChanged()方法来处理用户选择列表项事件的结果。该方法的声明如下：

void valueChanged(ListSelectionEvent e)

参数说明

e：表现更改特征的事件。

实现过程

（1）编写方法do_this_windowActivated()，用来监听窗体激活事件。在该方法中初始化了列表的数据。核心代码如下：

```
protected void do_this_windowActivated(WindowEvent e) {
  String[] listData = new String[7];
  listData[0]= "《Java从入门到精通（第2
版）》";                                    //初始化数据
  listData[1]= "《PHP从入门到精通（第2
版）》";                                    //初始化数据
  listData[2]= "《VisualBasic从入门到精通（第2
版）》";                                  //初始化数据
  listData[3]= "《VisualC++从入门到精通（第2
版）》";                                    //初始化数据
  listData[4]= "《Java编程词
典》";                                                  //初始化数
据
  listData[5]= "《细说
Java》";                                                    //初
始化数据
  listData[6]= "《视频学
Java》";                                                  //初始
化数据
  list.setListData(listData);
                    //设置列表中的数据
```

```
    }
```

（2）编写方法 do_list_valueChanged()，用来监听列表项选择事件。在该方法中，根据用户选择的列表项来更新标签的信息。核心代码如下：

```
    protected void do_list_valueChanged(ListSelectionEvent e)
{
      label.setText("感谢您购买："+
  list.getSelectedValue());                              //更新标
  签的信息
    }
```

举一反三

根据本实例，读者可以实现以下功能。

列表选择事件的应用。

监听窗体激活事件。

实例070 监听列表项双击事件

这是一个可以美化界面的实例

实例位置：光盘\mingrisoft\02\Ex02_70

实例说明

默认情况下，列表控件不支持双击列表项来触发事件。而有些软件将用户双击列表项作为选择列表项并触发相关事件的快捷方式。为了让列表支持该操作，需要为列表增加鼠标监听器。本实例将实现一个响应用户双击列表项的事件。实例运行效果如图2.26所示。

图2.26 实例运行效果

技术要点

MouseListener是用于接收控件上“感兴趣”的鼠标事件（按下、释放、单击、进入或离开）的侦听器接口。本实例中仅对鼠标单击事件感兴趣，因此重写了 MouseAdapter 中的mouseClicked()方法，该方法的声明如下：

```
void mouseClicked(MouseEvent e)
```

实现过程

（1）编写方法do_this_windowActivated()，用来监听窗体激活事件。在该方法中初始化了列表的数据。核心代码如下：

```
protected void do_this_windowActivated(WindowEvent e) {
  String[] listData = new String[7];
  listData[0]= "《Java从入门到精通（第2
版）》";                                    //初始化数据
  listData[1]= "《PHP从入门到精通（第2
版）》";                                    //初始化数据
  listData[2]= "《VisualBasic从入门到精通（第2
版）》";                                  //初始化数据
  listData[3]= "《VisualC++从入门到精通（第2
版）》";                                    //初始化数据
  listData[4]= "《Java编程词
典》";                                                  //初始化数
```

```
据
    listData[5]= "《细说
Java》";                                                    //初
始化数据
    listData[6]= "《视频学
Java》";                                                  //初始
化数据
    list.setListData(listData);
                    //设置列表中的数据
  }
```

（2）编写方法 do_list_mouseClicked()，用来监听双击列表项事件。在该方法中，根据用户选择的列表项来更新标签的信息。核心代码如下：

```
  protected void do_list_mouseClicked(MouseEvent e) {
    if(e.getClickCount()==2)
{                                                  //如果列表项上发
生双击事件
      JList source=
(JList)e.getSource();                                          //
获得鼠标单击的列表
      label.setText("感谢您购买："+
source.getSelectedValue());                  //更新标签的信息
    }
  }
```

举一反三

根据本实例，读者可以开发以下程序。

应用鼠标按下、释放、单击、进入或离开事件开发程序。

处理拖曳事件。

实例071 实现自动排序的列表

这是一个可以提高分析能力的实例

实例位置：光盘\mingrisoft\02\Ex02_71

实例说明

为了使用户方便地选择列表项，通常会将列表项按某种顺序进行排序，如升序或降序。此时有两种实现方式，第一种是将排好顺序的数据添加到列表中；第二种是自定义能够排序的列表模型，每当向列表中增加元素时自动计算其位置。显然第二种更加灵活。本实例将自定义一个能排序的列表模型。实例运行效果如图2.27所示。

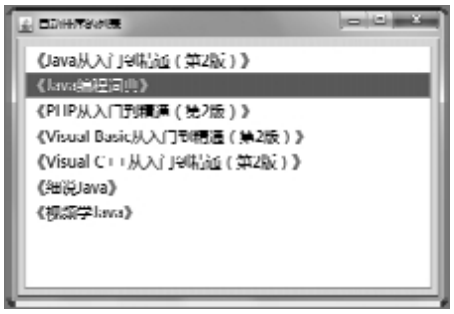


图2.27 实例运行效果

技术要点

列表的元素都是存储在列表模型中的。因此，需要使用ListModel接口的实现类，Java API中对该接口提供了若干不同的实现类，通常可以使用 DefaultListModel。该类使用 Vector 来存储元素，并不能很好地满足需求，所以本实例继承了其父类 AbstractListModel。它提供了ListModel接口定义的大部分方法。只需要实现获得列表长度的方法和获得指定索引处元素的方法，这两个方法的声明如下：

```
int getSize()
Object getElementAt(int index)
```

在元素的存储上，使用TreeSet，它会自动为存入其中的元素进行排序，能节省大量代码。

实现过程

（1）编写类SortedListModel，该类继承自AbstractListModel。在该类中使用TreeSet来保存数据，因此不用实现排序细节。除了重写抽象方法，该类还增加了一个add()方法，用来向TreeSet中增加数据。代码如下：

```
public class SortedListModel extends AbstractListModel {
  private static final long serialVersionUID
=-8908769624938773296L;
  private TreeSet<Object> model = new TreeSet<Object>();
  @Override
  publicObject getElementAt(int index)
{                                    //获得模型中指定索引的值
    return model.toArray()[index];
  }
  @Override
  public int getSize()
{                                                  //获得模型中
元素的个数
    return model.size();
  }
  publicvoid add(Object element)
{                                        //向TreeSet中增加元
素
    if (model.add(element)) {
      fireContentsChanged(this, 0, getSize());
```

```
        }
      }
    }
```

（2）编写类SortedListModelTest，该类继承自JFrame。在该类的构造方法中增加了一个列表，并设置其列表模型为支持排序的列表模型。构造方法的核心代码如下：

```
    public SortedListModelTest() {
      //省略框架属性设置和列表创建代码
      SortedListModelmodel=newSortedListModel();
          //创建可以排序的列表模型
      model.add("《Java从入门到精通（第2
版）》");                          //为列表模型增加元素
      model.add("《PHP从入门到精通（第2
版）》");                          //为列表模型增加元素
      model.add("《VisualBasic从入门到精通（第2
版）》");                  //为列表模型增加元素
      model.add("《VisualC++从入门到精通（第2
版）》");                  //为列表模型增加元素
      model.add("《Java编程词
典》");                                  //为列表模型增加元素
      model.add("《细说
Java》");                                    //为列表模型增
加元素
      model.add("《视频学
Java》");                                  //为列表模型增加
元素
```

```
        list.setModel(model);
            //设置列表模型
    }
```

举一反三

根据本实例，读者可以开发以下程序。

增加一组列表元素。

删除全部列表元素。

实例072 列表项的增加与删除

这是一个可以启发思维的实例

实例位置：光盘\mingrisoft\02\Ex02_72

实例说明

通常情况下，列表中的数据是由程序员指定的。用户只需要在列表中选择符合自己需求的数据即可，然而有时却显得不够灵活。本实例实现了让用户自己增加和删除列表项的功能，用户可以根据自己的需要添加列表项，也可以删除自己不喜欢的列表项。实例运行效果如图2.28所示。

图2.28 实例运行效果

技术要点

在使用列表时，需要注意的是列表本身并没有数据的增加、删除等操作，这些功能是在列表模型中实现的。通常，涉及列表模型的操作时，需要先创建列表模型的对象。ListModel是所有列表模型实现的接口，但是其只定义了 4 个必需的方法，因此并不常用。通常使用DefaultListModel，它增加了很多在 ListModel 中未定义的方法。本实例使用的方法如表 2.13所示。

表2.13 DefaultListModel的常用方法

方 法 名	作 用
addElement(Object obj)	将指定控件添加到此类表的末尾
removeElement(Object obj)	从此列表中移除参数的第一个（索引最小的）匹配项

实现过程

（1）编写类DynamicList，该类继承了JFrame。在框架中包含了一个列表和“增加”“删除”两个按钮。

（2）编写方法 do_addButton_actionPerformed()，用来监听单击“增加”按钮事件。在该方法中，实现了向列表中增加列表项的功能。核心代码如下：

```
protected void do_addButton_actionPerformed(ActionEvent e) {
    String text= JOptionPane.showInputDialog("添加元素");
                                    //利用对话框获得用户输入的新列表项
    if ((text !=null)&& (!text.trim().isEmpty()))
{                                   //判断列表项是否为空
      model.addElement(text.trim());
          //增加新列表项
    } else {
      return;
    }
  }
```

（3）编写方法 do_deleteButton_actionPerformed()，用来监听单击“删除”按钮事件。在该方法中实现了删除用户选择的列表项功能。核心代码如下：

```
    protected void
do_deleteButton_actionPerformed(ActionEvent e) {
      model.removeElement(list.getSelectedValue());
              //删除用户选择的列表项
    }
```

举一反三

根据本实例，读者可以实现以下程序。

增加列表项。

删除列表项。

实例073 查找特定的列表元素

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\02\Ex02_73

实例说明

对于列表项很少的列表，可以让用户一个一个查找自己需要的列表项。如果数据量非常大，则这种方式不适合。此时，最好为用户提供查找功能。本实例实现了根据用户指定的关键字在列表中运行查找的功能，实例运行效果如图2.29所示。

图2.29 实例运行效果

技术要点

在使用列表时，需要注意的是列表本身并没有数据的增加、删除等操作，这些功能是在列表模型中实现的。通常，涉及列表模型的操作时，需要先创建列表模型的对象。ListModel是所有列表模型实现的接口，但是其只定义了 4 个必需的方法，因此并不常用。通常使用DefaultListModel，它增加了很多在 ListModel 中未定义的方法。本实例使用的方法如表 2.14所示。

表2.14 DefaultListModel的常用方法

方 法 名	作 用
contains(Object elem)	测试指定对象是否为此类表中的控件
indexOf(Object elem)	搜索 elem 的第一次出现

实现过程

（1）编写类 SearchList，该类继承了 JFrame。在框架中包含了一个列表、一个文本框和一个“查找”按钮。

（2）编写方法do_button_actionPerformed()，用来监听单击“查找”按钮事件。在该方法中，实现了根据用户输入的关键字运行查找的功能。核心代码如下：

```
protected void do_button_actionPerformed(ActionEvent e) {
  String key=
textField.getText();                                    //获得用户
```

输入的关键字

```
    if ((key==null) || (key.trim().isEmpty()))
{                       //判断关键字是否为空
      JOptionPane.showMessageDialog(this, "请输入关键字",
  "",JOptionPane.WARNING_MESSAGE);
      //如果为空则提示用户输入关键字
      return;
    }
    if (model.contains(key))
{                                   //判断列表模型中是否包含
用户输入的关键字
      int
  index=model.indexOf(key);                          //如果包
  含则获得该关键字的索引
      list.setSelectedIndex(index);
      //设置该列表项处于选择状态
    } else {
      list.clearSelection();
      //清除列表项的选择状态
      JOptionPane.showMessageDialog(this, "未找到关键字",
  "",JOptionPane.WARNING_MESSAGE);
      //提示用户没有找到其输入的关键字
      return;
    }
  }
```

举一反三

根据本实例，读者可以实现以下程序。

使用add(int index, Object element)方法在指定位置插入元素。

使用remove(int index)方法删除列表中指定索引的元素。

实例074 包含图片的列表元素

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_74

实例说明

为了让软件的界面更加美观，可以添加图片来进行修饰。默认的列表项并不支持图片，因此，需要使用ListCellRenderer类来控制列表项的显示方式。本实例实现了为列表项增加图片的功能，实例运行效果如图2.30所示。

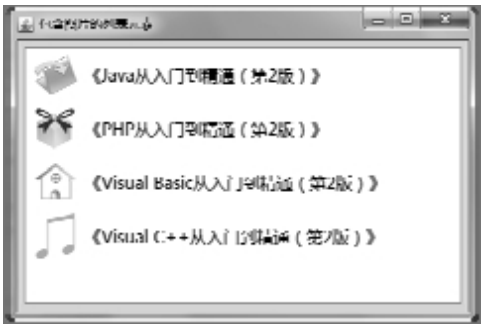


图2.30 实例运行效果

技术要点

JLabel对象可以显示文本、图像或同时显示文本和图像。可以通过设置垂直和水平对齐方式，指定标签显示区中的标签内容在何处对齐。默认情况下，标签在其显示区内垂直居中对齐。默认情况下，只显示文本的标签是开始边对齐，而只显示图像的标签则水平居中对齐。本实例使用的方法如表2.15所示。

表2.15 JLabel的常用方法

方 法 名	作 用
setIcon(Icon icon)	定义此控件将要显示的图标
setText(String text)	定义此控件将要显示的单行文本

实现过程

（1）编写类ImageListCellRenderer，该类实现了ListCellRenderer接口。在该接口定义的方法中，实现了为列表项增加图片的功能。代码如下：

```
public class ImageListCellRenderer implements
ListCellRenderer {
  @Override
  publicComponent getListCellRendererComponent(JList
 list,Object value, int index,boolean
 isSelected,booleancellHasFocus) {
    DefaultListCellRenderer
 defaultRenderer=newDefaultListCellRenderer();
    JLabel renderer= (JLabel)
 defaultRenderer.getListCellRendererComponent(list,value,
 index,
 isSelected,cellHasFocus);
          //获得getListCellRendererComponent的默认实现
    if (value instanceof Object[]) {
      Object values[]= (Object[]) value;          //
 将列表项转换成对象数组
      renderer.setIcon((Icon)
 values[0]);              //为标签设置图标
      renderer.setText((String)
 values[1]);            //为标签设置文本信息
```

```
        }
        return renderer;
      }
    }
```

（2）编写方法do_this_windowActivated()，用来监听窗体激活事件。在该方法中，为列表增加了数据并且设置了新的渲染类。核心代码如下：

```
    protected void do_this_windowActivated(WindowEvent e) {
      Object[][] data = new Object[4][2];
      data[0][0]=new
  ImageIcon("src/images/1.png");                    //初始化数据
      data[1][0]=new
  ImageIcon("src/images/2.png");                    //初始化数据
      data[2][0]=new
  ImageIcon("src/images/3.png");                    //初始化数据
      data[3][0]=new
  ImageIcon("src/images/4.png");                    //初始化数据
      data[0][1]= "《Java从入门到精通（第2
  版）》";                  //初始化数据
      data[1][1]= "《PHP从入门到精通（第2
  版）》";                  //初始化数据
      data[2][1]= "《VisualBasic从入门到精通（第2
  版）》";           //初始化数据
      data[3][1]= "《VisualC++从入门到精通（第2
  版）》";             //初始化数据
      list.setListData(data);
      //设置列表中的数据
```

```
    ListCellRenderer renderer = new
ImageListCellRenderer();
    list.setCellRenderer(renderer);
      //设置列表中的渲染工具
  }
```

举一反三

根据本实例，读者可以开发以下程序。

定义控件要显示的单行文本。

为列表项增加图片。

实例075 可以预览字体的列表

这是一个可以提高分析能力的实例

实例位置：光盘\mingrisoft\02\Ex02_75

实例说明

在Word 2007 软件中，可以在选择字体时预览字体的样式。这对于用户来说无疑是非常友好的。本实例也将实现一个可以预览字体样式的列表。列表项的名字是字体名，列表项的样式就是该字体的样式。实例运行效果如图2.31所示。

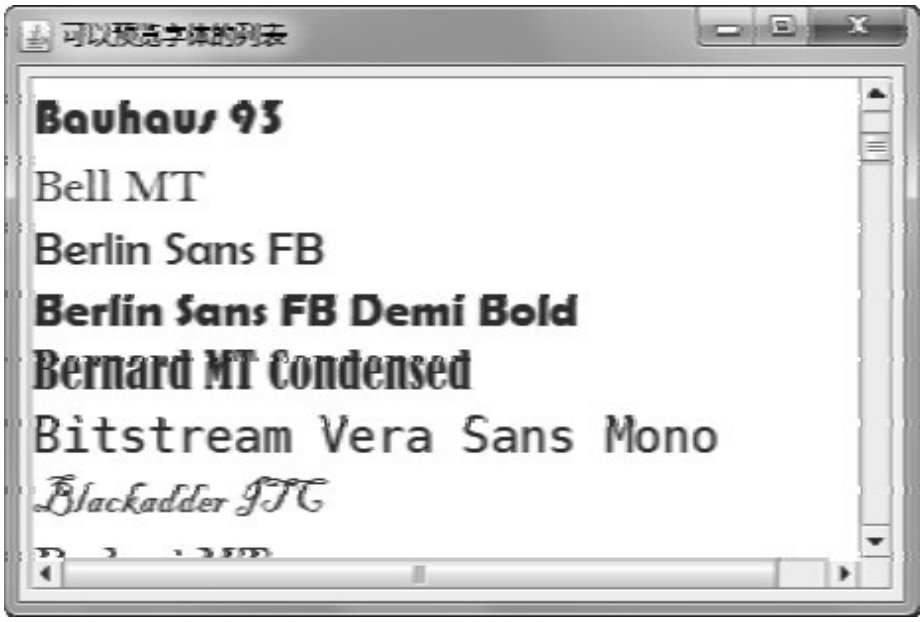


图2.31 实例运行效果

技术要点

为了获得本地计算机所支持的全部字体，需要使用GraphicsEnvironment 类。它描述了Java(tm)应用程序在特定平台上可用的 GraphicsDevice 对象和 Font 对象的集合。getLocalGraphicsEnvironment()方法可以获得该类的一个实例，该方法的声明如下：

```
public static GraphicsEnvironment
getLocalGraphicsEnvironment()
```

该类的getAvailableFontFamilyNames()方法可以获得本地计算机所支持的字体名称，该方法的声明如下：

```
public abstract String[] getAvailableFontFamilyNames()
```

实现过程

（1）编写类FontListCellRenderer，该类实现了ListCellRenderer接口。在该接口定义的方法中，实现了修改每个列表项字体的功能。代码如下：

```
    public class FontListCellRenderer implements
ListCellRenderer {
      @Override
      publicComponent getListCellRendererComponent(JList
  list,Object value, int index,boolean isSelected,boolean
  cellHasFocus){
        DefaultListCellRenderer
    defaultRenderer=newDefaultListCellRenderer();
        JLabel renderer= (JLabel)
    defaultRenderer.getListCellRendererComponent(list,value,
    index, isSelected,
    cellHasFocus);
                  //获得getListCellRendererComponent的默认实
    现
        Font font= (Font)
    value;                                  //获得列表项字体
        renderer.setFont(font);
      //设置标签的字体
        renderer.setText(font.getFontName());
        //设置标签的文本
        return renderer;
      }
    }
```

（2）编写方法do_this_windowActivated()，用来监听窗体激活事件。在该方法中，为列表增加了数据并且设置了新的渲染类。核心代码如下：

```
    protected void do_this_windowActivated(WindowEvent e) {
```

```
    String[]
fontNames=GraphicsEnvironment.getLocalGraphicsEnvironment()
.getAvailableFontFamilyNames();
    //获得系统支持的全部字体
    DefaultListModelmodel=newDefaultListModel();
    //创建表格模型
    for (String fontName : fontNames)
{                          //遍历全部字体并将其添加到表格模
型中
      model.addElement(new Font(fontName, Font.PLAIN, 24));
    }
    list.setModel(model);
        //设置表格模型
    ListCellRenderer renderer = new FontListCellRenderer();
    list.setCellRenderer(renderer);
            //设置列表中的渲染工具
  }
```

举一反三

根据本实例，读者可以开发以下程序。

实现可以预览字号的列表。

实现可以预览字体颜色的列表。

2.6 表格的应用

实例076 设置表头与列的高度

这是一个可以启发思维的实例

实例位置：光盘\mingrisoft\02\Ex02_76

实例说明

在默认情况下，表格中表头和表体的单元格高度是固定的。如果要修改单元格的外观，如放大字体，则会隐藏部分表格的内容。此时就需要修改单元格的高度。Swing 中的表格对于表头和表体的处理方式是不同的。本实例将演示如何自定义表头和表体的高度。实例运行效果如图2.32所示。

图2.32 实例运行效果

技术要点

JTable 控件用来显示和编辑常规二维单元表。JTable 有很多用来自定义其呈现和编辑的工具，同时提供了这些功能的默认设置，从而可以轻松地设置简单表。本实例使用其setRowHeight()方法来设置所有行的高度，该方法的声明如下：

public void setRowHeight(int rowHeight)

参数说明

rowHeight：新的行高。

对于表头，并没有直接修改其高度的方法。为此需要使用getTableHeader()方法获得JTableHeader对象。该方法的声明如下：

public JTableHeader getTableHeader()

然后使用从JComponent继承的setPreferredSize()方法来设置其大小，该方法的声明如下：

public void setPreferredSize(Dimension preferredSize)

参数说明

preferredSize：新的首选大小。

实现过程

（1）编写类ResizeTableTest，该类继承了JFrame。在框架中包含了两个表格，分别用来演示不同的设置方式，其构造方法的核心代码如下：

```
public ResizeTableTest() {
  //省略与自定义表格无关的其他代码
  table2 = new JTable();
  table2.setFont(newFont("微软雅
黑",Font.PLAIN,14));                    //修改表体的字体
  table2.setRowHeight(35);
      //修改表体的高度
```

```
    JTableHeader header=
table2.getTableHeader();                          //获得表头
    header.setFont(newFont("微软雅
黑",Font.PLAIN,16));                    //修改表头的字体
    header.setPreferredSize(newDimension(header.getWidth(),
40));          //修改表头的高度
    scrollPane2.setViewportView(table2);
          //显示表
  }
```

（2）编写方法do_this_windowActivated()，用来监听窗体激活事件。在该方法中，定义了一个默认的表格模型，并让两个表格同时使用这个表格模型。该方法的核心代码如下：

```
  protected void do_this_windowActivated(WindowEvent e) {
    DefaultTableModelmodel=newDefaultTableModel();
        //创建表格模型
    model.setRowCount(0);
      //将表格模型中的数据清空
    model.setColumnIdentifiers(newObject[] { "排名", "语言"
});            //设置表头
    model.addRow(newObject[] { "1", "Java"
});                          //增加行
    model.addRow(newObject[] { "2", "C"
});                            //增加行
    model.addRow(newObject[] { "3", "C#"
});                          //增加行
    table1.setModel(model);
        //为表格设置表格模型
```

```
    table2.setModel(model);
        //为表格设置表格模型
}
```

举一反三

根据本实例，读者可以开发以下程序。

设置用户是否可以通过在列头间拖动来调整各列的大小。

设置用户是否可以拖动列头。

实例077 调整表格各列的宽度

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\02\Ex02_77

实例说明

默认情况下，当用户调整一列的大小时，其他各列的大小也会随之改变，以便将表格的各列显示在框架中。为了适应不同的情况，表格提供了其他几个常量来设置表格列的变化方式。本实例将演示它们的用法。实例的运行效果如图2.33所示。

(a) 默认模式　　(b) 禁用模式

图2.33 实例运行效果

技术要点

当表格中一列的大小发生变化时，JTable类共提供了5种模式来调节其他列的变化。这些方式的说明如表2.16所示。

表2.16 表格常用域说明

域　名	作　用
AUTO_RESIZE_ALL_COLUMNS	在所有的调整大小操作中，按比例调整所有的列
AUTO_RESIZE_LAST_COLUMN	在所有的调整大小操作中，只对最后一列进行调整
AUTO_RESIZE_NEXT_COLUMN	在 UI 中调整了一个列时，对其下一列进行相反方向的调整
AUTO_RESIZE_OFF	不自动调整列的宽度；使用滚动条
AUTO_RESIZE_SUBSEQUENT_COLUMNS	在 UI 调整中，更改后续列以保持总宽度不变；此为默认行为

为了使用这些模式，需要使用setAutoResizeMode()方法来进行设置，该方法的声明如下：

public void setAutoResizeMode(int mode)

参数说明

mode：表2.16中的一个域。

实现过程

（1）编写方法do_this_windowActivated()，用来监听窗体激活事件。在该方法中，使用表格模型初始化了表格的数据。核心代码如下：

```
protected void do_this_windowActivated(WindowEvent e) {
  DefaultTableModel tableModel= (DefaultTableModel)
table.getModel();//获得表格模型
  tableModel.setRowCount(0);
                    //将表格模型中的数据清空
  tableModel.setColumnIdentifiers(new Object[] { "书名",
"出版社", "出版时间", "丛书类别", "定价" });//设置表头
  tableModel.addRow(new Object[] { "Java从入门到精通（第2
版）", "清华大学出版社", "2010-07-01", "软件工程师入门丛
书", "59.8元"
```

```
});
        //增加行
    tableModel.addRow(new Object[] { "PHP从入门到精通（第2
版）", "清华大学出版社", "2010-07-01", "软件工程师入门丛
书", "69.8元"
});
        //增加行
    tableModel.addRow(new Object[]{ "Visual Basic从入门到精
通（第2版）", "清华大学出版社", "2010-07-01", "软件工程师入
门丛书", "69.8
元"});
        //增加行
    tableModel.addRow(new Object[]{ "Visual C++从入门到精通
（第2版）", "清华大学出版社", "2010-07-01", "软件工程师入门
丛书", "69.8
元"});
        //增加行
    table.setModel(tableModel);
                    //更新表格模型
  }
```

（2）编写方法 do_comboBox_actionPerformed()，用来监听组合框动作事件。在该方法中，根据用户选择的组合框项来重新设置表格列的调整模式。核心代码如下：

```
   protected void do_comboBox_actionPerformed(ActionEvent e)
{
    //使用映射来保存组合框中字符串与表格调整模式之间的对应
  关系
```

```
    Map<String, Integer> columnModel = new HashMap<String,
Integer>();
    columnModel.put("AUTO_RESIZE_ALL_COLUMNS",
JTable.AUTO_RESIZE_ALL_COLUMNS);
    columnModel.put("AUTO_RESIZE_LAST_COLUMN",
JTable.AUTO_RESIZE_LAST_COLUMN);
    columnModel.put("AUTO_RESIZE_NEXT_COLUMN",
JTable.AUTO_RESIZE_NEXT_COLUMN);
    columnModel.put("AUTO_RESIZE_OFF",
JTable.AUTO_RESIZE_OFF);
    columnModel.put("AUTO_RESIZE_SUBSEQUENT_COLUMNS",
JTable.AUTO_RESIZE_SUBSEQUENT_COLUMNS);
    String text= (String)
comboBox.getSelectedItem();                                    //
获得用户的选择项
    table.setAutoResizeMode(columnModel.get(text));
                        //设置调整模式
  }
```

举一反三

根据本实例，读者可以开发以下程序。

按比例调整所有列。

调整一列的宽度时，同时调整其右侧各列的宽度。

实例078 设置表格的选择模式

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_78

实例说明

对于一个二维的表格而言，其选择模式有很多种。以行为例，可以选择一行、连续几行、任意几行，对于列同理。此外，对于单元格也可以设置成选择一个单元格、选择一个连续区域的单元格或选择一个不连续区域的单元格等。本实例将演示它们的用法。实例运行效果如图2.34所示。

(a) 单行选择　　(b) 多行选择

图2.34 实例运行效果

技术要点

利用选择模式，可以调整用户的选择方式。首先需要获得表格的模型，然后修改其选择模式。代码如下：

```
table.getSelectionModel().setSelectionMode(mode);
```

mode是ListSelectionModel接口定义的选择模式，其详细说明如表2.17所示。

表2.17 列表选择模式常量

常 量 名	作 用
MULTIPLE_INTERVAL_SELECTION	一次选择一个或多个连续的索引范围
SINGLE_INTERVAL_SELECTION	一次选择一个连续的索引范围
SINGLE_SELECTION	一次选择一个列表索引

实现过程

（1）编写类TableSelectModeTest，该类继承了JFrame。在框架中包含了一个表格、一个按钮组和一个复选框。按钮组包括“单行”

“连续多行”和“任意多行” 3 个单选按钮。通过单选按钮和复选框的组合，来实现不同的选择方式。

（2）编写方法do_this_windowActivated()，用来监听窗体激活事件。在该方法中，使用表格模型初始化了表格的数据。核心代码如下：

```
protected void do_this_windowActivated(WindowEvent e) {
  DefaultTableModel tableModel= (DefaultTableModel)
table.getModel();                    //获得表格模型
  tableModel.setRowCount(0);
                    //将表格模型中的数据清空
  tableModel.setColumnIdentifiers(newObject[] { "书名",
"出版社", "出版时间", "丛书类别", "定价" });  //设置表头
  tableModel.addRow(new Object[] { "Java从入门到精通（第2
版）", "清华大学出版社", "2010-07-01", "软件工程师入门丛
书", "59.8元"
});
    //增加行
  tableModel.addRow(new Object[] { "PHP从入门到精通（第2
版）", "清华大学出版社", "2010-07-01", "软件工程师入门丛
书", "69.8元"
});
    //增加行
  tableModel.addRow(new Object[] { "Visual Basic从入门到
精通（第2版）", "清华大学出版社", "2010-07-01", "软件工程师
入门丛书", "69.8元"
});
//增加行
```

```
        tableModel.addRow(newObject[] { "Visual C++从入门到精通
    （第2版）", "清华大学出版社", "2010-07-01", "软件工程师入门
    丛书", "69.8元"
    });
    //增加行
        table.setModel(tableModel);
                        //更新表格模型
    }
```

（3）编写方法do_rowRadioButton1_actionPerformed()，用来监听单击“单行”单选按钮事件。在该方法中，设置了表格行的选择模式是选择单行。核心代码如下：

```
    protected void
do_rowRadioButton1_actionPerformed(ActionEvent e) {
        table.getSelectionModel().setSelectionMode(ListSelectio
    nModel.SINGLE_SELECTION);
    }
```

（4）编写方法 do_checkBox_actionPerformed()，用来监听复选框选择事件。在该方法中，启动或禁用表格的列选择功能，并修改了复选框的文本。核心代码如下：

```
    protected void do_checkBox_actionPerformed(ActionEvent e)
{
        if (checkBox.isSelected()) {
            checkBox.setText("启动列选
    择");                                    //修改复选框的文本内容为
    “启动列选择”
            table.setColumnSelectionAllowed(true);
                  //启动列选择
```

```
    } else {
        checkBox.setText("禁用列选
择");                                    //修改复选框的文本内容为
“禁用列选择”
        table.setColumnSelectionAllowed(false);
                //禁止列选择
    }
}
```

举一反三

根据本实例，读者可以实现以下功能。

一次选择一个或多个连续的索引范围。

一次选择一个列表索引。

实例079 单元格的粗粒度排序

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_79

实例说明

在使用表格时，会出现根据不同的列来对表格进行排序的情况。例如，可以根据价格的不同来对各种书籍进行排序。本实例将演示如何实现简单的排序功能。实例运行效果如图2.35所示。

（a）升序排序

（b）降序排序

图2.35 实例运行效果

技术要点

默认情况下表格是不支持排序的。然而，使用setAutoCreateRowSorter()方法就可以让表格根据列来排序，该方法的声明如下：

public void setAutoCreateRowSorter(boolean autoCreateRowSorter)

参数说明

autoCreateRowSorter：是否应该自动创建 RowSorter。

实现过程

编写方法do_this_windowActivated()，用来监听窗体激活事件。在该方法中，初始化了表格中的数据并启动了排序功能。核心代码如下：

```
protected void do_this_windowActivated(WindowEvent e) {
  DefaultTableModel tableModel= (DefaultTableModel) table.getModel();                //获得表格模型
    tableModel.setRowCount(0);
                    //将表格模型中的数据清空
    tableModel.setColumnIdentifiers(newObject[] { "书名", "出版社", "出版时间", "丛书类别", "定价" });  //设置表头
    tableModel.addRow(new Object[] { "Java从入门到精通（第2版）", "清华大学出版社", "2010-07-01", "软件工程师入门丛书", "59.8元"
});
      //增加行
    tableModel.addRow(new Object[] { "PHP从入门到精通（第2版）", "清华大学出版社", "2010-07-01", "软件工程师入门丛
```

```
书", "69.8元"
});
        //增加行
    tableModel.addRow(new Object[]{ "Visual Basic从入门到精通（第2版）", "清华大学出版社", "2010-07-01", "软件工程师入门丛书", "69.8
元"});
        //增加行
    tableModel.addRow(new Object[]{ "Visual C++从入门到精通（第2版）", "清华大学出版社", "2010-07-01", "软件工程师入门丛书", "69.8
元"});
        //增加行
    table.setModel(tableModel);
                    //更新表格模型
    table.setAutoCreateRowSorter(true);
                        //启动排序功能
  }
```

举一反三

根据本实例，读者可以实现以下功能。

根据出版时间进行排序。

根据价格进行排序。

实例080 实现表格的查找功能

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\02\Ex02_80

实例说明

对于数据量较大的表格，通常会为用户提供查找功能。这可以让用户快速找到自己需要的行。如果没有符合用户需求的行则进行提示，节约用户的时间。本实例将演示如何在表格中实现这个功能。实例运行效果如图2.36所示。

（a）查找前　　（b）查找后

图2.36 实例运行效果

技术要点

RowFilter用于从模型中过滤条目，使得这些条目不会在视图中显示。例如，一个与JTable关联的 RowFilter 可能只允许包含带指定字符串的列的那些行。条目的含义取决于控件类型。例如，当过滤器与JTable关联时，一个条目对应于一行；当过滤器与JTree关联时，一个条目对应于一个节点。本实例使用其regexFilter()方法来实现文本过滤。该方法的声明如下：

public static <M,I> RowFilter<M,I> regexFilter(String regex,int... indices)

参数说明

● regex：在其上进行过滤的正则表达式。

● indices：要检查的值的索引如果没有提供，则计算所有的值。

实现过程

（1）编写类SearchTable，该类继承了JFrame。在框架中包含了一个文本域、一个表格和一个“查找”按钮。文本域用于获得用户输入的关键字，“查找”按钮用于在表格中查找用户输入的关键字。

（2）编写方法do_this_windowActivated()，用来监听窗体激活事件。在该方法中，初始化了表格的数据，并为表格设置了RowSorter。核心代码如下：

```
protected void do_this_windowActivated(WindowEvent e) {
  DefaultTableModel tableModel= (DefaultTableModel) table.getModel();    //获得表格模型
  tableModel.setRowCount(0);
      //将表格模型中的数据清空
  tableModel.setColumnIdentifiers(new Object[] { "书名", "出版社", "出版时间", "丛书类别", "定价" });  /设置表头
  tableModel.addRow(new Object[] { "Java从入门到精通（第2版）", "清华大学出版社", "2010-07-01", "软件工程师入门丛书", "59.8元"
});                                                                    //增加行
  tableModel.addRow(new Object[] { "PHP从入门到精通（第2版）", "清华大学出版社", "2010-07-01", "软件工程师入门丛书", "69.8元"
});                                                                    //增加行
  tableModel.addRow(new Object[]{ "Visual Basic从入门到精通（第2版）", "清华大学出版社", "2010-07-01", "软件工程师入门丛书", "69.8
```

```
元"});                                                    //增
加行
    tableModel.addRow(new Object[]{ "Visual C++从入门到精通
（第2版）", "清华大学出版社", "2010-07-01", "软件工程师入门
丛书", "69.8
元"});                                                    //增
加行
    sorter.setModel(tableModel);
        //为TableRowSorter对象增加表格模型
    table.setRowSorter(sorter);
        //设置RowSorter
  }
```

（3）编写方法do_button_actionPerformed()，用来监听单击“查找”按钮事件。在该方法中，使用用户在文本域输入的关键字过滤表格内容，核心代码如下：

```
  protected void do_button_actionPerformed(ActionEvent e) {
    sorter.setRowFilter(RowFilter.regexFilter(textField.get
Text()));          //实现过滤
  }
```

举一反三

根据本实例，读者可以开发以下程序。

进行模糊查询。

实现表格的查找功能。

实例081 在表格中应用组合框

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\02\Ex02_81

实例说明

对于商品的销售者而言，通常需要根据商品的销售情况来判断是否需要进货。当使用表格统计商品信息时，在一个单元格中使用组合框来设置商品的销售状态比提供代表不同状态的列让用户填写更加方便。本实例将演示如何在表格中实现这个功能。实例运行效果如图2.37所示。

图2.37 实例运行效果

技术要点

AbstractTableModel是一个抽象类，该类为TableModel接口中的大多数方法提供默认实现。它负责管理侦听器，并为生成TableModelEvents以及将其调度到侦听器提供方便。该类包含的抽象方法如表2.18所示。

表2.18

AbstractTableModel的抽象方法

方 法 名	作 用
getColumnCount()	返回该模型中的列数
getRowCount()	返回该模型中的行数
getValueAt(int rowIndex, int columnIndex)	返回 columnIndex 和 rowIndex 位置的单元格值

DefaultCellEditor是表单元格和树单元格的默认编辑器，本实例使用了组合框单元格，使用的构造方法如下：

```
public DefaultCellEditor(JCheckBox checkBox)
```

参数说明

checkBox：一个JCheckBox对象。

实现过程

（1）编写类ComboBoxTableModel，该类继承了AbstractTableModel。在该类中，实现了继承的抽象方法，并且重写了另外3个方法。代码如下：

```
    public class ComboBoxTableModel extends
AbstractTableModel {
      private static final long
  serialVersionUID=5523252281451951512L;          //定义序列
  化标识
      private staticString[] states= { "缺货", "需要进货",
  "不需要进货" };          //定义组合框的选项
      privateObject[][]data= { { "《Java从入门到精通（第2
  版）》", states[0] }, { "《PHP从入门到精通（第2版）》",
  states[1] },{ "《VisualC++从入门到精通（第2版）》",
  states[1] },{ "《VisualBasic从入门到精通（第2版）》",
  states[1] }, };          //用数组表示表格中的数据
      @Override
      public int getColumnCount() {
        return
    2;                                                //
    将表格的列数设置成两列
      }
      @Override
      public int getRowCount() {
```

```
        return
    data.length;                                          /
    /将表格的行数设置成数据的行数
      }
      @Override
      publicObject getValueAt(int rowIndex, int columnIndex)
{
        return data[rowIndex]
    [columnIndex];                                  //返回值是二
    维数组的对应值
      }
      @Override
      public String getColumnName(int column) {
        String[] names ={"书名", "状态"};
        return
    names[column];                                        /
    /设置表头
      }
      @Override
      publicboolean isCellEditable(int rowIndex, int
columnIndex) {
        return
    columnIndex==1;                                       //
    设置第二列可修改
      }
      @Override
```

```
    publicvoid setValueAt(Object aValue, int rowIndex, int
columnIndex) {
      data[rowIndex][columnIndex]=
  aValue;                                  //显示更新后的组合框
  内容
    }
    public static String[] getStates() {
      return
  states;                                                    //
  获得组合框的状态
    }
  }
```

（2）编写方法do_this_windowActivated()，用来监听窗体激活事件。在该方法中，为表格设置了自定义的表格模型并修改了列及其宽度。核心代码如下：

```
  protected void do_this_windowActivated(WindowEvent e) {
    ComboBoxTableModel
tableModel=newComboBoxTableModel();              //创建自定义
表格模型
    table.setModel(tableModel);
              //设置表格模型
    JComboBox comboBox=new
JComboBox(ComboBoxTableModel.getStates());    //创建组合框
对象
    comboBox.setFont(new Font("微软雅黑", Font.PLAIN, 14));
    DefaultCellEditor
editor=newDefaultCellEditor(comboBox);                  //利
```

用组合框创建单元格编辑器

```
    TableColumnModel columnModel=
table.getColumnModel();                    //获得表格的列模型
    columnModel.getColumn(1).setCellEditor(editor);
                    //设置第二列为组合框
    columnModel.getColumn(0).setPreferredWidth(250);
                    //设置第一列的宽度为250
    //设置第二列的宽度为
100columnModel.getColumn(1).setPreferredWidth(100);
  }
```

举一反三

根据本实例，读者可以开发以下程序。

在单元格中使用组合框。

增加组合框的选择项。

实例082 删除表格中选中的行

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\02\Ex02_82

实例说明

通常情况下，表格中的数据是经常变化的。因此，需要为用户提供增加行数据和删除行数据的功能。在表格中，实现这两种操作的代码类同。本实例将演示如何在表格中删除行数据，读者可以在本实例的基础上进行完善，实现增加行的功能。实例运行效果如图2.38所示。

（a）删除前　　（b）删除后

图2.38 实例运行效果

技术要点

为了删除表格中用户选择的行，需要获得用户选择行的索引。使用getSelectedRow()方法可以实现这个需求，该方法的声明如下：

```
public int getSelectedRow()
```

DefaultTableModel是TableModel的一个实现，它使用一个Vector来存储单元格的值对象，该Vector由多个Vector组成。在该类中提供了增加和删除表格中数据的常用方法，其说明如表2.19所示。

表2.19 DefaultTableModel的常用方法

方 法 名	作　用
addColumn(Object columnName)	将一列添加到模型中
addRow(Object[] rowData)	添加一行到模型的结尾
getColumnCount()	返回此数据表中的列数
getRowCount()	返回此数据表中的行数
removeRow(int row)	移除模型中 row 位置的行

实现过程

（1）编写类DeleteRows，该类继承了JFrame。在框架中包含了一个表格和一个“删除”按钮。

（2）编写方法do_this_windowActivated()，用来监听窗体激活事件。在该方法中，使用表格模型初始化了表格的数据。核心代码如下：

```
  protected void do_this_windowActivated(WindowEvent e) {
    DefaultTableModel tableModel= (DefaultTableModel)
table.getModel();            //获得表格模型
    tableModel.setRowCount(0);
                //将表格模型中的数据清空
    tableModel.setColumnIdentifiers(new Object[] { "书名",
"出版社", "出版时间", "丛书类别", "定价" });  //设置表头
    tableModel.addRow(new Object[] { "Java从入门到精通（第2
版）", "清华大学出版社", "2010-07-01", "软件工程师入门丛
书", "59.8元"
});
//增加行
    tableModel.addRow(new Object[] { "PHP从入门到精通（第2
版）", "清华大学出版社", "2010-07-01", "软件工程师入门丛
书", "69.8元"
});
//增加行
    tableModel.addRow(new Object[]{ "Visual Basic从入门到精
通（第2版）", "清华大学出版社", "2010-07-01", "软件工程师入
门丛书", "69.8
元"});
  //增加行
    tableModel.addRow(new Object[]{ "Visual C++从入门到精通
（第2版）", "清华大学出版社", "2010-07-01", "软件工程师入门
丛书", "69.8
元"});
  //增加行
```

```
    table.setModel(tableModel);
                //更新表格模型
  }
```

（3）编写方法do_button_actionPerformed()，用来监听单击“删除”按钮事件。在该方法中，使用表格模型删除表格中的数据。核心代码如下：

```
  protected void do_button_actionPerformed(ActionEvent e) {
    DefaultTableModelmodel= (DefaultTableModel)
table.getModel();        //获得表格模型
    int index=
table.getSelectedRow();                                    //获
得用户选择的索引
    if (index== -1)
{                                                  //如果用户没有
选择任何行则进行提示
      JOptionPane.showMessageDialog(this, "请选择要删除的
行", "",JOptionPane.WARNING_MESSAGE);
      return;
    }
    model.removeRow(table.getSelectedRow());
          //删除用户选择的行
    table.setModel(model);
        //重新设置表格模型
  }
```

举一反三

根据本实例，读者可以开发以下程序。

实现删除表格中选择的列、向表格中增加一行数据。

向表格中增加一列数据。

删除数据时弹出提示是否删除窗口。

实例083 实现表格的分页技术

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\02\Ex02_83

实例说明

对于数据量比较大的表格而言，为了用户浏览方便会使用分页技术。对于 Java EE 程序员而言，有很多工具可以帮忙实现分页，如Hibernate等。另外，也可以在查询数据库中的数据时使用分页。本实例将自行实现一个分页算法，实例运行效果如图2.39所示。

(a) 首页

(b) 末页

图2.39 实例运行效果

技术要点

由于表格模型中的数据不方便截取，因此使用 getDataVector()方法将表格模型中的数据存储到向量中，然后操作向量中的数据。该方法的声明如下：

```
public Vector getDataVector()
```

为了获得表格模型中的总数据数，使用了getRowCount()方法，该方法的声明如下：

```
public int getRowCount()
```

在获得了总行数和每页的行数之后，就可以计算最大页数了。需要注意的是，Java中的整数除法是直接截取而不会进位的，如9/5的结果是1。因此总行数如果不是每页行数的整数倍，则需要加1。

实现过程

（1）编写类 PageTable，该类继承了 JFrame。在框架中包含了一个表格及“首页”、“前一页”、“后一页”和“末页”4个按钮。

（2）编写方法do_this_windowActivated()，用来监听窗体激活事件。在该方法中，使用表格模型初始化表格中的数据，计算了总页数，并设置了按钮的初始状态。核心代码如下：

```
protected void do_this_windowActivated(WindowEvent e) {
  defaultModel= (DefaultTableModel)
table.getModel();                          //获得表格模型
  defaultModel.setRowCount(0);
            //清空表格模型中的数据
  defaultModel.setColumnIdentifiers(newObject[] { "序号",
"平方数" });          //定义表头
  for (int i = 0; i < 23; i++) {
    defaultModel.addRow(newObject[] { i, i* i
});                          //向表格模型中增加数据
  }
  maxPageNumber=
(int)Math.ceil(defaultModel.getRowCount() /
pageSize);      //计算总页数
  table.setModel(defaultModel);
            //设置表格模型
  firstPageButton.setEnabled(false);
                //禁用“首页”按钮
```

```
        latePageButton.setEnabled(false);
                    //禁用“前一页”按钮
        nextPageButton.setEnabled(true);
                    //启用“后一页”按钮
        lastPageButton.setEnabled(true);
                    //启用“末页”按钮
    }
```

（3）编写方法 do_firstPageButton_actionPerformed()，用来监听单击“首页”按钮事件。在该方法中，创建了一个新的表格模型来保存原表格模型中的首页数据。核心代码如下：

```
    protected void
do_firstPageButton_actionPerformed(ActionEvent e) {
        currentPageNumber=1;
              //将当前页码设置成1
        Vector
  dataVector=defaultModel.getDataVector();
            //获得原表格模型中的数据
        DefaultTableModel
  newModel=newDefaultTableModel();                    //创建新
  的表格模型
        newModel.setColumnIdentifiers(newObject[] { "序号", "随
  机数" });              //定义表头
        for (int i = 0; i < pageSize; i++) {
          newModel.addRow((Vector)
    dataVector.elementAt(i));                         //根据页面大
    小来获得数据
        }
```

```
        table.setModel(newModel);
                //设置表格模型
        firstPageButton.setEnabled(false);
                    //禁用“首页”按钮
        latePageButton.setEnabled(false);
                    //禁用“前一页”按钮
        nextPageButton.setEnabled(true);
                    //启用“后一页”按钮
        lastPageButton.setEnabled(true);
                    //启用“末页”按钮
    }
```

（4）编写方法 do_latePageButton_actionPerformed()，用来监听单击“前一页”按钮事件。在该方法中，创建了一个新的表格模型来保存原表格模型中的前一页数据。核心代码如下：

```
    protected void
do_latePageButton_actionPerformed(ActionEvent e) {
        currentPageNumber-
    -;                                                    //将当前页
    面减1
        Vector
    dataVector=defaultModel.getDataVector();
            //获得原表格模型中的数据
        DefaultTableModel
    newModel=newDefaultTableModel();                      //创建新
    的表格模型
        newModel.setColumnIdentifiers(newObject[] { "序号", "随
    机数" });              //定义表头
```

```
    for (int i = 0; i < pageSize; i++) {
      newModel.addRow((Vector) dataVector.elementAt((int)
  (pageSize* (currentPageNumber - 1)+ i)));
      //根据页面大小来获得数据
    }
    table.setModel(newModel);
          //设置表格模型
    if (currentPageNumber == 1) {
      firstPageButton.setEnabled(false);
            //禁用“首页”按钮
      latePageButton.setEnabled(false);
          //禁用“前一页”按钮
    }
    nextPageButton.setEnabled(true);
          //启用“后一页”按钮
    lastPageButton.setEnabled(true);
          //启用“末页”按钮
  }
```

（5）编写方法do_nextPageButton_actionPerformed()，用来监听单击“后一页”按钮事件。在该方法中，创建了一个新的表格模型来保存原表格模型中的后一页数据。核心代码如下：

```
  protected void
do_nextPageButton_actionPerformed(ActionEvent e) {
    currentPageNumber++;
          //将当前页面加1
    Vector
  dataVector=defaultModel.getDataVector();
```

```
        //获得原表格模型中的数据
    DefaultTableModel
newModel=newDefaultTableModel();                    //创建新
的表格模型
    newModel.setColumnIdentifiers(newObject[] { "序号", "随
机数" });              //定义表头
    if (currentPageNumber == maxPageNumber) {
      int lastPageSize= (int) (defaultModel.getRowCount() -
  pageSize* (maxPageNumber - 1));
      for (int i = 0; i < lastPageSize; i++) {
        newModel.addRow((Vector) dataVector.elementAt((int)
    (pageSize* (maxPageNumber - 1)+ i)));
        //根据页面大小来获得数据
      }
      nextPageButton.setEnabled(false);
                //禁用“后一页”按钮
      lastPageButton.setEnabled(false);
                //禁用“末页”按钮
    } else {
      for (int i = 0; i < pageSize; i++) {
        newModel.addRow((Vector) dataVector.elementAt((int)
    (pageSize* (currentPageNumber - 1)+ i)));
        //根据页面大小来获得数据
      }
    }
    table.setModel(newModel);
          //设置表格模型
```

```
        firstPageButton.setEnabled(true);
                    //启用“首页”按钮
        latePageButton.setEnabled(true);
                    //启用“前一页”按钮
    }
```

（6）编写方法 do_lastPageButton_actionPerformed()，用来监听单击“末页”按钮事件。在该方法中，创建了一个新的表格模型来保存原表格模型中的末页数据。核心代码如下：

```
    protected void
do_lastPageButton_actionPerformed(ActionEvent e) {
        currentPageNumber=maxPageNumber;
              //将当前页面设置为末页
        Vector
  dataVector=defaultModel.getDataVector();
            //获得原表格模型中的数据
        DefaultTableModel
  newModel=newDefaultTableModel();                    //创建新
  的表格模型
        newModel.setColumnIdentifiers(newObject[] { "序号", "随
  机数" });              //定义表头
        int lastPageSize= (int) (defaultModel.getRowCount() -
  pageSize* (maxPageNumber - 1));
        if (lastPageSize == 5) {
          for (int i = 0; i < pageSize; i++) {
            newModel.addRow((Vector) dataVector.elementAt((int)
        (pageSize* (maxPageNumber - 1)+ i)));
            //根据页面大小来获得数据
```

```
    }
  } else {
    for (int i = 0; i < lastPageSize; i++) {
      newModel.addRow((Vector) dataVector.elementAt((int)
  (pageSize* (maxPageNumber - 1)+ i)));
      //根据页面大小来获得数据
    }
  }
  table.setModel(newModel);
          //设置表格模型
  firstPageButton.setEnabled(true);
              //启用“首页”按钮
  latePageButton.setEnabled(true);
              //启用“前一页”按钮
  nextPageButton.setEnabled(false);
              //禁用“后一页”按钮
  //禁用“末页”按钮lastPageButton.setEnabled(false);
}
```

举一反三

根据本实例，读者可以开发以下程序。

对每页显示的条数进行限制。

实现分页功能。

实例084 为单元格绘制背景色

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_84

实例说明

如果程序的界面中只有黑白两种颜色，则会让用户感觉界面很不好看。通常可以使用各种颜色的搭配来美化界面。本实例使用表格单元格渲染工具来为不同的单元格设置不同的颜色。读者可以从中选择自己喜欢的颜色用在自己的程序中。实例运行效果如图2.40所示。

图2.40 实例运行效果

技术要点

TableCellRenderer 接口定义了一个名为getTableCellRendererComponent()的方法，该方法可以用来在渲染单元格前配置渲染器。该方法的声明如下：

Component getTableCellRendererComponent(JTable table,Object value,boolean isSelected,boolean hasFocus,int row,int column)

该方法中各个参数的说明如表2.20所示。

表2.20 getTableCellRendererComponent()方法的参数说明

参 数 名	作 用
table	要求渲染器渲染的 JTable，可以为 null
value	要呈现的单元格的值
isSelected	是否使用选中样式的高亮显示来呈现该单元格
hasFocus	单元格是否具有焦点
row	要渲染的单元格的行索引
column	要渲染的单元格的列索引

在编写好渲染器之后，可以使用 setDefaultRenderer()方法为表格设置渲染器，该方法的声明如下：

public void setDefaultRenderer(Class<?> columnClass,TableCellRenderer renderer)

参数说明

● columnClass：设置此columnClass 的默认单元格渲染器。

● renderer：此columnClass 要使用的默认单元格渲染器。

实现过程

（1）编写类ColorTableCellRenderer，该类继承了JPanel，并实现了TableCellRenderer接口。在实现的方法中，为每个单元格设置了不同的背景色。代码如下：

```
public class ColorTableCellRenderer extends JPanel
implements TableCellRenderer {
    private static final long serialVersionUID =
  8932176536826008653L;
    @Override
    public Component getTableCellRendererComponent(JTable
  table, Object value, boolean isSelected, boolean hasFocus,
  int row, int column) {
      int
  times=50;                                          //设置
  背景色与行列索引的倍数关系
```

```
        int r= row* times%
    255;                                    //设置r值，代表红色
        int g= column * times%
    255;                                   //设置g值，代表绿色
        int b= (row+ column) * times%
    255;                               //设置b值，代表蓝色
        setBackground(newColor(r,g,b));
          //设置新的背景颜色
        return this;
      }
    }
```

（2）编写方法do_this_windowActivated()，用来监听窗体激活事件。在该方法中，设置了表格为5行5列。核心代码如下：

```
    protected void do_this_windowActivated(WindowEvent e) {
      DefaultTableModel model= (DefaultTableModel)
  table.getModel();         //获得默认表格模型
      model.setColumnCount(5);
          //设置列数
      model.setRowCount(5);
        //设置行数
    table.setModel(model);
            //更新表格模型
      table.setDefaultRenderer(Object.class,new
  ColorTableCellRenderer());       //设置渲染器
    }
```

举一反三

根据本实例，读者可以实现以下功能。

为单元格绘制两色交替的背景色。

当鼠标悬浮时单元格变成其他背景色。

实例085 实现表格的栅栏效果

这是一个可以启发思维的实例

实例位置：光盘\mingrisoft\02\Ex02_85

实例说明

对于长时间使用电脑的用户来说，如果表格中各行的颜色相同是很累眼睛的。为了让用户获得更舒适的体验，通常将表格设置成栅栏效果，即奇数行和偶数行的背景颜色是不同的。本实例将演示如何实现该效果。实例运行效果如图2.41所示。

图2.41 实例运行效果

技术要点

DefaultTableCellRenderer呈现（显示）JTable中每个单元格的标准类。它是TableCellRenderer接口的实现类，通常可以使用该类来简化渲染器编程。此类继承自一个标准的控件类JLabel。但是JTable

为其单元格的呈现使用了特殊的机制，因此要求稍微对其单元格渲染器的行为进行修改。在使用该类时，通常将其转型为JLabel，并调用getTableCellRendererComponent()方法来显示单元格中的文字。

实现过程

（1）编写类 FenseRenderer，该类实现了 TableCellRenderer 接口。在该接口的 getTableCell RendererComponent()方法中，为不同的行设置了不同的背景颜色和文本颜色。代码如下：

```
public class FenseRenderer implements TableCellRenderer {
  @Override
  public Component getTableCellRendererComponent(JTable
table, Object value, boolean isSelected, boolean hasFocus,
int row, int column) {
    JLabel renderer = (JLabel) new
DefaultTableCellRenderer().getTableCellRendererComponent(
table, value, isSelected, hasFocus, row, column);
    if (row%2==0)
{                                                    //偶数
行
      renderer.setForeground(Color.WHITE);
                  //将文本设置成白色
      renderer.setBackground(Color.BLUE);
                  //将背景设置成蓝色
    } else
{
  //奇数行
      renderer.setForeground(Color.BLUE);
                  //将文本设置成蓝色
```

```
            renderer.setBackground(Color.WHITE);
                                //将背景设置成白色
        }
          return renderer;
        }
    }
```

（2）编写方法do_this_windowActivated()，用来监听窗体激活事件。在该方法中，初始化了表格的数据并设置了新的渲染器。核心代码如下：

```
    protected void do_this_windowActivated(WindowEvent e) {
      DefaultTableModelmodel= (DefaultTableModel)
  table.getModel();                    //获得表格模型
      model.setRowCount(0);
                    //清空表格中的数据
      model.setColumnIdentifiers(new Object[] { "书名", "出版
  社", "出版时间", "丛书类别", "定价" });    //增加一行数据
      model.addRow(new Object[] { "Java从入门到精通（第2
  版）", "清华大学出版社", "2010-07-01", "软件工程师入门丛
  书", "59.8元"
  });
              //增加一行数据
      model.addRow(new Object[] { "PHP从入门到精通（第2版）",
  "清华大学出版社", "2010-07-01", "软件工程师入门丛书","69.8
  元"
  });
              //增加一行数据
```

```
        model.addRow(new Object[] { "Visual Basic从入门到精通（第2版）", "清华大学出版社", "2010-07-01", "软件工程师入门丛书", "69.8元"
    });
        //增加一行数据
        model.addRow(new Object[] { "Visual C++从入门到精通（第2版）", "清华大学出版社", "2010-07-01", "软件工程师入门丛书", "69.8元"
    });
        //增加一行数据
        table.setModel(model);
                        //设置表格模型
        table.setDefaultRenderer(Object.class,newFenseRenderer(
    ));                          //设置新的渲染器
    }
```

举一反三

根据本实例，读者可以开发以下程序。

实现栅栏效果。

鼠标悬浮后突出显示单元格。

实例086 单元格的细粒度排序

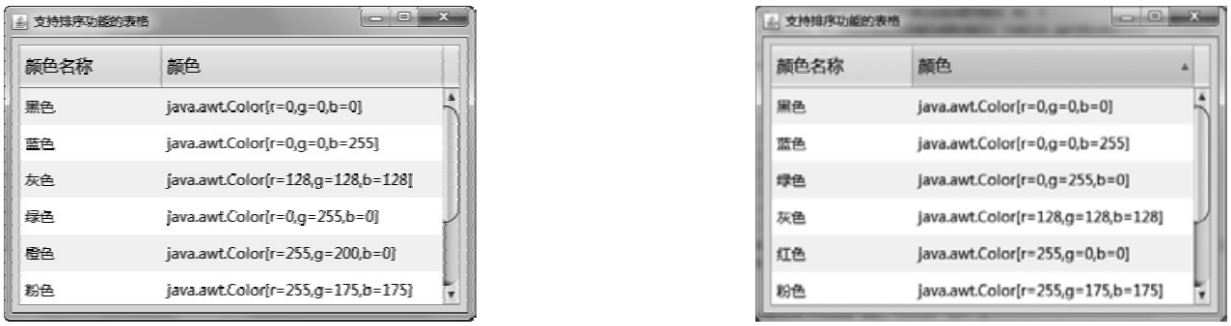
本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\02\Ex02_86

实例说明

除了可以使用表格的默认排序机制外，还可以为表格中的各行指定如何排序。例如，本实例中是根据红、绿、蓝三原色在具体颜色中

值的大小来排序的。先按红色升序排列，红色值相同则比较绿色，依次类推。实例运行效果如图2.42所示。



(a) 排序前　　(b) 排序后

图2.42 实例运行效果

技术要点

TableRowSorter<M extends TableModel>是RowSorter 的一个实现，它使用TableModel 提供排序和过滤操作。使用该类可以为每个列设置不同的比较方式。在使用该类时，需要用表格模型对该类进行实例化，使用的构造方法声明如下：

public TableRowSorter(M model)

参数说明

model：要使用的底层模型，或者 null。

为了加载自定义的 Comparable 接口实现类，需要使用方法setComparator()，该方法是从DefaultRowSorter继承而来的，其声明如下：

public void setComparator(int column,Comparator<?> comparator)

参数说明

● column：要应用Comparator 的列的索引（就底层模型而言）。

● comparator：要使用的Comparator。

实现过程

编写方法do_this_windowActivated()，用来监听窗体激活事件。在该方法中，使用表格模型初始化表格中的数据，并设置了颜色列的排序方式。核心代码如下：

```
protected void do_this_windowActivated(WindowEvent e) {
    DefaultTableModelmodel= (DefaultTableModel)
table.getModel();              //获得表格模型
    model.setRowCount(0);
          //清空表格中的数据
    model.setColumnIdentifiers(newObject[] { "颜色名称",
"颜色" });              //设置表头
    model.addRow(newObject[] { "黑色",Color.BLACK
});                    //为表格增加一行数据
    model.addRow(newObject[] { "蓝色",Color.BLUE
});                      //为表格增加一行数据
    model.addRow(newObject[] { "灰色",Color.GRAY
});                      //为表格增加一行数据
    model.addRow(newObject[] { "绿色",Color.GREEN
});                      //为表格增加一行数据
    model.addRow(newObject[] { "橙色",Color.ORANGE
});                      //为表格增加一行数据
    model.addRow(newObject[] { "粉色",Color.PINK
});                      //为表格增加一行数据
    model.addRow(newObject[] { "红色",Color.RED
});                      //为表格增加一行数据
    model.addRow(newObject[] { "白色",Color.WHITE
});                      //为表格增加一行数据
```

```
    model.addRow(newObject[] { "黄
色",Color.YELLOW});                          //为表格增加一行数据
    TableRowSorter<TableModel>
sorter=newTableRowSorter<TableModel>(model);
    sorter.setComparator(1,newComparator<Color>()
{                              //为第二列设置排序器
      @Override
      public int compare(Color o1,Color o2)
  {                                  //设置排序方式
        int r = o1.getRed() - o2.getRed();
        int g = o1.getGreen() - o2.getGreen();
        int b = o1.getBlue() - o2.getBlue();
        if (r !=0)
    {                                                  //首先按
    红色值排序
          return r;
        } else if (g !=0)
    {                                                  //其次按绿色
    值排序
          return g;
        } else {
          return
      b;                                                //最后
      按蓝色值排序
        }
      }
    });
```

```
    table.setRowSorter(sorter);
                //为表格增加排序器
    table.getColumnModel().getColumn(0).setPreferredWidth(1
50);                    //设置第一列的宽度为150
    table.getColumnModel().getColumn(1).setPreferredWidth(3
00);                    //设置第二列的宽度为300
  }
```

举一反三

根据本实例，读者可以开发以下程序。

按红色降序排列。

按绿色升序排列。

2.7 树控件的应用

实例087 编写中国省市信息树

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\02\Ex02_87

实例说明

对于具有层次关系的结构，使用树控件描述是非常方便的，如文件夹及其子文件夹之间的关系、国家的行政结构关系等。本实例将使用树控件来表示中国的各个行政区域。使用 Swing库中定义的工具类可以非常容易地实现。实例运行效果如图2.43所示。

图2.43 实例运行效果

技术要点

DefaultMutableTreeNode是树数据结构中的通用节点。一个树节点最多可以有一个父节点、0 或多个子节点。DefaultMutableTreeNode 为检查和修改节点的父节点和子节点提供操作，也为检查节点所属的树提供操作。节点的树是所有节点的集合，通过从某一节点开始并沿着父节点和子节点的所有可能的链接，可以访问这些节点。可以使用其含有参数的构造方法在创建节点对象时定义节点的内容，该方法的声明如下：

public DefaultMutableTreeNode(Object userObject)

参数说明

userObject：用户提供的 Object，它构成节点的数据。

使用add()方法为一个节点增加子节点就可以实现层次关系，该方法的声明如下：

public void add(MutableTreeNode newChild)

参数说明

newChild：作为此节点的子节点添加的节点。

实现过程

（1）编写类ChinaGeographyTree，该类继承了JFrame。在框架中包含了一棵树，在树中显示了中国的直辖市、省、自治区和特别行政区信息。

（2）编写方法do_this_windowActivated()，用来监听窗体激活事件。在该方法中，为树控件增加节点信息。核心代码如下：

```
protected void do_this_windowActivated(WindowEvent e) {
    DefaultMutableTreeNode
root=newDefaultMutableTreeNode("中国");          //创建根节点
    DefaultMutableTreeNodemunicipalities=newDefaultMutableTreeNode("直辖市");
    municipalities.add(new DefaultMutableTreeNode("北京"));          //为“直辖市”增加子节点“北京”
    municipalities.add(new DefaultMutableTreeNode("上海"));          //为“直辖市”增加子节点“上海”
    municipalities.add(new DefaultMutableTreeNode("天津"));          //为“直辖市”增加子节点“天津”
```

```
        municipalities.add(new DefaultMutableTreeNode("重
庆"));                    //为“直辖市”增加子节点“重庆”
        //省略其他节点的信息
        root.add(municipalities);
                    //为根节点增加“直辖市”节点
    root.add(province);
          //为根节点增加“省”节点
    root.add(ARegion);
          //为根节点增加“自治区”节点
    root.add(SARegion);
        //为根节点增加“特别行政区”节点
    DefaultTreeModel model=new
DefaultTreeModel(root);                    //利用根节点创建树
模型
    tree.setModel(model);
          //为树设置新的树模型
    }
```

举一反三

根据本实例，读者可以开发以下程序。

编写区县街道信息树。

编写商品种类信息树。

实例088 自定义树节点的图标

这是一个可以提高分析能力的实例

实例位置：光盘\mingrisoft\02\Ex02_88

实例说明

树默认的节点图标并不好看，为了美化界面，通常需要修改图标。有两种方式可以实现这个效果：第一是使用树渲染器，第二是使用UIManager类。由于第二种方法比较简单，本实例将使用该方法来改变树的图标。实例运行效果如图2.44所示。

图2.44 实例运行效果

技术要点

UIManager 管理当前外观、可用外观集合、外观更改时被通知的PropertyChangeListeners、外观默认值以及获取各种默认值的便捷方法。在树被绘制之前，可以使用put()方法将树节点不同状态的图标存入到系统中。这样当需要绘制树时，就会使用这些图标。put()方法的声明如下：

```
public static Object put(Object key,Object value)
```

参数说明

- key：一个指定检索键的Object。
- value：要存储的Object。

对于树节点，可以设置的图标有5种，详细说明如表2.21所示。

表2.21 树节点常用图标关键字

方 法 名	作 用
Tree.openIcon	树打开图标
Tree.closedIcon	树关闭图标
Tree.leafIcon	树叶子图标
Tree.expandedIcon	树展开图标
Tree.collaspedIcon	树合并图标

实现过程

（1）编写方法do_this_windowActivated()，用来监听窗体激活事件。在该方法中，为树增加了数据。核心代码如下：

```
protected void do_this_windowActivated(WindowEvent e) {
    DefaultMutableTreeNode root=new
DefaultMutableTreeNode("明日科技新书");          //设置根节点
    DefaultMutableTreeNode parent1=new
DefaultMutableTreeNode("从入门到精通系列");
    parent1.add(new DefaultMutableTreeNode("《Java从入门到
精通（第2版）》"));
    parent1.add(new DefaultMutableTreeNode("《PHP从入门到精
通（第2版）》"));
    parent1.add(new DefaultMutableTreeNode("《VisualBasic从
入门到精通（第2版）》"));
    parent1.add(new DefaultMutableTreeNode("《VisualC++从入
门到精通（第2版）》"));
    root.add(parent1);
                    //增加子节点
    DefaultMutableTreeNode parent2=new
DefaultMutableTreeNode("编程词典系列");
    parent2.add(new DefaultMutableTreeNode("《Java编程词
典》"));
    parent2.add(new DefaultMutableTreeNode("《PHP编程词
典》"));
    parent2.add(new DefaultMutableTreeNode("《Visual Basic
编程词典》"));
```

```
        parent2.add(new DefaultMutableTreeNode("《Visual C++编
    程词典》"));
        root.add(parent2);
                        //增加子节点
        DefaultTreeModelmodel=new
    DefaultTreeModel(root);                          //使用根节
    点创建默认树模型
        tree.setModel(model);
                        //更新树模型
      }
```

（2）编写main()方法，在该方法中，首先更改了树的图标和Swing的外观，然后运行了该程序。核心代码如下：

```
      public static void main(String[] args) {
        UIManager.put("Tree.openIcon",new
    ImageIcon("src/image/open.png"));                //设置节点打
    开图标
        UIManager.put("Tree.closedIcon",new
    ImageIcon("src/image/closed.png"));              //设置关闭图
    标
        UIManager.put("Tree.leafIcon",new
    ImageIcon("src/image/leaf.png"));                //设置子节
    点的图标
        try
    {
          //修改Swing的外观为Nimbus
          UIManager.setLookAndFeel("com.sun.java.swing.plaf.nim
      bus.NimbusLookAndFeel");
```

```
    } catch (Throwable e) {
      e.printStackTrace();
    }
    EventQueue.invokeLater(newRunnable()
{                                    //在事件派发线程中运行Swing
程序
      public void run() {
        try {
          NodeIcon frame = new NodeIcon();
          frame.setVisible(true);
        } catch (Exception e) {
          e.printStackTrace();
        }
      }
    });
  }
```

举一反三

根据本实例，读者可以开发以下程序。

使用UIManager修改树的字体。

修改菜单的字体。

实例089 监听节点的选择事件

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_89

实例说明

通常情况下，树控件是与其他控件一同使用的。当选择了树控件的一个节点后，会触发其他控件状态的改变事件。本实例将监听节点的选择事件，当用户选择不同的节点时，会在右侧的文本区中显示该节点的一些信息。实例运行效果如图2.45所示。

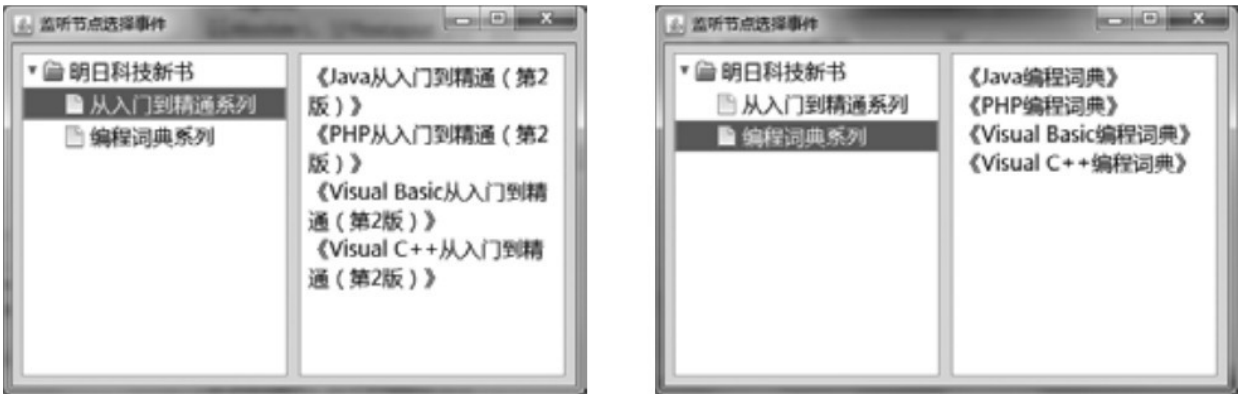


图2.45 实例运行效果

技术要点

为了能够监听用户选择树节点事件，必须为树添加选择监听器。该监听器必须实现TreeSelectionModel接口。在这个接口中定义了一个valueChanged()方法，该方法用于对用户选择的不同节点进行响应，其声明如下：

```
void valueChanged(TreeSelectionEvent e)
```

参数说明

e：表现更改的特征的事件。

为了对不同的节点选择事件做出不同的响应，需要知道用户选择了哪个节点。为此先使用getSelectionPath()方法获得用户选择的路径，该方法的声明如下：

```
public TreePath getSelectionPath()
```

然后使用getLastPathComponent()方法获得用户选择的节点，该方法的声明如下：

```
public Object getLastPathComponent()
```

实现过程

（1）编写类SelectedEventTest，该类继承了JFrame。在框架中包含了一棵树和一个文本区。文本区用来响应用户选择树节点的事件。

（2）编写方法do_this_windowActivated()，用来监听窗体激活事件。在该方法中，初始化了树中的节点信息。核心代码如下：

```
protected void do_this_windowActivated(WindowEvent e) {
  DefaultMutableTreeNode root=new
DefaultMutableTreeNode("明日科技新书");  //创建根节点
  DefaultMutableTreeNode parent1=new
DefaultMutableTreeNode("从入门到精通系列");
  root.add(parent1);
            //增加子节点
  DefaultMutableTreeNode parent2=new
DefaultMutableTreeNode("编程词典系列");
  root.add(parent2);
            //增加子节点
  DefaultTreeModel model=new
DefaultTreeModel(root);                    //使用根节点创建
默认树模型
  tree.setModel(model);
          //更新树模型
}
```

（3）编写方法do_tree_valueChanged()，用来监听用户选择不同的树节点事件。在该方法中，根据用户选择的节点更新了文本区内容。核心代码如下：

```
protected void do_tree_valueChanged(TreeSelectionEvent e)
{
```

```
    TreePath path=
tree.getSelectionPath();                                        //
//获得用户选择的节点路径
    if(path==null)
{                                                               //如果没
有选择节点则直接返回
      return;
    }
    DefaultMutableTreeNodenode=
(DefaultMutableTreeNode)path.getLastPathComponent();
    String text1= "《Java从入门到精通（第2版）》\n《PHP从入
门到精通（第2版）》\n《VisualBasic从入门到精通（第2版）》
\n《VisualC++从入门到精通（第2版）》";
    String text2= "《Java编程词典》\n《PHP编程词典》
\n《VisualBasic编程词典》\n《VisualC++编程词典》";
    if (node.toString().equals("从入门到精通系列"))
{                       //如果选择了“从入门到精通系列”节点
      textArea.setText(text1);
          //将文本区设置成text1
    } else {
      textArea.setText(text2);
          //将文本区设置为text2
    }
  }
```

举一反三

根据本实例，读者可以实现以下功能。

使用的树支持多选。

使用的树支持单选。

实例090 设置树控件选择模式

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_90

实例说明

对于树控件而言，其选择模式有：只能选择一个节点、可以选择连续若干节点和可以选择若干不连续的节点3种。本实例将演示它们的用法。读者可以通过选中单选按钮来选择树的选择模式。实例运行效果如图2.46所示。

（a）单行选择　　（b）多行选择

图2.46 实例运行效果

技术要点

树控件的选择模式是使用树模型来设置的，为此需要先获得树模型，使用 getModel()方法可以获得当前树的模型，该方法的声明如下：

```
public TreeModel getModel()
```

TreeSelectionModel接口表示树选择控件的当前状态。在该接口中定义了3个常量来表示树选择的模式，其说明如表2.22所示。

表2.22 TreeSelectionModel的常量

常 量 名	作 用
CONTIGUOUS_TREE_SELECTION	选择只能是连续的
DISCONTIGUOUS_TREE_SELECTION	选择可以包含任何数量的项，这些项不必是连续的
SINGLE_TREE_SELECTION	一次只能选择一个路径

实现过程

（1）编写类 TreeSelectModeTest，该类继承了 JFrame。在框架中包含了一棵树和一个单选按钮组。按钮组中包含 “单行”、“连续多行”和“任意多行”3 个单选按钮。

（2）编写方法do_this_windowActivated()，用来监听窗体激活事件。在该方法中，为树增加了数据。核心代码如下：

```
protected void do_this_windowActivated(WindowEvent e) {
    DefaultMutableTreeNode
root=newDefaultMutableTreeNode("明日科技新书");          //设置根节点
    DefaultMutableTreeNode parent1 = new
DefaultMutableTreeNode("从入门到精通系列");
    parent1.add(new DefaultMutableTreeNode("《Java从入门到精通（第2版）》"));
    parent1.add(new DefaultMutableTreeNode("《PHP从入门到精通（第2版）》"));
    parent1.add(new DefaultMutableTreeNode("《Visual Basic从入门到精通（第2版）》"));
    parent1.add(new DefaultMutableTreeNode("《Visual C++从入门到精通（第2版）》"));
    root.add(parent1);
                    //增加子节点
    DefaultMutableTreeNode parent2 = new
DefaultMutableTreeNode("编程词典系列");
```

```
        parent2.add(new DefaultMutableTreeNode("《Java编程词
    典》"));
        parent2.add(new DefaultMutableTreeNode("《PHP编程词
    典》"));
        parent2.add(new DefaultMutableTreeNode("《Visual Basic
    编程词典》"));
        parent2.add(new DefaultMutableTreeNode("《Visual C++编
    程词典》"));
        root.add(parent2);
                        //增加子节点
        DefaultTreeModelmodel=newDefaultTreeModel(root);
                          //使用根节点创建默认树模型
        tree.setModel(model);
                        //更新树模型
    }
```

（3）编写方法 do_radioButton1_actionPerformed()，用来监听单击“单行”单选按钮事件。在该方法中，设置了树控件的选择模式是选择单行。核心代码如下：

```
    protected void
do_radioButton1_actionPerformed(ActionEvent e) {
        tree.getSelectionModel().setSelectionMode(TreeSelection
    Model.SINGLE_TREE_SELECTION);
    }
```

举一反三

根据本实例，读者可以实现以下功能。

选择一个节点的树形控件。

选择连续若干节点的树形控件。

实例091 在树控件中增加节点

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_91

实例说明

除了让用户在给定的节点中进行选择外，还可以让用户在需要的位置添加自定义节点。这样程序会有更好的交互效果。本实例将实现让用户添加可以编辑的节点的功能，还可以为新的节点添加子节点。实例运行效果如图2.47所示。

（a）增加节点前

（b）增加节点后

图2.47 实例运行效果

技术要点

DefaultTreeModel是使用TreeNodes的简单树数据模型。在该类中定义了各种与树控件数据相关的方法，如节点的增加、修改、删除等。此外，还有一些方法可以通知树控件其中的数据已经被修改。本实例使用了向树控件增添节点的方法 insertNodeInto，该方法的声明如下：

```
public void insertNodeInto(MutableTreeNode
newChild,MutableTreeNode parent,int index)
```

参数说明

- newChild：要增添的新节点。

- parent：新节点的父节点。
- index：新增添的节点在父节点中的位置。

实现过程

（1）编写方法do_this_windowActivated()，用来监听窗体激活事件。在该方法中，为树增加了数据。核心代码如下：

```
protected void do_this_windowActivated(WindowEvent e) {
    DefaultMutableTreeNode root=new
DefaultMutableTreeNode("明日科技新书");          //设置根节点
    DefaultMutableTreeNodeparent1=newDefaultMutableTreeNode
("从入门到精通系列");
    parent1.add(new DefaultMutableTreeNode("《Java从入门到
精通（第2版）》"));
    parent1.add(new DefaultMutableTreeNode("《PHP从入门到精
通（第2版）》"));
    parent1.add(new DefaultMutableTreeNode("《VisualBasic从
入门到精通（第2版）》"));
    parent1.add(new DefaultMutableTreeNode("《VisualC++从入
门到精通（第2版）》"));
    root.add(parent1);
                    //增加子节点
    DefaultMutableTreeNode parent2=new
DefaultMutableTreeNode("编程词典系列");
    parent2.add(new DefaultMutableTreeNode("《Java编程词
典》"));
    parent2.add(new DefaultMutableTreeNode("《PHP编程词
典》"));
```

```
    parent2.add(new DefaultMutableTreeNode("《Visual Basic
编程词典》"));
    parent2.add(new DefaultMutableTreeNode("《Visual C++编
程词典》"));
    root.add(parent2);
                    //增加子节点
    DefaultTreeModel model=new
DefaultTreeModel(root);                          //使用根节
点创建默认树模型
    tree.setModel(model);
                    //更新树模型
  }
```

（2）编写方法do_button_actionPerformed()，用来监听单击“增加节点”按钮事件。在该方法中，实现了在用户选择的位置增加节点的功能。核心代码如下：

```
  protected void do_button_actionPerformed(ActionEvent e) {
    DefaultMutableTreeNode selectNode =
(DefaultMutableTreeNode)tree.getLastSelectedPathComponent()
;                                        //获得用户选择的节点
    if (selectNode==null)
{                                                //如果用户没有
选择节点则返回
      return;
    }
    DefaultTreeModelmodel= (DefaultTreeModel)
tree.getModel();                  //获得当前树的模型
```

```
    DefaultMutableTreeNode newNode=new
DefaultMutableTreeNode("NewNode");  //新建节点
    model.insertNodeInto(newNode, selectNode,
selectNode.getChildCount());        //增加节点
    TreeNode[]nodes=model.getPathToRoot(newNode);
        //向上构建节点的父节点一直到根节点
    TreePath path=new
TreePath(nodes);                                    //创建
TreePath对象
    tree.scrollPathToVisible(path);                 //确保路径
中所有的路径控件均展开（最后一个路径控件除外）并滚动
    tree.setSelectionPath(path);
                //选择指定路径标识的节点
    tree.startEditingAtPath(path);
                //设置新建的节点处于可编辑状态
    tree.repaint();
            //重新绘制树
  }
```

举一反三

根据本实例，读者可以实现以下功能。

添加自定义的树形节点。

删除自定义的树形节点。

实例092 在树控件中删除节点

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_92

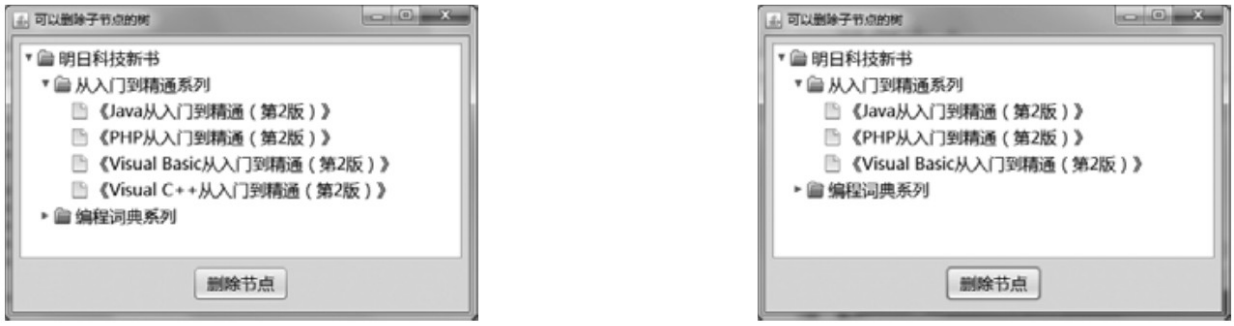
实例说明

除了让用户在给定的节点中进行选择外，还可以让用户在需要的位置删除节点。这样程序会有更好的交互效果。本实例将实现让用户删除节点的功能，如果该节点包括了子节点则一并删除。实例运行效果如图2.48所示。

技术要点

DefaultTreeModel 是使用 TreeNodes 的简单树数据模型。在该类中定义了各种与树控件数据相关的方法，如节点的增加、修改、删除等。此外，还有一些方法可以通知树控件其中的数据已经被修改。本实例使用了向树控件删除节点的方法removeNodeFromParent，该方法的声明如下：

```
public void removeNodeFromParent(MutableTreeNode node)
```



（a）删除节点前　　（b）删除节点后

图2.48 实例运行效果

参数说明

node：被删除的节点。

实现过程

（1）编写方法do_this_windowActivated()，用来监听窗体激活事件。在该方法中，为树增加了数据。核心代码如下：

```
protected void do_this_windowActivated(WindowEvent e) {
  DefaultMutableTreeNode root=new
DefaultMutableTreeNode("明日科技新书");          //设置根节
```

点

```
    DefaultMutableTreeNode parent1 = new
DefaultMutableTreeNode("从入门到精通系列");
    parent1.add(new DefaultMutableTreeNode("《Java从入门到
精通（第2版）》"));
    parent1.add(new DefaultMutableTreeNode("《PHP从入门到精
通（第2版）》"));
    parent1.add(new DefaultMutableTreeNode("《Visual Basic
从入门到精通（第2版）》"));
    parent1.add(new DefaultMutableTreeNode("《Visual C++从
入门到精通（第2版）》"));
    root.add(parent1);
                    //增加子节点
    DefaultMutableTreeNode parent2 = new
DefaultMutableTreeNode("编程词典系列");
    parent2.add(new DefaultMutableTreeNode("《Java编程词
典》"));
    parent2.add(new DefaultMutableTreeNode("《PHP编程词
典》"));
    parent2.add(new DefaultMutableTreeNode("《Visual Basic
编程词典》"));
    parent2.add(new DefaultMutableTreeNode("《Visual C++编
程词典》"));
    root.add(parent2);
                    //增加子节点
    DefaultTreeModelmodel=newDefaultTreeModel(root);
                      //使用根节点创建默认树模型
```

```
    tree.setModel(model);
                    //更新树模型
  }
```

（2）编写方法do_button_actionPerformed()，用来监听单击“删除节点”按钮事件。在该方法中，实现了删除用户选择的节点功能，如果该节点有子节点则一并删除。核心代码如下：

```
  protected void do_button_actionPerformed(ActionEvent e) {
    DefaultMutableTreeNode selectNode =
(DefaultMutableTreeNode)tree.getLastSelectedPathComponent()
;                                          //获得用户选择的
节点
    if ((selectNode==null) || (selectNode.isRoot()))
{                  //如果没有选择节点或选择了根节点则返回
      return;
    }
    DefaultTreeModel model= (DefaultTreeModel)
tree.getModel();                    //获得当前树的模型
    model.removeNodeFromParent(selectNode);
                    //删除节点
    tree.repaint();
                    //重新绘制树
  }
```

举一反三

根据本实例，读者可以实现以下功能。

增加节点。

删除节点。

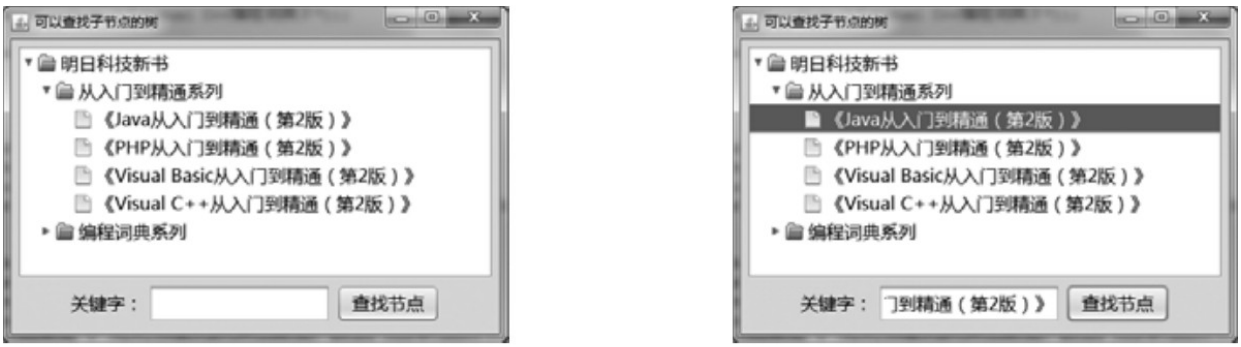
实例093 在树控件中查找节点

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_93

实例说明

对于复制的树控件，如果让用户展开每个节点来查找自己需要的信息是非常麻烦的。因此，最好为用户提供查找功能。根据用户的输入可以快速定位到用户需要的节点或者提示用户当前树中没有用户需要的节点。本实例实现了树节点的查找功能。实例运行效果如图2.49所示。



（a）查找节点前　　（b）查找节点后

图2.49 实例运行效果

技术要点

本实例应用到的关键技术请参见实例091。

实现过程

（1）编写方法do_this_windowActivated()，用来监听窗体激活事件。在该方法中，为树增加了数据。核心代码如下：

```
protected void do_this_windowActivated(WindowEvent e) {
    DefaultMutableTreeNode root = new
DefaultMutableTreeNode("明日科技新书"); //设置根节点
    DefaultMutableTreeNode parent1=new
DefaultMutableTreeNode("从入门到精通系列");
```

```
    parent1.add(new DefaultMutableTreeNode("《Java从入门到精通（第2版）》"));
    parent1.add(new DefaultMutableTreeNode("《PHP从入门到精通（第2版）》"));
    parent1.add(new DefaultMutableTreeNode("《VisualBasic从入门到精通（第2版）》"));
    parent1.add(new DefaultMutableTreeNode("《VisualC++从入门到精通（第2版）》"));
    root.add(parent1);
                //增加子节点
    DefaultMutableTreeNode parent2=new DefaultMutableTreeNode("编程词典系列");
    parent2.add(new DefaultMutableTreeNode("《Java编程词典》"));
    parent2.add(new DefaultMutableTreeNode("《PHP编程词典》"));
    parent2.add(new DefaultMutableTreeNode("《Visual Basic编程词典》"));
    parent2.add(new DefaultMutableTreeNode("《Visual C++编程词典》"));
    root.add(parent2);
                //增加子节点
    DefaultTreeModel model=new DefaultTreeModel(root);                    //使用根节点创建默认树模型
    tree.setModel(model);
            //更新树模型
```

```
}
```

（2）编写方法do_button_actionPerformed()，用来监听单击“查找节点”按钮事件。在该方法中，实现了根据用户输入的关键字进行查找的功能，如果未找到则进行提示。核心代码如下：

```
protected void do_button_actionPerformed(ActionEvent e) {
  String key=
textField.getText();                                    //获得用户输入的关键字
  if ((key==null) || (key.isEmpty()))
{                                  //如果关键字为空则提示用户重新输入
    JOptionPane.showMessageDialog(this, "请输入关键字", "",JOptionPane.WARNING_MESSAGE);
  DefaultTreeModel model= (DefaultTreeModel) tree.getModel();          //获得树模型
    return;
  }
  DefaultMutableTreeNode targetNode = null;
  Enumeration enums= root.breadthFirstEnumeration();                    //获得树的全部节点
  while (enums.hasMoreElements())
{                                  //遍历全部节点进行查找
    DefaultMutableTreeNode tempNode = (DefaultMutableTreeNode) enums.nextElement();
    if (""+ tempNode).equals(key)) {
      targetNode = tempNode;
```

```
        }
      }
      if (targetNode == null) {//如果没有找到则进行提示
        JOptionPane.showMessageDialog(this, "未找到需要的结果", "",JOptionPane.WARNING_MESSAGE);
        return;
      } else {//如果找到了则将该节点设置成选择状态
        TreeNode[] nodes = model.getPathToRoot(targetNode);
        TreePath path = new TreePath(nodes);
        tree.scrollPathToVisible(path);
        tree.setSelectionPath(path);
      }
    }
```

举一反三

根据本实例，读者可以实现以下功能。

快速查找树节点。

没查到的节点给出提示。

实例094 自定义树节点的外观

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_94

实例说明

使用树控件是非常简单的，只需要增加节点就可以实现一棵树，但是简单的代价是不能显示自己需要的节点效果。为了实现自定义的节点外观，需要使用 TreeCellRenderer 类。本实例使用该类来修改

树节点的颜色，并实现了显示多行的效果。实例运行效果如图2.50所示。

图2.50 实例运行效果

技术要点

TreeCellRenderer 接口是专门为了定制节点的外观而定义的。GetTreeCellRenderer Component()方法是该接口定义的唯一一个方法，该方法的声明如下：

Component getTreeCellRendererComponent(JTree tree,Object value,boolean selected,boolean expanded,boolean leaf,int row,boolean hasFocus)

该方法中各个参数的说明如表2.23所示。

表2.23 getTreeCellRendererComponent()方法的参数说明

方 法 名	作 用
tree	需要渲染的树
value	树的当前节点
selected	节点是否处于被选择状态

续表

方 法 名	作 用
expanded	节点是否处于展开状态
leaf	节点是否是叶子
row	节点的索引
hasFocus	节点是否具有焦点

实现过程

（1）编写类Book，在该类中定义了5个域变量，分别代表图书的书名、出版社、出版时间、丛书类别和定价属性。为了节约空间，省略了get和set方法。核心代码如下：

```
public class Book {
    privateString
title;                                          //书
名
    privateString
press;                                          //出
版社
    privateString
publicaitonDate;                                 //
出版时间
    privateString
booksCategory;                                  //丛
书类别
    privatedoubleprice;
            //定价
    //省略get和set方法
}
```

（2）编写类BookCellRenderer，该类实现了TreeCellRenderer接口。该类定义了5个域变量，分别用来显示图书的5种属性。在该类的构造方法中，为5个标签设置了不同的颜色和相同的字体，并将其增加到面板中。在 getTreeCellRendererComponent()方法中，渲染了 Book 类型的节点。核心代码如下：

```
public class BookCellRenderer implements TreeCellRenderer
{
```

```
    private JLabel titleLabel=new
JLabel();                                    //书名标签
    private JLabel pressLabel=new
JLabel();                                    //出版社标签
    private JLabel publicationDateLabel=new
JLabel();                              //出版时间标签
    private JLabel booksCategoryLabel=new
JLabel();                              //丛书类别标签
    private JLabel priceLabel=new
JLabel();                                    //定价标签
    private JPanel panel=new JPanel(new
GridLayout(5,1,5,5));                    //使用网格布局的面板
    public BookCellRenderer() {
      titleLabel.setForeground(Color.RED);
                  //设置标签的文本颜色
      titleLabel.setFont(new Font("微软雅
  黑",Font.PLAIN,16));                    //设置标签的字体
      panel.add(titleLabel);
              //在面板中增加标签
      pressLabel.setForeground(Color.GREEN);
                //设置标签的文本颜色
      pressLabel.setFont(new Font("微软雅
  黑",Font.PLAIN,16));                    //设置标签的字体
      panel.add(pressLabel);
              //在面板中增加标签
      publicationDateLabel.setForeground(Color.BLUE);
                  //设置标签的文本颜色
```

```
        publicationDateLabel.setFont(new Font("微软雅
    黑",Font.PLAIN,16));       //设置标签的字体
        panel.add(publicationDateLabel);
                    //在面板中增加标签
        booksCategoryLabel.setForeground(Color.ORANGE);
                    //设置标签的文本颜色
        booksCategoryLabel.setFont(new Font("微软雅
    黑",Font.PLAIN,16));       //设置标签的字体
        panel.add(booksCategoryLabel);
                  //在面板中增加标签
        priceLabel.setForeground(Color.PINK);
                    //设置标签的文本颜色
        priceLabel.setFont(new Font("微软雅
    黑",Font.PLAIN,16));                  //设置标签的字体
        panel.add(priceLabel);
                  //在面板中增加标签
        panel.setPreferredSize(newD
    imension(350,110));                          //设置面板的大小
      }
      @Override
      public Component getTreeCellRendererComponent(JTree
  tree, Object value, boolean selected, boolean expanded,
  boolean leaf, int row, boolean hasFocus) {
        Object userObject = ((DefaultMutableTreeNode)
    value).getUserObject();
        if (userObject instanceofBook)
    {                              //对于Book类型的节点使用自定
```

```
义渲染器
        Book book= (Book)
  userObject;                                          //获得
  Book类型的对象
        titleLabel.setText("书
  名："+book.getTitle());                               //设置
  属性
        pressLabel.setText("出版
  社："+book.getPress());                               //设置属
  性
        publicationDateLabel.setText("出版时
  间："+book.getPublicationDate());       //属性
        booksCategoryLabel.setText("丛书类
  别："+book.getBooksCategory());       //设置属性
        priceLabel.setText("定价："+book.getPrice()+
  "元");                          //设置属性
        return panel;
     } else
{                                                      //对于其
他节点使用默认的渲染器
        return new
  DefaultTreeCellRenderer().getTreeCellRendererComponent(
  tree, value, selected, expanded, leaf, row, hasFocus);
     }
  }
}
```

（3）编写方法do_this_windowActivated()，用来监听窗体激活事件。在该方法中，创建了两个Book对象作为树的节点，并为树设置了渲染器。核心代码如下：

```
protected void do_this_windowActivated(WindowEvent e) {
    DefaultMutableTreeNode root=new
DefaultMutableTreeNode("从入门到精通系列");  //根节点
    Book java=new
Book();                                                    //创建
Book对象并为其设置属性
    java.setTitle("《Java从入门到精通（第2版）》");
    java.setPress("清华大学出版社");
    java.setPublicationDate("2010-07-01");
    java.setBooksCategory("软件工程师入门丛书");
    java.setPrice(59.8);
    DefaultMutableTreeNode javaNode=new
DefaultMutableTreeNode(java);          //创建树节点
    root.add(javaNode);
            //为根节点增加节点
    Book php=new
Book();                                                    //创建
Book对象并为其设置属性
    php.setTitle("《PHP从入门到精通（第2版）》");
    php.setPress("清华大学出版社");
    php.setPublicationDate("2010-07-01");
    php.setBooksCategory("软件工程师入门丛书");
    php.setPrice(69.8);
```

```
    DefaultMutableTreeNode phpNode=new
DefaultMutableTreeNode(php);          //创建树节点
    root.add(phpNode);
            //为根节点增加节点
    DefaultTreeModel model= (DefaultTreeModel)
tree.getModel();                    //获得树的模型
    model.setRoot(root);
              //为模型设置根节点
    tree.setModel(model);
            //使用新的模型
    tree.setCellRenderer(new
BookCellRenderer());                                  //使用新的
渲染器
  }
```

举一反三

根据本实例，读者可以实现以下功能。

为节点设置背景图片。

修改节点的背景色。

实例095 为树节点增加提示信息

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_95

实例说明

在比较复杂的程序中，为了节约界面的空间，会为树控件的节点提供比较简略的文本说明信息。作为对节点的补充说明，可以为其增

加提示信息。本实例将演示如何为节点增加自定义的提示信息。实例运行效果如图2.51所示。

图2.51 实例运行效果

技术要点

ToolTipManager类管理系统中的所有ToolTips。ToolTipManager包含众多属性，用于配置该工具提示需要多长时间显示出来、需要多长时间隐藏。该类使用了单例模式，为了获得该类的实例，需要使用其sharedInstance()方法，该方法的声明如下：

```
public static ToolTipManager sharedInstance()
```

使用registerComponent()方法可以注册一个工具提示管理控件，该方法的声明如下：

```
public void registerComponent(JComponent component)
```

参数说明

component：要添加的 JComponent 对象。

实现过程

（1）编写类 ToolTipNode，该类实现了 TreeCellRenderer 接口。在构造方法中，使用 map参数来初始化键值对。在getTreeCellRendererComponent()方法中，为默认树节点渲染器设置了自定义的提示信息。代码如下：

```
public class ToolTipNode implements TreeCellRenderer {
```

```
    private static final long serialVersionUID
=-1884123073630846839L;
    private DefaultTreeCellRenderer renderer = new
DefaultTreeCellRenderer();
    privateMap<DefaultMutableTreeNode,String>map;
                    //保存键值对
    public ToolTipNode(Map<DefaultMutableTreeNode, String>
map) {
      this.map=map;
              //初始化键值对
    }
    @Override
    public Component getTreeCellRendererComponent(JTree
tree, Object value, boolean selected, boolean expanded,
boolean leaf, int row, boolean hasFocus) {
      renderer.getTreeCellRendererComponent(tree, value,
  selected, expanded, leaf, row, hasFocus);//调用默认的
  getTreeCellRendererComponent()方法
      renderer.setToolTipText("<html><font face=微软雅黑
  size=16 color=red>"+ map.get(value) +"</font></html>");
      //设置提示信息
      return renderer;
    }
  }
```

（2）编写方法do_this_windowActivated()，用来监听窗体激活事件。在该方法中，为树增加了数据并使用了自定义的渲染器。核心代码如下：

```
  protected void do_this_windowActivated(WindowEvent e) {
    DefaultMutableTreeNode root = new
DefaultMutableTreeNode("明日科技新书");//创建根节点
    DefaultMutableTreeNode parent1 = new
DefaultMutableTreeNode("从入门到精通系列");
    parent1.add(new DefaultMutableTreeNode("《Java从入门到
精通（第2版）》"));
    parent1.add(new DefaultMutableTreeNode("《PHP从入门到精
通（第2版）》"));
    parent1.add(new DefaultMutableTreeNode("《Visual Basic
从入门到精通（第2版）》"));
    parent1.add(new DefaultMutableTreeNode("《Visual C++从
入门到精通（第2版）》"));
    root.add(parent1);//增加子节点
    DefaultMutableTreeNode parent2 = new
DefaultMutableTreeNode("编程词典系列");
    parent2.add(new DefaultMutableTreeNode("《Java编程词
典》"));
    parent2.add(new DefaultMutableTreeNode("《PHP编程词
典》"));
    parent2.add(new DefaultMutableTreeNode("《Visual Basic
编程词典》"));
    parent2.add(new DefaultMutableTreeNode("《Visual C++编
程词典》"));
    root.add(parent2);//增加子节点
    DefaultTreeModel model = new DefaultTreeModel(root);//
使用根节点创建默认树模型
```

```
        tree.setModel(model);//更新树模型
        ToolTipManager.sharedInstance().registerComponent(tree)
    ;//为树注册提示信息管理器
        Map<DefaultMutableTreeNode, String> map = new
    HashMap<DefaultMutableTreeNode, String>();
        //利用映射保存提示信息
        map.put(root, "明日科技");//增加提示信息
        map.put(parent1, "明日科技");//增加提示信息
        map.put(parent2, "明日科技");//增加提示信息
        //设置新的渲染器tree.setCellRenderer(new
    ToolTipNode(map));
    }
```

举一反三

根据本实例，读者可以实现以下功能。

为节点添加自定义提示。

增加提示信息的显示时间。

实例096 双击编辑树节点功能

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_96

实例说明

有时由程序员定义的树节点并不能很好地满足用户的需要。此时用户会希望能够编辑现有的树节点。本实例通过设置节点编辑器来实现这个功能。实例运行效果如图2.52所示。

（a）初始状态　　（b）编辑状态

图2.52 实例运行效果

技术要点

DefaultCellEditor是表单元格和树单元格的默认编辑器。它实现了TreeCellEditor接口，因此可以修改树的节点。DefaultCellEditor的构造方法支持将树的节点设置成使用复选框、组合框和文本域。本实例将其设置成了文本域，使用的构造方法声明如下：

```
public DefaultCellEditor(JTextField textField)
```

参数说明

textField：一个JTextField对象。

实现过程

编写方法do_this_windowActivated()，用来监听窗体激活事件。在该方法中，初始化了树的各个节点并设置了节点编辑器。核心代码如下：

```
protected void do_this_windowActivated(WindowEvent e) {
    DefaultMutableTreeNode root=new
DefaultMutableTreeNode("明日科技新书");              //创建根
节点
    DefaultMutableTreeNode parent1=new
DefaultMutableTreeNode("从入门到精通系列");
```

```
    parent1.add(newDefaultMutableTreeNode("《Java从入门到精通（第2版）》"));
    parent1.add(new DefaultMutableTreeNode("《PHP从入门到精通（第2版）》"));
    parent1.add(new DefaultMutableTreeNode("《Visual Basic从入门到精通（第2版）》"));
    parent1.add(new DefaultMutableTreeNode("《Visual C++从入门到精通（第2版）》"));
    root.add(parent1);
                                //增加子节点
    DefaultMutableTreeNode parent2=new DefaultMutableTreeNode("编程词典系列");
    parent2.add(new DefaultMutableTreeNode("《Java编程词典》"));
    parent2.add(new DefaultMutableTreeNode("《PHP编程词典》"));
    parent2.add(new DefaultMutableTreeNode("《Visual Basic编程词典》"));
    parent2.add(new DefaultMutableTreeNode("《Visual C++编程词典》"));
    root.add(parent2);
                                //增加子节点
    DefaultTreeModel model=new DefaultTreeModel(root);                    //创建树模型
    tree.setModel(model);
                                //设置树模型
```

```
    JTextField textField=new
JTextField();                                          //创
建文本域对象
    textField.setFont(new Font("微软雅
黑",Font.PLAIN,16));                                   //为文本
域设置字体
    TreeCellEditor editor=new
DefaultCellEditor(textField);
//创建树编辑器
    tree.setEditable(true);
                          //设置树节点可编辑
    tree.setCellEditor(editor);
                            //使用树编辑器
  }
```

举一反三

根据本实例，读者可以实现以下功能。

编辑根节点的文本。

编辑子节点的文本。

2.8 JTextPane控件的应用

实例097 自定义文档标题的样式

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_97

实例说明

在使用Word等工具软件时，可以为文档中不同的文本指定不同的样式。这样不仅能让文档看起来丰富多彩，还可以突出重点内容。本实例将演示如何使用JTextPane控件显示定义了样式的文本。实例运行效果如图2.53所示。

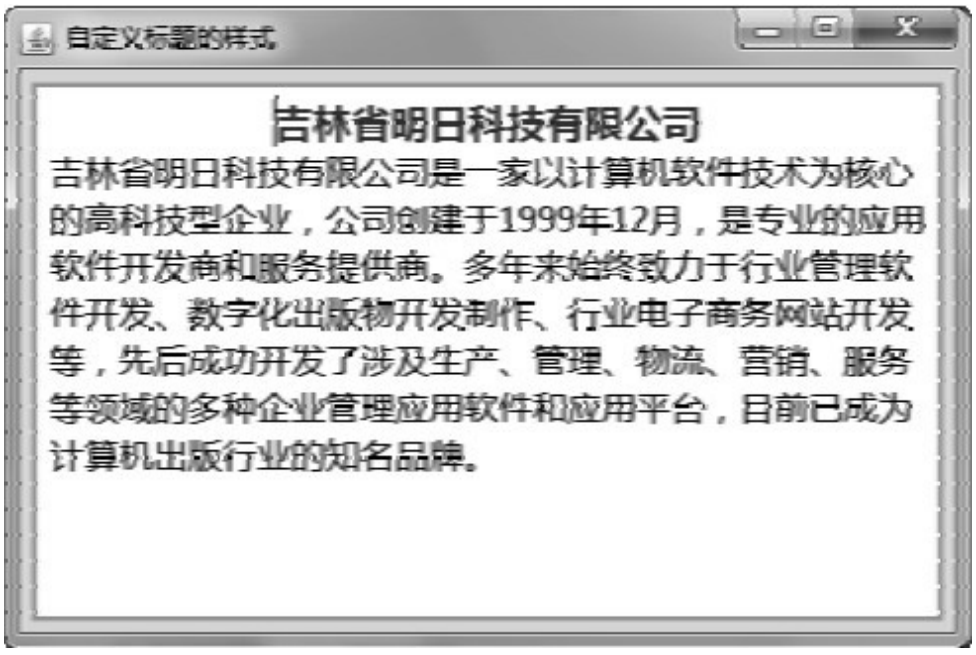


图2.53 实例运行效果

技术要点

Swing中文本的样式是通过Style接口定义的。由于该接口并没有提供直接的实现类。推荐使用StyleContext的addStyle()方法来获得Style对象。该方法的声明如下：

public Style addStyle(String nm,Style parent)

参数说明

● nm：样式的名称，其在文档中命名样式的集合内必须是唯一的。

● parent：父样式。如果未指定的属性不需要以其他样式解析，则此值可以为null。

在获得了 Style 对象之后，需要使用 addAttribute()方法为其增加新样式的属性。该方法的声明如下：

void addAttribute(Object name,Object value)

参数说明

● name：新增属性的键。

● value：新增属性的值。

属性的键通常取自 StyleConstants 类。在该类中定义了大量与样式有关的域，本实例使用的域如表2.24所示。

表2.24 StyleConstants的常用域

域　名	作　用	域　名	作　用
Alignment	段落的对齐方式	FontFamily	段落的字体名称
ALIGN_CENTER	设置段落居中对齐	FontSize	段落的字体大小
Bold	段落粗体显示	Foreground	字体的颜色

在定义完样式后，需要使用DefaultStyledDocument类的setParagraphAttributes()方法将样式应用于指定的段落上。该方法的声明如下：

public void setParagraphAttributes(int offset,int length,AttributeSet s,boolean replace)

其参数说明如表2.25所示。

表2.25 setParagraphAttributes()方法的参数说明

参 数 名	作 用	参 数 名	作 用
offset	段落偏移量，该偏移量≥0	s	段落的样式
length	所影响的字符数，该字符数≥0	replace	确定是替换现有属性还是合并现有属性

实现过程

（1）编写类HeadingStyle，该类继承了JFrame。在框架中包含了一个JTextPane，用来显示格式化的文本。

（2）编写方法do_this_windowActivated()，用来监听窗体激活事件。在该方法中，定义了文本的样式并将其应用到第一个段落。核心代码如下：

```
protected void do_this_windowActivated(WindowEvent e) {
    String heading ="吉林省明日科技有限公司\n";
    String content ="吉林省明日科技有限公司是一家以计算机软件技术为核心的高科技型企业，公司创建于1999年12月，是专业的应用软件开发商和服务提供商。多年来始终致力于行业管理软件开发、数字化出版物开发制作、行业电子商务网站开发等，先后成功开发了涉及生产、管理、物流、营销、服务等领域的多种企业管理应用软件和应用平台，目前已成为计算机出版行业的知名品牌。";
    StyleheadingStyle=newStyleContext().addStyle("Heading", null);                    //新建样式
    headingStyle.addAttribute(StyleConstants.Alignment, StyleConstants.ALIGN_CENTER);
    headingStyle.addAttribute(StyleConstants.Bold, new Boolean(true));
    headingStyle.addAttribute(StyleConstants.FontFamily, "微软雅黑");
```

```
    headingStyle.addAttribute(StyleConstants.FontSize, new
Integer(18));
    headingStyle.addAttribute(StyleConstants.Foreground,
Color.RED);
    DefaultStyledDocument document = new
DefaultStyledDocument();
    try {
      document.insertString(0,heading+
  content,null);                              //插入文本
    } catch (BadLocationException e1) {
      e1.printStackTrace();
    }
    document.setParagraphAttributes(0,1,headingStyle,
false);                              //为段落设置样式
    textPane.setDocument(document);
                  //显示带有格式的文本内容
  }
```

举一反三

根据本实例，读者可以实现以下功能。

自定义文档标题的颜色。

自定义文档标题的字体。

实例098 文档中显示自定义图片

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_98

实例说明

通过在文档中增加图片，不仅可以让文档更加美观，还有助于读者理解。本实例将演示如何使用JTextPane来显示明日科技公司的Logo，实例运行效果如图2.54所示。

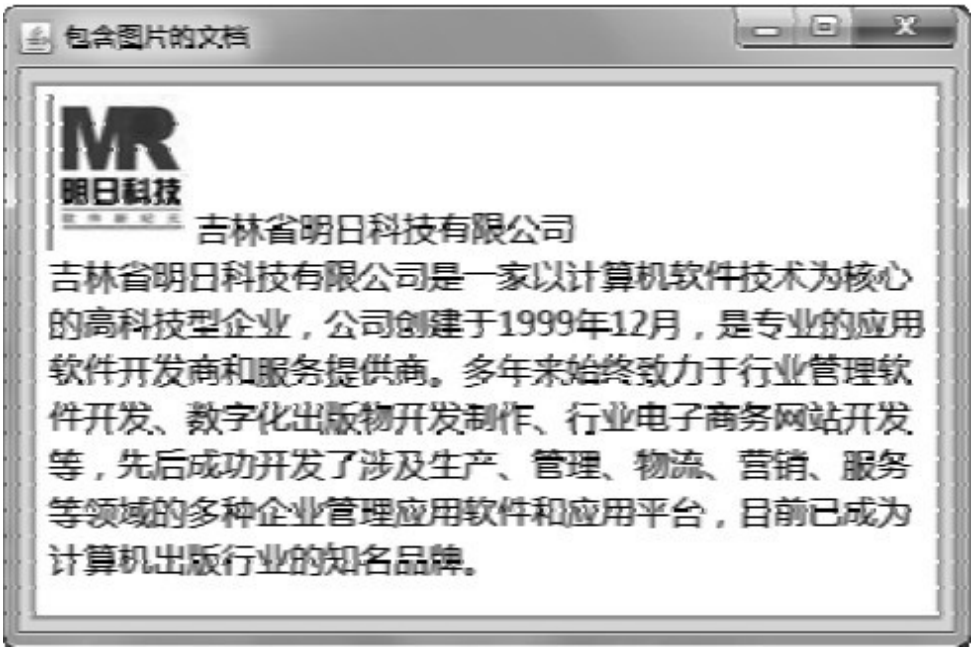


图2.54 实例运行效果

技术要点

StyleConstants类的setIcon()方法可用于为样式设置图标属性，该方法的声明如下：

```
public static void setIcon(MutableAttributeSet a,Icon c)
```

参数说明

- a：属性集合。
- c：图标。

包含图标的样式也可以像文本样式一样，使用insertString()方法设置。

实现过程

（1）编写类HeadingStyle，该类继承了JFrame。在框架中包含了一个JTextPane，用来显示格式化的文本。

（2）编写方法do_this_windowActivated()，用来监听窗体激活事件。在该方法中，定义了一个图片样式并在文档的开头使用了该样式。核心代码如下：

```
protected void do_this_windowActivated(WindowEvent e) {
  String heading ="吉林省明日科技有限公司\n";
  String content ="吉林省明日科技有限公司是一家以计算机软件技术为核心的高科技型企业，公司创建于1999年12月，是专业的应用软件开发商和服务提供商。多年来始终致力于行业管理软件开发、数字化出版物开发制作、行业电子商务网站开发等，先后成功开发了涉及生产、管理、物流、营销、服务等领域的多种企业管理应用软件和应用平台，目前已成为计算机出版行业的知名品牌。";
  Style imageStyle=new StyleContext().addStyle("Image",null);                    //新建样式
  StyleConstants.setIcon(imageStyle, new ImageIcon("src/images/logo.jpg"));
  DefaultStyledDocument document = new DefaultStyledDocument();
  try {
    document.insertString(0, "image", imageStyle);                    //插入图片
    document.insertString(document.getLength(),heading+content,null);          //插入文本
  } catch (BadLocationException e1) {
    e1.printStackTrace();
  }
```

```
        textPane.setDocument(document);
                        //显示带有格式的文本内容
    }
```

举一反三

根据本实例，读者可以实现以下功能。

使用insertString()方法显示图标。

调整图标的显示位置。

实例099 检查代码中的括号是否匹配

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_99

实例说明

在编写文档时，括号总是成对出现的，如“(”和“)”。对于程序设计而言，如果出现了不匹配的括号，通常会导致程序不能运行。因此，通过检查来避免这种错误就显得十分重要。本实例自定义了一个ParenthesisMatcher类，它支持对括号匹配性的检查。实例运行效果如图2.55所示。

图2.55 实例运行效果

技术要点

StyledDocument 接口用于定义通用的样式文档。setCharacterAttributes()方法用于更改内容元素属性，该属性是用来给定文档中现有内容范围的。给定Attributes参数中定义的所有属性都适用于此给定的范围。此方法可用来完全移除给定范围的所有内容层次的属性，这是通过提供尚未定义属性的AttributeSet参数和将replace参数设置为true实现的。该方法的声明如下：

```
void setCharacterAttributes(int offset,int
length,AttributeSet s,boolean replace)
```

其参数说明如表2.26所示。

表2.26 setCharacterAttributes()方法的参数说明

参 数 名	作 用	参 数 名	作 用
offset	段落偏移量，该偏移量≥0	s	段落的样式
length	所影响的字符数，该字符数≥0	replace	确定是替换现有属性还是合并现有属性

实现过程

（1）编写类ParenthesisMatcher，该类继承了JTextPane。其中包含两个域，分别表示匹配和不匹配的样式。在 validate()方法中，实现了比较括号是否匹配的功能。重写 replaceSelection()方法可以让新输入的文本不受上次检查结果的影响而改变颜色。match()方法用于检查括号是否匹配。代码如下：

```
public class ParenthesisMatcher extends JTextPane {
  private static final long serialVersionUID
=-5040590165582343011L;
  privateAttributeSetmismatch;
            //不匹配的样式
  privateAttributeSetmatch;
          //匹配的样式
```

```
  public ParenthesisMatcher() {
    StyleContext context =
StyleContext.getDefaultStyleContext();
    mismatch=
context.addAttribute(SimpleAttributeSet.EMPTY,StyleConsta
nts.Foreground,Color.RED);
    //如果不匹配就设置成红色
    match=
context.addAttribute(SimpleAttributeSet.EMPTY,StyleConsta
nts.Foreground,Color.BLACK);
    //如果匹配就设置成黑色
  }
  public void validate() {
    StyledDocument document = getStyledDocument();
    String text = null;
    try {
      text=document.getText(0,
  document.getLength());                    //获得文档中的
  内容
    } catch (BadLocationException e) {
      e.printStackTrace();
    }
    Stack<String> stack=newStack<String>
();                              //使用栈结构保存括号
    for (int i=0; i< text.length(); i++)
{                                       //遍历整个文档
      char c = text.charAt(i);
```

```
    if (c== '(' || c== '[' || c== '{'}
  {                                    //如果是左括号就入栈
      stack.push(""+ c + i);
      document.setCharacterAttributes(i,1,match,
    false);                 //设置文档的样式
    }
    if (c == ')' || c == ']' || c == '}') {
      String peek = stack.empty() ? ".": (String)
    stack.peek();
      if (match(peek.charAt(0), c))
    {                       //如果是右括号且和栈中的括号匹配
    就出栈
        stack.pop();
        document.setCharacterAttributes(i,1,match,
    false);       //设置文档的样式
    } else {
        document.setCharacterAttributes(i,1,mismatch,
    false);  //设置文档的样式
    }
    }
  }
  while (!stack.empty())
{                                      //如果栈非空则剩下的
全是未匹配的
    String pop = (String) stack.pop();
    int offset= Integer.parseInt(pop.substring(1));
```

```
            document.setCharacterAttributes(offset,1,mismatch,
        false);       //设置文档的样式
          }
        }
        @Override
        publicvoid replaceSelection(String content)
    {                          //删除文档的文字颜色属性
          getInputAttributes().removeAttribute(StyleConstants.F
      oreground);
          super.replaceSelection(content);
        }
        privatebooleanmatch(char left, char right)
    {                          //检查括号是否匹配
          if ((left == '(') && (right == ')')) {
            return true;
          }
          if ((left == '[') && (right == ']')) {
            return true;
          }
          if ((left == '{') && (right == '}')) {
            return true;
          }
          return false;
        }
      }
```

（2）编写方法do_button_actionPerformed()，用来监听单击“检查”按钮事件。在该方法中，调用了ParenthesisMatcher的

validate()方法。核心代码如下：

```
    protected void do_button_actionPerformed(ActionEvent e) {
      textPane.validate();
            //进行检查
    }
```

举一反三

根据本实例，读者可以实现以下功能。

括号匹配程序的改进。

修改该程序，提高匹配的概率。

实例100 描红显示100以内的质数

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_100

实例说明

当用户浏览文档时，通常希望关键信息能够突出显示，描红就是一种常见的方式。本实例为100以内所有的素数描红，方便用户查找自己需要的素数。实例运行效果如图2.56所示。

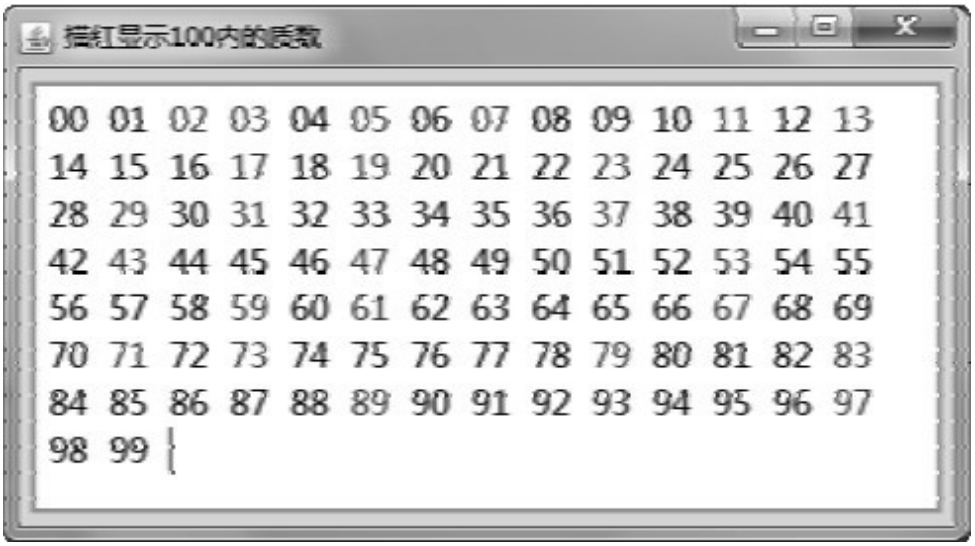


图2.56 实例运行效果

技术要点

JTextPane 是用来显示具有样式信息的文本控件。本实例在该类的基础上增加了一个append()方法，该方法可以根据用户指定的颜色和字符串，将其连续显示在JTextPane中。本实例使用的方法如表2.27所示。

表2.27 JTextPane类的常用方法

方 法 名	作 用
setCaretPosition(int position)	设置 TextComponent 的文本插入符的位置
setCharacterAttributes(AttributeSet attr,boolean replace)	将给定属性应用于字符内容
replaceSelection(String content)	用给定字符串所表示的新内容替换当前选择的内容

实现过程

（1）编写类 ColorPane，该类继承了 JTextPane。在该类中定义了 append()方法，该方法有两个参数：color用来设置文本的颜色，key用来设置文本的内容。代码如下：

```
public class ColorPane extends JTextPane {
  private static final long serialVersionUID =
7039422656649417533L;
  public void append(Color color, String key) {
    StyleContext
context=StyleContext.getDefaultStyleContext();    //创建
样式
    AttributeSet style=
context.addAttribute(SimpleAttributeSet.EMPTY,StyleConsta
nts.Foreground, color);
    int
length=getText().length();                          //获
得文档的长度
```

```
    setCaretPosition(length);                    //将光标定位
于文档的末尾，这样新插入的文档总是从最后插入
    setCharacterAttributes(style,
true);                          //应用样式
    replaceSelection(key);
  //设置文本的内容
  }
}
```

（2）编写方法isPrime()，用来判断给定的数number是否为素数，如果是则返回true。代码如下：

```
private boolean isPrime(int number) {
  if (number<2)
{                                          //0、1不是素数
    return false;
  } else {
    int sqrt=
(int)Math.sqrt(number);                          //求给定数
的平方根
    for (int i=2; i<= sqrt; i++)
{                           //遍历可能的公因数
      if (number% i==0) {                              //如
果有公因数则不是素数
        return false;
      }
    }
  }
  return true;
```

```
}
```

（3）编写类ColorPaneTest，该类继承了JFrame。在框架中包含了一个自定义的文本控件，用来描红显示0～99的全部素数。核心代码如下：

```
public ColorPaneTest() {
  //省略域ColorPane无关的代码
  ColorPane textPane = new ColorPane();
  textPane.setFont(new Font("微软雅黑", Font.PLAIN, 16));
  for (int i = 0; i < 10; i++) {
    if (isPrime(i)) {
      textPane.append(Color.RED, "0"+ i+
"  ");              //素数设置成红色
    } else {
      textPane.append(Color.BLACK, "0"+ i+
"  ");            //其他设置成黑色
    }
  }
  for (int i = 10; i < 100; i++) {
    if (isPrime(i)) {
      textPane.append(Color.RED, ""+ i+
"  ");                //素数设置成红色
    } else {
      textPane.append(Color.BLACK, ""+ i+
"  ");              //其他设置成黑色
    }
  }
  scrollPane.setViewportView(textPane);
```

```
}
```

举一反三

根据本实例，读者可以实现以下功能。

描红显示200以内的质数。

描红显示10以内的质数。

2.9 JEditorPane控件的应用

实例101 自定义RTF文件查看器

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_101

实例说明

RTF文件也称为富文本文件，它是微软规定的文件保存格式。这种格式的特点是既能保存文本的样式信息，又能提供很好的兼容性。本实例将自定义一个RTF文件查看器，以此来演示JEditorPane控件的用法。实例运行效果如图2.57所示。

图2.57 实例运行效果

技术要点

RTFEditorKit类是对RTF编辑功能的默认实现，使用该类的read()方法可以从输入流中读取带有格式的文件。该方法的声明如下：

public void read(InputStream in,Document doc,int pos)throws IOException,BadLocationException

参数说明

- in：从中读取数据的流。
- doc：插入的目标。
- pos：文档中存放内容的位置。

实现过程

（1）编写类RTFViewer，该类继承了JFrame。在框架中包含了一个“打开”按钮，用来打开指定的RTF文件；一个文本域，用来显示打开的文件名称；一个 JEditorPane控件，用来显示打开后的文件。

（2）编写方法do_button_actionPerformed()，用来监听单击“打开”按钮事件。在该方法中，定义了一个文件选择器，根据用户选择的文件来进行读入。核心代码如下：

```
protected void do_button_actionPerformed(ActionEvent e) {
    JFileChooser chooser=new
JFileChooser();                                    //创建文件选择
器
    chooser.setMultiSelectionEnabled(false);
                    //不能支持多选
    chooser.setFileFilter(new FileNameExtensionFilter("RTF
文件", "rtf"));          //过滤可选的文件
    int result=
chooser.showOpenDialog(this);
//获得用户操作文件选择器的结果
    if (result== JFileChooser.APPROVE_OPTION)
{                              //如果用户选择打开
```

```
    File file=
chooser.getSelectedFile();
  //获得选择的文件
    textField.setText(file.getName());
                  //在文本域中设置文件名
    try {
      FileInputStream
  in=newFileInputStream(file);                    //创建
  输入流对象
      editor.read(in,
  editorPane.getDocument(),0);                      //读
  入RTF文件
    } catch (FileNotFoundException e1) {
      e1.printStackTrace();
    } catch (IOException e1) {
      e1.printStackTrace();
    } catch (BadLocationException e1) {
      e1.printStackTrace();
    }
  }
}
```

举一反三

根据本实例，读者可以实现以下功能。

自定义文件查看器。

用RTF查看文件。

实例102 编写简单的浏览器

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_102

实例说明

在日常上网中，用户可能会使用各种不同的浏览器，如IE、Firefox等。其实，使用JEditorPane控件就可以非常简单地实现一个浏览器。当然，简单的代价是这个浏览器的功能是非常有限的。本实例将演示其用法，实例运行效果如图2.58所示。

图2.58 实例运行效果

技术要点

JEditorPane是可编辑各种内容的文本控件。当使用它显示HTML时只能显示简单的文件，对于大多数网站而言显示的效果非常不好。可以使用该控件来显示自定义的HTML文件，因为可以随时修改。该控件显示网页的代码非常简单，调用setPage()方法即可。该方法的声明如下：

```
public void setPage(String url)throws IOException
```

参数说明

url：要显示的URL。

实现过程

（1）编写类HTMLViewer，该类继承了JFrame。在框架中包含了一个文本域，用来获得用户输入的网址；一个JEditorPane控件，用来显示网页。

（2）编写方法do_textField_actionPerformed()，用来监听用户在文本域单击回车事件。在该方法中，让JEditorPane显示用户输入的网址。核心代码如下：

```
    protected void do_textField_actionPerformed(ActionEvent
e) {
      String url=
  textField.getText();                                    //获
  得用户输入的网址
      try {
        editorPane.setPage(url);
            //显示用户输入的网址
      } catch (IOException e1) {
        e1.printStackTrace();
      }
    }
```

举一反三

根据本实例，读者可以实现以下功能。

增加前进、后退功能。

实现刷新功能。

实例103 支持超链接的浏览器

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_103

实例说明

一个网站是由若干个不同的网页组成的。当用户需要浏览不同的页面时就需要使用超链接功能。当用户单击一个超链接时，就会打开一个新的网页。本实例将演示如何使用JEditorPane监听该类事件。实例运行效果如图2.59所示。

(a) 单击超链接前

(b) 单击超链接后

图2.59 实例运行效果

技术要点

为了打开超链接，需要为 JEditorPane 控件增加 HyperlinkListener 监听。HyperlinkListener接口中定义了一个hyperlinkUpdate()方法，该方法的声明如下：

```
void hyperlinkUpdate(HyperlinkEvent e)
```

参数说明

e：负责更新的事件。

HyperlinkEvent 用于通知感兴趣的参与者发生了与超文本链接有关的事情。它的getEventType()方法可以用于获得事件的具体类型，通常关心的是激活事件。然后就可以对该事件做出响应。

实现过程

（1）编写类HTMLViewer，该类继承了JFrame。在框架中包含了一个文本域，用来获得用户输入的网址；一个JEditorPane控件，用来显示网页；一个“浏览”按钮，用来打开用户输入的网页。

（2）编写方法 do_button_actionPerformed()，用来监听单击“浏览”按钮事件。在该方法中，使用JEditorPane控件显示用户输入的网址。核心代码如下：

```
protected void do_button_actionPerformed(ActionEvent e) {
  try {
    editorPane.setPage(textField.getText());
                    //显示用户输入的网址
  } catch (IOException e1) {
    e1.printStackTrace();
  }
}
```

（3）编写方法do_editorPane_hyperlinkUpdate()，用来监听超链接激活事件。在该方法中，使用JEditorPane控件显示新的网页。核心代码如下：

```
protected void
do_editorPane_hyperlinkUpdate(HyperlinkEvent e) {
  if
(e.getEventType()==HyperlinkEvent.EventType.ACTIVATED)
{              //如果超链接被激活
    try {
      editorPane.setPage(e.getURL());
              //显示新的网页
    } catch (IOException e1) {
      e1.printStackTrace();
    }
  }
}
```

举一反三

根据本实例，读者可以实现以下功能。

单击图片代替文本的超链接。

添加鼠标事件。

实例104 高亮显示指定的关键字

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_104

实例说明

在文档中查找内容时，通常有两种做法。一种类似于 Word 2007，可以依次定位各个符合条件的关键字，另一种类似于 NetBeans，可以高亮显示全部符合条件的关键字。本实例实现了第二种方法，实例运行效果如图2.60所示。

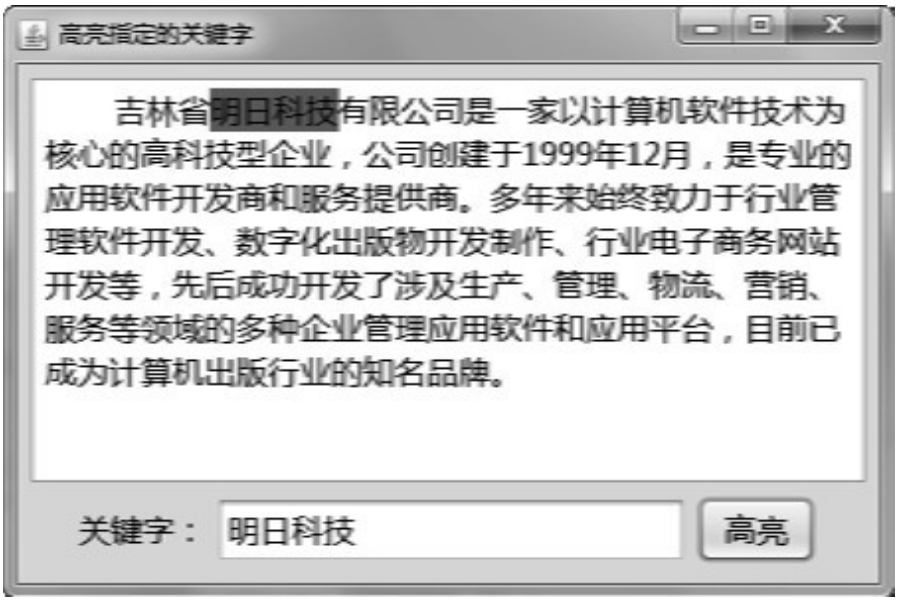


图2.60 实例运行效果

技术要点

Highlighter接口的addHighlight()方法可以用来给指定区域的文本增加高亮，该方法的声明如下：

Object addHighlight(int p0,int p1,Highlighter.HighlightPainter p)throws BadLocationException

参数说明

- p0：范围的开头，该值≥0。
- p1：范围的结尾，该值≥p0。
- p：用于实际高亮显示的painter。

因此需要确定高亮的起始位置p0，终止位置是p0加上关键字的长度。

String类的indexOf()方法可以用来查找关键字在字符串中第一次出现的位置，该方法的声明如下：

public int indexOf(String str)

参数说明

str：任意字符串。

实现过程

（1）编写类HighLightKeyWord，该类继承了JFrame。在框架中包含了一个文本域，用来获得用户输入的关键字；一个JEditorPane控件，用来显示文本；一个“高亮”按钮，用来高亮用户输入的关键字。

（2）编写方法do_button_actionPerformed()，用来监听单击“高亮”按钮事件。在该方法中，遍历JEditorPane控件查找用户输入的关键字并将其设置成高亮。核心代码如下：

```
protected void do_button_actionPerformed(ActionEvent e) {
  String key=
textField.getText();                                    /
/获得关键字
  String content=
editorPane.getText();                                   //获
```

```
得JEditorPane中的所有文本
    Highlighter highlighter=
editorPane.getHighlighter();                          //获得
默认的Highlighter对象
    highlighter.removeAllHighlights();
                //移除原有的高亮显示区域
    if (content.contains(key))
{                                                  //如果包含关键字
      int index=
  content.indexOf(key);                                    //
  确定第一个关键字的位置
      while (true) {
        if (index != -1)
    {                                                 //如果还有关
    键字为高亮
          try
        {                                                   //高
        亮关键字
            highlighter.addHighlight(index,
          index+key.length(),DefaultHighlighter.DefaultPainte
          r);
          } catch (BadLocationException e1) {
            e1.printStackTrace();
          }
          index=
        content.indexOf(key,++index);                       //
        确定下一个关键字的位置
```

```
        } else {
          break;
        }
      }
    }
}
```

举一反三

根据本实例，读者可以实现以下功能。

实现高亮显示。

设置高亮的区域形状。

2.10 进度指示器的应用

实例105 显示完成情况的进度条

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_105

实例说明

当程序执行一些耗时操作时，使用进度指示器提示用户程序正在运行是非常有用的。Swing中的进度指示器可以分成3类，本实例将演示在文本区中输出500以内的全部素数，并使用进度条提示用户输出的进度。实例运行效果如图2.61所示。

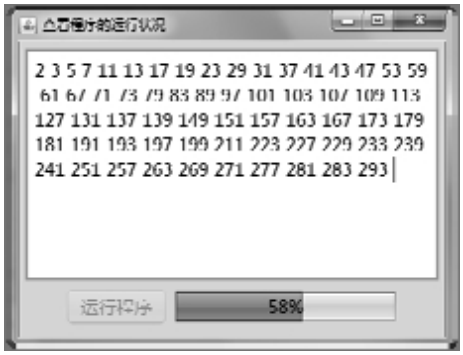


图2.61 实例运行效果

技术要点

JProgressBar 类是以可视化形式显示某些任务进度的控件。在任务的完成进度中，进度条显示该任务完成的百分比。此百分比通常由一个矩形以可视化形式表示，该矩形开始是空的，随着任务的完成逐渐被填充。此外，进度条能够显示此百分比的文本表示形式。JProgressBar类的常用方法如表2.28所示。

表2.28 JProgressBar类的常用方法

方 法 名	作 用
setMaximum(int n)	将进度条的最大值（存储在进度条的数据模型中）设为 n
setMinimum(int n)	将进度条的最小值（存储在进度条的数据模型中）设为 n
setOrientation(int newOrientation)	将进度条的方向设置为 newOrientation
setStringPainted(boolean b)	设置进度条是否应该显示进度字符串
setValue(int n)	将进度条的当前值设置为 n

实现过程

（1）编写类ProgressBarTest，该类继承了JFrame。在框架中包含了一个文本区，用来显示计算出来的素数；一个“运行程序”按钮，用来执行程序；一个进度条，用来显示进度。

（2）编写类 Activity，该类继承自 SwingWorker。它负责计算素数并更新进度条，由于SwingWorker是在事件分发线程中调用方法，因此可以避免线程问题。代码如下：

```
    private class Activity extends SwingWorker<Void, Integer>
{
      protectedVoid doInBackground() throwsException
  {                    //筛选出所有满足条件的素数
      private int current;
      private int target;
      public Activity(int target) {
        this.target = target;
      }
      @Override
        while (current < target) {
          Thread.sleep(100);
          if (isPrime(current)) {
            publish(current);
          }
```

```
        current++;
      }
      return null;
    }
    @Override
    protected void process(List<Integer> chunks)
{                       //更新文本区和进度条
      for (Integer chunk : chunks) {
        textArea.append(chunk +"");
        progressBar.setValue(chunk / 5);
        if (chunk == 499) {
          progressBar.setValue(100);
        }
      }
    }
    @Override
    protected void done()
{                                          //启用按钮
      button.setEnabled(true);
    }
    privateboolean isPrime(int number)
{                         //计算素数值
      if (number<2)
  {                                        //0、1不是素数
        return false;
      } else {
```

```
        int sqrt=
    (int)Math.sqrt(number);                    //求给定数的
    平方根
        for (int i=2; i<= sqrt; i++)
    {                          //遍历可能的公因数
          if (number% i==0) {                          //如
      果有公因数则不是素数
            return false;
          }
        }
      }
      return true;
    }
}
```

（3）编写方法do_button_actionPerformed()，用来监听单击“运行程序”按钮事件。在该方法中，禁用了按钮并启动了SwingWorker线程。核心代码如下：

```
protected void do_button_actionPerformed(ActionEvent e) {
  button.setEnabled(false);
  //禁用按钮
  Activity activity = new Activity(500);
  activity.execute();
  //启动线程
}
```

举一反三

根据本实例，读者可以实现以下功能。

可以拖动的进度条。

进度条的显示位置。

实例106 监听进度条的变化事件

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_106

实例说明

除了可以用进度条提示用户程序正在运行外，还可以用来监视进度条的变化。当进度条显示某一数值时，执行一些预定的操作。本实例将监听进度条的变化事件，如果到 100%则更新程序的界面。实例运行效果如图2.62所示。

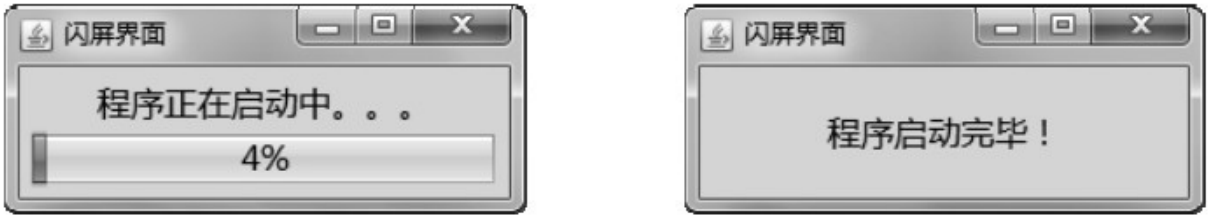


图2.62 实例运行效果（进度条运行完毕后显示如右图）

技术要点

为了能够让程序监听进度条的变化事件，需要为其添加ChangeListener。该接口中定义了stateChanged()方法，它可以用来处理变化所引发的各种事件，其声明如下：

```
void stateChanged(ChangeEvent e)
```

参数说明

e：ChangeEvent 对象。

实现过程

（1）编写类 Activity，该类继承自 SwingWorker。它用来更新进度条，由于 SwingWorker是在事件分发线程中调用方法，因此可以避免线程问题。代码如下：

```
    private class Activity extends SwingWorker<Void, Integer>
{
      @Override
      protectedVoid doInBackground() throwsException
  {                        //存入100个整数
        for (int i=1;i<101;i++) {
          Thread.sleep(100);
          publish(i);
        }
        return null;
      }
      @Override
      protected void process(List<Integer> chunks)
  {                              //更新进度条
        for (Integer chunk:chunks) {
          progressBar.setValue(chunk);
        }
      }
    }
```

（2）编写方法 do_this_windowActivated()，用来监听窗体激活事件。在该方法中，启动了SwingWorker线程。核心代码如下：

```
    protected void do_this_windowActivated(WindowEvent e) {
      Activity activity = new Activity();
      activity.execute();
              //启动线程
    }
```

（3）编写方法 do_progressBar_stateChanged()，用来监听进度条变化事件。在该方法中，更新了标签内容并隐藏了进度条。核心代码如下：

```
    protected void do_progressBar_stateChanged(ChangeEvent e)
{
      JProgressBar comp = (JProgressBar) e.getSource();
      int value = comp.getValue();
      if(value==100) {
        label.setText("程序启动完
    毕！");                                    //更新标签内容
        comp.setVisible(false);
                //隐藏进度条
      }
    }
```

举一反三

根据本实例，读者可以实现以下功能。

增加进度条的透明度。

当进度条达到某一值时，启动相应操作。

实例107 进度监视器控件的应用

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_107

实例说明

当程序执行一些耗时操作时，使用进度指示器提示用户程序正在运行是非常有用的。Swing中的进度指示器可以分成3类，本实例将演

示在文本区中输出500以内的全部素数，并使用进度监视器提示用户输出的进度。实例运行效果如图2.63所示。

图2.63 实例运行效果

技术要点

进度监视器是一个包含进度条和“取消”按钮的对话框。单击“取消”按钮可以停止程序的运行。创建进度监视器时，向它提供数字范围和描述字符串。在操作进行时，调用setProgress方法，以指示操作的[min,max]范围有多大。其构造方法如下：

public ProgressMonitor(Component parentComponent,Object message,String note,int min,int max)

该构造方法中各个参数的说明如表2.29所示。

表2.29 ProgressMonitor构造方法的参数说明

参 数 名	作 用
parentComponent	对话框的父控件
message	要显示给用户的描述消息，以指示在监视什么操作
note	描述操作状态的简短注释
min	范围的下边界
max	范围的上边界

实现过程

（1）编写类ProgressMonitorTest，该类继承了JFrame。在框架中包含了一个文本区，用来显示计算出来的素数；一个“运行程序”按钮，用来执行程序，当单击该按钮后会弹出一个进度监视器。

（2）编写类Activity，该类继承自SwingWorker。它负责计算素数并更新进度监视器，由于SwingWorker是在事件分发线程中调用方法，因此可以避免线程问题。代码如下：

```
private class Activity extends SwingWorker<Void, Integer>
{
    protectedVoid doInBackground() throwsException
  {                    //筛选出所有满足条件的素数
    private int current;
    private int target;
    public Activity(int target) {
      this.target = target;
    }
    @Override
      while (current < target) {
        Thread.sleep(100);
        if (isPrime(current)) {
          publish(current);
        }
        current++;
      }
      return null;
    }
    @Override
```

```
    protected void process(List<Integer> chunks)
{                       //更新文本区和进度监视器
      for (Integer chunk : chunks) {
        textArea.append(chunk +"");
        setProgress(chunk / 5);
      }
    }
    privateboolean isPrime(int number)
{                               //计算素数值
      if (number<2)
  {                                                 //0、1不是素数
        return false;
      } else {
        int sqrt=
    (int)Math.sqrt(number);                                 //求给定
    数的平方根
        for (int i=2; i<= sqrt; i++)
    {                               //遍历可能的公因数
          if (number% i==0)
      {                                 //如果有公因数则不是素
      数
            return false;
          }
      }
      return true;
    }
```

```
}
```

（3）编写类CancelAction，该类实现了ActionListener接口。该类负责监视是否单击了“取消”按钮，如果没有则更新进度。代码如下：

```
private class CancelAction implements ActionListener {
  @Override
  public void actionPerformed(ActionEvent e) {
    if (monitor.isCanceled())
{                                    //如果单击“取消”按钮则停止线程
      activity.cancel(true);
    } else if (activity.isDone())
{                                    //如果线程运行结束则停止进度监视器
      monitor.close();
      button.setEnabled(true);
    } else
{                                                    //更新进度监视器
      monitor.setProgress(activity.getProgress());
    }
  }
}
```

（4）编写方法do_button_actionPerformed()，用来监听单击“运行程序”按钮事件。在该方法中，创建了进度监视器对象，并每间隔 0.5 秒对其是否单击了“取消”按钮进行判断。核心代码如下：

```
protected void do_button_actionPerformed(ActionEvent e) {
```

```
        button.setEnabled(false);
            //禁用按钮
        int max = 500;
        activity = new Activity(max);
        activity.execute();
            //启动线程
        monitor=newProgressMonitor(ProgressMonitorTest.this,
    "正在计算素数",null,0,max);
        newTimer(500,newCancelAction()).start();                    /
    /利用计时器周期性地检查是否单击了“取消”按钮
    }
```

举一反三

根据本实例，读者可以实现以下功能。

对100以内的素数描红并添加进度条。

进度条达到某一个值时显示素数。

实例108 监视文件读入的进度

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_108

实例说明

当程序执行一些耗时操作时，使用进度指示器提示用户程序正在运行是非常有用的。Swing中的进度指示器可以分成3类，本实例将演示从文本磁盘中读取一个非常大的文本文件，使用ProgressMonitorInputStream来显示读取的进度。实例运行效果如图2.64所示。

图2.64 实例运行效果

技术要点

ProgressMonitorInputStream类可以自动弹出一个对话框，监视已经读取了多少流。它使用InputStream类的available()方法来确定流中的总字节数。该类的构造方法如下：

public ProgressMonitorInputStream(Component parentComponent,Object message,InputStream in)

该构造方法中各个参数的说明如表2.30所示。

表2.30 ProgressMonitorInputStream构造方法的参数说明

参 数 名	作 用
parentComponent	触发被监视操作的控件
message	在对话框（如果弹出）中放置的描述性文本
in	要监视的输入流

实现过程

（1）编写类 ProgressMonitorInputStreamTest，该类继承了JFrame。在框架中包含了一个文本域，用来显示用户选择的文件名；

一个“打开文件”按钮，用来选择文件；一个文本区，用来显示读入的文本。

（2）编写方法do_button_actionPerformed()，用来监听单击“打开文件”按钮事件。在该方法中，使用文件选择器让用户选择文件并将其显示在文本区中。核心代码如下：

```
protected void do_button_actionPerformed(ActionEvent e) {
  JFileChooser chooser=new
JFileChooser();                                    //创建文件选择
器
    chooser.setMultiSelectionEnabled(false);
                  //限制不能多选
    chooser.setFileFilter(new FileNameExtensionFilter("TXT
文件", "txt"));          //过滤非txt文件
    int result = chooser.showOpenDialog(this);
    if (result== JFileChooser.APPROVE_OPTION)
{                           //如果用户选择了文件
      File file=
  chooser.getSelectedFile();
    //获得文件
      textField.setText(file.getName());
                  //显示文件名称
      try {
        FileInputStream fileIn=new
    FileInputStream(file);                    //创建文件输入
    流
        ProgressMonitorInputStream progressIn=new
    ProgressMonitorInputStream(this, "正在读入文
```

```
件："+file.getName(),
fileIn);
            //创建输入流进度显示器
        final Scanner in = new Scanner(progressIn);
        textArea.setText("");
              //清空文本区
        SwingWorker<Void,Void>worker=new
    SwingWorker<Void,Void>() {
          @Override
          protected Void doInBackground() throwsException {
            while (in.hasNextLine())
        {                                  //读入文本并在文本区中
        显示
              textArea.append(in.nextLine());
            }
            in.close();
            //关闭输入流
            return null;
          }
        };
        worker.execute();
      } catch (IOException e1) {
        e1.printStackTrace();
      }
    }
}
```

举一反三

根据本实例，读者可以实现以下功能。
对文件等长度可知的输入流使用监听器。
监听器监听进度条。

2.11 微调控件

实例109 使用微调控件调整时间

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_109

实例说明

当确定用户的输入只能取自一个连续有界的范围时，可以使用滑块或微调控件来让用户进行选择。这样就可以避免烦琐的校验步骤。为了节约空间，可以使用微调控件。本实例将使用微调控件调整当前的时间，实例运行效果如图2.65所示。

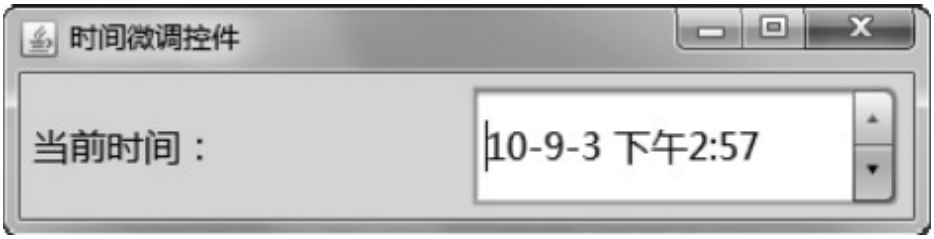


图2.65 实例运行效果

技术要点

JSpinner 让用户从一个有序序列中选择一个数字或者一个对象值的单行输入字段。Spinner 通常提供一对带小箭头的按钮，以便逐步遍历序列元素。键盘的向上/向下方向键也可循环遍历元素。用户可以在 Spinner 中直接输入合法值。尽管组合框提供了相似的功能，但因为Spinner不要求隐藏重要数据的下拉列表，所以有时它也成为首要选择。为了让控件显示当前时间，需要使用setModel()方法设置其模型，该方法的声明如下：

```
public void setModel(SpinnerModel model)
```

参数说明

model：新的 SpinnerModel。

SpinnerDateModel 是 Date 序列的一个 SpinnerModel。序列的上下边界由称为 start 和 end的属性定义，而通过 nextValue 和 previousValue 方法计算的增加和减少的大小由称作calendarField的属性定义。start和end属性可以为null，以指示序列没有下限和上限。

实现过程

编写方法do_this_windowActivated()，用来监听窗体激活事件。在该方法中，为微调控件设置了新的模型。核心代码如下：

```
protected void do_this_windowActivated(WindowEvent e) {
  spinner.setModel(newSpinnerDateModel());
                                //设置模型
}
```

举一反三

根据本实例，读者可以实现以下功能。

使用微调控件调整日期。

使用微调控件修改系统时间。

实例110 使用微调控件浏览图片

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_110

实例说明

除了可以使用默认的微调控件显示文本信息外，还可以通过自定义其显示模型的控件来显示图片等其他信息。本实例将使用微调控件实现浏览图片的功能，实例运行效果如图2.66所示。

图2.66 实例运行效果

技术要点

SpinnerListModel类是由数组或List定义的SpinnerModel的简单实现。此类只存储对该数组或 List 的引用，所以，如果基础序列的元素发生变化，则应用程序有责任通过调用fireStateChanged通知ChangeListeners。本实例使用该类存储了一个图标数组。为了能够显示图标，需要使用setEditor()方法将显示模型的控件换成标签，该方法的声明如下：

```
public void setEditor(JComponent editor)
```

参数说明

editor：新编辑器。

实现过程

（1）编写类ImageLabel，该类继承了JLabel并且实现了ChangeListener接口。在响应微调控件改变事件的stateChanged()方法中，设置了标签的图标。代码如下：

```
public class ImageLabel extends JLabel implements
ChangeListener {
    private static final long serialVersionUID
  =-5189246904858249548L;
    private JSpinner spinner;
    private ImageIcon image;
```

```
    public ImageLabel(JSpinner spinner) {
      this.spinner = spinner;
      this.image= (ImageIcon)
  spinner.getValue();                    //获得微调控件模型中保
  存的图标
      spinner.addChangeListener(this);
      //为微调控件增加监听
    }
    @Override
    publicvoid stateChanged(ChangeEvent e)
{                          //对微调控件的变化事件做出响应，更换图
标
      image = (ImageIcon) spinner.getValue();
      setIcon(image);
    }
    @Override
    public Icon getIcon()
{                                      //获得图标
      return image;
    }
  }
```

（2）编写方法do_this_windowActivated()，用来监听窗体激活事件。在该方法中，为微调控件设置了新的模型和模型显示控件。核心代码如下：

```
  protected void do_this_windowActivated(WindowEvent e) {
    ImageIcon[] images=new
ImageIcon[6];                          //利用数组来保存图片
```

```
    for (int i = 0; i < images.length; i++) {
      images[i]=new ImageIcon("src/images/"+ (i+1)+
".png");
    }
    spinner.setModel(newSpinnerListModel(images));
      //设置微调控件模型
    spinner.setEditor(new
ImageLabel(spinner));                    //设置微调控件模型
的显示方式
  }
```

举一反三

根据本实例，读者可以实现以下功能。

使用微调控件浏览文件。

使用微调控件浏览指定格式的文件。

2.12 自定义控件

实例111 石英钟控件

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_111

实例说明

程序设计的GUI界面要包含的信息很多，其中日期和时间都可以通过标签控件以文字的方式显示，但是拥有完整的控件集才能为程序开发添砖加瓦。为此本实例自定义了一个显示时钟控件，这个控件在显示时钟的同时还可以显示其覆盖的控件，因为它是背景透明的控件。实例运行效果如图2.67所示。

图2.67 石英钟控件在窗体中的显示效果

技术要点

本实例的关键技术在于绘图上下文的透明合成规则。这在Java中是通过AlphaComposite类来实现的，该类可以实现很多不同的透明合成规则，本实例只用到了SRC_OVER 规则。本实例中设置透明合成规则的相关代码如下：

```
public void paint(Graphics g) {
```

```
    Graphics2Dg2= (Graphics2D)
g.create();                                    //转换为2D绘图上
下文
    Composite
composite=g2.getComposite();                                   /
/保存原有合成规则
    g2.setComposite(AlphaComposite.SrcOver.derive(0.6f));
                    //设置60%透明的合成规则
    Calendar calendar = Calendar.getInstance();
    drawClock(g2,
calendar);                                                     //绘
制时钟
    g2.setComposite(composite);
            //恢复原有合成规则
    g2.drawImage(background.getImage(),0,0,
this);                                 //绘制背景图
    g2.dispose();
  }
```

实现过程

（1）在项目中新建窗体类ClockFrame。设置窗体的标题、大小和位置等属性，并把自定义的石英钟控件添加到窗体中。

（2）继承JLabel类编写石英钟控件，该控件代码段的关键在于drawClock()方法的实现，控件通过该方法绘制石英钟界面。关键代码如下：

```
  private void drawClock(Graphics2D g2, Calendar calendar)
{
    int millisecond = calendar.get(MILLISECOND);
```

```
    int sec = calendar.get(SECOND);
    int minutes = calendar.get(MINUTE);
    int hours = calendar.get(HOUR);
    double secAngle= (60 - sec) * 6 - (millisecond /
150);                          //秒针角度
    intminutesAngle= (60 -minutes) *
6;                                      //分针角度
    int hoursAngle= (12 - hours) * 360 / 12 - (minutes /
2);                           //时针角度
    //计算秒针、分针、时针指向坐标
    int secX = (int) (secLen *
Math.sin(Math.toRadians(secAngle)));
    int secY = (int) (secLen *
Math.cos(Math.toRadians(secAngle)));
    int minutesX = (int) (minuesLen *
Math.sin(Math.toRadians(minutesAngle)));
    int minutesY = (int) (minuesLen *
Math.cos(Math.toRadians(minutesAngle)));
    int hoursX = (int) (hoursLen *
Math.sin(Math.toRadians(hoursAngle)));
    int hoursY = (int) (hoursLen *
Math.cos(Math.toRadians(hoursAngle)));
    g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,Ren
deringHints.VALUE_ANTIALIAS_ON);
    //分别绘制时针、分针、秒针
    g2.setColor(Color.BLACK);
    g2.setStroke(HOURS_POINT_WIDTH);
```

```
        g2.drawLine(centerX, centerY, centerX - hoursX, centerY
    - hoursY);
        g2.setStroke(MINUETES_POINT_WIDTH);
        g2.setColor(new Color(0x2F2F2F));
        g2.drawLine(centerX, centerY, centerX - minutesX,
    centerY - minutesY);
        g2.setColor(Color.RED);
        g2.setStroke(SEC_POINT_WIDTH);
        g2.drawLine(centerX, centerY, centerX - secX, centerY -
    secY);
        //绘制3个指针的中心圆
        g2.fillOval(centerX - 5, centerY - 5, 10, 10);
    }
```

举一反三

根据本实例，读者可以实现以下功能。

制作一个电子钟。

为石英钟添加背景。

实例112 IP输入文本框控件

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_112

实例说明

文本框可以接收用户输入，但是用在复合数据的输入方面就没有那么方便了，如在文本框中输入系统IP地址，用户要输入完整的字符串表示的IP地址，这容易导致输入错误，而且程序还要把字符串转换为InetAddress类的对象来应用于网络程序，本实例把用户输入约束和

生成InetAddress类的步骤集成在一起，开发了一个IP输入文本框控件，该控件应用于窗体中的效果如图2.68所示。

图2.68 实例运行效果

技术要点

本实例的关键技术在于屏蔽文本框的非数字输入值。这需要利用按键事件监听器来实现，在监听器的keyTyped方法中通过字符串的索引查询结果，来判断输入字符是否符合规范字符串的要求。关键代码如下：

```
public void keyTyped(KeyEvent e) {
  if (("0123456789"+ (char) 8).indexOf(e.getKeyChar()) <
0) {
    e.consume();                                          /
/屏蔽非数字与回退键的输入
    return;
  }
  //省略其他代码……
}
```

实现过程

（1）在项目中新建窗体类 IPFrame。设置窗体的标题、大小和位置等属性。同时在窗体中添加文本框、按钮和自定义的IP文本框控

件，组成一个设置服务器地址的界面。关键代码如下：

```
    JLabel label=new JLabel("设置服务器名称与IP地
址");                  //创建标题标签
    label.setHorizontalAlignment(SwingConstants.CENTER);
          //居中对齐
    label.setFont(newFont("SansSerif",Font.PLAIN,18));
          //设置字体
    label.setBounds(6, 6, 298, 39);
    contentPane.add(label);
    JLabel label_1=new JLabel("服务器名
称：");                        //创建标签
    label_1.setBounds(6,57,83,18);
      //设置标签大小
    contentPane.add(label_1);
    JLabel label_2=new JLabel("服务器
IP：");                          //创建标签
    label_2.setBounds(6,95,83,18);
      //设置标签大小
    contentPane.add(label_2);
    textField=new
JTextField();                                    //创建输入服务
器名称的文本框
    textField.setBounds(82, 51, 251, 30);
    JButton button=new JButton("确
定");                                  //创建“确定”按钮
    contentPane.add(textField);
    textField.setColumns(10);
```

```
    button.setBounds(54, 132, 90, 30);
    contentPane.add(button);
    JButton button_1=new JButton("关
闭");                                    //创建“关闭”按钮
    button_1.setBounds(177, 132, 90, 30);
    contentPane.add(button_1);
    IpField ipField=new
IpField();                                          //创建IP文本框控件
    ipField.setBounds(82,88,251,25);
      //设置控件大小
    contentPane.add(ipField);
```

（2）自定义IP文本框需要用到4个文本框并分别输入每个地址段的数字，在这之前需要定义符合要求的待用文本框对象。所以要继承 JTextField 类实现自己需要的文本框，除在构造方法中对文本框做初始设置之外，还要通过按键事件监听器屏蔽文本框的非数字输入。关键代码如下：

```
    public CText() {
      setBorder(null);
    //取消边框
      setHorizontalAlignment(SwingConstants.CENTER);
        //文本居中
      setFont(getFont().deriveFont(16f));
        //绘制默认16号字体
      addKeyListener(newKeyAdapter()
  {                                  //添加按键事件监听器
        @Override
        public void keyTyped(KeyEvent e) {
```

```
    if (("0123456789"+ (char)
8).indexOf(e.getKeyChar())<0) {
      e.consume();                                    //屏
  蔽非数字与回退键的输入
      return;
    }
    if (e.getKeyChar() == (char) 8) {
      return;                                         //屏蔽
  回退键
    }
    String text=getText()+
e.getKeyChar();                  //获取最新输入
    if (!text.isEmpty())
{                                //如果输入非空
      int value=
  Integer.parseInt(text);                  //把输入解析
  为整数
      if (value>225) {                                //如
  果整数大于225
        e.consume();                                  //取消
  本次输入
        return;
      }
    }
    //如果输入文本过长或输入的是dot字符
    if (getText().length()>2 || e.getKeyChar()== '.') {
```

```
            e.consume();                                    //取
        消本次输入
            transferFocus();                                //
        把输入焦点传递给下一个控件
            return;
          }
        }
        @Override
        public void keyPressed(KeyEvent e) {
          //屏蔽粘贴快捷键
          if (e.getKeyCode() == KeyEvent.VK_V &&
      e.isControlDown()) {
            e.consume();
          }
        }
      });
    }
```

（3）继承JPanel编写自己的IP地址输入文本框控件，在构造方法中对控件大小和边框进行初始化，然后添加4个自定义的CText文本框控件。关键代码如下：

```
    public IpField() {
      setPreferredSize(newDimension(141,25));
            //设置控件初始首选大小
      setBorder(UIManager.getBorder("TextField.border"));
                  //采用文本框默认的边框
      setBackground(UIManager.getColor("TextField.background"
  ));          //采用文本框默认的背景色
```

```
        setSize(200,25);
          //初始大小
        setLayout(newBoxLayout(this,BoxLayout.X_AXIS));
              //设置布局管理器
        textField=newCText();
          //创建自定义文本框
        add(textField);
          //添加文本框到面板
        JLabel label=new
JLabel(".");                                    //创建IP分隔
符的标签控件
        add(label);
        textField_1=newCText();
          //创建自定义文本框
        add(textField_1);
          //添加文本框到面板
        JLabel label_3=new
JLabel(".");                                    //创建IP分隔符
的标签控件
        add(label_3);
        textField_2=newCText();
          //创建自定义文本框
        add(textField_2);
        JLabel label_2=new
JLabel(".");                                    //创建IP分隔符
的标签控件
        add(label_2);
```

```
    textField_3=newCText();
      //创建自定义文本框
    add(textField_3);
      //添加文本框到面板
    setFocusTraversalPolicy(new FocusTraversalOnArray(new
  Component[]
      { textField, textField_1, textField_2, textField_3
    }));
    }
```

（4）为自定义控件编写获取字符串IP地址值的方法，这是为控件提供获取值的途径，如果缺少则控件只能做显示用。关键代码如下：

```
    publicString getIpString()
{                                          //编写获取IP字符串值
的方法
      String ipstr= textField+ "."+ textField_1+ "."+
  textField_2+ "."+
  textField_3;                                          //把4个
  文本框的值连接为IP地址字符串
      return ipstr;
    }
```

（5）编写获取InetAddress类型的IP地址对象，这是控件获取IP值的另一种途径，获取的返回值是对象，更符合部分应用的需求。关键代码如下：

```
    public InetAddress getIpAddress()
{                                      //编写获取IP对象的方法
      InetAddress
  ia=null;                                          //创建一
```

个空的IP地址对象

```
    try {
      ia=
  InetAddress.getByName(getIpString());
    //把字符串转换为IP地址对象
    } catch (UnknownHostException e) {
      e.printStackTrace();
          //处理异常
    }
    return
ia;                                                    //返回
```

地址对象

```
  }
```

举一反三

根据本实例，读者可以实现以下功能。

判断文本框内容是否合法。

判断输入的IP地址是否正确。

实例113 日历控件

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_113

实例说明

日历控件既是日期类型的显示控件，又是日期类型的输入控件，用户可以通过单击控件上的按钮与日期来改变日期控件的值，在Java语言中，Swing并没有提供这样一个日历控件的实现。本实例通过自定义的方式实现了自己的日历控件，并为控件实现了事件监听，如图

2.69所示。通过日历控件的单击修改日期事件会改变右侧标签控件上显示的文本。

图2.69 日历控件在窗体程序中的应用

技术要点

本实例的关键技术在于 Timer 控件的应用。该控件能够在指定的时间间隔内重复执行Action。控件的ActionListener监听器将不断地捕获和处理该事件。创建一个Timer控件的构造方法的声明如下：

```
public Timer(int delay, ActionListener listener)
```

参数说明

- delay：初始延迟和动作事件间延迟的毫秒数。
- listener：初始侦听器，可以为null。

实现过程

（1）在项目中新建窗体类CalendarFrame。设置窗体的标题、大小和位置等属性。同时将自定义的日历控件添加到窗体，并添加一个显示日历控件当前时间值的标签控件。关键代码如下：

```
contentPane.setLayout(null);
        //使用绝对定位布局
calendarPanel=newCalendarPanel();
    //创建日历控件
```

```
    calendarPanel.addDateChangeListener(newPropertyChangeList
ener() {
      publicvoid propertyChange(PropertyChangeEvent evt) {
        do_calendarPanel_propertyChange(evt);
            //调用事件处理方法
      }
    });
    calendarPanel.setBounds(6, 6, 162, 170);
    contentPane.add(calendarPanel);
    //创建字符串模板
    InfoStr ="<html>您选择的日期是：<br><font size=6
color=red>%1s</font></html>";
    //设置标签控件显示日期
    label = new JLabel(String.format(InfoStr,
calendarPanel.getDate()));
    label.setBounds(180, 6, 162, 170);
    contentPane.add(label);
```

（2）通过自定义日历控件的事件监听器改变标签控件中的时间值。该事件处理方法的关键代码如下：

```
    protected void
do_calendarPanel_propertyChange(PropertyChangeEvent evt) {
      //通过事件更新标签控件的日期
      label.setText(String.format(InfoStr,
  calendarPanel.getDate()));
    }
```

（3）创建CalendarPanel类实现自定义的日历控件，并实现定义界面的关键方法getJPanel1()，在该方法中创建日历控件中的星期标

题和日期按钮。关键代码如下：

```
    private JPanel getJPanel1()
{                                                    //创建星期标题和
日期按钮
      if (jPanel1 == null) {
        GridLayout gridLayout2 = new GridLayout();
        gridLayout2.setColumns(7);
        gridLayout2.setRows(0);
        jPanel1=new
    JPanel();                                        //创建
    面板
        jPanel1.setOpaque(false);
        jPanel1.setLayout(gridLayout2);
                //设置布局管理器
        JLabel[]week=new
    JLabel[7];                                       //标题数组
        week[0]=new
    JLabel("日");                                    //星期
    标题
        week[0].setForeground(Color.MAGENTA);
                //特色颜色值
        week[1]=new
    JLabel("一");                                    //初始
    化其他星期标题
        week[2] = new JLabel("二");
        week[3] = new JLabel("三");
        week[4] = new JLabel("四");
```

```
    week[5] = new JLabel("五");
    week[6] = new JLabel("六");
    week[6].setForeground(Color.ORANGE);
            //为周六设置特色颜色值
    for (JLabel theWeek :week)
{                                          //初始化所有标题标签
      //文本居中对齐
      theWeek.setHorizontalAlignment(SwingConstants.CENTE
  R);
      Font font=
  theWeek.getFont();                                    //获
  取字体对象
      Font deriveFont=
  font.deriveFont(Font.BOLD);                          //字体加
  粗样式
      theWeek.setFont(deriveFont);
              //更新标签字体
      String info = theWeek.getText();
      if (!info.equals("日")&&
  !info.equals("六"))                          //改变周六周日
  前景色
        theWeek.setForeground(Color.BLUE);
      getJPanel1().add(theWeek);
    }
    days=new JLabel[6]
[7];                                          //创建日期控
件按钮（有标签实现）
```

```
    for (int i = 0; i < 6; i++) {
      for (int j=0; j<7; j++)
    {                                          //初始化每个日期按
    钮
          days[i][j] = new JLabel();
          //文本水平居中
          days[i]
      [j].setHorizontalTextPosition(SwingConstants.CENTER);
          //文本垂直居中
          days[i]
      [j].setHorizontalAlignment(SwingConstants.CENTER);
          days[i]
      [j].setOpaque(false);
      //控件透明
          days[i]
      [j].addMouseListener(dayClientListener);
          //添加事件监听器
          getJPanel1().add(days[i][j]);
        }
      }
      initDateField();
              //初始化日期文本框
      initDayButtons();
            //初始化日期按钮
    }
    return jPanel1;
}
```

举一反三

根据本实例，读者可以实现以下功能。

双击日期实现记录本功能。

在日历上添加待办事项提醒功能。

实例114 平移面板控件

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_114

实例说明

桌面应用程序开发中，容器的功能决定了它在界面设计中的重要性，Swing 中包含各种各样的容器，如分割面板、滚动面板、普通面板、分层面板、桌面面板等，其中滚动面板可以为容器添加滚动条，使其可以显示更多的内容。本实例作为这类面板的扩展，开发了更绚丽实用的平移面板，实例运行效果如图2.70所示。面板中在水平方向添加了多个控件，通过左右平移两个按钮可以动态调整显示内容。

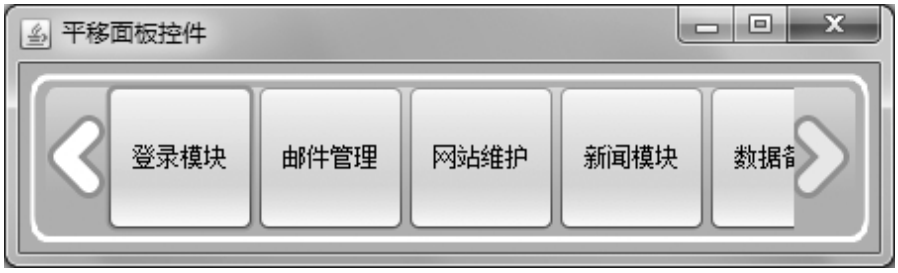


图2.70 实例运行效果

技术要点

本实例的关键技术在于控制滚动面板中滚动条的当前值，这需要获取滚动面板的滚动条与设置滚动条当前值的相关知识。下面分别进行介绍。

□ 获取滚动面板的水平滚动条

滚动面板包含水平和垂直两个方向的滚动条，通过适当的方法可以获取它们，下面的方法可以获取控制视口的水平视图位置的水平滚动条。其方法声明如下：

public JScrollBar getHorizontalScrollBar()

□ 获取滚动条当前值

滚动条的控制对象就是当前值，这个值控制着滚动条滑块的位置和滚动面板视图的位置。可以通过getValue()方法来获取这个值，其声明如下：

public int getValue()

□ 设置滚动条当前值

public void setValue(int value)

参数说明

value：滚动条新的当前值。

实现过程

（1）在项目中新建窗体类PanelFrame。设置窗体的标题、大小和位置等属性。同时将自定义的平移滚动面板添加到窗体中，并把包含多个按钮控件的面板设置为平移滚动面板的管理视图。关键代码如下：

```
public PanelFrame() {
  setTitle("平移面板控
件");                                          //设置窗体标
题
  setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  setBounds(100, 100, 450, 133);
  contentPane = new JPanel();
  contentPane.setBackground(new Color(102, 204, 204));
  contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
```

```
        contentPane.setLayout(newBorderLayout(0,
    0));                                    //设置布局管理器
        setContentPane(contentPane);
        //创建平移滚动面板
        SmallScrollPanel smallScrollPanel = new
    SmallScrollPanel();
        //添加面板到窗体
        contentPane.add(smallScrollPanel, BorderLayout.CENTER);
        ButtonPanel
    buttonPanel=newButtonPanel();                                    //
    创建按钮组面板
        buttonPanel.setOpaque(false);
        //把按钮组面板设置为平移面板的管理视图
        smallScrollPanel.setViewportView(buttonPanel);
      }
```

（2）编写SmallScrollPanel类，它是本实例自定义的平移面板控件，由于代码过多，这里只介绍关键技术，也就是左右微调按钮的事件监听器。关键代码如下：

```
    private final class ScrollMouseAdapter extends
MouseAdapter implements Serializable {
        //获取滚动面板的水平滚动条
        JScrollBar scrollBar =
    getAlphaScrollPanel().getHorizontalScrollBar();
        privateboolean isPressed=
    true;                                          //定义线程控制变
    量
        public void mousePressed(MouseEvent e) {
```

```
    Object source=
e.getSource();                                          //获取
事件源
    isPressed = true;
    //判断事件源是左侧按钮还是右侧按钮，并执行相应操作
    if (source == getLeftScrollButton()) {
      scrollMoved(-1);
    } else {
      scrollMoved(1);
    }
  }
  /**
  *移动滚动条的方法
  *
  *@param orientation
  *    移动方向-1是左或上移动，1是右或下移动
  */
  private void scrollMoved(final int orientation) {
    newThread()
{                                                       //开辟新
的线程
      //保存原有滚动条的值
      private int oldValue = scrollBar.getValue();
      public void run() {
        while (isPressed)
    {                                                   //循环移动面板
          try {
```

```
            Thread.sleep(1);
          } catch (InterruptedException e1) {
            e1.printStackTrace();
          }
          //获取滚动条当前值
          oldValue = scrollBar.getValue();
          EventQueue.invokeLater(new Runnable() {
            public void run() {
              //设置滚动条移动3个像素
              scrollBar.setValue(oldValue + 4 *
          orientation);
            }
          });
        }
      }
    }.start();
  }
  public void mouseExited(java.awt.event.MouseEvent e) {
    isPressed = false;
  }
  @Override
  public void mouseReleased(MouseEvent e) {
    isPressed = false;
  }
}
```

举一反三

根据本实例，读者可以实现以下功能。

实现动态系统托盘。

使用普通模板。

实例115 背景图面板控件

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_115

实例说明

JPanel是Swing的面板类，它作为一个控件容器，用于GUI界面的规划与设计，但是该控件没有提供对图片设置的支持，这就导致面板只能显示一个单一颜色的背景，难以实现界面美化的设计。本实例继承JPanel重写了控件绘制方法，实现了对背景图片的支持。实例运行效果如图2.71所示。

图2.71 背景面板添加按钮后的效果

技术要点

本实例的关键技术在于重写控件的绘制方法 paintComponent()，这个方法负责控件外观的绘制，通过重写这个方法可以把背景图片绘

制到控件界面上。该方法的声明如下：

protected void paintComponent(Graphics g)

参数说明

g：控件的绘图上下文对象。

实现过程

（1）在项目中继承JPanel类编写自定义的面板控件类BGPanel。设置控件的布局、初始大小等属性。

（2）重写控件的 paintComponent()方法，在该方法中完成控件原有外观的绘制的同时，根据自定义的填充属性来绘制背景图片。关键代码如下：

```
protected void paintComponent(Graphics g) {
  super.paintComponent(g);
          //完成原来控件外观的绘制
  if (image !=null)
{                                                    //开始自定
义背景的绘制
    switch (iconFill)
{                                                    //判断背景
填充方式
      caseNO_FILL:
        //不填充
        g.drawImage(image,0,0,
    this);                                  //绘制原始图片大小
        break;
      caseHORIZONGTAL_FILL:
      //水平填充
        //绘制与控件等宽的图片
```

```
            g.drawImage(image, 0, 0, getWidth(),
        image.getHeight(this),this);
            break;
          caseVERTICAL_FILL:
            //垂直填充
            //绘制与控件等高的图片
            g.drawImage(image, 0, 0, image.getWidth(this),
        getHeight(), this);
            break;
          caseBOTH_FILL:
            //双向填充
            //绘制与控件同等大小的图片
            g.drawImage(image, 0, 0, getWidth(), getHeight(),
        this);
            break;
          default:
            break;
        }
      }
    }
```

举一反三

根据本实例，读者可以实现以下功能。

更换背景图片。

用按钮实现主题切换。

2.13 控件渲染

实例116 支持图标的列表控件

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_116

实例说明

JList是Swing列表控件类。该控件可以在界面中显示一个文本列表，用户在该列表中可以选择特定的项，然后由其他业务处理程序对选项做条件判断与处理。功能实现了，但界面中只是单调显示文字列表项，这未免有些枯燥，不适合目前用户对GUI界面美观的追求。本实例利用渲染器的原理为JList列表控件实现了支持图标的选项。实例运行效果如图2.72所示。

图2.72 显示图标的列表控件

技术要点

本实例的关键技术在于 JList 列表控件的渲染器的创建与使用。列表控件的渲染器是ListCellRenderer 接口的实现，本实例实现这个接口编写自己的实现类，在实现该接口的getListCellRendererComponent()方法时可以创建指定的控件并根据方法参数对控件被选择、存在焦点等状态进行渲染。该方法在接口中的声明如下：

Component getListCellRendererComponent(JList list, Object value, int index, boolean isSelected, boolean cellHasFocus)

该方法中的参数说明如表2.31所示。

表2.31 getListCellRendererComponent方法的参数说明

参 数 名	作 用
list	要渲染的 JList 列表控件的引用
value	由 list.getModel().getElementAt(index)返回的值
index	单元格索引
isSelected	如果选择了指定的单元格，则为 true
cellHasFocus	如果指定的单元格拥有焦点，则为 true

实现过程

（1）在项目中新建窗体类IconList。设置窗体的标题、大小和位置等属性。

（2）在窗体类构造方法中创建JList列表控件，然后实现ListCellRenderer接口编写渲染器的实现对象，并把该对象作为JList控件的渲染器属性。关键代码如下：

```
final String[] values = new String[] {"西瓜", "吃剩的苹果", "香蕉", "玉米", "葡萄","菠萝", "西红柿"
};                                          //创建列表项数组
final ImageIcon[] icons=new
ImageIcon[values.length];                   //创建图标数组
```

```
    for (int i=0; i< icons.length; i++)
{                                        //遍历图标数组
      icons[i]=new ImageIcon(getClass().getResource("/res/"+
  i+ ".png"));                                    //初始化每一
  个数组元素
    }
    JList list=new
JList(values);                                          //创建
列表控件
    ListCellRenderer renderer=newListCellRenderer()
{                          //创建渲染器实现
      JLabel label=new
  JLabel();                                       //创建标签控件
      Color
  background=newColor(0,0,0,0);                          //创
  建透明的背景色
      @Override
      publicComponent getListCellRendererComponent(final
  JList list,Object value, int index,boolean
  isSelected,boolean cellHasFocus) {
        label.setBackground(background);
            //设置标签控件的背景色
        label.setOpaque(true);
            //使标签不透明
        if (value.equals(values[index])) {
          label.setText(value+
      "");                                  //设置标签文本
```

```
            label.setIcon(icons[index]);
                //设置标签图标
        }
        if (isSelected) {
            label.setBackground(Color.PINK);
              //设置选择时的背景色
        } else {
            label.setBackground(background);
              //设置未选择时的背景色
        }
        return
label;                                                    //返回
标签控件作为渲染控件
    }
};
list.setCellRenderer(renderer);
          //设置列表控件的渲染器
scrollPane.setViewportView(list);
          //把列表控件添加到滚动面板
```

举一反三

根据本实例，读者可以实现以下功能。

使用默认的列表控件渲染器显示其他文本。

使用控件模型的默认实现类。

实例117 在列表控件中显示单选按钮

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_117

实例说明

JList 控件可以设置列表项的选择方式，默认情况下支持多选操作，用户可以通过 Ctrl 或Shift功能键与鼠标的配合实现列表项的多选，也可以设置列表的选择方式为单选或限制连选。既然可以设置为单选，那么就类似单选按钮组的功能，同一时刻只能选择一个单选项，那么就可以把列表控件的选项渲染成单选按钮的样式。本实例通过列表控件的渲染器实现了这个效果。实例运行效果如图2.73所示。

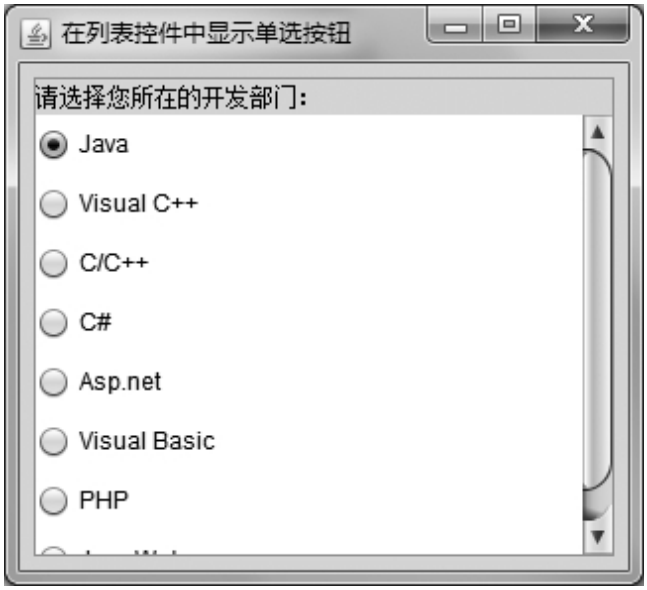


图2.73 实例运行效果

技术要点

本实例的关键技术请参见实例116。

实现过程

（1）在项目中新建窗体类RadioList。设置窗体的标题、大小和位置等属性。

（2）在窗体类构造方法中创建JList列表控件，然后实现ListCellRenderer接口编写渲染器的实现对象，并把该对象作为JList控件的渲染器属性。关键代码如下：

```
    finalString[]values=newString[] { "Java", "VisualC++",
"C/C++","C#", "Asp.net", "VisualBasic", "PHP", "JavaWeb"
};                  //创建列表项数组
    JList list=new
JList(values);                                          //创建
列表控件
    list.setSelectionMode(ListSelectionModel.SINGLE_SELECTION
);              //列表项单选
    list.setSelectedIndex(0);
            //设置默认选择状态的选项
    list.setFixedCellHeight(30);
            //设置列表项的固定高度
    ListCellRenderer renderer=newListCellRenderer()
{                       //创建渲染器实现
      JRadioButton radio=new
  JRadioButton();                           //创建单选按钮控件
      Color
  background=newColor(0,0,0,0);                              //创
  建透明的背景色
      @Override
      publicComponent getListCellRendererComponent(final
  JList list,Object value, int index,boolean
  isSelected,boolean cellHasFocus) {
        radio.setBackground(background);
            //设置单选按钮控件的背景色
        radio.setOpaque(true);
          //使单选按钮不透明
```

```
        if (value.equals(values[index])) {
          radio.setText(value+
    "");                                    //设置单选按钮文本
        }
        radio.setSelected(isSelected);
        return
    radio;                                                  //返回
    单选按钮控件作为渲染控件
      }
    };
    list.setCellRenderer(renderer);
            //设置列表控件的渲染器
    scrollPane.setViewportView(list);
            //把列表控件添加到滚动面板
```

举一反三

根据本实例，读者可以实现以下功能。

列表控件的透明属性。

列表控件中显示其他按钮控件。

实例118 列表控件折行显示列表项

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_118

实例说明

列表控件可以将多个选项显示在一个控件中，用户可以选择单独的选项或者多个选项。默认列表控件的列表项是垂直方向排列的，本实例将介绍如何让列表项水平方向显示，并且可以自动折行。实例运

行效果如图 2.74 所示。读者可以联想到 Windows 的资源管理器以图标方式显示文件时经常用到这种布局方式。

图2.74 折行显示列表项的运行效果

技术要点

本实例的关键技术在于设置JList控件的布局方向。另外，还要设置显示行数限制参数以使布局方向参数生效。下面分别进行介绍。

□ 设置列表控件布局方向

public void setLayoutOrientation(int layoutOrientation)

参数说明

layoutOrientation：新的布局方向，可选值包括 VERTICAL、HORIZONTAL_WRAP 或VERTICAL_WRAP。

□ 设置列表控件显示行数限制

public void setVisibleRowCount(int visibleRowCount)

参数说明

visibleRowCount：一个整数值，指示要显示的首选行数（不要求滚动）。

实现过程

（1）在项目中新建窗体类RadioList。设置窗体的标题、大小和位置等属性。

（2）在窗体类构造方法中创建JList列表控件，然后实现ListCellRenderer接口编写渲染器的实现对象，并把该对象作为JList

控件的渲染器属性。关键代码如下：

```
    finalString[]values=newString[15];
          //创建列表项数组
    for (int i = 0; i < values.length; i++) {
      values[i] ="选项"+ (i+1);
    }
    JList list=new
JList(values);                                          //创建
列表控件
    list.setLayoutOrientation(JList.HORIZONTAL_WRAP);
    list.setVisibleRowCount(-1);
    list.setSelectionMode(ListSelectionModel.SINGLE_SELECTION
);            //列表项单选
    list.setSelectedIndex(0);
           //设置默认选择状态的选项
    list.setFixedCellHeight(30);
           //设置列表项的固定高度
    ListCellRenderer renderer=newListCellRenderer()
{                        //创建渲染器实现
      JRadioButton radio=new
  JRadioButton();                          //创建单选按钮控件
      Color
  background=newColor(0,0,0,0);                              //创
  建透明的背景色
      publicComponent getListCellRendererComponent(final
  JList list,Object value, int index,boolean
  isSelected,boolean cellHasFocus) {
```

```
        radio.setBackground(background);
            //设置单选按钮控件的背景色
        radio.setOpaque(true);
          //使单选按钮不透明
        if (value.equals(values[index])) {
          radio.setText(value+
    "");                                    //设置单选按钮文本
        }
        radio.setSelected(isSelected);
        return
radio;                                                  //返回
单选按钮控件作为渲染控件
    }
};
list.setCellRenderer(renderer);
            //设置列表控件的渲染器
scrollPane.setViewportView(list);
            //把列表控件添加到滚动面板
```

举一反三

根据本实例，读者可以实现以下功能。

定义每行显示的个数。

实现折行显示的功能。

实例119 使用图片制作绚丽按钮

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_119

实例说明

按钮控件的外观由Swing的LookAndFeel指定，但是程序的美工设计经常要考虑整体设计效果，从而把按钮设计得更有特色。要实现美工设计的按钮界面，可能需要彻底摧毁原有的按钮界面，本实例就利用图片替换了按钮控件原有的界面效果，如图2.75所示。

图2.75 绚丽的登录按钮界面

技术要点

本实例的关键技术在于按钮属性的设置，其中包括取消按钮边框绘制、取消按钮内容绘制和取消按钮焦点绘制等。下面分别进行介绍。

□ 取消按钮边框绘制

public void setFocusPainted(boolean focusPaint)

参数说明

focusPaint：如果为true，则应绘制焦点状态，否则取消焦点的绘制。

□ 取消按钮内容绘制

public void setBorderPainted(boolean borderPaint)

参数说明

borderPaint：如果为true并且边框属性不为null，则绘制该边框。

□ 取消按钮焦点绘制

public void setContentAreaFilled(boolean fill)

参数说明

fill：如果为true，则应该填充内容；如果为false，则不填充内容区域。

□ 设置按钮图标

public void setIcon(Icon defaultIcon)

参数说明

defaultIcon：用作默认图像的图标。

□ 设置按钮的按下图标

public void setPressedIcon(Icon pressedIcon)

参数说明

pressedIcon：用作“按下”图像的图标。

□ 设置按钮的翻转图标

public void setRolloverIcon(Icon rolloverIcon)

参数说明

rolloverIcon：用作“翻转”图像的图标。

实现过程

（1）在项目中新建窗体类LoginFrame。设置窗体的标题、大小和位置等属性。

（2）在窗体类构造方法中创建JList列表控件，然后实现ListCellRenderer接口编写渲染器的实现对象，并把该对象作为JList控件的渲染器属性。关键代码如下：

```
public LoginFrame() {
  super();
```

```
    setTitle("使用图片制作绚丽按钮");
    //设置窗体内容面板
    jContentPane = new JPanel();
    //设置布局管理器
    jContentPane.setLayout(new BorderLayout());
    loginPanel = new LoginPanel();
    loginPanel.setLayout(null);
    JButton loginButton = new JButton();
    loginButton.setBounds(266, 81, 68, 68);
    loginButton.setFocusPainted(false);
    loginButton.setBorderPainted(false);
    //设置按钮图标
    loginButton.setIcon(new
ImageIcon(getClass().getResource("/com/lzw/logBut1.png")));
    loginButton.setContentAreaFilled(false);
    //设置按钮按下动作的图标
    loginButton.setPressedIcon(new
ImageIcon(getClass().getResource(
"/com/lzw/logBut2.png")));
    //设置鼠标经过按钮的图标
    loginButton.setRolloverIcon(new
ImageIcon(getClass().getResource(
"/com/lzw/logBut3.png")));
    //添加按钮事件监听器
    loginPanel.add(loginButton);
            //添加登录按钮
```

```
    textField=new
JTextField();                                          //创建
文本框
    textField.setBounds(94, 81, 155, 30);
    loginPanel.add(textField);
                //添加文本框到窗体
    passwordField=new
JPasswordField();                                      //创建密
码框
    passwordField.setBounds(94, 113, 155, 30);
    loginPanel.add(passwordField);
                //添加密码框到窗体
    //添加登录面板到内容面板
    jContentPane.add(loginPanel, BorderLayout.CENTER);
    this.setContentPane(jContentPane);
    //设置窗体大小
    setSize(newDimension(513,248));
            //调用初始化界面的方法
    setLocationRelativeTo(null);
                //窗体居中
  }
```

举一反三

根据本实例，读者可以实现以下功能。

制作水晶按钮。

用图片代替按钮背景。

实例120 实现按钮关键字描红

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_120

实例说明

按钮控件用于执行 UI 界面中的控制命令，功能虽强大，但是显示文本的能力不足，只能显示指定字体与大小的文字，而且不能换行。Swing 为控件摆脱了这个陈旧的控件文本显示方式，可以像在网页中一样在控件中显示任意类型的文字。本实例就实现了按钮文字描红与换行的效果，如图2.76所示。

图2.76 按钮关键字描红与换行效果

技术要点

本实例的关键技术在于控件文本的设置，Swing 的控件不但可以设置普通的文本，它还支持HTML文本。也就是说，在控件中把文本属性设置为一个HTML代码是有效的。例如，本实例对按钮文本的设置。代码如下：

```
JButton button = new JButton("<html>"
  +"<body align=center>"
  +"<Font size=6 color=red>登录</font><br>"
  +"明日科技管理系统"
  +"</body>"
  + "</html>");                                    //创建按
```

钮控件并设置html文本

这个代码将会把HTML文本效果显示在界面中。

实现过程

（1）在项目中新建窗体类ButtonReadFont。设置窗体的标题、大小和位置等属性。

（2）在窗体类构造方法中添加文本框与密码框，最重要的是添加一个足够大的按钮控件，然后在按钮控件中设置HTML文本，使按钮可以显示关键字描红的UI界面。关键代码如下：

```
    JLabel label=new JLabel("用户
名：");                          //创建标签
    label.setBounds(20, 23, 55, 18);
    contentPane.add(label);
    textField=new
JTextField();                                  //创建文本框
    textField.setBounds(75, 17, 122, 30);
    contentPane.add(textField);
    textField.setColumns(10);
    JLabel label_1=new JLabel("密
码：");                          //创建标签
    label_1.setBounds(20, 72, 55, 18);
    contentPane.add(label_1);
    passwordField=new
JPasswordField();                          //创建密码框
    passwordField.setBounds(75, 66, 122, 30);
    contentPane.add(passwordField);
    JButton button = new JButton("<html>"
      +"<body align=center>"
      +"<Font size=6 color=red>登录</font><br>"
      +"明日科技管理系统"
```

```
        +"</body>"
        + "</html>");                                    //创建按
```
钮控件并设置html文本

```
    button.setBounds(209, 23, 141, 76);
    contentPane.add(button);
```

举一反三

根据本实例，读者可以实现以下功能。

实现可修改字体大小的功能。

实现自动换行功能。

实例121 忙碌的按钮控件

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_121

实例说明

控件可以设置鼠标位于其UI范围内时的光标，本实例利用这个特性实现了一个趣味界面，如图2.77所示，当用户单击“非常相信”按钮时虽然没有实现任何操作，但是一切都和从前的普通按钮一样。但是当用户准备单击“鬼才信呢”按钮之前，鼠标刚刚停留到按钮之上，鼠标的光标就显示忙碌状态了。

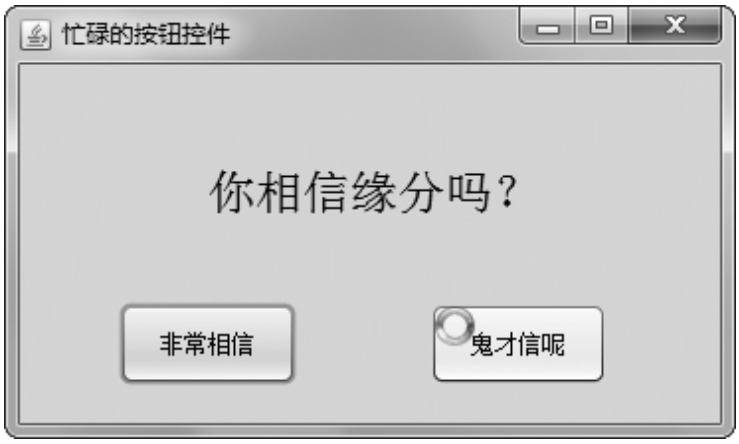


图2.77 Windows 7 下忙碌的按钮

技术要点

本实例的关键技术在于设置鼠标在指定按钮上的光标，这要通过具体按钮的cursor属性来设置。下面将介绍按钮控件设置鼠标光标的方法。

public void setCursor(Cursor cursor)

参数说明

cursor：Cursor类定义的常量之一。如果此参数为null，则此组件继承其父级的光标。

实现过程

（1）在项目中新建窗体类BusyButton。设置窗体的标题、大小和位置等属性。

（2）在窗体类构造方法中添加标签和两个按钮控件，在其中一个按钮控件的代码中设置鼠标光标属性为忙碌状态的光标。关键代码如下：

```
public BusyButton() {
  setTitle("忙碌的按钮控件");
  setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  setBounds(100, 100, 370, 219);
  contentPane = new JPanel();
  contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
  setContentPane(contentPane);
  contentPane.setLayout(null);
  JLabel label=new JLabel("你相信缘分
吗？");                                    //创建标签
  label.setHorizontalAlignment(SwingConstants.CENTER);
```

```
        label.setFont(newFont("SansSerif",Font.PLAIN,24));
                                        //设置标签字体
        label.setBounds(6, 32, 347, 66);
        contentPane.add(label);
        JButton button=new JButton("非常相
信");                                                    //创建按钮
        button.setBounds(50, 120, 90, 42);
        contentPane.add(button);
        JButton button_1=new JButton("鬼才信
呢");                                                    //创建忙碌按钮
        //设置按钮的鼠标光标为忙碌状态
        button_1.setCursor(Cursor.getPredefinedCursor(Cursor.WA
IT_CURSOR));
        button_1.setBounds(207, 120, 90, 42);
        contentPane.add(button_1);
    }
```

举一反三

根据本实例，读者可以实现以下功能。

鼠标光标悬浮于按钮上显示忙碌状态。

把鼠标光标替换成等待状态。

实例122 实现透明效果的表格控件

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_122

实例说明

程序开发经常利用漂亮的背景做界面美化，但是如果大面积的控件完全被那个背景遮盖，将破坏这个美观的设计。本实例以 UI 中面积较大的表格控件为例，实现透明的表格控件，让它可以显示底层的背景，这样程序看上去更漂亮，实例运行效果如图2.78所示。

图2.78 实例运行效果

技术要点

本实例的关键技术在于设置每个单元格的透明属性，单元格的控件是由表格内部控制的，所以要重写表格的某个方法，就要自定义表

格控件。本实例重写了表格的 prepareRenderer()方法，把渲染后的表格单元格控件设置为透明，该方法的声明如下：

public Component prepareRenderer(TableCellRenderer renderer, int row, int column)

参数说明

- renderer：要准备的TableCellRenderer。
- row：要呈现的单元格所在的行，其中第一行为0。
- column：要呈现的单元格所在的列，其中第一列为0。

实现过程

（1）在项目中新建窗体类LimpidityTable。设置窗体的标题、大小和位置等属性。

（2）在窗体类构造方法中向窗体添加面板和表格控件，其中添加表格控件时使用匿名类的方法自定义表格控件，并重写渲染方法透明显示表格的所有单元格。关键代码如下：

```
public LimpidityTable() {
  setTitle("实现透明效果的表格控
件");                                    //设置窗体标题
  setResizable(false);
          //禁止调整大小
  setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  setBounds(100, 100, 520, 549);
  contentPane = new JPanel();
  contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
  contentPane.setLayout(new BorderLayout(0, 0));
  setContentPane(contentPane);
  ImgPanel imgPanel=new
ImgPanel();                                  //创建图片面板
```

```
    contentPane.add(imgPanel, BorderLayout.CENTER);
    imgPanel.setLayout(null);
              //取消布局管理器
    table=new JTable()
{                                                    //创建自定义
表格
      {
        setOpaque(false);
            //初始化表格为透明
        setGridColor(Color.MAGENTA);
              //设置表格网格颜色
        setShowVerticalLines(true);
              //显示网格竖线
        setShowHorizontalLines(true);
              //显示网格横线
        setRowHeight(20);
            //设置表格行高
        setBorder(newLineBorder(Color.PINK));
              //设置边框
        setForeground(Color.BLACK);
            //设置表格文字颜色
        setFont(newFont("SansSerif",Font.PLAIN,18));
                  //设置表格单元格字体
      }
      @Override
      publicComponent prepareRenderer(TableCellRenderer
  renderer,int row, int column)
```

```
    {                                          //重写渲染方法
        //获取渲染后的控件
        Component component =
    super.prepareRenderer(renderer, row, column);
        ((JComponent)
    component).setOpaque(false);                          //设
    置控件透明
        return
    component;                                             //
    返回控件
      }
    };
    table.setModel(newDefaultTableModel(newObject[][]
{                        //初始化表格内容与列名
      {"Java", "Java", "Java", "Java", "Java"},
      {"Java", "Java", "Java", "Java", "Java"},
      {"Java", "Java", "Java", "Java", "Java"},
      {"Java", "Java", "Java", "Java", "Java"},
      {"Java", "Java", "Java", "Java", "Java"},
      {"Java", "Java", "Java", "Java", "Java"},
      {"Java", "Java", "Java", "Java", "Java"},
      {"Java", "Java", "Java", "Java", "Java"},
      {"Java", "Java", "Java", "Java", "Java"},
      {"Java", "Java", "Java", "Java", "Java"},
      {"Java", "Java", "Java", "Java", "Java"}
    }, new String[] {"列名1", "列名2", "列名3", "列名4",
"列名5"}));
```

```
        table.setBounds(40,161,421,254);
                    //设置表格大小
        imgPanel.add(table);
        JPanel panel=new
    JPanel();                                                    //创建表
    头面板
        panel.setLayout(new BorderLayout(0, 0));
        panel.add(table.getTableHeader(),BorderLayout.CENTER);
                        //添加表头
        panel.setBounds(40, 126, 421, 34);
        imgPanel.add(panel);
    }
```

举一反三

根据本实例，读者可以实现以下功能。

更改每个单元格的透明属性。

实现表格的透明效果。

实例123 在表格中显示工作进度百分比

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_123

实例说明

表格用于显示复合数据，其中可以指定表格的表头和表文，这在Swing控件中是以表头和单元格控件进行显示的。默认的表格控件完全是以文本方式显示目标数据，本实例为表格控件设置了自定义的渲染器，实现了表格中以进度条显示百分比的界面效果，如图2.79所示。

图2.79 实例运行效果

技术要点

本实例的关键技术在于实现TableCellRenderer接口编写自己的渲染器。这个接口中定义了getTableCellRendererComponent()方法，这个方法将被表格控件回调来渲染指定的单元格控件。重写这个方法并在方法体中控制单元格的渲染就可以把进度条作为表格的单元格控件。该方法的声明如下：

Component getTableCellRendererComponent(JTable table, Object value, boolean isSelected, boolean hasFocus, int row, int column)

该方法中的参数说明如表2.32所示。

表2.32 getTableCellRendererComponent方法的参数说明

参 数 名	作 用
table	要求渲染器绘制的 JTable，可以为 null
value	要呈现的单元格的值。由具体的渲染器解释和绘制该值。例如，如果 value 是字符串“true”，则它可呈现为字符串，或者也可呈现为已选中的复选框。null 是有效值
isSelected	如果使用选中样式的高亮显示来呈现该单元格，则为 true；否则为 false
hasFocus	如果为 true，则适当地呈现单元格。例如，在单元格上放入特殊的边框，如果可以编辑该单元格，则以彩色呈现它，用于指示正在进行编辑
row	要绘制的单元格的行索引。绘制头时，row 值是-1
column	要绘制的单元格的列索引

实现过程

（1）在项目中新建窗体类TablePercent。设置窗体的标题、大小和位置等属性。

（2）在窗体类构造方法中向窗体添加面板和表格控件，为表格设置数据模型和渲染器，这个渲染器将对第4列表格单元格进行渲染，渲染结果是使用进度条显示整数为百分比。关键代码如下：

```
public TablePercent() {
    setTitle("在表格中显示工作进度百分
比");                                          //设置窗体标题
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(100,100,470,300);
                  //设置窗体位置与大小
    contentPane=new
JPanel();                                                  //创
建内容面板
    contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
    contentPane.setLayout(new BorderLayout(0, 0));
    setContentPane(contentPane);
    JScrollPane scrollPane=new
JScrollPane();                                        //创建滚动
面板
```

```
    contentPane.add(scrollPane,BorderLayout.CENTER);
                                //添加滚动面板到窗体
    table=new
JTable();
  //创建表格控件
    table.setModel(newDefaultTableModel(newObject[][]
{                                   //设置表格数据模型
      {"油田管理系统登录模块", "李某", "应用程序", new
  Integer(93) },
      {"油田管理系统部门模块", "张某", "应用程序", new
  Integer(63) },
      {"油田管理系统业务模块", "刘某", "应用程序", new
  Integer(73) },
      {"油田管理系统统计模块", "王某", "应用程序", new
  Integer(43) },
      {"油田管理系统登录模块", "李某", "应用程序", new
  Integer(93) },
      {"油田管理系统部门模块", "张某", "应用程序", new
  Integer(63) },
      {"油田管理系统业务模块", "刘某", "应用程序", new
  Integer(73) },
      {"油田管理系统统计模块", "王某", "应用程序", new
  Integer(43) },
      {"油田管理系统登录模块", "李某", "应用程序", new
  Integer(93) },
      {"油田管理系统部门模块", "张某", "应用程序", new
  Integer(63) },
```

```
        {"油田管理系统业务模块", "刘某", "应用程序", new
    Integer(73) },
        {"油田管理系统统计模块", "王某", "应用程序", new
    Integer(43) },
        {"油田管理系统报表模块", "误某", "应用程序", new
    Integer(53) }},
        new String[] {"项目名称", "项目负责人", "项目类型",
    "开发进度"}));
      table.getColumnModel().getColumn(0).setPreferredWidth(1
46);                          //设置列宽
      TableColumn column=
table.getColumnModel().getColumn(3);                          //
获取表格第4列对象
      column.setCellRenderer(newTableCellRenderer()
{                                  //设置第4列的渲染器
        @Override
        public Component getTableCellRendererComponent(JTable
    table, Object value, boolean isSelected, boolean
    hasFocus, int row, int column) {
          if (value instanceof Integer)
      {                                          //创建整数渲染控
      件
            JProgressBar bar=new
        JProgressBar();                              //创建进度
        条
            Integer percent= (Integer)
        value;                                     //把当前值转换
```

```
        为整数
            bar.setValue(percent);
                        //设置进度条的值
            bar.setStringPainted(true);
                        //显示进度条文本
            return
        bar;                                                    /
        /把进度条作为渲染控件
          } else {
            return null;
          }
        }
      });
      scrollPane.setViewportView(table);
                        //把表格添加到滚动面板
}
```

举一反三

根据本实例，读者可以实现以下功能。

在表格中显示进度百分比。

100%后提示完成功能。

实例124 在表格中显示图片

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_124

实例说明

表格用于显示复合数据，虽然复合数据的类型可以多种多样，但是在表格中只能以字符串文本来显示。有些 UI 界面需要根据程序的需求在表格中体现特殊数据的另类表现形式，其中以图片显示数据标识就非常常用。本实例为普通的表格控件添加了渲染器，实现表格中显示图片的效果，如图2.80所示。

图2.80 实例运行效果

技术要点

本实例的关键技术请参见实例123。

实现过程

（1）在项目中新建窗体类TableImage。设置窗体的标题、大小和位置等属性。

（2）在窗体类构造方法中添加滚动面板与表格控件，然后为表格控件添加数据模型与渲染器，在渲染器的实现中，对表格的第一列数据以标签控件渲染，并且把数据模型中的图标对象显示在标签控件中。关键代码如下：

```
public TableImage() {
```

```
    setTitle("在表格中显示图
片");                                                    //设置窗体
标题
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(100,100,470,300);
                    //设置窗体位置与大小
    contentPane=new
JPanel();                                                  //创
建内容面板
    contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
    contentPane.setLayout(new BorderLayout(0, 0));
    setContentPane(contentPane);
    JScrollPane scrollPane=new
JScrollPane();                                          //创建滚动
面板
    contentPane.add(scrollPane,BorderLayout.CENTER);
                        //添加滚动面板到窗体
    table=new
JTable();
  //创建表格控件
    ImageIcon[] icons = new ImageIcon[12];
    for (int i = 0; i < icons.length; i++) {
      icons[i] = new ImageIcon(getClass().getResource(
  "/res/"+ (i + 1) +".png"));
    }
    table.setModel(new DefaultTableModel(
```

```
    newObject[][]
{                                                    //设置表
格数据模型
    { icons[0], "油田管理系统部门模块", "李某", "应用程
序"},
        { icons[0], "油田管理系统部门模块", "张某", "应用程
    序"},
        { icons[1], "油田管理系统业务模块", "刘某", "应用程
    序"},
        { icons[2], "油田管理系统统计模块", "王某", "应用程
    序"},
        { icons[3], "油田管理系统登录模块", "李某", "应用程
    序"},
        { icons[4], "油田管理系统部门模块", "张某", "应用程
    序"},
        { icons[5], "油田管理系统业务模块", "刘某", "应用程
    序"},
        { icons[6], "油田管理系统统计模块", "王某", "应用程
    序"},
        { icons[7], "油田管理系统登录模块", "李某", "应用程
    序"},
        { icons[8], "油田管理系统部门模块", "张某", "应用程
    序"},
        { icons[9], "油田管理系统业务模块", "刘某", "应用程
    序"},
        { icons[10], "油田管理系统统计模块", "王某", "应用
    程序"},
```

```
        { icons[11], "油田管理系统报表模块", "误某", "应用
    程序"}},
      new String[] {"模块标识", "项目名称", "项目负责人",
  "项目类型"}));
    table.getColumnModel().getColumn(1).setPreferredWidth(1
46);                        //设置列宽
    TableColumn column=
table.getColumnModel().getColumn(0);                        //
获取表格第4列对象
    table.setRowHeight(32);
    column.setCellRenderer(newTableCellRenderer()
{                               //设置第4列的渲染器
      @Override
      public Component getTableCellRendererComponent(JTable
  table, Object value, boolean isSelected, boolean
  hasFocus, int row, int column) {
        ImageIcon icon = (ImageIcon) value;
        JLabel label=new
    JLabel(icon);                                  //创建进度条
        label.setBackground(table.getSelectionBackground())
    ;
        if
    (isSelected)                                                /
    /把选择的标签设置为不透明
          label.setOpaque(true);
        return
    label;                                                  //把
```

```
    进度条作为渲染控件
      }
      });
    scrollPane.setViewportView(table);
                          //把表格添加到滚动面板
}
```

举一反三

根据本实例，读者可以实现以下功能。

在表格中显示图片。

修改表格中的图片。

2.14 为控件添加动态效果

实例125 鼠标经过时按钮放大效果

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_125

实例说明

Swing 应用程序中的按钮控件本身有焦点效果和鼠标经过效果，但是根据个别项目的界面要求，可能需要突出鼠标范围内的控件。本实例实现了为按钮控件突出鼠标悬停效果，如图2.81所示。当用户把鼠标悬停在按钮控件上时，按钮会放大；而当用户把鼠标从按钮上移走时，按钮会恢复原始大小。

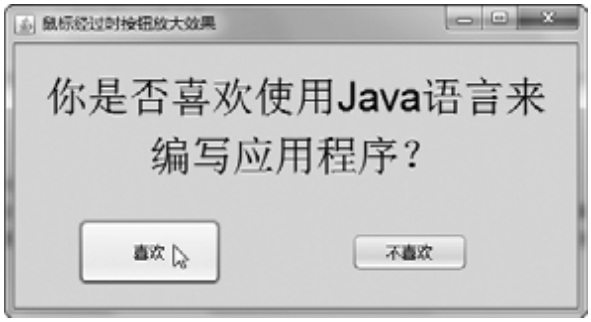


图2.81 实例运行效果

技术要点

本实例的关键技术在于鼠标事件适配器的创建，它是鼠标事件监听器接口的一个默认实现，它实现了接口的所有方法，但是没有为任何方法添加业务处理，而且它是一个抽象类，其用途主要是用于继承并重写需要的事件处理方法，避免代码对大部分不需要的方法进行空实现而浪费代码控件导致的代码混乱。鼠标事件监听器的适配器由

MouseAdapter类实现，读者可以继承该类并重写指定的事件处理方法，而不用实现所有的监听器接口方法。

实现过程

（1）在项目中新建窗体类MouseZoomButton。设置窗体的标题、大小和位置等属性。

（2）在窗体类构造方法中添加创建标签控件和两个按钮控件，然后为按钮控件编写鼠标事件监听器，并设置为两个按钮的监听器属性。当鼠标停留在按钮上时，监听器将放大按钮控件；如果鼠标离开按钮的区域，则按钮恢复原始大小。关键代码如下：

```
public MouseZoomButton() {
    setTitle("鼠标经过时按钮放大效
果");                                          //设置窗体标题
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(100,100,449,241);
                //设置窗体大小和位置
    contentPane = new JPanel();
    contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
    setContentPane(contentPane);
    contentPane.setLayout(null);
    //创建问题标签控件
    JLabel label=new JLabel("<html><body align=center>你是
否喜欢使用Java"+ "语言来<br>编写应用程序？</body></html>");
    label.setHorizontalAlignment(SwingConstants.CENTER);
                        //标签文本居中
    label.setFont(new Font("SansSerif", Font.PLAIN, 32));
    label.setBounds(6, 6, 421, 106);
    contentPane.add(label);
```

```
    JButton button=new JButton("喜
欢");                                          //创建按钮控件
    MouseAdaptermouseAdapter=newMouseAdapter()
{                                  //创建鼠标事件监听器
      privateRectangle
  sourceRec;                                                  //创
  建矩形对象
      @Override
      public void mouseEntered(MouseEvent e) {
        JButton button= (JButton)
    e.getSource();                                  //获取事件
    源按钮
        sourceRec=button.getBounds();
                  //保存按钮大小
        button.setBounds(sourceRec.x - 10, sourceRec.y -
    10,sourceRec.width+20,
    sourceRec.height+20);                    //把按钮放大
        super.mouseEntered(e);
      }
      @Override
      public void mouseExited(MouseEvent e) {
        JButton button= (JButton)
    e.getSource();                                  //获取事件
    源按钮
        if (sourceRec !=null)
    {                                    //如果有备份矩形则用它
    恢复按钮大小
```

```
            button.setBounds(sourceRec);
                        //设置按钮大小
        }
        super.mouseExited(e);
      }
    };
    button.addMouseListener(mouseAdapter);
                        //为按钮添加事件监听器
    button.setBounds(59,145,90,30);
                    //设置按钮大小
    contentPane.add(button);
    JButton button_1=new JButton("不喜
欢");                                          //创建按钮控件
    button_1.setBounds(259,145,90,30);
                        //绘制按钮初始大小
    button_1.addMouseListener(mouseAdapter);
                        //为按钮添加事件监听器
    contentPane.add(button_1);
  }
```

举一反三

根据本实例，读者可以实现以下功能。

实现鼠标经过时按钮放大效果。

实现鼠标经过时按钮更换背景色。

实例126 迟到的登录按钮

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_126

实例说明

本实例实现了按钮的移动动画，整个场景是一个系统的登录界面，当窗体处于激活状态时，按钮从左上角移动到右下角的位置，这个位置本来是“登录”按钮的正确位置，但是通过这个动画体现“登录”按钮是最后一个就位的界面控件，这个动画时间虽然短促，但是可以成功地把用户的注意力集中在登录界面中。实例运行效果如图2.82所示。

(a) 奔跑中的“登录”按钮

(b) 到位后的“登录”按钮

图2.82 实例运行效果

技术要点

本实例的关键技术在于按钮的移动，要实现这个技术，需要改变按钮的位置，这可以通过控件的setBounds()方法来实现。

改变按钮的位置和大小方法的声明如下：

```
public void setBounds(int x, int y, int width, int height)
```

该方法的参数说明如表2.33所示。

表2.33 方法的参数说明

参 数 名	作 用
x	控件的新 x 坐标
y	控件的新 y 坐标
width	控件的新宽度
height	控件的新高度

实现过程

（1）在项目中新建窗体类LoginFrame。设置窗体的标题、大小和位置等属性。

（2）在窗体类构造方法中添加标签控件、文本框控件、密码框控件和“登录”按钮。关键代码如下：

```
    JLabel label=new JLabel("天雨系统登录界
面");                                          //创建标签控件
    label.setHorizontalAlignment(SwingConstants.CENTER);
                                  //标签文本居中对齐
    label.setFont(newFont("SansSerif",Font.PLAIN,24));
                                  //设置标签控件字体
    label.setBounds(6, 6, 309, 51);
    contentPane.add(label);
    JLabel label_1=new JLabel("用户
名：");                                          //创建标签控件
    label_1.setBounds(16, 69, 55, 18);
    contentPane.add(label_1);
    JLabel label_2=new JLabel("密
码：");                                          //创建标签控件
    label_2.setBounds(16, 103, 55, 18);
    contentPane.add(label_2);
    textField=new
JTextField();
  //创建文本框
    textField.setBounds(65, 63, 242, 30);
    contentPane.add(textField);
    textField.setColumns(10);
                                  //设置文本框列数
```

```
    passwordField=new
JPasswordField();                                          //
创建密码框
    passwordField.setBounds(65, 99, 143, 30);
    contentPane.add(passwordField);
    button=new JButton("登
录");                                                    //创建“登录”
按钮但没有定位
    contentPane.add(button);
```

（3）编写窗体激活事件的处理方法，在该方法中创建匿名的线程对象，这个线程在循环中实现“登录”按钮的移动效果。关键代码如下：

```
    protected void do_this_windowActivated(WindowEvent e)
{                              //创建激活事件处理方法
      newThread()
  {                                                        //
  创建匿名线程
        @Override
        public void run() {
          for (int i=0; i<217; i++)
      {                                          //循环控制按钮
      的移动
            button.setBounds(i, i>99 ? 99 :
      i,90,30);                              //移动按钮
            getRootPane().setComponentZOrder(button,0);
                        //把按钮置顶显示
            try {
```

```
                sleep(1);
                    //线程休眠
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}.start();
                    //启动线程
}
```

举一反三

根据本实例，读者可以实现以下功能。

改变“登录”按钮的出场位置。

根据本实例可以制作退出按钮。

实例127 焦点按钮的缩放

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_127

实例说明

焦点是控件特有的属性，如当文本框处于输入状态时它是有焦点的，而且文本框的光标会不停地闪烁。这就说明当前的任何键盘输入在这个窗体中都是针对该文本框的，各种操作系统对于焦点的体现方式有所不同，本实例为焦点控件实现放大效果，使焦点能够更明显地呈现给用户。实例运行效果如图2.83所示。

图2.83 实例运行效果

技术要点

实例的关键技术在于焦点事件适配器的创建和窗体控件数组的获取。

焦点事件适配器是鼠标焦点监听器接口的一个默认实现，它实现了接口的所有方法，但是没有为任何方法添加业务处理，而且它是一个抽象类，其用途主要是用于继承并重写需要的事件处理方法，避免代码对大部分不需要的方法进行空实现而浪费代码控件导致的代码混乱。焦点事件监听器的适配器由FocusAdapter类实现，读者可以继承该类并重写指定的事件处理方法，而不用实现所有的监听器接口方法。

Swing的容器控件可以通过getComponents()方法获取容器中包含的所有控件组成的数组，通过遍历该数组可以对窗体中的控件进行统一设置，本实例就是通过该方法为窗体中的所有控件添加焦点事件监听器的。该方法的声明如下：

```
public Component[] getComponents()
```

该方法的返回值是当前容器中所有控件组成的数组。

实现过程

（1）在项目中新建窗体类ZoomControl。设置窗体的标题、大小和位置等属性。

（2）在窗体类构造方法中创建焦点事件适配器，实现控件获取焦点时放大、失去焦点时缩小的动作处理。然后获取窗体中所有控件并为它们添加该事件监听器。关键代码如下：

```
    focusAdapter=newFocusAdapter()
{                                    //创建焦点适配器
      privateRectangle
  sourceRec;                                    //创建矩形对
  象
      @Override
      public void focusGained(FocusEvent e) {
        JComponent component= (JComponent)
    e.getSource();                //获取事件源按钮
        sourceRec=
    component.getBounds();                              //保存按
    钮大小
        component.setBounds(sourceRec.x - 5, sourceRec.y -
    5,sourceRec.width+10, sourceRec.height+10);              //
    放大按钮
      }
      @Override
      public void focusLost(FocusEvent e) {
        JComponent component= (JComponent)
    e.getSource();                //获取事件源按钮
```

```
        if (sourceRec !=null)
    {                                          //如果有备份矩形则用
    它恢复按钮大小
          component.setBounds(sourceRec);
          //设置按钮大小
        }
      }
    };
    //获取窗体中的所有控件
    Component[] components =
getContentPane().getComponents();
    for (Component component : components)
{                                  //遍历所有控件
      component.addFocusListener(focusAdapter);
            //为所有控件添加焦点事件监听器
    }
```

举一反三

根据本实例，读者可以实现以下功能。

获得焦点时控件变大。

失去焦点时控件变回原来大小。

实例128 标签文本的跑马灯特效

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_128

实例说明

桌面程序开发中经常会有一些实时性的信息需要显示，这可以通过跑马灯的文本标签来实现，既可以显示提示信息，又可以通过动画达到醒目的效果。说起来有点像是网页上的广告，但是在桌面应用程序中，除了对话框以外，这也是一个实时信息提醒的好办法。实例运行效果如图2.84所示。

(a) 跑马灯起点

(b) 移动过程

图2.84 实例运行效果

技术要点

本实例的关键技术在于计算文本标签中应该插入空格的数量，这样才能够计算出当前窗体可以在单行中容纳多少字符。然后使用线程类动态添加空格字符，这样就形成了跑马灯的特效。这期间涉及窗体的一个获取宽度属性的方法getWidth()，该方法的声明如下：

```
public int getWidth()
```

实现过程

（1）在项目中新建窗体类LabelText。设置窗体的标题、大小和位置等属性。重要的是向窗体添加标签控件，并为窗体添加打开事件处理监听器。关键代码如下：

```
public LabelText() {
  addWindowListener(newWindowAdapter()
{                              //为窗体添加打开事件处理器
    @Override
    public void windowOpened(WindowEvent e) {
```

```
            do_this_windowOpened(e);
            //调用窗体打开事件处理方法
        }
    });
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            //设置窗体默认关闭方式
    setBounds(100,100,450,179);
        //设置窗体大小
    contentPane=new
JPanel();                                          //创建内容面板
    setContentPane(contentPane);
            //设置内容面板
    contentPane.setLayout(newBorderLayout(0,
0));                              //设置窗体布局
    label=new
JLabel("");                                              //创建标
签控件
    label.setHorizontalAlignment(SwingConstants.RIGHT);
            //文本右对齐
    contentPane.add(label);
            //添加标签到窗体
  }
```

（2）编写窗体打开事件的处理方法，在该方法中创建并启动自定义的线程对象，这个线程对象要完成窗体字符数量的计算和跑马灯动画特效。关键代码如下：

```
  protected void do_this_windowOpened(WindowEvent e) {
```

```
    newThread()
{                                                    //创建新的匿
名线程对象
      @Override
      publicvoid run()
  {                                                  //重写run()方法
        int
    len=getWidth()/12;                                          //
    获取跑马灯LED数量
        String info="Java编程词
    典";                                  //定义跑马灯文字
        while (true)
    {                                                  //创建无限循环
          String space=
      "";                                              //创建空白字符串
          for (int i=0; i< len - info.length()-2; i++)
      {                 //遍历LED数量
            len=getWidth()/12;                                   /
      /获取跑马灯LED数量
            space+= "  ";                                        //
      为空白字符串添加空格字符
            label.setText(info+
      space);                              //设置标签文本
            try {
              sleep(300);                                        //
      线程休眠
            } catch (InterruptedException e) {
```

```
                e.printStackTrace();
            }
        }
    }
  }
  }.start();
    //启动线程
}
```

举一反三

根据本实例，读者可以实现以下功能。

利用像素实现跑马灯。

控制跑马灯的滚动速度。

实例129 延迟生效的按钮

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_129

实例说明

在网站的注册页面中经常看到这样一种按钮效果，某注册信息下方的“接受”按钮处于不可用状态，并且文本右侧有个倒计时的数字，当用户在指定时间过后，才可以使用该按钮进入下一页面，这样可以为用户强制预留一些时间来看注册协议。本实例模拟这个效果，在程序的许可协议界面中实现了这样一个按钮。实例运行效果如图2.85所示。

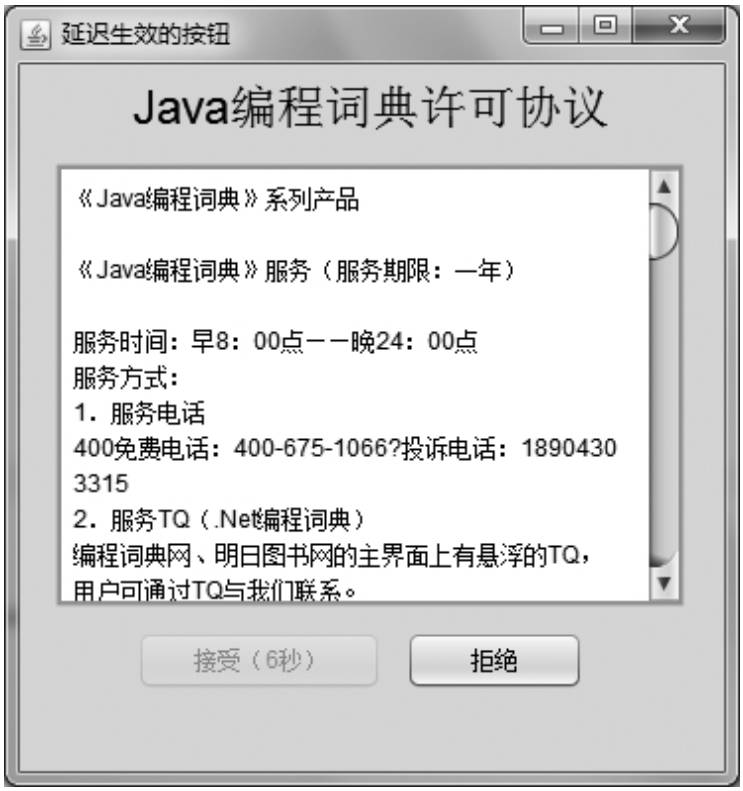


图2.85 实例运行效果

技术要点

本实例的关键技术在于 Timer 控件的应用。该控件能够在指定的时间间隔内重复执行Action。控件的ActionListener监听器将不断地捕获和处理该事件。创建一个Timer控件的构造方法的声明如下：

```
public Timer(int delay, ActionListener listener)
```

参数说明

- delay：初始延迟和动作事件间延迟的毫秒数。
- listener：初始侦听器，可以为null。

实现过程

（1）在项目中新建窗体类LazyButton。设置窗体的标题、大小和位置等属性。然后为窗体添加文本域控件并从文件中加载协议信息显示到控件中，再为窗体添加“接受”和“拒绝”两个按钮。关键代码如下：

```
    JTextArea textArea=new
JTextArea();                                    //创建文本域控件
    textArea.setLineWrap(true);
          //自动折行
    StringBuilder
sb=newStringBuilder();                                  //创建
字符串构建器
    //创建文本扫描器
    Scanner
scan=newScanner(getClass().getResourceAsStream("lzw.txt"));
    while (scan.hasNext())
{                                                  //遍历文本扫描器
      String string= (String)
  scan.nextLine();                                  //逐行获取数据
      sb.append(string+
  "\n");                                            //把所有行数
  据添加到字符串构建器
    }
    textArea.setText(sb.toString());
              //释放字符串构建器中的字符串到文本域
    textArea.setSelectionStart(0);
              //在滚动面板中把文本域滚至首行
    textArea.setSelectionEnd(0);
    scrollPane.setViewportView(textArea);
    //创建标签控件
    JLabel lblJava = new JLabel("Java编程词典许可协议");
```

```
lblJava.setFont(newFont("SansSerif",Font.PLAIN,24));
            //指定标签字体
lblJava.setHorizontalAlignment(SwingConstants.CENTER);
            //标签文本居中
lblJava.setBounds(18, 6, 318, 32);
contentPane.add(lblJava);
//创建接受按钮
button = new JButton("接受（10秒）");
button.setEnabled(false);
      //取消按钮的可用状态
button.setBounds(59, 286, 124, 30);
contentPane.add(button);
//创建拒绝按钮
JButton button_1 = new JButton("拒绝");
button_1.setBounds(195, 286, 90, 30);
contentPane.add(button_1);
```

（2）编写窗体打开事件的处理方法，在该方法中创建Timer控件，并在其事件监听器中实现按钮文本中的倒计时，在10秒之后使按钮处于激活状态。关键代码如下：

```
protected void do_this_windowOpened(WindowEvent e)
{                   //窗体打开事件处理方法
  timer=newTimer(1000,newActionListener()
 {                        //创建Timer对象并实现事件处理监听器
    int tNum=10;                                          //定义倒计时描述
    @Override
    public void actionPerformed(ActionEvent e) {
```

```
            button.setText("接受（"+ --tNum+
        "秒）");              //更新按钮的计时文本
            if (tNum<=0) {                          //计时结束后，激
        活按钮可用状态并停止Timer控件
                button.setEnabled(true);
                timer.stop();
            }
        }
        });
        timer.start();
            //启动Timer控件
    }
```

举一反三

根据本实例，读者可以实现以下功能。

增加复选框，当复选框勾选的时候“接受”按钮可用。

实现延迟生效的“确定”按钮。

实例130 动态加载表格数据

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\02\Ex02_130

实例说明

表格是一种显示复合数据的控件。它可以容纳大量的数据，但是如果将大量数据一次性添加到表格中，数据读取与显示到表格的动作都会消耗大量CPU时间，导致程序界面的假死现象。如果某程序的数据库被设计用于保存大量数据，又需要把这些数据显示在窗体界面中，

则可以通过Timer控件把所有数据逐渐导入表格控件并动态加载到界面中，这样就不会影响UI线程。实例运行效果如图2.86所示。

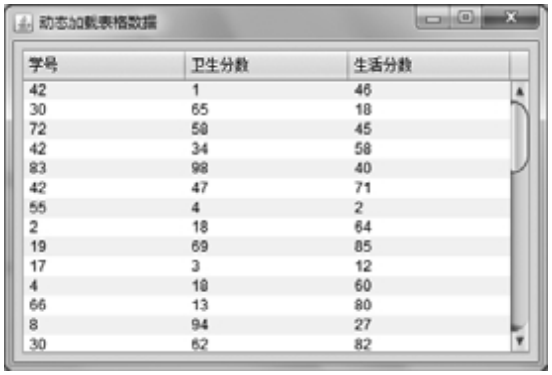


图2.86 正在不断加载数据的表格

技术要点

本实例的关键技术请参见实例129。

实现过程

（1）在项目中新建窗体类ExampleFrame。设置窗体的标题、大小和位置等属性，然后为窗体添加滚动面板和表格控件，并设置表格控件的数据模型来指定表格的列名。关键代码如下：

```
    JScrollPane scrollPane=new
JScrollPane();                                  //创建滚动面板
    contentPane.add(scrollPane, BorderLayout.CENTER);
    table=new
JTable();                                                  //创建
表格控件
    model=new DefaultTableModel(new Object[][] {},new
String[]{ "学号", "卫生分数", "生活分数"
});                                        //创建默认的表格数据模型
    table.setModel(model);
        //设置表格数据模型
```

```
scrollPane.setViewportView(table);
            //把表格添加到滚动面板视图
```

（2）编写窗体打开事件的处理方法，在该方法中创建Timer对象，使程序以每0.5秒的间隔为表格数据模型添加一行数据，其中数据是随机生成的。关键代码如下：

```
protected void do_this_windowOpened(WindowEvent e) {
  //创建Timer控件
  Timer timer = new Timer(500, new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
      Random random=new
  Random();                          //创建随机数对象
      Integer[] values = new Integer[]
        //创建整数数组作为表格行数据
        { random.nextInt(100),
    random.nextInt(100),random.nextInt(100) };
      model.addRow(values);
    //为表格数据模型添加一行数据
    }
  });
  timer.start();
      //启动Timer控件
}
```

举一反三

根据本实例，读者可以实现以下功能。

使用Timer为表格添加数据。

控制加载表格数据的时间。

第3章 Commons组件应用

Commons Lang 组件

Commons IO 组件

Commons BeanUtils 组件

其他Commons组件

3.1 Commons Lang组件

菜单是程序开发中经常使用的界面元素，合理利用菜单不但可以使用户非常方便地使用程序的功能，而且还能提高工作效率。下面通过几个应用实例，介绍菜单设计的方法和技术。

实例131 添加数组元素

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\03\Ex03_131

实例说明

Java语言中的数组并不好用：在创建时需要指定数组的长度，并且一旦创建完成则长度不能再发生变化。为了弥补这个不足，Java SE API 中提供了ArrayList 类。对于数组的超级粉丝，推荐使用Commons Lang 组件。其中的ArrayUtils 类对数组操作进行了增强，实现了向数组中增加元素的方法。本实例将演示如何使用这些方法，实例运行效果如图3.1所示。

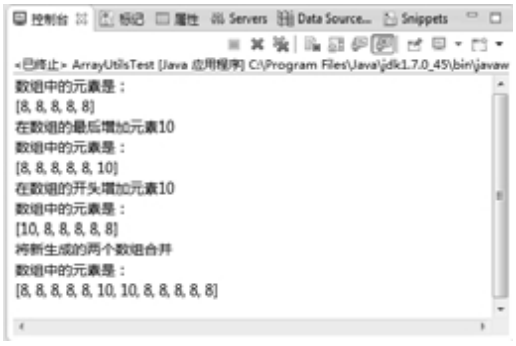


图3.1 实例运行效果

技术要点

ArrayUtils类提供了对基本类型（如int）、包装类型（如Integer）和其他引用类型数组的支持。该类尝试优雅地处理null值。如果数组为null，并不会抛出异常；如果数组中某个元素为null，才会抛出异常。ArrayUtils类增加数组元素的方法如表3.1所示。

表3.1 ArrayUtils类增加数组元素的方法

方 法 名	作 用
add(int[] array, int element)	复制给定的数组 array，并将 element 元素增加到新数组末尾
add(int[] array, int index, int element)	将 element 元素插入到数组 array 的 index 位置
addAll(int[] array1, int[] array2)	生成一个新的数组，该数组按顺序包含 array1 和 array2 的元素

实现过程

编写ArrayUtilsTest类，在该类的main()方法中，实现了向数组的不同位置增加元素和合并数组的操作。代码如下：

```
public class ArrayUtilsTest {
  public static void main(String[] args) {
    int[] array0=new
int[5];                                    //创建长度为5的
int类型数组
    Arrays.fill(array0,
8);                                        //将数组中的元素
全部初始化为8
    System.out.println("数组中的元素是：");
    System.out.println(Arrays.toString(array0));
                //输出数组中的全部元素
    System.out.println("在数组的最后增加元素10");
    int[] array1=ArrayUtils.add(array0,
10);                              //在数组的最后增加元素10
    System.out.println("数组中的元素是：");
```

```
        System.out.println(Arrays.toString(array1));
                    //输出新数组中的全部元素
        System.out.println("在数组的开头增加元素10");
        int[]
array2=ArrayUtils.add(array0,0,10);
//在数组的开头增加元素10
        System.out.println("数组中的元素是：");
        System.out.println(Arrays.toString(array2));
                        //输出新数组中的全部元素
        System.out.println("将新生成的两个数组合并");
        int[] array3=ArrayUtils.addAll(array1,
array2);                                    //合并新生成的两个数组
        System.out.println("数组中的元素是：");
        System.out.println(Arrays.toString(array3));
                        //输出新数组中的全部元素
    }
}
```

举一反三

根据本实例，读者可以开发以下程序。

实现数组元素的修改。

数组元素的删除。

实例132 删除数组元素

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\03\Ex03_132

实例说明

Java语言中的数组并不好用：在创建时需要指定数组的长度，并且一旦创建完成则长度不能再发生变化。为了弥补这个不足，Java SE API 中提供了ArrayList 类。对于数组的超级粉丝，推荐使用Commons Lang 组件。其中的ArrayUtils 类对数组操作进行了增强，实现了向数组中删除元素的方法。本实例将演示如何使用这些方法，实例运行效果如图3.2所示。

图3.2 实例运行效果

技术要点

ArrayUtils 类提供了对基本类型（如 int）、包装类型（如Integer）和其他引用类型数组的支持。该类尝试优雅地处理null值。如果数组为null，并不会抛出异常；如果一个数组中某个元素为null，才会抛出异常。ArrayUtils类删除数组元素的方法如表3.2所示。

表3.2 ArrayUtils类删除数组元素的方法

方 法 名	作 用
remove(int[] array, int index)	删除数组 array 中索引为 index 的元素
removeElement(int[] array, int element)	删除数组 array 中的第一个 element 元素，如果没有则不发生变化

实现过程

编写ArrayUtilsTest类，在该类的main()方法中演示了如何删除数组中的元素。代码如下：

```
public class ArrayUtilsTest {
  public static void main(String[] args) {
```

```
        int[] array0= { 1,2,3,2,1
};                                          //在定义数组时实
现初始化
        System.out.println("数组中的元素是：");
        System.out.println(Arrays.toString(array0));
                        //输出数组中的全部元素
        System.out.println("删除最后一个元素");
        int[] array1=ArrayUtils.remove(array0,
4);                                  //删除索引为4的元素
        System.out.println("数组中的元素是：");
        System.out.println(Arrays.toString(array1));
                        //输出新数组中的全部元素
        System.out.println("删除元素2");
        int[] array2=ArrayUtils.removeElement(array0,
2);                                  //删除元素2
        System.out.println("数组中的元素是：");
        System.out.println(Arrays.toString(array2));
                        //输出新数组中的全部元素
    }
}
```

举一反三

根据本实例，读者可以实现以下功能。

数组元素的复制。

查询数组元素。

实例133 生成随机字符串

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\03\Ex03_133

实例说明

在用户注册和登录等操作时，为了防止用户使用软件进行恶意操作（如批量注册用户），通常会要求用户输出一些随机的字符串。Commons Lang 组件的RandomStringUtils 类提供了生成随机字符串的方法。本实例将演示如何使用这些方法，实例运行效果如图3.3所示。

图3.3 实例运行效果

技术要点

RandomStringUtils类提供了多种生成随机字符串的方法。本实例使用的方法如表3.3所示。

表3.3 RandomStringUtils类生成随机字符串的方法

方 法 名	作 用
randomAlphabetic(int count)	生成一个长度为 count 的字符串，字符取自全部大小写字母
randomAlphanumeric(int count)	生成一个长度为 count 的字符串，字符取自全部大小写字母和数字
randomAscii(int count)	生成一个长度为 count 的字符串，字符的 ASCII 编码在 32~126 之间
randomNumeric(int count)	生成一个长度为 count 的字符串，字符取自数字

实现过程

编写RandomStringUtilsTest类，在该类的main()方法中输出了4类随机字符串。代码如下：

```
public class RandomStringUtilsTest {
  public static void main(String[] args) {
```

```
        System.out.println("生成长度为5的由字母组成的字符串");
        String randomString=RandomStringUtils.randomAlphabetic(5);
                                //获得随机字符串
        System.out.println(randomString);
        System.out.println("生成长度为5的由字母和数字组成的字符串");
        randomString=RandomStringUtils.randomAlphanumeric(5);
                                //获得随机字符串
        System.out.println(randomString);
        System.out.println("生成长度为5的由ASCII编码在32~126间字符组成的字符串");
        randomString=RandomStringUtils.randomAscii(5);
                                //获得随机字符串
        System.out.println(randomString);
        System.out.println("生成长度为5的由数字组成的字符串");
        randomString=RandomStringUtils.randomNumeric(5);
                                //获得随机字符串
        System.out.println(randomString);
    }
}
```

举一反三

根据本实例，读者可以实现以下功能。

生成字符取自全部大小写字母的字符串。

生成字符ASCII编码在32-126之间的字符串。

实例134 实现序列化与反序列化

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\03\Ex03_134

实例说明

当需要保存对象的状态时，可以考虑使用序列化。Java SE API中提供了对序列化的支持，但是使用起来十分不方便。Commons Lang组件的SerializationUtils 类提供了简化的序列化与反序列化的方法。本实例将演示如何使用这些方法，实例运行效果如图3.4所示。

图3.4 实例运行效果

技术要点

SerializationUtils类提供了简单的序列化与反序列化的方法。本实例使用的方法如表3.4所示。

表3.4 SerializationUtils类的常用方法

方 法 名	作 用
clone(Serializable object)	使用序列化深度克隆对象 object
deserialize(byte[] objectData)	将 byte 数组 objectData 反序列化成一个对象
deserialize(InputStream inputStream)	从 inputStream 中反序列化一个对象
serialize(Serializable obj)	将 obj 对象序列化成一个 byte 数组
serialize(Serializable obj, OutputStream outputStream)	将 obj 对象序列化成 outputStream

实现过程

（1）编写Student类，在该类中定义了两个域：id表示学生序号，name表示学生姓名。为这两个域提供了get和set方法，并且重写了toString()方法。代码如下：

```
  public class Student implements Serializable {
    private static final long serialVersionUID
=-8396517822004869094L;
    private int
id;                                          //表示学生
的序号
    privateString
name;                                        //表示学生的
姓名
    public int getId()
{                                            //获得学生的序号
      return id;
    }
    publicvoid setId(int id)
{                                            //设置学生的序号
      this.id = id;
    }
    publicString getName()
{                                            //获得学生的姓名
      return name;
    }
    publicvoid setName(String name)
{                                    //设置学生的姓名
      this.name = name;
```

```
  }
  @Override
  public String toString() {
    return "学生id：" + id + "，学生姓名：" + name;
  }
}
```

（2）编写SerializationUtilsTest类，在该类的main()方法中，首先创建了student对象，然后将其序列化，再反序列化，然后输出该对象。代码如下：

```
public class SerializationUtilsTest {
  public static void main(String[] args) {
    Student student=new
Student();                                    //创建student对象
    student.setId(10);
      //初始化id属性
    student.setName("明日科
技");                                          //初始化name属性
    System.out.println("将student对象序列化成byte数组");
    byte[]
studentByte=SerializationUtils.serialize(student);
      //将对象转换成byte数组
    System.out.println("输出序列化数组：");
    System.out.println(Arrays.toString(studentByte));
              //输出byte数组
    System.out.println("将student对象序列化到本地文件");
    FileOutputStreamout=null;
    //创建文件输出流对象
```

```
    try {
      out=newFileOutputStream(newFile("d:\\student.txt").
  getAbsoluteFile());
    } catch (FileNotFoundException e) {
      e.printStackTrace();
    }
    SerializationUtils.serialize(student,out);
              //将对象写入到student.txt文件
    System.out.println("文件生成成功！");
    System.out.println("从本地文件反序列化student对象");
    FileInputStream
in=null;                                    //创建文件输入
流对象
    try {
      in = new FileInputStream(new
  File("d:\\student.txt").getAbsoluteFile());
    } catch (FileNotFoundException e) {
      e.printStackTrace();
    }
    Student newStudent=
(Student)SerializationUtils.deserialize(in);      //读入
对象
    System.out.println("查看student对象的属性");
    System.out.println(newStudent);
  }
}
```

举一反三

根据本实例，读者可以开发以下程序。

将数组反序列化一个对象。

将obj对象序列化成一个byte数组。

实例135 整数取值范围判断

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\03\Ex03_135

实例说明

很多数学问题都涉及数值的取值范围，如求定积分等。如果使用普通方法要判断两次，即数的上限和下限。Commons Lang 组件的 IntRange 类提供了更加合理的方法。本实例将演示如何使用这些方法，实例运行效果如图3.5所示。

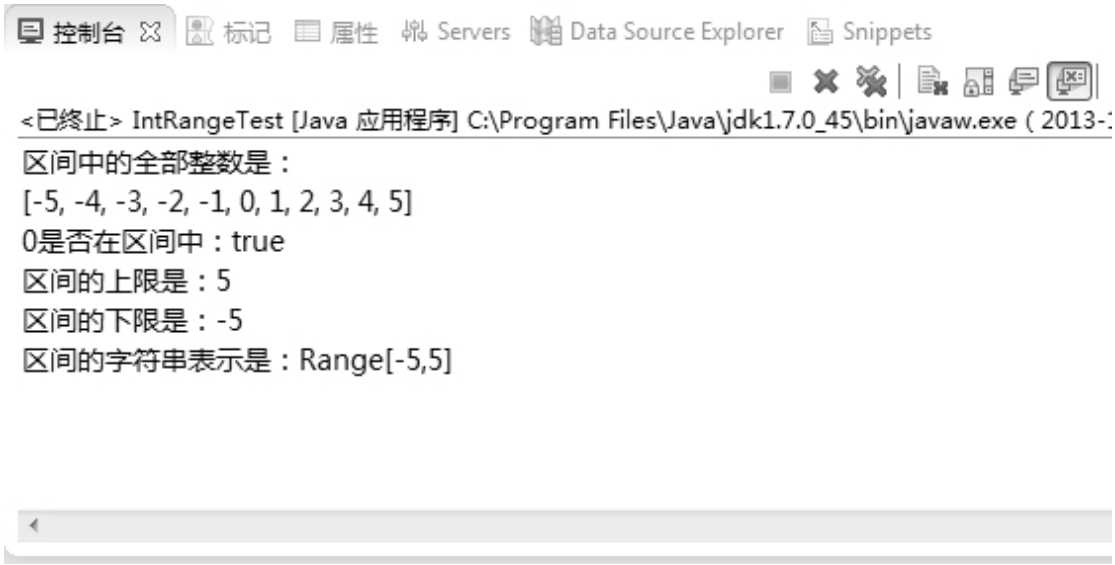


图3.5 实例运行效果

技术要点

IntRange表示一个包含端点的整数区间。IntRange类的常用方法如表3.5所示。

表3.5 IntRange类的常用方法

方 法 名	作 用
IntRange(int number1, int number2)	创建一个整数区间，上限是 number2，下限是 number1
containsInteger(int value)	判断整数区间是否包含 value
getMaximumInteger()	获得区间中的最大整数
getMinimumInteger()	获得区间中的最小整数
toArray()	获得表示区间所有整数的一个数组
toString()	以字符串形式表示一个区间

实现过程

编写IntRangeTest类，在该类的main()方法中输出了区间的上下限等信息。代码如下：

```
public class IntRangeTest {
  public static void main(String[] args) {
    IntRange range=new IntRange(-5,
5);                                    //创建一个-5~5的区间
    System.out.println("区间中的全部整数是：");
    System.out.println(Arrays.toString(range.toArray()));
                    //输出区间中的全部整数
    System.out.print("0是否在区间中：");
    System.out.println(range.containsInteger(0));
                  //判断0是否在区间中
    System.out.print("区间的上限是：");
    System.out.println(range.getMaximumInteger());
                  //输出区间的上限
    System.out.print("区间的下限是：");
    System.out.println(range.getMinimumInteger());
                  //输出区间的下限
    System.out.print("区间的字符串表示是：");
```

```
        System.out.println(range.toString());
                        //输出区间的数学表示形式
    }
}
```

举一反三

根据本实例，读者可以实现以下功能。

获取区间中最大的整数。

随机生成数字，判断是否在此区间中。

3.2 Commons IO组件

实例136 简化文件（夹）删除

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\03\Ex03_136

实例说明

Java中的文件和文件夹统一使用File类管理，该类提供了delete()方法用来删除文件和空文件夹。Commons IO 组件的FileDeleteStrategy 类定义了删除非空文件夹的方法。本实例将演示如何使用这些方法，实例运行效果如图3.6所示。

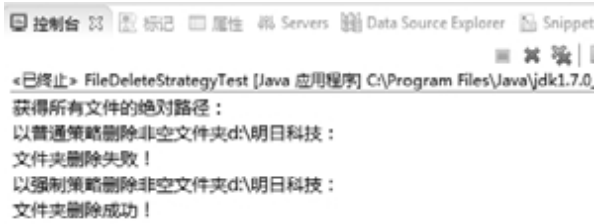


图3.6 实例运行效果

技术要点

FileDeleteStrategy类定义了删除文件（夹）的常用方法。本实例使用的方法如表3.6所示。

表3.6 FileDeleteStrategy类的常用方法

方 法 名	作 用
delete(File fileToDelete)	删除文件（夹）fileToDelete，会抛出 IOException
deleteQuietly(File fileToDelete)	删除文件（夹）fileToDelete，如果出现异常则返回 false，否则返回 true
toString()	以字符串描述当前删除策略

实现过程

编写FileDeleteStrategyTest类，在该类的main()方法中演示了删除文件（夹）的常用方式。代码如下：

```
public class FileDeleteStrategyTest {
  public static void main(String[] args) {
    File rootFile=newFile("d:\\明日科技\\推荐图书");                //创建要删除的文件夹对象
    System.out.println("获得所有文件的绝对路径：");
    File[] list = rootFile.listFiles();
    for (File file : list) {
      System.out.println(file.getAbsolutePath());
            //输出文件夹中的所有文件的绝对路径
    }
    FileDeleteStrategy strategy=FileDeleteStrategy.NORMAL;          //使用普通删除策略
    System.out.println("以普通策略删除非空文件夹d:\\明日科技：");
    try {
      strategy.delete(new File("d:\\明日科技"));
      System.out.println("文件夹删除成功！");                    //如果删除成功则提示删除成功
    } catch (IOException e) {
      System.out.println("文件夹删除失败！");                    //如果删除失败则提示删除失败
    }
```

```
        strategy=FileDeleteStrategy.FORCE;
            //使用强制删除策略
        System.out.println("以强制策略删除非空文件夹d:\\明日
    科技：");
        try {
          strategy.delete(new File("d:\\明日科技"));
          System.out.println("文件夹删除成
      功！");                    //如果删除成功则提示删除成功
        } catch (IOException e) {
          System.out.println("文件夹删除失
      败！");                    //如果删除失败则提示删除失败
        }
      }
    }
```

举一反三

根据本实例，读者可以实现以下功能。

删除非空文件。

使用FileDeleteStrategy类删除非空文件夹。

实例137 简化文件（夹）复制

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\03\Ex03_137

实例说明

Java中的文件和文件夹统一使用File类管理，该类并没有提供文件（夹）复制的相关方法。Commons IO 组件的FileUtils 类定义了复

制文件（夹）的方法。本实例将演示如何使用这些方法，实例运行效果如图3.7所示。

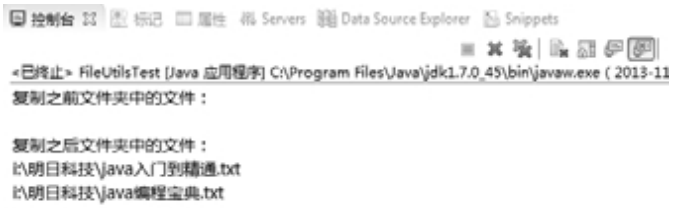


图3.7 实例运行效果

技术要点

FileUtils类定义了复制文件（夹）的常用方法。本实例使用的方法如表3.7所示。

表3.7 FileUtils类的常用方法

方 法 名	作 用
copyDirectory(File srcDir, File destDir)	将 srcDir 中的文件（夹）复制到 destDir 中
copyDirectory(File srcDir, File destDir, FileFilter filter)	作用同上，但可以选择复制的类型，如只复制文件夹
copyDirectoryToDirectory(File srcDir, File destDir)	将 srcDir 文件夹复制到 destDir 中
copyFile(File srcFile, File destFile)	将 srcFile 文件复制到 destFile 文件
copyFileToDirectory(File srcFile, File destDir)	将 srcFile 文件复制到 destDir 文件夹

实现过程

编写FileUtilsTest类，在该类的main()方法中演示了复制文件（夹）的常用方式。代码如下：

```
public class FileUtilsTest {
  public static void main(String[] args) throws
IOException {
    File srcDir = new File("D:\\明日科技");
    File destDir = new File("E:\\明日科技");
    List<String> list = new ArrayList<String>();
    System.out.println("复制之前文件夹中的文件：");
    getFilePath(list, destDir);
    for (String string : list) {
```

```
            System.out.println(string);
                    //输出复制前文件夹中的所有文件
        }
        System.out.println();
        System.out.println("复制之后文件夹中的文件：");
        FileUtils.copyDirectory(srcDir, destDir);
        getFilePath(list, destDir);
        for (String string : list) {
            System.out.println(string);
                    //输出复制后文件夹中的所有文件
        }
    }
    //获得rootFile文件夹中所有文件的绝对路径并将其保存在
list中
    private static List<String> getFilePath(List<String>
list, File rootFile) {
        File[] files = rootFile.listFiles();
        for (File file : files) {
            if (file.isDirectory()) {
                getFilePath(list, file);
            } else {
                list.add(file.getAbsolutePath().replace("\\",
        File.separator));
            }
        }
        return list;
    }
```

}

举一反三

根据本实例，读者可以实现以下功能。

不同文件夹中复制同一文件。

复制文件到指定文件夹。

实例138 简化文件（夹）排序

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\03\Ex03_138

实例说明

文件的排序通常涉及以下几个属性的比较：名称、大小、项目类型和修改时间。Commons IO组件的comparator包对每个属性定义了一个类，用来简化文件的排序。本实例将演示如何使用SizeFileComparator类对文件按大小排序，实例运行效果如图3.8所示。

图3.8 实例运行效果

技术要点

SizeFileComparator类定义了一些字段代表不同的排序方式。详细说明如表3.8所示。

表3.8 SizeFileComparator类的常用字段

字 段 名	作 用
SIZE_COMPARATOR	使文件按从小到大排序，文件夹的大小为 0
SIZE_REVERSE	使文件按从大到小排序，文件夹的大小为 0
SIZE_SUMDIR_COMPARATOR	使文件按从小到大排序，文件夹的大小为其中文件的大小总和
SIZE_SUMDIR_REVERSE	使文件按从大到小排序，文件夹的大小为其中文件的大小总和

实现过程

编写SizeFileComparatorTest类，在该类的main()方法中演示了排序文件（夹）的常用方式。代码如下：

```
public class SizeFileComparatorTest {
  @SuppressWarnings("unchecked")
  public static void main(String[] args) throws
IOException {
    File rootFile=newFile("D:\\明日科
技");                              //创建一个文件夹对象
    File[] files=
rootFile.listFiles();
//获得该文件夹中的所有文件（夹）
    System.out.println("文件（夹）的原始排序：");
    for (File file : files) {
      System.out.print(file.getName()+
  "\t");                           //输出文件夹中文件
   （夹）的名称
    }
    System.out.println();
    Arrays.sort(files,SizeFileComparator.SIZE_COMPARATOR)
;              //对files数组进行排序
    System.out.println("文件（夹）的SIZE_COMPARATOR排
序：");
```

```
    for (File file : files) {
      System.out.print(file.getName()+
  "\t");                              //输出文件夹中文件
  （夹）的名称
    }
    System.out.println();
    Arrays.sort(files,SizeFileComparator.SIZE_REVERSE);
                  //对files数组进行排序
    System.out.println("文件（夹）的SIZE_REVERSE排序：");
    for (File file : files) {
      System.out.print(file.getName()+
  "\t");                              //输出文件夹中文件
  （夹）的名称
    }
    System.out.println();
    Arrays.sort(files,
SizeFileComparator.SIZE_SUMDIR_COMPARATOR);
    System.out.println("文件（夹）的
SIZE_SUMDIR_COMPARATOR排序：");
    for (File file : files) {
      System.out.print(file.getName()+
  "\t");                              //输出文件夹中文件
  （夹）的名称
    }
    System.out.println();
    Arrays.sort(files,
SizeFileComparator.SIZE_SUMDIR_REVERSE);
```

```
        System.out.println("文件（夹）的SIZE_SUMDIR_REVERSE排
    序：");
        for (File file : files) {
          System.out.print(file.getName()+
      "\t");                              //输出文件夹中文件
       （夹）的名称
        }
      }
    }
```

举一反三

根据本实例，读者可以实现以下功能。

使文件按照从大到小排序。

使文件按照从小到大排序。

实例139 简化文件（夹）过滤

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\03\Ex03_139

实例说明

假设文件夹中有大量不同的文件，而读者又仅对某种类型的文件感兴趣，就可以使用文件的过滤功能。Commons IO 组件的filefilter包提供了大量与过滤相关的实现类。本实例将演示如何使用SizeFileFilter类获得大小超过1MB的文件，实例运行效果如图3.9所示。

图3.9 实例运行效果

技术要点

SizeFileFilter 类能根据指定的文件大小实现过滤功能，可以过滤掉小于指定数值的文件或者不小于指定数值的文件。

SizeFileFilter类的构造方法如表3.9所示。

表3.9 SizeFileFilter类的构造方法

构造方法名	作　用
SizeFileFilter(long size)	创建一个能过滤掉大小不小于 size 的文件过滤器
SizeFileFilter(long size, boolean acceptLarger)	根据 acceptLarger 值来确定要过滤的文件大小，如果 acceptLarger 为 true，则过滤掉小于 size 的文件；如果 acceptLarger 为 false，则过滤掉不小于 size 的文件

实现过程

编写SizeFileFilterTest类，在该类的main()方法中过滤掉文件夹中文件大小小于1MB的文件。代码如下：

```
public class SizeFileFilterTest {
  public static void main(String[] args) {
    Filedir=newFile("d:\\明日科
技");                                    //创建一个文件夹对象
    System.out.println("过滤前文件夹中的文件：");
```

```
    File[]
files=dir.listFiles();                                          /
/获得该文件夹中所有的文件和子文件夹
    for (File file : files)
{                                                    //输出文件夹
中文件的名字和大小
      System.out.println(file.getName()+ "的大小是："+
  file.length());
    }
    System.out.println("过滤后文件夹中的文件：");
    String[] fileNames=dir.list(newSizeFileFilter(1024 *
1024));                  //过滤掉小于1MB的文件
    for (int i = 0; i < fileNames.length; i++) {
      System.out.println(fileNames[i]);
    }
  }
}
```

举一反三

根据本实例，读者可以实现以下功能。

利用其他属性对文件过滤。

创建一个文件过滤器。

实例140 简化文件的读写操作

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\03\Ex03_140

实例说明

Java IO操作中会出现大量的IOException，需要对其捕获或抛出。Commons IO组件的IOUtils对此进行了封装，并提供了大量简化IO 操作的方法，本实例简单介绍与文件读写相关的方法。实例运行效果如图3.10所示。

图3.10 实例运行效果

技术要点

IOUtils 类为 input/output 操作提供静态工具方法。该类中与读流有关的方法都已经被缓冲了，所以不需要使用BufferedRead。缓冲的大小是4KB，经测试这是效率最高的。该类的方法并不是及时关闭流，这意味着需要手动关闭。IOUtils类的常用方法如表3.10所示。

表3.10 IOUtils类的常用方法

方 法 名	作 用
closeQuietly(InputStream input)	无条件关闭 InputStream
closeQuietly(OutputStream output)	无条件关闭 OutputStream
closeQuietly(Reader input)	无条件关闭 Reader
closeQuietly(Writer output)	无条件关闭 Writer
readLines(InputStream input)	将 InputStream 的内容转换成一个字符串列表
readLines(Reader input)	将 Reader 的内容转换成一个字符串列表
write(String data, OutputStream output)	将字符串写入 OutputStream 中
write(String data, Writer output)	将字符串写入 Writer 中

实现过程

编写IOUtilsTest类，在该类的main()方法中先向文件中写入5个字符串，然后输出。代码如下：

```
public class IOUtilsTest {
  @SuppressWarnings("unchecked")
  public static void main(String[] args) {
```

```
  FileOutputStream out = null;
  FileInputStream in = null;
  try {
    out=new FileOutputStream("d:\\明日科
技.txt");                          //创建文件输出流对象
    in=new FileInputStream("d:\\明日科
技.txt");                           //创建文件输入流对象
    System.out.println("向文件中写入5个随机字符串");
    for (int i=0; i<5; i++)
{                                  //向文件中写入5个
随机字符串
      IOUtils.write(RandomStringUtils.randomAlphanumeri
  c(5)+ "\n",out);
    }
    System.out.println("输出文件中的随机字符串");
    List<String> list=
IOUtils.readLines(in);                                //
从文件中读取字符串
    for (String string : list) {
      System.out.println(string);
    }
  } catch (FileNotFoundException e) {
    e.printStackTrace();
  } catch (IOException e) {
    e.printStackTrace();
  } finally {
```

```
            IOUtils.closeQuietly(out);
                        //释放资源
            IOUtils.closeQuietly(in);
                        //释放资源
        }
    }
}
```

举一反三

根据本实例，读者可以实现以下功能。

利用IOUtils类简化操作。

将InputStream转换成字节数组。

3.3 Commons BeanUtils组件

实例141 设置JavaBean简单属性

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\03\Ex03_141

实例说明

为了实现面向对象的封装特性，Java程序员经常使用JavaBean。即将类的所有域设置成私有的，然后对每个域提供 get 和 set 方法来获得和修改域的值。本实例演示在不能事先获得JavaBean的对象和要获得（修改）的域时，如何用BeanUtils组件实现动态获得和修改JavaBean属性的功能。实例运行效果如图3.11所示。

图3.11 实例运行效果

技术要点

JavaBean 的属性类型可以分成 3 类，即标准 JavaBean 规范支持的 Simple、Indexed 和BeanUtils包支持的Mapped，这些类型的概要说明如下。

□ Simple：用来存储单一值，如Java基本类型int、引用类型String 等。

□ Indexed：用来存储一组相同类型的数据，使用从 0 开始的整数索引，如 Java 数组、列表。

□ Mapped：用来存储一组键值对，利用String 类型的键可以获得相应的值，如Java 映射。

本实例使用PropertyUtils类来完成获得和修改JavaBean属性的功能，主要应用的方法如表3.11所示。

表3.11 PropertyUtils类的常用方法

方 法 名	作 用
getSimpleProperty(Object bean, String name)	获得 bean 对象 name 属性的值
setSimpleProperty(Object bean, String name, Object value)	修改 bean 对象 name 属性的值为 value
getIndexedProperty(Object bean, String name, int index)	获得 bean 对象 name 属性的第 index 个值
setIndexedProperty(Object bean, String name, int index, Object value)	修改 bean 对象 name 属性的第 index 个值为 value
getMappedProperty(Object bean, String name, String key)	获得 bean 对象 name 属性的 key 键对应的值
setMappedProperty(Object bean, String name, String key, Object value)	修改 bean 对象 name 属性的 key 键对应的值为 value

实现过程

（1）编写Employee类，该类定义了3个域：name表示员工姓名，phoneNumber表示员工手机号码，address表示员工的地址，并且提供了相应的get和set方法。代码如下：

```
public class Employee {
  privateString
name;                                                    //
表示员工的姓名
  privateString[]phoneNumber=newString[10];
                    //表示员工的手机号码
  privateMap<String,String> address=new
HashMap<String,String>();              //表示员工的地址
  publicString getName()
{                                                        //获得员工
```

```
的姓名
        return name;
    }
    publicvoid setName(String name)
{                                          //修改员工的姓名
        this.name = name;
    }
    publicString[]getPhoneNumber()
{                                          //获得员工的手机号
码
        return phoneNumber;
    }
    publicvoid setPhoneNumber(String[]phoneNumber)
{                              //修改员工的手机号码
        this.phoneNumber = phoneNumber;
    }
    publicMap<String,String>getAddress()
{                                       //获得员工的地址
        return address;
    }
    publicvoid setAddress(Map<String,String> address)
{                              //修改员工的地址
        this.address = address;
    }
  }
```

（2）编写Test类进行测试，在main()方法中先创建了employee对象，并输出对该对象赋值前后对象的属性值。代码如下：

```
public class Test {
  public static void main(String[] args) {
    Employee
employee=newEmployee();                    //获得一个
Employee对象
    //获得Employee对象的属性值，由于事先并未对其赋值，所
以应该为空
    String name = employee.getName();
    String phoneNumber = employee.getPhoneNumber()[0];
    String address = employee.getAddress().get("home");
    //输出刚获得的属性值
    System.out.println("设置属性值之前：");
    System.out.println("name属性："+ name);
    System.out.println("phoneNumber属性的第一个
值："+phoneNumber);
    System.out.println("address属性home键所对应的值："+
address);
    try {                                      //使用
PropertyUtils类的相关方法对Employee对象的域赋值
      PropertyUtils.setSimpleProperty(employee, "name",
  "明日科技");
      PropertyUtils.setIndexedProperty(employee,
  "phoneNumber", 0, "1234567");
      PropertyUtils.setMappedProperty(employee,
  "address", "home", "中国");
      //获得Employee对象的属性值，由于刚刚对其赋值，因此
  应该不为空
```

```
            name = (String)
        PropertyUtils.getSimpleProperty(employee, "name");
            phoneNumber = (String)
        PropertyUtils.getIndexedProperty(employee,
        "phoneNumber", 0);
            address = (String)
        PropertyUtils.getMappedProperty(employee, "address",
        "home");
            //输出刚获得的属性值
            System.out.println("设置属性值之后：");
            System.out.println("name属性："+ name);
            System.out.println("phoneNumber属性的第一个值："+
        phoneNumber);
            System.out.println("address属性home键所对应的值："+
        address);
          } catch (IllegalAccessException e) {
            e.printStackTrace();
          } catch (InvocationTargetException e) {
            e.printStackTrace();
          } catch (NoSuchMethodException e) {
            e.printStackTrace();
          }
        }
    }
```

举一反三

根据本实例，读者可以实现以下功能。

实现动态获得JavaBean的功能。

实现动态修改JavaBean的功能。

实例142 设置JavaBean级联属性

这是一个可以用来提高基础性能的实例

实例位置：光盘\mingrisoft\03\Ex03_142

实例说明

如果JavaBean的一个域是引用对象，而需要获得（修改）该引用对象的域时，会调用两次get（set）方法，代码显得十分麻烦。本实例使用BeanUtils组件来简化这个操作。实例运行效果如图3.12所示。

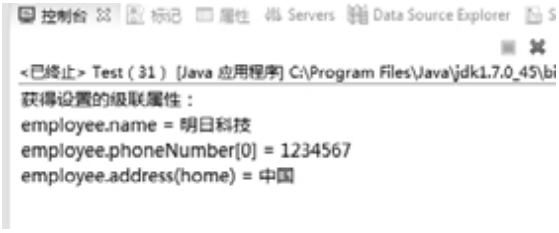


图3.12 实例运行效果

技术要点

本实例使用PropertyUtils类来完成获得和修改JavaBean属性的功能，主要应用的方法如表3.12所示。

表3.12 PropertyUtils类的常用方法

方 法 名	作 用
getNestedProperty(Object bean, String name)	获得 bean 对象的级联属性值
setNestedProperty(Object bean, String name, Object value)	修改 bean 对象的级联属性值为 value

实现过程

（1）编写Employee类，该类定义了3个域：name表示员工姓名，phoneNumber表示员工手机号码，address表示员工的地址，并且提供了相应的get和set方法。代码如下：

```
public class Employee {
```

```
    private String
name;                                                    //表示员
工的姓名
    private String[]phoneNumber=new
String[10];                                  //表示员工的手机号码
    private Map<String,String> address=new
HashMap<String,String>();              //表示员工的地址
    public String getName()
{                                                        //获得员工的姓
名
      return name;
    }
    public void setName(String name)
{                                        //修改员工的姓名
      this.name = name;
    }
    public String[]getPhoneNumber()
{                                        //获得员工的手机号码
      return phoneNumber;
    }
    public void setPhoneNumber(String[]phoneNumber)
{                          //修改员工的手机号码
      this.phoneNumber = phoneNumber;
    }
    public Map<String,String>getAddress()
{                                    //获得员工的地址
      return address;
```

```
    }
    public void setAddress(Map<String,String> address)
{                          //修改员工的地址
      this.address = address;
    }
  }
```

（2）编写 Company 类，该类定义了一个域：employee 代表公司的员工，并对该域提供了get和set方法。代码如下：

```
  public class Company {
    private Employee employee=new
Employee();                                    //实例化公司的员
工
    public Employee getEmployee()
{                                              //获得公司的员工
      return employee;
    }
    public void setEmployee(Employee employee)
{                                    //修改公司的员工
      this.employee = employee;
    }
  }
```

（3）编写Test类进行测试，在该类的main()方法中为对象的级联属性赋值并输出赋值后的结果。代码如下：

```
  public class Test {
    public static void main(String[] args) {
      Company company = new Company();
```

```
    try
{
//设置级联属性并输出
        PropertyUtils.setNestedProperty(company,
  "employee.name", "明日科技");
        PropertyUtils.setNestedProperty(company,
  "employee.phoneNumber[0]", "1234567");
        PropertyUtils.setNestedProperty(company,
  "employee.address(home)", "中国");
        System.out.println("获得设置的级联属性：");
        String name = (String)
  PropertyUtils.getNestedProperty(company,
  "employee.name");
        String phoneNumber = (String)
  PropertyUtils.getNestedProperty(company,
  "employee.phoneNumber[0]");
        String address = (String)
  PropertyUtils.getNestedProperty(company,
  "employee.address(home)");
        System.out.println("employee.name ="+ name);
        System.out.println("employee.phoneNumber[0] ="+
  phoneNumber);
        System.out.println("employee.address(home) ="+
  address);
    } catch (IllegalAccessException e) {
        e.printStackTrace();
    } catch (InvocationTargetException e) {
```

```
            e.printStackTrace();
        } catch (NoSuchMethodException e) {
            e.printStackTrace();
        }
    }
}
```

举一反三

根据本实例，读者可以实现以下功能。

获得JavaBean。

修改JavaBean。

实例143 动态生成JavaBean

这是一个可以提高基础性能的实例

实例位置：光盘\mingrisoft\03\Ex03_143

实例说明

如果希望使用JavaBean的优势又不方便创建JavaBean对象（如对象的属性值会动态发生变化），则可以使用BeanUtils相关工具类动态生成JavaBean。本实例将动态生成一个employee对象。实例运行效果如图3.13所示。

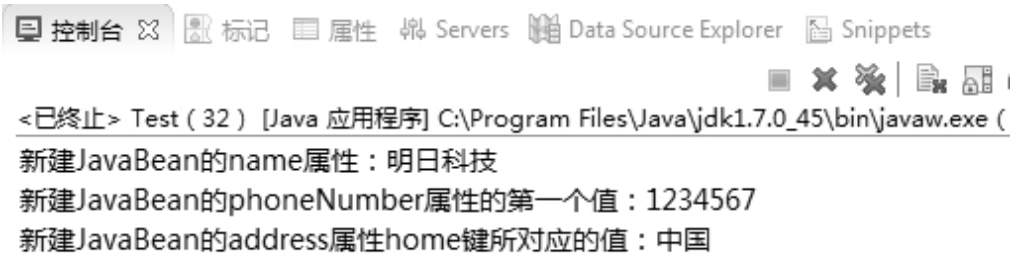


图3.13 实例运行效果

技术要点

DynaProperty 类是用于描述独立的 DynaBean 属性的。本实例主要使用了它的两个非空构造方法，其说明如下：

public DynaProperty(String name)

参数说明

name：初始化的属性的名称。

public DynaProperty(String name,Class type,Class contentType)

参数说明

- name：初始化的属性的名称。
- type：指定的属性的类型。
- contentType：Indexed 或Mapped 类型属性的元素类型。

BasicDynaClass 最小化地实现了 DynaClass 接口，它可以作为编写更加专业的 DynaClass接口实现类的基础。本实例使用了它的一个非空构造方法，其声明如下：

public BasicDynaClass(String name,Class dynaBeanClass,DynaProperty[] properties)

参数说明

- name：DynaBean 类的名字。
- dynaBeanClass：DynaBean 的实现类，null 代表实现类为BasicDynaBean。
- properties：新JavaBean 所支持的属性。

DynaBean 接口用来支持属性名字、类型和值可以动态修改的JavaBean。实现该接口的类在BeanUtils组件中可以当JavaBean使用。DynaBean接口的常用方法如表3.13所示。

表3.13 DynaBean接口的常用方法

方 法 名	作 用
get(String name)	获得名为 name 的简单属性的值
get(String name, int index)	获得名为 name、序号为 index 的索引属性的值
get(String name, String key)	获得名为 name、键为 key 的映射属性的值
set(String name, Object value)	修改名为 name 的简单属性的值为 value
set(String name, int index, Object value)	修改名为 name、序号为 index 的索引属性的值为 value
set(String name, String key, Object value)	修改名为 name、键为 key 的映射属性的值为 value

实现过程

编写Test类，在该类的main()方法中创建了一个动态Bean，对其属性赋值之后输出。代码如下：

```
public class Test {
  public static void main(String[] args) {
    DynaProperty[]properties=new
DynaProperty[3];                          //声明保存3个属性值
的数组
    //指定属性名称和类型
    properties[0] = new DynaProperty("name",
String.class);
    properties[1] = new DynaProperty("phoneNumber",
String[].class, String.class);
    properties[2] = new DynaProperty("address",
Map.class, String.class);
    BasicDynaClass dynaClass = new
BasicDynaClass("employee", null, properties);
    DynaBean employee = null;
    try{
      employee=dynaClass.newInstance();
            //获得DynaBean的实例
    } catch (IllegalAccessException e) {
```

```
            e.printStackTrace();
        } catch (InstantiationException e) {
            e.printStackTrace();
        }
        //为属性赋值
        employee.set("name", "明日科技");
        employee.set("phoneNumber",new
String[10]);                              //索引类型要先初始化
        employee.set("phoneNumber", 0, "1234567");
        employee.set("address", new HashMap<String,String>
());              //映射类型要先初始化
        employee.set("address", "home", "中国");
        String name = (String) employee.get("name");
        String phoneNumber = (String)
employee.get("phoneNumber", 0);
        String address = (String) employee.get("address",
"home");
        //输出属性值
        System.out.println("新建JavaBean的name属性："+ name);
        System.out.println("新建JavaBean的phoneNumber属性的第
一个值："+ phoneNumber);
        System.out.println("新建JavaBean的address属性home键所
对应的值："+ address);
    }
}
```

举一反三

根据本实例，读者可以实现以下功能。

动态生成JavaBean。

使用BeanUtils相关工具类动态生成JavaBean。

实例144 复制JavaBean属性

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\03\Ex03_144

实例说明

在使用Hibernate等ORM框架操作数据库时，其操作和返回的对象都是JavaBean。如果使用普通的方式复制JavaBean属性，将会调用大量的get和set方法。本实例使用BeanUtils组件来简化这个操作。实例运行效果如图3.14所示。

图3.14 实例运行效果

技术要点

BeanUtils 利用反射机制提供了操作 bean 的方法，本实例使用了其 copyProperties()方法来实现Bean属性的复制操作，该方法的声明如下：

public static void copyProperties(Object dest,Object orig)throws IllegalAccessException,InvocationTargetException

参数说明

- dest：目标对象。
- orig：源对象。

实现过程

（1）编写Employee类，该类定义了3个域：name表示员工姓名，phoneNumber表示员工手机号码，address表示员工的地址，并且提供了相应的get和set方法。代码如下：

```
  public class Employee {
    private String name;        //表示员工的姓名
    private String[] phoneNumber = new
String[10];        //表示员工的手机号码
    private Map<String, String> address = new
HashMap<String, String>();    //表示员工的地址
    public String getName() {        //获得员工的姓名
      return name;
    }
    public void setName(String name) {        //修改员工的
姓名
      this.name = name;
    }
    public String[] getPhoneNumber() {        //获得员工的
手机号码
      return phoneNumber;
    }
    public void setPhoneNumber(String[] phoneNumber)
{        //修改员工的手机号码
      this.phoneNumber = phoneNumber;
    }
    public Map<String, String> getAddress() {        //获得
员工的地址
      return address;
```

```
    }
    public void setAddress(Map<String, String> address)
{       //修改员工的地址
      this.address = address;
    }
  }
```

（2）编写Test类进行测试，在main()方法中，首先创建了两个Employee对象，然后对其中一个赋值，对另外一个复制，再输出两个的域。代码如下：

```
  public class Test {
    public static void main(String[] args) {
      Employee employee1 = new Employee();        //声明
  Employee变量
      Employee employee2 = new Employee();        //声明
  Employee变量
      try{
        //为employee1赋值
        PropertyUtils.setSimpleProperty(employee1, "name",
    "明日科技");
        PropertyUtils.setIndexedProperty(employee1,
    "phoneNumber", 0, "1234567");
        PropertyUtils.setMappedProperty(employee1,
    "address", "home", "中国");
        BeanUtils.copyProperties(employee2,
    employee1);        //将employee1复制到employee2
        //获得employee2的属性值
```

```
            String name = (String)
        PropertyUtils.getSimpleProperty(employee2, "name");
            String phoneNumber = (String)
        PropertyUtils.getIndexedProperty(employee2,
        "phoneNumber", 0);
            String address = (String)
        PropertyUtils.getMappedProperty(employee2, "address",
        "home");
            //输出employee2的属性值
            System.out.println("复制属性值之后：");
            System.out.println("name属性："+ name);
            System.out.println("phoneNumber属性的第一个值："+
        phoneNumber);
            System.out.println("address属性home键所对应的值："+
        address);
          } catch (IllegalAccessException e) {
            e.printStackTrace();
          } catch (InvocationTargetException e) {
            e.printStackTrace();
          } catch (NoSuchMethodException e) {
            e.printStackTrace();
          }
        }
    }
```

举一反三

根据本实例，读者可以实现以下功能。

使用BeanUtils组件复制JavaBean。

利用反射机制操作Bean方法。

实例145 动态排序JavaBean

这是一个可以启发思维的实例

实例位置：光盘\mingrisoft\03\Ex03_145

实例说明

Java中如果对对象排序可以考虑实现Comparable接口，但是需要排序的属性一旦指定就不能再修改。BeanUtils 组件提供了对 JavaBean 动态排序的支持，即可以在运行时指定排序的属性。实例运行效果如图3.15所示。

技术要点

BeanComparator通过指定的属性来比较两个bean。它也可以用来比较级联属性、索引属性、映射属性和组合属性等。BeanComparator默认把指定的bean属性传递给ComparableComparator。如果比较的属性值可能有空值，那么应该传递一个合适的Comparator或ComparatorChain给构造方法。

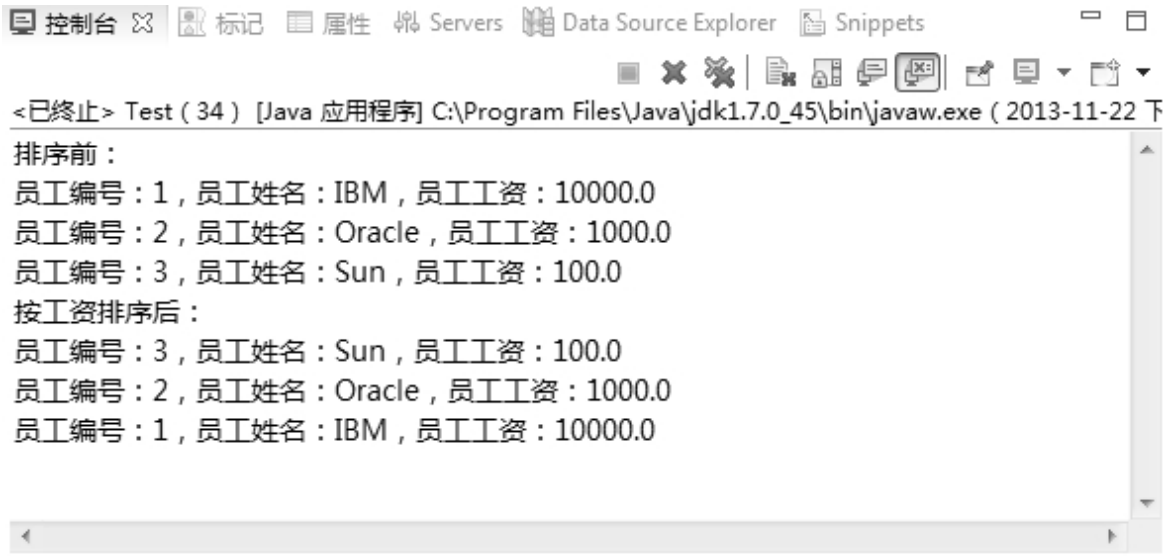


图3.15 实例运行效果

实现过程

（1）编写Employee类，该类定义了3个域：id表示员工的序号，name表示员工的姓名， salary表示员工的薪水，并且提供了相应的get和set方法。代码如下：

```
public class Employee {
  private int id;        //表示员工的序号
  private String name;        //表示员工的姓名
  private double salary;        //表示员工的薪水
  //省略get和set方法
  @Override
  public String toString() {
    return "员工编号："+ id +"，员工姓名："+ name +"，员工工资："+ salary;
  }
}
```

（2）编写Test类，在该类的main()方法中创建了3个Employee对象并进行初始化，然后使用salary域进行排序。代码如下：

```
public class Test {
  @SuppressWarnings("unchecked")
  public static void main(String[] args) {
    Employee employee1 = new Employee();        //创建employee1对象并初始化
    employee1.setId(1);
    employee1.setName("IBM");
    employee1.setSalary(10000);
    Employee employee2 = new Employee();        //创建employee2对象并初始化
```

```
    employee2.setId(2);
    employee2.setName("Oracle");
    employee2.setSalary(1000);
    Employee employee3 = new Employee();        //创建
employee3对象并初始化
    employee3.setId(3);
    employee3.setName("Sun");
    employee3.setSalary(100);
    List<Employee> list = new ArrayList<Employee>
();        //创建list对象并保存全部员工对象
    list.add(employee1);
    list.add(employee2);
    list.add(employee3);
    System.out.println("排序前：");
    for (Employee employee : list) {
      System.out.println(employee);        //输出所有对象
    }
    Collections.<Employee> sort(list, new
BeanComparator("salary"));        //进行排序
    System.out.println("按工资排序后：");
    for (Employee employee : list) {
      System.out.println(employee);        //输出所有对象
    }
  }
}
```

举一反三

根据本实例，读者可以实现以下功能。

实现动态排序。
按指定条件进行动态排序。

3.4 其他Commons组件

实例146 优雅的JDBC代码

本实例可以提高工作效率

实例位置：光盘\mingrisoft\03\Ex03_146

实例说明

在使用JDBC的过程中，SQLException几乎处处可见。这不仅增加了代码量（要处理异常），而且影响代码的阅读（逻辑混乱）。Commons DbUtils 组件提供了一些工具类来优化JDBC 代码。本实例将演示如何使用它们实现向表格中添加数据，对于删除和修改，只要换成相应的SQL语句即可。实例运行效果如图3.16所示。

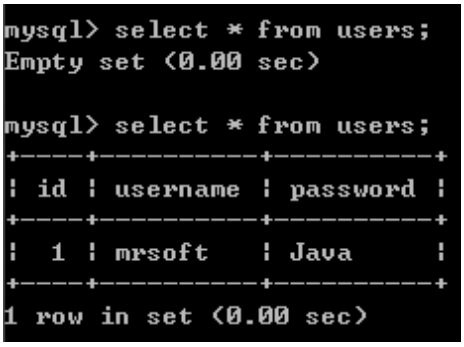


图3.16 实例运行效果

技术要点

DbUtils类是一组JDBC工具集。DbUtils类的常用方法如表3.14所示。

表3.14 DbUtils类的常用方法

方 法 名	作 用
close(Connection conn)	关闭数据库连接，不会隐藏 SQLException
close(ResultSet rs)	关闭结果集，不会隐藏 SQLException
close(Statement stmt)	关闭语句，不会隐藏 SQLException
closeQuietly(Connection conn)	关闭数据库连接，会隐藏 SQLException
closeQuietly(ResultSet rs)	关闭结果集，会隐藏 SQLException
closeQuietly(Statement stmt)	关闭语句，会隐藏 SQLException
loadDriver(String driverClassName)	加载驱动

QueryRunner类用来执行SQL语句并获得适当的返回值。本实例使用的方法如表3.15所示。

表3.15 QueryRunner类的常用方法

方 法 名	作 用
update(Connection conn, String sql)	执行没有参数的增加、修改、删除操作
update(Connection conn, String sql, Object param)	执行有一个参数的增加、修改、删除操作
update(Connection conn, String sql, Object... params)	执行有多个参数的增加、修改、删除操作
update(String sql)	已经获得连接时执行没有参数的增加、修改、删除操作
update(String sql, Object param)	已经获得连接时执行有一个参数的增加、修改、删除操作
update(String sql, Object... params)	已经获得连接时执行有多个参数的增加、修改、删除操作

实现过程

（1）创建users表，该表包括了3个字段：id用来做标识列，username用来保存用户名， password用来保存密码。代码如下：

```
create table users (
  id int auto_increment primary key,
  username varchar(20),
  password varchar(20)
)
```

（2） 编写QueryRunnerTest 类，在该类中定义了3 个方法：getConnection()方法用于获得数据库连接，operate()方法用于操作数据库，main()方法用于测试。代码如下：

```
public class QueryRunnerTest {
```

```
    //定义JDBC相关参数
    private static String URL
="jdbc:mysql://localhost:3306/db_database18";
    private static String DRIVER ="com.mysql.jdbc.Driver";
    private static String USERNAME ="mr";
    private static String PASSWORD ="mingrisoft";
    private static Connection conn;
    public static Connection getConnection() {        //用
于获得数据库连接的工具方法
      try {
        DbUtils.loadDriver(DRIVER);        //加载驱动
        conn = DriverManager.getConnection(URL, USERNAME,
    PASSWORD);        //建立连接
      } catch (SQLException e) {
        e.printStackTrace();
      }
      return conn;
    }
    public static int operate(String sql, Object... params)
{       //用于执行有参数的SQL语句
      int result = 0;
      QueryRunner runner = new QueryRunner();
      try {
        result = runner.update(getConnection(), sql,
    params);        //执行SQL语句
      } catch (SQLException e) {
        e.printStackTrace();
```

```
    } finally {
      DbUtils.closeQuietly(conn);          //关闭连接
    }
    return result;
  }
  public static void main(String[] args) {
    String sql ="insert into users(username, password)
values (?, ?)";
    Object[] params ={"mrsoft", "Java"};
    operate(sql, params);          //向数据库中插入一条数据
  }
}
```

举一反三

根据本实例，读者可以实现以下功能。

JDBC代码加载驱动、关闭连接等操作。

DbUtils类加载驱动、关闭连接等操作。

实例147 结果集与Bean列表

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\03\Ex03_147

实例说明

使用 JDBC 进行查询得到的结果是一个 ResultSet 对象，该对象使用起来非常不方便。Commons DbUtils 组件的handlers 包提供了近10 种方法对其转换。本实例演示如何将结果集转换成Bean列表。实例运行效果如图3.17所示。

图3.17 实例运行效果

技术要点

QueryRunner类用来执行SQL语句并获得适当的返回值。

实现过程

（1）创建users表，该表包括了3个字段：id用来做标识列，username用来保存用户名， password用来保存密码。代码如下：

```
create table users (
  id int auto_increment primary key,
  username varchar(20),
  password varchar(20)
)
```

（2）针对users表，编写User类。在该类中定义了3个域分别和表中的字段相对应，并且自动提供了get和set方法。代码如下：

```
public class User {
  private int id;
  private String username;
  private String password;
  //省略get和set方法
  @Override
  public String toString() {
```

```
        return "序号："+ id +"，用户名："+ username +"，密
    码："+ password;
      }
    }
```

（3）编写QueryRunnerTest类，在该类中定义了3个方法：getConnection()方法用于创建数据库连接，query()方法用于查询，main()方法用于进行测试。代码如下：

```
    public class QueryRunnerTest {
      //定义JDBC相关参数
      private static String URL
  ="jdbc:mysql://localhost:3306/db_database18";
      private static String DRIVER ="com.mysql.jdbc.Driver";
      private static String USERNAME ="mr";
      private static String PASSWORD ="mingrisoft";
      private static Connection conn;
      public static Connection getConnection() {        //用
  于获得数据库连接的工具方法
        try {
          DbUtils.loadDriver(DRIVER);        //加载驱动
          conn = DriverManager.getConnection(URL, USERNAME,
      PASSWORD); //建立连接
        } catch (SQLException e) {
          e.printStackTrace();
        }
        return conn;
      }
```

```
    public static List<User> query(String sql) {        //
用来将查询结果转换成bean列表的工具方法
      QueryRunner qr = new QueryRunner();
      List<User> list = null;
      try {
        list = qr.query(getConnection(), sql, new
    BeanListHandler<User>(User.class));
      } catch (SQLException e) {
        e.printStackTrace();
      } finally {
        DbUtils.closeQuietly(conn);        //关闭连接
      }
      return list;
    }
    public static void main(String[] args) {
      System.out.println("表users中的全部数据如下：");
      List<User> list = query("select * from
  users");        //查询users表中的全部数据
    for (User user : list) {
      System.out.println(user);
    }
  }
```

举一反三

根据本实例，读者可以实现以下功能。

将结果集转换成Bean列表。

将结果集转换成Object数组列表。

实例148 编写MD5查看器

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\03\Ex03_148

实例说明

MD5 可以为软件生成一个唯一的标识，防止软件在传播过程中遭到恶意修改。Commons Codec组件的DigestUtils类提供了对MD5、SHA等算法的支持。本实例将制作一个MD5查看器。实例运行效果如图3.18所示。

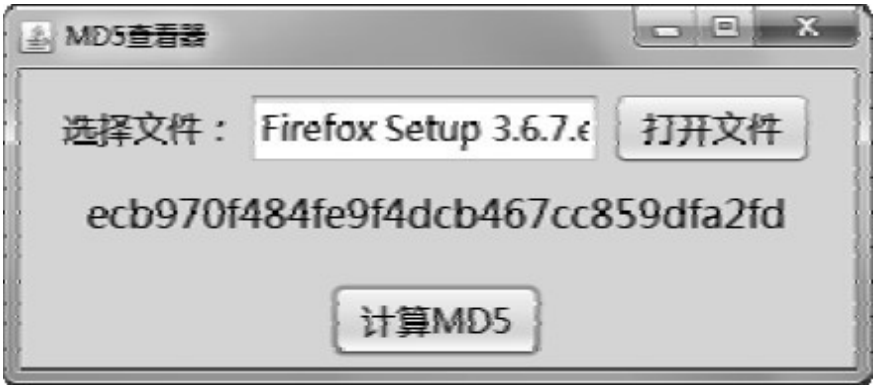


图3.18 实例运行效果

技术要点

DigestUtils类提供了很多工具方法来支持数字信息算法。本实例使用md5Hex()方法对输入流进行分析，然后计算其MD5值，该方法的声明如下：

public static String md5Hex(InputStream data)throws IOException

参数说明

data：需要进行计算的数据流。

实现过程

（1）继承JFrame编写一个窗体类，名称为MD5Viewer。

（2）设计MD5Viewer窗体类时用到的控件及说明如表3.16所示。

表3.16 窗体的控件及说明

控 件 类 型	控 件 命 名	控 件 用 途
JLabel	fileLabel	显示文本域的作用
	messageLabel	显示 MD5 计算结果
JTextField	textField	显示用户选择的文件的绝对路径
JButton	fileButton	让用户选择要计算的文件，并计算 MD5
	md5Button	显示 MD5 结果

（3）编写按钮激活事件监听器调用的do_fileButton_actionPerformed()方法，该方法是在类中自定义的，主要用途是实现选择文件并计算MD5。代码如下：

```
    protected void do_fileButton_actionPerformed(ActionEvent arg0) {
        JFileChooser fileChooser = new JFileChooser();        //创建文件选择器
        fileChooser.setFileSelectionMode(JFileChooser.FILES_ONLY);        //让文件选择器只能选择文件
        fileChooser.setMultiSelectionEnabled(false);        //不能选择多个文件
        int result = fileChooser.showOpenDialog(this);
        if (result == JFileChooser.APPROVE_OPTION) {
            File selectFile = fileChooser.getSelectedFile();        //获得用户选择的文件
            textField.setText(selectFile.getName ());        //显示选择文件的名称
            FileInputStream in = null;
            try {
```

```
            in = new FileInputStream(selectFile);          //获得
        文件输入流
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
        try {
            md5 = DigestUtils.md5Hex(in);          //计算MD5值，
        并保存在域变量md5中
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

举一反三

根据本实例，读者可以实现以下功能。

MD5计算。

SHA计算。

第4章 数据库技术

通过JDBC-ODBC桥连接数据库

JDBC技术连接数据库

数据库与数据表

数据增加、更新与删除操作

4.1 通过JDBC-ODBC桥连接数据库

实例149 通过JDBC-ODBC桥连接SQL Server 2000数据库

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\04\Ex04_149

实例说明

本实例实现的是通过JDBC-ODBC桥建立数据库连接。由于ODBC驱动程序被广泛使用，建立这种桥连接之后，使得JDBC有能力访问几乎所有类型的数据库。本实例实现配置数据库，并测试是否可通过JDBC-ODBC桥进行连接。连接成功给出图4.1所示的提示信息。

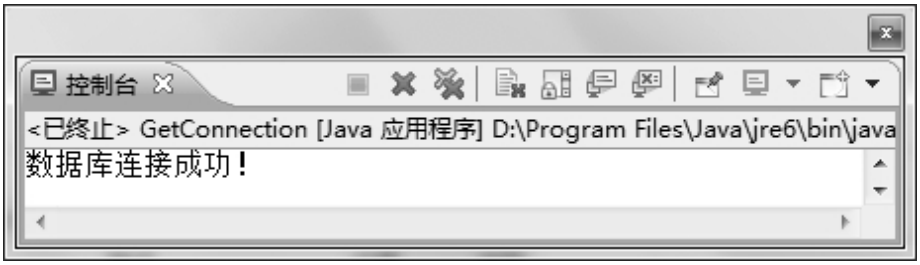


图4.1 实例运行结果

技术要点

建立数据库连接，需要指定数据库的驱动和路径。

指定驱动：

```
String driverClass="sun.jdbc.odbc.JdbcOdbcDriver";
```

指定路径：

```
String url="jdbc:odbc:db_database22";
```

实现过程

（1）配置ODBC数据源。在“控制面板”窗口中双击“管理工具”图标，打开图4.2所示的“管理工具”窗口。

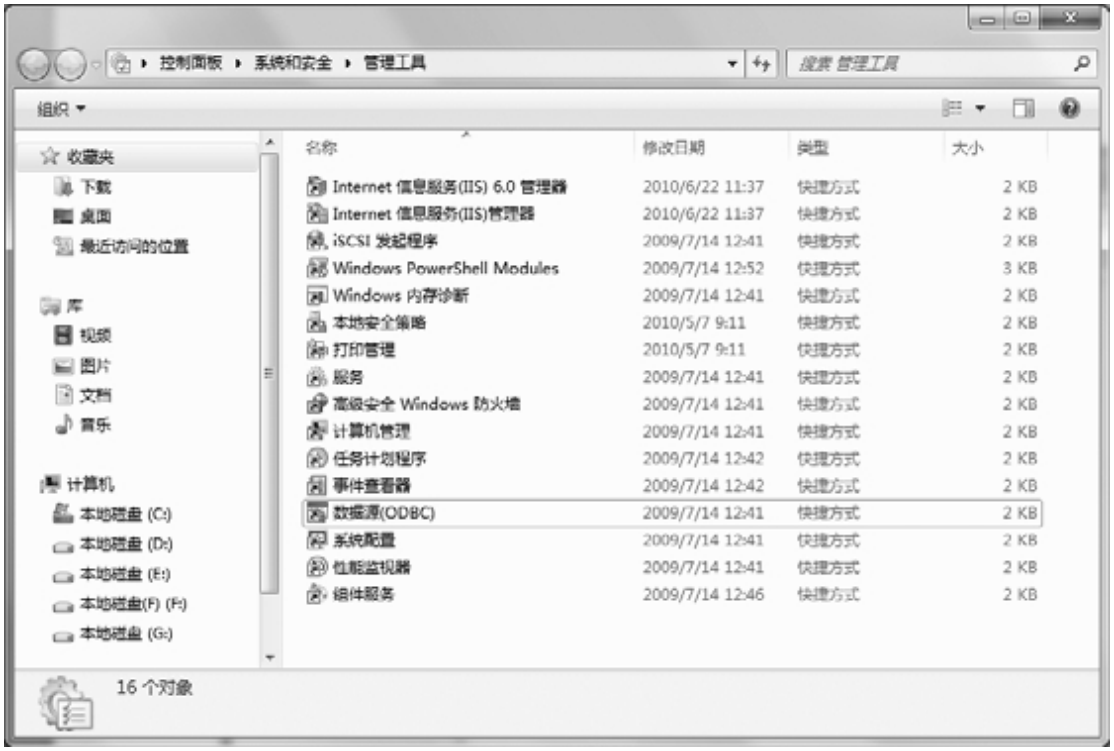


图4.2 “管理工具”窗口

（2）在“管理工具”窗口中双击“数据源（ODBC）”图标，打开图4.3所示的“ODBC数据源管理器”对话框。

（3）在“ODBC数据源管理器”对话框中选择“系统DSN”选项卡，然后单击“添加”按钮，打开图4.4所示的“创建新数据源”对话框。

图4.3 “ODBC 数据源管理器”对话框

图4.4 “创建新数据源”对话框

（4）在“创建新数据源”对话框中双击SQL Server 列表项，打开图4.5 所示的对话框。

（5）在“名称”文本框中输入数据源名称，本实例为db_database22；在“服务器”下拉列表框中选择合适的服务器（local为本机服务器），然后单击“下一步”按钮，打开图4.6所示的对话框。

图4.5 配置DSN 名称及连接的SQL Server 服务器

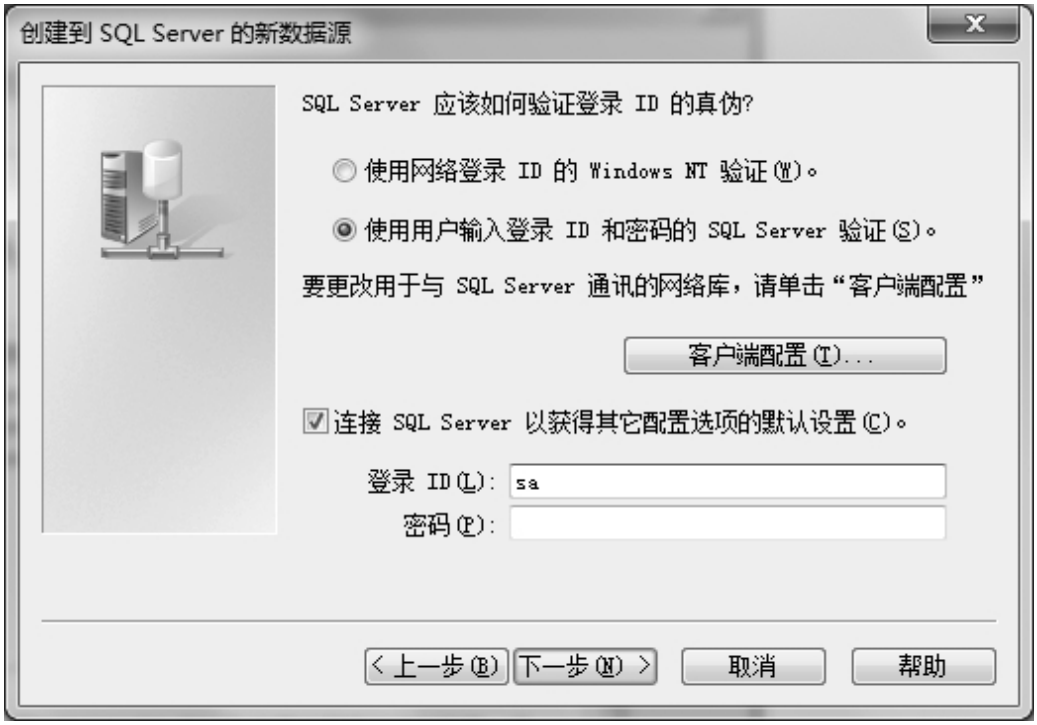


图4.6 设置与数据库系统连接的用户账号

（6）在图4.6 所示的对话框中选中“使用用户输入登录ID 和密码的 SQL Server 验证”单选按钮，此时，“登录ID”和“密码”文

本框被激活。在“登录ID”文本框中输入“sa”（本机服务器的默认ID为sa，密码为空）。单击“下一步”按钮，打开图4.7所示的对话框。

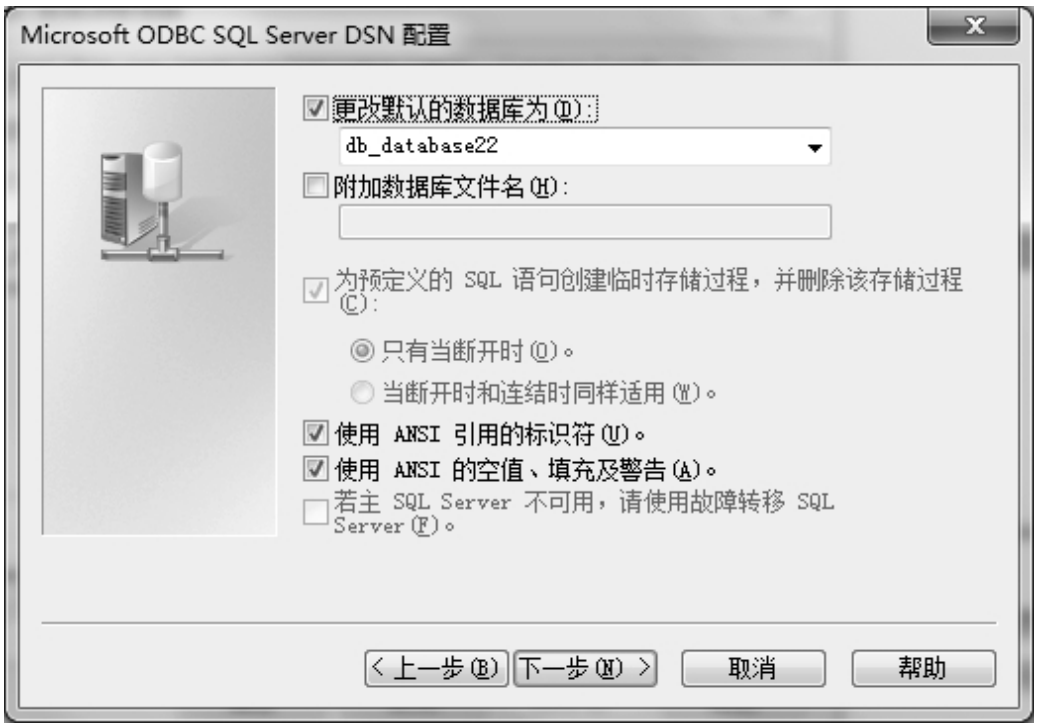


图4.7 指定数据库

（7）在图 4.7 所示的对话框中选中“更改默认的数据库为”复选框，并在可用的数据库下拉列表框中选择要使用的数据库，本实例为 db_database22。单击“下一步”按钮，打开图 4.8所示的对话框，该对话框用于设置有关ODBC 的一些杂项，包括更改SQL Server 系统消息的语言、执行字符数据翻译等。

（8）单击“完成”按钮，打开“ODBC Microsoft SQL Server 安装”对话框，如图4.9 所示。在该对话框中会显示用户设置的信息，并提供“测试数据源”按钮。

图4.8 设置ODBC 杂项

图4.9 “ODBC Microsoft SQL Server 安装”对话框

（9）单击“测试数据源”按钮，测试数据源的正确性，系统会弹出显示测试结果的对话框。如果确认连接无误，连续确认操作，完成

创建系统DSN的任务。

（10）数据源测试成功后，可以在Java程序中通过JDBC-ODBC桥连接数据库。具体代码如下：

```
private Connection conn;          //定义Connection对象
public String con(){
  try {
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
  //加载ODBC数据库驱动
//获取数据库连接
    conn =
DriverManager.getConnection("jdbc:odbc:db_database22","sa",
"");
    return "数据库连接成
功！";                                    //返回连接对象
  } catch (Exception e) {
    e.printStackTrace();
    return "数据库连接失败！";
  }
}
```

举一反三

根据本实例，读者可以开发以下程序。

通过JDBC-ODBC桥建立数据库连接。

创建数据库驱动类。

实例150 JDBC-ODBC桥连接Access数据库

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\04\Ex04_150

实例说明

Access 作为关系型桌面数据库管理系统，在建立中小型数据库管理系统中得到了非常广泛的应用。通过 JDBC-ODBC桥连接数据库是一种很简单的方法，无须在项目中添加驱动文件。本实例实现的是通过JDBC-ODBC桥连接Access数据库，连接成功后，运行结果如图4.10所示。

图4.10 实例运行结果

技术要点

在使用 ODBC 时，经常提到 DSN 这个名词。DSN（Data Source Name）是指数据源名。ODBC是一种访问数据库的方法，只要系统中有相应的ODBC驱动程序，任何程序都可以通过ODBC驱动程序操纵数据库。

在给ODBC驱动程序传递SQL指令时，通过DSN来告诉ODBC驱动程序到底操作哪一个数据库。如果数据库的平台发生改变，如改为SQL Server 数据库，只要其中表的结构没变，就不用改写程序，只需重新在系统中配置DSN即可。

由此可见，DSN 是应用程序和数据库之间的桥梁，要通过 ODBC 访问数据库，前提是必须配置好DSN。即为DSN指定一个名称，而这个名称的作用就是通知系统调用哪个ODBC驱动程序。

实现过程

（1）配置Microsoft Access 数据库文件的DSN。选择“控件面板”→“管理工具”命令，再双击“数据源（ODBC）”图标，打开图

4.11所示的“ODBC数据源管理器”对话框。

（2）选择“系统DSN”选项卡，再单击“添加”按钮，打开图4.12所示的“创建新数据源”对话框。

图4.11 “ODBC 数据源管理器”对话框

图4.12 “创建新数据源”对话框

（3）从列表框中选择Microsoft Access Driver 列表项，然后单击“完成”按钮，即可打开图4.13 所示的“ODBC Microsoft Access 安装”对话框。

（4）在“数据源名”文本框中输入数据源名称Access，然后单击“选择”按钮，打开图4.14所示的“选择数据库”对话框。在该对话框中选择要和数据源连接的数据库，单击“确定”按钮。

图4.13 “ODBC Microsoft Access 安装”对话框

图4.14 “选择数据库”对话框

（5）至此，完成Microsoft Access 数据库文件DSN 的配置工作。

（6）数据库创建成功后，可以通过Java应用程序测试数据源是否连接成功。在项目中创建Java类GetConnectionAccess，在该类中定义验证数据库连接的方法。具体代码如下：

```
public boolean Connection(){
  try {
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
  //加载数据库驱动
    Connection con =
DriverManager.getConnection("jdbc:odbc:access");        /
```

```
/获取数据库连接
    if(con != null){
      System.out.println("通过JDBC-ODBC桥连接Access数据
  库");
    }
    return true;
  } catch (Exception e) {
    e.printStackTrace();
    return false;
  }
}
```

举一反三

根据本实例，读者可以实现以下功能。

尝试用JDBC连接其他类型的数据库。

通过JDBC-ODBC桥连接Access数据库。

实例151 JDBC-ODBC桥连接Oracle数据库

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\04\Ex04_151

实例说明

Oracle数据库可存储海量数据，数据结构复杂，是企业级开发的首选。本实例向大家介绍的是应用JDBC-ODBC桥连接Oracle数据库，配置好数据源之后，就可以在应用程序中测试是否连接成功，连接成功给出图4.15所示的运行结果。

图4.15 实例运行结果

技术要点

本实例实现通过JDBC-ODBC桥连接Oracle数据库，与连接其他数据库相同，都是要在控制面板中配置好数据源后，才能通过Java程序与数据库建立连接。

实现过程

（1）配置Oracle 数据库文件的DSN。选择“控件面板”→“管理工具”命令，再双击“数据源（ODBC）”图标，打开图4.16所示的“ODBC数据源管理器”对话框。

（2）选择“系统DSN”选项卡，再单击“添加”按钮，打开图4.17所示的“创建新数据源”对话框。

图4.16 “ODBC 数据源管理器”对话框

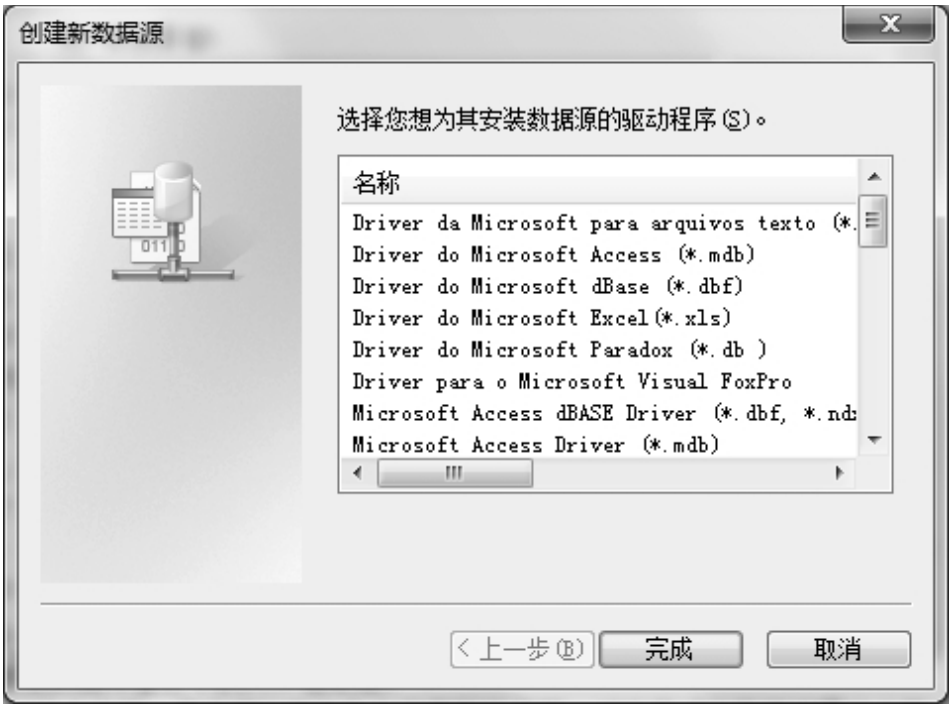


图4.17 “创建新数据源”对话框

（3）从列表框中选择Oracle in OraDb10g_home1 列表项，如图4.18 所示，然后单击“完成”按钮，即可打开图4.19所示的“Oracle数据源配置”对话框。

图4.18 选择Oracle in OraDblog_home 列表项

图4.19 Oracle 数据源配置

（4）在Data Source Name 文本框中输入数据源名称data，然后单击“OK”按钮，完成Oracle数据源配置。

（5）至此，完成Oracle数据库文件DSN的配置工作。

（6）在项目中创建类CreateOracleJoin，来测试是否可通过JDBC-ODBC桥连接Oracle数据库，在该类中定义连接数据库的方法Connection()。具体代码如下：

```
public boolean Connection(){
  try {
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
  //加载数据库驱动
    System.out.println("数据库驱动加载成功！！");
    //获取数据库连接
    Connection con =
DriverManager.getConnection("jdbc:odbc:data","system","aa
a");
    if(con != null){        //判断Connection对象是否为空
      System.out.println("成功地与Oracle数据库建立连
  接！！");        //给出提示信息
    }
```

```
            return true;
        } catch (Exception e) {
            e.printStackTrace();
            return false;
        }
    }
```

举一反三

根据本实例，读者可以实现以下功能。

封装数据库连接。

JDBC-ODBC桥连接Oracle数据库。

4.2 JDBC技术连接数据库

实例152 通过JDBC连接SQLServer 2000数据库

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\04\Ex04_152

实例说明

JDBC是Java连接数据库的一种非常快速和有效的方法。但是JDBC不能直接使用，必须配合数据库提供商所提供的数据库连接驱动才能实现数据库的连接。本实例应用JDBC技术连接SQL Server 2000 数据库。实例运行效果如图4.20 所示。

图4.20 实例运行效果

技术要点

使用JDBC连接数据库主要分为3个步骤，即定义数据库连接字符串、加载数据库连接的驱动和创建数据库连接。

（1）定义数据库连接字符串，其关键代码如下：

```
String url
="jdbc:jtds:sqlserver://localhost:1433;DatabaseName=db_database22"; //连接数据库URL
String userName ="sa";          //连接数据库的用户名
String passWord ="";          //连接数据库的密码
private Connection con = null;          //定义操作接口变量
```

```
private Statement stmt = null;          //定义操作接口变量
private ResultSet rs = null;          //定义操作接口变量
```

（2）加载数据库连接的驱动。在 Java 中使用 Class.forName 来加载数据库驱动。通过Class.forName()方法注册SQL Server 数据库驱动类的关键代码如下：

```
Class.forName("com.microsoft.jdbc.sqlserver.SQLServerDriver");                    //注册SQL Server数据库驱动
```

（3）创建数据库连接。在Java中使用位于java.sql包中的DriverManager类管理JDBC驱动程序的基本服务，通过java.sql包中的Connection接口与特定的数据库进行连接。其语法如下：

```
Connection con = DriverManager.getConnection(url, userName, passWord);          //建立数据库连接
```

实现过程

（1）定义 CreateJoin 类，用于创建与数据库的连接。在该类的静态块中加载数据库驱动，具体代码如下：

```
static {
  try {
    Class.forName("net.sourceforge.jtds.jdbc.Driver");
      //加载数据库驱动
    System.out.println("数据库驱动加载成功！");
  } catch (ClassNotFoundException e) {
    e.printStackTrace();
  }
}
```

（2）在该类中定义连接数据库的方法getConn()，该方法以数据库连接对象Connection作为返回值。具体代码如下：

```
public Connection getConn() {
```

```
  //连接数据库URL
    String url
="jdbc:jtds:sqlserver://localhost:1433;DatabaseName=db_data
base22";
    String userName=
"sa";                                          //连接数据
库的用户名
    String passWord=
"";                                            //连接数据
库的密码
    try {
      conn = DriverManager.getConnection(url, userName,
  passWord);        //获取数据库连接
      if (conn != null) {
        System.out.println("已成功地与SQL Server 2000数据库
    建立连接！");
      }
    } catch (SQLException e) {
      e.printStackTrace();
    }
    return conn;        //返回Connection对象
  }
```

举一反三

根据本实例，读者可以开发以下程序。

使用mssqlserver.jar、msutil.jar和msbase.jar驱动包。

JDBC 技术连接SQL Server 2000 数据库。

实例153 JDBC连接MySQL数据库

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\04\Ex04_153

实例说明

MySQL 数据库以其易于使用和管理、跨平台、开源等优点备受程序员的青睐，在使用MySQL数据库时，首先要建立与MySQL数据库的连接。本实例向大家介绍的是利用JDBC技术与MySQL数据库建立连接，实例运行效果如图4.21所示。

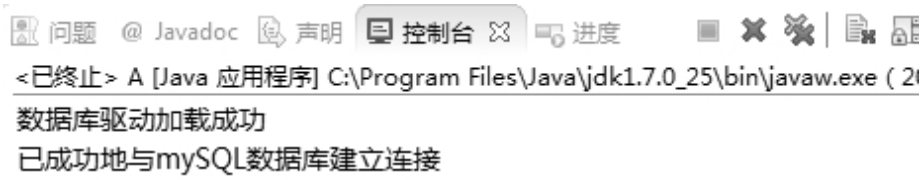


图4.21 实例运行效果

技术要点

目前，MySQL的JDBC包主要有Jconnector和org.git.mm.mysql，下面分别进行介绍。

□ Jconnector 包

该包是MySQL官方网站公布的，其更新速度比较快，很多程序员都使用该包。

□ org.git.mm.mysql包

该包是国外一些Java爱好者编写的，出现的时间比较长，国际化程度做得比较好，而且对中文支持也比较好。本实例使用的就是该包。

连接MySQL数据库的驱动程序，代码如下：

org.gjt.mm.mysql.Driver

URL地址的代码如下：

jdbc:mysql://IP:PORT/databaseName?
user=UserName&password=PWD&useUnicode=true

代码说明

● IP：是指MySQL 主机的IP 地址。

● PORT：是指MySQL 主机的端口号，3306 为安装MySQL 时的默认端口号。

● useUnicode：用于设置是否使用Unicode 输出。

实现过程

创建类CreateMySQL，用于建立与MySQL数据库的连接。在该类中定义getConnection()方法，以数据库连接对象Connection作为返回值。具体代码如下：

```
public Connection getConnection() {
  try {
    Class.forName("com.mysql.jdbc.Driver");
              //加载MySQL数据库驱动
    System.out.println("数据库驱动加载成功！！");
    String url=
"jdbc:mysql://localhost:3306/db_database22";
    //定义连接数据库的URL
    String user=
"root";                                          //定义
连接数据库的用户名
    String passWord=
"111";                                          //定义连接
数据库的密码
```

```
            conn=DriverManager.getConnection(url,user,passWord);
                        //获取数据库连接
            System.out.println("已成功地与MySQL数据库建立连
接！！");
        } catch (Exception e) {
            e.printStackTrace();
        }
        return conn;
    }
```

举一反三

根据本实例，读者可以实现以下功能。

连接MySQL数据库。

写出已经学过的数据库驱动。

实例154 JDBC连接SQL Server 2005数据库

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\04\Ex04_154

实例说明

SQL Server 2005数据库比SQL Server 2000数据库有了很大的改进，因此在获取SQL Server 2005 数据库的连接与获取SQL Server 2000 数据库的连接时有一些差距。本实例为大家介绍通过JDBC 技术与SQL Server 2005 数据库建立连接，实例运行效果如图4.22 所示。

图4.22 实例运行效果

技术要点

连接SQL Server 2005 数据库应用的驱动程序是sqljdbc.jar，可以到microsoft 的官方网站上下载，网址为http://www.microsoft.com。很多初学者会使用连接2000数据库的代码来连接2005，结果导致一些问题。连接2000与连接2005的驱动与URL的差别如表4.1所示。

表4.1 连接2000与连接2005的驱动与URL的差别

数 据 库	驱　　动	URL
SQL Server 2000	com.microsoft.jdbc.sqlserver.SQLServerDriver	jdbc:microsoft:sqlserver://localhost:1433;DatabaseName= 数据库名称
SQL Server 2005	com.microsoft.sqlserver.jdbc.SQLServerDriver	Jdbc:sqlserver://localhost:1433;DatabaseName=数据库名称

实现过程

在项目中创建类CreateConn，用于获取与SQL Server 2005 数据库的连接，在该类中定义连接数据库的方法getConnection()。具体代码如下：

```
    private Connection conn
;                                                    //定义
Connection对象
    public Connection getConnection(){          //定义连接数据
库的方法
      try {
        Class.forName("com.microsoft.sqlserver.jdbc.SQLServer
    Driver");          //加载数据库驱动
        System.out.println("数据库驱动加载成功！");
        //定义连接数据库URL
        String url
    ="jdbc:sqlserver://localhost:1433;DatabaseName=db_databas
```

```
e22";
    String userName ="sa";
    String passWord ="";
    conn = DriverManager.getConnection(url,userName
,passWord);        //获取数据库连接
    if(conn != null){
      System.out.println("已成功地与SQL Server 2005数据库
  建立连接！");
    }
  } catch (Exception e) {
    e.printStackTrace();
  }
  return conn;
}
```

举一反三

根据本实例，读者可以实现以下功能。

使用JDBC 技术与SQL Server 2005 数据库建立连接。

使用JDBC 技术与SQL Server 2005 数据库断开连接。

实例155 JDBC技术连接Oracle数据库

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\04\Ex04_155

实例说明

Oracle数据库的数据管理功能比较强大，对计算机的要求也比较高。但Oracle数据库在IT行业所占的地位比较重要，要求程序员必须学会使用，使用Oracle数据库的前提是与数据库建立连接。本实例向

大家介绍应用JDBC技术连接Oracle数据库，实例运行效果如图4.23所示。

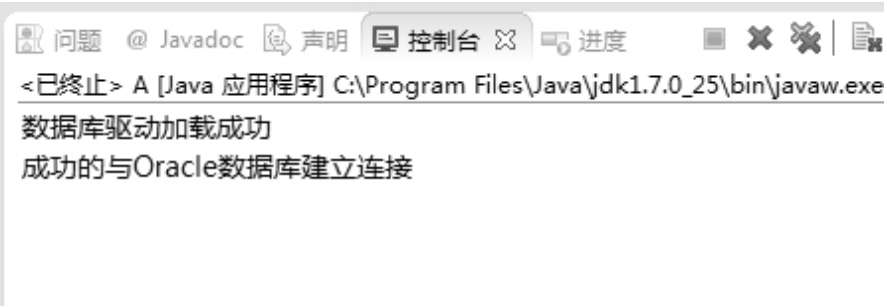


图4.23 实例运行效果

技术要点

通过JDBC连接数据库仍然可分为两步，即加载数据库驱动和获取数据库连接，本实例连接Oracle数据库应用的驱动是classes12.jar。加载数据库驱动的代码如下：

```
Class.forName("oracle.jdbc.driver.OracleDriver");
```

在获取数据库连接时，需要定义连接数据库的URL，连接Oracle与连接SQL Server 数据库的URL有较大差异。本实例获取数据库连接的URL地址，代码如下：

```
String url="jdbc:oracle:thin:@localhost:1521:orcl3";
```

参数说明

- @：分隔符。
- 1521：数据库端口。
- orcl3：数据库名或（SID）。

在项目中创建 CreateOracle 类，在该类中定义连接数据库的方法 getConnection()，该方法以Connection对象作为返回值。具体代码如下：

```
public Connection getConnection() {
  Connection conn = null;
  try {
```

```
        Class.forName("oracle.jdbc.driver.OracleDriver");
                                //加载数据库驱动
        System.out.println("数据库驱动加载成
    功！");                                    //输出的信息
        String url=
    "jdbc:oracle:thin:@localhost:1521:orcl3";
            //获取连接URL
        String user=
    "system";                                                    //
    连接用户名
        String password=
    "aaa";                                                  //连
    接密码
        Connection
    con=DriverManager.getConnection(url,user,password);
            //获取数据库连接
        if (con != null) {
          System.out.println("成功地与Oracle数据库建立连
      接！！");
        }
      } catch (Exception e) {
        e.printStackTrace();
      }
      return
conn;
      //返回Connection实例
    }
```

举一反三

根据本实例，读者可以实现以下功能。

应用JDBC技术连接Oracle数据库。

封装JDBC技术连接Oracle数据库的代码。

实例156 JDBC连接JavaDB数据库

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\04\Ex04_156

实例说明

JavaDB是在安装JDK时自动安装的，不需要另外安装数据库系统，使用起来很简单。对于小型应用程序，使用JavaDB数据库非常方便，但JavaDB数据库并没有提供企业管理器等用户交互界面。在使用该数据库时需要注意，本实例实现的是通过Java程序连接JavaDB数据库。在连接的过程中需要注意，如果连接的数据库不存在，需要通过程序创建相应的数据库；如果连接成功，将给出图4.24所示的运行结果。

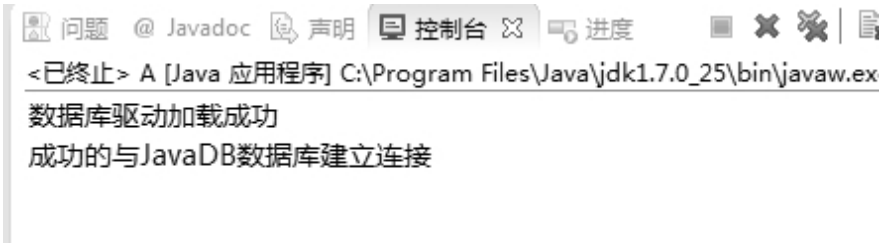


图4.24 实例运行结果

技术要点

连接JavaDB数据库需要的驱动为derby.jar，读者可在相应的网站上下载。连接JavaDB数据库与连接其他类型的数据库方法相同，读者可参考本节中其他连接数据库的实例。

实现过程

（1）在项目中创建 CreateJavaDBJoin类，用于建立JavaDB数据库的连接，在该类中定义表示连接数据库驱动与连接数据库URL的字符串对象。代码如下：

```
    //数据库驱动
    private static final String DRIVERCLASS
="org.apache.derby.jdbc.EmbeddedDriver";
    private static final String URL
="jdbc:derby:db_database22";        //数据库URL
    //创建用来保存数据库连接的线程
    private static final ThreadLocal<Connection> threadLocal
= new ThreadLocal<Connection>();
    private static Connection conn = null;        //数据库连
接
```

（2）在该类的静态块中加载数据库驱动。如果要连接的数据库不存在，实现新建数据库，并获取与该数据库的连接，具体代码如下：

```
    static {
      //通过静态方法加载数据库驱动，并且在数据库不存在的情况
  下创建数据库
      try {
        Class.forName(DRIVERCLASS);
                  //加载数据库驱动
        System.out.println("数据库驱动加载成功！！");
        File albumF=new
    File("db_database22");                                    //
    创建数据库文件对象
        if (!albumF.exists())
    {                                                        //判断数
```

据库文件是否存在

```
        String[] sqls=new
    String[1];                                          //定义
    创建数据库的SQL语句
        sqls[0] ="create table tb_album (name varchar(200)
    not null)";
      } else {
        conn=DriverManager.getConnection(URL+
    ";create=true");                    //创建数据库连接
        System.out.println("已成功地与JavaDB数据库建立连
    接！！");
        threadLocal.set(conn);
                    //保存数据库连接
      }
    } catch (Exception e) {
      e.printStackTrace();
    }
}
```

举一反三

根据本实例，读者可以实现以下功能。

通过Java程序连接JavaDB数据库。

断开数据库的连接。

4.3 数据库与数据表

实例157 列举SQL Server数据库下的数据表

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\04\Ex04_157

实例说明

在程序开发时，有时需要创建大量的数据表。要获取数据库中的所有数据表信息，可以通过java.sql包中的DatabaseMetaData接口提供获取数据库综合信息的方法。本实例实现获取SQL Server指定数据库中的所有数据表信息，并将其结果在控制台上输出。实例运行效果如图21.25所示。

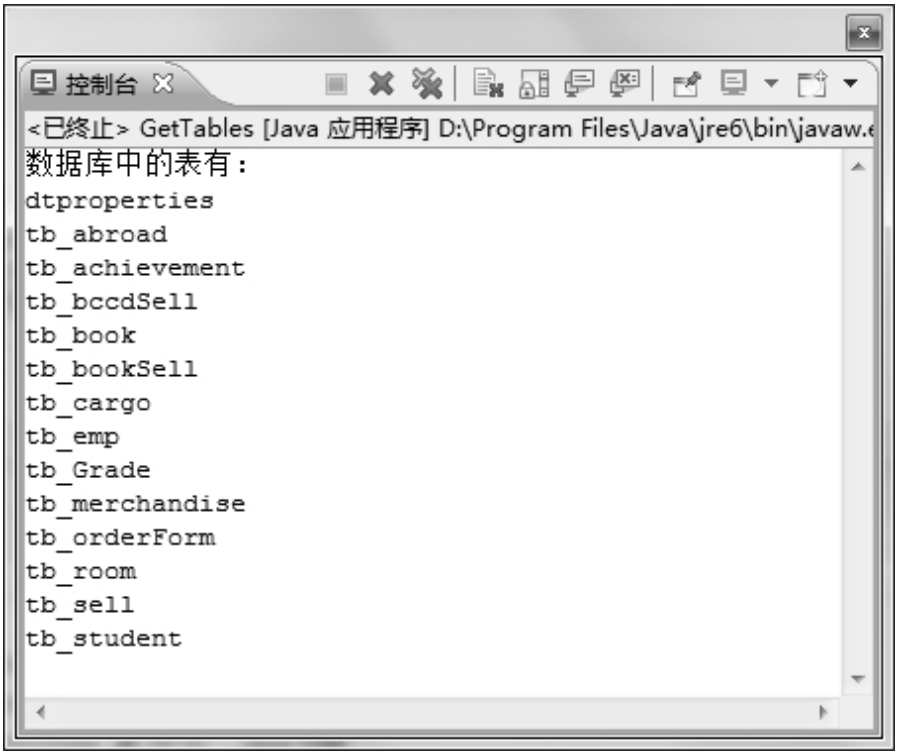


图4.25 实例运行效果

技术要点

DatabaseMetaData 接口由驱动程序供应商实现，让用户了解 Database Management System （DBMS）与驱动程序相结合时的功能。不同的关系数据库管理系统常常支持不同的功能，它能够以不同方式实现这些功能，且可以使用不同的数据类型。此外，驱动程序可以实现DBMS提供的顶级功能。

使用该接口中的getTables()方法，可获取指定数据库中的所有数据表名。getTables()方法返回ResultSet数据结果集。具体语法格式如下：

getTables(String catalog, String schemaPattern, String tableNamePattern, String[] types)

该方法的参数说明如表4.2所示。

表4.2 getTables()方法的参数说明

参　数	类　型	描　述
catalog	String	类别名称，必须与存储在数据库中的类别名称匹配
schemaPattern	String	模式名称，必须与存储在数据库中的模式名称匹配
tableNamePattern	String	表名称模式，必须与存储在数据库中的表名称匹配
types	String[]	要包括的表类型所组成的列表

实现过程

（1）创建类GetTables，在该类中首先定义连接数据库的方法getConn()，该方法以Connection对象作为返回值。具体代码如下：

```
public static Connection getConn() {
  try {
    Class.forName("net.sourceforge.jtds.jdbc.Driver");
                //加载数据库驱动
  } catch (ClassNotFoundException e) {
    e.printStackTrace();
}
```

```
  //连接数据库URL
    String url
="jdbc:jtds:sqlserver://localhost:1433;DatabaseName=db_data
base20";
    String userName=
"sa";                                          //连接数据库的
用户名
    String passWord=
"";                                            //连接数据库的
密码
    try {
      conn=DriverManager.getConnection(url,userName,passWor
  d);      //获取数据库连接
      if (conn != null) {
      }
    } catch (SQLException e) {
      e.printStackTrace();
    }
    return
conn;                                                  //返回
Connection对象
  }
```

（2）在该类中定义查询数据库方法GetRs()，该方法以ResultSet作为返回值，将数据库中的所有表取出来。具体代码如下：

```
  public static ResultSet GetRs() {
    try {
```

```
    String[] tableType=
{"TABLE"};                                  //指定要进行查询
的表类型
    Connection
conn=getConn();                                  //调用与数
据库建立连接的方法
    DatabaseMetaData databaseMetaData =
conn.getMetaData();          //获取DatabaseMetaData实例
    //获取数据库中所有数据表集合
    ResultSet resultSet =
databaseMetaData.getTables(null, null, "%", tableType);
    return resultSet;
  } catch (SQLException e) {
    System.out.println("记录数量获取失败！");
    return null;
  }
}
```

举一反三

根据本实例，读者可以实现以下功能。

查看SQL Server 数据库中的系统表。

查看SQL Server 下的表。

实例158 列举MySQL数据库下的数据表

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\04\Ex04_158

实例说明

在实际开发中有时需要获取指定数据库的数据库表，本实例向大家介绍的是如何获取MySQL 数据库中的数据表名称。运行本实例，在控制台上显示指定数据库下的数据表，结果如图4.26所示。

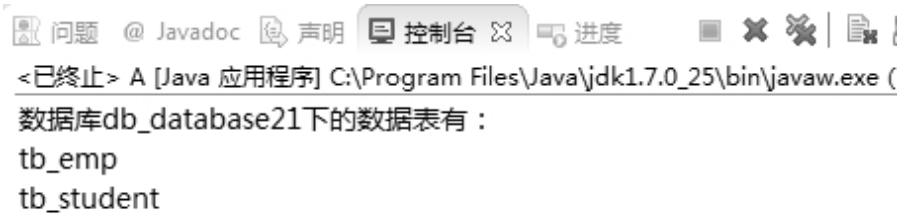


图4.26 实例运行结果

技术要点

在MySQL中通过SHOW语句可以列出指定数据库中的数据表。语法格式如下：

SHOW TABLES [FROM databaseName] [LIKE expression];

参数说明

● databaseName 子句：可选项，用于指定要获取数据表的数据库。省略该子句，则列举当前打开数据库中的数据表，如果当前没有打开数据库，则返回错误信息。

● expression 子句：可选项，用于设置列举条件。expression是一个字符型表达式，可以包括通配符，如百分号（代表多个字符）、下划线（代表一个字符）等。

例如，“LIKE ’%book%’”将获取名称中包括book 的数据表。

实现过程

（1）创建类 GetTables ，在该类中定义查询数据方法。首先定义连接数据库的方法getConnection()，具体代码读者可参考光盘中的源程序，这里不再赘述。

（2）在该类中定义查询数据库的方法 listDB()，用于查询数据库中的所有数据表。具体代码如下：

```
  public ResultSet listDB() {
    String sql= "show
tables;";                                          //定义查
询数据SQL语句
    try {
      conn=getConnection();
          //获取数据库连接
      Statement stmt=
  conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,Re
  sultSet.CONCUR_READ_ONLY);                        //实例
  化Statement对象
      ResultSet rs=
  stmt.executeQuery(sql);                             //执
  行查询SQL语句
      return
  rs;                                                  //
  返回查询结果
    } catch (SQLException ex) {
      System.out.println(ex.getMessage());
      return null;
    }
  }
```

举一反三

根据本实例，读者可以实现以下功能。

在MySQL数据库下添加新表。

MySQL数据库下的系统表。

实例159 查看数据表结构

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\04\Ex04_159

实例说明

数据表结构是数据库维护的主要操作，查看数据表结构有助于操作者了解软件的内部体系结构。本实例实现查看用户所选的数据表结构，运行程序，可在窗体中将用户选择的数据表结构显示在表格中。实例运行效果如图4.27所示。

图4.27 实例运行效果

技术要点

本实例主要应用SQL Server系统表Sysobjects（系统对象）、Syscolumns（字段）、Syspro perties （描述）、Systypes（字段类别）和Syscomments（默认值）等相对应的表关系显示出表的结构。这些系统表都可以通过对象编号和相关表编号进行关联。

实现过程

（1）创建类GetFrame，用于定义查询数据库的方法。在该类中首先定义连接数据库的方法Con()，具体代码读者可参考光盘中的源程

序，这里不再赘述。

（2）本实例中，实现将数据库中所有数据表显示在下拉列表中。

（3）在该类中定义GetRs()方法，指定SQL语句。具体代码如下：

```
public ResultSet GetRs(final String SQL) {
  try {
    Connection
Con=Con();                                          //获取
数据库连接
    Statement Smt =
Con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,Resul
tSet.CONCUR_UPDATABLE);                    //获取Statement
对象
    ResultSet
Rs=Smt.executeQuery(SQL);                                //执
行查询语句获取查询结果集
    return Rs;
  } catch (SQLException e) {
    System.out.println("记录数量获取失败！");
    return null;
  }
}
```

（4）在该类中定义获取数据表结构的方法 getMessage()，该方法用于查询数据表结构，并以List作为返回值。具体代码如下：

```
public List getMessage(String tableName) {
  List list = new ArrayList();        //定义保存返回值的
List集合
```

```
String SQL =" Select case when c.colid=1 then o.name end 表名,"
    +" c.ColId 字段编号,c.name 字段名,c.length 字段长度,t.name 字段类别,"
    +" p.value 描述,case when c.isnullable=0 then '1' end 是否为空,"
    +" c.scale 小数位数,REPLACE (REPLACE (REPLACE (m.text,'(',''),')',''),'''','') 默认值,"
    +" case when (" Select Count(*) From SysObjects where name in ("
    +" Select name From Sysindexes Where id=c.id and indid in ("
    +" Select indid From Sysindexkeys where id=c.id and colid in ("
    +" Select colid From Syscolumns where id=c.id and colid=c.colid))) and xtype='pk')>0"
    +" then '1' end 是否为主键"
    +" From Sysobjects o"
    +" left join Syscolumns c on o.id=c.id"
    +" left join Sysproperties p on o.id=p.id and c.colid=p.smallid"
    +" left join Systypes t on t.xtype=c.xtype"
    +" left join Syscomments m on m.id=c.cdefault"
    +" where (o.xtype='u' or o.xtype='v') and o.status>0 and o.name='"
    + tableName+ "'"+ " order byo.name,c.colid";                    //定义查询SQL语句
```

```
    ResultSet
res=GetRs(SQL);                                          //
调用执行SQL语句方法
    ResultSetMetaData
Rsmd;                                                    //获取
ResultSetMetaData方法
    try {
      Rsmd=
  res.getMetaData();                                      /
  /实例化 ResultSetMetaData对象
      while (res.next())
  {                                                       //循环遍历查
  询结果集
        Student student=new
    Student();                                           //创建与数据
    库对应的JavaBean对象
        student.setId(res.getString("字段编
    号"));                                  //设置对象属性
        student.setName(res.getString("字段名"));
        student.setType(res.getString("字段类别"));
        student.setAcquiescence(res.getString("默认值"));
        student.setDepict(res.getString("描述"));
        student.setDigit(res.getString("小数位数"));
        student.setLength(res.getString("字段长度"));
        student.setIfNull(res.getString("是否为空"));
        list.add(student);
              //将对象添加到List集合中
```

```
        }
      } catch (SQLException e) {
        e.printStackTrace();
      }
      return
list;
//返回List集合
  }
```

举一反三

根据本实例，读者可以实现以下功能。

查看自定义表结构。

对自定义表结构的修改。

实例160 动态维护投票数据库

这是一个可以用来提高基础性能的实例

实例位置：光盘\mingrisoft\04\Ex04_160

实例说明

数据库维护只指向数据库中添加和删除数据库表中的字段，本实例以投票数据库为例，向大家介绍如何通过Java程序实现数据库表的维护。实例运行效果如图4.28所示。

图4.28 实例运行效果

技术要点

本实例主要使用ALTER TABLE 语句中的ADD 和DROP 子句。

□ ADD 子句

该子句用于向数据表中添加字段。该子句的语法格式如下：

```
ALTER TABLE table
ADD [ < column_definition > ] | column_name AS
computed_column_expression
```

参数说明

- table：添加字段的数据表名称。
- column_definition：字段的定义。
- column_name：字段的名字。
- computed_column_expression：计算字段的表达式。

□ DROP 子句

该子句用于删除数据表中的字段。该子句的语法格式如下：

```
ALTER TABLE table
DROP constraint_name
```

参数说明

- table：需要删除字段的数据表名。
- constraint_name：需要删除字段的名称。

实现过程

（1）在项目中创建窗体类BallotUtilFrame，该类继承自JFrame类，实现窗体类。向该窗体中添加选项卡面板、标签控件、文本框控件与按钮控件等实现窗体布局。

（2）在项目中创建工具类BallotUtil，在该类中定义删除数据表中字段与添加数据表中字段的方法，该方法有两个String类型的参数，分别用于指定字段名称与字段的数据类型。添加数据表字段方法的代码如下：

```
public void addField(String fieldName,String type) {
  conn=getConn();
  //获取数据库连接
  try {
    Statement statement=
conn.createStatement();                    //获取
Statement方法
    String sql= "alter table tb_ballot add "+fieldName+"
"+type;          //向数据表中添加字段
    statement.executeUpdate(sql);
        //执行更新数据表SQL语句
    conn.close();
    //关闭数据库连接
  } catch (Exception e) {
    e.printStackTrace();
  }
}
```

（3）在工具类中定义删除数据表中字段的方法，该方法有一个String类型的参数，用于指定要删除的数据表的字段。具体代码如

下：

```
public void deleteField(String fieldName) {
  conn=getConn();
  //获取数据库连接
  try {
    Statement statement = conn.createStatement();
    //定义从数据库中删除字段的SQL语句
    String sql ="alter table tb_ballot drop column
"+fieldName;
    statement.executeUpdate(sql);
        //执行删除操作
    conn.close();
  } catch (Exception e) {
    e.printStackTrace();
  }
}
```

举一反三

根据本实例，读者可以实现以下功能。

向数据库中添加表中的字段。

向数据库中删除表中的字段。

实例161 SQL Server数据备份

这是一个可以提高基础性能的实例

实例位置：光盘\mingrisoft\04\Ex04_161

实例说明

对于大型的或者安全性较高的应用程序，都需要具有数据库的备份和恢复功能。这是因为当数据库因某些意外原因被破坏或删除时，将会给企业的经济效益带来巨大的损失。解决该问题的唯一途径就是及时对数据库进行备份。本实例实现对用户选择的数据库进行备份，当用户单击“备份”按钮时，即可生成备份文件，并保存到C盘。实例运行效果如图4.29所示。

图4.29 实例运行效果

技术要点

本实例运用SQLDMO.backup对象完成整个系统数据库的备份，这使得在系统或数据库发生故障（如硬盘发生故障）时可以重建系统。

备份整个数据库的语法如下：

```
BACKUP DATABASE { database_name |@database_name_var }
TO < backup_device > [ ,...n ]
[ WITH
  [ BLOCKSIZE ={ blocksize |@blocksize_variable } ]
  [ [ , ] DESCRIPTION ={ 'text' |@text_variable } ]
  [ [ , ] DIFFERENTIAL ]
  [ [ , ] EXPIREDATE ={ date |@date_var }
    | RETAINDAYS ={ days |@days_var } ]
  [ [ , ] PASSWORD ={ password |@password_variable } ]
  [ [ , ] FORMAT | NOFORMAT ]
  [ [ , ] { INIT | NOINIT } ]
  [ [ , ] MEDIADESCRIPTION ={ 'text' |@text_variable } ]
```

```
    [ [ , ] MEDIANAME ={ media_name |@media_name_variable }
]
    [ [ , ] MEDIAPASSWORD ={ mediapassword
|@mediapassword_variable } ]
    [ [ , ] NAME ={ backup_set_name |@backup_set_name_var }
]
    [ [ , ] { NOSKIP | SKIP } ]
    [ [ , ] { NOREWIND | REWIND } ]
    [ [ , ] { NOUNLOAD | UNLOAD } ]
    [ [ , ] RESTART ]
    [ [ , ] STATS [ = percentage ] ]
  ]
```

参数说明

□ DATABASE：指定一个完整的数据库备份。假如指定了一个文件和文件组的列表，那么仅有这些被指定的文件和文件组被备份。

□ {database_name |@database_name_var }：指定了一个数据库，从该数据库中对事务日志、部分数据库或完整的数据库进行备份。如果作为变量（@database_name_var）提供，则可将该名称指定为字符串常量（@database_name_var = database name）或字符串数据类型（ntext或text数据类型除外）的变量。

□ <backup_device>：指定备份操作时要使用的逻辑或物理备份设备。可以是下列一种或多种形式。

● { logical_backup_device_name } | { @logical_backup_device_name_var } ：是由sp_addumpdevice 创建的备份设备的逻辑名称，数据库将备份到该设备中，其名称必须遵守标识符规则。如果将其作为变量（@logical_backup_device_name_var）提供，则可将该备份设备名称

指定为字符串常量（@logical_backup_device_name_var = logical backup device name）或字符串数据类型（ntext或text数据类型除外）的变量。

● { DISK | TAPE }='physical_backup_device_name' |@physical_backup_device_name_var：允许在指定的磁盘或磁带设备上创建备份。在执行BACKUP语句之前不必存在指定的物理设备。如果存在物理设备且BACKUP语句中没有指定INIT选项，则备份将追加到该设备。

□ n：是表示可以指定多个备份设备的占位符。备份设备数目的上限为 64。

□ BLOCKSIZE = {blocksize | @blocksize_variable}：用字节数来指定物理块的大小。在Windows NT 系统上，默认设置是设备的默认块大小。一般情况下，当SQL Server选择适合于设备的块大小时不需要此参数。在基于Windows 2000的计算机上，默认设置是65536（64KB是SQL Server支持的最大大小）。

□ DESCRIPTION ={ 'text' |@text_variable }：指定描述备份集的自由格式文本。该字符串最长可以有255个字符。

□ DIFFERENTIAL：指定数据库备份或文件备份应该与上一次完整备份后改变的数据库或文件部分保持一致。差异备份一般会比完整备份占用更少的空间。对于上一次完整备份时备份的全部单个日志，使用该选项可以不必再进行备份。

□ EXPIREDATE ={date |@date_var}：指定备份集到期和允许被重写的日期。如果将该日期作为变量（@date_var）提供，则可以将该日期指定为字符串常量（@date_var =date）、字符串数据类型变量（ntext或text数据类型除外）、smalldatetime或者datetime变量，并且该日期必须符合已配置的系统datetime格式。

□ RETAINDAYS ={days|@days_var}：指定必须经过多少天才可以重写该备份媒体集。假如用变量（@days_var）指定，该变量必须为整型。

□ PASSWORD ={ password |@password_variable }：为备份集设置密码。PASSWORD 是一个字符串。如果为备份集定义了密码，必须提供这个密码才能对该备份集执行任何还原操作。

□ FORMAT：指定应将媒体头写入用于此备份操作的所有卷。任何现有的媒体头都被重写。FORMAT选项使整个媒体内容无效，并且忽略任何现有的内容。

□ NOFORMAT：指定媒体头不应写入所有用于该备份操作的卷中，并且不要重写该备份设备，除非指定了INIT。

□ INIT：指定应重写所有备份集，但是保留媒体头。如果指定了INIT，将重写那个设备上的所有现有的备份集数据。

当遇到以下几种情况之一时不重写备份媒体。

● 媒体上的备份设置没有全部过期。

● 如果BACKUP 语句给出了备份集名，该备份集名与备份媒体上的名称不匹配。

□ NOINIT：表示备份集将追加到指定的磁盘或磁带设备上，以保留现有的备份集。NOINIT是默认设置。

□ MEDIADESCRIPTION ={'text'|@text_variable }：指明媒体集的自由格式文本描述，最多为255个字符。

□ MEDIANAME ={ media_name |@ media_name _variable }：为整个备份媒体集指明媒体名，最多为128个字符。假如指定了MEDIANAME，则它必须与以前指定的媒体名相匹配，该媒体名已存在于备份卷中。假如没有指定 MEDIANAME，或指定了SKIP选项，将不会对媒体名进行验证检查。

□ MEDIAPASSWORD={mediapassword |@mediapassword_variable}：为媒体集设置密码。MEDIAPASSWORD是一个字符串。

□ NAME ={ backup_set_name |@backup_set_name_var }：指定备份集的名称。名称最长可达128个字符。假如没有指定NAME，它将为空。

□ NOSKIP：指示 BACKUP语句在可以重写媒体上的所有备份集之前先检查它们的过期日期。

□ SKIP：禁用备份集过期和名称检查，这些检查一般由 BACKUP语句执行以防重写备份集。

□ NOUNLOAD：指定不在备份后从磁带驱动器中自动卸载磁带。设置始终为NOUNLOAD，直到指定UNLOAD为止。该选项只用于磁带设备。

□ UNLOAD：指定在备份完成后自动倒带并卸载磁带。启动新用户会话时其默认设置为UNLOAD。该设置一直保持到用户指定了NOUNLOAD时为止。该选项只用于磁带设备。

□ RESTART：指定 SQL Server 重新启动一个被中断的备份操作。因为 RESTART 选项在备份操作被中断处重新启动该操作，所以它节省了时间。若要重新启动一个特定的备份操作，可重复整个 BACKUP语句并且加入 RESTART 选项。不一定非要使用RESTART选项，但是它可以节省时间。

□ STATS [= percentage]：每当另一个percentage 结束时显示一条消息，它被用于测量进度。如果省略 percentage，SQL Server将每完成10 个百分点显示一条消息。

实现过程

（1）在项目中创建类BackUpFrame，该类继承自JFrame类，实现窗体类。

（2）向窗体中添加控件。实现窗体布局，该窗体中的主要控件及说明如表4.3所示。

表4.3 窗体中的主要控件及说明

控 件 类 型	控 件 命 名	控 件 用 途
JComboBox	dataNamecomboBox	供用户选择要备份的数据库
JTextField	backTextField	用户选择的备份后的文件名称
JButton	backButton	显示“备份”的按钮控件

（3）创建类BackupData，用于定义数据库备份方法，在该类中定义连接数据库的方法Con()，具体代码读者可参考光盘中的源程序，这里不再赘述。

（4）在该类中定义getDatabase()方法，该方法用于获取所有数据库方法，返回值为List集合对象。具体代码如下：

```
    public List getDatabase() {
      List list=new
  ArrayList();                                          //定义
  List集合对象
      Connection
  con=Con();                                            //获取数
  据库连接
      Statement
  st;                                                    //定义
  Statement对象
      try {
        st=
    con.createStatement();                                    /
    /实例化Statement对象
        //指定查询所有数据库方法
```

```
        ResultSet rs = st.executeQuery("select name from
    dbo.sysdatabases");
        while (rs.next())
    {                                          //循环遍历查询结
    果集
          list.add(rs.getString(1));
              //将查询数据添加到List集合中
        }
      } catch (Exception e) {
        e.printStackTrace();
      }
      return
  list;                                        //返回
  查询结果
    }
```

（5）定义执行数据库备份的方法 getBak()，该方法包含两个 String 类型的参数，分别用于定义要进行备份的数据库与备份文件的保存名称。具体代码如下：

```
    public void getBak(String databaseName, String
databasePath) {
      Connection
  con=Con();                                   //获取数
  据库连接
      Statement st;
      try {
        st=
    con.createStatement();                          /
```

```
/实例化Statement对象
    st.executeUpdate("backup database "+databaseName+ "
to disk='"+databasePath+
"'");                                    //指定数据库备份SQL语
句
    con.close();
      //关闭连接
  } catch (SQLException e) {
    e.printStackTrace();
  }
}
```

举一反三

根据本实例，读者可以实现以下功能。

SQL Server 数据库备份数据。

恢复数据。

实例162 SQL Server数据恢复

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\04\Ex04_162

实例说明

对于大型应用程序，具有数据恢复功能非常重要。因为数据恢复功能可以在数据遭到破坏时，将备份的数据恢复到系统中，保证系统重新正常运转，从而避免因数据异常丢失所带来的损失。因此建议读者在程序开发中添加数据库恢复功能。本实例实现将数据库恢复至备份时状态，实例运行效果如图4.30所示。

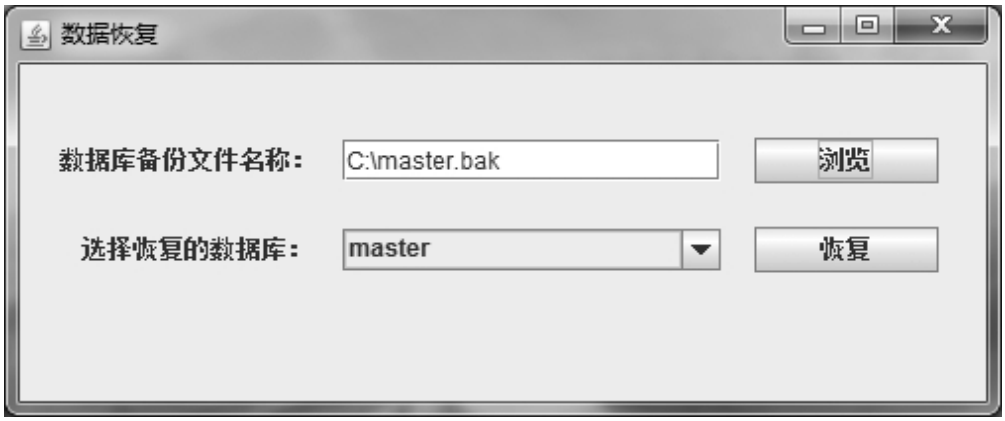


图4.30 实例运行效果

技术要点

本实例运用SQLDMO.Restore对象恢复使用BACKUP命令所做的整个数据库备份。

RESTORE的语法如下：

```
RESTORE DATABASE { database_name |@database_name_var }
[ FROM < backup_device > [ ,...n ] ]
[ WITH
  [ RESTRICTED_USER ]
  [ [ , ] FILE ={ file_number |@file_number } ]
  [ [ , ] PASSWORD ={ password |@password_variable } ]
  [ [ , ] MEDIANAME ={ media_name |@media_name_variable }
]
  [ [ , ] MEDIAPASSWORD ={ mediapassword
|@mediapassword_variable } ]
  [ [ , ] MOVE 'logical_file_name' TO
'operating_system_file_name' ]
    [ ,...n ]
  [ [ , ] KEEP_REPLICATION ]
```

```
    [ [ , ] { NORECOVERY | RECOVERY | STANDBY =
undo_file_name } ]
    [ [ , ] { NOREWIND | REWIND } ]
    [ [ , ] { NOUNLOAD | UNLOAD } ]
    [ [ , ] REPLACE ]
    [ [ , ] RESTART ]
    [ [ , ] STATS [ = percentage ] ]
  ]
```

参数说明

□ DATABASE：指定备份还原整个数据库。如果指定了文件和文件组列表，则只还原那些文件和文件组。

□ {database_name |@database_name_var}：是将日志或整个数据库还原到的数据库。如果将其作为变量（@database_name_var）提供，则可将该名称指定为字符串常量（@database_name_var = database name）或字符串数据类型（ntext 或 text 数据类型除外）的变量。

□ FROM：指定从中还原备份的备份设备。如果没有指定FROM 子句，则不会发生备份还原，而是恢复数据库。可用省略 FROM 子句的办法尝试恢复通过 NORECOVERY选项还原的数据库，或切换到一台备用服务器上。如果省略FROM子句，则必须指定NORECOVERY、RECOVERY或STANDBY。

□ < backup_device >：指定还原操作要使用的逻辑或物理备份设备。可以是下列一种或多种形式。

● @{'logical_backup_device_name' | @logical_backup_device_name_var} ： 是 由sp_addumpdevice 创建的备份设备（数据库将从该备份设备还原）的逻辑名称，该名称必须符合标识符规则。如果作为变量

（@logical_backup_device_name_var）提供，则可以指定字符串常量（@logical_backup_device_name_var = logical_backup_device_name）或字符串数据类型（ntext或text数据类型除外）的变量作为备份设备名。

● @{DISK | TAPE }= 'physical_backup_device_name' |@physical_backup_device_name_var：允许从命名磁盘或磁带设备还原备份。磁盘或磁带的设备类型应该用设备的真实名称（如完整的路径和文件名）来指定：DISK = 'C:\Program Files\Microsoft SQL Server\MySQL\BACKUP\Mybackup.dat'或TAPE = '\\.\TAPE0'。如果指定为变量（@physical_backup_device_name_var），则设备名称可以是字符串常量（@physical_backup_device_ name_var ='physical_backup_device_name'）或字符串数据类型（ntext或text数据类型除外）的变量。

□ n：是表示可以指定多个备份设备和逻辑备份设备的占位符。备份设备或逻辑备份设备最多可以为64个。

□ RESTRICTED_USER：限制只有db_owner、dbcreator 或 sysadmin 角色的成员才能访问最近还原的数据库。

□ FILE ={file_number |@file_number}：标识要还原的备份集。例如，file_number为1表示备份媒体上的第一个备份集，file_number为2表示第二个备份集。

□ PASSWORD ={ password |@password_variable }：提供备份集的密码。PASSWORD 是一个字符串。如果在创建备份集时提供了密码，则从备份集执行还原操作时必须提供密码。

□ MEDIANAME ={media_name |@media_name_variable}：指定媒体名称。如果提供媒体名称，该名称必须与备份卷上的媒体名称相匹配，否则还原操作将终止。如果RESTORE语句没有给出媒体名称，将不对备份卷执行媒体名称匹配检查。

□ MEDIAPASSWORD ={mediapassword |@mediapassword_variable}：提供媒体集的密码。MEDIAPASSWORD是一个字符串。如果格式化媒体集时提供了密码，则访问该媒体集上的任何备份集时都必须提供该密码。

□ MOVE 'logical_file_name' TO 'operating_system_file_name'：指定应将给定的logical_file_name移到operating_system_file_name。默认情况下，logical_file_name将还原到其原始位置。如果使用RESTORE语句将数据库复制到相同或不同的服务器上，则可能需要使用MOVE选项重新定位数据库文件，以避免与现有文件冲突。可以在不同的MOVE语句中指定数据库内的每个逻辑文件。

□ n：占位符，表示可通过指定多个MOVE 语句移动多个逻辑文件。

□ NORECOVERY：指示还原操作不回滚任何未提交的事务。如果需要应用另一个事务日志，则必须指定 NORECOVERY 或 STANDBY 选项。如果 NORECOVERY、RECOVERY和STANDBY均未指定，则默认为RECOVERY。当还原数据库备份和多个事务日志时，或在需要多个RESTORE语句时（如在完整数据库备份后进行差异数据库备份），SQL Server 要求在除最后的 RESTORE 语句外的所有其他语句上使用WITH NORECOVERY 选项。

□ RECOVERY：指示还原操作回滚任何未提交的事务。在恢复进程后即可随时使用数据库。如果安排了后续RESTORE 操作（RESTORE LOG或从差异数据库备份RESTORE DATABASE），则应改为指定的NORECOVERY或STANDBY。

□ STANDBY = undo_file_name：指定撤销文件名以便可以取消恢复效果。撤销文件的大小取决于因未提交的事务所导致的撤销操作量。如果 NORECOVERY、RECOVERY和STANDBY均未指定，则默认为

RECOVERY。STANDBY允许将数据库设定为在事务日志还原期间只能读取，并且可用于备用服务器情形，或用于需要在日志还原操作之间检查数据库的特殊恢复情形。

□ KEEP_REPLICATION：指示还原操作在将发布的数据库还原到创建它的服务器以外的服务器上时保留复制设置。当设置复制与日志传送一同使用时，需使用KEEP_REPLI CATION。这样，当在备用服务器上还原数据库或日志备份并且恢复数据库时，可防止删除复制设置。还原备份时若指定了该选项，则不能选择NORECOVERY选项。

□ NOUNLOAD：指定不在 RESTORE 后从磁带机中自动卸载磁带。设置始终为NOUNLOAD，直到指定UNLOAD为止。该选项只用于磁带设备。如果对RESTORE使用非磁带设备，将忽略该选项。

□ NOREWIND：指定 SQL Server 在备份操作完成后使磁带保持打开。磁带保持打开将防止其他过程访问磁带。直到颁发REWIND或UNLOAD语句，或直到服务器关闭时才释放该磁带。通过查询master数据库中的sysopentapes表可查找当前打开的一系列磁带。NOREWIND即NOUNLOAD。该选项只用于磁带设备。如果对RESTORE使用非磁带设备，将忽略该选项。

□ REWIND：指定SQL Server 将释放磁带和倒带。如果NOREWIND和REWIND 均未指定，则默认设置为REWIND。该选项只用于磁带设备，如果对RESTORE使用非磁带设备，将忽略该选项。

□ UNLOAD：指定在还原完成后自动倒带并卸载磁带。启动新用户会话时其默认设置为UNLOAD。设置始终为UNLOAD，直到指定NOUNLOAD为止。该选项只用于磁带设备，如果对RESTORE使用非磁带设备，将忽略该选项。

□ REPLACE：指定即使存在另一个具有相同名称的数据库，SQL Server 也应该创建指定的数据库及其相关文件。在这种情况下将删除

现有的数据库。如果没有指定REPLACE选项，则将进行安全检查以防止意外重写其他数据库。

□ RESTART：指定SQL Server 应重新启动被中断的还原操作。RESTART从中断点重新启动还原操作。

□ STATS [= percentage]：每当另一个percentage 结束时显示一条消息，并用于测量进度。如果省略percentage，则SQL Server 每完成10 个百分比显示一条消息。

实现过程

（1）在项目中创建类ResumeFrame，该类继承自JFrame类，实现窗体类。

（2）向窗体中添加控件。实现窗体布局，该窗体中的主要控件及说明如表4.4所示。

表4.4 窗体中的主要控件及说明

控 件 类 型	控 件 命 名	控 件 用 途
JTextField	fileNameTextField	为用户提供显示要恢复的数据库文件
JComboBox	dataBaseComboBox	显示要恢复的数据库
JButton	browseButton	显示“浏览”的按钮控件
	resumeButton	显示“恢复”的按钮控件

（3）创建类Resume，用于定义数据恢复方法getBak()。该方法包含两个String类型的参数，分别用于定义要恢复的数据库与数据库的备份文件。具体代码如下：

```
    public void getBak(String databaseName, String
databasePath) {
        Connection
  con=Con();                                          //获
  取数据库连接
        Statement st;
        try {
```

```
        st=
con.createStatement();
          //实例化Statement对象
        st.executeUpdate("restoredatabase "+databaseName+ "
fromdisk='"+databasePath+ "'");                          //
指定数据库备份SQL语句
        con.close();
              //关闭连接
      } catch (SQLException e) {
        e.printStackTrace();
      }
}
```

举一反三

根据本实例，读者可以实现以下功能。

将数据库恢复至备份时状态。

将数据恢复到系统中。

实例163 MySQL数据备份

这是一个可以启发思维的实例

实例位置：光盘\mingrisoft\04\Ex04_163

实例说明

在实际开发中，MySQL数据库的应用非常广泛，因此掌握MySQL数据库的备份技术非常重要。备份MySQL 数据库的方法很多，可以通过MySQL Tools 或者phpMy Admin 管理工具进行备份，也可以通过命令进行备份。本实例使用的是MySQL DUMP 命令备份数据库，运行本实例，结果如图4.31所示。

图4.31 实例运行结果

技术要点

本实例主要应用 java.lang 软件包中的 Runtime、Process 和 StringBuffer 类。首先通过getRuntime()方法获取与当前 Java 应用程序相关的 Runtime 对象，然后应用 exec()方法执行MySQL DUMP 命令。接着应用Process 类中的getInputStream()方法获取子进程的输入流，最后应用StringBuffer类中的append()方法将流中的数据追加到指定的字符序列中，完成数据的备份。下面对使用的方法进行详细讲解。

（1）getRuntime()方法

语法如下：

```
public static Runtime getRuntime()
```

该方法返回与当前Java 应用程序相关的Runtime 对象。Runtime 类的大多数方法是实例方法，并且必须根据当前的运行对象对其进行调用。

（2）exec()方法

语法如下：

```
public Process exec(String command) throws IOException
```

该方法在单独的进程中执行指定的字符串命令。返回一个新的 Process 对象，用于管理子进程。

参数说明

command：一条指定的系统命令。

抛出异常，则：

● SecurityException：如果安全管理器存在，并且其checkExec方法不允许创建子进程。

● IOException：如果发生I/O 错误。

● NullPointerException：如果command 为null。

● IllegalArgumentException：如果command 为空。

（3）getInputStream()方法

语法如下：

public abstract InputStream getInputStream()

该方法用于获取子进程的输入流。输入流由该Process对象表示的进程的标准输出流获得。返回连接到子进程正常输出的输入流。

（4）append()方法

语法如下：

public StringBuffer append(String str)

该方法将指定的字符串追加到此字符序列。按顺序追加String变量中的字符，使此序列增加该变量的长度。如果str为null，则追加4个字符null。返回值为此对象的一个引用。

参数说明

str：一个指定的字符串。

此字符序列的长度在执行append()方法前为n。如果k小于n，则新字符序列中索引k处的字符等于原序列中索引k处的字符；否则它等于参数str中索引k-n处的字符。

MySQ LDUMP 命令的语法如下：

mysqldump -uUser -pPass DataBase > Path

其中，User是用户名，Pass是密码，DataBase是数据库名，Path是数据库备份存储的位置。

实现过程

（1）在项目中创建类BackFrame，该类继承自JFrame类，实现窗体类。

（2）向窗体中添加控件。实现窗体布局，该窗体中的主要控件及说明如表4.5所示。

表4.5 窗体中的主要控件及说明

控 件 类 型	控 件 命 名	控 件 用 途
JTextField	nameTextField	显示保存备份后的文件名称
JComboBox	dataBaseComboBox	显示要进行备份的数据库名称
JButton	backButton	显示“备份”的按钮控件

（3）创建类MySQLConn，在该类中定义备份MySQL数据库的方法mysqldump()，该方法包含两个String类型的参数，分别用于定义要进行备份的数据库与备份数据库的文件名称。具体代码如下：

```
public boolean mysqldump(String database, String path)
{        //备份数据库
  try {
    Process p = Runtime.getRuntime().exec("cmd.exe /c
  mysqldump -uroot -p111 "+ database +">"+ path
  +"");        //定义进行数据备份的语句
    StringBuffer out1 = new StringBuffer();        //定义
  字符串缓冲对象
    byte[] b = new byte[1024];        //定义字节数组
    for (int i; ((i = p.getInputStream().read(b)) !=-1);)
  {        //将数据写入到指定文件中
    out1.append(new String(b, 0, i));        //向流中追
  加数据
```

```
        }
      } catch (IOException e) {
        e.printStackTrace();
        return false;
      }
      return true;
    }
```

举一反三

根据本实例，读者可以实现以下功能。

通过phpMy Admin管理工具进行备份。

通过MySQL Tools管理工具进行备份。

通过命令进行备份。

实例164 MySQL数据恢复

本实例可以提高工作效率

实例位置：光盘\mingrisoft\04\Ex04_164

实例说明

对于大型应用系统来说，具有数据恢复功能非常重要。数据恢复可以在数据遭到破坏时将备份的数据恢复到数据库系统中，以保证系统的正常运行，避免数据丢失带来的损失。本实例向大家介绍 MySQL 数据恢复的方法，与 MySQL 数据备份方法类似，都是使用命令实现。实例运行效果如图4.32所示。

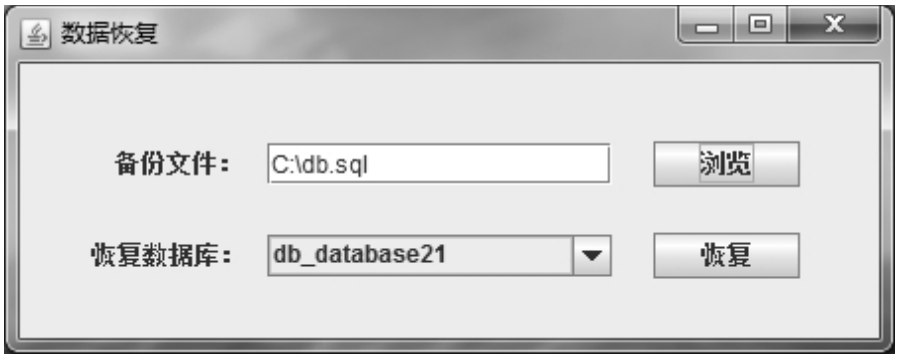


图4.32 实例运行效果

技术要点

本实例中介绍的 MySQL 数据库的恢复方法与 MySQL 数据库备份的方法类似，有关具体类和方法的详细讲解可以参考实例164，这里不再赘述。

唯一的不同之处是所使用的命令，MySQL数据库的恢复使用的是MySQL命令，其语法格式如下：

```
mysql  -uUser  -pPass DataBase <Path
```

其中User指定数据库用户名，Pass指定密码，DataBase为指定备份的数据库，Path是备份文件存储的位置。

实现过程

（1）在项目中创建类BackFrame，该类继承自JFrame类，实现窗体类。

（2）向窗体中添加控件。实现窗体布局，该窗体中的主要控件及说明如表4.6所示。

表4.6 窗体中的主要控件及说明

控 件 类 型	控 件 命 名	控 件 用 途
JTextField	fileTextField	显示要恢复数据的文件
JComboBox	databaseComboBox	显示要进行恢复的数据库名称
JButton	browseButton	显示“浏览”的按钮控件
	resumeButton	显示“恢复”的按钮控件

（3）创建类ResumeUtil，在该类中定义恢复数据库的方法mysqlresume()，该方法包含两个String类型的参数，分别用于定义要

恢复的数据库与备份数据文件的保存地址。具体代码如下：

```
    public boolean mysqlresume(String database, String path)
{       //恢复数据库
      try {
        Process p = Runtime.getRuntime().exec("cmd.exe /c
    mysql -uroot -p111 "+ database +"<"+ path +"");        //
    执行恢复语句
        StringBuffer out1 = new StringBuffer();        //定义
    字符串缓冲对象
        byte[] b = new byte[1024];        //定义字节数组
        for (int i; ((i = p.getInputStream().read(b)) !=-1);)
    {       //将数据写入到指定文件中
          out1.append(new String(b, 0, i));        //向流中追
      加数据
        }
      } catch (IOException e) {
        e.printStackTrace();
        return false;
      }
      return true;
    }
```

举一反三

根据本实例，读者可以实现以下功能。

将数据恢复到系统中。

将数据库恢复至备份时状态。

实例165 动态附加数据库

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\04\Ex04_165

实例说明

在项目开发完成时，需要为用户提供安装程序，即使最简单的安装也需要提供数据库导入功能。本实例实现的是通过 Java 应用程序，将 SQL Server 2000 数据库文件附加到本地SQL Server 服务器中。运行本实例，用户在“数据库名称”文本框中输入附加数据库后的数据库名，并为用户提供了可浏览本地数据库文件与数据库日志文件。实例运行效果如图4.33所示。

图4.33 实例运行效果

技术要点

本实例主要应用SQL Server 中的系统存储过程sp_attach_db 完成。

sp_attach_db 用于将数据库附加到SQL Server 服务器中。语法格式如下：

sp_attach_db [@dbname =] 'dbname', [@filename1 =] 'filename_n' [,...16]

参数说明

● [@dbname =] 'dbname'：要附加到服务器的数据库的名称。该名称必须是唯一的。dbname的数据类型为sysname，默认值为null。

● [@filename1 =] 'filename_n'：数据库文件的物理名称，包括路径。filename_n 的数据类型为 nvarchar(260)，默认值为 null，最多可以指定 16 个文件名。参数名称以@filename1开始，递

增到@filename16。文件名列表必须包括主文件，因为主文件包含指向数据库中其他文件的系统表。该列表还必须包括数据库分离后所有被移动的文件。

实现过程

（1）在项目中创建类SubjoinFrame，该类继承自JFrame类，实现窗体类。

（2）向窗体中添加控件。实现窗体布局，该窗体中的主要控件及说明如表4.7所示。

表4.7 窗体中的主要控件及说明

控件类型	控件命名	控件用途
JTextField	nameTextField	显示数据库名称文本框控件
	dataTextField	显示数据库文件地址文本框控件
	logTextField	显示数据库日志文件文本框控件
JButton	brownButton	显示浏览数据库文件的按钮控件
	logBrowButton	显示浏览日志文件的按钮控件
	subjoinButton	显示附加数据库的按钮控件

（3）创建类SubjoinDate，在该类中定义附加数据库的方法executeUpdate()，该方法包含3个String类型的参数，分别用于定义附加数据库的名称、数据库文件地址与数据库日志文件地址。具体代码如下：

```
public boolean executeUpdate(String dataName,String
mPath,String lPath) {
    if (con == null) {
      Connection();
          //数据库连接
    }
    try {
      stmt = con.createStatement();
```

```
    int iCount = stmt.executeUpdate("EXEC sp_attach_db
@dbname = '"+dataName+"', @filename1='"+mPath+"',
@filename2 = '"+lPath+"'");        //执行数据库附加
  } catch (SQLException e) {
    System.out.println(e.getMessage());
    return false;
  }
  closeConnection();
          //调用关闭数据库连接方法
return true;
}
```

举一反三

根据本实例，读者可以实现以下功能。

动态附加数据库。

分离数据库。

实例166 生成SQL数据库脚本

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\04\Ex04_166

实例说明

SQL 脚本包含用于创建数据库及其对象的语句描述。可以从现有数据库中的对象生成脚本，然后通过运行该数据库的脚本将这些对象添加到其他数据库。实际上，这样做是重新创建了整个数据库结构，以及所有的单个数据库对象。

技术要点

使用SQL Server 2000数据库，可以通过生成一个或多个SQL脚本，编写现有的数据库结构（架构）文档。可使用SQLServer企业管理器、SQL查询分析器或任何文本编辑器查看 SQL脚本。

作为SQL脚本生成的架构有许多用途，其中包括：

（1）维护备份脚本，该脚本将允许用户重新创建所有用户、组、登录和权限。

（2）创建和更新数据库开发代码。

（3）从现有的架构创建一个测试或开发环境。

SQL脚本可以生成表4.8所示的对象架构，并保存为脚本。

表4.8 脚本对象

对　　象	说　　明	对　　象	说　　明
表	用户定义的数据类型	存储过程	登录
索引	触发器	默认值	规则
视图	用户、组和角色	表键/声明引用完整性（DRI）	对象级权限

所生成的对象架构可以保存在一个SQL脚本文件中，或者保存在多个文件中，其中每个文件都只包含一个对象的架构，还可以将所生成的单个对象（或一组对象）的架构保存到一个或多个 SQL 脚本文件中。可以生成的SQL 脚本文件示例包括：

（1）保存到单个SQL脚本文件中的整个数据库。

（2）保存到SQL脚本文件中的数据库中的表只有表的架构。

（3）保存到一个SQL脚本文件表和索引架构，保存到其他SQL脚本文件中的存储过程，以及保存到又一个SQL脚本文件中的默认设置和规则。

实现过程

（1）下面以db_database22数据库为例介绍生成SQL数据库脚本的步骤。打开企业管理器，展开“数据库”节点。

（2）选中欲生成数据库脚本的数据库，单击鼠标右键，在弹出的快捷菜单中选择“所有任务”→“生成SQL脚本”命令，如图4.34所

示。

（3）打开“生成SQL脚本”对话框，打开“常规”选项卡，大部分的选项处于不可编辑状态。单击“全部显示”按钮，此时编写脚本的对象的全部功能处于可用状态，然后选中“编写全部对象脚本”复选框，所有的脚本对象处于选中状态，设置效果如图4.35所示。

图4.34 选择“生成SQL 脚本”命令

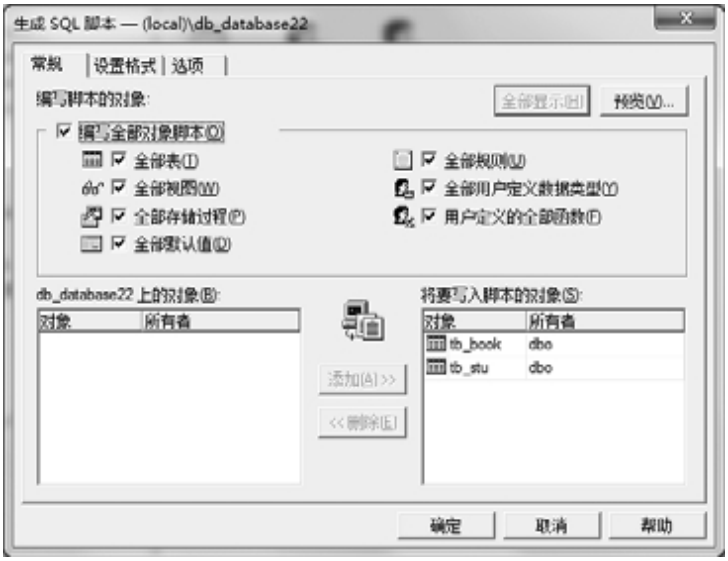


图4.35 设置脚本选项

（4）选择“选项”选项卡，为SQL脚本设置选项。选中“编写数据库脚本”复选框，如图4.36所示。

（5）单击“确定”按钮，弹出“另存为”对话框，选择欲生成脚本的存储路径，在“文件名”文本框中输入脚本文件的名称，单击“保存”按钮，即可成功编写SQL数据库脚本，如图4.37所示。

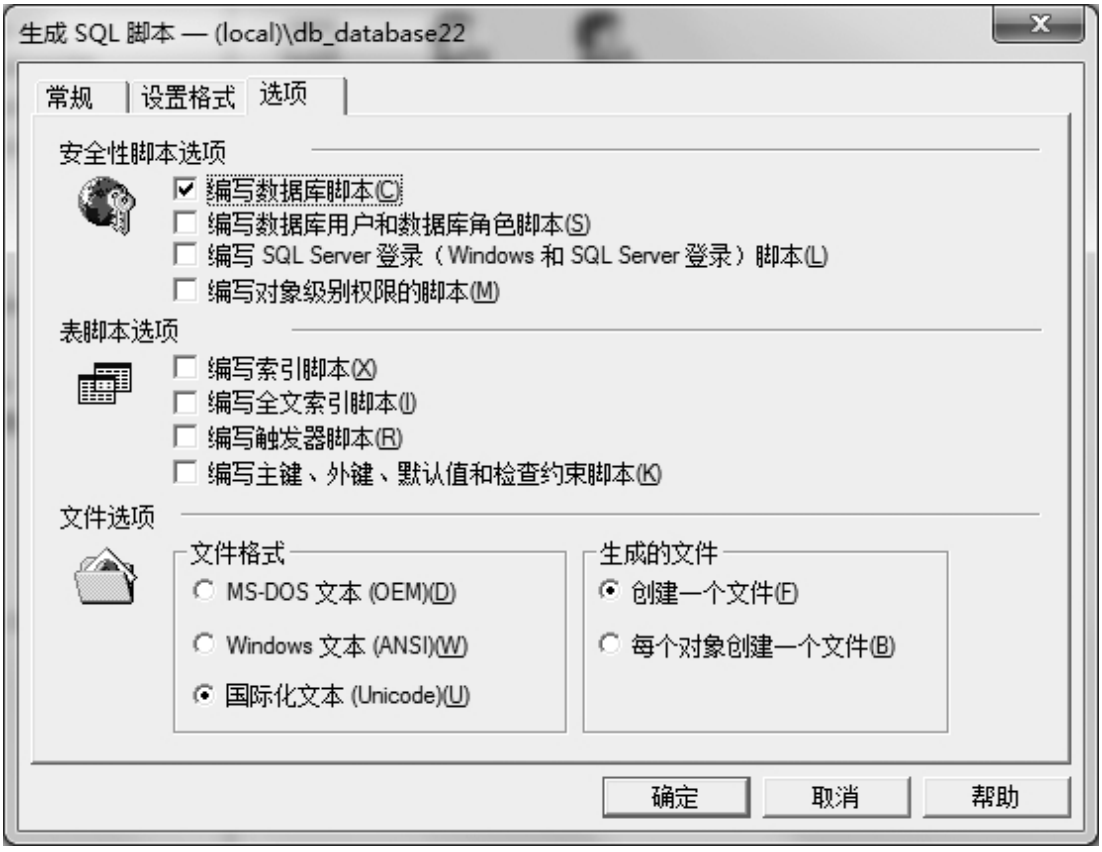


图4.36 选中“编写数据库脚本”复选框

图4.37 “另存为”对话框

举一反三

根据本实例，读者可以实现以下功能。

生成SQL Server 数据库脚本。

重新创建数据库结构。

4.4 数据增加、更新与删除操作

实例167 将员工信息添加到数据表

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\04\Ex04_167

实例说明

数据录入对每个应用程序来说都是必不可少的。本实例实现的是单条数据的录入。在实现数据录入时，可以对录入进行验证，来保证程序的完整性。本实例实现当用户填写完以“*”标注的信息后，单击“添加”按钮，系统会将用户添加的信息写入数据库中。实例运行效果如图4.38所示。

图4.38 实例运行效果

技术要点

本实例在实现数据录入时，使用的是INSERT插入语句。INSERT语句的语法格式如下：

INSERT INTO 表名[(字段名1,字段名2…)]

VALUES(属性值1,属性值2, …)

例如，向数据表tb_emp（包含字段id,name,sex,department）中插入数据。代码如下：

insert into tb_emp values(2,'lili', '女', '销售部');

实现过程

（1）在项目中创建类InsertEmpFrame，该类继承自JFrame类，实现窗体类。

（2）向窗体中添加控件，实现窗体布局，该窗体中的主要控件及说明如表4.9所示。

表4.9 窗体中的主要控件及说明

控件类型	控件命名	控件用途
JTextField	nameTextField	供用户添加员工“姓名”的文本框控件
	ageTextField	供用户添加“年龄”的文本框控件
	deptTextField	供用户添加“部门”信息的文本框控件
	phoneTextField	供用户添加“电话”信息的文本框控件

续表

控件类型	控件命名	控件用途
JComboBox	sexComboBox	为用户提供“性别”的下拉列表控件
JTextArea	remakeTextArea	为用户提供备注信息的下拉列表控件
JButton	insertButton	添加数据的单击按钮
	closeButton	显示“关闭”的按钮

（3）创建类JdbcUtil，在该类中定义insertEmp()方法，用于实现添加数据，该方法包含一个与员工表tb_emp对应的JavaBean类，Emp为参数。具体代码如下：

public void insertEmp(Emp emp){

```
  conn=getConn();
      //获取数据库连接
  try {
    PreparedStatement statement =
conn.prepareStatement("insert into tb_emp
values(?,?,?,?,?,?)");
    //定义插入数据库的预处理语句
    statement.setString(1,
emp.getName());                                    //设置预处理
语句的参数值
    statement.setString(2, emp.getSex());
    statement.setInt(3, emp.getAge());
    statement.setString(4, emp.getDept());
    statement.setString(5, emp.getPhone());
    statement.setString(6, emp.getRemark());
    statement.executeUpdate();
            //执行预处理语句
  } catch (SQLException e) {
    e.printStackTrace();
  }
}
```

举一反三

根据本实例，读者可以实现以下功能。

将数据添加到数据表中。

查看数据表中的数据。

实例168 添加数据时使用数据验证

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\04\Ex04_168

实例说明

在将数据添加到数据库时，为了达到数据的完整性，在添加数据时实现数据验证是很常见的功能。本实例实现的是在添加员工信息时，使用数据验证，可以验证要添加的员工信息是否存在于数据库中，如果不存在，则允许添加，如果存在，则不允许执行添加操作。在“年龄”文本框中也添加了数据验证，只允许用户输入数字。实例运行效果如图4.39所示。

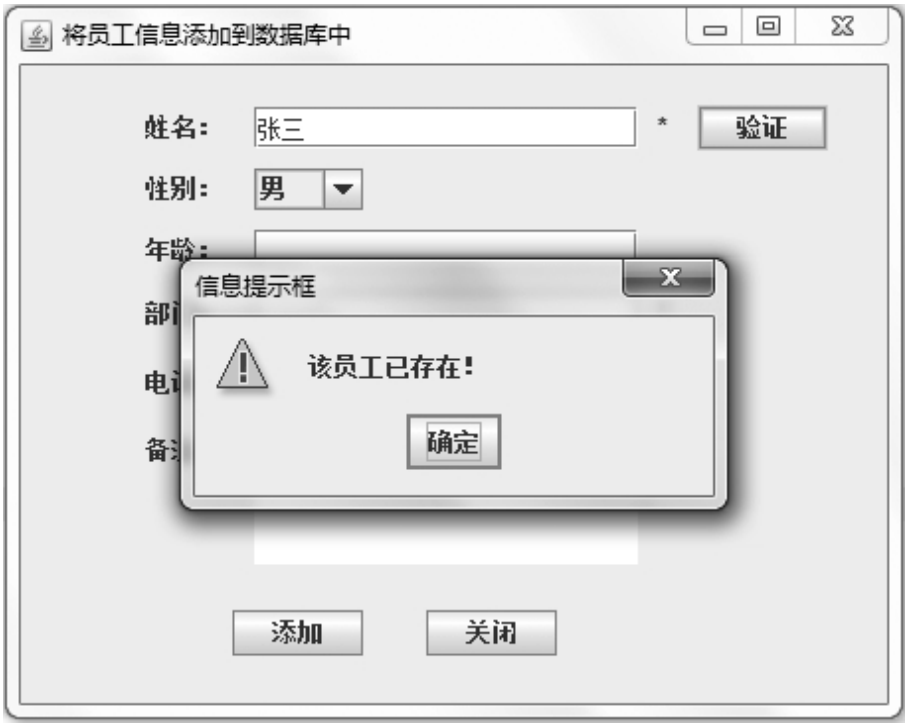


图4.39 实例运行效果

技术要点

本实例是通过用户输入的员工姓名来查询员工表，如果员工表中包含相应的信息，则给出相应的提示。通过为文本框添加键盘事件，来限制在“年龄”文本框中只能输入数字。

实现过程

（1）在项目中创建类InsertEmpFrame，该类继承自JFrame类，实现窗体类。

（2）向窗体中添加控件，实现窗体布局，该窗体中的主要控件及说明如表4.10所示。

表4.10 窗体中的主要控件及说明

控件类型	控件命名	控件用途
JTextField	nameTextField	供用户添加员工“姓名”的文本框控件
	ageTextField	供用户添加“年龄”的文本框控件
	deptTextField	供用户添加“部门”信息的文本框控件
	phoneTextField	供用户添加“电话”信息的文本框控件
JComboBox	sexComboBox	为用户提供“性别”下拉列表控件
JTextArea	remakeTextArea	为用户提供“备注”信息的下拉列表控件
JButton	insertButton	添加数据的单击按钮
	closeButton	显示“关闭”的按钮
	validateButton	显示“验证”的单击按钮

（3）创建类JdbcUtil，在该类中定义insertEmp()方法，用于实现添加数据，该方法包含一个与员工表tb_emp对应的JavaBean类，Emp为参数。具体代码如下：

```
public void insertEmp(Emp emp){
  conn=getConn();
  //获取数据库连接
  try {
    PreparedStatement statement =
conn.prepareStatement("insert into tb_emp
values(?,?,?,?,?,?)");          //定义插入数据库的预处理语句
    statement.setString(1,
emp.getName());                              //设置预处理语句的参数值
    statement.setString(2, emp.getSex());
```

```
            statement.setInt(3, emp.getAge());
            statement.setString(4, emp.getDept());
            statement.setString(5, emp.getPhone());
            statement.setString(6, emp.getRemark());
            statement.executeUpdate();
                //执行预处理语句
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
```

（4）当用户输入员工姓名后，单击“验证”按钮，系统会调用JdbcUtil类的selectEmpUseName()方法检验用户输入的员工是否存在于员工表中。selectEmpUseName()方法的具体代码如下：

```
    public int selectEmpUseName(String name){
        conn=getConn();
        //获取数据库连接
        Statement statment;
        int
    id=0;                                                    //定
    义保存返回值的int对象
        try {
            statment=
    conn.createStatement();                                  //获取
    Statement对象
            String sql= "select id from tb_emp where name=
    '"+name+"'";          //定义查询SQL语句
```

```
        ResultSet rest=
    statment.executeQuery(sql);                    //执行查
    询语句获取查询结果集
        while(rest.next())
    {                                              //循环遍历查询结
    果集
          id=
      rest.getInt(1);                                        //
      获取查询结果
        }
      } catch (Exception e) {
        e.printStackTrace();
      }
      return
id;                                                //返回
查询结果
    }
```

举一反三

根据本实例，读者可以实现以下功能。

验证电话号码。

验证用户姓名。

实例169 插入用户登录日志信息

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\04\Ex04_169

实例说明

几乎所有的应用程序都提供了一个登录窗体。该窗体用于验证用户是否有权限登录此程序。在用户登录的同时也可以将用户的登录信息记录到数据库中。本实例实现将用户登录的用户名、密码、当前登录时间，写入数据库中。实例运行效果如图4.40所示。

图4.40 实例运行效果

技术要点

本实例实现将用户登录的时间写入数据库中，并对日期进行了格式化。SimpleDateFormat类的format()方法可实现将日期进行格式化。

SimpleDateFormat 类是一个以与语言环境有关的方式来格式化和解析日期的具体类。该类的常用构造方法介绍如下。

□ 用默认的模式和默认语言环境的日期格式符号构造SimpleDateFormat类

SimpleDateFormat()

□ 用给定的模式和默认语言环境的日期格式符号构造SimpleDateFormat类

SimpleDateFormat(String pattern)

参数说明

pattern：描述日期和时间格式的模式。

该类的format()方法可将给定的Date格式化为日期/时间字符串。以String返回格式化后的字符串。该方法的语法格式如下：

format(Date date)

参数说明

date：要格式化为时间字符串的时间值。

实现过程

（1）在项目中创建类InsertInfoFrame，该类继承自JFrame类，实现窗体类。

（2）在该窗体中添加文本框、按钮的控件，实现窗体布局，该窗体中的主要控件及说明如表4.11所示。

表4.11 窗体中的主要控件及说明

控 件 类 型	控 件 命 名	控 件 用 途
JTextField	nameTextField	供用户添加登录“用户名”的文本框控件
	passWordTextField	供用户添加“密码”的文本框控件
JButton	enterButton	显示“登录”的按钮控件
	closeButton	显示“关闭”的按钮控件

（3）在项目中定义类 InsertInfo ，在该类中定义将用户登录信息插入到数据库的方法insertUser()。该方法的具体代码如下：

```
public void insertUser(User user) {
  conn=getConn();
            //获取数据库连接
try {
    //定义插入数据库的预处理语句
    PreparedStatement statement =
conn.prepareStatement("insert into tb_user
values(?,?,?)");
```

```
        statement.setString(1,
    user.getUserName());                                    //设
    置预处理语句参数
        statement.setString(2, user.getPassWord());
        //根据指定格式定义SimpleDateFormat对象
        SimpleDateFormat date_time = new
    SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
        String datetime=date_time.format(new
    Date());                                    //对当前日期进行格式
    化
        statement.setString(3, datetime);
        statement.executeUpdate();
                        //执行预处理语句
      } catch (SQLException e) {
        e.printStackTrace();
      }
    }
    =ContentAlignment.MiddleCenter;          //光标离开时改变
button6图片位置
    }
```

举一反三

根据本实例，读者可以实现以下功能。

将用户的登录信息插入的数据库。

从数据库中读出用户登录信息。

实例170 生成有规律的编号

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\04\Ex04_170

实例说明

在程序开发中，经常会遇到这样的情况，数据表中的主键是唯一的，如果要求用户手工输入是很不方便的，也容易产生错误，解决这个问题的最好方法就是让系统自动生成唯一的 ID号。生成 ID 号可以有多种形式，本实例为创建一个商品信息添加窗体，在该窗体中用户可以添加商品信息，运行结果如图4.41所示。当用户单击“查看”按钮时，系统会给出所有商品信息，如图4.42所示。

技术要点

商品编号由字母“CG”、系统日期和5位数字组成。首先判断采购信息表中的采购编号是否为空，如果为空，则采购编号等于字母“CG”+系统日期+“00001”；如果不为空，则先查找数据表中最大的采购编号，此时采购编号等于字母“CG”+系统日期+5位数字编码加1。在实现编码加1时，数字前面的0被忽略，实现数字前加0的格式，可以采用字符串的format()方法。

format()方法用于输出指定格式的字符串，其语法格式如下：

```
String.format("%05d", ID + 1)
```

图4.41 添加商品信息

图4.42 查看所有商品

参数说明

- ID：是数字格式的变量，将其加1 实现编号增值。
- "%05d"：设定数字的格式的位数是5 位，如果不足5 位，则前面补0。

实现过程

（1）在项目中创建添加商品信息窗体 InsertWareFrame 和查看商品信息窗体SelectWareFrame，查看商品信息窗体比较简单，在该窗体中有标签与表格控件。添加商品信息窗体中的主要控件及说明如表4.12所示。

表4.12 添加商品信息窗体中的主要控件及说明

控 件 类 型	控 件 命 名	控 件 用 途
JTextField	nameTextField	商品“名称”文本框控件
	specTextField	商品“规格”文本框控件
	casingTextField	商品“包装”文本框控件
	unitTextField	商品“单位”文本框控件
	amountTextField	商品“数量”文本框控件
JButton	insertButton	“添加”按钮控件
	watchButton	“查看”按钮控件

（2）在项目中创建工具类WareUtil，在该类中定义向商品中添加数据的方法insertWare()，该方法以与数据库对应的JavaBean为参数。具体代码如下：

```
public void insertWare(Ware ware) {
  conn=getConn();
//获取数据库连接
    try {
    //定义插入数据库的预处理语句
    PreparedStatement statement =
conn.prepareStatement("insert into tb_ware
values(?,?,?,?,?,?,?)");
    statement.setString(1,ware.getSID()
);                          //设置预处理语句的参数值
    statement.setString(2,ware.getsName());
    statement.setString(3, ware.getSpec());
    statement.setString(4,ware.getCasing());
```

```
            statement.setString(5,ware.getUnit() );
            statement.setString(6, ware.getsDate());
            statement.setInt(7, ware.getAmout());
            statement.executeUpdate();
            //执行预处理语句
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
```

（3）当用户单击商品添加表中的“添加”按钮后，系统会将用户添加的信息保存在数据库中。“添加”按钮的单击事件中的代码如下：

```
    protected void
do_insertButton_actionPerformed(ActionEvent arg0) {
        String name = nameTextField.getText();          //获取用户添加的商品名称
        …//省略了获取其他商品信息代码
        int count =
    Integer.parseInt(amountTextField.getText());          //获取用户添加的商品数量
        int ID = 0;
        String sDate = WareUtil.getDateTime();          //调用获取系统时间方法
        List list = util.selectWare();          //获取商品表中全部的商品
        String sid ="";
```

```
    for(int i = 0;i<list.size();i++){          //循环遍历查询
结果集
      Ware ware = (Ware)list.get(i);          //获取商品
      sid = ware.getSID();          //获取商品编号
    }
    if(list.size()==0){          //如果商品集合中为空
      sid ="CS"+sDate.replace("-", "")+"00001";          //定
  义商品编号
    }
    else{
    //如果商品集合不为空
      sid = sid.trim();
      ID =
  Integer.parseInt(sid.substring(sid.length()-5));          /
  /截取商品编号中的后5位
      sid = sid.substring(0, sid.length()-5) +
  String.format("%05d", ID + 1) //商品编号
    }
    Ware ware=new
Ware();                                          //定义与商品表
对应的JavaBean对象
    ware.setSID(sid);
      //设置JavaBean编号
    …//省略了设置其他属性代码
    util.insertWare(ware);
          //添加商品信息
```

JOptionPane.showMessageDialog(getContentPane(), "数据添加成功！","信息提示框", JOptionPane.CANCEL_OPTION);

}

举一反三

根据本实例，读者可以开发以下程序。

生成有规律的编号。

根据编号查询相关信息。

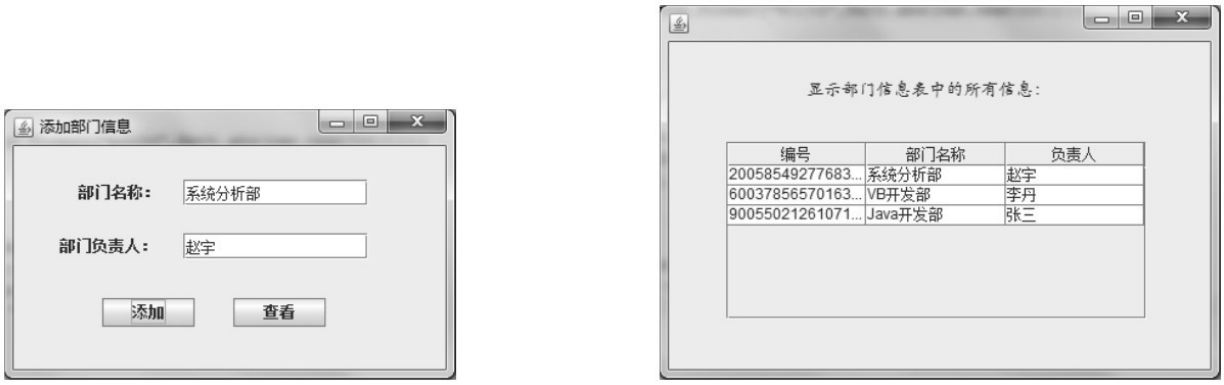
实例171 生成无规律的编号

这是一个可以启发思维的实例

实例位置：光盘\mingrisoft\04\Ex04_171

实例说明

在实例170中为读者介绍了生成有规律的编号，当然也可以生成无规律的编号，这种无规则的编号，不仅可以用在一张表中作主键，也可以用在其他方面。实例运行效果如图4.43所示。



(a) (b)

图4.43 实例运行效果

技术要点

用户编号由一个3位的整型随机数加上10位的整型随机数和一个19位的长整型随机数的字符串连接而成，总长度为32位。由于使用该方

法可能出现编号重复的情况，所以实例中对插入编号的唯一性进行了判断，如果编号重复将重新生成。

实现过程

（1）在项目中创建添加部门信息的窗体类InsertDeptFrame，在该类中添加标签、文本框与按钮控件，实现窗体布局，并创建显示部门信息的SelectDeptFrame类，在该类中添加标签与表格控件，实现窗体布局。

（2）在项目中定义工具类 DeptUtil，在该类中定义部门信息表方法，该类中的方法不是本实例的重点，用户可参考光盘中的源程序，这里不再赘述。

（3）当用户单击添加部门信息窗体中的“添加”按钮时，系统会将用户添加的信息保存到数据库中。“添加”按钮的单击事件中的代码如下：

```
    protected void
do_insertButton_actionPerformed(ActionEvent arg0) {
      String
  name=deptNameTextField.getText();                          //
  获取用户添加的部门名称
      String
  person=personTextField.getText();
  //获取用户添加的部门负责人
      Random ran=new
  Random(System.currentTimeMillis());                  //根据当
  前时间的毫秒数创建随机数流
    StringBuilder idb=new
StringBuilder();                                       //创建字符串
缓冲对象
```

```
    idb.append(String.format("%019d",Math.abs(ran.nextLong(
))))+String.format("%03d",(int)(Math.random()*100%9)+1));
    idb=idb.reverse();
        //对字符串缓冲对象取反
    String id=idb+""+
String.format("%010d",Math.abs(ran.nextInt()));
    Dept
dept=newDept();                                          //创
建与数据表对应的JavaBean对象
    dept.setDid(id);
        //设置对象属性
    dept.setdName(name);
    dept.setPriName(person);
    DeptUtil deptUtil=new
DeptUtil();                                  //创建工具类对象
    deptUtil.insertDept(dept);
          //调用添加方法
    JOptionPane.showMessageDialog(getContentPane(),"数据添
加成功！", "信息提示框", JOptionPane.WARNING_MESSAGE);
  }
```

举一反三

根据本实例，读者可以实现以下功能。

生成不重复的无规律编号。

用生成的编号当主键查询数据。

实例172 在插入数据时过滤掉危险字符

这是一个可以美化界面的实例

实例位置：光盘\mingrisoft\04\Ex04_172

实例说明

程序开发中，数据的完整性与安全性是必须要考虑的问题，而对危险字符的过滤不仅可以应用在应用程序中，还可以应用在Web程序中。本实例以在插入数据时过滤掉危险字符为例，为大家介绍对危险字符的过滤技术。本实例实现的是向图书销售表中插入数据，插入时将图书名称中的危险字符过滤掉。实例运行效果如图4.44所示。

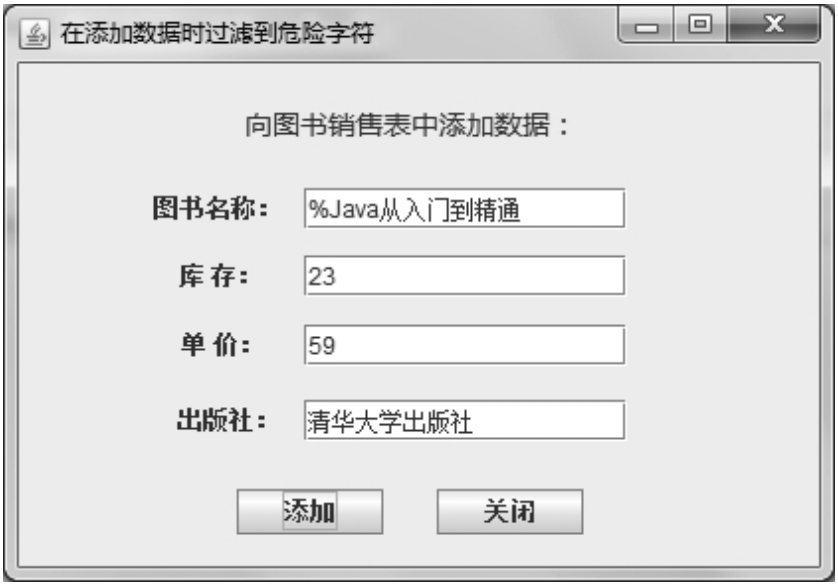


图4.44 实例运行效果

技术要点

本实例主要应用String类中的replaceAll()方法，其语法格式如下：

replaceAll(String source,String replace)

功能：替换字符。

参数说明

- source：要替换掉的字符串。
- replace：用来替代的字符串。

实现过程

（1）在项目中创建窗体类 InsertBookFrame，该类继承自 JFrame 类，实现窗体类。向该窗体中添加标签、文本框与按钮控件，实现窗体布局。

（2）创建字符串处理类 DoString，在该类中定义过滤掉危险字符的方法。该类中的主要代码如下：

```
private String getstr;
private String checkstr;
public DoString(){}
public void setGetstr(String getstr){
  this.getstr=getstr;
  dostring();
}
public String getGetstr(){
  return this.getstr;
}
public String getCheckstr(){
  return this.checkstr;
}
public void dostring(){
  this.checkstr=this.getstr;
  this.checkstr=this.checkstr.replaceAll("&","&amp;");
                                  //替换字符处理
  this.checkstr=this.checkstr.replaceAll(";","");
  this.checkstr=this.checkstr.replaceAll("'","");
  this.checkstr=this.checkstr.replaceAll("<","&lt;");
  this.checkstr=this.checkstr.replaceAll(">","&gt;");
  this.checkstr=this.checkstr.replaceAll("--","");
```

```
        this.checkstr=this.checkstr.replaceAll("\"\"","&quot;")
;
        this.checkstr=this.checkstr.replaceAll("/","");
        this.checkstr=this.checkstr.replaceAll("%","");
    }
```

举一反三

根据本实例，读者可以实现以下功能。

向数据表中插入数据。

过滤数据表中的敏感词。

实例173 将用户选择的爱好以字符串形式保存到数据库

这是一个可以美化界面的实例

实例位置：光盘\mingrisoft\04\Ex04_173

实例说明

在用户注册模块中，要求用户添加个人爱好是十分常见的，通常情况下，系统会以复选框的形式来供用户选择。本实例实现将用户选择的爱好信息，以字符串的形式保存到数据库中。实例运行效果如图4.45所示。

图4.45 实例运行效果

技术要点

本实例中使用了字符串缓冲区对象 StringBuffer，当用户选择了一项爱好时，系统会通过append()方法追加内容。在向数据库中添加数据时，只需要调用该类的toString()方法即可。

实现过程

（1）在项目中创建窗体类ToStringFrame()，该类继承自JFrame类，实现窗体类。在该窗体中添加标签、文本框、复选框与按钮控件，实现窗体布局。

（2）在各个复选框控件中添加ItemEvent，当用户选择某项爱好时，会将字符串缓冲区对象做追加处理。该事件中的代码如下：

```
protected void do_checkBox_itemStateChanged(ItemEvent arg0) {
  buff.append(checkBox.getText()+"、");
}
```

举一反三

根据本实例，读者可以实现以下功能。

将复选框勾选的内容存入数据库。

将单选按钮的信息存入数据表。

实例174 将数据从一张表复制到另一张表

这是一个可以美化界面的实例

实例位置：光盘\mingrisoft\04\Ex04_174

实例说明

将数据从一张表中复制到另一张表中的情况很多。例如，有两张表分别为学生表与优秀学生表，这两张表具有相同的数据结构，要求将推举出来的学生信息添加到优秀学生表中。本实例实现了这一功能，实例运行效果如图4.46所示。

图4.46 实例运行效果

技术要点

本实例使用的是INSERT与SELECT语句的组合使用，通常以下情况会使用该种组合形式。

（1）创建查询表。创建查询表主要可以提高检索速度，因为查询时对多个表进行链接操作比较复杂，比简单查询速度要慢，可以创建包含多个表中数据的查找表，这样查询时仅对查找表比较查询，可以加快查询的速度。

（2）修改表。在使用数据库的过程中，可以发现原先的表有不合理的地方，需要对表进行修改。对于简单的修改可使用alter table语句，而对于改动较大的表，可以创建一个包含所需列的表，然后结合INSERT SELECT 语句，将原有表中的数据转换为新表中的数据，这样就不会造成原有数据的丢失。

实现过程

（1）在项目中创建类CopyFrame，该类继承自JFrame类，实现窗体类。向该窗体中添加标签、下拉列表、按钮与表格控件，实现窗体

布局。表格控件中显示学生表中的信息，用于根据该信息选择将哪名学生添加到优秀学生表。

（2）在项目中创建类CopyDate，该类用于定义将学生表中数据添加到优秀学生表中的方法insertStu()，该方法包含一个int类型的参数，用于定义学生的编号。该方法的具体代码如下：

```
public void insertStu(int id) {
  conn=getConn();
      //获取与数据库的连接
  try {
    Statement statement=
conn.createStatement();                                    //获取
Statement对象
    String sql ="insert into tb_excellenceStu select
name,sex,specialty,grade from tb_stu where id ="+id;
    //定义插入数据的SQL语句
    statement.executeUpdate(sql);
            //执行插入语句
  } catch (Exception e) {
    e.printStackTrace();
  }
}
```

举一反三

根据本实例，读者可以开发以下程序。

进行多表间数据的复制。

两表之间的数据复制。

实例175 使用 UNION ALL语句批量插入数据

这是一个可以提高分析能力的实例

实例位置: 光盘\mingrisoft\04\Ex04_175

实例说明

批量向数据表中插入数据是很常见的功能，实现这一功能可以使用多种形式，JDBC 中就有实现批处理的类（本书的后面章节会向大家介绍关于批处理的实例）。本实例向大家介绍的是使用 UNION ALL 语句实现一次向学生表中插入 3 条记录，插入前与插入后的学生表数据如图4.47所示。

id	name	sex	specialty	grade
1	张雪	女	小学教育	07d02
2	格蕾	女	计算机技术	08d01
3	艳旭	男	生物化学	06d01
4	科锦	男	化工燃料	07d04

(a)

id	name	sex	specialty	grade
1	张雪	女	小学教育	07d02
2	格蕾	女	计算机技术	08d01
3	艳旭	男	生物化学	06d01
4	科锦	男	化工燃料	07d04
6	双双	女	生物科学	08d02
7	王爽	女	计算机应用	08d02
8	朱莉	女	英语	07d02

(b)

图4.47 运行本实例后数据表中数据的变化

技术要点

本实例使用UNION ALL 语句实现批处理，该语句的语法格式如下：

```
INSERT tableName
SELECT columnValue,…
UNION ALL
SELECT columnValue,…
```

参数说明

- tableName：要添加数据的数据表。
- columnValue：要添加到数据表中的数据。

实现过程

（1）在项目中创建类BatchInsert，在该类中首先定义数据库连接方法getConn()，该方法以Connection对象作为返回值，具体代码读

者可参考光盘中的源程序，这里不再赘述。

（2）在该类中定义insertStu()方法，该方法包含有一个String类型的参数，用于指定要执行的SQL语句。该方法的具体代码如下：

```
public void insertStu(String sql){
  conn=getConn();
  //获取数据库连接
  try {
    Statement statement=
conn.createStatement();                    //创建
Statement对象
    statement.executeUpdate(sql);
        //执行插入SQL语句
  } catch (Exception e) {
    e.printStackTrace();
  }
}
}
```

（3）在该类的主方法中调用insertStu()方法，实现向学生表中添加数据。具体代码如下：

```
public static void main(String[] args) {
  BatchInsert insert = new BatchInsert();        //创建本
类对象
  String sql ="insert tb_stu select '双双','女','生物科
学','08d02' "+"union all select '王爽','女','计算机应
用','08d02' "+"union all select '朱莉','女','英
语','07d02'";        //定义插入的SQL语句
  insert.insertStu(sql);        //调用插入数据方法
```

}

举一反三

根据本实例，读者可以开发以下程序。

单条插入数据。

批量插入数据。

实例176 更新指定记录

这是一个可以启发思维的实例

实例位置：光盘\mingrisoft\04\Ex04_176

实例说明

在程序开发中，实现对数据的更新是一项很常见的操作。本实例为大家演示了更新学生表中特定记录的实例。运行本实例，首先将学生表中的数据全部显示在窗体中，用户可在该窗体中选择要修改的学生记录，进行修改操作。实例运行效果如图4.48所示。

技术要点

本实例在实现数据更新时，使用了UPDATE语句，该语句的具体语法格式如下：

UPDATE 数据表名 SET 字段名=新的字段值 WHERE 条件表达式

例如，将员工表tb_emp中编号为2的员工姓名修改为“葛雷”。代码如下：

```
update tb_emp set name = '葛雷' where id = 2;
```

(a)

(b)

图4.48 实例运行效果

通过Statement实例的executeUpdate()方法可实现通过Java程序向数据库发送修改SQL语句。

实现过程

（1）在项目中创建类UpdateStuFrame，该类继承自JFrame类，实现窗体类。该类用于显示学生表中的全部信息，该窗体中包含一个JTable控件与两个JButton控件。

（2）创建窗体类 UpdateFrame，用于显示学生信息，供用户修改，在该窗体中添加文本框等控件，主要控件及说明如表4.13所示。

表4.13 UpdateFrame窗体中的主要控件及说明

控 件 类 型	控 件 命 名	控 件 用 途
JTextField	idTextField	显示学生“编号”的文本框控件
	nameTextField	显示学生“姓名”的文本框控件
	gradeTextField	显示学生“年级”的文本框控件
	specialityTextField	显示学生“专业”的文本框控件
JComboBox	sexComboBox	显示学生“性别”的文本框控件
JButton	updateButton	显示“修改”的按钮控件
	closeButton	显示“关闭”的按钮控件

（3）在项目中定义类UpdateStu，在该类中定义更新数据的方法updateStu()，该方法以与学生表tb_stu对应的JavaBean对象为参数。该方法的具体代码如下：

```
public void updateStu(Stu stu){
```

```
  conn=getConn();
      //获取数据库连接
  try {
    PreparedStatement statement =
conn.prepareStatement("update tb_stu set name = ?,sex =
?,grade = ?,specialty = ?where id = ?");        //定义更
新SQL语句
    statement.setString(1,
stu.getName());                                  //设置预处
理语句参数
    statement.setString(2, stu.getSex());
    statement.setString(3,stu.getGrade());
    statement.setString(4, stu.getSpecialty());
    statement.setInt(5, stu.getId());
    statement.execute();
            //执行预处理语句
  } catch (Exception e) {
    e.printStackTrace();
  }
}
```

举一反三

根据本实例，读者可以实现以下程序。

根据查询条件更新数据表中的记录。

更新记录时给出提示。

实例177 在删除数据时给出提示信息

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\04\Ex04_177

实例说明

删除数据表中的数据是一个很常用的技术，在删除数据时，由于删除之后就不能自动恢复，所以在删除数据前给用户相应的提示信息是必要的。本实例实现删除学生表中指定的信息，在删除信息前给出提示信息。实例运行效果如图4.49所示。

图4.49 实例运行效果

技术要点

在 SQL 语句中 DELETE 语句用于删除数据，通过使用 JDBC 技术实现向数据库中发送DELETE语句可实现删除数据。

DELETE语句的语法格式如下：

DELETE FROM 数据表名 where 条件表达式

例如，将tb_emp表中编号为1024的员工信息删除。代码如下：

```
delete from tb_emp where id = 1024;
```

使用Statement对象的executeUpdate()方法可实现向数据库中发送删除语句。

实现过程

（1）在项目中创建类DeleteFrame，该类继承自JFrame类，实现窗体类。在该类中添加表格控件，用于显示学生表信息；再添加按钮

控件。

（2）在项目中定义类DeleteUtil，在该类中定义删除数据的方法deleteStu()，该方法以一个int类型对象为参数，用来指定要删除信息的学生编号。该方法的具体代码如下：

```
public void deleteStu(int id){
  conn = getConn();          //获取数据库连接
  try {
    Statement statement =
conn.createStatement();          //定义更新SQL语句
    statement.executeUpdate("delete from tb_stu where
id="+id);          //执行预处理语句
  } catch (Exception e) {
    e.printStackTrace();
  }
}
```

举一反三

根据本实例，读者可以实现以下程序。

根据指定条件删除表中的信息。

删除数据库中的指定表。

实例178 将数据表清空

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\04\Ex04_178

实例说明

如果 DELETE 语句后面不加任何修饰限制，就会将指定数据表中的数据全部删除。使用TRUNCATE TABLE 语句也可实现将数据表中的数

据全部清除，并且使用TRUNCATE TABLE语句执行速度较快。本实例实现使用TRUNCATE TABLE 语句删除数据表中的所有记录，实例运行效果如图4.50所示。

图4.50 实例运行效果

技术要点

TRUNCATE TABLE 在功能上与不带WHERE 子句的DELETE 语句相同，两者均可以删除表中的全部行，但TRUNCATE TABLE 比DELETE 速度快，且使用的系统和事务日志资源少。

DELETE 语句每次删除一行，会在事务日志中为所删除的每行做一项记录。TRUNCATE TABLE通过释放存储表数据所用的数据页来删除数据，并且只通过在事务日志中释放记录页来减少日志的空间。

实现过程

（1）创建窗体类ClearFrame，该类继承自JFrame类，实现窗体类。在该窗体中添加下拉列表控件，用于显示数据库中的数据表，添加按钮控件。

（2）创建工具类ClearUtil，在该类中定义删除数据表中所有记录的方法deleteDate()，该方法有一个String类型的参数，用于指定要删除记录的数据表。具体代码如下：

```
public void deleteDate(String dataName){
  conn = getConn();          //获取数据库连接
  try {
```

```
    Statement statement =
conn.createStatement();         //获取Statement对象
    statement.executeUpdate("TRUNCATE TABLE
"+dataName);       //指定删除语句
  } catch (Exception e) {
    e.printStackTrace();
  }
}
```

举一反三

根据本实例，读者可以开发以下程序。

清空表中的数据。

删除指定的数据。

第5章 SQL查询相关技术

大小比较与逻辑应用
排序和分组
聚集函数与日期查询
使用子查询
嵌套查询
连接查询
函数查询
JTextPane控件的应用

5.1 大小比较与逻辑应用

实例179 在查询结果中不显示重复记录

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\05\Ex05_179

实例说明

不显示重复记录查询是很常见的一种查询方式。在SQL Server 数据库中提供了DISTINCT关键字来删除重复记录。本实例实现的是查询编程词典销量表中各词典的总销售量。由于在销量表中包含各词典的重复记录，因此在查询时，使用了DISTINCT关键字删除重复记录。实例运行效果如图5.1所示。

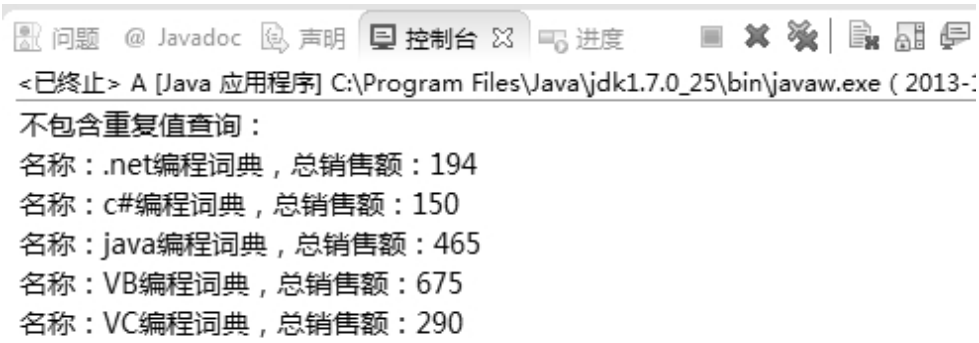


图5.1 实例运行效果

技术要点

在实现查询操作时，如果查询的选择列表中包含一个表的主键，那么每个查询结果中的记录将是唯一的（因为主键在每一条记录中有一个不同的值）。如果主键不包含在查询结果中，就可能出现重复记录。使用DISTINCT关键字以后就可以删除重复记录。

DISTINCT的语法格式如下：

SELECT DISTINCT select_list

参数说明

select_list：要查询的列。

实现过程

（1）创建类BccdDistinctSell，在该类中定义查询数据库中非重复值的方法。首先定义连接数据库的方法getConn()，具体代码读者可参考光盘中的源程序，这里不再赘述。

（2）在该类中定义getBccdSell()方法，用于查询各词典的总销量，该方法返回getBccdSell()集合对象。具体代码如下：

```
public List getBccdSell() {
  List list = new ArrayList();          //定义用于保存返回值的List集合
    conn = getConn();          //获取数据库连接
    try {
      Statement staement = conn.createStatement();
      String sql ="select distinct bccdName , sum(bccdCount) from tb_bccdSell group by bccdName";
      //定义查询数据的SQL语句
      ResultSet set = staement.executeQuery(sql);          //执行查询语句返回查询结果集
      while (set.next()) {          //循环遍历查询结果集
        Bccd bccd = new Bccd();          //定义与数据对应的JavaBean对象
        bccd.setBccdName(set.getString(1));          //设置对象属性
        bccd.setSum(set.getInt(2));
        list.add(bccd);          //向集合中添加对象
```

```
        }
      } catch (Exception e) {
        e.printStackTrace();
      }
      return list;          //返回List集合
    }
```

举一反三

根据本实例，读者可以开发以下程序。

用DISTINCT不显示重复记录。

在显示结果中不显示重复记录。

实例180 使用NOT查询不满足条件的记录

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\05\Ex05_180

实例说明

查询满足某条件的记录很简单，要查询不满足指定条件的数据需要使用 NOT 关键字。本实例实现查询客房表中没有入住的非普间客房信息。本实例实现操作客房表，该表中的数据如图5.2所示。

运行本实例，没有入住的非普间客房显示在控制台上，结果如图5.3所示。

roomid	roomType	roomPrice	roomState	roomFacility	roomDate
201	普间	98	入住	电视	2010/7/8
202	标间	138	空房	电视、空调	
203	标间	138	空房	电视、空调	
204	普间	98	空房	电视	
205	标间	138	入住	电视、空调	2010/7/8
206	普间	98	入住	电视	2010/6/30
207	标间	138	入住	电视、空调	2010/6/30
208	普间	98	空房	电视	

图5.2 客房表中的数据

图5.3 实例运行结果

技术要点

NOT关键字表示“不等”。NOT关键字还可以与其他谓词进行相连来实现组合查询。NOT与谓词进行组合所形成的表达式分别是[NOT] BETWEEN、IS [NOT] NULL和[NOT] IN。

□ [NOT] BETWEEN

该条件指定值的包含范围，使用 AND将开始值和结束值分开。该表达式的语法格式如下：

test_expression [NOT] BETWEEN begin_expression AND end_expression

结果类型为Boolean，返回的值为：如果test_expression的值小于等于begin_expression的值或者大于等于end_expression 的值，则NOT BETWEEN 返回true。

用户还须注意若要指定排除范围，还可以使用大于（>）和小于（<）运算符来代替BETWEEN。

□ IS [NOT] NULL

根据所使用的关键字指定对空值或非空值的查询，如果有任何操作数是 null，表达式取值为null。

□ [NOT] IN

根据使用的关键字是包含在列表内还是排除在列表外，来指定对表达式的查询，查询表达式可以使用常量或列名，而列表可以是一组常量或更多情况下是子查询。如果列表为一组常量则应该放置在一对圆括号内。该表达式的语法格式如下：

```
test_expression[NOT] IN
(
  subquery
  expression[,…n]
)
```

参数说明

● test_expression：SQL 表达式。

● subquery：包含某列结果集的子查询，该列必须与test_expression 具有相同的数据类型。

● expression[,…n]：一个表达式列表，用来测试是否匹配，所有的表达式必须和test_expression具有相同的数据类型。

该语句的返回值是把test_expression与subquery返回的值进行比较，如果两个值相等，或与逗号分隔的列表中的任何expression相等，那么结果值就为true，否则结果值为false。另外，使用NOT IN可以对返回值取反。

实现过程

（1）创建类 RoomUtil ，在该类中定义查询数据的方法。首先定义连接数据库的方法getConn()，该方法的具体代码读者可参考光盘中的源程序，这里不再赘述。

（2）在该类中定义查询没有入住的非普间客房信息方法getRoom()，该方法将查询出的满足条件的客房信息保存在List集合中。具体代码如下：

```
public List getRoom() {
  List list=new
ArrayList();                                    //定
义用于保存返回值的List集合
```

```
  conn=getConn();
      //获取数据库连接
  try {
    Statement staement = conn.createStatement();
    String sql= "select * from tb_room where (not
roomState= '入住') and ( not roomType= '普间')";
    //定义查询数据的SQL语句
    ResultSet set=
staement.executeQuery(sql);                          //执
行查询语句返回查询结果集
    while (set.next())
{                                                    //循环遍历查
询结果集
      Room room = new Room();
      room.setRoomDate(set.getString("roomDate"));
      room.setRoomFacility(set.getString("roomFacility"))
  ;
      room.setRoomid(set.getInt(1));
      room.setRoomPrice(set.getInt("roomPrice"));
      room.setRoomState(set.getString("roomState"));
      room.setRoomType(set.getString("roomType"));
      list.add(room);
    }
  } catch (Exception e) {
    e.printStackTrace();
  }
```

```
        return
    list;
    //返回List集合
      }
```

举一反三

根据本实例，读者可以实现以下功能。

查询不满足条件的记录。

查询满足条件的记录。

实例181 列出销量表中的重复记录和记录条数

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\05\Ex05_181

实例说明

本实例实现了查询编程词典销量表中有多次销售记录的词典信息，还实现了统计重复的次数、词典名称以及各词典的总销量。本实例实现操作编程词典销量表，并将查询结果在控制台上输出，实例运行效果如图5.4所示。

图5.4 实例运行效果

技术要点

实现列出数据中的重复记录和记录条数主要应用了 COUNT 函数和 GROUP BY 子句。

在对数据进行分组和聚集后，再使用HAVING子句中的条件。只有符合条件的组才出现在查询中。

实现过程

（1）定义类BccdSell，在该类中定义连接数据库的方法getConn()，该方法的具体代码读者可参考光盘中的源程序，这里不再赘述。

（2）在该类中定义查询编程词典销量表中数据的方法getBccdSell()，该方法的返回值是List集合。具体代码如下：

```
public List getBccdSell() {
  List list=new
ArrayList();                                          //定
义用于保存返回值的List集合
  conn=getConn();
     //获取数据库连接
  try {
    Statement staement = conn.createStatement();
    String sql ="select count(id) as countId
,bccdName,sum(bccdCount) as sum from tb_bccdSell group by
bccdName having count(id)>1"; //查询有多次销售记录的销售
信息
     ResultSet set=
staement.executeQuery(sql);                           //执
行查询语句返回查询结果集
     while (set.next())
{                                                     //循环遍历查
询结果集
      Bccd bccd = new Bccd();
```

```
            bccd.setCountId(set.getInt(1));
            bccd.setBccdName(set.getString(2));
            bccd.setSum(set.getInt(3));
            list.add(bccd);
          }
        } catch (Exception e) {
          e.printStackTrace();
        }
        return
    list;
    //返回List集合
      }
```

举一反三

根据本实例，读者可以实现以下功能。

列出表中的重复记录。

列出销售表中不重复的记录条数。

实例182 使用关系运算符查询某一时间段数据

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\05\Ex05_182

实例说明

关系运算符包含“<”“>”和“=”等，使用关系运算符对数值类型的数据进行比较很常见，但使用关系运算符同样可以对日期类型的数据进行比较。本实例实现的是查询编程词典销量表中，在2010/6/1～2010/6/30之间有销量的词典信息，并将其在控制台上输出。实例运行效果如图5.5所示。

图5.5 实例运行效果

技术要点

时间比较是按照日期顺序进行比较，排在时间后面的日期大于排在时间顺序前面的日期。例如，时间“2010-07-14”大于时间“2010-01-14”。

逻辑运算符AND可以连接两个或两个以上的条件，只有当AND连接的条件都为true（真）时，AND返回的结果才是true（真）。如果其中有一个条件为false（假），AND返回的值就是false（假）。SQL提供了AND、OR、NOT这3个逻辑运算符。它们都有各自的优先级，这样当它们同时在WHERE子句中出现时，就可以按照规定的优先级顺序执行，不会出现混乱。优先级由高到低的顺序是NOT、AND、OR。

实现过程

（1）定义类BccdSell，在该类中定义连接数据库的方法getConn()，该方法的具体代码读者可参考光盘中的源程序，这里不再赘述。

（2）在该类中定义查询在 2010/6/1～2010/6/30 日之间有销量记录的词典信息方法getBccdSell()，该方法返回List集合。具体代码如下：

```
    public List getBccdSell() {
      List list=new
  ArrayList();                                    //定义用
  于保存返回值的List集合
```

```
    conn=getConn();
    //获取数据库连接
    try {
      Statement staement = conn.createStatement();
      String sql ="select * from tb_bccdSell where bccdDate
  > '2010/6/1' and bccdDate < '2010/6/30'";
      //查询满足条件的SQL语句
      ResultSet set=
  staement.executeQuery(sql);                    //执行查
  询语句返回查询结果集
      while (set.next())
  {                                              //循环遍历查询结
  果集
        Bccd bccd = new Bccd();
        bccd.setId(set.getInt(1));
        bccd.setBccdName(set.getString("bccdName"));
        bccd.setBccdCount(set.getInt("bccdCount"));
        bccd.setBccdPrice(set.getFloat("bccdPrice"));
        bccd.setBccdDate(set.getString("bccdDate"));
        list.add(bccd);
      }
    } catch (Exception e) {
      e.printStackTrace();
    }
    return
list;                                            //返回
List集合
```

}

举一反三

根据本实例，读者可以开发以下程序。

使用between关键字查询某一时间段的数据。

查询“>”某一时间段的数据。

实例183 计算两个日期之间的月份数

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\05\Ex05_183

实例说明

在SQL Server 数据库中提供了DATEDIFF 函数，可用于获取两个日期之间的天数、月份数等。这样对一些数据的统计是十分有用的。本实例实现的是统计商品表中各种商品的上市时间。实例运行效果如图5.6所示。

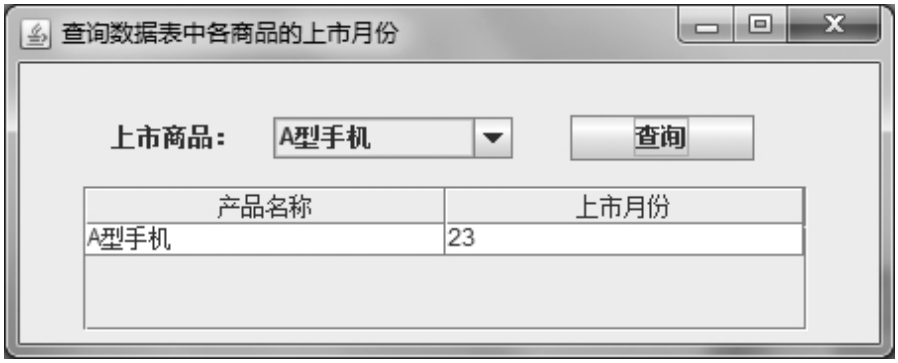


图5.6 实例运行效果

技术要点

DATEDIFF函数用于返回跨两个指定日期的日期与时间边界数。该函数的语法格式如下：

DATEDIFF(datepart,startdate,enddate)

参数说明

● datepart：规定了应在哪一部分计算差额的参数。SQL Server可识别的日期部分及其缩写如表5.1所示。

表5.1 日期部分和缩写

日 期 部 分	缩　　写	日 期 部 分	缩　　写
Year	Yy,yyyy	Week	Wk,ww
Quarter	Qq ,q	Hour	Hh
Month	Mm, m	Minute	Mi,n
Dayofyear	Dy ,y	Second	Ss,s
Day	Dd, d	Millisecond	ms

例如，本实例中获取两个日期之间的月份数，可以设置参数datepart 为 mm；如果要计算两个日期之间的天数，可以将datepart参数设置为day。

● startdate：计算的开始日期。其返回值为datetime 或smalldatetime 值或日期格式字符串的表达式。

● enddate：计算的终止日期。其返回值为datetime 或smalldatetime 值或日期格式字符串的表达式。

实现过程

（1）在项目中创建类MerchandiseFrame，该类继承自JFrame类，实现窗体类。

（2）向窗体中添加控件，实现窗体布局。该窗体中的主要控件及说明如表5.2所示。

表5.2 窗体中的主要控件及说明

控 件 类 型	控 件 命 名	控 件 用 途
JComboBox	nameComboBox	显示商品名称的下拉列表控件
JButton	okButton	显示“查询”的按钮控件
JTable	table	显示查询结果的表格控件

（3）创建MerchandiseUtil类，该类定义了获取数据的方法。首先定义与数据库建立连接的方法getConn()，该方法返回Connection对象，具体代码读者可参考光盘中的源程序，这里不再赘述。

（4）在该类中定义查询商品表中所有信息的方法getMerchandise()，该方法的返回值为List类型。具体代码如下：

```
    public List getMerchandise() {
      conn=getConn();
      //获取数据库连接
      List list=new
  ArrayList();                                          //定义保
  存返回值的List对象
      try {
        Statement staement=
  conn.createStatement();                          //定义
  Statement对象
        String sql= "select* from
  tb_merchandise";                              //定义查询的SQL语句
        ResultSet set=
  staement.executeQuery(sql);                          //执行查
  询语句返回查询结果集
        while (set.next())
  {                                                 //循环遍历查询结
  果集
          Merchandise merchandise = new Merchandise();
          merchandise.setId(set.getInt(1));
          merchandise.setWareName(set.getString(2));
          merchandise.setWareDate(set.getString(3));
          list.add(merchandise);
        }
      } catch (Exception e) {
```

```
        e.printStackTrace();
      }
      return
  list;                                                    //返回
  查询结果
    }
```

（5）在该类中定义getgetMerchandiseDate()方法，用于查询某种商品到当前时间共上市的月份。该方法有一个String类型的参数，用于指定要进行查询的商品名称。具体代码如下：

```
    public String getgetMerchandiseDate(String term) {
      String date ="";
      conn=getConn();                                          //
  获取数据库连接
      try {
        Statement staement = conn.createStatement();
        String sql= "select datediff(mm,(select wareDate from
    tb_merchandise where wareName='"+ term+
    "'),getDate())";                              //定义将图书表中
    信息进行排序的SQL语句
        ResultSet set=
    staement.executeQuery(sql);                    //执行查询语句
    返回查询结果集
        while (set.next())
    {                                          //循环遍历查询结果集
          date = set.getString(1);
        }
      } catch (Exception e) {
```

```
            e.printStackTrace();
        }
        return date;
    }
```

举一反三

根据本实例，读者可以实现以下功能。

计算两个日期之间的天数。

计算两个日期之间的月份数。

实例184 在查询语句中过滤掉字符串中的空格

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\05\Ex05_184

实例说明

数据库中存储的数据可能是不规则的，有可能会带有空格或其他不规范字符。在显示查询结果时，空格之类的不规则字符是不允许出现在查询结果中的，这就需要程序员在处理查询结果时，将查询出来的字符串中的空格去除。本实例实现的操作对象是图书销售表，该表中的数据如图5.7所示。

运行本实例，结果如图5.8所示。

id	bookName	stock	price	bookConcern
1	Java从入门到精通	10	59	清华大学出版社
2	JavaWeb范例宝典	20	89	人民邮电出版社
3	Java自学手册	20	59	人民邮电出版社
4	Java开发实战宝典	10	79	清华大学出版社
5	VB从入门到精通	15	59	清华大学出版社
6	VC从入门到精通	15	59	清华大学出版社
7	Java标准教程	12	59	人民邮电出版社
8	Java从入门到精通	23	59	清华大学出版社

图5.7 图书销售表中的数据

图5.8 实例运行结果

技术要点

本实例主要应用String类中的replaceAll()方法，具体语法格式如下：

replaceAll(String source,String replace)

功能：替换字符。

参数说明

- source：要替换掉的字符串。
- replace：用来替代的字符串。

实现过程

（1）在项目中创建窗体类ConditionFrame，该类继承自JFrame类。向该窗体中添加标签与表格控件，标签控件用于给用户提示信息，表格控件用于显示查询结果。

（2）在项目中创建工具类 ConditionUtil，在该类中定义查询数据的方法 getBookSell()，该方法以List对象作为返回值。具体代码如下：

```
public List getBookSell() {
  List list=new
ArrayList();                                    //定义用于保存
返回值的List集合
```

```
    conn=getConn();                                          //
获取数据库连接
    try {
      Statement staement = conn.createStatement();
      String sql= "select* from
  tb_bookSell";                        //定义查询图书销售表中的
  全部数据
      ResultSet set=
  staement.executeQuery(sql);                  //执行查询语句
  返回查询结果集
      while (set.next())
  {                                        //循环遍历查询结果集
        BookSell sell=new
    BookSell();                            //定义与数据表对应的
    JavaBean对象
        sell.setId(set.getInt(1));
        //设置对象属性
        sell.setBookName(set.getString(2).replace("", ""));
        sell.setStock(set.getString(3).replace("", ""));
        sell.setPrice(set.getFloat(4));
        sell.setBookConcern(set.getString(5).replace("",
    ""));
        list.add(sell);
    //将对象添加到集合中
      }
    } catch (Exception e) {
      e.printStackTrace();
```

```
    }
    return
list;                                                    //返回查询
结果
  }
```

举一反三

根据本实例，读者可以实现以下功能。

过滤掉字符串中的空格。

去除字符串中两端的空格。

5.2 排序和分组

实例185 对数据进行降序查询

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\05\Ex05_185

实例说明

对数据库中的数据进行排序查询是非常有用的查询方式。例如，查询学生表中英语成绩最高的学生信息，就可以按照英语成绩将成绩表进行降序排序，这样得到的第一条数据就是成绩最高的学生信息。本实例实现将图书表中的信息进行降序排序，用户可以灵活地设置查询条件。实例运行效果如图5.9所示。

图5.9 实例运行效果

技术要点

在实现对数据进行降序排序时，应用到了ORDER BY 子句与DESC关键字，只需在SQL查询语句中添加ORDER BY 子句即可。使用ORDER BY 子句可以将数据表中包含的一列或多列表达式的值按升序或降序排序，不会改变数据表中行的顺序，只是改变了查询输出值。ORDER BY子句的语法格式如下：

```
SELECT …
ORDER BY expression[ASC|DESC],…
WHERE…
```

参数说明

- expression：一个表达式，用来指定排序的字段。
- [ASC|DESC]：可选项，表示升序排序或者降序排序。

实现过程

（1）在项目中创建类SelectBookFrame，该类继承自JFrame类，实现窗体类。

（2）向窗体中添加控件，实现窗体布局。该窗体中的主要控件及说明如表5.3所示。

表5.3 窗体中的主要控件及说明

控 件 类 型	控 件 命 名	控 件 用 途
JComboBox	termComboBox	显示排序条件的下拉列表
JButton	termButton	显示“查询”的按钮控件
JTable	resultTable	显示查询结果的表格控件

（3）创建 JDBCUtil 类，该类定义了获取数据的方法，首先定义与数据库建立连接的方法getConn()，该方法返回Connection对象。具体代码如下：

```
public Connection getConn() {
  try {
    Class.forName("net.sourceforge.jtds.jdbc.Driver");
                        //加载数据库驱动
```

```
    } catch (ClassNotFoundException e) {
      e.printStackTrace();
    }//连接数据库URL
    String url
="jdbc:jtds:sqlserver://localhost:1433;DatabaseName=db_data
base22";
    String userName=
"sa";                                          //连接数据
库的用户名
    String passWord=
"";                                            //连接数据
库的密码
    try {
      conn = DriverManager.getConnection(url, userName,
  passWord);        //获取数据库连接
      if (conn != null) {
        System.out.println("数据库连接成功！");
      }
    } catch (SQLException e) {
      e.printStackTrace();
    }
    return
conn;                                                  //
返回Connection对象
  }
```

（4）定义getBook()方法，用于获取图书表中所有数据，以List集合作为返回值。具体代码如下：

```
  public List getBook() {
    conn=getConn();
        //获取数据库连接
    List list = new ArrayList();        //定义保存返回值的
List对象
    try {
      Statement staement =
  conn.createStatement();        //定义Statement对象
      String sql ="select * from tb_book";        //定义查
  询的SQL语句
      ResultSet set = staement.executeQuery(sql);        //
  执行查询语句，返回查询结果集
      while (set.next())
  {                                                    //循环遍历查
  询结果集
        Book book=new
    Book();                                            //定义与数据
    表对应的JavaBean对象
        book.setId(set.getInt(1));
                //设置对象的id值
        book.setBookName(set.getString(2));
        book.setAuthor(set.getString("author"));
        book.setPrice(set.getFloat("price"));
        book.setStock(set.getInt("stock"));
        list.add(book);
            //向集合中添加对象
      }
```

```
    } catch (Exception e) {
      e.printStackTrace();
    }
    return
list;
//返回查询结果
  }
```

（5）定义 getBooKDesc()方法，该方法可将图书表中的数据以指定参数进行排序，将排序后的结果以List形式返回。具体代码如下：

```
  public List getBooKDesc(String term){
    List list = new ArrayList();        //定义用于保存返回
值的List集合
    conn = getConn();        //获取数据库连接
    try {
      Statement staement = conn.createStatement();
      //定义将图书表中信息进行排序的SQL语句
      String sql ="select * from tb_book order by "+term +"
desc";
      ResultSet set = staement.executeQuery(sql);        //
执行查询语句返回查询结果集
      while (set.next()) {        //循环遍历查询结果集
        Book book = new Book();
        book.setId(set.getInt(1));
        book.setBookName(set.getString(2));
        book.setAuthor(set.getString("author"));
        book.setPrice(set.getFloat("price"));
        book.setStock(set.getInt("stock"));
```

```
            list.add(book);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return list;
}
```

举一反三

根据本实例，读者可以实现以下功能。

对数据进行升序查询。

按照特定条件对数据库进行降序查询。

实例186 对数据进行多条件排序查询

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\05\Ex05_186

实例说明

ORDER BY 子句支持多条件查询，可以在ORDER BY 子句中定义多个排序条件。本实例实现的是对图书表中的数据进行多条件查询。用户可以指定其按照售价的升序或降序，库存的升序或降序排序，并将排序结果显示在窗体中。实例运行效果如图5.10所示。

图5.10 实例运行效果

技术要点

本实例实现对图书表中的数据以售价、库存进行排序，其中应用了ORDER BY子句和ASC、DESC关键字。ASC关键字表示升序排序，DESC关键字表示降序排序。默认情况下是按升序排序，对数据进行升序排序时，ASC关键字可以省略。

实现过程

（1）创建TaxisFrame类，该类继承自JFrame类，实现窗体类。

（2）向该窗体中添加控件。窗体的主要控件及说明如表5.4所示。

表5.4 窗体的主要控件及说明

控 件 类 型	控 件 命 名	控 件 用 途
JComboBox	priceComboBox	显示价钱的排序方式
	stockComboBox	显示库存的排序方式
JButton	queryButton	显示“查询”按钮控件
JTable	table	显示查询结果的表格控件

（3）编写工具类JDBCUtil，在该类中定义查询数据库方法，连接数据库代码读者可参考光盘中的源程序，这里不再赘述。定义多条件排序查询方法的代码如下：

```
public List getBooKDesc(String term , String compositor)
{
    List list = new ArrayList();          //定义用于保存返回
  值的List集合
    conn=getConn();
        //获取数据库连接
    try {
      Statement staement = conn.createStatement();
      //定义将图书表中信息进行排序的SQL语句
```

```
    String sql ="select * from tb_book order by price "+
term +", stock "+ compositor;
    ResultSet set = staement.executeQuery(sql);        //
执行查询语句返回查询结果集
    while (set.next()) {         //循环遍历查询结果集
      Book book = new Book();
      book.setId(set.getInt(1));
      book.setBookName(set.getString(2));
      book.setAuthor(set.getString("author"));
      book.setPrice(set.getFloat("price"));
      book.setStock(set.getInt("stock"));
      list.add(book);
    }
  } catch (Exception e) {
    e.printStackTrace();
  }
  return list;
}
```

举一反三

根据本实例，读者可以实现以下功能。

对数据进行多条件升序查询。

对数据进行单一条件降序查询。

实例187 对统计结果进行排序

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\05\Ex05_187

实例说明

在设计数据表时，不会包含可以通过运算得到的列（如成绩表中不可以包含有总分列），这样如果用户需要按照总分进行排序查询，就要通过一定的运算。本实例为大家介绍的就是对统计结果进行排序查询。实例运行效果如图5.11所示。

图5.11 实例运行效果

技术要点

本实例对学生成绩表中的数据进行统计排序。学生成绩表中包含学生的数学、英文、语文等成绩，要计算各成绩之和，就要对各成绩进行相加运算，这时要使用到连接符“+”，经过运算后的结果可以使用关键字“as”为统计结果设置别名，在ORDER BY 排序语句中可以使用别名进行排序。

实现过程

（1）创建ResultFrame类，该类继承自JFrame类，实现窗体类。

（2）向窗体中添加控件，实现窗体布局。窗体中的主要控件及说明如表5.5所示。

表5.5 窗体中的主要控件及说明

控 件 类 型	控 件 命 名	控 件 用 途
JButton	ascButton	显示“按总分升序排序”按钮
	descButton	显示“按总分降序排序”按钮
JTable	table	显示查询结果的表格控件

（3）定义工具类ResultUtil，在该类中定义连接数据库和查询数据的方法，具体代码读者可参考光盘中的源程序。定义按照统计结果进行排序方法，具体代码如下：

```
public List getGradeDesc(String taxis) {
  List list = new ArrayList();        //定义用于保存返回值的List集合
  conn=getConn();                                    //获取数据库连接
  try {
    Statement staement = conn.createStatement();
    String sql ="select id,name,(chinses+math+english+physics+history) as sum from tb_grade order by sum "+taxis;
    //定义将图书表中信息进行排序的SQL语句
    ResultSet set=staement.executeQuery(sql);                    //执行查询语句返回查询结果集
    while (set.next())
    {                                          //循环遍历查询结果集
      Grade grade=new Grade();                                    //创建与学生成绩表对应的Grade对象
      grade.setId(set.getInt(1));
    //设置对象属性
      grade.setName(set.getString(2));
      grade.setSum(set.getFloat("sum"));
      list.add(grade);                                            //将Grade对象添加到集合中
```

```
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    return
list;                                                    //返回查
```
询集合

```
    }
```

举一反三

根据本实例，读者可以实现以下功能。

对统计结果进行升序排列。

对统计结果进行降序排列。

实例188 查询SQL Server数据库中的前3条数据

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\05\Ex05_188

实例说明

SQL Server 数据库有一项特别的功能就是利用TOP 子句来显示结果集，通过该子句可以列出结果集中前几条或者后几条的记录。本实例实现使用TOP关键字求出学生成绩表中英语成绩排在前3名的学生信息，实例运行效果如图5.12所示。

图5.12 实例运行效果

技术要点

采用TOP关键字实现本实例，TOP关键字在查询语句中位于SELECT子句的后面，语法格式如下：

```
SELECT TOP n [PERCENT]
FROM tbaleName
WHERE …
ORDER BY …
```

参数说明

- n：查询的数据行数。
- [PERCENT]：可选参数，返回行的百分比。

实现过程

（1）定义类 ResultUtil，用于操作数据库连接。在该类中定义数据库连接方法 getConn()，具体代码可参考光盘中的源程序。

（2）在该类中定义 getGradeDesc()方法，用于查询英语成绩排在前 3 名的学生信息。具体代码如下：

```
public List getGradeDesc() {
  List list = new ArrayList();          //定义用于保存返回值的List集合
  conn=getConn();
      //获取数据库连接
  try {
    Statement staement = conn.createStatement();
    String sql ="select top 3 id ,name,english from tb_Grade order by english desc";
    //定义将图书表中信息进行排序的SQL语句
```

```
    ResultSet set = staement.executeQuery(sql);          //
执行查询语句返回查询结果集
    while (set.next())
{                                                    //循环遍历查
询结果集
      Grade grade = new Grade();          //创建与学生成绩
  表对应的Grade对象
      grade.setId(set.getInt(1));
            //设置对象属性
      grade.setName(set.getString(2));
      grade.setEnglish(set.getFloat(3));
      list.add(grade);
          //将Grade对象添加到集合中
    }
  } catch (Exception e) {
    e.printStackTrace();
  }
  return
list;
//返回查询集合
  }
```

举一反三

根据本实例，读者可以实现以下功能。

查询后三条记录。

对查询出的前三条记录按照升序排列。

实例189 查询SQL Server数据库中的后3条数据

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\05\Ex05_189

实例说明

查询后3条数据与查询前3条数据的方法类似，都是使用TOP关键字。将数据进行升序排序后，取前3条数据，就可以获取满足条件的数据。本实例应用TOP关键字查询学生成绩表中英语成绩排在后3名的学生信息，实例运行效果如图5.13所示。

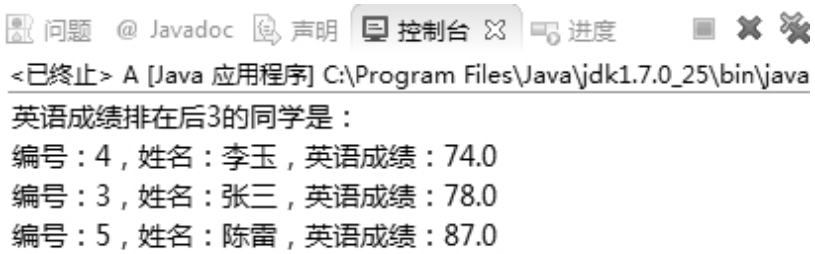


图5.13 实例运行效果

技术要点

查询后3 条数据同样使用的是TOP 关键字。在本实例中，无须在ORDER BY 子句后添加ASC关键字，就可以将学生成绩表中的数据按升序排序。

实现过程

（1）定义ResultUtil类，用于编写操作数据方法。在该类中定义连接数据库方法getConn()，具体代码可参考光盘中的源程序。

（2）定义查询SQL Server 数据库中后3 条数据的方法getGradeDesc()。具体代码如下：

```
public List getGradeDesc() {
  List list = new ArrayList();          //定义用于保存返回值的List集合
  conn=getConn();                                    //获取数据库连接
```

```
    try {
      Statement staement = conn.createStatement();
      //定义将图书表中信息进行排序的SQL语句
      String sql ="select top 3 id ,name,english from
  tb_Grade order by english";
      ResultSet set = staement.executeQuery(sql);        //
  执行查询语句返回查询结果集
      while (set.next())
  {                                        //循环遍历查询结果集
        Grade grade = new Grade();         //创建与学生成绩
    表对应的Grade对象
        grade.setId(set.getInt(1));        //设置对象属性
        grade.setName(set.getString(2));
        grade.setEnglish(set.getFloat(3));
        list.add(grade);                                    /
    /将Grade对象添加到集合中
      }
    } catch (Exception e) {
      e.printStackTrace();
    }
    return
list;                                              //返回查询
集合
  }
```

举一反三

根据本实例，读者可以实现以下功能。

按照指定条件查询后3条记录。

对后3条数据进行有序排列。

实例190 查询MySQL数据库中的前3条数据

这是一个可以用来提高基础性能的实例

实例位置：光盘\mingrisoft\05\Ex05_190

实例说明

MySQL数据库中没有TOP关键字，要实现查询MySQL数据库中的前3条数据，可以使用limit关键字，limit可实现限制SQL语句返回的行数。本实例实现查询MySQL数据库的学生成绩表中语文成绩排在前3名的同学信息，实例运行效果如图5.14所示。

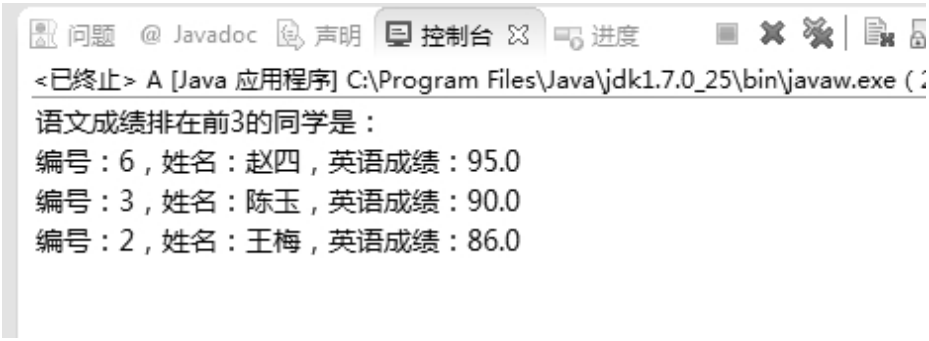


图5.14 实例运行效果

技术要点

MySQL数据库中提供了LIMIT函数用于显示SELECT查询语句返回的行数。如果查询语句中包含GROUP BY 与ORDER BY 子句，则LIMIT 子句在GROUP BY 与ORDER BY 子句的后面。语法格式如下：

```
SELECT [DISTIN|UNIQUE](*,columname[AS alias],…)
FROM table
WHERE …
ORDER BY…
LIMIT([offset],rows)
```

参数说明

● offset：指定要返回的第一行的偏移量。开始行的偏移量是0。

● rows：指定返回行的最大数目。

如查询学生表中英语成绩排在前3名的学生信息，SQL语句如下：

```
select * from tb_student order by english desc limit 0,3
```

实现过程

（1）创建类MySQLConn，该类定义了查询MySQL数据库中前3条数据的方法，首先定义与MySQL连接的方法。该方法以Connection对象作为返回值，具体代码如下：

```
public Connection getConnection() {
  try {
    Class.forName("com.mysql.jdbc.Driver");          //加载MySQL数据库驱动
    String url ="jdbc:mysql://localhost:3306/db_database21";          //定义连接数据库的url
    String user ="root";          //定义连接数据库的用户名
    String passWord ="111";          //定义连接数据库的密码
    conn = DriverManager.getConnection(url, user, passWord);          //获取数据库连接
  } catch (Exception e) {
    e.printStackTrace();
  }
  return conn;
}
```

（2）在该类中定义getOrderDesc()方法，该方法用于获取查询MySQL数据库中tb_student表按语文成绩降序排序后的前3条记录。具

体代码如下：

```
    public List getOrderDesc() {
      List list = new ArrayList();          //定义用于保存返回
  值的List集合
      conn = getConnection();          //获取数据库连接
      try {
        Statement staement = conn.createStatement();
        //定义查询数据表中前3条记录的SQL语句
        String sql ="select * from tb_student order by
    chinese desc limit 0,3";
        ResultSet set = staement.executeQuery(sql);          //
    执行查询语句返回查询结果集
        while (set.next()) {          //循环遍历查询结果集
          Student student = new Student();          //定义与数
    据库对应的JavaBean对象
          student.setId(set.getInt(1));          //设置对象属性
    值
          student.setName(set.getString("name"));
          student.setSex(set.getString("sex"));
          student.setClassName(set.getString("className"));
          student.setChinese(set.getFloat("chinese"));
          list.add(student);          //将JavaBean添加到集合中
        }
      } catch (Exception e) {
        e.printStackTrace();
      }
      return list;
```

}

举一反三

根据本实例，读者可以实现以下功能。

对数据库中的前3条记录进行有序排列。

查询数据库中前10条记录。

实例191 查询MySQL数据库中的后3条数据

这是一个可以提高基础性能的实例

实例位置：光盘\mingrisoft\05\Ex05_191

实例说明

在MySQL数据库中应用limit关键字同样可以查询数据表中后几名的数据。本实例实现在学生成绩表中查询语文成绩排在后3名的学生信息，实例运行效果如图5.15所示。

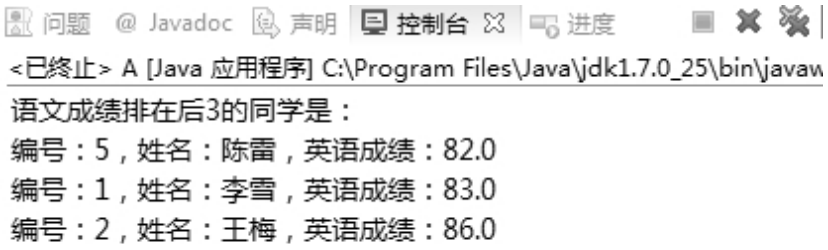


图5.15 实例运行效果

技术要点

实现在MySQL数据库中查询后3条记录的方法与查询前3条记录的方法基本一致。同样采用limit关键字。但需要注意的是limit关键字本身并没有获取某表中指定数据的能力，要完成获取前几条或后几条数据，需要借助ORDER BY 子句，先将数据进行排序后再使用limit 关键字来限制SQL语句返回的行数。

本实例查询学生成绩表中后3条记录的SQL语句如下：

select id,name,sex,className,chinese from tb_student order by chinese limit 0,3

实现过程

（1）在项目中定义类 MySQLConn，该类定义从学生成绩表中查询数据的方法。首先定义连接数据库方法，具体代码可参考光盘中的源程序，这里不再赘述。

（2）在该类中定义查询学生成绩表中语文成绩排在后 3 名的学生信息的方法getOrderDesc()。具体代码如下：

```
public List getOrderDesc() {
  List list=new
ArrayList();                                    //定义用
于保存返回值的List集合
  conn=getConnection();
    //获取数据库连接
  try {
    Statement staement = conn.createStatement();
    String sql ="select id,name,sex,className,chinese
from tb_student order by chinese limit 0,3";
    //定义查询数据表中后3条记录的SQL语句
    ResultSet set=
staement.executeQuery(sql);                     //执行查
询语句返回查询结果集
    while (set.next())
{                                               //循环遍历查询结
果集
      Student student=new
  Student();                                    //定义与数据库对应
```

```
的JavaBean对象
            student.setId(set.getInt(1));
                  //设置对象属性值
            student.setName(set.getString("name"));
            student.setSex(set.getString("sex"));
            student.setClassName(set.getString("className"));
            student.setChinese(set.getFloat("chinese"));
            list.add(student);
              //将JavaBean添加到集合中
          }
        } catch (Exception e) {
          e.printStackTrace();
        }
        return list;
    }
```

举一反三

根据本实例，读者可以实现以下功能。

对数据库中后3条的记录按照指定条件筛选。

对后3条记录按照指定条件排列。

实例192 按照字母顺序对留学生表进行排序

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\05\Ex05_192

实例说明

在实际应用中会遇到将数据表按某一字母排序，将字母靠前的内容显示在查询结果的前面。本实例实现的是将留学生表 tb_abrod 按

名字字段中第一个字母排序，主要应用 ORDER BY 子句和 SUBSTRING 函数。没有进行排序的数据表内容如图 5.16 所示，排序后的内容如图 5.17所示。

id	name	surname	nationality
1	kelly	KONG	美国
2	chuch	TAYLOR	加拿大
3	armando	PURVIS	美国
4	laura	CHRISINT	英国

图5.16 没有进行排序的数据表内容

图5.17 排序后的内容

技术要点

本实例是按SUBSTRING函数返回的字符串排序，从效果图可以看出3号留学生，由于名字的第一个字母为“a”，所以他排在了第一位。

SUBSTRING函数主要用于返回字符、binary、text或image表达式的一部分。该函数的语法格式如下：

SUBSTRING(expression,start,length)

参数说明

● expression：字符串、二进制字符串、text、image、列或包含列的表达式。不可以使用包含聚合函数的表达式。

● start：一个整数，指定字符串的开始位置。

● length：一个整数，指定字符串的长度（要返回的字符数或字节数）。

实现过程

（1）创建工具类ResultUtil，用于实现查询留学生表数据，在该类中定义了与数据库建立连接的方法getConn()，具体代码读者可参考光盘中的源程序，这里不再赘述。

（2）定义将留学生表按音序排序的方法 getGradeDesc()，该方法将查询后的结果以 List 集合返回。具体代码如下：

```
public List getGradeDesc() {
  List list = new ArrayList();        //定义用于保存返回值的List集合
  conn=getConn();                                    //获取数据库连接
  try {
    Statement staement = conn.createStatement();
    //定义将图书表中信息进行排序的SQL语句
    String sql ="select * from tb_abroad order by substring(name,1,1)";
    ResultSet set = staement.executeQuery(sql);        //执行查询语句返回查询结果集
    while (set.next()) {        //循环遍历查询结果集
      Abroad abrod = new Abroad();        //创建与学生成绩表对应的Grade对象
      abrod.setId(set.getInt(1));        //设置对象属性
      abrod.setName(set.getString(2));
      abrod.setSurname(set.getString(3));
      abrod.setNationality(set.getString(4));
      list.add(abrod);
    }
  } catch (Exception e) {
    e.printStackTrace();
  }
```

```
    return list;                                    //返
回查询集合
    }
```

举一反三

根据本实例，读者可以实现以下功能。

按照字母顺序对表进行排序。

统计来自同一国家的学生人数。

实例193 按姓氏笔画排序

这是一个可以启发思维的实例

实例位置：光盘\mingrisoft\05\Ex05_193

实例说明

按姓氏笔画排序是一种常见的排序方式，例如，对图书中很多作者的排名都是按照姓氏笔画排序的。SQL Server 数据库中提供了COLLATE 子句可实现将数据表中数据按姓氏笔画排序。本实例将学生成绩表中的数据按照姓名列进行姓氏笔画排序。实例运行效果如图5.18所示。

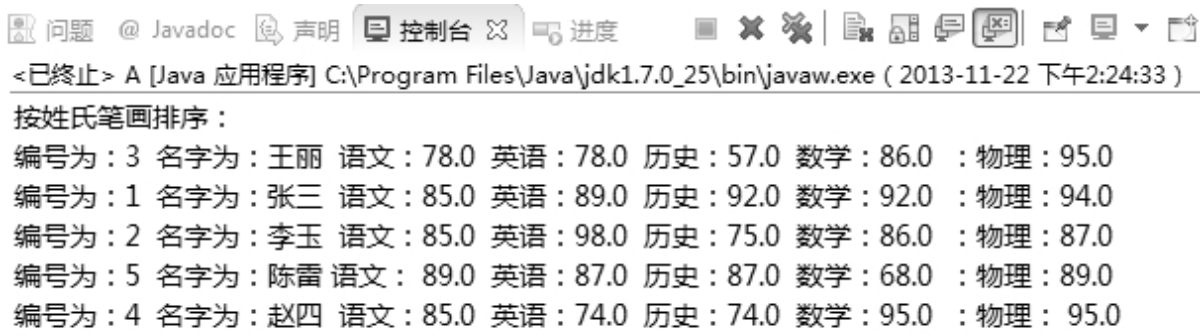


图5.18 实例运行效果

技术要点

COLLATE 子句可应用于数据库定义或列定义以定义排序规则，或应用于字符串表达式以应用排序规则投影。语法格式如下：

COLLATE < collation_name >< collation_name >::={ Windows_collation_name }|{ SQL_collation_name }

参数说明

● collation_name：应用于表达式、列定义或数据库定义的排序规则的名称。

● Windows_collation_name：Windows 排序规则的排序规则名称。

● SQL_collation_name：SQL 排序规则名称。

COLLATE子句只能应用于char、varchar、text、nchar、nvarchar和ntext数据类型。排序规则一般由排序规则名标识。例如，“chinese_prc_stroke_cs_as_ks_ws”表示的就是排序规则，其中“chinese_prc_”指对汉字的UNICODE排序规则，“stroke”标识查询结果按姓氏笔画排序。排序规则的后半部分后缀的含义如下。

□ _bin：二进制排序。

□ _ci(cs)：是否区分大小写，ci 不区分，cs 区分。如果想让比较将大写字母和小写字母视为不等，可以选择该选项。

□ _ai(as)：是否区分重音，ai 不区分，as 区分。如果想让比较将重音和非重音字母视为不等，可以选择该选项。如果选择该选项，比较还将重音不同的字母视为不等。

□ _ki(ks)：是否区分假名类型，ki 不区分，ks 区分。如果想让比较将片假名和平假名日语音节视为不等，可以选择该选项。

□ _wi(ws)：是否区分宽度，wi 不区分，ws 区分。如果想让比较将半角字符和全角字符视为不等，可以选择该选项。

实现过程

（1）创建类 GradeOrderUtil，实现将数据表中的内容按音序排序。在该类中定义了连接数据库的方法getConn()，具体代码可参考光盘中的源程序，这里不再赘述。

（2）在该类中定义getGradeOrder()方法，实现将学生成绩表按笔画排序。具体代码如下：

```
public List getGradeOrder() {
  List list = new ArrayList();          //定义用于保存返回值的List集合
    conn=getConn();
    //获取数据库连接
    try {
      Statement staement = conn.createStatement();
      String sql ="select * from tb_Grade order by name collate chinese_prc_stroke_cs_as_ks_ws";          //定义按笔画排序的SQL语句
      ResultSet set = staement.executeQuery(sql);          //执行查询语句返回查询结果集
      while (set.next()) {          //循环遍历查询结果集
        Grade grade = new Grade();          //实例化学生成绩对象
          grade.setId(set.getInt(1));          //设置对象的编号
          grade.setName(set.getString(2));
          grade.setChinses(set.getFloat("chinses"));
          grade.setEnglish(set.getFloat("english"));
          grade.setHistory(set.getFloat("history"));
          grade.setMath(set.getFloat("math"));
          grade.setPhysics(set.getFloat("physics"));
```

```
            list.add(grade);
          }
        } catch (Exception e) {
          e.printStackTrace();
        }
        return
    list;                                                    //返回
    查询集合
      }
```

举一反三

根据本实例，读者可以实现以下功能。

按姓氏笔画进行排序。

按照汉字拼音进行排序。

实例194 将汉字按音序排序

本实例可以提高工作效率

实例位置：光盘\mingrisoft\05\Ex05_194

实例说明

SQL Server 汉字排序规则可以按音序、笔画排序。本实例实现的是将学生成绩表中的内容按音序排序后输出。实例运行效果如图5.19所示。

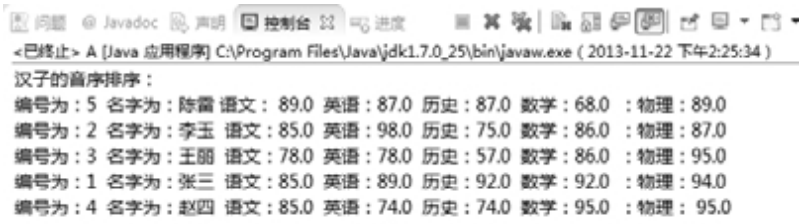


图5.19 实例运行效果

技术要点

按音序排序同样是用到了 COLLATE 函数来完成查询排列，chinese_prc_ci_as 指定排序规则。排序规则可以参考按姓氏笔画排序的规则。不使用后缀_stroke，数据库会按指定的表达式音序进行排序。

实现过程

（1）在项目中创建类 GradeOrderUtil，实现将数据表中的内容按音序排序。在该类中定义了连接数据库方法getConn()，具体代码可参考光盘中的源程序，这里不再赘述。

（2）在该类中定义getGradeOrder()方法，实现将学生成绩表按音序排序。具体代码如下：

```
public List getGradeOrder() {
  List list = new ArrayList();          //定义用于保存返回值的List集合
  conn=getConn();
  //获取数据库连接
  try {
    Statement staement = conn.createStatement();
    String sql ="select * from tb_Grade order by name collate chinese_prc_stroke_cs_as ";          //定义按音序排序的SQL语句
    ResultSet set = staement.executeQuery(sql);          //执行查询语句返回查询结果集
    while (set.next()) {          //循环遍历查询结果集
      Grade grade = new Grade();          //实例化学生成绩对象
      grade.setId(set.getInt(1));          //设置对象的编号
```

```
            grade.setName(set.getString(2));
            grade.setChinses(set.getFloat("chinses"));
            grade.setEnglish(set.getFloat("english"));
            grade.setHistory(set.getFloat("history"));
            grade.setMath(set.getFloat("math"));
            grade.setPhysics(set.getFloat("physics"));
            list.add(grade);
          }
        } catch (Exception e) {
          e.printStackTrace();
        }
        return
    list;                                                    //返回
    查询集合
      }
```

举一反三

根据本实例，读者可以实现以下功能。

将汉字按笔画升序排序。

筛选出笔画最少的汉字姓氏。

实例195 从表中随机返回记录

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\05\Ex05_195

实例说明

SQL Server 数据库中提供了两种随机数函数，分别为RAND 与NOWID 函数。其中RAND函数在一个查询中只能返回一个结果，NOWID

函数返回的列可以做ORDER BY 排序处理。本实例使用NOWID函数实现从表中随机返回3条记录。实例运行效果如图5.20所示。

图5.20 实例运行效果

技术要点

要从数据表中随机返回数据记录信息或者对数据记录进行随机排序，可以使用随机数。由于随机数RAND函数在一个查询中只能返回一个结果，因此可以在NOWID函数返回的列上做ORDER BY 分组处理。

实现过程

（1）在项目中创建类 RandomFrame，该类继承自 JFrame 类，实现窗体类。向该窗体中添加标签、按钮与表格控件，标签控件用于给用户提示信息，按钮控件可用于查询数据，表格控件用于显示查询结果。

（2）在项目中创建工具类AbroadUtil，在该类中定义查询数据的方法getAbroad()，该方法以List集合返回查询结果。具体代码如下：

```
    public List getAbroad() {
      List list=new
  ArrayList();                                    //定义用于保存
```

```
返回值的List集合
    conn=getConn();                                          //
获取数据库连接
    try {
      Statement staement = conn.createStatement();
      //定义查询数据的SQL语句
      String sql ="select top 3 * from tb_abroad order by
  newid()";
      ResultSet set=
  staement.executeQuery(sql);                  //执行查询语句
  返回查询结果集
      while (set.next())
  {                                          //循环遍历查询结果集
        Abroad abord=new Abroad();                          //
    定义与数据表对应的JavaBean对象
        abord.setId(set.getInt(1));
        abord.setFristName(set.getString(2));
        //设置对象属性
        abord.setLastName(set.getString(3));
        abord.setNationality(set.getString(4));
        abord.setSpeciality(set.getString(5));
        list.add(abord);
      }
    } catch (Exception e) {
      e.printStackTrace();
    }
```

```
    return
list;                                                    //返回List
集合
  }
```

举一反三

根据本实例，读者可以实现以下功能。

使用RAND函数随机返回记录。

使用NOWID函数随机返回记录。

实例196 使用GROUP BY子句实现对数据的分组统计

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\05\Ex05_196

实例说明

使用GROUP BY 子句对查询具有重要的作用，可以对数据的某一列进行分组统计。本实例实现对定义表中的数据按签约人进行分组统计，这样就可以查询出各员工签订的总订单信息，进而可以进行业绩考核。本实例实现总结员工签订的订单总额，并按降序排序。实例运行效果如图5.21所示。

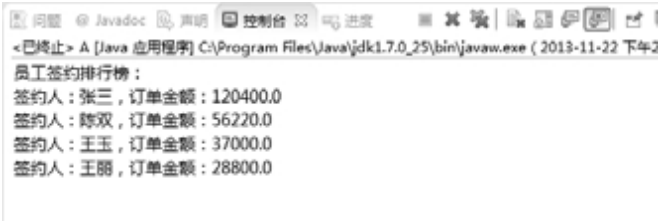


图5.21 实例运行效果

技术要点

实现数据分组时，应用GROUP BY 子句与聚集函数相结合获取需要的数据。GROUP BY子句是指定用来放置输出行的组，具体语法格式如下：

[GROUP BY[ALL] expression[,…n]
[WHERE {CUBE|ROLLUP}]
]

参数说明

● ALL：包含与选定列表中匹配的所有组合结果集，如果用户指定了 ALL，将对组中不满足搜索条件的汇总列返回空值。

● expression：对查询执行分组的表达式，即要进行分组的列。在选择列表中定义的列的别名，不能用于指定分组列。

● CUBE：指定在查询的结果集内不仅包含GROUP BY子句提供的正常行，还包含汇总行。

● ROLLUP：指定在结果集内不仅包含GROUP BY 子句提供的正常行，还包含汇总行。

实现过程

（1）创建类OrderFormUtil，用于实现分组统计。在该类中定义连接数据库的方法getConn()，具体代码读者可参考光盘中的源程序，这里不再赘述。

（2）在该类中定义分组统计方法getOrderDesc()，该方法以List作为返回值。具体代码如下：

```
public List getOrderDesc() {
  List list=new
ArrayList();                                    //定义用于保存
返回值的List集合
  conn=getConn();                                          //
获取数据库连接
  try {
    Statement staement = conn.createStatement();
```

```
    String sql= "select visePerson,sum(clientMoney) as
money from tb_orderForm  group by visePerson order by
money
desc";
      //定义将订单表进行分组统计的SQL语句
    ResultSet set=
staement.executeQuery(sql);                    //执行查询语句
返回查询结果集
    while (set.next())
{                                          //循环遍历查询结果集
      OrderForm orderForm = new OrderForm();
      orderForm.setClientMoney(set.getFloat("money"));
      orderForm.setVisePerson(set.getString("visePerson")
  );
      list.add(orderForm);
    }
  } catch (Exception e) {
    e.printStackTrace();
  }
```

举一反三

根据本实例，读者可以实现以下功能。

使用GROUP BY 实现指定条件分组。

使用GROUP BY 进行普通分组。

实例197 使用GROUP BY子句实现多表分组统计

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\05\Ex05_197

实例说明

在实际开发中，可能会有多个表都包含同一种数据的相关信息。例如，在进货表中保存了商品的进货信息，在销售表中保存了商品的相关销售信息，要查询某种商品的相关信息，就要对进货表与销售表都进行查询。本实例使用分组函数查询有销售记录的商品信息。实例运行效果如图5.22所示。

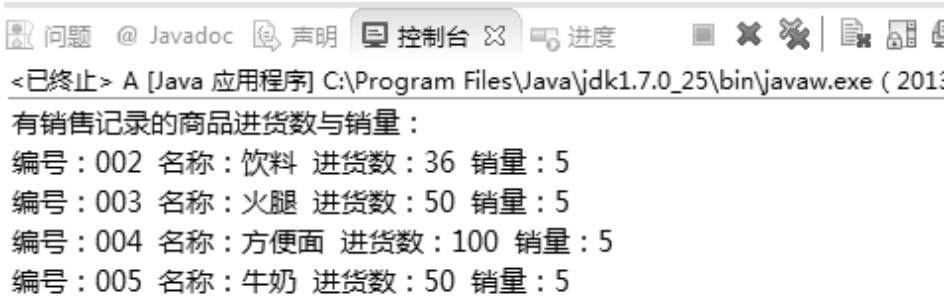


图5.22 实例运行效果

技术要点

本实例中使用聚集函数SUM 计算出某种商品的总销量，使用GROUP BY 子句进行分组统计。

实现过程

（1）创建JDBCUtil类，用于实现有销售记录的商品库存和销量数据。在该类中定义连接数据库的方法，具体代码读者可参考光盘中的源程序，这里不再赘述。

（2）在该类中定义查询进货表和销售表的方法 getBooKDesc()，获取有销售记录的商品信息。代码如下：

```
public ResultSet getBooKDesc() {
  List list=new
ArrayList();                                    //定义用于保存返回
值的List集合
```

```
    conn=getConn();                                    //获取
数据库连接
    ResultSet set = null;
    try {
      Statement staement = conn.createStatement();
      String sql ="select
  cargoId,cargoName,cargoCount,sum(sellCount)as count "+"
  from tb_cargo,tb_sell where cargoId =sellId group by
  sellId,cargoName,cargoId,cargoCount";
      //定义查询数据库的SQL语句
      set= staement.executeQuery(sql);                  //
  执行查询语句返回查询结果集
    } catch (Exception e) {
      e.printStackTrace();
    }
    return set;
  }
```

举一反三

根据本实例，读者可以实现以下功能。

使用GROUP BY 子句进行多表分组统计。

使用GROUP BY 指定分组查询。

5.3 聚集函数与日期查询

实例198 利用SUM函数实现数据汇总

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\05\Ex05_198

实例说明

SUM聚集函数用来表示返回数据的总和。本实例应用SUM聚集函数计算出订单表中第一季度的总订单金额。本实例查询的是订单表tb_achievement。该表中的数据如图5.23所示。

本实例查询第一季度的订单金额的运行效果如图5.24所示。

id	dept	achievement	monthCount
1	华北区	25000	1
2	东北区	36000	1
3	华北区	68000	2
4	华北区	98000	3
5	东北区	68700	2
6	东北区	26000	4
7	华北区	58700	4
8	东北区	56000	5
9	华北区	59800	5

图5.23 tb_achievement 表中的数据

图5.24 实例运行效果

技术要点

SUM聚集函数用于返回表达式中所有值的和，但需要注意的是该函数只能用于数据类型的数字列。在统计的过程中null值被忽略。

SUM函数的具体语法格式如下：

SUM([ALL | DISTINCT] expression)

参数说明

● ALL：对所有的值进行聚集函数运算。ALL 是默认设置。

● DISTINCT：指定SUM 返回唯一值的和。

● expression：常量、列名、函数以及算术运算符、按位运算符和字符串运算符的任意组合。expression 是精确数字或近似数字数据类型分类（bit 数据类型除外）的表达式，不允许使用聚集函数和子查询。

实现过程

（1）创建类AchievementUtil，在该类中定义连接数据库的方法getConn()。具体代码读者可参考光盘中的源程序，这里不再赘述。

（2）在该类中定义查询订单表中第一季度的订单总和。具体代码如下：

```
public float getBooKDesc() {
  conn=getConn();
      //获取数据库连接
  float sum = 0;
  try {
    Statement staement = conn.createStatement();
    String sql ="select sum(achievement)as sum from
tb_achievement where monthCount <= 3";
    //定义查找第一季度定义总和的SQL语句
    ResultSet set=
staement.executeQuery(sql);                          //执
行查询语句返回查询结果集
    while (set.next())
{                                                    //循环遍历查
```

询结果集

```
        sum=
    set.getFloat(1);
      //获取查询结果
      }
    } catch (Exception e) {
      e.printStackTrace();
    }
    return sum;
}
```

举一反三

根据本实例，读者可以实现以下功能。

SUM函数进行数据汇总。

SUM函数进行列统计总和。

实例199 利用AVG函数实现计算平均值

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\05\Ex05_199

实例说明

在查询数据表中某列字段的平均值时，可以使用AVG函数。本实例实现通过AVG函数，计算出学生成绩表中学生各科成绩的平均值。学生成绩表中的原有数据如图5.25所示。

本实例实现查询学生成绩表中各科成绩的平均值，实例运行效果如图5.26所示。

id	name	chinese	math	english	physics	history
1	张三	85	92	89	94	92
2	李玉	85	86	98	87	75
3	王丽	78	86	78	95	57
4	赵四	85	95	74	95	74
5	陈雷	89	68	87	89	87

图5.25 学生成绩表中的原有数据

图5.26 实例运行效果

技术要点

AVG聚集函数主要用于返回组中值的平均值，空值将被忽略。该函数的语法格式如下：

AVG ([ALL|DISTINCT] expression)

参数说明

● ALL：对所有的值进行聚集函数运算。ALL 是默认设置。

● DISTINCT：指定AVG 操作只使用每个值的唯一实例，而不管该值出现了多少次。

● expression：精确数字或近似数字数据类型分类（bit 数据类型除外）的表达式，不允许使用聚集函数和子查询。

实现过程

（1）创建类 GradeUtil，在该类中定义查询数据库的方法。首先定义连接数据库的方法getConn()，具体代码读者可参考光盘中的源程序，这里不再赘述。

（2）在该类中定义 getResult()方法，该方法定义查询学生成绩表中各科成绩的平均值。代码如下：

```
public ResultSet getResult() {
  conn=getConn();
  //获取数据库连接
```

```
  float sum = 0;
  ResultSet set = null;
  try {
    Statement staement = conn.createStatement();
    String sql ="select avg(chinese)as chinese,avg(math)
as math,avg(english) as english"+",avg(physics) as
physics,avg(history)as history from tb_Grade"; //定义查找
各科成绩的平均值的SQL语句
    set=
staement.executeQuery(sql);
//执行查询语句返回查询结果集
  } catch (Exception e) {
    e.printStackTrace();
  }
  return set;
}
```

举一反三

根据本实例，读者可以实现以下功能。

计算每个班的平均分。

将各科平均分降序排列。

实例200 利用MIN函数求数据表中的最小值

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\05\Ex05_200

实例说明

利用MIN可以求数值表达式中的最小值。本实例使用MIN函数求学生信息表中年纪最小的学生信息，将其信息在控制台上输出。学生成绩表中的原有数据如图5.27所示。

本实例将年龄最小的学生信息在控制台上输出的运行效果如图5.28所示。

id	name	sex	age	specialty
01D01	陈宁	女	20	计算机软件
01D02	王建	男	21	市场营销
01D03	田子	男	19	国际金融
01D04	张红	女	23	英语
01D05	周丹	女	22	市场营销
01D06	刘静	女	21	英语
01D07	李丽	女	20	英文

图5.27 学生成绩表中的原有数据

图5.28 实例运行效果

技术要点

MIN聚集函数主要用于返回在某一集合上对数值表达式求得的最小值。该函数的语法格式如下：

MIN([ALL|DISTINCT] expression)

参数说明

● ALL：该参数是默认设置，如果没有该参数，将对所有的值进行聚集函数运算。

● DISTINCT：指定每个唯一值都被考虑。DISTINCT 对MIN 无意义。

● expression：该参数为表达式，由常量、列名、函数以及算术运算符、按位运算符和字符串运算符任意组合而成。

实现过程

（1）定义类 StudentUtil，用于查询学生表中年龄最小的学生信息，在该类中定义连接数据库的方法getConn()。具体代码读者可参考光盘中的源程序，这里不再赘述。

（2）在该类中定义查询数据表中年龄最小的学生信息的方法getMINStudent()。具体代码如下：

```
    public List getMINStudent() {
      List list=new
  ArrayList();                                          //定义用
  于保存返回值的List集合
      conn=getConn();
      //获取数据库连接
      try {
        Statement staement = conn.createStatement();
        String sql ="select name,sex,age,specialty from
    tb_student where age = (select min(age) from
    tb_student)";
      //定义查询学生表中年龄最小的学生信息SQL语句
      ResultSet set=
    staement.executeQuery(sql);                         //执行查
    询语句返回查询结果集
      while (set.next())
    {                                                   //循环遍历查询结
    果集
        Student student=new
      Student();                                        //定义学生对象
        student.setName(set.getString("name"));
              //设置学生对象的姓名属性
```

```
            student.setSex(set.getString("sex"));
            student.setAge(set.getInt("age"));
            student.setSpecialty(set.getString("specialty"));
            list.add(student);
                //将学生对象添加到集合中
          }
        } catch (Exception e) {
          e.printStackTrace();
        }
        return
    list;                                                    //返回
    List集合
      }
```

举一反三

根据本实例，读者可以开发以下程序。

求出数据表中的最小值。

求出数据表中的最大值。

实例201 利用MAX函数求数据表中的最大值

这是一个可以启发思维的实例

实例位置：光盘\mingrisoft\05\Ex05_201

实例说明

利用MAX函数可以求出数据表中指定列的最大值。本实例使用MAX函数求出订单表中签订的最大金额的客户信息。本实例操作的订单表中的数据如图5.29所示。

本实例将签订的最大金额订单的客户信息在控制台上显示，实例运行效果如图5.30所示。

id	clientName	clientArea	clientDate	clientMoney	visePerson
1	李先生	华北地区	2010-6-8	25000	王玉
2	葛先生	东北地区	2010-7-12	100000	张三
3	陈小姐	内蒙古地区	2010-5-30	19800	王丽
4	刘小姐	京津地区	2010-7-12	56220	陈双
5	王先生	长江三角洲	2010-7-1	20400	张三
6	李先生	华北地区	2010-5-20	12000	王玉
7	陈小姐	内蒙古地区	2010-3-16	9000	王丽

图5.29 订单表中的数据

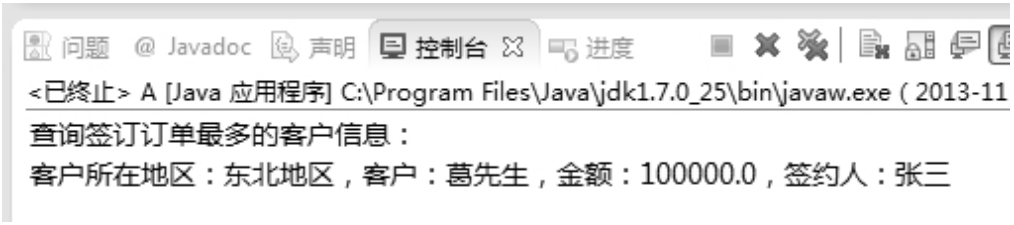


图5.30 实例运行效果

技术要点

MAX 聚集函数主要用于返回在某一集合上对数值表达式求得的最大值。该函数的语法格式如下：

MAX([ALL|DISTINCT] expression)

参数说明

● ALL：该参数是默认设置，如果没有该参数，将对所有的值进行聚集函数运算。

● DISTINCT：指定每个唯一值都被考虑。DISTINCT 对MAX 无意义。

● expression：该参数为表达式，由常量、列名、函数以及算术运算符、按位运算符和字符串运算符任意组合而成。

实现过程

（1）定义类OrderFormUtil，用于查询订单表中签订订单最大的客户信息，在该类中定义连接数据库的方法getConn()。具体代码读者可参考光盘中的源程序，这里不再赘述。

（2）在该类中定义查询数据表中签订的最大金额订单的客户信息方法 getMAXOrder()。具体代码如下：

```
public List getMAXOrder() {
  List list=new
ArrayList();                                    //定义用
于保存返回值的List集合
  conn=getConn();
  //获取数据库连接
  try {
    Statement staement = conn.createStatement();
    String sql ="select
clientName,clientArea,clientMoney,visePerson from
tb_orderForm "+"where clientMoney =(select
max(clientMoney) from tb_orderForm)";        //定义查询签
订的订单金额最大的信息
    ResultSet set=
staement.executeQuery(sql);                      //执行查
询语句返回查询结果集
    while (set.next())
{                                          //循环遍历查询结
果集
      OrderForm orderForm = new OrderForm();
      orderForm.setClientArea(set.getString("clientArea")
);
      orderForm.setVisePerson(set.getString("visePerson")
);
```

```
            orderForm.setClientMoney(set.getFloat("clientMoney"
        ));
            orderForm.setClientName(set.getString("clientName")
        );
            list.add(orderForm);
          }
        } catch (Exception e) {
          e.printStackTrace();
        }
        return
    list;                                                    //返回
    List集合
      }
```

举一反三

根据本实例，读者可以实现以下功能。

求出数据表中的最大值。

求出最大值与最小值的平均值。

实例202 利用COUNT函数求销售额大于某值的图书种类

这是一个可以美化界面的实例

实例位置：光盘\mingrisoft\05\Ex05_202

实例说明

COUNT函数用于返回表达式中值的个数，COUNT函数操作的表达式通常是数据表中字段的名称。本实例使用COUNT函数计算图书表中日销

售额大于400的图书种类。本实例实现操作图书表，该表中的数据如图5.31所示。

本实例查询图书表中日销量超过400的图书种类的运行效果如图5.32所示。

bookId	bookName	total	sellDate
001	Java从入门到精通	360	2010/7/10
002	Java范例完全自学手册	102	2010/6/30
003	Java开发实战宝典	350	2010/7/2
004	VB从入门到精通	350	2010/7/2
005	Java典型模块大全	460	2010/7/2
006	Java视频学	560	2010/2/20
007	Java Web视频学	520	2010/5/15
008	Struts完全手册	270	2010/6/30

图5.31 图书表中的数据

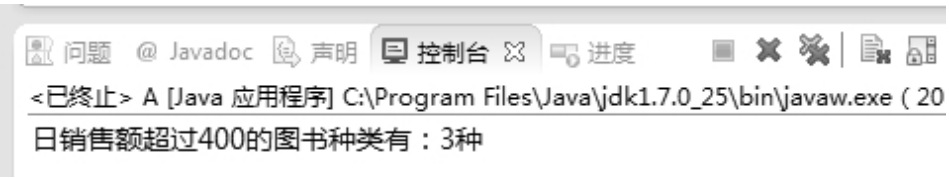


图5.32 实例运行效果

技术要点

COUNT函数能够计算在一个记录中表达式的数目，即数据条数。一般地，COUNT函数主要用于查询结果中的数据条数，因此通常以通配符*作为COUNT函数的参数。事实上，COUNT函数也是唯一允许使用通配符作为参数的聚集函数。该函数的语法格式如下：

COUNT({[ALL | DISTINCT]expression} | *)

参数说明

● ALL：该参数是默认设置，如果没有该参数，SQLServer对所有的值进行聚集函数运算。

● DISTINCT：该参数指定COUNT 返回唯一非空值的数量。

● expression：该参数为表达式，它的类型可以是除uniqueidentifier、text、image 或ntext之外的任何类型。

● *：指定应该计算所有行以返回表中行的总数，COUNT(*)不需要任何参数，而且不能与DISTINCT一起使用。COUNT(*)不需要

expression参数，因为根据定义，该函数不使用有关任何特定列的信息。COUNT(*)返回指定表中行的数量而不消除副本，它对每行分别进行计数，包括含有空值的行。

实现过程

（1）定义工具类 BookSellUtil，在该类中定义查询图书表中数据的方法。首先定义连接数据库的方法getConn()，具体代码读者可参考光盘中的源程序，这里不再赘述。

（2）在该类中定义查询图书表中日销售额在400以上的图书种类的方法getMAXOrder()。具体代码如下：

```
public int getMAXOrder() {
  List list=new
ArrayList();                                    //定义用
于保存返回值的List集合
  conn=getConn();
  //获取数据库连接
  int count = 0;
  try {
    Statement staement = conn.createStatement();
    //查询日销售额在400以上的图书种类
    String sql ="select count(bookName) as bookType from
tb_bookSell where total >400";
    ResultSet set=
staement.executeQuery(sql);                     //执行查
询语句返回查询结果集
    while (set.next())
{                                               //循环遍历查询结
果集
```

```
            count = set.getInt("bookType");
          }
        } catch (Exception e) {
          e.printStackTrace();
        }
        return
    count;                                                    //返
    回List集合
      }
```

举一反三

根据本实例，读者可以实现以下功能。

用COUNT函数求所销售图书的种类数。

用COUNT函数求销售额小于某值的图书。

实例203 查询编程词典6月份的销售量

这是一个可以美化界面的实例

实例位置：光盘\mingrisoft\05\Ex05_203

实例说明

在SQL Server 数据库中提供了按月查询数据的函数MONTH。使用该函数对数据进行按月份统计十分方便。本实例实现的是对编程词典销量表中的数据进行按月份统计，并将6月份的总销售额输出。本实例操作的是编程词典销量表，表中的数据如图5.33所示。

本实例实现查询编程词典销量表6月份的销量统计的运行效果如图5.34所示。

id	bccdName	bccdCount	bccdPrice	bccdDate
1	java编程词典	300	59	2010/6/20
2	VB编程词典	550	59	2010/6/15
3	VC编程词典	290	59	2010/5/21
4	C#编程词典	150	59	2010/4/25
5	java编程词典	165	59	2010/6/1
6	.net编程词典	194	59	2010/5/13
7	VB编程词典	125	59	2010/6/21

图5.33 编程词典销量表中的数据

图5.34 实例运行效果

技术要点

要实现按月查询数据可以使用日期函数MONTH()。该函数的语法格式如下：

MONTH(date)

参数说明

● date：返回date 或smalldatetime 值或日期格式字符串的表达式。仅对1753 年1 月1 日后的日期使用datetime数据类型。其返回值的数据类型为int。

● MONTH：该函数能够将日期时间表达式date 中的月份返回，返回的月份是以数值1～12来表示，1代表1月，2代表2月，依次类推。

实现过程

（1）定义工具类 BccdSell，在该类中定义查询数据的方法。首先定义连接数据库的方法getConn()，具体代码读者可参考光盘中的源程序，这里不再赘述。

（2）在该类中定义查询编程词典销量表中6月份的总销售额的方法getBccdSell()。该方法的具体代码如下：

```
public int getBccdSell() {
  List list=new
ArrayList();                                    //定义用
```

```
于保存返回值的List集合
    conn=getConn();
    //获取数据库连接
    int count = 0;
    try {
      Statement staement = conn.createStatement();
      //查询编程词典销量表中6月份的总销量
      String sql ="select sum(bccdCount) from tb_bccdSell
  where month(bccdDate) = 6";
      ResultSet set=
  staement.executeQuery(sql);                    //执行查
  询语句返回查询结果集
      while (set.next())
  {                                              //循环遍历查询结
  果集
        count = set.getInt(1);
      }
    } catch (Exception e) {
      e.printStackTrace();
    }
    return
count;                                                  //返
回List集合
  }
```

举一反三

根据本实例，读者可以实现以下功能。

对6月份的销量进行降序排序。

查询出最畅销的图书种类。

实例204 查询与张静同一天入司的员工信息

这是一个可以美化界面的实例

实例位置：光盘\mingrisoft\05\Ex05_204

实例说明

查询与某人同一天入司的员工信息，需要使用CONVERT函数与like关键字。CONVERT函数可以对指定的日期进行格式化。like 关键字可以实现对数据的模糊查询。本实例实现查询员工表中和张静同一天入司的员工。本实例实现查询员工表，该表中的数据如图5.35所示。

运行本实例，结果如图5.36所示。

id	name	sex	dept	ddate
1	张静	男	Java部	2008/5/20
2	张三	男	质量部	2007/10/7
3	王经理	女	VB部	2009/5/6
4	李丽	女	Java部	2008/5/20
5	刘娜	女	VC部	2008/9/7
6	玲玲	女	C#部	2008/5/20

图5.35 员工表中的数据

图5.36 实例运行效果

技术要点

本实例在对日期型数值进行模糊查询时，应用了CONVERT函数，该函数为日期格式化函数，可将日期转化成“yyyy-mm-dd”等格式。该函数的语法格式如下：

CONVERT(date_type[(length)] , expression , style)

参数说明

- date_type：要转化的数据类型。

● expression：DATETIME 类型的数据。

● style：指定转化形式。其取值不同，对应的日期、时间格式也不同。表 5.6 为 style 取不同值所对应的日期、时间格式。

表5.6 style不同取值对应的日期、时间格式

style	格　　式	示　　例
0	mon dd yyyy hh:miAM/PM	Jun 22 2009 08:30 AM
1	mm/dd/yy	08/28/09
2	yy.mm.dd	09.04.24
3	dd/mm/yy	02/12/09
4	dd.mm.yy	22.06.09
5	dd-mm-yy	22-07-09

续表

style	格　　式	示　　例
6	dd mon yy	21 Jun 09
7	Mon dd yy	Jun 21 09
8	hh:mm:ss	11:30:21
9	mon dd yyyy hh:mi:ss:mmmmAM/PM	Jun 10 2009 08:25:25:048 AM
10	mm-dd-yy	07-25-09
11	yy/mm/dd	09/07/24
12	yymmdd	090712`
13	dd mon yyyy hh:mm:ss:mmm	21 Jun 2009 09:20:22:045
14	hh:mi:ss:mmm	09:31:22:048
20	yyyy-mm-dd hh:mi:ss	2009-07-25 08:25:20

本实例实现在select查询语句中使用like关键字。在查询语句中like关键字位于WHERE子句中。like关键字可以与NOT运算符组合使用，在查询语句中使用like关键字的语法格式如下：

```
SELECT [DISTIN|UNIQUE](*,columname[AS alias],…)
FROM table
WHERE columname LIKE '[ _|% ]value'
```

参数说明

● columname：要进行查询匹配的字段。

● value：要进行匹配的字符串。

● 匹配符“_”表示任意单个字符，该运算符只能匹配一个字符。匹配符“%”可以匹配 0个或更多字符的任意长度的字符串，且不计较字符的多少。

实现过程

（1）定义类BccdSell，在该类中定义连接数据库的方法getConn()。具体代码读者可参考光盘中的源程序，这里不再赘述。

（2）在该类中定义 getBccdSell()方法，用于查询与张静同一天入司的员工信息。具体代码如下：

```
public List getBccdSell() {
  List list=new
ArrayList();                                    //定义用
于保存返回值的List集合
  conn=getConn();
  //获取数据库连接
  try {
    Statement staement = conn.createStatement();
    //定义查询数据的SQL语句
    String sql ="select * from tb_emp where
convert(varchar(10),ddate,21) "+"like (select
convert(varchar(10),ddate,21) from tb_emp where name =
'张静')";
    ResultSet set=
staement.executeQuery(sql);                     //执行查
询语句返回查询结果集
    while (set.next())
{                                               //循环遍历查询结
果集
```

```
            Emp emp = new Emp();
            emp.setId(set.getInt(1));
            emp.setName(set.getString(2));
            emp.setSex(set.getString("sex"));
            emp.setDdate(set.getString("ddate"));
            emp.setDept(set.getString("dept"));
            list.add(emp);
          }
        } catch (Exception e) {
          e.printStackTrace();
        }
        return
    list;                                                    //返回
    List集合
      }
```

举一反三

根据本实例，读者可以开发以下程序。

指定时间查询所有入司的员工。

查询所有入司女员工的人数。

实例205 使用between进行区间查询

这是一个可以提高分析能力的实例

实例位置: 光盘\mingrisoft\05\Ex05_205

实例说明

区间查询是指查询某两个值之间的数据。在SQL Server 数据库中要进行区间查询可以采用两种方式，一种是分别使用符号“>”与

“<”，另一种就是使用between…and关键字。这两种查询方式都要写在where子句中，作为查询条件。本实例实现使用between关键字查询英语成绩在80～90之间的学生信息。本实例实现操作学生成绩表，该表中的数据如图5.37所示。

运行本实例，将英语成绩在80～90之间的学生姓名与英文成绩在控制台上输出，结果如图5.38所示。

id	name	chinese	math	english	physics	history
1	张三	85	92	89	94	92
2	李玉	85	86	98	87	75
3	王丽	78	86	78	95	57
4	赵四	85	95	74	95	74
5	陈雷	89	68	87	89	87

图5.37 学生成绩表中的数据

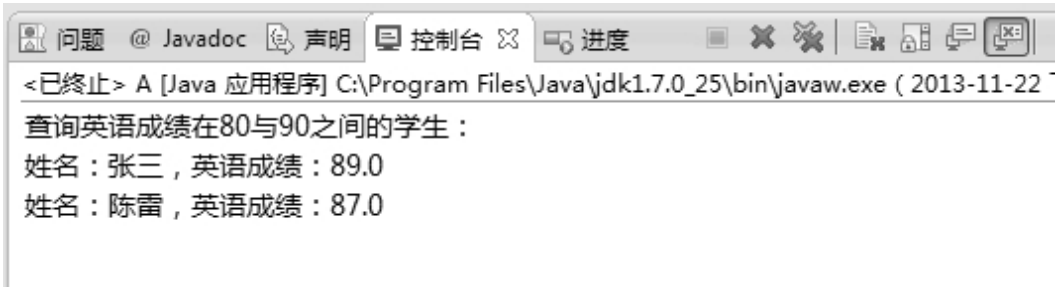


图5.38 实例运行结果

技术要点

本实例使用between…and运算符进行区间查询，其语法格式如下：

expression[NOT]between expression1 and expression2

参数说明

● expression：用来在由 expression1 和 expression2 定义的范围内进行测试的表达式。expression必须与expression1和expression2具有相同的数据类型。

● NOT：指定谓词的结果被取反。

● and 占位符：表示expression 应该处于expression1 和expression2 指定的范围内。

如果 expression 的值大于或等于 expression1 的值，并且小于或等于 expression2 的值，则between 返回 true；如果 expression 的值小于 expression1 的值或者大于 expression2 的值，则between 返回false。

实现过程

（1）创建类 BetweenUtil，在该类中定义查询数据的方法。首先定义连接数据库的方法getConn()，该方法以Connection对象作为返回值。具体代码读者可参考光盘中的源程序，这里不再赘述。

（2）在该类中定义查询英文成绩在80～90之间的学生姓名和英文成绩的方法getGrade()，该方法以List作为返回值。具体代码如下：

```
public List getGrade() {
  List list=new
ArrayList();                                    //定义用
于保存返回值的List集合
  conn=getConn();
  //获取数据库连接
  try {
    Statement staement = conn.createStatement();
    //定义查询英文成绩在80～90之间的学生信息SQL语句
    String sql ="select name,english from tb_Grade where
english between 80 and 90";
    ResultSet set=
staement.executeQuery(sql);                     //执行查
询语句返回查询结果集
    while (set.next())
{                                               //循环遍历查询结
果集
```

```
            Gradegrade=newGrade();
          //创建与学生成绩表对应的Grade对象
            grade.setName(set.getString(1));
            grade.setEnglish(set.getFloat(2));
            list.add(grade);
            //将Grade对象添加到集合中
          }
        } catch (Exception e) {
          e.printStackTrace();
        }
        return
    list;                                                    //返回
    查询集合
      }
```

举一反三

根据本实例，读者可以开发以下程序。

使用“>”与“<”组合实现区间查询。

使用 between 进行区间查询。

实例206 使用IN谓词查询某几个时间的数据

这是一个可以启发思维的实例

实例位置：光盘\mingrisoft\05\Ex05_206

实例说明

IN谓词可限定查询结果返回的范围。本实例实现的是查询在2010/6/20和2010/6/21都有销售记录的编程词典销售信息。实例实现的是操作编程词典销量表，该表中的数据如图5.39所示。

运行本实例，结果如图5.40所示。

	id	bccdName	bccdCount	bccdPrice	bccdDate
	1	java编程词典	300	59	2010/6/20
	2	VB编程词典	550	59	2010/6/15
	3	VC编程词典	290	59	2010/5/21
	4	C#编程词典	150	59	2010/4/25
	5	java编程词典	165	59	2010/6/1
	6	.net编程词典	194	59	2010/5/13
	7	VB编程词典	125	59	2010/6/21

图5.39 编程词典销量表中的数据

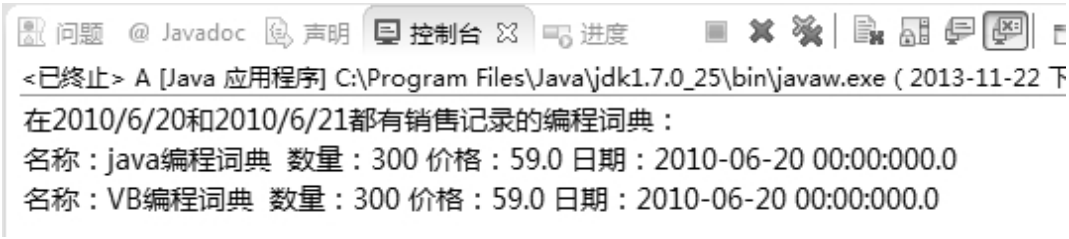


图5.40 实例运行效果

技术要点

IN 运算符返回与列出的值中指定某一个值相等的记录。IN 运算符列表中的项目必须使用逗号分隔，并且必须将整个列表都包括在括号中。

利用 IN 运算符查询某几个时间段的数据。例如，查询与王一和王二同一天出生的员工信息，示例代码如下：

```
select * from tb_emp
where 出生日期 in((select 出生日期 from tb_emp where 姓名 = '王一'), (select 出生日期 from tb_emp where 姓名='王二'))
```

在日期查询时还可以通过聚集函数对其进行查询，但值得注意的是排在后面的时间大于排在前面的时间，如2008-12-21大于2008-01-01。例如，查询最后出版与最先出版的图书信息，示例代码如下：

```
select * from tb_booksell
where 出版日期 in ((select max(出版日期) from tb_booksell), (select min(出版日期) from tb_booksell))
```

实现过程

（1）在项目中创建类GetMessage，用于定义查询数据方法，在该类中首先定义连接数据库的方法getConn()，该方法以Connection对象作为返回值，具体代码读者可参考光盘中的源程序。

（2）在该类中定义 getBccdSell()方法，用于在编程词典销量表中查询数据，并将查询结果以List集合返回。具体代码如下：

```
public List getBccdSell() {
  List list=new
ArrayList();                                    //定义用于保存
返回值的List集合
  conn=getConn();                                          //
获取数据库连接
  try {
    Statement staement = conn.createStatement();
    //定义查询数据的SQL语句
    String sql ="select * from tb_bccdSell where bccdDate
in ('2010/6/20','2010/6/21')";
    ResultSet set=
staement.executeQuery(sql);                  //执行查询语句
返回查询结果集
    while (set.next())
{                                          //循环遍历查询结果集
      Bccd bccd=new Bccd();                                //定
义与数据对应的JavaBean对象
      bccd.setId(set.getInt(1));
      //设置对象属性
      bccd.setBccdName(set.getString(2));
      bccd.setBccdCount(set.getInt(3));
```

```
            bccd.setBccdPrice(set.getFloat(4));
            bccd.setBccdDate(set.getString(5));
            list.add(bccd);                                    //
        向集合中添加对象
          }
        } catch (Exception e) {
          e.printStackTrace();
        }
        return
    list;                                                  //返回List
    集合
      }
```

举一反三

根据本实例，读者可以实现以下程序。

使用NOT IN 查询数据。

使用ALL和AND查询数据。

实例207 日期查询中避免千年虫问题

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\05\Ex05_207

实例说明

在日期中有的使用的是4位的年份，有的使用的是两位的年份。在使用两位年份时，最需要注意的就是千年虫问题，如 1904 年与 2004 年使用两位纪年就都表示为 04。在 SQL Server中可以通过设置两位年份的间隔避免千年虫问题。本实例实现按某学生的出生日期对该学生进行查询，实例运行效果如图5.41所示。

图5.41 实例运行效果

技术要点

避免出现千年虫问题，在SQL Server数据库中可以通过设置两位年份的支持范围来避免这一问题，步骤如下。

（1）在SQL Server企业管理器中，右击SQL Server，在弹出的快捷菜单中选择“属性”命令。

（2）在弹出的“SQL Server 属性”对话框中选择“服务器设置”选项卡，设置“两位数年份支持”栏中的数值微调器，再单击“确定”按钮即可，如图5.42所示。

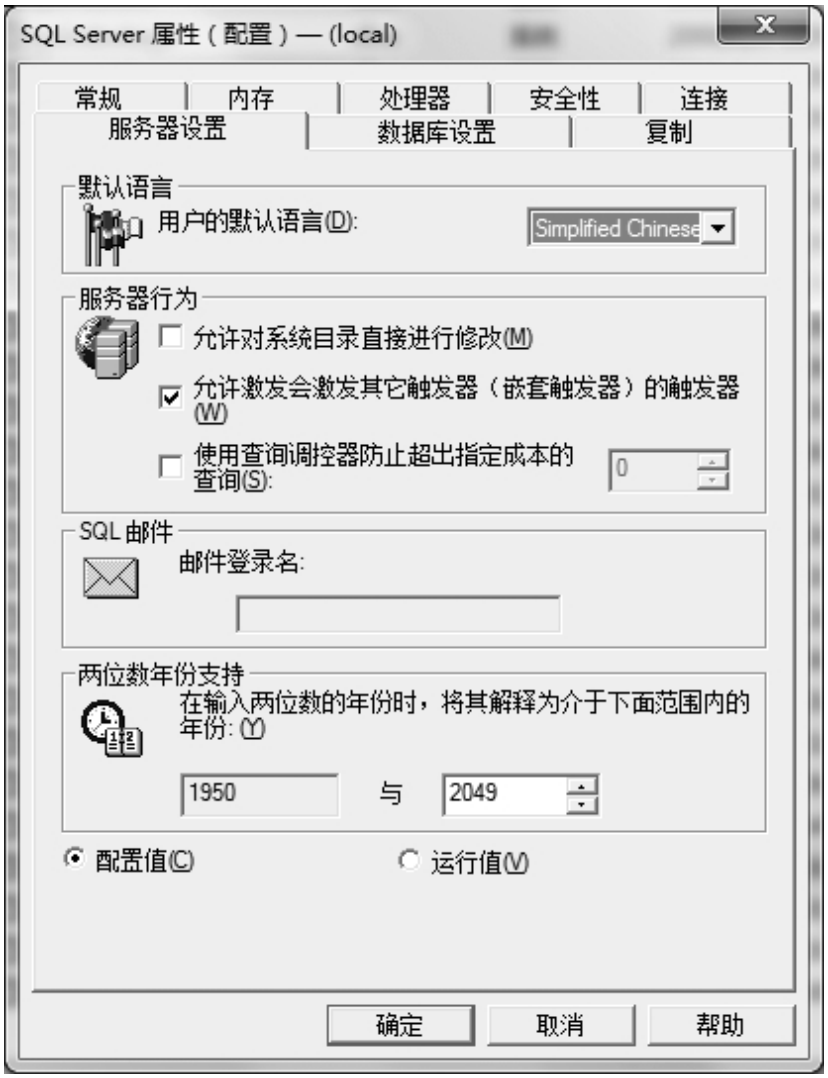


图5.42 设置两位年份的支持范围

实现过程

（1）在项目中创建窗体类 SelectDateFrame，该类继承自 JFrame 类，实现窗体类，向该窗体中添加标签、下拉列表、按钮与表格控件实现窗体布局。

（2）在项目中创建工具类 SelectDateUtil，在该类中定义按照出生日期查询学生信息的方法getStuUseDate()，该方法包含一个String类型的参数，用于指定学生出生日期。具体代码如下：

```
public List getStuUseDate(String sDate) {
```

```
    List list=new
ArrayList();                                    //定义用于保存
返回值的List集合
    conn=getConn();                                          //
获取数据库连接
    try {
      Statement staement = conn.createStatement();
      String sql ="select * from tb_StuInfo where sBrithday
  = '"+sDate+"'";
      ResultSet set=
  staement.executeQuery(sql);                  //执行查询语句
  返回查询结果集
      while (set.next())
  {                                          //循环遍历查询结果集
        StuInfo stuInfo=newS
    tuInfo();                              //创建与数据表对应的
    JavaBean对象
        stuInfo.setId(set.getInt(1));
        //设置对象属性
        stuInfo.setsName(set.getString(2));
        stuInfo.setsSex(set.getString(3));
        stuInfo.setsBrithday(set.getString(4));
        stuInfo.setsSpeciality(set.getString(5));
        stuInfo.setsAddress(set.getString(6));
        list.add(stuInfo);
      //向集合中添加对象
      }
```

```
    } catch (Exception e) {
      e.printStackTrace();
    }
    return
list;                                                    // 返回
List集合
  }
```

举一反三

根据本实例，读者可以实现以下程序。

日期查询避免千年虫问题。

跨年月份的表示。

5.4 使用子查询

实例208 将子查询作为表达式

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\05\Ex05_208

实例说明

将子查询作为表达式，是指在SELECT语句指定查询元素时，仍然使用SELECT查询语句。这种查询方式很常见，是程序员必须掌握的一种查询技巧。本实例实现的是通过子查询查询学生成绩表中的学生英语成绩、英语平均成绩和每位同学的英语成绩与平均成绩的差额。实例运行效果如图5.43所示。

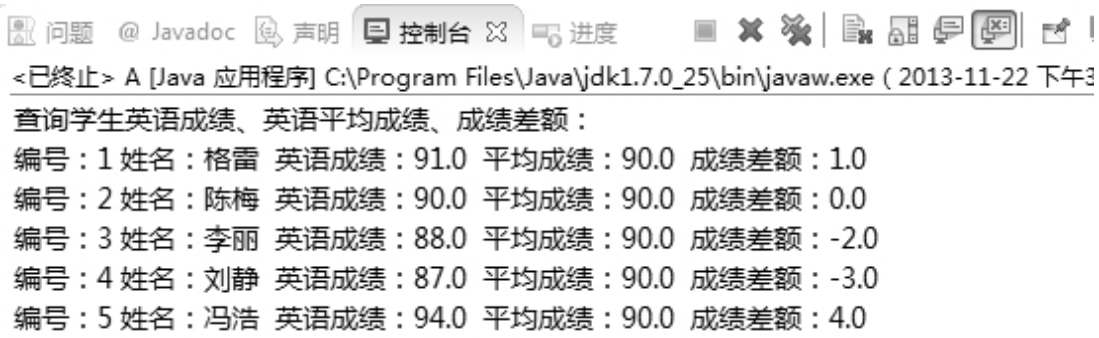


图5.43 实例运行效果

技术要点

本实例的子查询是应用在SELECT子句中的，将子查询应用在SELECT子句中，其查询结构就可以以表达式的形式出现。在应用子查询时有一些子查询的控制规则，了解了这些规则可以使读者对子查询的应用更加游刃有余。

（1）由比较运算符引入的内层查询SELECT列表或IN只包括一个表达式或列名。在外层语句的WHERE子句中命名的列必须能与查询SELECT列表中命名的列连接兼容。

（2）由不可更改的比较运算符引入的子查询（比较运算符后面不跟关键字ANY和ALL）不能包括GROUPBY 或HAVING 子句，除非预先确定了组或单个的值。

（3）由 EXISTS 引入的 SELECT 列表一般都由星号（*）组成，而不必指定具体的列名，也可以在嵌套子查询WHERE子句中限定行。对于EXISTS引入的子查询，SELECT列表规则和标准选择列表中的规则是一样的。

（4）子查询不能在内部处理它们的结果，也就是说，子查询不能包括ORDER BY 子句。可选择的DISTINCT关键字可以有效地给子查询结果排序，因为一些系统会通过首先给结果排序来消除重复记录。

实现过程

（1）在项目中创建类GetEnglishAvg，用于查询学生成绩信息。在该类中定义与数据库建立连接的方法，具体代码读者可参考光盘中的源程序，这里不再赘述。

（2）在该类中定义查询数据的方法getAvg()，该方法以List集合返回查询结果。具体代码如下：

```
public List getAvg(){
  List list=newArrayList<Grade>
();                                          //定义List集合对象
  conn=getConn();
      //获取与数据库的连接
  try {
    Statement statement=
conn.createStatement();                              //获取
```

Statement对象

```
    String sql ="select id,name,english,( select
avg(english) from tb_grade ) as avgEnglish,"+"(english-(
selectavg(english) from tb_grade )) as diffAvgEnglish
from tb_grade";        //定义查询语句
    ResultSet rest=
statement.executeQuery(sql);                              //
执行查询语句获取查询结果集
    while(rest.next())
{                                                  //循环遍历查
询结果集
      Grade grade=new
  Grade();                                        //定义与数据
  表对应的JavaBean对象
      grade.setId(rest.getInt(1));
                //设置对象属性
      grade.setName(rest.getString(2));
      grade.setEnglish(rest.getFloat(3));
      grade.setAvgEng(rest.getFloat(4));
      grade.setBalance(rest.getFloat(5));
      list.add(grade);
            //向集合中添加元素
    }
  } catch (Exception e) {
    e.printStackTrace();
  }
```

```
    return
list;
//返回查询结果
  }
```

举一反三

根据本实例，读者可以开发以下程序。

使用子查询进行多表联查。

满足子查询条件的查询。

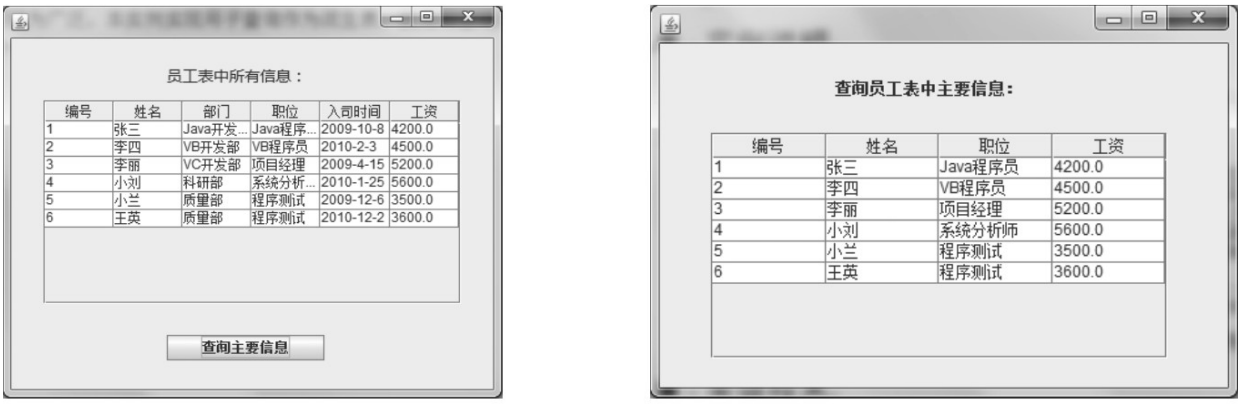
实例209 用子查询作为派生表

这是一个可以提高分析能力的实例

实例位置：光盘\mingrisoft\05\Ex05_209

实例说明

用子查询作为派生表应用较为广泛。本实例实现用子查询作为派生表，就是将查询结果集作为一个表使用。本实例在员工表中派生出一个含有员工编号、员工姓名、职位、工资的表，运行程序，单击“查询主要信息”按钮，即可显示新表的信息在表格中。实例运行效果如图5.44所示。



(a) (b)

图5.44 实例运行效果

技术要点

子查询是一个用于处理多表操作的附加方法。语法如下：

（SELECT [ALL | DISTINCT]<select item list>

FROM <table list>

[WHERE<search condition>]

[GROUP BY <group item list>

[HAVING <group by search conditoon>]]）

把子查询用作派生的表可以应用在很多方面，如下面几个示例。

将分组统计数据作为派生表（将销售单按商品名称统计分组后查询销售数量大于14的商品），代码如下：

SELECT * FROM (SELECT proname, COUNT(*) AS sl FROM xsd GROUP By proname) WHERE (sl > 14)

将过滤数据作为派生表（对商品销售表中销售数量前100名进行分组统计），代码如下：

SELECT sl,COUNT(*) FROM (SELECT TOP 100 FROM T_ZDxxb ORDER BY zdbh) GROUP BY sl

将过滤数据作为派生表（统计客户表中未结账客户的欠款金额），代码如下：

SELECT name,SUM(xsje) FROM (SELECT * FROM kh WHERE NOT jz) GROUP BY name

实现过程

（1）在项目中创建类FullMessage，该类继承自JFrame类，实现窗体类。该类用于实现显示员工表中的全部信息，该窗体中包含表格、按钮等控件。

（2）创建MostlyFrame类，该类继承自JFrame类，实现窗体类。在该类中定义表格控件，用于显示查询的信息。

（3）定义工具类DeriveTable，在该类中定义查询员工表中主要信息的方法getSubTable()，该方法以List集合对象作为返回值。具体代码如下：

```
public List getSubTable() {
  List list=new ArrayList<Emp>
();                                    //定义List集合对象
  conn=getConn();
  //获取与数据库的连接
  try {
    Statement statement=
conn.createStatement();                    //获取
Statement对象
    String sql ="select * from (select
id,eName,headship,laborage from tb_emp)tb";
    ResultSet rest=
statement.executeQuery(sql);                    //执行查
询语句获取查询结果集
    while (rest.next())
{                                          //循环遍历查询结
果集
      Emp emp=new
  Emp();                                    //定义与数据表
  对应的JavaBean对象
      emp.setId(rest.getInt(1));
          //设置对象属性
      emp.setName(rest.getString(2));
      emp.setHeadship(rest.getString(3));
```

```
            emp.setLaborage(rest.getFloat(4));
            list.add(emp);
            //将对象添加到集合中
          }
        } catch (Exception e) {
          e.printStackTrace();
        }
        return
    list;                                                    //返回
    查询结果
      }
```

举一反三

根据本实例，读者可以开发以下程序。

为派生表起别名。

用子查询做派生表。

实例210 通过子查询关联数据

这是一个可以启发思维的实例

实例位置：光盘\mingrisoft\05\Ex05_210

实例说明

本实例利用EXISTS谓词引入到子查询将学生表和学生成绩表关联起来，查询英文成绩大于90分的学生姓名、学院、地址等信息。本实例操作两张表，分别为学生表与学生成绩表，这两张表中的数据如图5.45所示。

id	name	college	speciality	address
1	格雷	计算机学院	计算机科学与技术	吉林省长春市
2	陈梅	外国语学院	俄语	山东淄博市
3	李丽	管理学院	国际贸易	内蒙古包头市
4	刘静	理工大学	化工材料	江苏省海门市
5	封号	机械学院	汽车制造	北京市

id	name	sex	english	chinese	math
1	格雷	男	91	92	84
2	陈梅	女	90	88	82
3	李丽	女	88	90	91
4	刘静	女	87	86	90
5	冯洁	男	94	90	89

图5.45 学生表与学生成绩表数据

本实例实现关联两张表查询，将查询结果在控制台上输出。实例运行效果如图5.46所示。

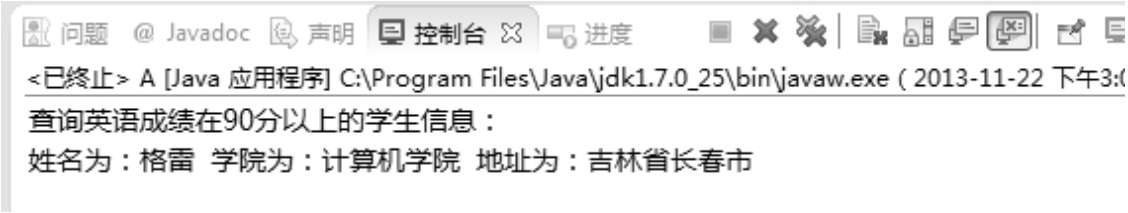


图5.46 实例运行效果

技术要点

只要子查询返回一个真值或假值，并只考虑是否满足谓词条件，数据内容本身并不重要，在这种情况下，可用 EXISTS 谓词来定义子查询。如果子查询返回一行或多行，EXISTS 谓词为真，否则为假。要使EXISTS谓词有用，应该在子查询中建立查询条件以匹配子查询连接起来的两个表中的值。语法格式如下：

EXISTS subquery

参数说明

subquery：一个受限的SQL语句（不允许有COMPUTE子句和INTO关键字）。

其结果类型为boolean，如果子查询包含行，则返回true。

实现过程

（1）在项目中创建类 FindMessage，实现查询数据的方法。在该类中定义连接数据库的方法getConn()，具体代码读者可参考光盘中的源程序，这里不再赘述。

（2）在该类中定义查询英文成绩在90分以上的学生信息的方法getMessage()，该方法的返回值为ResultSet。具体代码如下：

```
public ResultSet getMessage() {
  ResultSet rest = null;
```

```
    conn=getConn();
        //获取与数据库的连接
    try {
      Statement statement=
  conn.createStatement();                          //获取
  Statement对象
        //定义查询语句
      String sql ="select name,college,address from
  tb_student I where exists "+"(select name from tb_grade M
  where M.name=I.name and english >90)";
      rest=
  statement.executeQuery(sql);
      //执行查询语句获取查询结果集
    } catch (Exception e) {
      e.printStackTrace();
    }
    return
rest;
//返回查询结果
  }
```

举一反三

根据本实例，读者可以开发以下程序。

进行丢失精度的类型转换。

通过子查询关联表中数据。

实例211 使用IN谓词限定查询范围

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\05\Ex05_211

实例说明

IN谓词允许测试实际值或表达式的值。当使用IN谓词来引入子查询时，告诉数据库管理系统执行一种所谓的“子查询成员测试”。IN语句可以替代Where子句中的表达式用以限定查询的范围。本实例使用IN 谓词限定要查询的员工工资范围。运行本实例，可以输入要查询的员工工资范围，结果如图5.47所示。

图5.47 实例运行效果

技术要点

IN谓词用于确定给定的值是否与子查询或列表中的值相匹配，常用于引入子查询。语法格式如下：

```
test_expression [ NOT ] IN
  (
    subquery
    | expression [ ,...n ]
  )
```

参数说明

● test_expression：任何有效的SQL 表达式。

● subquery：包含某列结果集的子查询，该列必须与test_expression 有相同的数据类型。

● expression [,...n]：一个表达式列表，用来测试是否匹配。所有的表达式必须和 test_expression具有相同的数据类型。

如果 test_expression 与 subquery 返回的任何值相等，或与逗号分隔的列表中的任何expression相等，那么结果值就为true，否则结果值为false。

实现过程

（1）在项目中创建类ConditionFrame，该类继承自JFrame类，实现窗体类。

（2）向窗体中添加控件，实现窗体布局，该窗体中的主要控件及说明如表5.7所示。

表5.7 窗体中的主要控件及说明

控 件 类 型	控 件 命 名	控 件 用 途
JTextField	labTextField1	限制查询工资范围的文本框控件
	labTextField2	限定查询工资范围的文本框控件
JTable	table	显示查询结果的表格控件
JButton	findButton	按钮控件

（3）创建ConditionUseIn类，在该类中定义获取查询结果的方法getSubTable()，该方法有两个int类型的参数，分别用于指定两个查询的工资范围。具体代码如下：

```
public List getSubTable(int lab1,int lab2) {
  List list=newArrayList<Emp>
();                                    //定义List集合对象
  conn=getConn();
  //获取与数据库的连接
```

```
  try {
    Statement statement=
conn.createStatement();                    //获取
Statement对象
    String sql ="select * from tb_emp where laborage in
(select laborage from tb_emp "+"where laborage
between"+lab1+" and "+lab2+")";        //定义查询语句
    ResultSet rest=
statement.executeQuery(sql);                    //执行查
询语句获取查询结果集
    while (rest.next())
{                                          //循环遍历查询结
果集
      Emp emp=newEmp();                                  //
  定义与数据表对应的JavaBean对象
      emp.setId(rest.getInt(1));
          //设置对象属性
      emp.setName(rest.getString(2));
      emp.setDept(rest.getString(3));
      emp.setHeadship(rest.getString(4));
      emp.setJoinDate(rest.getString(5));
      emp.setLaborage(rest.getFloat(6));
      list.add(emp);
      //将对象添加到集合中
    }
  } catch (Exception e) {
    e.printStackTrace();
```

```
    }
    return
list;                                                    //返回
查询结果
  }
```

举一反三

根据本实例，读者可以开发以下程序。

IN谓词来引入子查询。

switch语句限定查询范围。

实例212 使用NOT IN子查询实现差集运算

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\05\Ex05_212

实例说明

实例212 中为大家介绍了使用IN 关键字进行查询，下面为大家介绍使用NOT IN 组合查询实现两张表中数据的差集运算。本实例实现查询没有成绩记录的学生信息，实例运行效果如图5.48所示。

图5.48 实例运行效果

技术要点

带NOT IN 谓词的查询语法格式如下：

WHERE 查询表达式 NOT IN 子查询

NOT IN 和IN 查询过程相似，读者可参考实例211。

实现过程

（1）在项目中创建类SelectNotIn，在该类中首先定义连接数据库的方法getConn()，该方法以Connection对象作为返回值，具体代码读者可参考光盘中的源程序，这里不再赘述。

（2）在该类中定义getNotIn()方法，该方法用于查询在学生成绩表中不存在的学生信息。具体代码如下：

```
public List getNotIn() {
  List list=new
ArrayList();                                    //定义用
于保存返回值的List集合
  conn=getConn();
  //获取数据库连接
  try {
    Statement staement = conn.createStatement();
    //定义查询数据的SQL语句
    String sql ="select * from tb_student where name not
in (select name from tb_grade)";
    ResultSet set=
staement.executeQuery(sql);                     //执行查
询语句返回查询结果集
    while (set.next())
{                                               //循环遍历查询结
果集
      Student student=new
Student();                                      //定义与数据库对应
的JavaBean对象
      student.setId(set.getInt(1));
          //设置对象属性
```

```
            student.setName(set.getString(2));
            student.setCollege(set.getString(3));
            student.setSpeciality(set.getString(4));
            student.setAddress(set.getString(5));
            list.add(student);
              //将对象添加到集合中
          }
        } catch (Exception e) {
          e.printStackTrace();
        }
        return
    list;                                                    //返回
    List集合
      }
```

举一反三

根据本实例，读者可以实现以下功能。

使用NOT IN 进行数据查询。

对查询后的数据进行有序排列。

实例213 使用NOT IN子查询实现反向查询

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\05\Ex05_213

实例说明

谓词IN可以查询在某范围内的数据。该谓词与谓词NOT一起使用，可以查询不在指定范围内的数据。本实例实现操作学生信息表，该表中的数据如图5.49所示。

本实例使用NOT IN组合查询，实现查询与格雷不在同一学院的学生信息，运行效果如图5.50所示。

id	name	college	speciality	address
1	格雷	计算机学院	计算机科学与技术	吉林省长春市
2	陈梅	外国语学院	俄语	山东淄博市
3	李丽	管理学院	国际贸易	内蒙古包头市
4	刘静	理工大学	化工材料	江苏省海门市
5	封号	机械学院	汽车制造	北京市

图5.49 学生信息表中的数据

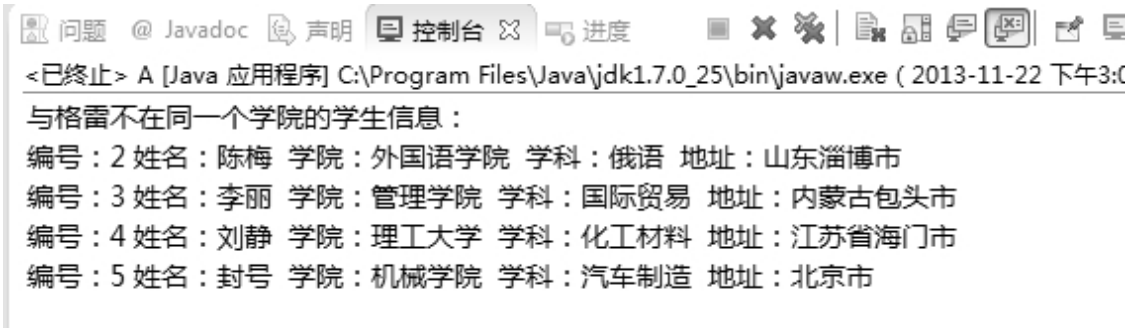


图5.50 实例运行效果

技术要点

NOT与IN两个关键字连用表示不在列表范围内的数据。查询表达式可以是常量或列名，而列表可以是一组常量，更多情况下子查询将列表值放在圆括号内。

实现过程

（1）在项目中创建类UseNotIn，用于实现定义查询数据库的方法，在该类中定义连接数据库的方法getConn()，该方法返回Connection对象，具体代码读者可参考光盘中的源程序，这里不再赘述。

（2）在该类中定义查询数据方法getSubTable()，该方法以List集合对象作为返回值。具体代码如下：

```
public List getSubTable() {
  List list=new ArrayList<Student>
();                              //定义List集合对象
```

```
    conn=getConn();                                          //
获取与数据库的连接
    try {
      Statement statement=
  conn.createStatement();              //获取Statement对象
      String sql ="select * from tb_student where college
  not in (select college from tb_student where name ='格雷'
  )";
      //定义查询语句
      ResultSet rest=
  statement.executeQuery(sql);                  //执行查询语
  句获取查询结果集
      while (rest.next())
  {                                        //循环遍历查询结果集
        Student student=new
    Student();                           //定义与数据表对应的
    JavaBean对象
        student.setId(rest.getInt(1));
        //设置对象属性
        student.setName(rest.getString(2));
        student.setCollege(rest.getString(3));
        student.setSpeciality(rest.getString(4));
        student.setAddress(rest.getString(5));
        list.add(student);                                   /
    /向集合中添加元素
      }
    } catch (Exception e) {
```

```
            e.printStackTrace();
        }
        return
    list;                                                    //返回查询
    结果
    }
```

举一反三

根据本实例，读者可以实现以下功能。

使用IN子查询实现反向查询。

使用NOT IN 实现反向查询。

实例214 在子查询中使用聚集函数

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\05\Ex05_214

实例说明

在实际开发中子查询占有非常重要的位置，灵活地运用子查询可解决实际开发中的很多问题。本实例是实现在子查询中使用聚集函数查询工资高于员工平均工资的员工信息。实例运行效果如图5.51所示。

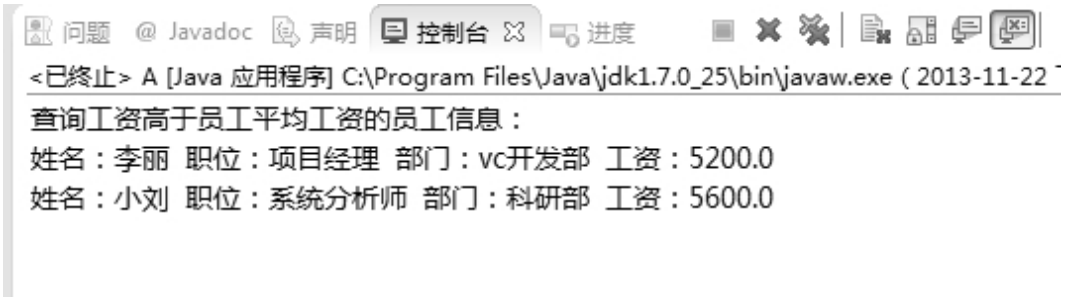


图5.51 实例运行效果

技术要点

本实例实现在子查询语句中使用聚集函数AVG来计算员工的平均工资。

实现过程

（1）在项目中创建类FindEmp，用于查询工资高于平均工资的员工信息。在该类中首先定义获取数据库连接的方法getConn()，具体代码读者可参考光盘中的源程序，这里不再赘述。

（2）在该类中定义查询工资高于平均工资的员工信息的方法getSubTable()，该方法以 List集合对象作为返回值。具体代码如下：

```
public List getSubTable() {
  List list=new ArrayList<Emp>
();                              //定义List集合对象
  conn=getConn();                                    //
获取与数据库的连接
  try {
    Statement statement = conn.createStatement();//获取
Statement对象
    String sql ="select eName,headship,dept,laborage from
tb_emp where laborage >(select avg(laborage) from
tb_emp)";        //定义查询数据的SQL语句
    ResultSet rest=
statement.executeQuery(sql);                  //执行查询语
句获取查询结果集
    while (rest.next())
{                                    //循环遍历查询结果集
      Emp emp=new Emp();                                //定
义与数据表对应的JavaBean对象
```

```
            emp.setName(rest.getString(1));
            emp.setHeadship(rest.getString(2));
            emp.setDept(rest.getString(3));
            emp.setLaborage(rest.getFloat(4));
            list.add(emp);                                    //
        将对象添加到集合中
          }
        } catch (Exception e) {
          e.printStackTrace();
        }
        return
    list;                                                     //返回查询
    结果
      }
```

举一反三

根据本实例，读者可以开发以下程序。

查询年龄高于员工平均年龄的员工信息。

查询工资低于员工平均工资的员工信息。

实例215 在删除数据时使用子查询

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\05\Ex05_215

实例说明

在使用DELETE语句删除数据时，可以使用子查询限制删除的数据，当子查询结果集存在时删除记录。本实例使用DELETE语句删除部

门表tb_mrdept中在员工表tb_mremp中不存在的记录，并将删除前与删除后部门表中的信息输出在控制台上。实例运行效果如图5.52所示。

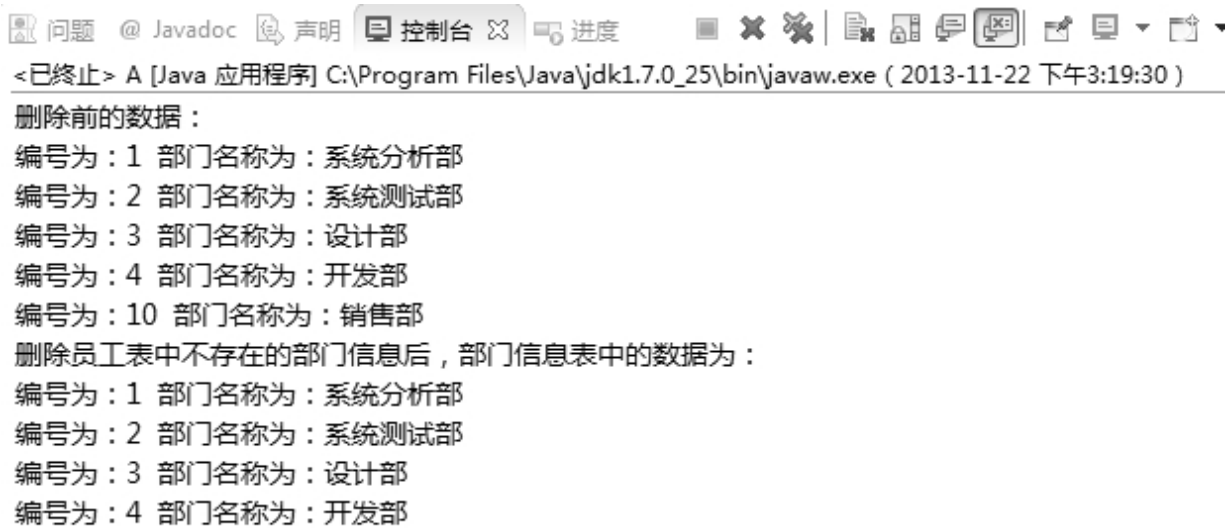


图5.52 实例运行效果

技术要点

DELETE语句可以和子查询联合使用，子查询指定删除数据记录的条件，使用的语法如下：

```
DELETE FROM tableName
WHERE search_condition IN|NOT IN(子查询)
```

参数说明

- tableName：指定要操作的数据表。
- search_condition：指定条件的数据列。

实现过程

（1）在项目中创建类 DeleteEmp，用于定义查询部门信息表中所有数据信息的方法和删除部门表中在员工表中不存在的信息。查询部门信息表中的所有记录不是本章的重点，读者可参考光盘中的源程序。

（2）在该类中定义删除员工表中所有记录的方法deleteEmp()，该方法的具体代码如下：

```
public void deleteEmp() {
  conn=getConn();
      //获取与数据库的连接
  try {
    Statement statement=
conn.createStatement();                              //获取
Statement对象
    //在删除语句中使用子查询
    String sql ="delete from tb_mrdept where person not
in (select name from tb_mremp )";
    statement.executeUpdate(sql);
            //执行删除SQL语句
  } catch (Exception e) {
    e.printStackTrace();
  }
}
```

举一反三

根据本实例，读者可以开发以下程序。

修改数据时使用子查询。

使用子查询增加数据。

5.5 嵌套查询

实例216 查询平均成绩在85分以上的学生信息

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\05\Ex05_216

实例说明

本实例实现操作两张表，分别为学生信息表与学生成绩表，以学生信息表作为主查询，以学生成绩表作为子查询，以学生姓名作为连接条件。实例运行效果如图5.53所示。

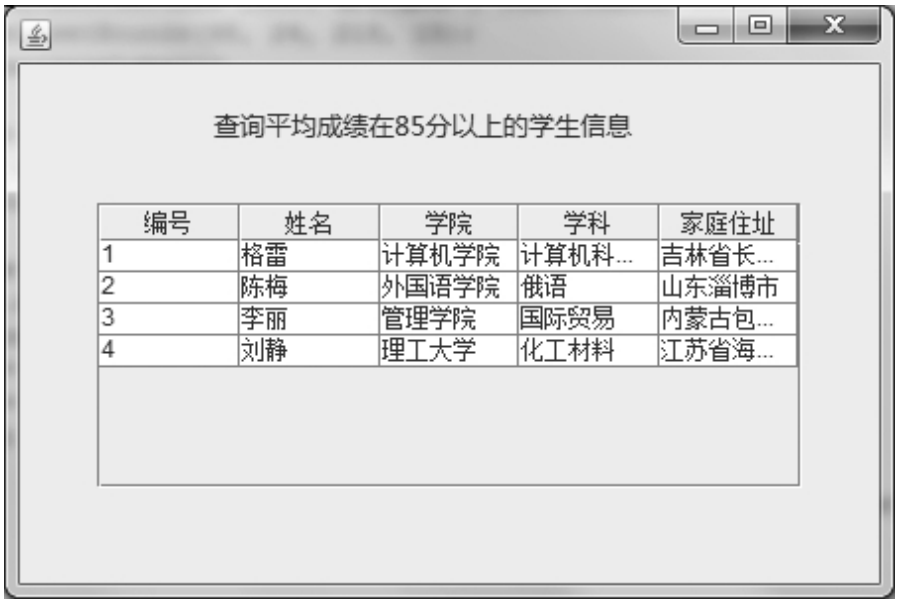


图5.53 实例运行效果

技术要点

本实例利用一个嵌套子查询来实现。子查询是一个 SELECT 查询，返回单个值且嵌套在SELECT、INSERT、UPDATE、DELECT语句或其他子查询中。任何可以使用表达式的地方都可以使用子查询。

子查询也称为内部查询或内部连接，而包含子查询的语句也称为外部查询或外部连接。许多包含子查询的SQL语句都可以通过连接实现。

子查询可以把一个复杂的查询分解成一系列的逻辑步骤，这样就可以用一个简单语句解决复杂的查询问题。当查询依赖于另一个查询的结果时，子查询会很有用。

实现过程

（1）在项目中创建类FindStuFrame，该类继承自JFrame类，实现窗体类。向该窗体中添加标签、表格控件，标签控件用于向用户显示提示信息，表格控件用于显示查询的结果。

（2）创建工具类 FindStu，用于定义查询数据的方法。在该类中定义查询数据库的方法getSubTable()，该方法将查询结果以List对象返回。具体代码如下：

```
public List getSubTable() {
  List list=new ArrayList<Student>
();                                    //定义List集合对象
  conn=getConn();
     //获取与数据库的连接
  try {
    Statement statement = conn.createStatement();//获取
Statement对象
    String sql ="select * from tb_student where name in
(select name from tb_grade
where((math+english+chinese)/3)>=85)";        //定义查询
语句
    ResultSet rest=
statement.executeQuery(sql);                          //
```

```
  执行查询语句获取查询结果集
      while (rest.next())
  {                                                    //循环遍历查
  询结果集
        Student student=new
    Student();                                         //定义与数据
    表对应的JavaBean对象
        student.setId(rest.getInt(1));
                  //设置对象属性
        student.setName(rest.getString(2));
        student.setCollege(rest.getString(3));
        student.setSpeciality(rest.getString(4));
        student.setAddress(rest.getString(5));
        list.add(student);
              //向集合中添加元素
      }
    } catch (Exception e) {
      e.printStackTrace();
    }
    return
list;
//返回查询结果
  }
```

举一反三

根据本实例，读者可以开发以下程序。

查询平均成绩在85以上的学生人数。

查询平均成绩在90以上的男学生人数。

实例217 查询本科部门经理月收入情况

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\05\Ex05_217

实例说明

本实例实现在员工表、部门表、工资表之间进行查询，求得学历是本科的部门经理的月收入情况。本实例实现将这3张表进行嵌套连接查询，实例运行效果如图5.54所示。

图5.54 实例运行效果

技术要点

本实例应用了嵌套查询。这里介绍有关IN谓词在嵌套查询中的使用。语法格式如下：

```
test_expression[NOT] IN
(
  subquery
  expression[, …n]
)
```

参数说明

- test_expression：SQL 表达式。
- subquery：包含某列结果集的子查询，该列必须与test_expression 具有相同的数据类型。
- expression[,…n]：一个表达式列表，用来测试是否匹配，所有的表达式必须和test_expression具有相同的数据类型。

其返回值是把test_expression与subquery返回的值进行比较，如果两个值相等，或与逗号分隔的列表中的任何expression相等，那么结果值就为true，否则结果值为false。

另外，使用NOT IN 将对返回值取反。

实现过程

（1）创建类 FindLaborage，用来定义查询数据的方法。在该类中定义连接数据库的方法getConn()，具体代码读者可参考光盘中的源程序，这里不再赘述。

（2）在该类中定义查询数据信息的方法 getMessageEmp()，该方法以查询结果集 ResultSet对象作为返回值。具体代码如下：

```
public ResultSet getMessageEmp() {
  conn=getConn();
  //获取与数据库的连接
  try {
    Statement statement=
conn.createStatement();                    //获取
Statement对象
    String sql ="select distinct
dName,laborage.name,laborage.laborage,lYear,lDate from
tb_laborage laborage,tb_dept dept,tb_employee emp "+
    "where laborage.name in(select name from tb_employee
where job = '部门经理' "+
    "and schoolAge= '本科' and dID=  dept.id
)";                    //定义查询语句
    ResultSet rest=
statement.executeQuery(sql);                    //执行查
询语句获取查询结果集
```

```
        return
    rest;                                          //返回查
    询结果
      } catch (Exception e) {
        e.printStackTrace();
        return null;
      }
    }
```

（3）在该类的主方法中调用getMessageEmp()方法，遍历查询结果集。具体代码如下：

```
    public static void main(String[] args) {
      FindLaborage Dlaborage=new
  FindLaborage();                          //创建本科类对象
      ResultSet
  rest=Dlaborage.getMessageEmp();                          //调
  用查询方法
      System.out.println("查询本科的部门经理的月收入情况");
      try {
        while(rest.next())
  {                                          //循环遍历查询结
  果集
          String dName=
  rest.getString(1);                          //获取结果
  集中信息
          String name = rest.getString(2);
          float laborage = rest.getFloat(3);
          int lYear = rest.getInt(4);
```

```
            int lDate = rest.getInt(5);
            System.out.println("姓名："+name+"部门："+dName+"工
        资："+laborage+"年份："+lYear+" 月份："+lDate);
          }
        } catch (Exception e) {
          e.printStackTrace();
        }
    }
```

举一反三

根据本实例，读者可以开发以下程序。

查询本科部门经理的平均月收入。

查询本科部门大于平均月收入的人员信息。

实例218 在嵌套中使用EXISTS关键字

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\05\Ex05_218

实例说明

EXISTS关键字用于指定子查询中是否有指定的行存在。本实例在查询语句中嵌套EXISTS关键字，实现查询语文成绩大于等于90分的学生信息。本实例实现操作两张表，分别为学生信息表tb_student与学生成绩表tb_grade。实例运行效果如图5.55所示。

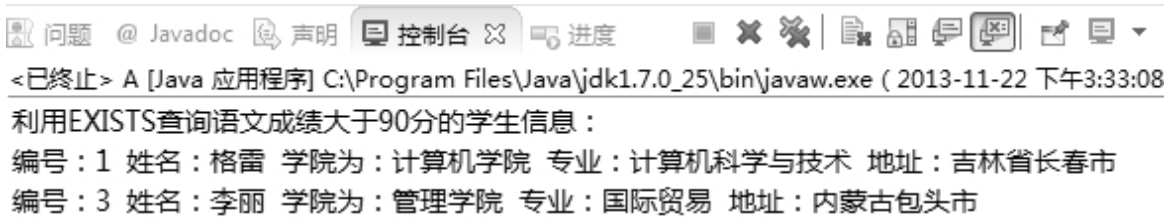


图5.55 实例运行效果

技术要点

如果要在子查询中返回一个真值或假值，则只考虑是否满足谓词条件，数据内容本身并不重要。如果子查询返回一行或多行，则返回真，否则返回假。要使EXISTS谓词有用，应该在查询中建立查询条件以匹配子查询连接起来的两个表中的值。语法格式如下：

EXISTS subquery

参数说明

subquery：不允许有COMPUTE子句和INTO关键字的一个受限SQL语句。

其结果类型为boolean，如果子查询包含行，则返回true，否则返回false。

实现过程

（1）创建类 FindStuExists ，用于定义查询数据方法。在该类中定义连接数据库的方法getConn()，具体代码读者可参考光盘中的源程序，这里不再赘述。

（2）在该类中定义getMessageEmp()方法，该方法用于查询语文成绩大于等于90分的学生信息。具体代码如下：

```
public List getMessageEmp() {
  conn=getConn();
      //获取与数据库的连接
  List list = new ArrayList();
  try {
    Statement statement=
conn.createStatement();                              //获取
Statement对象
    String sql ="select * from tb_student s where exists
(select name from tb_grade g where chinese >=90 and
s.name =g.name)";        //定义查询语句
```

```
    ResultSet rest=
statement.executeQuery(sql);                                    //
执行查询语句获取查询结果集
    while(rest.next()){
      Student student=new
  Student();                                          //定义与数据
  表对应的JavaBean对象
      student.setId(rest.getInt(1));
                //设置对象属性
      student.setName(rest.getString(2));
      student.setCollege(rest.getString(3));
      student.setSpeciality(rest.getString(4));
      student.setAddress(rest.getString(5));
      list.add(student);
            //向集合中添加对象
    }
    return
list;                                                           //
返回查询结果
  } catch (Exception e) {
    e.printStackTrace();
    return null;
  }
}
```

举一反三

根据本实例，读者可以实现以下功能。

查询子查询中是否有指定的行存在。

查询成绩表中大于平均成绩的学生信息。

实例219 动态指定查询条件

这是一个可以启发思维的实例

实例位置：光盘\mingrisoft\05\Ex05_219

实例说明

本实例实现查询学生成绩表中的数据，用户在查询该表中的数据时，可以动态指定查询的字段，以及字段与查询值之间的关系。实例运行效果如图5.56所示。

图5.56 实例运行效果

技术要点

本实例将要查询的字段以下拉列表的形式显示给用户。在利用下拉列表进行查询时，应使用连接符“+”获取下拉列表的值与查询字符串连接，但查询字段的两端不需要添加单引号“’ ”。

实现过程

（1）在项目中创建类SelectTrendsFrame，该类继承自JFrame类，实现窗体类。向该窗体中添加表格、下拉列表等控件，完成窗体布局，该窗体中的主要控件及说明如表5.8示。

表5.8 窗体中的主要控件及说明

控 件 类 型	控 件 命 名	控 件 用 途
JComboBox	subjectComboBox	供用户选择“学科”的下拉列表
	connectionComboBox	供用户选择“关系”的下拉列表
JTextField	valueTextField	供用户添加查询值的文本框控件
JButton	findButton	按钮控件
JTable	table	显示查询结果的表格控件

（2）在项目中创建TrendsSelect类，在该类中定义指定条件动态查询数据的方法getGrade，该方法包含两个String类型的参数，分别用于指定查询的字段以及字段与值之间的关系；一个int类型的参数，用于指定字段满足的值。具体代码如下：

```
    public List getGrade(String operator,String
denotation,int mark) {
      conn=getConn()  ;
            //获取与数据库的连接
      ResultSet rest;
      List list = new ArrayList();
      try {
        Statement statement=
    conn.createStatement();                          //获取
    Statement对象
        String sql ="select * from tb_grade where
    "+operator+denotation+mark;
        rest=
    statement.executeQuery(sql);
        //执行查询语句获取查询结果集
        while (rest.next())
    {                                                //循环遍历查询结
    果集
```

```
        Grade grade=new
    Grade();                                    //定义与数据表对
    应的JavaBean对象
        grade.setId(rest.getInt(1));
              //设置对象属性值
        grade.setName(rest.getString(2));
        grade.setSex(rest.getString(3));
        grade.setEnglish(rest.getInt(4));
        grade.setChinese(rest.getInt(5));
        grade.setMath(rest.getInt(6));
        list.add(grade);
      }
    } catch (Exception e) {
      e.printStackTrace();
    }
    return
list;                                                   //返回
集合
  }
```

举一反三

根据本实例，读者可以开发以下程序。

动态指定查询条件。

动态指定字段与查询值之间的关系。

5.6 连接查询

实例220 使用UNION运算符使学生档案归档

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\05\Ex05_220

实例说明

UNION指的是并运算，就是从两个或多个类似的结果集中选择行，并将其组合在一起形成一个单独的结果集。本实例利用 UNION 运算符将多个年级的学生档案归档，并保存在一张表中。实例运行效果如图5.57所示。

图5.57 实例运行效果

技术要点

UNION运算符主要用于将两个或更多查询的结果组合为单个结果集，该结果集包含联合查询中所有查询的全部行。在使用UNION运算符

时应遵循以下准则。

（1）在使用 UNION 运算符组合的语句中，所有选择列表的表达式数目必须相同（列名、算术表达式、聚集函数等）。

（2）在使用 UNION 运算符组合的结果集中的相应列或个别查询中使用的任意列的子集必须具有相同的数据类型，并且两种数据类型之间必须存在可能的隐性转换或提供了显式转换。例如，在datetime数据类型的列和binary数据类型的列之间不可能存在UNION运算符，除非提供了显式转换，而在money数据类型的列和int数据类型的列之间可以存在UNION运算符，因为它们可以进行隐性转换。

（3）用UNION运算符组合的各语句中对应的结果集列出现的顺序必须相同，因为UNION运算符是按照各个查询给定的顺序逐个比较各列。

（4）UNION 运算符组合不同的数据类型时，这些数据类型将使用数据类型优先级的规则进行转换。例如，int值转换成float值，因为float类型的优先权比int类型高。

（5）通过UNION运算符生成的表中的列名来自UNION语句中的第一个单独的查询。若要用新名称引用结果集中的某列（如在ORDER BY 子句中），必须按第一个SELECT 语句中的方式引用该列。

实现过程

（1）在项目中创建类SelectUseExists，该类继承自JFrame类，实现窗体类。在该窗体中添加标签和表格控件，标签控件用于给用户提示信息，表格控件用于显示查询结果。

（2）定义StudentUnion类，在该类中定义getMessageEmp()方法，该方法以List集合作为返回值，用于获取查询结果。具体代码如下：

```
public List getMessageEmp() {
```

```
  conn=getConn();
  //获取与数据库的连接
  List list = new ArrayList<Student>();
  try {
    Statement statement=
conn.createStatement();                    //获取
Statement对象
    String sql ="select * from tb_stu2006 union select *
from tb_stu2007 union select * from tb_stu2008";
    //定义查询语句
    ResultSet rest=
statement.executeQuery(sql);                    //执行查
询语句获取查询结果集
    while(rest.next()){
      Student student = new Student();
      student.setId(rest.getString(1));
      student.setName(rest.getString(2));
      student.setSex(rest.getString(3));
      student.setSpciality(rest.getString(4));
      student.setAddress(rest.getString(5));
      list.add(student);
    }
    return
list;                                          //返回
查询结果
  } catch (Exception e) {
    e.printStackTrace();
```

```
        return null;
      }
    }
```

举一反三

根据本实例，读者可以开发以下程序。

用UNION运算符将两个结果组合为单个结果集。

用UNION运算符将更多查询的结果组合为单个结果集。

实例221 内连接获取指定课程的教师信息

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\05\Ex05_221

实例说明

严格来说，一张数据表不允许出现冗余的字段、不允许有经过计算可得到的数据列。这样，如果程序员要获取某信息就要通过查询语句来获取。所以灵活地使用查询语句，对于一个程序员来说非常重要。内连接查询（INNER JOIN）使用比较运算符对表之间某些列数据进行比较，并列出这些表中与连接条件相匹配的数据行。本实例采用内连接查询两张表，分别为课程信息表tb_coursey与教师表tb_teacher，得到某课程对应的教师。本实例涉及的两张表中的数据如图5.58所示。

id	cName
1	语文
2	英语
3	数学
4	物理
5	化学
6	历史

(a)

id	cId	tName
1	1	张三
2	2	赵宇
3	3	李丽
4	4	李静
5	5	陈梅
6	6	孙宇

(b)

图5.58 课程信息表与教师表中的数据

本实例显示查询课程对应的教师信息，并将其在控制台上输出，运行结果如图5.59所示。

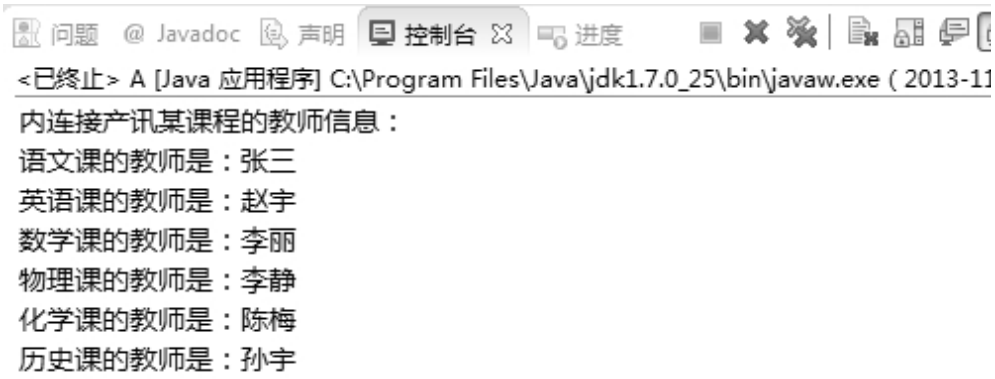


图5.59 实例运行结果

技术要点

内连接可分为等值连接、自然连接和不等连接。本实例使用的是等值连接，就是使用等号运算符比较被连接列的值，在查询结果中列出本连接表中的所有列，包括其中的重复列。等值连接用于返回所有连接表中具有匹配值的行，而排除其他行。

等值连接查询的语法格式如下：

```
select fildList from table1 inner join table2 on
table1.column = table2.column
```

参数说明

fildList：要查询的字段列表。

实现过程

（1）在项目中创建类 CreateJoin，用于定义查询数据的方法。在该类中定义获取数据库连接的方法getConn()，具体代码读者可参考光盘中的源程序，这里不再赘述。

（2）在该类中定义 getJoin()方法，用于通过内连接获取指定课程的教师信息，该方法以ResultSet对象作为返回值。具体代码如下：

```
public ResultSet getJoin() {
```

```
  conn=getConn();
      //获取与数据库的连接
  ResultSet rest;
  try {
    Statement statement=
conn.createStatement();                              //获取
Statement对象
    String sql ="select cName,tName from tb_course c
inner join tb_teacher t on c.id = t.cId ";
    //定义查询语句
    rest=
statement.executeQuery(sql);
    //执行查询语句获取查询结果集
    return
rest;                                                    //
返回查询结果
  } catch (Exception e) {
    e.printStackTrace();
    return null;
  }
}
```

举一反三

根据本实例，读者可以开发以下程序。

内连接查询课程的信息。

内连接查询指定课程的教师信息。

实例222 左外连接查询员工信息

这是一个可以提高分析能力的实例

实例位置：光盘\mingrisoft\05\Ex05_222

实例说明

实例221为大家介绍了使用内连接查询数据，还有一个查询方式就是外连接。外连接与内连接的区别就是，内连接只返回两张表项匹配的数据，而外连接是对内连接的扩展，使用外连接查询的一个好处就是可以使查询更加完整，不会丢失数据。本实例实现对部门表左外连接员工表，实例运行效果如图5.60所示。

图5.60 实例运行效果

技术要点

左外连接的关键字为LEFT JOIN，内连接与外连接的区别如下。

假设有两张表，分别为表A与表B，两张表公共的部分是C。

内连接只有在两个表都存在的记录才会得出，可以说A内连B得到的是C。

表A左连接B，那么A不受影响，查询结果为：公共部分C加表A的记录集。

表A右连接B，那么B不受影响，查询结果为：公共部分C加表B的记录集。

全外连接表示两张表都不加限制。

实现过程

（1）在项目中创建类SelectUseLeftFrame，该类继承自JFrame类，实现窗体类。向该类中添加标签与表格控件，标签控件用于给出用户提示信息，表格控件用于显示查询后的结果。

（2）在项目中创建类SelectUseLeft，用于获取查询结果。在该类中定义对数据表进行左外连接的方法getLeft()，该方法以List集合作为返回值。具体代码如下：

```
public List getLeft() {
  conn=getConn();
  //获取与数据库的连接
  ResultSet rest;
  List list = new ArrayList<MrEmp>();
  try {
    Statement statement=
conn.createStatement();                    //获取
Statement对象
    String sql ="select
e.id,dName,person,name,sex,schoolAge from tb_mrdept d
left join tb_mremp e on d.id = e.dId";
    //定义查询语句
    rest=
statement.executeQuery(sql);                          /
/执行查询语句获取查询结果集
    while(rest.next()){
      MrEmp mrEmp=new
  MrEmp();                          //定义与数据表对应的
  JavaBean对象
```

```
            mrEmp.setId(rest.getInt(1));
                //设置对象属性
            mrEmp.setdName(rest.getString(2));
            mrEmp.setPerson(rest.getString(3));
            mrEmp.setName(rest.getString(4));
            mrEmp.setSex(rest.getString(5));
            mrEmp.setSchoolAge(rest.getString(6));
            list.add(mrEmp);
        }
        return
list;                                                    //返回
```
查询结果
```
      } catch (Exception e) {
        e.printStackTrace();
        return null;
      }
}
```
举一反三

根据本实例，读者可以开发以下程序。

用左外连接查询员工信息。

左外连接查询数据库中其他表的信息。

实例223 右外连接查询员工信息

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\05\Ex05_223

实例说明

实例222中已经提到了左外连接，右外连接查询同样是为了保护数据的完整性。左外连接与右外连接的侧重点不同，开发人员可根据自己的需要选择适合的连接。本实例实现了部门信息表右外连接员工表。实例运行效果如图5.61所示。

图5.61 实例运行效果

技术要点

右外连接的运算符为RIGHT JOIN，如果表A 右外连接表B，则结果为公共部分C 加表B的结果集。如果表A中没有与表B匹配的项，就使用NULL进行连接。

实现过程

（1）在项目中创建类SelectUseRightFrame，该类继承自JFrame类。实现窗体类，在该窗体中添加标签、表格控件，标签控件用于给出用户提示信息，表格控件用于显示查询结果。

（2）在项目中定义SelectUseRight类，在该类中定义getRight()方法，用于获取查询结果，该方法以List集合对象作为返回值。具体代码如下：

```
public List getRight() {
  conn=getConn();
  //获取与数据库的连接
  ResultSet rest;
  List list = new ArrayList<MrEmp>();
```

```
  try {
    Statement statement=
conn.createStatement();                    //获取
Statement对象
    String sql ="select
e.id,dName,person,name,sex,schoolAge from tb_mrdept d
right join tb_mremp e on d.id = e.dId";
    //定义查询语句
    rest=
statement.executeQuery(sql);                          /
/执行查询语句获取查询结果集
    while (rest.next()) {
      MrEmp mrEmp=new
  MrEmp();                          //定义与数据表对应的
  JavaBean对象
      mrEmp.setId(rest.getInt(1));
          //设置对象属性
      mrEmp.setdName(rest.getString(2));
      mrEmp.setPerson(rest.getString(3));
      mrEmp.setName(rest.getString(4));
      mrEmp.setSex(rest.getString(5));
      mrEmp.setSchoolAge(rest.getString(6));
      list.add(mrEmp);
    }
    return
list;                                              //返回
查询结果
```

```
    } catch (Exception e) {
      e.printStackTrace();
      return null;
    }
  }
```

举一反三

根据本实例，读者可以实现以下功能。

用右外连接查询员工信息。

右外连接查询数据库中其他表的信息

实例224 多表外连接查询

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\05\Ex05_224

实例说明

外连接不仅可以用在两张表上，还可以将多个表进行外连接查询。本实例实现将员工表（tb_personnel）、工资表（tb_wage）、请假表（tb_leave）进行外连接查询，得出汇总信息。实例运行效果如图5.62所示。

图5.62 实例运行效果

技术要点

在本实例中两次使用了左外连接查询，即员工表与工资表进行左外连接后再与请假表进行左外连接，将工资表中符合条件的数据、请假表中符合条件的数据和员工表中的全部信息显示出来。如果给出外连接查询条件，则应该使用ON关键字。

实现过程

（1）在项目中定义类FindMore，在该类中定义查询数据的方法。首先定义连接数据库的方法getConn()，该方法以Connection对象作为返回值，具体代码读者可参考光盘中的源程序，这里不再赘述。

（2）在该类中定义查询数据的方法getMore()，用于获取多表外连接查询结果，该方法以查询结果集ResultSet对象作为返回值。具体代码如下：

```
public ResultSet getMore() {
  conn=getConn();
  //获取与数据库的连接
  ResultSet rest;
  try {
    Statement statement=
conn.createStatement();                    //获取
Statement对象
    String sql ="select
p.id,p.sName,w.wId,w.wage,l.pID,l.monthL,l.lDate,l.lMoney
from (tb_personnel p left join tb_wage w on p.id =
w.perId)"+" left join tb_leave l on l.pID =
p.id";        //定义查询语句
    rest=
statement.executeQuery(sql);                              /
/执行查询语句获取查询结果集
```

```
    return
rest;                                                    //返回查
询结果
  } catch (Exception e) {
    e.printStackTrace();
    return null;
  }
}
```

举一反三

根据本实例，读者可以实现以下功能。

用多表外连接进行数据查询。

用外连接显示请假表和员工表中的数据。

实例225 完全连接查询

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\05\Ex05_225

实例说明

左外连接和右外连接都是针对某一张表的完整查询，如果要对连接的两张表都实现完整查询，就要使用完全连接查询，完全连接查询的关键字为FULL JOIN。本实例为大家介绍的是对部门表和员工表实现完全连接查询。实例运行效果如图5.63所示。

图5.63 实例运行效果

技术要点

完全连接查询就是将左表的所有数据分别与右表的每条记录进行连接组合，返回的结果除了连接数据外，还有两个表中不符合条件的数据，并在左或右表的相应列中填上NULL值。本实例中员工“孙梅”，只在部门表中存在，因此其他信息都是 NULL 值；员工“陈双”只在员工表中存在，因此其他信息都是NULL值。

实现过程

（1）在项目中创建类SelectUseFullFrame，该类继承自JFrame类，实现窗体类。在该窗体中添加标签和表格控件，标签控件用于显示提示信息，表格控件用于显示查询结果。

（2）在项目中创建类 SelectUseFull，用于编写查询数据的方法。在该类中定义 getFull()方法，用于获取两张表的完全连接数据，以List集合作为返回值。具体代码如下：

```
public List getFull() {
  conn=getConn();
      //获取与数据库的连接
```

```
  ResultSet rest;
  List list = new ArrayList<MrEmp>();
  try {
    Statement statement=
conn.createStatement();                                    //获取
Statement对象
    String sql ="select
e.id,dName,person,name,sex,schoolAge from tb_mrdept d
full join tb_mremp e on d.id = e.dId";
    //定义查询语句
    rest=
statement.executeQuery(sql);
    //执行查询语句获取查询结果集
    while (rest.next()) {
      MrEmp mrEmp=new
  MrEmp();                                          //定义与数据表对
  应的JavaBean对象
      mrEmp.setId(rest.getInt(1));
              //设置对象属性
      mrEmp.setdName(rest.getString(2));
      mrEmp.setPerson(rest.getString(3));
      mrEmp.setName(rest.getString(4));
      mrEmp.setSex(rest.getString(5));
      mrEmp.setSchoolAge(rest.getString(6));
      list.add(mrEmp);
    }
```

```
    return
list;                                                    //
返回查询结果
  } catch (Exception e) {
    e.printStackTrace();
    return null;
  }
}
```

举一反三

根据本实例，读者可以实现以下功能。

使用完全链接查询两张表中的数据。

将左表的所有数据分别与右表的每条记录进行连接组合。

5.7 函数查询

实例226 在查询中使用patindex()函数进行模糊查询

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\05\Ex05_226

实例说明

应用Like关键字可以进行模糊查询，此外还有一种可以进行模糊查询的方法，即patindex()函数。本实例向大家介绍使用patindex()函数实现模糊查询。patindex()函数返回字符串表达式中第一次出现的位置，本实例实现的是查询某省的客户信息。实例运行效果如图5.64所示。

图5.64 实例运行效果

技术要点

patindex()函数常常用于搜索字符或字符串。如果被搜索的字符串中包含要搜索的字符，那么该函数返回一个非零的整数，表示pattern字符串在表达式expression中第一次出现的位置，起始值为1。patindex()函数支持使用通配符来进行搜索。该函数的语法格式如下：

patindex('%pattern%',expression)

参数说明

● pattern：要搜索的字符串，可以使用通配符。pattern 之前和之后需要使用%号，除非搜索的字符串在被搜索的字符串的最前面或最后面。

● expression：表达式，表示被搜索的字符串，expression 通常是一个表中的字段。

该函数中还可以使用“[ ]”来返回指定某些字符其中之一的位置，例如：

select patindex('%[c,d]%','abcdeft')

该句表达式的意思是字符“c”或“d”，其中一个在表达式“abcdeft”中最先出现的位置，由于“c”在字符串中第一次出现的位置是3，因此该句代码返回值为3。

（2）定义工具类CreatePatindex，用于实现查询数据的方法。在该类中定义getPatindex()方法，该方法包含一个String类型的参数，用于指定要查询的地点。该方法的具体代码如下：

实现过程

（1）在项目中创建类PatindexFrame，该类继承自JFrame类，实现窗体类。向该窗体中添加下拉列表、按钮和表格控件，下拉列表控件用于显示要查询的地区，表格控件用于显示查询结果。

```
public List getPatindex(String address) {
  conn=getConn();
  //获取与数据库的连接
  ResultSet rest;
  List list = new ArrayList<Order>();
  try {
```

```
    Statement statement=
conn.createStatement();                    //获取
Statement对象
    String sql ="select * from tb_order where
patindex('"+ address +"%',address)>0";        //定义查询
语句
    rest=
statement.executeQuery(sql);                          /
/执行查询语句获取查询结果集
    while (rest.next())
{                                          //循环遍历查询结
果集
      Order order=new
  Order();                                 //定义与数据表对
  应的JavaBean对象
      order.setId(rest.getInt(1));
            //设置对象属性
      order.setName(rest.getString(2));
      order.setAddress(rest.getString(3));
      order.setPhone(rest.getString(4));
      list.add(order);
    }
    return
list;                                              //返回
查询结果
  } catch (Exception e) {
    e.printStackTrace();
```

```
        return null;
    }
}
```

举一反三

根据本实例，读者可以实现以下功能。

用patindex()函数进行模糊查询。

用LIKE进行模糊查询。

实例227 在查询中使用ALL谓词

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\05\Ex05_227

实例说明

比较运算符与谓词 ALL 连用，表示与查询子集中每个值进行比较。本实例实现将比较运算符与谓词ALL连用，查询工资比质量部门中所有员工都高的员工信息。本实例实现操作员工表，该表中的数据如图5.65所示。

本实例实现将查询出来的信息在控制台上显示，实例运行效果如图5.66所示。

id	eName	headship	dept	joinDate	laborage
1	张三	Java程序员	Java开发部	2009-10-8	4200
2	李四	VB程序员	VB开发部	2010-2-3	4500
3	李丽	项目经理	VC开发部	2009-4-15	5200
4	小刘	系统分析师	科研部	2010-1-25	5600
5	小兰	程序测试	质量部	2009-12-6	3500
6	王英	程序测试	质量部	2010-12-2	4600

图5.65 员工表中的数据

图5.66 实例运行效果

技术要点

本实例实现查询比质量部门所有员工的工资都高的员工信息，使用了比较运算符（>）与ALL连用的形式，“>ALL”表示大于条件的每一个值，换句话说，就是大于最大值，如>ALL(1,2,3)表示大于3。

ALL用标量值与单列集中的值进行比较，返回一个布尔变量。其语法格式如下：

```
scalar_expression{=|<>| !=|>|>=| !>|<=| !<} ALL { subquery }
```

参数说明

- scalar_expression：任意有效的SQL 表达式。
- =|<>| !=|>|>=| !>|<=| !<：比较运算符。
- subquery：返回单列结果集的子查询。返回列的数据类型必须与scalar_expression 的数据类型相同。

实现过程

（1）在项目中创建类FinMaxEmpLaborage，用于定义查询数据的方法。在该类中定义连接数据库的方法getConn()，该方法的具体代码读者可参考光盘中的源程序，这里不再赘述。

（2）在该类中定义getLaborage()方法，用于获取查询比质量部门所有员工工资都高的员工信息，该方法以List集合对象作为返回值。具体代码如下：

```
public List getLaborage() {
```

```
    conn=getConn();                                          //
获取与数据库的连接
    ResultSet rest;
    List list = new ArrayList<Emp>();
    try {
      Statement statement=
  conn.createStatement();                  //获取Statement对象
      String sql ="select eName,headship,laborage from
  tb_emp where laborage > all(select laborage from tb_emp
  where dept = '质量部')";          //定义查询语句
      rest=
  statement.executeQuery(sql);                             //执行
  查询语句获取查询结果集
      while (rest.next())
  {                                          //循环遍历查询结果集
        Emp emp=new Emp();                                   //定
    义与数据表对应的JavaBean对象
        emp.seteName(rest.getString(1));
      //设置对象属性
        emp.setHeadship(rest.getString(2));
        emp.setLaborage(rest.getFloat(3));
        list.add(emp);                                       //
    向集合中添加对象
      }
      return
  list;                                          //返回查询结
  果
```

```
    } catch (Exception e) {
      e.printStackTrace();
      return null;
    }
  }
```

举一反三

根据本实例，读者可以实现以下功能。

查询工资比质量部门中所有员工都高的员工信息。

查询工资比质量部门中所有员工平均工资都高的员工信息。

实例228 在查询中使用ANY谓词

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\05\Ex05_228

实例说明

实例227为大家介绍了在查询中使用ALL谓词，与该谓词对应的是ANY谓词。ANY谓词表示的是任意的一个，该谓词与比较运算符连用，表示大于或小于子查询中的任意值。本实例实现查询员工表中工资高于质量部门中任意一名员工的员工工资情况，实例运行效果如图5.67所示。

图5.67 实例运行效果

技术要点

本实例功能的实现应用了比较运算符与ANY谓词的连用。例如，“>ANY”表示至少大于条件中的一个值，换句话说，就是大于最小值，如“>ANY(1,2,3)”表示大于1。

要使用带有>ANY的子查询表示要使用某一列满足外部查询中指定的条件，引入子查询的列中的值必须至少大于由子查询返回值的列值中的一个值。

ANY用标量值与单列集合的值进行比较，返回一个布尔变量。其语法格式如下：

scalar_expression {=|<>| !=|>|<|<=| !<} ANY subquery

参数说明

● scalar_expression：任意有效的SQL 表达式。

● =|<>| !=|>|<|<=| !<：任何有效的比较运算符。

● subquery：包含某列结果集的子查询。所返回列的数据类型必须与scalar_expression 的数据类型相同。

实现过程

（1）在项目中创建类FinMINEmpLaborage，用于定义查询数据方法。在该类中定义获取数据库连接的方法getConn()，该方法以Connection对象作为返回值，具体代码读者可参考光盘中的源程序，这里不再赘述。

（2）在该类中定义getLaborage()方法，用于获取比质量部门任意一名员工工资高的员工工资情况。具体代码如下：

```
public List getLaborage() {
  conn=getConn();                                    //
获取与数据库的连接
  ResultSet rest;
  List list = new ArrayList<Emp>();
  try {
```

```
    Statement statement=
conn.createStatement();                    //获取Statement对象
    String sql ="select eName,headship,laborage from
tb_emp where laborage > any(select laborage from tb_emp
where dept = '质量部')";          //定义查询语句
    rest=
statement.executeQuery(sql);                          //执行
查询语句获取查询结果集
    while (rest.next())
{                                          //循环遍历查询结果集
      Emp emp=new Emp();                                  //定
  义与数据表对应的JavaBean对象
      emp.seteName(rest.getString(1));
    //设置对象属性
      emp.setHeadship(rest.getString(2));
      emp.setLaborage(rest.getFloat(3));
      list.add(emp);                                       //
  向集合中添加对象
    }
    return
list;                                          //返回查询结
果
  } catch (Exception e) {
    e.printStackTrace();
    return null;
  }
}
```

举一反三

根据本实例，读者可以实现以下功能。

使用ANY查询小于子查询中的任意值。

使用ANY查询大于子查询中的任意值。

实例229 使用UNION运算符消除重复的行

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\05\Ex05_229

实例说明

去除重复值查询是一种很常见的查询方式，去除重复值查询的实现方式有很多种，本实例向大家介绍使用UNION关键字实现消除重复的行。本实例向大家介绍查询部门表tb_dept和部门表2tb_dept2中的数据，在查询时去除重复的行。实例运行效果如图5.68所示。

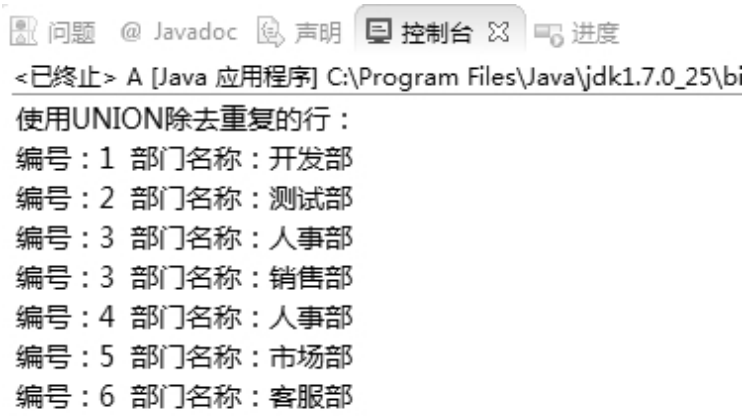


图5.68 实例运行效果

技术要点

将两个或更多查询的结果组合为单个结果集，该结果集包含联合查询中的所有查询的全部行。这与使用连接组合两个表中的列不同。使用 UNION 组合两个查询的结果集的两个基本规则如下：

□ 所有查询中的列数和列的顺序必须相同。

□ 数据类型必须兼容。

语法格式如下：

{< query specification >| (< query expression >) }

UNION [ALL]

< query specification | (< query expression >)

[UNION [ALL] < query specification | (< query expression >)

[...n]]

参数说明

< query specification >：查询规范或查询表达式，用以返回与另一个查询规范或查询表达式所返回的数据组合的数据。

实现过程

（1）在项目中创建类 FinDept，用于定义查询数据的方法。在该类中定义连接数据库的方法getConn()，该方法以Connection对象作为参数，具体代码读者可参考光盘中的源程序，这里不再赘述。

（2）在该类中定义获取数据方法getDept()，该方法以查询结果集对象ResultSet作为返回值。具体代码如下：

```
public ResultSet getDept() {
  conn=getConn();
  //获取与数据库的连接
  ResultSet rest;
  try {
    Statement statement=
conn.createStatement();                    //获取
Statement对象
    String sql= "select * from tb_dept union  select *
from tb_dept2";    //定义查询语句
```

```
    rest=
statement.executeQuery(sql);                                    /
/执行查询语句获取查询结果集
    return
rest;                                                          //返回查
询结果
  } catch (Exception e) {
    e.printStackTrace();
    return null;
}
}
```

举一反三

根据本实例，读者可以实现以下功能。

使用其他方式去除表中的重复行。

使用UNION运算符去除重复行。

实例230 使用 UNION ALL 运算符保留重复的行

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\05\Ex05_230

实例说明

UNION 关键字与ALL 关键字组合使用，可以保留查询结果中的重复值。使用UNIONALL组合查询的两张表必须要有相同的结构，本实例实现查询部门表tb_dept和tb_dept2两表的数据，并使用UNION ALL 关键字保留重复的记录。实例运行效果如图5.69 所示。

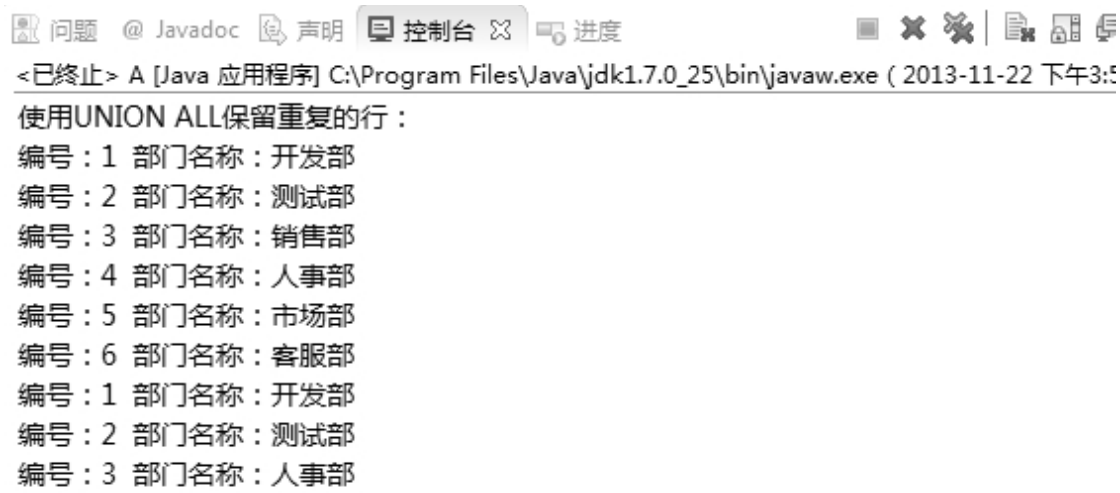


图5.69 实例运行效果

技术要点

UNION加上关键字ALL的功能是不删除重复行也不对行自动排序。加上ALL关键字需要的计算资源少，所以尽可能使用它，尤其是处理大型表时。下列情况应该使用UNION ALL 组合查询。

□ 知道有重复行并想保留这些行。

□ 知道不可能有任何重复的行。

□ 不在乎是否有任何重复的行。

实现过程

（1）在项目中创建类 FinDept，用于定义查询数据的方法。在该类中定义连接数据库的方法getConn()，该方法以Connection对象作为参数，具体代码读者可参考光盘中的源程序，这里不再赘述。

（2）在该类中定义获取数据方法getDept()，该方法以查询结果集对象ResultSet作为返回值。具体代码如下：

```
public ResultSet getDept() {
  conn=getConn();
  //获取与数据库的连接
  ResultSet rest;
  try {
```

```
    Statement statement=
conn.createStatement();                    //获取
Statement对象
    //定义查询语句
    String sql ="select * from tb_dept union all select *
from tb_dept2";
    rest=
statement.executeQuery(sql);                              /
/执行查询语句获取查询结果集
    return
rest;                                                  //返回查
询结果
  } catch (Exception e) {
    e.printStackTrace();
    return null;
  }
}
```

举一反三

根据本实例，读者可以实现以下功能。

保留表中的重复行。

删除表中的重复行。

实例231 计算商品销售额所占的百分比

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\05\Ex05_231

实例说明

很多程序都要求程序员将一些数据的百分比显示给用户，方便用户查询。本实例实现查询商品表中的商品销售额，以及该销售额所占总销售额的百分比。实例运行效果如图5.70所示。

图5.70 实例运行效果

技术要点

本实例利用 SUM 函数将总的销售额计算出来，再将某商品的销售额除以总的销售额乘以100后得到商品销售额所占的百分比。

实现过程

（1）在项目中创建类WareFrame，该类继承自JFrame类，实现窗体类。在该窗体中添加标签、按钮与表格控件，标签控件给出用户提示信息，表格控件实现查询结果。

（2）在项目中创建工具类WareUtil，在该类中定义查询数据的方法getWare()，该方法以List集合作为返回值。具体代码如下：

```
public List getWare() {
  conn=getConn();                                          //
获取与数据库的连接
  ResultSet rest;
  List list = new ArrayList();
  try {
    Statement statement=
conn.createStatement();                  //获取Statement对象
```

```
    String sql ="select
id,wName,price,convert(varchar(30),price/(select
sum(price) from tb_ware) * 100)+'%' as percente from
tb_ware";        //定义查询语句
    rest=
statement.executeQuery(sql);                    //执行
查询语句获取查询结果集
    while(rest.next())
{                                    //循环遍历查询结果集
      Ware ware=new Ware();                          //定
  义与数据表对应的JavaBean对象
      ware.setId(rest.getInt(1));
    //设置对象属性
      ware.setwName(rest.getString(2));
      ware.setPrice(rest.getFloat(3));
      ware.setPercent(rest.getString(4));
      list.add(ware);                                //
  向集合中添加对象
    }
  } catch (Exception e) {
    e.printStackTrace();
  }
  return
list;                                          //返回集合
  }
```

举一反三

根据本实例，读者可以实现以下功能。

使用聚集函数计算百分比。

计算销售额占总额的百分比。

第6章 数据库高级应用

在Java程序中使用存储过程
使用触发器
批处理的应用
使用视图

6.1 在Java程序中使用存储过程

实例232 调用存储过程实现用户身份验证

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\06\Ex06_232

实例说明

存储过程是为完成特定的功能而汇集在一起的一组经编译后存储在数据库中的SQL语句。存储过程可以用参数的形式输入、输出数据，并支持用户设计的变量和流程控制。一个存储过程中可以包含查询、插入、删除、更新等操作，当一个存储过程被执行时，这些操作也会同时执行。本实例实现用户登录时调用存储过程。实例运行效果如图6.1所示。

图6.1 实例运行效果

技术要点

本实例首先在SQL Server 2000 数据库下创建存储过程，之后在Java 程序中调用存储过程。主要用到SQL语句中的CREATEPROCEDURE来创建存储过程。语法格式如下：

```
CREATE PROC [ EDURE ] procedure_name [ ; number ]
  [ {@parameter data_type }
```

[VARYING] [= default] [OUTPUT]
] [,...n]
[WITH
{ RECOMPILE | ENCRYPTION | RECOMPILE , ENCRYPTION }]
[FOR REPLICATION]
AS sql_statement [...n]

参数说明

● procedure_name：表示新存储过程的名称。

● number：表示可选的整数，用来对同名的过程分组，以便用一条DROP PROCEDURE 语句将同组的过程一起删除。

● @parameter：表示过程中的参数。

● data_type：表示参数的数据类型。

在Java程序中调用存储过程实现登录验证。调用存储过程的语法格式如下：

{ CALL procname (?,?)}

参数说明：

● procname：表示存储过程的名称。

● ？：表示传递的参数。

实现过程

（1）在项目中创建类EnterFrame，该类继承自JFrame类，实现窗体类。向窗体中添加控件，主要控件及说明如表6.1所示。

表6.1 窗体中的主要控件及说明

控件类型	控件命名	控件用途
JTextField	userTextField	显示“用户名”文本框控件
	passwordTextField	显示“密码”文本框控件
JButton	enterButton	显示“登录”按钮控件
	closeButton	显示“关闭”按钮控件

（2）在数据库中创建存储过程，代码如下：

```
create procedure validateSelect
@userName varchar(20),
@password varchar(20)
as select * from tb_user where userName =@userName and
password =@password
```

（3）在项目中创建类 TransferProcure，用于实现调用存储过程。在该类中定义调用存储过程的方法executeQuery()，该方法包含有两个String类型的参数，与存储过程中的参数相对应。具体代码如下：

```
public String executeQuery(String userName,String
passWord){
    String message= "验证失败";                          //
  定义保存返回值的字符串对象
    conn=getConn();                                    //获取
  数据库连接
    CallableStatement cs=null;                          //
  定义CallableStatement对象
    //定义调用存储过程语句
    String sql ="{call
  validateSelect('"+userName+"','"+passWord+"')}";
    try {
      cs= conn.prepareCall(sql);                        //
    调用存储过程
      ResultSet rest=
    cs.executeQuery();                  //获取结果集
      while(rest.next()){                               //循
    环遍历结果集对象
```

```
            message= "验证成功";                          //设置对
        象信息
          }
        } catch (SQLException e) {
          e.printStackTrace();
        }
        return message;                                    //返
    回String对象
      }
```

举一反三

根据本实例，读者可以开发以下程序。

应用存储过程实现登录。

调用存储过程修改数据。

实例233 应用存储过程添加数据

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\06\Ex06_233

实例说明

在实际开发中，经常需要对数据进行操作。由于存储过程是线程同步的，并且具有事务回滚的功能，因此应用存储过程对数据库进行操作比较安全。本实例实现在Java程序中调用存储过程，完成将数据添加到数据库中。实例运行效果如图6.2所示。

图6.2 实例运行效果

技术要点

在Java中可以使用CallableStatement接口来创建调用存储过程的语句。语法如下：

```
CallableStatement cs = con.prepareCall(sql);
```

创建一个CallableStatement 对象，使用该对象就可以完成向存储过程传递参数的功能。语法如下：

```
cs.setString(1,"mm");
```

在上面语句中，1代表存储过程中参数的序列位置，"mm"代表所传递的参数。

执行存储过程可以使用executeUpdate语句。语法如下：

```
cs.executeUpdate();
```

实现过程

（1）在项目中创建类InsertUserFrame，该类继承自JFrame类，实现窗体类。向该窗体中添加标签、文本框与按钮控件，实现窗体布局。该窗体中的主要控件及说明如表6.2所示。

表6.2 窗体中的主要控件及说明

控 件 类 型	控 件 命 名	控 件 用 途
JTextField	userNameTextField	显示“用户名”的文本框控件
	passWordTextField	显示“密码”的文本框控件
	ageTextField	显示“年龄”的文本框控件
	jobTextField	显示“工作”的文本框控件
JComboBox	sexComboBox	显示“性别”的下拉列表控件
JButton	insertButton	显示“添加”的按钮控件
	closeButton	显示“关闭”的按钮控件

（2）在数据库中创建存储过程，当用户调用该存储过程时，实现向用户表中添加数据。代码如下：

```
create procedure insertUser
@userName varchar(20),
@passWord varchar(20),
@age int,
@sex varchar(4),
@job varchar(20)
as
insert into tb_user
values(@userName,@passWord,@age,@sex,@job)
go
```

（3）在项目中创建工具类UserUtil实现调用存储。在该类中定义executeUpdate()方法实现调用存储过程，该方法有一个与数据表对应的JavaBean类对象作为参数。具体代码如下：

```
public boolean executeUpdate(User user) {
  conn=getConn();
  //获取数据库连接
  CallableStatement
cs=null;                                    //定义
CallableStatement对象
```

```
    String sql ="{call insertUser('"+ user.getUserName()
+"','"+ user.getPassword() +"','"+ user.getAge()
+"','"+user.getSex()+ "','"+user.getJob()+
"')}";                    //定义调用存储过程的SQL语句
    try {
      cs=
  conn.prepareCall(sql);                                    /
  /实例化CallableStatement对象
      cs.executeUpdate();
        //执行SQL语句
      return true;
    } catch (SQLException e) {
      e.printStackTrace();
      return false;
    }
  }
```

举一反三

根据本实例，读者可以实现以下功能。

创建存储过程。

修改存储过程。

实例234 修改存储过程

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\06\Ex06_234

实例说明

如果要修改存储过程，可以在数据库中完成，当然也可以通过Java程序实现。本实例实现编写程序为用户提供修改存储过程的文本域，实现创建窗体，用户可通过在窗体中书写代码，实现修改指定的存储过程。实例运行效果如图6.3所示。当用户添加完成信息后，单击“修改”按钮，可实现修改存储过程。

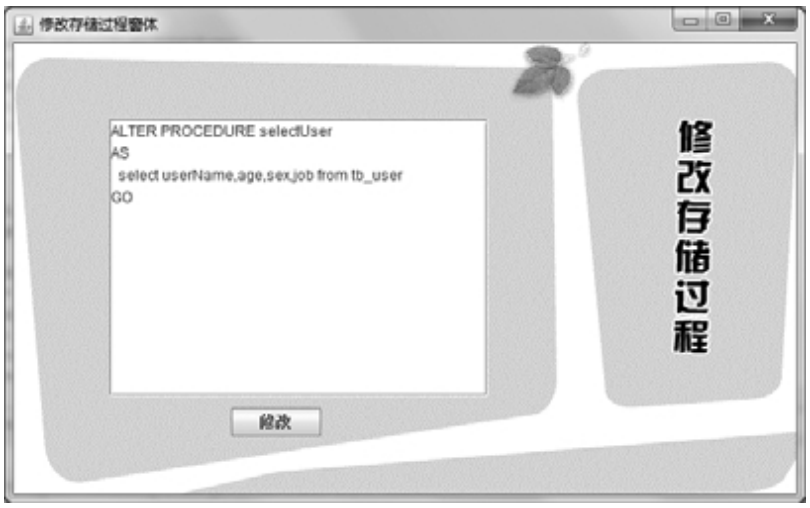


图6.3 实例运行效果

技术要点

本实例主要利用SQL 语句中的ALTER PROCEDURE 语句来实现。语法格式如下：

```
ALTER PROCEDURE [ EDURE ] procedure_name [ ; number ]
  [ {@parameter data_type }
    [ VARYING ] [ = default ] [ OUTPUT ]
  ] [ ,...n ]
[ WITH
  { RECOMPILE | ENCRYPTION
    | RECOMPILE , ENCRYPTION
  }
]
[FOR REPLICATION]
```

AS

sql_statement[...n]

参数说明

● procedure_name：表示要更改的存储过程的名称。

● number：表示现有的可选整数，该整数用来对同一名称的过程进行分组，这样可以用一条DROPPROCEDURE语句全部删除它们。

● @parameter：表示过程中的参数。

● data_type：表示参数的数据类型。

● VARYING：指定作为输出参数支持的结果集。

● default：表示参数的默认值。

● OUTPUT：表明参数是返回参数。

● n：表示最多可指定2～100 个参数的占位符。

● RECOMPILE：表明Microsoft® SQL Server ™ 不会高速缓存该过程的计划，该过程将在运行时重新编译。

● ENCRYPTION：表示SQL Server 加密syscomments 表中包含ALTER PROCEDURE 语句文本的条目。使用ENCRYPTION 可防止将过程作为SQL Server 复制的一部分发布。

实现过程

（1）在项目中创建类AlterProcFrame，该类继承自JFrame类，实现窗体类。在该窗体中添加文本域与按钮控件，文本域控件用于为用户提供修改存储过程控件，按钮控件显示“修改”，用户单击该按钮可实现修改操作。

（2）在项目中定义工具类AlterProce，用于定义执行指定的SQL语句的方法，在该类中定义获取数据库连接的方法getConn()，具体代码读者可参考光盘中的源程序，这里不再赘述。

（3）在该类中定义执行修改存储过程的方法executeUpdate()，该方法有一个String类型的参数，用来指定要执行的SQL语句。具体代

码如下：

```
public boolean executeUpdate(String sql) {
  conn=getConn();
            //获取数据库连接
  try {
    Statement stmt=
conn.createStatement();
//实例化Statement对象
    int iCount=
stmt.executeUpdate(sql);
    //执行修改语句
    System.out.println("操作成功，所影响的记录数
为"+String.valueOf(iCount));      //给出提示信息
    conn.close();
              //关闭连接
  } catch (SQLException e) {
    e.printStackTrace();
    return false;
  }
  return true;
}
```

举一反三

根据本实例，读者可以实现以下功能。

在数据库中修改存储过程。

在Java程序中修改存储过程。

实例235 删除存储过程

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\06\Ex06_235

实例说明

对某一创建完成的存储过程，如果不再需要，可以将其删除。删除存储过程的语句为DROP PROCEDURE。为了方便用户操作，本实例将某数据库下的所有存储过程都列在表格中，用户只需要在表格中选择要删除的存储过程，单击“删除”按钮即可。实例运行效果如图6.4所示。

图6.4 实例运行效果

技术要点

本实例主要是用SQL 语句中的DROP PROCEDURE 语句实现的。DROP PROCEDURE 语句用来删除一个或者多个存储过程或者过程组，其语法格式如下：

```
DROP PROCEDURE { procedure } [ ,...n ]
```

参数说明

● procedure：表示要删除的存储过程或存储过程组的名称。过程名称必须符合标识符规则。

● n：表示可以指定多个过程的占位符。

实现过程

（1）在项目中创建类DeleteProcedureFrame，该类继承自JFrame类，实现窗体类。在该窗体中添加表6.3所示的控件，实现窗体布局。

表6.3 窗体中的主要控件及说明

控件类型	控件命名	控件用途
JTable	table	显示查询得到的数据库中的存储过程
JButton	deleteButton	显示“删除”按钮控件
	closeButton	显示“关闭”按钮控件

（2）在项目中创建工具类DeleteProcedure，用于操作数据库的方法，在该类中定义删除存储过程的方法executeUpdate()，该方法包含一个String数组参数，表示删除的存储过程集合。代码如下：

```
public boolean executeUpdate(String[] sql) {
  conn=getConn();
  //获取数据库连接
  try {
    Statement stmt=
conn.createStatement();                          //实例化
Statement对象
    for (int i = 0; i < sql.length; i++) {
      stmt.executeUpdate("DROP PROCEDURE
  "+sql[i]);              //执行删除操作
    }
    conn.close();
    //关闭连接
  } catch (SQLException e) {
    e.printStackTrace();
    return false;
```

```
    }
    return true;
}
```

举一反三

根据本实例，读者可以开发以下程序。

删除数据库表中的存储过程。

添加存储过程。

6.2 使用触发器

实例236 应用触发器添加日志信息

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\06\Ex06_236

实例说明

触发器是一种特殊类型的存储过程，是为响应数据库操作语句DML 事件或数据定义语言DDL事件而执行的存储过程。当用户对表进行相应操作时，触发器启动执行。触发器可以基于表，也可以基于视图。SQL支持3种触发器，即INSERT、UPDATE和DELETE。本实例应用的是 INSERT 触发器，实现当向用户表中插入信息后，同时将该用户的登录时间插入日志表（tb_info）中。实例运行效果如图6.5所示。

图6.5 实例运行效果

技术要点

实现本实例，首先需要在数据库中创建触发器，这样当执行相应的数据操作后，会自动调用触发器。SQL 语句中使用命令CREATE TRIGGER 语句来创建触发器。具体语法如下：

```
CREATE TRIGGER trigger_name
ON { table | view }
[ WITH ENCRYPTION ]
{
  {{FOR|AFTER|INSTEAD OF}{[INSERT][,][UPDATE]}
    [ WITH APPEND ]
    [ NOT FOR REPLICATION ]
    AS
    [ { IF UPDATE ( column )
      [ { AND | OR } UPDATE ( column ) ]
        [ ...n]
    | IF ( COLUMNS_UPDATED ( ) { bitwise_operator }
updated_bitmask )
      { comparison_operator } column_bitmask [ ...n]
    } ]
    sql_statement [ ...n]
  }
}
```

参数说明：

● trigger_name：所要创建的触发器的名称。

● table|view：创建触发器所在的表或视图，也可以称为触发器表或触发器视图。

● AFTER：指定触发器只有在完成指定的所有SQL 语句之后才会被触发。

● AS：触发器要执行的操作。

● sql_statement：触发器的条件或操作。触发器条件指定其他准则，以确定DELETE、INSERT或UPDATE语句是否导致执行触发器。

本实例实现在SQL Server 数据库下创建触发器，该数据库中有两个非常有用的临时表，分别为 INSERTED 和 DELETED。当由插入操作产生触发器时，会将插入的数据先存储在INSERTED临时表中；当由删除操作产生触发器时，会将删除的数据先存储在DELETED临时表中；当由更新操作产生触发器时，会将更新前的数据存储在DELETED临时表中，而将更新后的数据存储在INSERTED临时表中。了解了这两个临时表，对实现本实例中涉及的触发器知识非常重要。

实现过程

（1）在数据库中创建触发器实现当在用户表（tb_user）中添加数据后，系统会在日志表（tb_info）中添加信息。数据库中创建触发器的代码如下：

```
create trigger triInfoInsert on tb_user
for insert
as
  declare @leavePerson varchar(20)
  select @leavePerson = userName from inserted
  insert tb_info (userName,info) values
(@leavePerson,getDate())
```

（2）在项目中创建类InsertUserFrame，该类继承自JFrame类，实现窗体类。向该窗体中添加标签、文本框、按钮控件，实现窗体布局。该窗体中的主要控件及说明如表6.4所示。

表6.4 窗体中的主要控件及说明

控 件 类 型	控 件 命 名	控 件 用 途
JTextField	userNameTextField	显示“用户名”的文本框控件
	passwordTextField	显示“密码”的文本框控件
	ageTextField	显示“年龄”的文本框控件
	jobTextField	显示“工作”的文本框控件
JButton	insertButton	显示“添加”的按钮控件
	closeButton	显示“关闭”的按钮控件
JComboBox	sexComboBox	显示“性别”的下拉列表控件

（3）在项目中创建工具类 UserTrigger ，在该类中定义向用户表中插入数据的方法insertInfo()，当该方法被调用时，将会产生触发器。该方法的具体代码如下：

```
public void insertInfo(User user){
  conn=getConn();
      //获取数据库连接
  try {
    PreparedStatement statement =
conn.prepareStatement("insert into tb_user
values(?,?,?,?,?)");
    //定义添加数据的SQL语句
    statement.setString(1,
user.getUserName());                          //设置预处
理语句的参数值
    statement.setString(2, user.getPassword());
    statement.setInt(3, user.getAge());
    statement.setString(4,user.getSex());
    statement.setString(5, user.getJob());
    statement.executeUpdate();
          //执行预处理语句
```

```
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
```

举一反三

根据本实例，读者可以实现以下功能。

创建触发器。

删除触发器。

实例237 在删除成绩表时将学生表中的数据删除

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\06\Ex06_237

实例说明

在实际开发中，经常会涉及两个有关联的表，如学生表与学生成绩表，这样当对一张表进行操作时，就需要考虑到另一张表的数据变化，对于这种情况应用触发器是很方便的，也可以很好地维护数据完整性，避免由于程序员的疏忽导致的错误。本实例实现创建DELETE触发器，实现当用户从学生成绩表中删除数据时，会产生触发器，实现从学生信息表中删除记录。实例运行效果如图6.6所示。

(a)

(b)

图6.6 实例运行效果

技术要点

本实例实现的是创建DELETE触发器，DELETE触发器的工作流程是：

当触发DELETE触发器后，从特定的表中删除的行将被放置到一个特殊的DELETED表中。DELETED表示一个逻辑表，保留已被删除数据行的一个副本。DELETED表还允许引用由初始化DELETE语句产生的日志数据。

使用DELETE触发器时，需要考虑以下事项和原则。

□ 当某行被添加到DELETED 表中时，就不再存在于数据表中，因此，DELETED 表和数据库没有相同的行。

□ 创建DELETED表时，控件是从内存中分配的。DELETED表总是被存储在高速缓存中。

实现过程

（1）在数据库中创建触发器triGradeDelete，实现删除tb_grade表中的数据，同时也将tb_stu表中对应的数据删除。具体代码如下：

```
create trigger triGradeDelete on tb_grade
for delete
  as
    declare @name varchar(10)
  select @name = name from deleted
  delete from tb_stu where tb_stu.name =@name
```

（2）在项目中创建类FindStuFrame和GradeFrame，它们都继承自JFrame类，实现窗体类， FindStuFrame类用于显示查询的学生信息，GradeFrame类用于显示学生成绩信息。分别向这两个窗体中添加表格、按钮的控件，实现窗体布局。

（3）在项目中创建工具类DeleteGrade，用于定义删除学生成绩表中信息的方法，当该方法被调用时，会调用触发器。在该类中定义删除学生成绩表中数据的方法 deleteGrade()，该方法有一个int类型的参数，用于指定要删除的学生成绩的编号。该方法的具体代码如下：

```
public void deleteGrade(int id){
  conn=getConn();
            //获取数据库连接
  try {
    Statement statement=
conn.createStatement();                                    //定义Statement方法
    statement.executeUpdate("delete from tb_grade where id="+id);                    //执行删除操作
  } catch (SQLException e) {
    e.printStackTrace();
  }
}
```

举一反三

根据本实例，读者可以实现以下功能。

使用触发器删除表中的数据。

使用触发器删除表中的关联关系。

实例238 创建带有触发条件的触发器

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\06\Ex06_238

实例说明

和查询语句一样，在触发器中也可以添加触发条件，来确保只有在满意的条件下才会执行响应的操作。触发器的使用就是保证参照完整性和数据的一致性。本实例实现的是为学生成绩表 tb_grade 创建触发器，当向该表中添加数据时会调用触发器，在触发器中会做出判断，如果添加的学生姓名在学生信息表中不存在，则不允许添加数据。实例运行效果如图6.7所示。

图6.7 实例运行效果

技术要点

当通过该窗体向数据库中添加数据时，将触发triStuSelect触发器，如果添加的学生姓名在学生信息表中不存在，将执行rollback transaction 语句，取消工作，可以基于一张表创建多个触发器，DBMS把同一个表中所有触发器看作同一事务的一部分。因此，只要其中一个触发器执行了ROLLBACK TRANSACTION 语句，那么所有的操作都将被取消。

在存储过程中添加IF语句的语法格式如下：

If<条件表达式>

{命令行|程序块}

其中“条件表达式”可以是各种表达式的组合，但表达式的值必须是逻辑值“真”或“假”。命令行和语句块可以是合法的 T_SQL 任意语句，但含两条或两条以上的语句的程序块必须加begin…end子句。

实现过程

（1）在数据库中创建触发器triStuSelect。具体代码如下：

```
create trigger triStuSelect
  on tb_grade
  after insert as
    declare @name varchar(10)
    select @name = name from inserted
    if(@name not in (select name from tb_stu))
begin
  rollback transaction
  print ('输入的工资标号错误，请重新输入')
end
```

（2）在项目中创建类InsertTriggerFrame，该类继承自JFrame类，实现窗体类。向该窗体中添加标签、文本框与按钮控件，主要控件及说明如表6.5所示。

表6.5 窗体中的主要控件及说明

控 件 类 型	控 件 命 名	控 件 用 途
JTextField	nameTextField	显示“姓名”的文本框控件
	mathTextField	显示“数学”的文本框控件
	englishTextField	显示“英语”的文本框控件
	chineseTextField	显示“语文”的文本框控件
JButton	insetButton	显示“添加”的按钮控件
	closeButton	显示“关闭”的按钮控件

（3）在项目中创建工具类UserTrigger，在该类中定义添加数据方法，当该方法被调用时，触发器将被执行。该方法的代码如下：

```
public int insertGrade(Grade grade) {
  conn=getConn();
  //获取数据库连接
  PreparedStatement
cs=null;                                    //定义
PreparedStatement对象
  int count = 0;
  try {
    String sql= "insert into tb_grade
values(?,?,?,?)";                    //定义插入SQL语句
    cs = conn.prepareStatement(sql);
    cs.setString(1,
grade.getName());                           //设置预处
理语句参数
    cs.setFloat(2, grade.getMath());
    cs.setFloat(3,grade.getEnglist());
    cs.setFloat(4, grade.getChinese());
    count=
cs.executeUpdate();                                 //执
行预处理语句，实现插入操作
  } catch (SQLException e) {
    e.printStackTrace();
  }
  return count;
}
```

举一反三

根据本实例，读者可以实现以下功能。

创建触发器。

修改触发器的触发条件。

6.3 批处理的应用

实例239 使用批处理删除数据

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\06\Ex06_239

实例说明

如今的数据库处理数据的速度是惊人的，每次执行的吞吐量非常大，执行效率非常高，这时速度的瓶颈也随之出现在数据库的连接传输上。在Java中，每当需要执行SQL语句时都要创建一个数据库连接，并把要执行的SQL语句传送到数据库服务器。时间都消耗在了数据库的连接传输上，如果把要执行的SQL语句转载在一起，一次性发送给数据库执行，会大大地提高执行效率。本实例以删除学生信息为例，为大家介绍如何使用JDBC批量删除数据。实例运行效果如图6.8所示。

图6.8 实例运行效果

技术要点

本实例使用了Statement接口，实现数据的批量处理操作，该接口中包含两个很重要的方法，来执行批量处理操作，分别为addBatch()方法和executeBatch()方法。下面分别对这两个方法进行介绍。

□ addBatch()方法

该方法将给定的SQL语句添加到Statement对象当前命令列表中。声明语法如下：

addBatch(String sql)

参数说明

sql：标准的SQL语句。

□ executeBatch()方法

该方法将一批命令提交给数据库来执行。该方法的语法格式如下：

ex executeBatch()

该方法的返回值是每个命令的一个元素的更新计数所组成的数据。

实现过程

（1）在项目中创建类DeleteStuFrame，该类继承自JFrame类，实现窗体类。在该窗体中添加标签、表格与按钮控件，表格控件完成显示学生信息，按钮控件为用户提供操作信息。

（2）在项目中创建工具类BatchDelete，在该类中定义批量删除数据的方法deleteBatch()，该方法包含一个Integer类型的参数。具体代码如下：

```
public void deleteBatch(Integer[] id) {
  conn=getConn();
  //获取数据库连接
  Statement
cs=null;                                          //定义
Statement对象
  try {
```

```
        cs=
    conn.createStatement();
    //实例化Statement对象
        for (int i=0; i< id.length; i++)
    {                                   //循环遍历参数数组
          cs.addBatch("delete from tb_stu where id="+
      id[i]);          //删除数据
        }
        cs.executeBatch();
          //批量执行SQL语句
        cs.close();
        //将Statement对象关闭
        conn.close();
      } catch (Exception e) {
        e.printStackTrace();
      }
    }
```

（3）在用户选择要删除的信息后，单击“删除”按钮，可将用户选择的信息删除。“删除”按钮的单击事件代码如下：

```
    protected void
do_deleteButton_actionPerformed(ActionEvent arg0) {
        int [] ids=
  table.getSelectedRows();                                    //返
  回选定行的索引
        Integer values[] = new Integer[ids.length];
        for(int i=0;i<ids.length;i++)
  {                                       //遍历选定行的数组
```

```
        //获取用户选择某单元格的内容
        values[i] = new Integer(table.getValueAt(ids[i],
    0).toString());
      }
      batchDelete.deleteBatch(values);
              //调用批处理方法
      JOptionPane.showMessageDialog(getContentPane(),"数据删
    除成功！", "信息提示框", JOptionPane.WARNING_MESSAGE);
    }
```

举一反三

根据本实例，读者可以实现以下功能。

使用批处理删除数据。

使用批处理修改数据。

实例240 使用批处理提升部门员工工资

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\06\Ex06_240

实例说明

由于在一个批处理语句中可以执行多条SQL语句，因此在实际开发中批处理被应用得十分广泛，本实例应用批处理技术实现提升某些部门员工的工资。实例运行效果如图6.9所示。

图6.9 实例运行效果

技术要点

本实例实现批量修改工资表中员工的工资，仍然是使用Statement类中的批处理方法。本实例将部门信息与要提升的工资使用JList控件进行显示，其中提升部门信息可实现多选，而提升工资只能进行单选。设置JList控件的选择模式，可以使用setSelectionMode()方法，JList的默认模式是可进行多选；要设置 JList 的单选，可调用setSelectionMode()方法，并将该方法的参数设置为0即可。

实现过程

（1）在项目中创建类UpdateBatchFrame，该类继承自JFrame类，实现窗体类。向该窗体中添加列表控件、按钮控件，实现窗体布局，主要控件及说明如表6.6所示。

表6.6 窗体中的主要控件及说明

控 件 类 型	控 件 命 名	控 件 用 途
JList	deptlist	显示“提升部门”的列表控件
	laboragelist	显示“提升工资”的列表控件
JButton	okButton	显示“确定”的按钮控件
	closeButton	显示“关闭”的按钮控件

（2）在项目中创建工具类 BatchUpdate ，在该类中定义批量修改工资表中数据的方法updateBatch()，该方法包含一个Object类型的参数，用于定义修改的部门数组；一个int类型参数，用于指定增加的工资额度。该方法的具体代码如下：

```
public void updateBatch(Object[] dept, int laborage) {
  conn=getConn();
  //获取数据库连接
  Statement
cs=null;                                          //定义
Statement对象
```

```
    try {
      cs=
conn.createStatement();
//实例化Statement对象
      for (int i = 0; i < dept.length; i++) {
        cs.addBatch("update tb_laborage set laborage =
laborage +"+ laborage +" where dept = '"+ dept[i]
+"'");//修改数据
      }
      cs.executeBatch();
        //批量执行SQL语句
      cs.close();
      //将Statement对象关闭
      conn.close();
    } catch (Exception e) {
      e.printStackTrace();
    }
}
```

（3）当用户选择了要修改的部门和调整的工资额度后，单击“确定”按钮，可修改制定部门的工资。“确定”按钮的单击事件的代码如下：

```
protected void do_okButton_actionPerformed(ActionEvent
arg0) {
    Object[]dept=deptlist.getSelectedValues();
        //获取用户选择的要增加工资的所有部门数组
    //获取用户选择的增加工资的额度
```

```
    String laborage =
laboragelist.getSelectedValue().toString();
    if (dept.length>0&& !laborage.equals(""))
{                    //如果用户选择的信息不为空
      update.updateBatch(dept,
  Integer.parseInt(laborage));          //调用批量修改方法
    }
    JOptionPane.showMessageDialog(getContentPane(),"数据修
改成功！", "信息提示框", JOptionPane.WARNING_MESSAGE);
  }
```

举一反三

根据本实例，读者可以实现以下功能。

使用批处理修改员工的个人信息。

使用批处理删除员工。

实例241 将教师表中的数据全部添加到选课表

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\06\Ex06_241

实例说明

使用批处理可快速地完成相关的操作。例如，学生选课表和教师表是两个有关联关系的数据表，其中教师表中的数据全部都包含在选课表中。要实现快速地将教师表中的数据添加到选课表中，可以使用批处理。本实例向大家介绍的是使用批处理技术，将教师表中的数据全部添加到选课表中。本实例执行完毕后，选课表中的数据如图6.10所示。

id	elective	teacher	classRoom
11	计算机科学	张丹	待定
12	古典文学	赵华	待定
13	现代汉语	陈双	待定
14	儿童文学	李静	待定
15	微格教学	赵梅	待定

图6.10 选课表中的数据

技术要点

实现本实例，首先要将教师表中的数据检索出来，再通过批处理将教师表中的数据添加到选课表。仍然是使用Statement接口中的addBatch()方法与executeBatch()方法。

实现过程

（1）在项目中创建类 BatchInsert，在该类中定义批处理方法。首先定义连接数据库的方法getConn()，该方法以Connection对象作为返回值，读者可参考光盘中的源程序，这里不再赘述。

（2）在该类中定义executeTeacher()方法，实现获取教师表中所有数据的方法，该方法以List作为返回值。具体代码如下：

```
public List executeTeacher() {
  conn=getConn();                                    //获取
数据库连接
  Statement
cs=null;                                             //定义
CallableStatement对象
  String sql= "select * from
tb_teacher";                          //定义调用存储过程的SQL
语句
  List list = new ArrayList();
  try {
    cs=
conn.createStatement();                              //实例
```

```
化Statement对象
    ResultSet rest=
cs.executeQuery(sql);                          //执行SQL语句
    while (rest.next())
{                                              //循环遍历查询结果集
      Teacher teacher=new
  Teacher();                                   //定义与数据库表对应的
  JavaBean对象
      teacher.setId(rest.getInt(1));
      //设置对象的参数值
      teacher.settName(rest.getString(2));
      teacher.setCourse(rest.getString(3));
      list.add(teacher);                                    /
  /向集合中添加对象
    }
  } catch (SQLException e) {
    e.printStackTrace();
  }
  return list;
}
```

（3）在该类中定义批量将教师表中的数据添加到选课表。具体代码如下：

```
public void insertBatch() {
  conn=getConn();                                          //
获取数据库连接
    Statement
cs=null;                                          //定义
```

Statement对象

```
    try {
      cs=
  conn.createStatement();                                    //实例
  化Statement对象
      List list = executeTeacher();
      for (int i = 0; i < list.size(); i++) {
        Teacher teacher = (Teacher) list.get(i);
        cs.addBatch("insert into tb_elective values ('"+
    teacher.getCourse() +"','"+ teacher.gettName()+"','待
    定')");        //添加SQL语句
      }
      cs.executeBatch();
    //批量执行SQL语句
      cs.close();
  //将Statement对象关闭
      conn.close();
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
```

举一反三

根据本实例，读者可以实现以下功能。

批量修改选课表中的数据。

批量增加教师的所教课程。

6.4 使用视图

实例242 使用视图过滤不想要的数据

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\06\Ex06_242

实例说明

视图显示了“伪表”，创建这些表是为了以特定的形式显示数据库的内容。使用视图可限制用户访问敏感的数据，帮助用户执行复杂的SQL语句。在Java程序中实现从视图中查询数据，与从普通表中查询数据的方法相同。本实例实现在数据库中创建视图，并通过Java程序将视图中的数据查询出来。实例运行效果如图6.11所示。

图6.11 实例运行效果

技术要点

如果已成功地创建视图，Java程序可以直接从视图中查询数据，方法与普通表一样。从视图中查询数据的SQL语句的语法如下：

SELECT column_name from viewName

参数说明

- column_name：从视图中查询的字段。
- viewName：要查询的视图名称。

实现过程

（1）在数据库中编写代码创建视图，具体代码如下：

```
create view v_laborage
as
  select id,name,dept,laborage from tb_laborage
```

（2）在项目中创建类UserViewFrame，该类继承自JFrame类，实现窗体类。在该窗体中添加标签、表格控件，实现窗体布局。

（3）在项目中编写工具类UserView，在该类中定义查询视图的方法。具体代码如下：

```
public List selectView() {
  conn=getConn();                                    //
获取数据库连接
   Statement
cs=null;                                          //定义
CallableStatement对象
   String sql= "Select * from
v_laborage";                              //定义查询视图的SQL语句
   List list=new
ArrayList();                                      //定义保存查询
结果的List集合
   try {
```

```
    cs=
conn.createStatement();                                    //实例
化Statement对象
    ResultSet rest=
cs.executeQuery(sql);                                  //执行SQL语句
    while (rest.next())
{                                                  //循环遍历查询结果集
      Laborage laborage = new Laborage();
      laborage.setId(rest.getInt(1));
      laborage.setName(rest.getString(2));
      laborage.setDept(rest.getString(3));
      laborage.setLaborage(rest.getString(4));
      list.add(laborage);
    }
  } catch (SQLException e) {
    e.printStackTrace();
  }
  return list;
}
```

举一反三

根据本实例，读者可以实现以下功能。

创建视图。

使用视图展示数据。

实例243 使用视图计算数据

这是一个可以用来提高基础性能的实例

实例位置：光盘\mingrisoft\06\Ex06_243

实例说明

利用视图可以简化用户对数据的操作。在视图中进行一些复杂的操作，可以保证程序的安全。本实例向大家介绍的是在视图中计算商品的利润，之后在Java程序中查询视图，这样就避免了对数据表进行操作。实例运行效果如图6.12所示。

图6.12 实例运行效果

技术要点

AS关键字后加定义视图的一个完整的 SELECT查询语句，它可以应用多个表。这个子查询还可以包括单行函数和聚集函数、WHERE 子句和GROUP BY 子句、嵌套的子查询等。不过不能包含ORDER BY 子句。这个子查询的结果将是所创建的视图的内容。

```
USE db_sql
GO
CREATE VIEW ware_v
AS
SELECT TOP 100 PERCENT *
FROM tb_ware14
ORDER BY 售价
GO
```

在SQL Server 中引入的TOP 结构与ORDER BY 子句结合使用是非常有用的。只有在与TOP 关键词结合使用时，SQL Server 才支持在视图中使用ORDER BY 子句。

实现过程

（1）在数据库中创建视图，实现从商品表中查询商品名称和商品利润。具体代码如下：

```
create view v_ware(wName,profit)
as
  select wName,(price - inPrice)as profit from tb_ware
```

（2）在项目中创建类GetProfitFrame，该类继承自JFrame类，实现窗体类。向该窗体中添加标签与表格控件，实现窗体布局。标签控件用于给用户提供提示信息，表格控件用于显示查询结果。

（3）在项目中创建工具类 UserViewData ，在该类中定义从视图中查询数据的方法selectView()，该方法将查询结果以List集合返回。具体代码如下：

```
public List selectView() {
  conn=getConn();                                    //
获取数据库连接
   Statement
cs=null;                                          //定义
Statement对象
   String sql= "Select * from
v_ware";                                 //定义查询视图的SQL语句
   List list=new
ArrayList();                                      //定义保存查询
结果的List集合
   try {
```

```
    cs=
conn.createStatement();                                    //实例
化Statement对象
    ResultSet rest=
cs.executeQuery(sql);                                  //执行SQL语句
    while (rest.next())
{                                                  //循环遍历查询结果集
      Ware ware = new Ware();
      ware.setwName(rest.getString(1));
      ware.setProfit(rest.getString(2));
      list.add(ware);
    }
  } catch (SQLException e) {
    e.printStackTrace();
  }
  return list;
}
```

举一反三

根据本实例，读者可以实现以下功能。

创建视图。

对视图进行数据修改操作。

实例244 修改视图

这是一个可以提高基础性能的实例

实例位置：光盘\mingrisoft\06\Ex06_244

实例说明

对于创建成功的视图，可以通过 ALTER 关键字进行修改。本实例为大家介绍的是对已创建的视图进行修改。实例运行效果如图6.13所示。

图6.13 实例运行效果

技术要点

使用SQL 语句中的ALTER VIEW 语句可以修改已创建的视图。语法格式如下：

```
ALTER VIEW [ < database_name > .] [ < owner > .]
view_name [ ( column [ ,...n ] ) ]
[ WITH < view_attribute > [ ,...n ] ]
AS
  select_statement
[ WITH CHECK OPTION ]
```

参数说明

● view_name：表示所要修改的视图名称。

● column：表示一列或多个列的名字，列之间用逗号分隔，指定视图所显示的列。

实现过程

（1）在项目中创建类UpdateViewFrame，该类继承自JFrame类，实现窗体类。向该窗体中添加标签、文本域与按钮控件。标签控件用于为用户提供信息，文本域控件为用户提供修改视图代码，按钮控件用于给用户提供相关的操作。

（2）在项目中创建类 UpdateView，用于定义执行 SQL 语句代码。在该类中首先定义连接数据库的方法getConn()，该方法读者可参考光盘中的源程序。

（3）在UpdateView类中定义执行SQL语句的方法executeUpdate()，该方法的具体代码如下：

```
public boolean executeUpdate(String sql) {
  if (conn == null) {
    getConn();
        //获取数据库连接
  }
  try {
    Statement stmt=
conn.createStatement();                              //创建
Statement实例
    int iCount=
stmt.executeUpdate(sql);                                  /
/执行SQL语句
    System.out.println("操作成功，所影响的记录数为"+
String.valueOf(iCount));
```

```
            conn.close();
                  //关闭连接
        } catch (SQLException e) {
            System.out.println(e.getMessage());
            return false;
        }
        return true;
    }
```

举一反三

根据本实例，读者可以实现以下功能。

建立视图，使用户只能看到标有自己用户名的行。

把视图授权给其他用户。

在表中增加一个标志用户名的列。

实例245 删除视图

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\06\Ex06_245

实例说明

通过 Java 程序不仅可以创建、修改数据库视图，还可以将没用的视图删除，以便节省数据库的空间。本实例主要介绍动态删除视图。首先将数据库中的所有视图都显示在窗体中，这样方便用户删除不想要的视图。单击“删除”按钮即可完成删除操作。实例运行效果如图6.14所示。

图6.14 实例运行效果

技术要点

本实例使用SQL 语句的DROP VIEW 删除已创建的视图。语法格式如下：

DROP VIEW [view name] [,…n]

参数说明

view name：要删除的视图名称。

实现过程

（1）在项目中创建类DeleteViewFrame，该类继承自JFrame类，实现窗体类。在该窗体中添加表格、标签与按钮控件。标签控件用于为用户提供提示信息，表格控件用于显示视图信息，按钮控件用于给用户提供相关操作。

（2）在项目中创建工具类DeleteView，在该类中定义删除视图的方法，该方法以String类型数组为参数，实现一次可删除多个视图。具体代码如下：

```
public boolean executeUpdate(String[] sql) {
```

```
  conn=getConn();
  //获取数据库连接
  try {
    Statement stmt=
conn.createStatement();                          //实例化
Statement对象
    for (int i = 0; i < sql.length; i++) {
      stmt.executeUpdate("DROP VIEW "+
  sql[i]);                  //执行删除操作
    }
    conn.close();
    //关闭连接
  } catch (SQLException e) {
    e.printStackTrace();
    return false;
  }
  return true;
}
```

举一反三

根据本实例，读者可以实现以下功能。

创建视图。

修改视图。

第7章 图形图像技术

绘制图形和文本
图形处理
绘制图案
图像处理
颜色处理
文字特效
图片特效
其他

7.1 绘制图形和文本

实例246 绘制直线

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\07\Ex07_246

实例说明

在几何中，直线是向两端无限延伸的，本实例所说的绘制直线，实际上是绘制直线上两点之间的线段，线段在实际生产和生活中经常使用。运行程序，将在窗体上绘制线段，效果如图7.1所示。

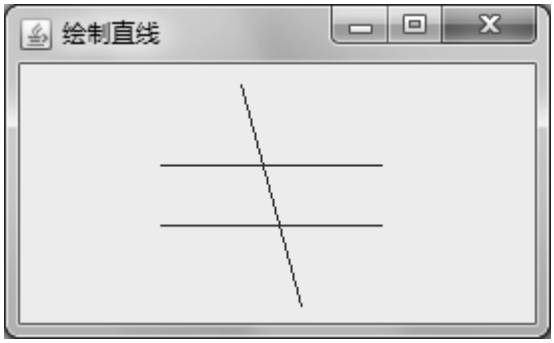


图7.1 绘制直线

技术要点

本实例主要是通过在JPanel类的子类中，重写JComponent类的paint()方法，并在该方法中使用Graphics类的drawLine()方法来实现的。

（1）在JPanel类的子类中，重写JComponent类的paint()方法，该方法的定义如下：

```
public void paint(Graphics g)
```

参数说明

g：图形上下文对象，用于绘制基本的形状和文本。

（2）使用Graphics类的drawLine()方法绘制直线，该方法的定义如下：

public abstract void drawLine(int x1, int y1, int x2, int y2)

参数说明

- x1：第1 个点的x 坐标。
- y1：第1 个点的y 坐标。
- x2：第2 个点的x 坐标。
- y2：第2 个点的y 坐标。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的DrawLineFrame窗体类。

（3）在DrawLineFrame窗体类中创建内部面板类DrawLinePanel，并重写JComponent类的paint()方法，在该方法中使用Graphics类的drawLine()方法绘制直线。

（4）将内部面板类DrawLinePanel的实例添加到窗体类DrawLineFrame的内容面板上，用于在窗体上显示绘制的直线，代码如下：

```
class DrawLinePanel extends JPanel {          //创建内部面板类
    public void paint(Graphics g) {           //重写paint()方法
        g.drawLine(70,50,180,50);             //绘制第1条水平线
        g.drawLine(70,80,180,80);             //绘制第2条水平线
```

```
        g.drawLine(110,10,140,120);              //绘制斜线
    }
}
```

举一反三

根据本实例，读者可以开发以下程序。

绘制长方形。

绘制三角形。

实例247 绘制矩形

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\07\Ex07_247

实例说明

矩形在实际生产和生活中经常使用，例如书桌的桌面、房屋的门窗等，本实例将通过绘制矩形让读者初步了解Java绘图技术。运行程序，将在窗体上绘制矩形，效果如图7.2所示。

图7.2 绘制矩形

技术要点

本实例主要是通过在JPanel类的子类中，重写JComponent类的paint()方法，并在该方法中使用Graphics类的drawRect()和fillRect()方法来实现的。

（1）使用Graphics类的drawRect()方法绘制的矩形，只有线条而没有填充色，该方法的定义如下：

public abstract void drawRect(int x, int y, int width, int height)

参数说明

- x：矩形左上角的x 坐标。
- y：矩形左上角的y 坐标。
- width：矩形的宽度。
- height：矩形的高度。

（2）使用Graphics类的fillRect()方法可绘制带填充色的矩形，该方法的定义如下：

public abstract void fillRect(int x, int y, int width, int height)

参数说明

- x：填充矩形左上角的x 坐标。
- y：填充矩形左上角的y 坐标。
- width：填充矩形的宽度。
- height：填充矩形的高度。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的DrawRectangleFrame窗体类。

（3）在 DrawRectangleFrame 窗体类中，创建内部面板类DrawRectanglePanel，并重写JComponent类的paint()方法，在该方法中使用Graphics类的drawRect()和fillRect()方法绘制矩形。

（4）将内部面板类DrawRectanglePanel的实例添加到窗体类DrawRectangleFrame的内容面板上，用于在窗体上显示绘制的矩形，

代码如下：

```
    class DrawRectanglePanel extends JPanel {          //创建内部面板类
      public void paint(Graphics g) {          //重写paint()方法
        g.drawRect(30, 40, 80, 60) ;          //绘制空心矩形
        g.fillRect(140, 40, 80, 60);          //绘制实心矩形
      }
    }
```

举一反三

根据本实例，读者可以实现以下功能。

绘制正方形。

绘制田字格。

实例248 绘制椭圆

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\07\Ex07_248

实例说明

本实例演示如何在Java中绘制椭圆。运行程序，将在窗体上绘制椭圆，效果如图7.3所示。

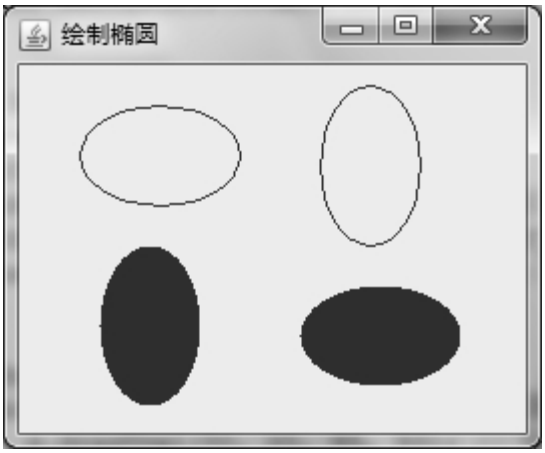


图7.3 绘制椭圆

技术要点

本实例主要是通过在JPanel类的子类中，重写JComponent类的paint()方法，并在该方法中使用Graphics类的drawOval()和fillOval()方法来实现的。

（1）使用Graphics类的drawOval()方法绘制的椭圆，只有线条而没有填充色，该方法的定义如下：

```
public abstract void drawOval(int x, int y, int width, int height)
```

参数说明

- x：要绘制椭圆的左上角的x 坐标。
- y：要绘制椭圆的左上角的y 坐标。
- width：要绘制椭圆的宽度。
- height：要绘制椭圆的高度。

（2）使用Graphics类的fillOval()方法可绘制带填充色的椭圆，该方法的定义如下：

```
public abstract void fillOval(int x, int y, int width, int height)
```

参数说明

- x：要填充椭圆的左上角的x 坐标。

- y：要填充椭圆的左上角的y 坐标。
- width：要填充椭圆的宽度。
- height：要填充椭圆的高度。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的DrawEllipseFrame窗体类。

（3）在DrawEllipseFrame窗体类中，创建内部面板类DrawEllipsePanel，并重写JComponent类的paint()方法，在该方法中使用Graphics类的drawOval()和fillOval()方法绘制椭圆。

（4）将内部面板类DrawEllipsePanel的实例添加到窗体类DrawEllipseFrame的内容面板上，用于在窗体上显示绘制的椭圆，代码如下：

```
class DrawEllipsePanel extends JPanel {          //创建内部面板类
    public void paint(Graphics g) {          //重写paint()方法
        g.drawOval(30, 20, 80, 50);          //绘制空心椭圆
        g.drawOval(150, 10, 50, 80);          //绘制空心椭圆
        g.fillOval(40, 90, 50, 80);          //绘制实心椭圆
        g.fillOval(140, 110, 80, 50);          //绘制实心椭圆
    }
}
```

举一反三

根据本实例，读者可以实现以下功能。

绘制简易小人。

绘制笑脸。

实例249 绘制圆弧

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\07\Ex07_249

实例说明

本实例演示如何在Java中绘制圆弧。运行程序，将在窗体上绘制圆弧，效果如图7.4所示。

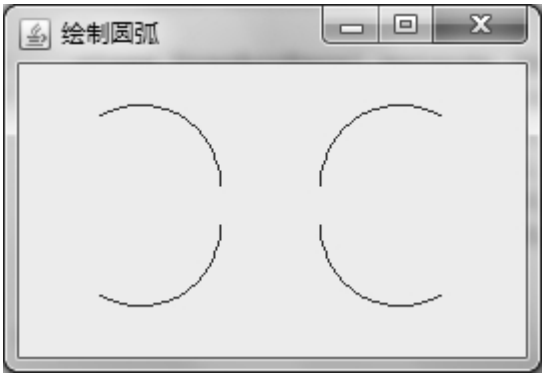


图7.4 绘制圆弧

技术要点

本实例主要是通过在JPanel类的子类中，重写JComponent类的paint()方法，并在该方法中使用Graphics类的drawArc()方法来实现的。

drawArc()方法的定义如下：

```
public abstract void drawArc(int x, int y, int width, int height, int startAngle, int arcAngle)
```

参数说明

- x：要绘制弧的左上角的x 坐标。
- y：要绘制弧的左上角的y 坐标。
- width：要绘制弧的宽度。
- height：要绘制弧的高度。
- startAngle：开始角度。

● arcAngle：相对于开始角度而言，弧跨越的角度。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的DrawArcFrame窗体类。

（3）在DrawArcFrame窗体类中，创建内部面板类DrawArcPanel，并重写JComponent类的paint()方法，在该方法中使用Graphics类的drawArc()方法绘制圆弧。

（4）将内部面板类DrawArcPanel的实例添加到窗体类DrawArcFrame的内容面板上，用于在窗体上显示绘制的圆弧，代码如下：

```
class DrawArcPanel extends JPanel {          //创建内部面板类
  public void paint(Graphics g) {          //重写paint()方法
    g.drawArc(20, 20, 80, 80, 0, 120);          //绘制圆弧
    g.drawArc(20, 40, 80, 80, 0, -120);          //绘制圆弧
    g.drawArc(150, 20, 80, 80, 180, -120);          //绘制圆弧
    g.drawArc(150, 40, 80, 80, 180, 120);          //绘制圆弧
  }
}
```

举一反三

根据本实例，读者可以开发以下程序。

绘制扇形。

绘制正方形。

实例250 绘制指定角度的填充扇形

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\07\Ex07_250

实例说明

本实例演示如何在 Java 中绘制指定角度的填充扇形。运行程序，将在窗体上绘制填充扇形，效果如图7.5所示。

图7.5 绘制指定角度的填充扇形

技术要点

本实例主要是通过在JPanel类的子类中，重写JComponent类的paint()方法，并在该方法中使用Graphics类的fillArc()方法来实现的。

fillArc()方法的定义如下：

public abstract void fillArc(int x, int y, int width, int height, int startAngle, int arcAngle)

参数说明

- x：要绘制填充扇形的左上角的x 坐标。
- y：要绘制填充扇形的左上角的y 坐标。
- width：要绘制填充扇形的宽度。
- height：要绘制填充扇形的高度。
- startAngle：开始角度。

● arcAngle：相对于开始角度而言，填充扇形的弧跨越的角度。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的DrawSectorFrame窗体类。

（3）在DrawSectorFrame窗体类中，创建内部面板类DrawSectorPanel，并重写JComponent类的paint()方法，在该方法中使用Graphics类的fillArc()方法绘制填充扇形。

（4）将内部面板类DrawSectorPanel的实例添加到窗体类DrawSectorFrame的内容面板上，用于在窗体上显示绘制的填充扇形，代码如下：

```
class DrawSectorPanel extends JPanel {          //创建内部面板类
  public void paint(Graphics g) {          //重写paint()方法
    g.fillArc(40, 20, 80, 80, 0, 150);          //绘制填充扇形
    g.fillArc(140, 20, 80, 80, 180, -150);          //绘制填充扇形
    g.fillArc(40, 40, 80, 80, 0, -110);          //绘制填充扇形
    g.fillArc(140, 40, 80, 80, 180, 110);          //绘制填充扇形
  }
}
```

举一反三

根据本实例，读者可以实现以下功能。

绘制饼形图。

绘制填充的扇形。

实例251 绘制多边形

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\07\Ex07_251

实例说明

本实例演示如何在Java中绘制多边形。运行程序，将在窗体上绘制多边形，效果如图7.6所示。

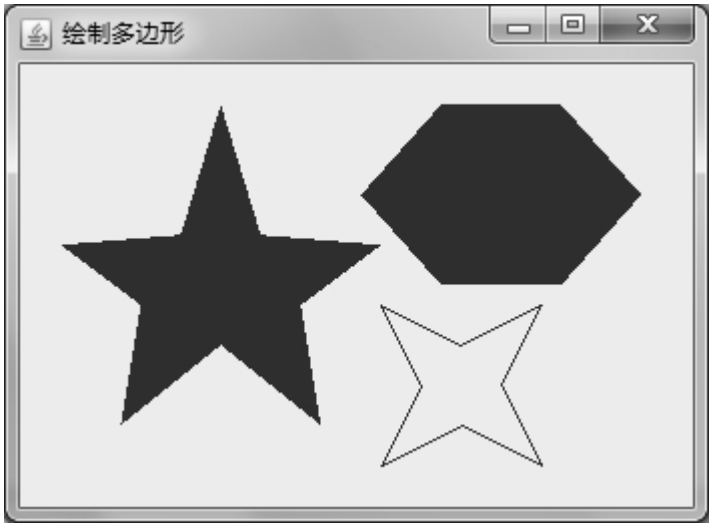


图7.6 绘制多边形

技术要点

本实例主要是通过在JPanel类的子类中，重写JComponent类的paint()方法，并在该方法中使用Graphics类的drawPolygon()和fillPolygon()方法来实现的。

（1）使用Graphics类的drawPolygon()方法绘制的多边形，只有线条而没有填充色，该方法的定义如下：

```
public abstract void drawPolygon(int[] xPoints, int[] yPoints, int nPoints)
```

参数说明

- xPoints：要绘制多边形的x 坐标数组。
- yPoints：要绘制多边形的y 坐标数组。
- nPoints：要绘制多边形的顶点总数。

（2）使用Graphics类的fillPolygon()方法可绘制带填充色的多边形，该方法的定义如下：

```
public abstract void fillPolygon(int[] xPoints, int[] yPoints, int nPoints)
```

参数说明

- xPoints：要绘制填充多边形的x 坐标数组。
- yPoints：要绘制填充多边形的y 坐标数组。
- nPoints：要绘制填充多边形的顶点总数。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的DrawPolygonFrame窗体类。

（3）在 DrawPolygonFrame 窗体类中，创建内部面板类 DrawPolygonPanel ，并重写JComponent类的paint()方法，在该方法中使用Graphics类的drawPolygon()和drawPolygon()方法绘制多边形。

（4）将内部面板类DrawPolygonPanel的实例添加到窗体类DrawPolygonFrame的内容面板上，用于在窗体上显示绘制的多边形，代码如下：

```
class DrawPolygonPanel extends JPanel {          //创建内部面板类
    public void paint(Graphics g) {          //重写paint()方法
```

```
        int[] x1 ={ 100,120,180,140,150,100,50,60,20,80
};     //多边形的横坐标
        int[] y1 ={ 20,85,90,120,180,140,180,120,90,85
};          //多边形的纵坐标
        int n1 = 10;          //多边形的边数
        g.fillPolygon(x1, y1, n1);          //绘制实心多边形
        int[] x2 ={ 210, 270, 310, 270, 210, 170 };          //
多边形的横坐标
        int[] y2 ={ 20, 20, 65, 110, 110, 65 };          //多边
形的纵坐标
        int n2 = 6;          //多边形的边数
        g.fillPolygon(x2, y2, n2);          //绘制实心多边形
        int[] x3 ={ 180, 220, 260, 240, 260, 220, 180, 200
};          //多边形的横坐标
        int[] y3 ={ 120, 140, 120, 160, 200, 180, 200, 160
};          //多边形的纵坐标
        int n3 = 8;          //多边形的边数
        g.drawPolygon(x3, y3, n3);          //绘制空心多边形
    }
}
```

举一反三

根据本实例，读者可以实现以下功能。

快速定义多边形的顶点坐标。

绘制三棱锥。

实例252 绘制二次曲线

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\07\Ex07_252

实例说明

本实例演示如何在Java中绘制二次曲线。运行程序，将在窗体上绘制二次曲线，效果如图7.7所示。

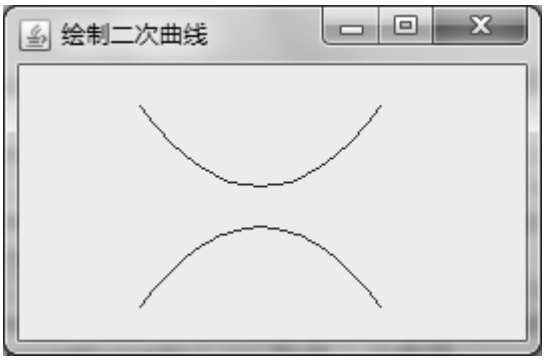


图7.7 绘制二次曲线

技术要点

本实例主要是通过在JPanel类的子类中，重写JComponent类的paint()方法，并在该方法中使用Graphics2D类的draw()方法和使用QuadCurve12D.Double类创建二次曲线对象来实现的。

（1）使用Graphics2D类的draw()方法，并将QuadCurve12D.Double类创建的二次曲线对象作为draw()方法的参数，实现绘制二次曲线的操作，draw()方法的定义如下：

```
public abstract void draw(Shape shape)
```

参数说明

shape：要绘制的形状。

（2）使用QuadCurve12D.Double类创建二次曲线对象，其构造方法的定义如下：

```
public QuadCurve2D.Double(double x1, double y1, double ctrlx, double ctrly, double x2, double y2)
```

参数说明

- x1：起始点的x 坐标。
- y1：起始点的y 坐标。
- ctrlx：控制点的x 坐标。
- ctrly：控制点的y 坐标。
- x2：结束点的x 坐标。
- y2：结束点的y 坐标。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的DrawQuadCurveFrame窗体类。

（3）在 DrawQuadCurveFrame 窗体类中，创建内部面板类DrawQuadCurvePanel，并重写JComponent类的paint()方法，在该方法中使用QuadCurve12D.Double类创建二次曲线对象，并使用Graphics2D类的draw()方法绘制该二次曲线。

（4）将内部面板类DrawQuadCurvePanel的实例添加到窗体类DrawQuadCurveFrame的内容面板上，用于在窗体上显示绘制的二次曲线，代码如下：

```
class DrawQuadCurvePanel extends JPanel {        //创建内部面板类
    public void paint(Graphics g) {          //重写paint()方法
        Graphics2D g2=(Graphics2D)g;          //获得Graphics2D对象
        //创建二次曲线，其中点(120,100)是控制点，点(60,20)是起始点坐标，点(180,20)是终点坐标
        QuadCurve2D.Double quadCurve1 = new QuadCurve2D.Double(60,20,120,100,180,20);
```

```
        g2.draw(quadCurve1);          //绘制二次曲线
        //创建二次曲线，其中点(120,40)是控制点，点(60,120)是起始点坐标，点(180,120)是终点坐标
        QuadCurve2D.Double quadCurve2 = new QuadCurve2D.Double(60,120,120,40,180,120);
        g2.draw(quadCurve2);          //绘制二次曲线
    }
}
```

举一反三

根据本实例，读者可以实现以下功能。

使用QuadCurve12D.Double类创建二次曲线。

使用QuadCurve12D.Float类创建二次曲线。

实例253 绘制三次曲线

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\07\Ex07_253

实例说明

本实例演示如何在Java中绘制三次曲线。运行程序，将在窗体上绘制三次曲线，效果如图7.8所示。

图7.8 绘制三次曲线

技术要点

本实例主要是通过在JPanel类的子类中，重写JComponent类的paint()方法，并在该方法中使用Graphics2D类的draw()方法和使用CubicCurve2D.Double类创建三次曲线对象来实现的。

使用CubicCurve2D.Double类创建三次曲线对象，其构造方法的定义如下：

public CubicCurve2D.Double(double x1, double y1, double ctrlx1, double ctrly1, double ctrlx2, double ctrly2, double x2, double y2)

参数说明

- x1：起始点的x 坐标。
- y1：起始点的y 坐标。
- ctrlx1：第1 个控制点的x 坐标。
- ctrly1：第1 个控制点的y 坐标。
- ctrlx2：第2 个控制点的x 坐标。
- ctrly2：第2 个控制点的y 坐标。
- x2：结束点的x 坐标。
- y2：结束点的y 坐标。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的DrawCubicCurveFrame窗体类。

（3）在 DrawCubicCurveFrame 窗体类中，创建内部面板类DrawCubicCurvePanel，并重写JComponent类的paint()方法，在该方法中使用CubicCurve2D.Double类创建三次曲线对象，并使用Graphics2D类的draw()方法绘制该三次曲线。

（4）将内部面板类DrawCubicCurvePanel的实例添加到窗体类DrawCubicCurveFrame的内容面板上，用于在窗体上显示绘制的三次曲线，代码如下：

```
class DrawCubicCurvePanel extends JPanel {        //创建内部面板类
  public void paint(Graphics g) {          //重写paint()方法
    Graphics2D g2=(Graphics2D)g;        //获得Graphics2D对象
    //创建三次曲线，其中点(140,-140)和点(140,300)是控制点，点(20,80)是起始点坐标，点(260,80)是终点坐标
    CubicCurve2D.Double cubicCurve = new CubicCurve2D.Double(20,80,140,-140,140,300,260,80);
    g2.draw(cubicCurve);          //绘制三次曲线
  }
}
```

举一反三

根据本实例，读者可以实现以下功能。

使用CubicCurve12D.Double类创建三次曲线。

使用CubicCurve12D.Float类创建三次曲线。

实例254 绘制文本

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\07\Ex07_254

实例说明

本实例演示如何在Java中绘制文本。运行程序，将在窗体上绘制文本，效果如图7.9所示。

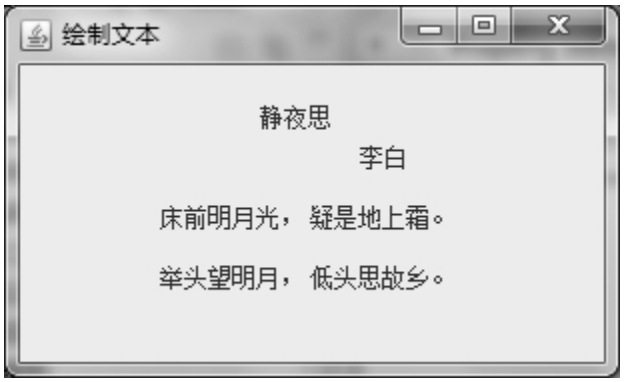


图7.9 绘制文本

技术要点

本实例主要是通过在JPanel类的子类中，重写JComponent类的paint()方法，并在该方法中使用Graphics类的drawString()方法来实现的。

drawString()方法的定义如下：

```
public abstract void drawString(String str, int x, int y)
```

参数说明

- str：绘制的文本内容。
- x：绘制点的x 坐标。
- y：绘制点的y 坐标。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的DrawTextStringFrame窗体类。

（3）在 DrawTextStringFrame 窗体类中，创建内部面板类DrawTextStringPanel，并重写JComponent类的paint()方法，在该方法中使用Graphics类的drawString()方法绘制文本。

（4）将内部面板类 DrawTextStringPanel的实例添加到窗体类DrawTextStringFrame的内容面板上，用于在窗体上显示绘制的文本，代码如下：

```
class DrawTextStringPanel extends JPanel {          //创建内部面板类
    public void paint(Graphics g) {          //重写paint()方法
        String value ="静夜思";
        int x = 120;          //文本位置的横坐标
        int y = 30;          //文本位置的纵坐标
        g.drawString(value, x, y);          //绘制文本
        //省略部分代码
    }
}
```

举一反三

根据本实例，读者可以实现以下功能。

绘制彩色文本。

修改绘制的字体。

实例255 设置文本的字体

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\07\Ex07_255

实例说明

本实例演示在Java中绘制文本时，如何设置文本的字体，其中包括字体名称、大小和样式。运行程序，效果如图7.10所示。

图7.10 设置文本字体的效果

技术要点

本实例主要是通过在JPanel类的子类中，重写JComponent类的paint()方法，并在该方法中使用Graphics类的setFont()方法和使用Font类创建字体对象来实现的。

（1）使用Graphics类的setFont()方法，并将Font类创建的字体对象作为setFont()方法的参数，实现为文本设置字体的操作，setFont()方法的定义如下：

```
public abstract void setFont(Font font)
```

参数说明

font：为文本设置的字体对象。

（2）使用Font类创建字体对象，其构造方法的定义如下：

```
public Font(String name, int style, int size)
```

参数说明

- name：字体的名称。
- style：字体的样式。
- size：字体的大小。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的TextFontFrame窗体类。

（3）在TextFontFrame窗体类中，创建内部面板类ChangeTextFontPanel，并重写JComponent类的paint()方法，在该方法中使用Font类创建字体对象，并使用Graphics类的setFont()方法设置文本的字体。

（4）将内部面板类ChangeTextFontPanel的实例添加到窗体类TextFontFrame的内容面板上，用于在窗体上显示指定字体后的文本，代码如下：

```
class ChangeTextFontPanel extends JPanel {          //创建内部面板类
    public void paint(Graphics g) {          //重写paint()方法
        String value ="明日编程词典社区";
        int x = 40;          //文本位置的横坐标
        int y = 50;          //文本位置的纵坐标
        Font font = new Font("华文行楷", Font.BOLD + Font.ITALIC, 26);    //创建字体对象
        g.setFont(font);          //设置字体
        g.drawString(value, x, y);          //绘制文本
        value ="http://community.mrbccd.com";
        x=10;
//文本位置的横坐标
        y=100;
      //文本位置的纵坐标
        font = new Font("宋体", Font.BOLD, 20);          //创建字体对象
        g.setFont(font);
          //设置字体
```

```
        g.drawString(value,x,y);
          //绘制文本
      }
    }
```

举一反三

根据本实例，读者可以实现以下功能。

单一字体样式及字体样式的组合。

设置字体的颜色。

实例256 设置文本和图形的颜色

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\07\Ex07_256

实例说明

本实例演示在Java中绘制文本和图形时，如何设置文本和图形的颜色。运行程序，效果如图7.11所示。

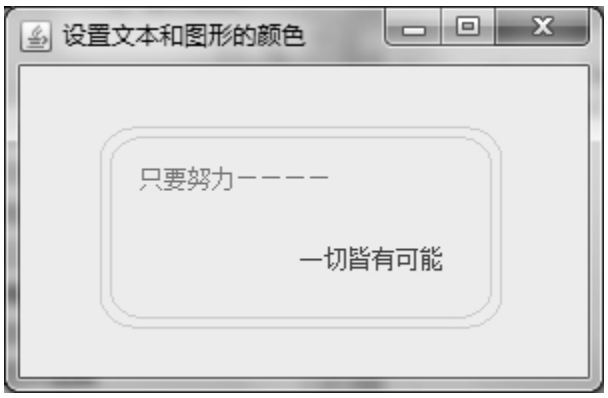


图7.11 设置文本和图形颜色的效果

技术要点

本实例主要是通过在JPanel类的子类中，重写JComponent类的paint()方法，并在该方法中使用Graphics类的setColor()方法和使用Color类创建颜色对象来实现的。

（1）使用Graphics类的setColor()方法，并将Color类创建的颜色对象作为setColor()方法的参数，实现为文本和图形设置颜色的操作，setColor()方法的定义如下：

public abstract void setColor(Color color)

参数说明

color：为文本或图形设置的颜色对象。

（2）使用Color类创建颜色对象，其构造方法的定义如下：

public Color(int r, int g, int b)

参数说明

- r：RGB 颜色的R 值。
- g：RGB 颜色的G 值。
- b：RGB 颜色的B 值。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的TextAndShapeColorFrame窗体类。

（3）在TextAndShapeColorFrame窗体类中，创建内部面板类TextAndShapeColorPanel，并重写JComponent类的paint()方法，在该方法中使用Color类创建颜色对象，并使用Graphics类的setColor()方法设置文本和图形的颜色。

（4）将内部面板类TextAndShapeColorPanel的实例添加到窗体类TextAndShapeColorFrame的内容面板上，用于在窗体上显示设置颜色后的文本和图形，代码如下：

```
class TextAndShapeColorPanel extends JPanel {          //创建内部面板类
    public void paint(Graphics g) {          //重写paint()方法
```

```
        String value ="只要努力————";
        int x = 60;          //文本位置的横坐标
        int y = 60;          //文本位置的纵坐标
        Color color = new Color(255,0,0);          //创建颜色对
象
        g.setColor(color);          //设置颜色
        g.drawString(value, x, y);          //绘制文本
        value ="一切皆有可能";
        x=140;                                    //文本位置的横
坐标
        y=100;                                    //文本位置的纵
坐标
        color = new Color(0,0,255);          //创建颜色对象
        g.setColor(color);          //设置颜色
        g.drawString(value, x, y);          //绘制文本
        color = Color.ORANGE;          //通过Color类的字段获得
颜色对象
        g.setColor(color);          //设置颜色
        g.drawRoundRect(40,30,200,100,40,30);          //绘制圆
角矩形
        g.drawRoundRect(45,35,190,90,36,26);          //绘制圆
角矩形
    }
}
```

举一反三

根据本实例，读者可以实现以下功能。

使用Color类的字段获得颜色。

7.2 图形处理

实例257 图形的加运算效果

这是一个可以用来提高基础性能的实例

实例位置：光盘\mingrisoft\07\Ex07_257

实例说明

本实例演示在Java中如何实现图形的加运算，即取两个图形的并集。运行程序，将在窗体上显示进行加运算后的图形，效果如图7.12所示。

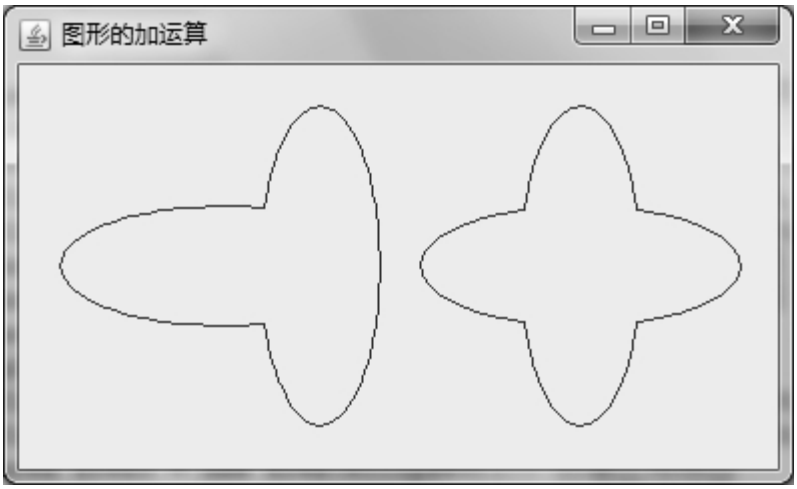


图7.12 图形进行加运算的效果

技术要点

本实例主要是通过在JPanel类的子类中，重写JComponent类的paint()方法，并在该方法中使用Graphics2D类的draw()方法和Area类来实现的，其中Area类用于封装图形对象，并通过add()方法对封装的图形对象进行加运算。

（1）使用Area类的构造方法封装图形对象，其构造方法的定义如下：

public Area(Shape s)

参数说明

s：Area类封装的图形对象。

（2）使用Area类的add()方法对封装的图形对象进行加运算，该方法的定义如下：

public void add(Area rhs)

参数说明

rhs：与当前Area对象进行加运算的Area对象。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的PlusOperationFrame窗体类。

（3）在 PlusOperationFrame 窗体类中，创建内部面板类 PlusOperationPanel ，并重写JComponent类的paint()方法，在该方法中实现图形的加运算。

（4）将内部面板类PlusOperationPanel的实例添加到窗体类PlusOperationFrame的内容面板上，用于在窗体上显示图形进行加运算后的效果，代码如下：

```
class PlusOperationPanel extends JPanel
{                                   //创建内部面板类
  public void paint(Graphics g)
 {                                          //重写paint()方法
    Graphics2D g2=
(Graphics2D)g;                                    //获得
Graphics2D对象
```

```
    Ellipse2D.Float ellipse1=new
Ellipse2D.Float(20,70,160,60);       //创建椭圆对象
    Ellipse2D.Float ellipse2=new
Ellipse2D.Float(120,20,60,160);       //创建椭圆对象
    Area area1=new
Area(ellipse1);                                  //创建区域椭
圆
    Area area2=new
Area(ellipse2);                                  //创建区域椭
圆
    area1.add(area2);
    //两个区域椭圆进行加运算
    g2.draw(area1);
    //绘制加运算后的区域椭圆
    Ellipse2D.Float ellipse3=new
Ellipse2D.Float(200,70,160,60);       //创建椭圆对象
    Ellipse2D.Float ellipse4=new
Ellipse2D.Float(250,20,60,160);       //创建椭圆对象
    Area area3=new
Area(ellipse3);                                  //创建区域椭
圆
    Area area4=new
Area(ellipse4);                                  //创建区域椭
圆
    area3.add(area4);
    //两个区域椭圆进行加运算
```

```
            g2.draw(area3);
            //绘制加运算后的区域椭圆
        }
    }
```

举一反三

根据本实例，读者可以实现以下功能。

长方形与三角形的加运算。

两个三角形的加运算。

实例258 图形的减运算效果

这是一个可以提高基础性能的实例

实例位置：光盘\mingrisoft\07\Ex07_258

实例说明

本实例演示在Java中如何实现图形的减运算，即从当前图形中减去与另一个图形的交集。运行程序，将在窗体上显示进行减运算后的图形，效果如图7.13所示。

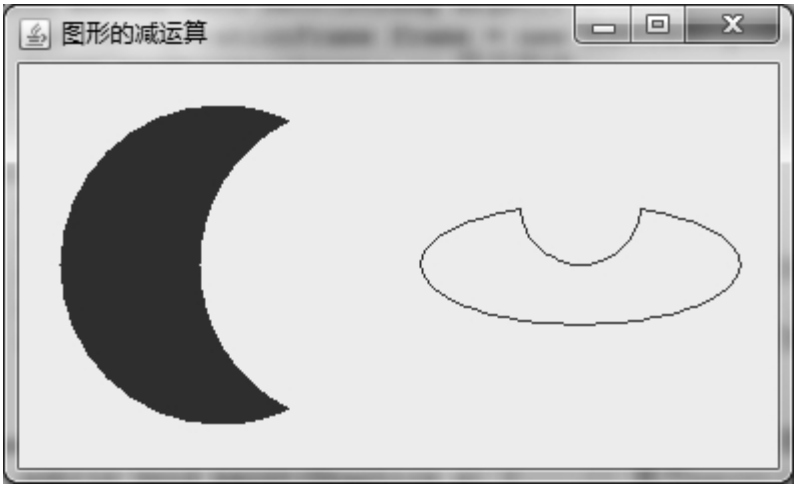


图7.13 图形进行减运算的效果

技术要点

本实例主要是通过在JPanel类的子类中，重写JComponent类的paint()方法，并在该方法中使用Graphics2D类的draw()方法和Area类来实现的，其中Area类用于封装图形对象，并通过subtract()方法对封装的图形对象进行减运算。

使用Area类的subtract()方法对封装的图形对象进行减运算，该方法的定义如下：

public void subtract(Area rhs)

参数说明

rhs：与当前Area对象进行减运算的Area对象。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的SubtractOperationFrame窗体类。

（3）在SubtractOperationFrame窗体类中，创建内部面板类SubtractOperationPanel，并重写JComponent类的paint()方法，在该方法中实现图形的减运算。

（4）将内部面板类SubtractOperationPanel的实例添加到窗体类SubtractOperationFrame的内容面板上，用于在窗体上显示图形进行减运算后的效果，代码如下：

```
class SubtractOperationPanel extends JPanel {          //创建内部面板类
    public void paint(Graphics g) {          //重写paint()方法
        Graphics2D g2 = (Graphics2D)g;          //获得Graphics2D对象
        Ellipse2D.Float ellipse1 = new Ellipse2D.Float(20, 20, 160, 160);     //创建圆对象
```

```
        Ellipse2D.Float ellipse2 = new Ellipse2D.Float(90,
20, 160, 160);     //创建圆对象
        Area area1 = new Area(ellipse1);          //创建区域圆
        Area area2 = new Area(ellipse2);          //创建区域圆
        area1.subtract(area2);          //两个区域圆进行减运算
        g2.fill(area1);          //绘制减运算后的区域圆
        Ellipse2D.Float ellipse3 = new Ellipse2D.Float(200,
70, 160, 60);     //创建椭圆对象
        Ellipse2D.Float ellipse4 = new Ellipse2D.Float(250,
40, 60, 60);     //创建圆对象
        Area area3 = new Area(ellipse3);          //创建区域椭
圆
        Area area4 = new Area(ellipse4);          //创建区域圆
        area3.subtract(area4);          //两个区域图形进行减运
算
        g2.draw(area3);          //绘制减运算后的区域图形
    }
}
```

举一反三

根据本实例，读者可以实现以下功能。

星形与T形的减运算。

长方形与正方形的减运算。

实例259 图形的交运算效果

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\07\Ex07_259

实例说明

本实例演示在Java中如何实现图形的交运算，即保留两个图形的交集。运行程序，将在窗体上显示进行交运算后的图形，效果如图7.14所示。

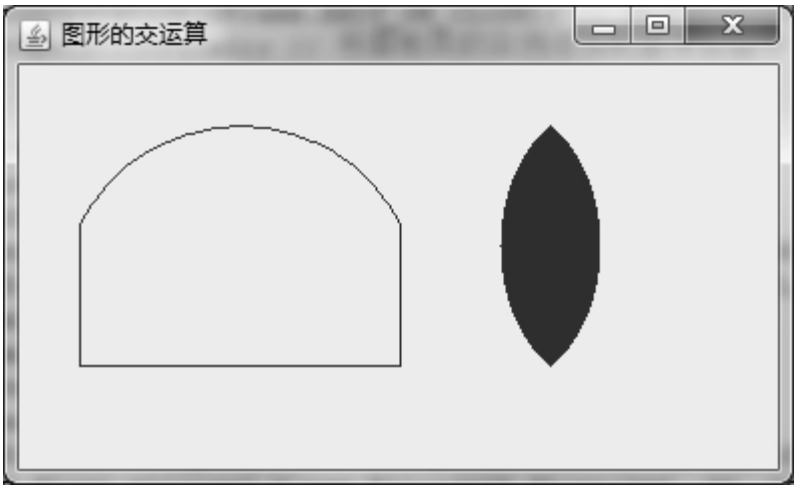


图7.14 图形进行交运算的效果

技术要点

本实例主要是通过在JPanel类的子类中，重写JComponent类的paint()方法，并在该方法中使用Graphics2D类的draw()方法和Area类来实现的，其中Area类用于封装图形对象，并通过intersect()方法对封装的图形对象进行交运算。

使用Area类的intersect()方法对封装的图形对象进行交运算，该方法的定义如下：

```
public void intersect(Area rhs)
```

参数说明

rhs：与当前Area对象进行交运算的Area对象。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的IntersectOperationFrame窗体类。

（3）在IntersectOperationFrame窗体类中，创建内部面板类IntersectOperationPanel，并重写JComponent类的paint()方法，在该方法中实现图形的交运算。

（4）将内部面板类IntersectOperationPanel的实例添加到窗体类IntersectOperationFrame的内容面板上，用于在窗体上显示图形进行交运算后的效果，代码如下：

```
class IntersectOperationPanel extends JPanel {        //创建内部面板类
    public void paint(Graphics g) {        //重写paint()方法
        Graphics2D g2 = (Graphics2D)g;        //获得Graphics2D对象
        Rectangle2D.Float rect = new Rectangle2D.Float(30, 30, 160, 120);    //创建矩形对象
        Ellipse2D.Float ellipse = new Ellipse2D.Float(20, 30, 180, 180);    //创建圆对象
        Area area1 = new Area(rect);        //创建区域矩形
        Area area2 = new Area(ellipse);        //创建区域圆
        area1.intersect(area2);        //两个区域图形进行交运算
        g2.draw(area1);        //绘制交运算后的区域图形
        Ellipse2D.Float ellipse1=new Ellipse2D.Float(190,20,100,140);    //创建椭圆对象
        Ellipse2D.Float ellipse2=new Ellipse2D.Float(240,20,100,140);    //创建椭圆对象
        Area area3 = new Area(ellipse1);        //创建区域椭圆
```

```
        Area area4 = new Area(ellipse2);          //创建区域椭
圆
        area3.intersect(area4);          //两个区域椭圆进行交运
算
        g2.fill(area3);          //绘制交运算后的区域椭圆
    }
}
```

举一反三

根据本实例，读者可以实现以下功能。

星形与T形的交运算。

实现两个图形的交运算。

实例260 图形的异或运算效果

这是一个可以启发思维的实例

实例位置：光盘\mingrisoft\07\Ex07_260

实例说明

本实例演示在Java中如何实现图形的异或运算，即两个图形去除交集后剩下的部分。运行程序，将在窗体上显示进行异或运算后的图形，效果如图7.15所示。

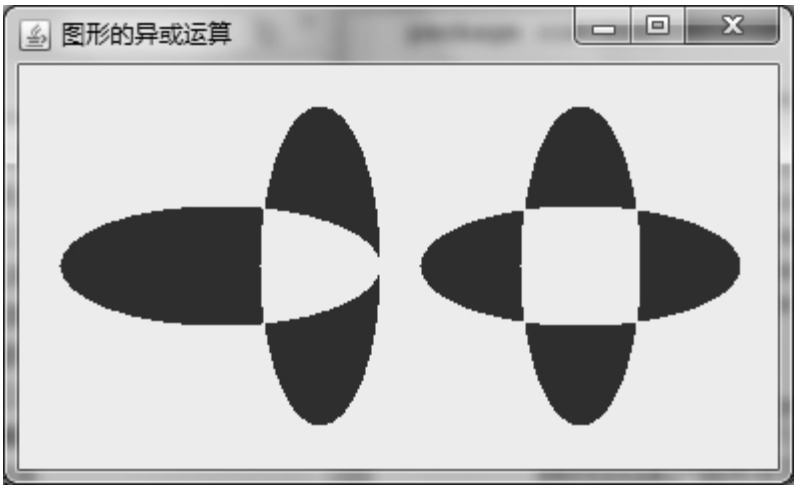


图7.15 图形进行异或运算的效果

技术要点

本实例主要是通过在JPanel类的子类中，重写JComponent类的paint()方法，并在该方法中使用Graphics2D类的draw()方法和Area类来实现的，其中Area类用于封装图形对象，并通过exclusiveOr()方法对封装的图形对象进行异或运算。

使用Area类的exclusiveOr()方法对封装的图形对象进行异或运算，该方法的定义如下：

public void exclusiveOr(Area rhs)

参数说明

rhs：与当前Area对象进行异或运算的Area对象。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的ExclusiveOrOperationFrame窗体类。

（3）在ExclusiveOrOperationFrame窗体类中，创建内部面板类ExclusiveOrOperationPanel，并重写JComponent类的paint()方法，在该方法中实现图形的异或运算。

（4）将内部面板类ExclusiveOrOperationPanel的实例，添加到窗体类ExclusiveOrOperation Frame的内容面板上，用于在窗体上显示图形进行异或运算后的效果，代码如下：

```
class ExclusiveOrOperationPanel extends JPanel
{        //创建内部面板类
  public void paint(Graphics g) {        //重写paint()方法
      Graphics2D g2 = (Graphics2D)g;        //获得Graphics2D对象
      Ellipse2D.Float ellipse1 = new Ellipse2D.Float(20, 70, 160, 60);    //创建椭圆对象
      Ellipse2D.Float ellipse2 = new Ellipse2D.Float(120, 20, 60, 160);    //创建椭圆对象
      Area area1 = new Area(ellipse1);        //创建区域椭圆
      Area area2 = new Area(ellipse2);        //创建区域椭圆
      area1.exclusiveOr(area2);        //两个区域椭圆进行异或运算
      g2.fill(area1);        //绘制异或运算后的区域椭圆
      Ellipse2D.Float ellipse3 = new Ellipse2D.Float(200, 70, 160, 60);    //创建椭圆对象
      Ellipse2D.Float ellipse4 = new Ellipse2D.Float(250, 20, 60, 160);    //创建椭圆对象
      Area area3 = new Area(ellipse3);        //创建区域椭圆
```

```
        Area area4 = new Area(ellipse4);          //创建区域椭圆
        area3.exclusiveOr(area4);          //两个区域椭圆进行异或运算
        g2.fill(area3);          //绘制异或运算后的区域椭圆
    }
}
```

举一反三

根据本实例，读者可以实现以下功能。

使用加、减运算绘制图形。

使用交、异或运算绘制图形。

实例261 缩放图形

本实例可以提高工作效率

实例位置：光盘\mingrisoft\07\Ex07_261

实例说明

本实例演示在Java中绘制图形时，如何对图形进行缩放，包括对图形进行放大、缩小和还原等操作。运行程序，效果如图7.16所示，用户可以通过单击窗体中的“放大”、“缩小”和“还原”按钮，对窗体上的图形进行相应操作。

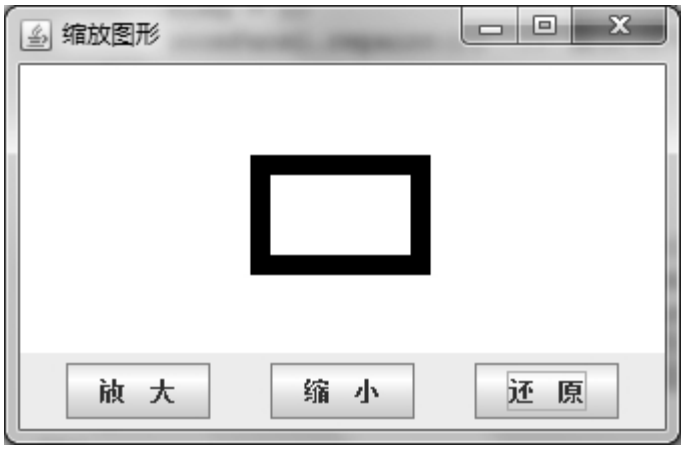


图7.16 缩放图形

技术要点

本实例主要是通过在JPanel类的子类中，重写JComponent类的paint()方法，并在该方法中使用Graphics2D类的scale()方法来实现的。

使用Graphics2D类的scale()方法，可以实现图形的缩放，该方法的定义如下：

```
public abstract void scale(double sx, double sy)
```

参数说明

● sx：与原图形x 坐标值相乘的量，如果sx 大于1.0，则在x 坐标轴上放大原图形；如果sx小于1.0，则在x坐标轴上缩小原图形。

● sy：与原图形y 坐标值相乘的量，如果sy 大于1.0，则在y 坐标轴上放大原图形；如果sy小于1.0，则在y坐标轴上缩小原图形。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的ZoomShapeFrame窗体类。

（3）在ZoomShapeFrame窗体类中，创建内部面板类ZoomShapePanel，并重写JComponent类的paint()方法，在该方法中使用Graphics2D类的scale()方法缩放图形。

（4）将内部面板类ZoomShapePanel的实例添加到窗体类ZoomShapeFrame的内容面板上，用于在窗体上显示缩放后的图形，代码如下：

```
class ZoomShapePanel extends JPanel {          //创建内部面板类
    public void paint(Graphics g) {          //重写paint()方法
        Graphics2D g2 = (Graphics2D) g;          //获得Graphics2D对象
        Rectangle2D.Float rect = new Rectangle2D.Float(120, 50, 80, 50);    //创建矩形对象
        BasicStroke stroke = new BasicStroke(10);          //创建宽度是10的笔画对象
        g2.setStroke(stroke);          //设置笔画对象
        g2.clearRect(0, 0, 338, 220);          //清除原有内容
        if (flag == 0) {
          g2.draw(rect);          //绘制原矩形
        } else if (flag == 1) {
          g2.scale(1.3, 1.3);          //放大1.3倍
          g2.draw(rect);          //绘制矩形
        } else if (flag == 2) {
          g2.scale(0.5, 0.5);          //缩小0.5倍
          g2.draw(rect);          //绘制矩形
        }
    }
}
```

举一反三

根据本实例，读者可以实现以下功能。

对长方形的放大操作。

对长方形的缩小操作。

实例262 旋转图形

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\07\Ex07_262

实例说明

本实例演示在Java中绘制图形时，如何对图形进行旋转。运行程序，单击窗体上的“顺时针”按钮，可以将图形顺时针旋转，效果如图7.17所示，用户还可以通过单击“逆时针”和“还原”按钮，对窗体上的图形进行逆时针旋转和还原等操作。

图7.17 顺时针旋转图形的效果

技术要点

本实例主要是通过在JPanel类的子类中，重写JComponent类的paint()方法，并在该方法中使用Graphics2D类的rotate()方法来实现的。

使用Graphics2D类的rotate()方法，可以实现图形的旋转，该方法的定义如下：

public abstract void rotate(double theta, double x, double y)

参数说明

- theta：旋转的角度，以弧度为单位。
- x：旋转原点的x 坐标。
- y：旋转原点的y 坐标。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的RotateShapeFrame窗体类。

（3）在RotateShapeFrame窗体类中，创建内部面板类RotateShapePanel，并重写JComponent类的paint()方法，在该方法中使用Graphics2D类的rotate()方法旋转图形。

（4）将内部面板类RotateShapePanel的实例添加到窗体类RotateShapeFrame的内容面板上，用于在窗体上显示旋转后的图形，代码如下：

```
class RotateShapePanel extends JPanel {          //创建内部面板类
    public void paint(Graphics g) {          //重写paint()方法
        Graphics2D g2 = (Graphics2D) g;          //获得Graphics2D对象
        Rectangle2D.Float rect = new Rectangle2D.Float(40, 40, 80, 50);//创建矩形对象
        BasicStroke stroke = new BasicStroke(10);          //创建宽度是10的笔画对象
        g2.setStroke(stroke);          //设置笔画对象
```

```
        g2.clearRect(0, 0, 338, 220);          //清除原有内容
        if (flag == 0) {
          g2.draw(rect);          //绘制原矩形
        } else if (flag == 1) {
          g2.rotate(rotateValue);          //顺时针旋转
          g2.draw(rect);          //绘制矩形
        } else if (flag == 2) {
          g2.rotate(rotateValue);          //逆时针旋转
          g2.draw(rect);          //绘制矩形
        }
      }
    }
```

举一反三

根据本实例，读者可以实现以下功能。

还原图形。

实例263 斜切图形

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\07\Ex07_263

实例说明

本实例演示在Java中绘制图形时，如何对图形进行斜切。运行程序，单击窗体上的“上斜切”按钮，可以实现对矩形进行向上斜切的操作，效果如图7.18所示，用户还可以通过单击窗体上的“下斜切”和“还原”按钮，对窗体上的图形进行向下斜切和还原等操作。

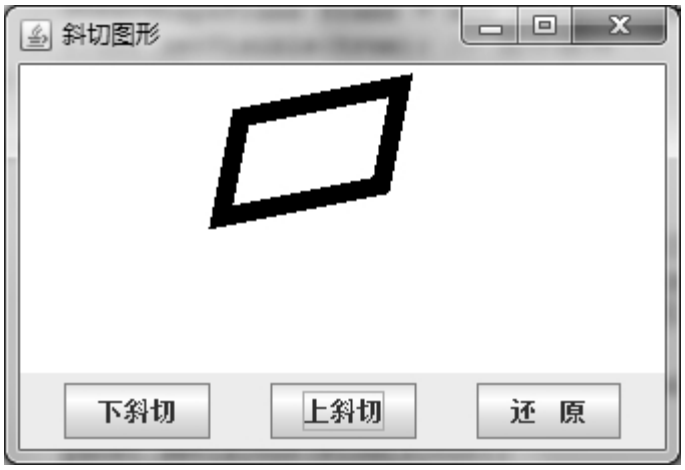


图7.18 向上斜切图形的效果

技术要点

本实例主要是通过在JPanel类的子类中，重写JComponent类的paint()方法，并在该方法中使用Graphics2D类的shear()方法来实现的。

使用Graphics2D类的shear()方法，可以实现图形的斜切，该方法的定义如下：

```
public abstract void shear(double shx, double shy)
```

参数说明

● shx：在正x 轴方向移动坐标的乘数，它可以作为相应y 坐标的函数。

● shy：在正y 轴方向移动坐标的乘数，它可以作为相应x 坐标的函数。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的ShearShapeFrame窗体类。

（3）在ShearShapeFrame窗体类中，创建内部面板类ShearShapePanel，并重写JComponent类的paint()方法，在该方法中

使用Graphics2D类的shear()方法斜切图形。

（4）将内部面板类ShearShapePanel的实例添加到窗体类ShearShapeFrame的内容面板上，用于在窗体上显示斜切后的图形，代码如下：

```
class ShearShapePanel extends JPanel {          //创建内部面板类
  public void paint(Graphics g) {          //重写paint()方法
    Graphics2D g2 = (Graphics2D) g;          //获得Graphics2D对象
    Rectangle2D.Float rect = new Rectangle2D.Float(120, 50, 80, 50);    //创建矩形对象
    BasicStroke stroke = new BasicStroke(10);          //创建宽度是10的笔画对象
    g2.setStroke(stroke);          //设置笔画对象
    g2.clearRect(0, 0, 338, 230);          //清除原有内容
    if (flag == 0) {
      g2.draw(rect);          //绘制原矩形
    } else if (flag == 1) {
      g2.shear(0.2,0.2);          //向下斜切
      g2.draw(rect);          //绘制矩形
    } else if (flag == 2) {
      g2.shear(-0.2,-0.2);          //向上斜切
      g2.draw(rect);          //绘制矩形
    }
  }
}
```

举一反三

根据本实例，读者可以实现以下功能。

使用Graphics2D 类的shear(double shx, double shy)方法对图形进行斜切。

向下斜切图形。

实例264 为图形填充渐变色

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\07\Ex07_264

实例说明

本实例演示在Java中绘制图形时，如何为图形填充渐变色。运行程序，效果如图7.19所示。

图7.19 为图形填充渐变色的效果

技术要点

本实例主要是通过在JPanel类的子类中，重写JComponent类的paint()方法，并在该方法中使用Graphics2D类的setPaint()方法和使用GradientPaint类创建封装渐变颜色的对象来实现的。

（1）使用Graphics2D类的setPaint()方法，并将GradientPaint类创建的封装渐变颜色的对象作为setPaint()方法的参数，实现为图

形填充渐变色的操作，setPaint()方法的定义如下：

public abstract void setPaint(Paint paint)

参数说明

paint：封装了渐变颜色的Paint对象。

（2）使用GradientPaint类创建封装渐变颜色的对象，其构造方法的定义如下：

public GradientPaint(float x1, float y1, Color color1, float x2, float y2, Color color2, boolean cyclic)

参数说明

- x1：用户空间中第1 个指定点的x 坐标。
- y1：用户空间中第1 个指定点的y 坐标。
- color1：第1 个指定点处的Color 对象。
- x2：用户空间中第2 个指定点的x 坐标。
- y2：用户空间中第2 个指定点的y 坐标。
- color2：第2 个指定点处的Color 对象。
- cyclic：如果渐变模式在两种颜色之间重复循环，则该值设置为true；否则设置为false。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的FillGradientFrame窗体类。

（3）在FillGradientFrame窗体类中，创建内部面板类FillGradientPanel，并重写JComponent类的paint()方法，在该方法中使用Graphics2D类的setPaint()方法设置封装了渐变色的对象，该对象是通过GradientPaint类创建的。

（4）将内部面板类FillGradientPanel的实例添加到窗体类FillGradientFrame的内容面板上，用于在窗体上显示填充了渐变颜色

的图形，代码如下：

```
    class FillGradientPanel extends JPanel {          //创建内部面板类
      public void paint(Graphics g) {          //重写paint()方法
        Graphics2D g2 = (Graphics2D) g;          //获得Graphics2D对象
        Rectangle2D.Float rect = new Rectangle2D.Float(20, 20, 280, 140);          //创建矩形对象
        GradientPaint paint = new GradientPaint(20,20,Color.BLUE,100,80,Color.RED,true); //创建循环渐变的GradientPaint对象
        g2.setPaint(paint);          //设置渐变
        g2.fill(rect);          //绘制矩形
      }
    }
```

举一反三

根据本实例，读者可以实现以下功能。

使用LinearGradientPaint类实现多颜色的线性渐变。

使用RadialGradientPaint类实现多颜色的径向渐变。

实例265 平移坐标轴

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\07\Ex07_265

实例说明

本实例演示在Java中绘制图形时，如何实现坐标轴的平移。运行程序，将在窗体的左上角绘制矩形，单击窗体上的“平移坐标轴”按钮，将实现坐标轴的平移，效果如图7.20所示。

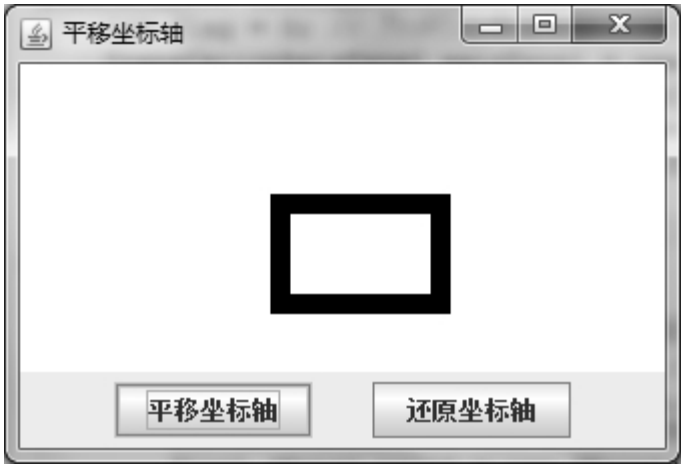


图7.20 平移坐标轴

技术要点

本实例主要是通过在JPanel类的子类中，重写JComponent类的paint()方法，并在该方法中使用Graphics2D类的translate()方法来实现的。

使用Graphics2D类的translate()方法，可以实现坐标轴的平移，该方法的定义如下：

```
public abstract void translate(int x,int y)
```

参数说明

- x：平移到指定的x 坐标处。
- y：平移到指定的y 坐标处。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的TranslationAxisFrame窗体类。

（3）在 TranslationAxisFrame 窗体类中，创建内部面板类TranslationAxisPanel，并重写JComponent类的paint()方法，在该方法中使用Graphics2D类的translate()方法，实现坐标轴的平移。

（4）将内部面板类 TranslationAxisPanel的实例添加到窗体类TranslationAxisFrame的内容面板上，用于在窗体上显示平移坐标轴的效果，代码如下：

```
class TranslationAxisPanel extends JPanel {          //创建内部面板类
    public void paint(Graphics g) {          //重写paint()方法
        Graphics2D g2 = (Graphics2D) g;          //获得Graphics2D对象
        Rectangle2D.Float rect = new Rectangle2D.Float(10, 10, 80, 50);//创建矩形对象
        BasicStroke stroke = new BasicStroke(10);          //创建宽度是10的笔画对象
        g2.setStroke(stroke);          //设置笔画对象
        g2.clearRect(0, 0, 338, 230);          //清除原有内容
        if (flag == 0) {
          g2.translate(0, 0);          //平移坐标轴
          g2.draw(rect);          //绘制矩形
        } else if (flag == 1) {
          g2.translate(120, 60);          //平移坐标轴
          g2.draw(rect);          //绘制矩形
        }
    }
}
```

举一反三

根据本实例，读者可以实现以下功能。

绕指定点旋转图形。

沿窗体对角线平移。

7.3 绘制图案

实例266 绘制五环图案

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\07\Ex07_266

实例说明

本实例演示奥林匹克运动会的会徽，即五环图案的绘制。运行程序，将在窗体上绘制五环图案，效果如图7.21所示。

图7.21 五环图案

技术要点

本实例主要是通过在JPanel类的子类中，重写JComponent类的paint()方法，并在该方法中使用Graphics2D类的setStroke()、setColor()和drawOval()方法来实现的。

（1）使用Graphics2D类的setStroke()方法，指定笔画的粗细。

（2）使用Graphics2D类的setColor()方法，指定颜色。

（3）使用Graphics2D类的drawOval()方法，在指定位置绘制圆环，该方法是从Graphics类中继承的。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的FiveDaisyChainFrame窗体类。

（3）在 FiveDaisyChainFrame 窗体类中，创建内部面板类FiveDaisyChainPanel，并重写JComponent类的paint()方法，在该方法中实现五环图案的绘制。

（4）将内部面板类FiveDaisyChainPanel的实例添加到窗体类FiveDaisyChainFrame的内容面板上，用于在窗体上显示五环图案，代码如下：

```
class FiveDaisyChainPanel extends JPanel {          //创建内部面板类
    public void paint(Graphics g) {          //重写paint()方法
        Graphics2D g2 = (Graphics2D)g;          //获得Graphics2D对象
        BasicStroke stroke = new BasicStroke(3);          //创建宽度是3的笔画对象
        g2.setStroke(stroke);          //设置笔画对象
        Color color = new Color(0,162,232);          //创建颜色对象
        g2.setColor(color);          //设置颜色
        g2.drawOval(30, 40, 60, 60);          //绘制第一个圆
        //省略了绘制其他圆的代码
    }
```

}

举一反三

根据本实例，读者可以实现以下功能。

获取五环图案的颜色。

绘制三角星图形。

实例267 绘制艺术图案

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\07\Ex07_267

实例说明

本实例演示如何使用坐标轴平移、图形旋转和获得随机数等技术绘制艺术图案。运行程序，将在窗体上绘制艺术图案，效果如图7.22所示。

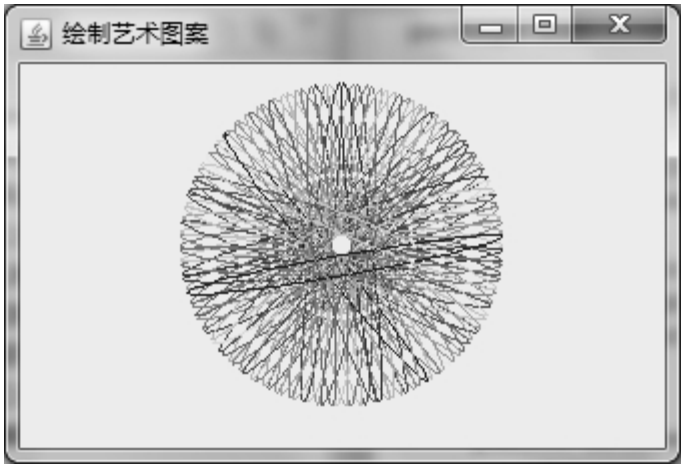


图7.22 艺术图案

技术要点

本实例主要是通过在JPanel类的子类中，重写JComponent类的paint()方法，并在该方法中使用Graphics2D类的translate()、setColor()、rotate()和draw()方法来实现的。

（1）使用Graphics2D类的translate()方法，将坐标轴平移到指定点。

（2）使用Graphics2D类的setColor()方法，设置颜色。

（3）使用Graphics2D类的rotate()方法，旋转绘图上下文。

（4）使用Graphics2D类的draw()方法，在指定位置绘制椭圆。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的ArtDesignFrame窗体类。

（3）在 ArtDesignFrame 窗体类中，创建内部面板类ArtDesignPanel，并重写 JComponent类的paint()方法，在该方法中实现艺术图案的绘制。

（4）将内部面板类ArtDesignPanel的实例添加到窗体类ArtDesignFrame的内容面板上，用于在窗体上显示艺术图案，代码如下：

```
class ArtDesignPanel extends JPanel {          //创建内部面板类
  public void paint(Graphics g) {          //重写paint()方法
    Graphics2D g2 = (Graphics2D)g;          //获得Graphics2D对象
    Ellipse2D.Float ellipse = new Ellipse2D.Float(-80, 5, 160, 10); //创建椭圆对象
    Random random = new Random();          //创建随机数对象
    g2.translate(160, 90);          //平移坐标轴
    int R = random.nextInt(256);          //随机产生颜色的R值
```

```
    int G = random.nextInt(256);          //随机产生颜色的G
值
    int B = random.nextInt(256);          //随机产生颜色的B
值
    Color color = new Color(R,G,B);          //创建颜色对象
    g2.setColor(color);          //指定颜色
    g2.draw(ellipse);          //绘制椭圆
    int i=0;
    while (i<100){
      R = random.nextInt(256);          //随机产生颜色的R值
      G = random.nextInt(256);          //随机产生颜色的G值
      B = random.nextInt(256);          //随机产生颜色的B值
      color = new Color(R,G,B);          //创建新的颜色对象
      g2.setColor(color);          //指定颜色
      g2.rotate(10);          //旋转画布
      g2.draw(ellipse);          //绘制椭圆
      i++;
    }
  }
}
```

举一反三

根据本实例，读者可以开发以下程序。

绘制花瓣。

绘制调色板。

实例268 绘制花瓣

这是一个可以启发思维的实例

实例位置：光盘\mingrisoft\07\Ex07_268

实例说明

本实例演示如何使用坐标轴平移和图形旋转等技术绘制花瓣。运行程序，将在窗体上绘制花瓣，效果如图7.23所示。

图7.23 绘制花瓣

技术要点

本实例主要是通过在JPanel类的子类中，重写JComponent类的paint()方法，并在该方法中使用Graphics2D类的translate()、setColor()、rotate()和fill()方法来实现的。

（1）使用Graphics2D类的translate()方法，将坐标轴平移到指定点。

（2）使用Graphics2D类的setColor()方法，设置颜色。

（3）使用Graphics2D类的rotate()方法，旋转绘图上下文。

（4）使用Graphics2D类的fill()方法，在指定位置绘制带填充色的椭圆。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的DrawFlowerFrame窗体类。

（3）在DrawFlowerFrame窗体类中，创建内部面板类DrawFlowerPanel，并重写JComponent类的paint()方法，在该方法中实现花瓣的绘制。

（4）将内部面板类DrawFlowerPanel的实例添加到窗体类DrawFlowerFrame的内容面板上，用于在窗体上显示绘制的花瓣，代码如下：

```
class DrawFlowerPanel extends JPanel {          //创建内部面板类
    public void paint(Graphics g) {          //重写paint()方法
        Graphics2D g2 = (Graphics2D)g;          //获得Graphics2D对象
        g2.translate(drawFlowerPanel.getWidth() / 2, drawFlowerPanel.getHeight() / 2);//平移坐标轴
        //绘制绿色花瓣
        Ellipse2D.Float ellipse = new Ellipse2D.Float(30, 0, 70, 20);          //创建椭圆对象
        Color color = new Color(0,255,0);          //创建颜色对象
        g2.setColor(color);          //指定颜色
        g2.fill(ellipse);          //绘制椭圆
        int i=0;
        while (i<8){
          g2.rotate(30);          //旋转画布
          g2.fill(ellipse);          //绘制椭圆
```

```
        i++;
    }
    //绘制红色花瓣
    ellipse = new Ellipse2D.Float(20, 0, 60,
15);          //创建椭圆对象
    color = new Color(255,0,0);          //创建颜色对象
    g2.setColor(color);          //指定颜色
    g2.fill(ellipse);          //绘制椭圆
    i=0;
    while (i<15){
        g2.rotate(75);          //旋转画布
        g2.fill(ellipse);          //绘制椭圆
        i++;
    }
    //绘制黄色花瓣
    ellipse = new Ellipse2D.Float(10, 0, 50,
15);          //创建椭圆对象
    color = new Color(255,255,0);          //创建颜色对象
    g2.setColor(color);          //指定颜色
    g2.fill(ellipse);          //绘制椭圆
    i=0;
    while (i<8){
        g2.rotate(30);          //旋转画布
        g2.fill(ellipse);          //绘制椭圆
        i++;
    }
    //绘制红色中心点
```

```
            color = new Color(255, 0, 0);          //创建颜色对象
            g2.setColor(color);          //指定颜色
            ellipse = new Ellipse2D.Float(-10, -10, 20,
        20);          //创建椭圆对象
            g2.fill(ellipse);          //绘制椭圆
        }
    }
```

举一反三

根据本实例，读者可以实现以下功能。

绘制五色花瓣。

绘制四叶草。

实例269 绘制公章

这是一个可以美化界面的实例

实例位置：光盘\mingrisoft\07\Ex07_269

实例说明

本实例演示如何使用坐标轴平移、缩放、绘制椭圆、绘制多边形和绘制文本等技术实现公章的绘制。运行程序，将在窗体上显示绘制的公章，效果如图7.24所示。

图7.24 绘制公章

技术要点

本实例主要是通过在JPanel类的子类中，重写JComponent类的paint()方法，并在该方法中使用 Graphics2D 类的 translate()、setColor()、scale()、drawString()、fillPolygon()和 draw()等方法来实现的。

（1）使用Graphics2D类的translate()方法，将坐标轴平移到指定点。

（2）使用Graphics2D类的setColor()方法，设置颜色。

（3）使用Graphics2D类的scale()方法，对公章中的文本进行缩放。

（4）使用Graphics2D类的drawString()方法，绘制文本，该方法是从Graphics类继承的。

（5）使用Graphics2D类的fillPolygon()方法，绘制公章的五星，该方法也是从Graphics类继承的。

（6）使用Graphics2D类的draw()方法，绘制表示公章的圆。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的DrawCachetFrame窗体类。

（3）在DrawCachetFrame窗体类中，创建内部面板类DrawCachetPanel，并重写JComponent类的paint()方法，在该方法中实现公章的绘制。

（4）将内部面板类DrawCachetPanel的实例添加到窗体类DrawCachetFrame的内容面板上，用于在窗体上显示绘制的公章，代码如下：

```
class DrawCachetPanel extends JPanel {          //创建内部面板类
    public void paint(Graphics g) {          //重写paint()方法
        Graphics2D g2 = (Graphics2D) g;          //获得Graphics2D对象
        g2.translate(170, 100);          //平移坐标轴
        BasicStroke stroke = new BasicStroke(6);          //创建宽度是6的笔画对象
        g2.setStroke(stroke);          //设置笔画对象
        //绘制圆
        Ellipse2D.Float ellipse = new Ellipse2D.Float(-80, -80, 160, 160);     //创建圆对象
        Color color = new Color(255, 0, 0);          //创建颜色对象
        g2.setColor(color);          //指定颜色
        g2.draw(ellipse);          //绘制圆
        //绘制五星
```

```
    int[] x1 ={ 0, 8, 30, 16, 25, 0, -25, -16, -30, -8
};        //多边形的横坐标
    int[] y1 ={-35, -10, -15, 5, 28, 10, 28, 5, -15, -10
};        //多边形的纵坐标
    int n1 = 10;          //多边形的边数
    g2.fillPolygon(x1, y1, n1);          //绘制多边形
    //绘制文本
    g2.scale(1.8, 1.8);          //放大
    Font font = new Font("宋体", Font.BOLD,
12);          //创建字体
    g2.setFont(font);          //设置字体
    g2.drawString("专用章", -25, 30);          //绘制文本
    int width = getWidth();          //获得面板宽度
    int height = getHeight();          //获得面板高度
    char[] array ="明日科技有限公
司    ".toCharArray();    //把字符串转换为字符数组
    int len = array.length * 2;          //定义半径
    font = new Font("宋体", Font.BOLD, 10);          //创建
新字体
    g2.setFont(font);          //设置字体
    double angle = 0;          //初始角度
    for (int i = 0; i < array.length; i++) {          //遍
历字符串中的字符
      int x = (int) (len * Math.sin(Math.toRadians(angle
  + 270)));    /计算每个文字的位置
      int y = (int) (len * Math.cos(Math.toRadians(angle
  + 270)));    //计算每个文字的位置
```

```
            g2.drawString(array[i] +"", width / 2 + x - 168,
        height / 2 - y - 95); //绘制每个字符，其中168和95是坐标
        平移值
            angle = angle + 360d / array.length;          //改变
        角度
          }
        }
    }
```

举一反三

根据本实例，读者可以实现以下功能。

绘制财务公章。

绘制公司公章。

7.4 图像处理

实例270 绘制图像

这是一个可以美化界面的实例

实例位置：光盘\mingrisoft\07\Ex07_270

实例说明

在进行应用程序开发时，为了使程序界面美观，可以为应用程序窗体添加背景图片，方法是通过Java的绘图技术在控件上绘制图像，并将带有图片的控件添加到窗体上。运行程序，可以看到在窗体中显示的图片，效果如图7.25所示。

图7.25 绘制图像的效果

技术要点

本实例主要是通过在JPanel类的子类中，重写JComponent类的paint()方法，并在该方法中使用Graphics类的drawImage()方法来实现的。

使用Graphics类的drawImage()方法可绘制图像，该方法的定义如下：

public abstract boolean drawImage(Image img, int x, int y, ImageObserver observer)

参数说明

- img：需要绘制的图像对象。
- x：绘制顶点的x 坐标。
- y：绘制顶点的y 坐标。
- observer：图像观察者。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的DrawImageFrame窗体类。

（3）在 DrawImageFrame 窗体类中创建内部面板类DrawImagePanel，并重写 JComponent类的paint()方法，在该方法中使用Graphics类的drawImage()方法绘制图像。

（4）将内部面板类DrawImagePanel的实例添加到窗体类DrawImageFrame的内容面板上，用于在窗体上显示绘制的图像。窗体类 DrawImageFrame 的构造方法中，创建图像对象的代码如下：

```
URL imgUrl = DrawImageFrame.class.getResource("/img/image.jpg"); //获取图片资源的路径
img = Toolkit.getDefaultToolkit().getImage(imgUrl);          //获取图像资源
```

面板类DrawImagePanel的代码如下：

```
class DrawImagePanel extends JPanel {
```

```
    public void paint(Graphics g) {          //重写paint()方
法
        g.drawImage(img, 0, 0, this);        //绘制图像对象
    }
}
```

举一反三

根据本实例，读者可以实现以下功能。

美化应用程序窗体界面。

绘制带阴影效果的图案。

实例271 缩放图像

这是一个可以美化界面的实例

实例位置：光盘\mingrisoft\07\Ex07_271

实例说明

本实例演示如何使用Java绘图技术对图像进行缩放。运行程序，通过调整窗体上的滑块，可以对窗体上显示的图片进行缩放，效果如图7.26所示。

图7.26 缩放图像

技术要点

本实例主要是通过在JPanel类的子类中，重写JComponent类的paint()方法，并在该方法中使用Graphics类的drawImage()方法来实现的。

使用Graphics类的drawImage()方法可缩放图像，该方法的定义如下：

```
public abstract boolean drawImage(Image img, int x, int y, int width, int height, ImageObserver observer)
```

参数说明

- img：需要绘制的图像对象。
- x：绘制顶点的x 坐标。
- y：绘制顶点的y 坐标。
- width：显示图像的矩形的宽度。
- height：显示图像的矩形的高度。
- observer：图像观察者。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的ZoomImageFrame窗体类。

（3）在ZoomImageFrame窗体类中，创建内部面板类ZoomImagePanel，并重写JComponent类的paint()方法，在该方法中使用Graphics类的drawImage()方法绘制缩放后的图像。

（4）将内部面板类ZoomImagePanel的实例添加到窗体类ZoomImageFrame的内容面板上，并在窗体下方添加一个滑块控件，通过拖动滑块可以在窗体上显示图像的缩放效果。窗体上滑块控件的事件代码如下：

```
slider.addChangeListener(new ChangeListener() {
```

```
    public void stateChanged(final ChangeEvent e) { //滑块位置改变时执行该方法
        imagePanel.repaint();                    //重新调用面板类的paint()方法
    }
});
```

面板类ZoomImagePanel用于绘制调整滑块缩放的图像，该类的代码如下：

```
class ZoomImagePanel extends JPanel {
    public void paint(Graphics g) {
        g.clearRect(0, 0, this.getWidth(), this.getHeight()); //清除绘图上下文的内容
        imgWidth = img.getWidth(this);          //获取图片宽度
        imgHeight = img.getHeight(this);        //获取图片高度
        float value = slider.getValue();        //滑块组件的取值
        newW = (int) (imgWidth * value / 100);          //计算图片缩放后的宽度
        newH = (int) (imgHeight * value / 100);         //计算图片缩放后的高度
        g.drawImage(img, 0, 0, newW, newH, this);       //绘制指定大小的图片
    }
}
```

举一反三

根据本实例，读者可以开发以下程序。

通过滑块控件调整颜色的RGB值。

实例272 翻转图像

这是一个可以提高分析能力的实例

实例位置：光盘\mingrisoft\07\Ex07_272

实例说明

本实例演示如何利用Java的绘图技术，实现翻转图像的操作。运行程序，单击窗体上的“水平翻转”和“垂直翻转”按钮，可以实现图像的水平翻转和垂直翻转操作。单击窗体上的“垂直翻转”按钮，将对图像进行垂直翻转，效果如图7.27所示。

技术要点

本实例也是通过在JPanel类的子类中，重写JComponent类的paint()方法，并在该方法中使用Graphics类的drawImage()方法来实现的。

使用Graphics类的drawImage()方法可翻转图像，该方法的定义如下：

```
public abstract boolean drawImage(Image img, int dx1, int dy1, int dx2, int dy2, int sx1, int sy1, int sx2, int sy2, ImageObserver observer)
```

图7.27 垂直翻转图像的效果

参数说明

- img：需要绘制的图像对象。
- dx1：目标矩形第1 个角的x 坐标。
- dy1：目标矩形第1 个角的y 坐标。
- dx2：目标矩形第2 个角的x 坐标。
- dy2：目标矩形第2 个角的y 坐标。
- sx1：源矩形第1 个角的x 坐标。
- sy1：源矩形第1 个角的y 坐标。
- sx2：源矩形第2 个角的x 坐标。
- sy2：源矩形第2 个角的y 坐标。
- observer：图像观察者。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的PartImageFrame窗体类。

（3）在 PartImageFrame 窗体类中，创建内部面板类 PartImagePanel，并重写 JComponent类的paint()方法，在该方法中使用Graphics类的drawImage()方法绘制图像。

（4）将内部面板类PartImagePanel的实例添加到窗体类PartImageFrame的内容面板上，在窗体上添加两个按钮，分别用于实现图像的水平翻转和垂直翻转，并在窗体上显示翻转后的图像。“水平翻转”按钮的事件代码如下：

```
btn_h.addActionListener(new ActionListener() {
  public void actionPerformed(final ActionEvent e) {
    //下面3行代码用于交换sx1和sx2的值
    int x = sx1;
    sx1 = sx2;
    sx2 = x;
    imagePanel.repaint();                          //重新调用面板类的paint()方法
  }
});
```

“垂直翻转”按钮的事件代码如下：

```
btn_v.addActionListener(new ActionListener() {
  public void actionPerformed(final ActionEvent e) {
    //下面3行代码用于交换sy1和sy2的值
    int y = sy1;
    sy1 = sy2;
    sy2 = y;
    imagePanel.repaint();                          //重新调用面板类的paint()方法
  }
});
```

面板类PartImagePanel用于绘制原图像和翻转后的图像，该类的代码如下：

```
class PartImagePanel extends JPanel {
  public void paint(Graphics g) {
    g.clearRect(0, 0, this.getWidth(),
this.getHeight());          //清除绘图上下文的内容
    g.drawImage(img, dx1, dy1, dx2, dy2, sx1, sy1, sx2,
sy2, this); //绘制图像
  }
}
```

举一反三

根据本实例，读者可以开发以下程序。

交换两个变量的值。

设置恢复图像初始状态按钮。

实例273 旋转图像

这是一个可以启发思维的实例

实例位置：光盘\mingrisoft\07\Ex07_273

实例说明

本实例演示如何利用Java的绘图技术，实现图像的旋转操作，即让图像绕坐标原点进行旋转。运行程序，效果如图7.28所示。

图7.28 旋转图像的效果

技术要点

本实例主要是通过在JPanel类的子类中，重写JComponent类的paint()方法，并在该方法中使用Graphics2D类的rotate()方法和从Graphics类继承的drawImage()方法来实现的。

使用Graphics2D类的rotate()方法可以实现图像的旋转，该方法的定义如下：

public abstract void rotate(double theta)

参数说明

theta：旋转的角度，以弧度为单位。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的RotateImageFrame窗体类。

（3）在RotateImageFrame窗体类中，创建内部面板类RotateImagePanel，并重写JComponent类的 paint()方法，在该方法

中使用从 Graphics 类继承的 drawImage()方法和 Graphics2D 类的 rotate()方法实现图像旋转。

（4）将内部面板类RotateImagePanel的实例添加到窗体类RotateImageFrame的内容面板上，用于在窗体上显示旋转后的图像，代码如下：

```
class RotatePanel extends JPanel {
  public void paint(Graphics g) {
    Graphics2D g2 = (Graphics2D) g;          //获得Graphics2D对象
    g2.drawImage(img, 80, 10, 260, 150, this);       //绘制指定大小的图片
    g2.rotate(Math.toRadians(10));          //将图片旋转10°
    g2.drawImage(img, 80, 10, 260, 150, this);       //绘制指定大小的图片
    g2.rotate(Math.toRadians(10));          //将图片旋转10°
    g2.drawImage(img, 80, 10, 260, 150, this);       //绘制指定大小的图片
  }
}
```

举一反三

根据本实例，读者可以实现以下程序。

逆时针旋转图像。

恢复初始状态。

实例274 倾斜图像

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\07\Ex07_274

实例说明

本实例演示如何在应用程序中利用Java的绘图技术，实现倾斜图像的操作。运行程序，将在窗体上显示倾斜的图像，效果如图7.29所示。

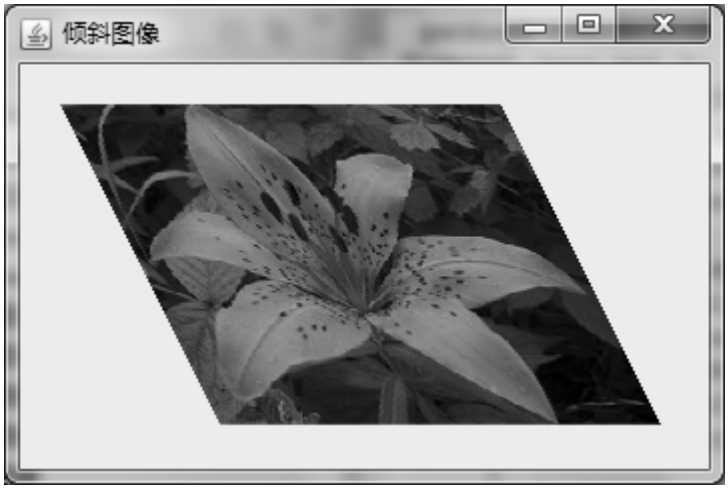


图7.29 倾斜图像的效果

技术要点

本实例主要是通过在JPanel类的子类中，重写JComponent类的paint()方法，并在该方法中使用Graphics2D类的shear()方法和从Graphics类继承的drawImage()方法来实现的。

使用Graphics2D类的shear()方法可以实现图像的倾斜，该方法的定义如下：

public abstract void shear(double shx, double shy)

参数说明

● shx：在正x 轴方向移动坐标的乘数，它可以作为其y 坐标的函数。

● shy：在正y 轴方向移动坐标的乘数，它可以作为其x 坐标的函数。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的ShearImageFrame窗体类。

（3）在ShearImageFrame窗体类中，创建内部面板类ShearImagePanel，并重写JComponent类的 paint()方法，在该方法中使用从 Graphics 类继承的 drawImage()方法和 Graphics2D 类的 shear()方法实现图像倾斜。

（4）将内部面板类ShearImagePanel的实例添加到窗体类ShearImageFrame的内容面板上，用于在窗体上显示倾斜后的图像，代码如下：

```
class ShearImagePanel extends JPanel {          //绘制倾斜图像的面板类
  public void paint(Graphics g) {
    Graphics2D g2=(Graphics2D) g;          //获得Graphics2D对象
    g2.shear(0.5, 0);          //倾斜图像
    g2.drawImage(img, 10, 20, 220, 160, this);          //绘制指定大小的图片
  }
}
```

举一反三

根据本实例，读者可以实现以下程序。

使用Graphics2D对象的setTransform()方法实现图像倾斜。

将倾斜后的图像恢复到初始状态。

实例275 裁剪图片

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\07\Ex07_275

实例说明

本实例演示如何利用Java的绘图技术，实现裁剪图片的操作。运行程序，通过鼠标在分割面板的左侧选择图片的裁剪区域，将在分割面板的右侧显示裁剪区域中的图片，效果如图7.30所示。

图7.30 裁剪图片的效果

技术要点

本实例主要是通过Robot类的createScreenCapture()方法，以BasicStroke类创建虚线对象，并结合Graphics类的drawRect()方法实现图片的裁剪和绘制选区的。

（1）使用BasicStroke类重载的构造方法创建笔画对象，并指定笔画为虚线模式。

（2）使用Graphics类的drawRect()方法绘制矩形。

（3）使用Robot类的createScreenCapture()方法裁剪图片，该方法的定义如下：

public BufferedImage createScreenCapture(Rectangle screenRect)

参数说明

- screenRect：屏幕上被截取的矩形区域。
- 返回值：从屏幕上截取的缓冲图像对象。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的CutImageFrame窗体类。

（3）在CutImageFrame窗体类中，创建内部面板类OldImagePanel，并重写JComponent类的paint()方法，在该方法中使用Graphics类的drawImage()方法绘制原图片对象以及使用drawRect()方法绘制表示选择区域的矩形。

（4）在CutImageFrame窗体类中，创建内部面板类CutImagePanel，并重写JComponent类的paint()方法，在该方法中使用Graphics类的drawImage()方法绘制裁剪所得的图片。

（5）将内部面板类OldImagePanel和CutImagePanel的实例分别添加到分割面板的左右两侧，然后将分割面板添加到窗体类CutImageFrame的内容面板上，用于在窗体上显示原图片和裁剪所得的图片，OldImagePanel面板类的代码如下：

```
class OldImagePanel extends JPanel {          //创建绘制原图像的面板类
    public void paint(Graphics g) {
      Graphics2D g2 = (Graphics2D) g;
      g2.drawImage(img, 0, 0, this.getWidth(),
this.getHeight(), this);          //绘制图像
      g2.setColor(Color.WHITE);
      if (flag) {
        float[] arr ={ 5.0f };          //创建虚线模式的数组
```

```
        BasicStroke stroke = new BasicStroke(1,
    BasicStroke.CAP_BUTT, BasicStroke.JOIN_BEVEL, 1.0f,
    arr, 0);        //创建宽度是1的平头虚线笔画对象
        g2.setStroke(stroke);        //设置笔画对象
        g2.drawRect(pressPanelX, pressPanelY, releaseX -
    pressX, releaseY - pressY);        //绘制矩形选区
      }
    }
  }
```

CutImagePanel面板类的代码如下：

```
  class CutImagePanel extends JPanel {        //创建绘制裁
剪图像的面板类
    public void paint(Graphics g) {
      g.clearRect(0, 0, this.getWidth(),
  this.getHeight());        //清除绘图上下文的内容
      g.drawImage(buffImage, 0, 0, releaseX - pressX,
  releaseY - pressY, this);    //绘制图像
    }
  }
```

举一反三

根据本实例，读者可以开发以下程序。

裁剪图片。

将裁剪后的图片保存到磁盘。

7.5 颜色处理

实例276 调整图片的亮度

这是一个可以提高分析能力的实例

实例位置：光盘\mingrisoft\07\Ex07_276

实例说明

本实例演示如何利用Java的绘图技术，调整图片的亮度。运行程序，效果如图7.31所示，单击窗体上的“变亮”按钮，可以使图片变亮；单击“变暗”按钮，可以使图片变暗；单击“恢复”按钮，可以将图片恢复为原来的样式。

图7.31 调整图片的亮度

技术要点

本实例主要是通过在JPanel类的子类中，重写JComponent类的paint()方法，并在该方法中使用Graphics类的drawImage()方法绘制

缓冲图像，该缓冲图像是通过RescaleOp类的filter()方法进行缩放处理后，使图片变亮或变暗的图像。

（1）使用Graphics类的drawImage()方法绘制图像。

（2）使用RescaleOp类的filter()方法，对原缓冲图像对象进行重缩放，从而达到调整图片亮度的目的，filter()方法的定义如下：

public final BufferedImage filter(BufferedImage src, BufferedImage dst)

参数说明

- src：要过滤的源BufferedImage 对象。
- dst：目标BufferedImage 对象，或null。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的ImageBrightenFrame窗体类。

（3）在 ImageBrightenFrame 窗体类中，创建内部面板类ImageBrightenPanel，并重写JComponent类的paint()方法，在该方法中使用Graphics类的drawImage()方法绘制图像。

（4）将内部面板类ImageBrightenPanel的实例添加到窗体类ImageBrightenFrame的内容面板上，用于在窗体上显示变亮后的图像、原图像和变暗后的图像，代码如下：

```
class ImageBrightenPanel extends JPanel {
  public void paint(Graphics g) {        //重写paint()方法
    if (image != null) {
      g.drawImage(image, 0, 0, null); //将缓冲图像对象绘制到面板上
    }
```

```
  }
}
```
（5）“变亮”按钮的事件代码，用于使图片变亮，代码如下：
```
button.addActionListener(new ActionListener() {
  public void actionPerformed(final ActionEvent e) {
    float a = 1.0f;         //定义缩放因子
    float b = 5.0f;         //定义偏移量
    RescaleOp op = new RescaleOp(a,b,null); //创建具有指定缩放因子和偏移量的 RescaleOp对象
    image = op.filter(image, null);         //对源图像中的数据进行逐像素重缩放，达到变亮的效果
    repaint();          //重新绘制图像
  }
});
```
（6）“变暗”按钮的事件代码，用于使图片变暗，代码如下：
```
button_3.addActionListener(new ActionListener() {
  public void actionPerformed(final ActionEvent e) {
    float a = 1.0f;         //定义缩放因子
    float b =-5.0f;         //定义偏移量
    RescaleOp op = new RescaleOp(a,b,null); //创建具有指定缩放因子和偏移量的 RescaleOp对象
    image = op.filter(image, null);         //对源图像中的数据进行逐像素重缩放，达到变暗的效果
    repaint();          //重新绘制图像
  }
});
```
举一反三

根据本实例，读者可以开发以下程序。

使用RescaleOp类缩放图像。

调整亮度后截取图像。

实例277 转换彩色图片为灰度图片

这是一个可以启发思维的实例

实例位置：光盘\mingrisoft\07\Ex07_277

实例说明

本实例演示如何利用Java的绘图技术，将彩色图片转换为灰度图片。运行程序，将在窗体上显示彩色图片，单击窗体上的“转换为灰度”按钮，可以将窗体上的彩色图片转换为灰度图片，效果如图7.32所示。

图7.32 彩色图片转换为灰度图片

技术要点

本实例主要是通过在JPanel类的子类中，重写JComponent类的paint()方法，并在该方法中使用Graphics类的drawImage()方法绘制

缓冲图像，该缓冲图像是通过ColorConvertOp类的filter()方法进行颜色转换后得到的灰度图像。

（1）使用Graphics类的drawImage()方法绘制图像。

（2）使用ColorConvertOp类的构造方法创建ColorConvertOp对象，其构造方法定义如下：

public ColorConvertOp(ColorSpace srcCspace, ColorSpace dstCspace, RenderingHints hints)

参数说明

- srcCspace：原颜色空间对象。
- dstCspace：目标颜色空间对象。
- hints：用于控制颜色转换的RenderingHints 对象，或null。

（3）使用ColorConvertOp类的filter()方法，将彩色图像转换为灰度图像，该方法的定义如下：

public final BufferedImage filter(BufferedImage src, BufferedImage dst)

参数说明

- src：要过滤的源BufferedImage 对象。
- dst：目标BufferedImage 对象，或null。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的MultiColorToGrayFrame窗体类。

（3）在 MultiColorToGrayFrame 窗体类中，创建内部面板类ColorToGrayPanel，并重写JComponent类的paint()方法，在该方法中使用Graphics类的drawImage()方法绘制图像。

（4）将内部面板类ColorToGrayPanel的实例添加到窗体类MultiColorToGrayFrame的内容面板上，用于在窗体上显示原图像和转

换为灰度后的图像，代码如下：

```
class ColorToGrayPanel extends JPanel {
  public void paint(Graphics g) {        //重写paint()方法
    if (image != null) {
      g.drawImage(image, 0, 0, null);        //将缓冲图像对象绘制到面板上
    }
  }
}
```

（5）“转换为灰度”按钮的事件用于将彩色图片转换为灰度图片，代码如下：

```
button.addActionListener(new ActionListener() {
  public void actionPerformed(final ActionEvent e) {
    ColorSpace colorSpace1 = ColorSpace.getInstance(ColorSpace.CS_GRAY);        //创建内置线性为灰度的颜色空间
    ColorSpace colorSpace2 = ColorSpace
    .getInstance(ColorSpace.CS_LINEAR_RGB);        //创建内置线性为 RGB的颜色空间
    ColorConvertOp op = new ColorConvertOp(colorSpace1,colorSpace2, null); //创建进行颜色转换的对象
    image = op.filter(image, null);        //对缓冲图像进行颜色转换
    repaint();        //重新绘制图像
  }
```

});

举一反三

根据本实例，读者可以开发以下程序。

使用PixelGrabber类和ColorModel对象将图像转换为灰度。

还原图像。

实例278 使用像素值生成图像

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\07\Ex07_2785

实例说明

本实例演示如何使用像素值生成图像，效果如图7.33所示。

图7.33 使用像素值生成图像

技术要点

本实例主要是通过在JPanel类的子类中，重写JComponent类的paint()方法，并在该方法中使用 Graphics 类的 drawImage()方法绘制生成的图像，生成图像的方法是通过MemoryImageSource类创建ImageProducer对象，并把ImageProducer对象传递给从Component类继承的createImage()方法，从而实现了图像的创建。

（1）使用MemoryImageSource类创建ImageProducer对象，该类构造方法的定义如下：

public MemoryImageSource(int w, int h, int[] pix, int off, int scan)

参数说明

- w：像素矩形的宽度。
- h：像素矩形的高度。
- pix：像素数组。
- off：数组中存储第一个像素的偏移量。
- scan：数组中一行像素到下一行像素之间的距离，通常与像素矩形的宽度相同。

（2）使用从Component类继承的createImage()方法创建图像对象，该方法定义如下：

public Image createImage(ImageProducer producer)

参数说明

- producer：图像生成器。
- 返回值：该方法成功返回一个Image 对象。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的CreateImageFrame窗体类。

（3）在CreateImageFrame窗体类中，创建内部面板类CreateImagePanel，并重写JComponent类的paint()方法，在该方法中使用Graphics类的drawImage()方法绘制生成的图像。

（4）将内部面板类CreateImagePanel的实例添加到窗体类CreateImageFrame的内容面板上，用于在窗体上显示生成的图像，面板类的代码如下：

```
class CreateImagePanel extends JPanel
{                    //创建用像素值生成图像的面板类
  public void paint(Graphics g) {
    int w=300;                                    //宽度
    int h=220;                                    //高度
    int pix[]=new int[w * h];                          //存
储像素值的数组
    int index=0;                                   //存储
数组的索引
    for (int y=0;y<h;y++) {                         //在纵
向进行调整，从黑色渐变到红色
      int red= (y* 255) / (h - 1);                    //
  计算纵向的颜色值
      for (int x=0; x<w;x++) {                     //在横
  向进行调整，从黑色渐变到蓝色
        int blue= (x * 255) / (w - 1);                //
  计算横向的颜色值
        //通过移位运算和逻辑或运算计算像素值，并赋值给像
  素数组
        pix[index++] = (255 << 24) | (red << 16) | blue;
      }
    }
    //创建使用数组为Image生成像素值的ImageProducer对象
    ImageProducer imageProducer = new
MemoryImageSource(w, h, pix, 0, w);
    Image img = createImage(imageProducer);         //创建
图像对象
```

```
    g.drawImage(img, 0, 0,getWidth(),getHeight(),
this);       //绘制图像
  }
}
```

举一反三

根据本实例，读者可以开发以下程序。

使用PixelGrabber类获得图像像素在数组中的存储位置。

使用像素生成图像。

7.6 文字特效

实例279 立体效果的文字

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\07\Ex07_279

实例说明

本实例演示如何利用Java的绘图技术，实现立体效果文字的绘制。运行程序，将在窗体上显示具有立体效果的文字，效果如图7.34所示。

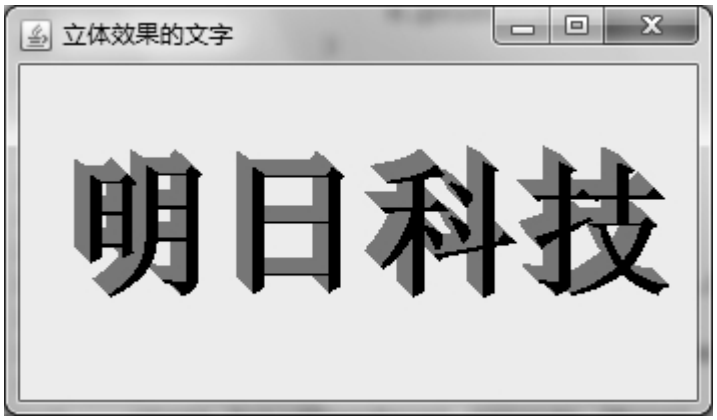


图7.34 立体效果的文字

技术要点

使用Graphics类的setFont()方法设置完字体、字形和字号后，使用Graphics类的setColor()方法将绘图上下文的前景色设置为灰色，然后使用Graphics类的drawString()方法绘制文本，并且每次绘制的文本都向右下方移动一小段距离，最后将绘图上下文的前景色更改为黑色，再绘制一次文本，从而实现立体字效果。

（1）使用Graphics类的setFont()方法，并将Font类创建的字体对象作为setFont()方法的参数，实现为文本设置字体的操作，setFont()方法的定义如下：

public abstract void setFont(Font font)

参数说明

font：为文本设置的字体对象。

（2）使用Graphics类的setColor()方法，并将Color类创建的颜色对象作为setColor()方法的参数，实现为文本和图形设置颜色的操作，setColor()方法的定义如下：

public abstract void setColor(Color color)

参数说明

color：为文本或图形设置的颜色对象。

（3）使用Graphics类的drawString()方法绘制文本。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的SolidTextFrame窗体类。

（3）在SolidTextFrame窗体类中创建内部面板类SolidTextPanel，并重写JComponent类的paint()方法，在该方法中使用Graphics类的setFont()、setColor()和drawString()方法，完成立体效果文字的绘制。

（4）将内部面板类 SolidTextPanel 的实例添加到窗体类 SolidTextFrame 的内容面板上，用于在窗体上显示绘制的立体字，代码如下：

```
class SolidTextPanel extends JPanel {          //创建内部面板类
    public void paint(Graphics g) {          //重写paint()方法
```

```
    String value ="明日科技";
    int x = 16;                    //文本位置的横坐标
    int y = 100;                    //文本位置的纵坐标
    Font font = new Font("宋体", Font.BOLD, 72);     //创
建字体对象
    g.setFont(font);                  //设置字体
    g.setColor(Color.GRAY);           //设置颜色为灰色
    int i = 0;                      //循环变量
    while (i<8){
      g.drawString(value, x, y);          //绘制文本
      x+=1;                      //调整绘制点的横坐标值
      y+=1;                      //调整绘制点的纵坐标值
      i++;                      //调整循环变量的值
    }
    g.setColor(Color.BLACK);          //设置颜色为黑色
    g.drawString(value, x, y);          //绘制文本
  }
}
```

举一反三

根据本实例，读者可以实现以下功能。

实现文本移动的动画。

实现旋转立体字体。

实现彩色立体字体。

实例280 阴影效果的文字

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\07\Ex07_280

实例说明

本实例演示如何利用Java的绘图技术，实现阴影效果文字的绘制。运行程序，将在窗体上显示具有阴影效果的文字，效果如图7.35所示。

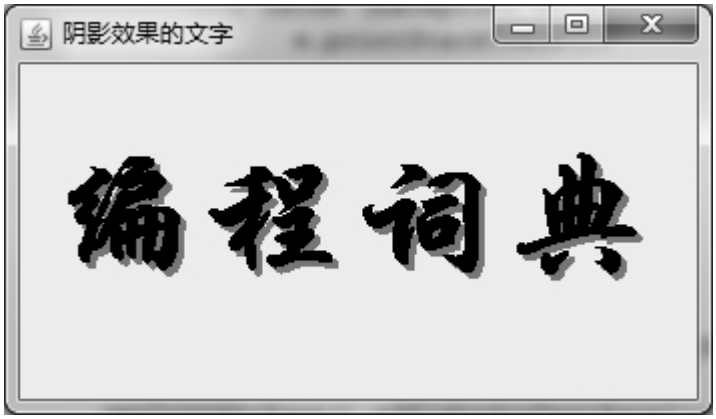


图7.35 阴影效果的文字

技术要点

使用Graphics类的setFont()方法设置完字体、字型和字号后，使用Graphics类的setColor()方法将绘图上下文的前景色设置为灰色，然后使用Graphics类的drawString()方法绘制文本，然后再将绘图上下文的前景色更改为黑色，并且将绘制的文本都向左上方移动一小段距离，从而实现阴影文字的效果。

（1）使用Graphics类的setFont()方法，并将Font类创建的字体对象作为setFont()方法的参数，实现为文本设置字体的操作。

（2）使用Graphics类的setColor()方法，并将Color类创建的颜色对象作为setColor()方法的参数，实现为文本和图形设置颜色的操作。

（3）使用Graphics类的drawString()方法绘制文本。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的ShadowTextFrame窗体类。

（3）在ShadowTextFrame窗体类中创建内部面板类ShadowTextPanel，并重写JComponent类的paint()方法，在该方法中使用Graphics类的setFont()、setColor()和drawString()方法，完成阴影效果文字的绘制。

（4）将内部面板类ShadowTextPanel的实例添加到窗体类ShadowTextFrame的内容面板上，用于在窗体上显示阴影效果的文字，代码如下：

```
    class ShadowTextPanel extends JPanel {          //创建内部面板类
        public void paint(Graphics g) {          //重写paint()方法
            String value ="编程词典";
            int x = 16;          //文本位置的横坐标
            int y = 100;          //文本位置的纵坐标
            Font font = new Font("华文行楷", Font.BOLD, 72);    //创建字体对象
            g.setFont(font);          //设置字体
            g.setColor(Color.GRAY);          //设置颜色为灰色
            int i = 0;          //循环变量
            g.drawString(value, x, y);          //绘制文本
            x -= 3;          //调整绘制点的横坐标值
            y -= 3;          //调整绘制点的纵坐标值
            g.setColor(Color.BLACK);          //设置颜色为黑色
            g.drawString(value, x, y);          //绘制文本
        }
```

}

举一反三

根据本实例，读者可以实现以下功能。

绘制彩色阴影文字。

绘制倾斜的阴影文字。

实例281 倾斜效果的文字

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\07\Ex07_281

实例说明

本实例演示如何利用Java的绘图技术，实现倾斜效果文字的绘制。运行程序，将在窗体上显示具有倾斜效果的文字，效果如图7.36所示。

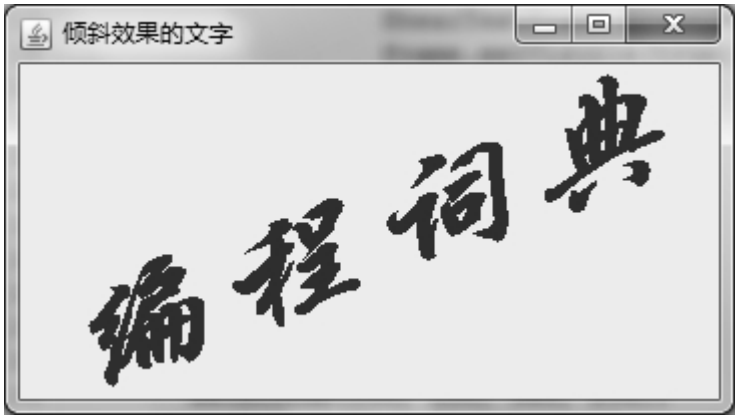


图7.36 倾斜效果的文字

技术要点

使用 Graphics2D 类的 setShear()方法，倾斜绘图上下文对象，然后使用从 Graphics 类继承的setFont()方法设置字体、字形和字号，使用drawString()方法绘制文本，从而实现倾斜文字的效果。

（1）使用从Graphics类继承的setFont()方法设置绘图上下文的字体。

（2）使用 Graphics2D 类的 setShear()方法，使绘图上下文倾斜，这样绘制的文本就是倾斜的文本，该方法的定义如下：

public abstract void set shear(double shx, double shy)

参数说明

● shx：在正x 轴方向移动坐标的乘数，它可以作为其y 坐标的函数。

● shy：在正y 轴方向移动坐标的乘数，它可以作为其x 坐标的函数。

（3）使用从Graphics类继承的drawString()方法绘制文本。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的ShearTextFrame窗体类。

（3）在ShearTextFrame窗体类中创建内部面板类ShearTextPanel，并重写JComponent类的paint()方法，在该方法中使用Graphics2D类的setShear()方法，以及从Graphics类继承的setFont()方法和drawString()方法，完成倾斜文字的绘制。

（4）将内部面板类ShearTextPanel的实例添加到窗体类ShearTextFrame的内容面板上，用于在窗体上显示倾斜效果的文字，代码如下：

```
class ShearTextPanel extends JPanel {          //创建内部面板类
    public void paint(Graphics g) {          //重写paint()方法
        Graphics2D g2 = (Graphics2D)g;          //转换为Graphics2D类型
```

```
        String value ="编程词典";          //绘制的文本
        int x = 10;          //文本位置的横坐标
        int y = 170;          //文本位置的纵坐标
        Font font = new Font("华文行楷", Font.BOLD,
    72);     //创建字体对象
        g2.setFont(font);          //设置字体
        g2.shear(0.1, -0.4);          //倾斜画布
        g2.drawString(value, x, y);          //绘制文本
      }
    }
```

举一反三

根据本实例，读者可以开发以下程序。

向下倾斜的文字。

向两端倾斜的文字。

实例282 渐变效果的文字

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\07\Ex07_282

实例说明

在程序当中绘制的文本信息，通常都是单一的颜色，本实例使用GradientPaint类创建封装渐变颜色的对象，并为绘图上下文指定该对象，从而实现绘制渐变效果文字的功能。运行程序，将在窗体上显示具有渐变效果的文字，效果如图7.37所示。

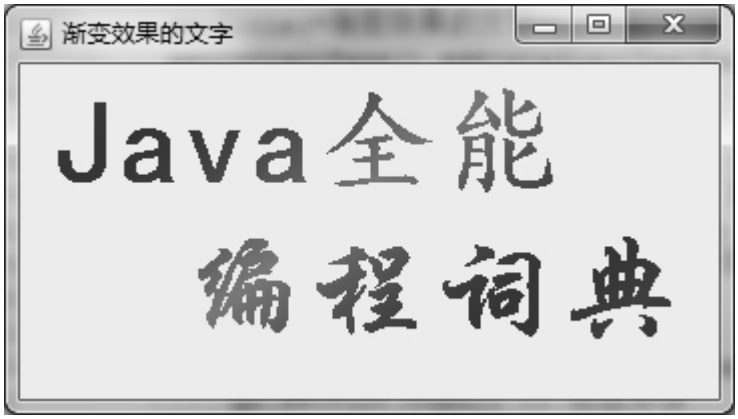


图7.37 渐变效果的文字

技术要点

本实例主要是通过在JPanel类的子类中，重写JComponent类的paint()方法，并在该方法中使用Graphics2D类的setPaint()方法，为绘图上下文指定GradientPaint类创建的渐变色对象，从而实现绘制渐变效果文字的功能。

（1）使用Graphics2D类的setPaint()方法，并将GradientPaint类创建的封装渐变颜色的对象作为setPaint()方法的参数，实现为图形填充渐变色的操作，setPaint()方法的定义如下：

```
public abstract void setPaint(Paint paint)
```

参数说明

paint：封装了渐变颜色的Paint对象。

（2）使用GradientPaint类创建封装渐变颜色的对象。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的GradientTextFrame窗体类。

（3）在GradientTextFrame窗体类中，创建内部面板类GradientTextPanel，并重写JComponent类的paint()方法，在该方法中使用Graphics2D类的setPaint()方法，为绘图上下文指定渐变色，

并使用从Graphics类继承的drawString()绘制文本，完成绘制渐变颜色效果的文字。

（4）将内部面板类GradientTextPanel的实例添加到窗体类GradientTextFrame的内容面板上，用于在窗体上显示渐变效果的文字，代码如下：

```
class GradientTextPanel extends JPanel {          //创建内部面板类
    public void paint(Graphics g) {          //重写paint()方法
        Graphics2D g2 = (Graphics2D) g;          //转换为Graphics2D类型
        String value ="Java全能";          //绘制的文本
        int x = 15;          //文本位置的横坐标
        int y = 60;          //文本位置的纵坐标
        Font font = new Font("楷体", Font.BOLD, 60); //创建字体对象
        g2.setFont(font);          //设置字体
        //创建循环渐变的GradientPaint对象
        GradientPaint paint = new GradientPaint(20, 20, Color.BLUE, 100,120, Color.RED, true);
        g2.setPaint(paint);          //设置渐变
        g2.drawString(value, x, y);          //绘制文本
        font = new Font("华文行楷", Font.BOLD, 60); //创建新的字体对象
        g2.setFont(font);          //设置字体
        x = 80;          //文本位置的横坐标
        y = 130;          //文本位置的纵坐标
```

```
        value ="编程词典";          //绘制的文本
        g2.drawString(value, x, y);          //绘制文本
    }
}
```

举一反三

根据本实例，读者可以开发以下程序。

实现让颜色快速渐变。

具有渐变效果的阴影文字。

实例283 会变色的文字

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\07\Ex07_283

实例说明

本实例使用Java的绘图和多线程技术，实现在程序运行时，动态改变文字的颜色。运行程序，在窗体上绘制的文字颜色不断改变，效果如图7.38所示。

图7.38 会变色的文字

技术要点

本实例主要是通过在JPanel类的子类中，重写JComponent类的paint()方法，并实现Runnable接口中的run()方法实现的，其中paint()方法中添加的代码用于绘制文本，run()方法中的代码用于随机获得颜色的RGB值，并创建Color颜色对象。

（1）使用 Random 类的实例生成伪随机数流，并使用该类的nextInt()方法，随机产生颜色的RGB值。

（2）使用从Graphics类继承的setColor()方法设置颜色，该颜色是通过RGB值创建的Color对象。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的ChangeColorTextFrame窗体类。

（3）在ChangeColorTextFrame窗体类中，创建内部面板类ChangeColorTextPanel，在面板类中重写JComponent类的paint()方法，并实现Runnable接口的run()方法，从而实现会变色文字的绘制。

（4）将内部面板类ChangeColorTextPanel的实例添加到窗体类ChangeColorTextFrame的内容面板上，用于在窗体上显示会变色的文字，代码如下：

```
class ChangeColorTextPanel extends JPanel implements Runnable { //创建内部面板类
    Color color = new Color(0,0,255);
    public void paint(Graphics g) {          //重写paint()方法
        Graphics2D g2 = (Graphics2D) g;          //转换为Graphics2D类型
        String value =″《视频学Java编程》″;          //绘制的文本
```

```
        int x = 2;          //文本位置的横坐标
        int y = 90;          //文本位置的纵坐标
        Font font = new Font("楷体", Font.BOLD,
40);          //创建字体对象
        g2.setFont(font);          //设置字体
        g2.setColor(color);          //设置颜色
        g2.drawString(value, x, y);          //绘制文本
    }
    public void run() {
        Random random = new Random();          //创建随机数对象
        while(true){
            int R = random.nextInt(256);          //随机产生颜色
    的R值
            int G = random.nextInt(256);          //随机产生颜色
    的G值
            int B = random.nextInt(256);          //随机产生颜色
    的B值
            color = new Color(R,G,B);          //创建颜色对象
            repaint();          //调用paint()方法
            try {
                Thread.sleep(300);          //休眠300毫秒
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

举一反三

根据本实例，读者可以开发以下程序。

动态改变字体的颜色。

制作颜色渐变的字体。

实例284 水印文字特效

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\07\Ex07_284

实例说明

水印文字是通过改变绘图上下文的透明度实现的，本实例将演示水印文字的实现。运行程序，将在窗体上绘制图片，并为图片添加水印文字，效果如图7.39所示。

图7.39 水印文字特效

技术要点

本实例主要是通过 Graphics2D类的 setComposite()方法，为绘图上下文指定表示透明度的AlphaComposite对象实现的。

（1）使用 AlphaComposite 类获得表示透明度的 AlphaComposite 对象，该对象使用AlphaComposite类的字段SrcOver

调用derive()方法获得，该方法的定义如下：

public AlphaComposite derive(float alpha)

参数说明

● alpha：闭区间0.0f ～1.0f 之间的一个浮点数字，为0.0f 时完全透明，为1.0f 时不透明。

● 返回值：表示透明度的AlphaComposite 对象。

（2）使用 Graphics2D 类的 setComposite()方法，为绘图上下文指定表示透明度的AlphaComposite对象，该方法的定义如下：

public abstract void setComposite(Composite comp)

参数说明

comp：表示透明度的AlphaComposite对象。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的WatermarkTextFrame窗体类。

（3）在 WatermarkTextFrame 窗体类中，创建内部面板类 WatermarkTextPanel ，并重写JComponent类的paint()方法，在该方法中使用Graphics2D类的setComposite()方法设置透明度，然后绘制文本实现水印文字的绘制。

（4）将内部面板类WatermarkTextPanel的实例添加到窗体类WatermarkTextFrame的内容面板上，用于在窗体上显示绘制的水印文字，代码如下：

```
class WatermarkTextPanel extends JPanel {
  public void paint(Graphics g) {
    Graphics2D g2 = (Graphics2D)g;        //获得
Graphics2D对象
```

```
        g2.drawImage(img, 0, 0, 300, 237, this);          //绘制图像
        g2.rotate(Math.toRadians(-30));          //旋转绘图上下文对象
        Font font = new Font("楷体",Font.BOLD,60);          //创建字体对象
        g2.setFont(font);          //指定字体
        g2.setColor(Color.WHITE);          //指定颜色
        AlphaComposite alpha = AlphaComposite.SrcOver.derive(0.3f); //获得表示透明度的AlphaComposite对象
        g2.setComposite(alpha);          //指定AlphaComposite对象
        g2.drawString("编程词典", -60, 180);          //绘制文本，实现水印
    }
}
```

举一反三

根据本实例，读者可以开发以下程序。

使用AlphaComposite对象表示透明度。

使用AlphaComposite类提供的getInstance()方法表示透明度。

实例285 顺时针旋转文字

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\07\Ex07_285

实例说明

本实例演示如何在Java中顺时针旋转文字。运行程序，效果如图7.40所示。

图7.40 顺时针旋转文字

技术要点

本实例的实现主要是通过文本组件的CaretListener监听器，该监听器用于监听文本组件插入符的位置是否更改，如果更改了就会执行caretUpdate()方法，同时，为了能够顺时针绘制文本，需要计算出每个文字应该在哪个角度进行绘制，因此可以用360除以文字个数，计算出每个文字的角度。

（1）监听器接口 CaretListener 的 caretUpdate()方法，用于监听文本组件插入符的位置是否改变，该方法的定义如下：

```
void caretUpdate(CaretEvent e)
```

参数说明

e：插入符事件。

（2）计算每个文字的绘制角度，假设有6个字，要计算每个字的绘制角度，就可以用360/6计算出相邻两个字的角度差是60°。例如：

```
String s= "全能编程词典";                                    //声明字符串
```

```
    int len=
s.length;                                          //获得字符串的
长度，即字符个数
    double=360D / len;                                          //
计算出每个字的角度
```

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的ClockwiseTextFrame窗体类。

（3）在 ClockwiseTextFrame 窗体类中，创建内部面板类ClockwiseTextPanel，并重写JComponent类的paint()方法，在该方法中添加代码，实现顺时针旋转文本的功能。

（4）将内部面板类ClockwiseTextPanel的实例添加到窗体类ClockwiseTextFrame的内容面板上，用于在窗体上顺时针绘制文本，面板类ClockwiseTextPanel的代码如下：

```
    class ClockwiseTextPanel extends JPanel
{                              //创建内部面板类
      private String text;
      public ClockwiseTextPanel() {
        setOpaque(false);
    //设置面板为透明
        setLayout(null);                                        /
    /设置为绝对布局
      }
      public String getText() {
        return
    text;                                              //获得成员变
```

```
量的值
  }
  public void setText(String text) {
    this.text=
text;                                          //为成员变量赋
值
    repaint();                                          /
/调整paint()方法
  }
  public void paint(Graphics g)
{                                  //重写paint()方法
    Graphics2D g2= (Graphics2D)
g;                             //获得Graphics2D的实例
    int
width=getWidth();                                  //获得面
板的宽度
    int
height=getHeight();                                //获得面
板的高度
    if (text != null) {
      char[] array=
text.toCharArray();                        //将文本转换为字
符数组
      int len= array.length *
5;                              //定义圆的半径，同时可以调
整文字的距离
```

```
    Font font=new Font("宋
体",Font.BOLD,22);           //创建字体
    g2.setFont(font);                                    //
设置字体
    double angle=0;                                      //定
义初始角度
    for (int i=0; i< array.length; i++)
{                    //遍历字符串中的字符
      if (i == 0) {
        g2.setColor(Color.BLUE);                         //第1
    个字符用蓝色
      } else {
        g2.setColor(Color.BLACK);                        //其
    他字符用黑色
      }
      int x= (int) (len
  *Math.sin(Math.toRadians(angle+270)));  //计算每个文
  字的横坐标位置
      int y= (int) (len
  *Math.cos(Math.toRadians(angle+270)));  //计算每个文
  字的纵坐标位置
      g2.drawString(array[i]+ "",width / 2+x,height / 2
  - y);       //绘制字符
      angle= angle+360d /
  array.length;                        //改变角度
    }
  }
```

```
    }
  }
```

（5）本实例使用监听器接口CaretListener，对文本组件的插入符进行监听，当插入符位置改变后，程序就会响应插入符事件，并执行caretUpdate()方法，文本组件的插入符事件代码如下：

```
  textField.addCaretListener(new CaretListener() {
    public void caretUpdate(CaretEvent arg0) {
      String text=
  textField.getText();                          //获取文本
  框字符串
      clockwiseTextPanel.setText(text);
      //为面板中的text变量赋值
    }
  });
```

举一反三

根据本实例，读者可以实现以下功能。

停止旋转文字。

逆时针旋转文字。

实例286 动态绘制文本

这是一个可以启发思维的实例

实例位置：光盘\mingrisoft\07\Ex07_286

实例说明

本实例演示如何在Java中利用绘图技术，动态绘制文本，也就是将一个文件中的文本逐字绘制到程序界面上。运行程序，将在窗体上动态绘制文本，效果如图7.41所示。

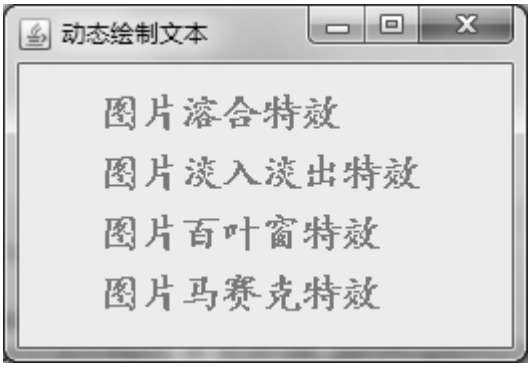


图7.41 动态绘制文本

技术要点

本实例的实现主要是使用线程和BufferedReader缓冲流，从指定文件中读取一个字符，然后绘制该字符，并改变下一个字符绘制点的x、y坐标值，最终完成动态绘制文本的功能。

（1）使用System类的getProperty()方法，并为其传递实参字符串user.dir，这样就可以获得项目的当前路径，getProperty()方法的定义如下：

public static String getProperty(String key)

参数说明

● key：系统属性的名称。

● 返回值：返回系统属性的字符串值，如果没有指定键的属性，则返回null。

（2）使用BufferedReader类的read()方法，从文本中读取一个字符，该方法的定义如下：

public int read() throws IOException

参数说明

● 返回值：作为一个范围从0～65535 整数读入的字符，如果已到达流末尾，则返回-1。

● IOException：如果发生I/O 错误，则抛出IOException 异常。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的DynamicDrawText窗体类。

（3）在 DynamicDrawText 窗体类中，创建内部面板类DynamicDrawTextPanel，实现Runnable接口中的run()方法，并重写JComponent类的paint()方法，在该方法中完成动态绘制文本的操作。

（4）将内部面板类 DynamicDrawTextPanel 的实例添加到窗体类DynamicDrawText 的内容面板上，用于在窗体上动态绘制文本，代码如下：

```
    class DynamicDrawTextPanel extends JPanel implements
Runnable {
      private BufferedReader
  read;                              //声明缓冲流对象
      int x=20;                                          //起始
  点的x坐标
      int y=30;                                          //起始
  点的y坐标
      String value ="";
      public DynamicDrawTextPanel(){
        String
    projectPath=System.getProperty("user.dir");       //获得当
    前项目
        String filePath=projectPath+
    "/src/com/zzk/dyn.txt";     //获得项目中loadText.java文件
    的完整路径
        InputStream in = null;
```

```
    try {
      in=new FileInputStream(filePath);                    //创
  建输入流对象
    } catch (FileNotFoundException e) {
      e.printStackTrace();
    }
    read=new BufferedReader(new
InputStreamReader(in));  //创建缓冲流对象
  }
  public void paint(Graphics g) {
    Font font=new Font("华文楷体",Font.BOLD,20);      //
创建字体对象
    g.setFont(font);                                    //指
定字体
    g.setColor(Color.RED);                              //指
定颜色
    g.drawString(value,x,y);                            //绘
制文本
  }
  public void run() {
    int len=0;                                         //存储
读取的字符
    try {
      while ((len= read.read()) != -1)
  {                  //读取内容
        Thread.sleep(400);                             //当前线
    程休眠400毫秒
```

```
            value=String.valueOf((char) len);          //获得
        读取的内容
            if (value.equals("\n") ||value.equals("\r"))
        {    //是回车或换行符
              x=20;                                    //下一行起始点
          的x坐标
              y+=15;                                   //下一行文本的y
          坐标
            } else {                                     //不是回车
        或换行符
              x+=20;                                   //当前行下一个
          字的x坐标
            }
            repaint();                                   //调用
        paint()方法
          }
        } catch (Exception e) {
          e.printStackTrace();
        }
      }
    }
```

举一反三

根据本实例，读者可以开发以下程序。

实现字节流到字符流的转换。

动态绘制彩色文字。

实例287 中文验证码

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\07\Ex07_287

实例说明

为了保证软件的安全性，通常要求在登录界面中输入验证码，为此本实例演示了如何实现中文验证码。运行程序，输入正确的用户名mrsoft、密码 mrsoft和中文验证码即可登录，如图7.42所示。

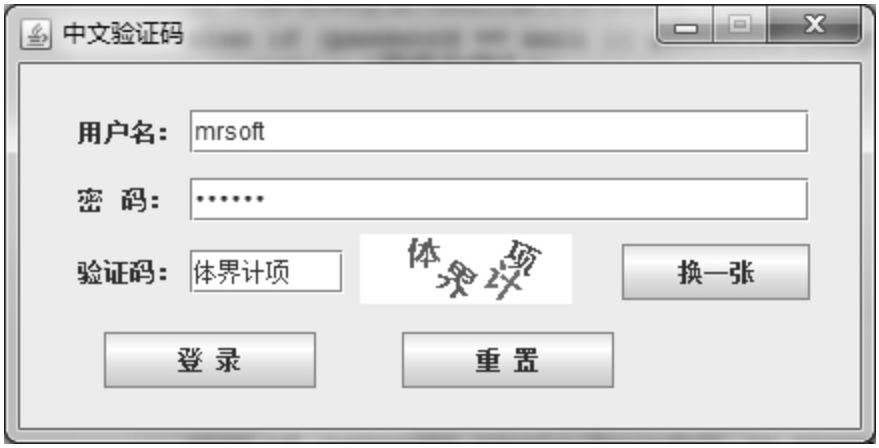


图7.42 中文验证码

技术要点

本实例以字符串存储作为验证码的汉字，然后使用Random类获得随机数，作为字符串中字符的索引值，并使用String类的substring()方法，从字符串中截取一个汉字作为验证码。

使用String类的substring()方法，可截取字符串中的字符，该方法的定义如下：

public String substring(int beginIndex, int endIndex)

参数说明

- beginIndex：起始索引（包括）。
- endIndex：结束索引（不包括）。
- 返回值：起始索引和终止索引之间的子字符串。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 MainFrame 窗体类和一个继承 Jpanel 类的ChineseCodePanel面板类。

（3）在面板类 ChineseCodePanel 中，重写 JComponent 类的paint()方法，在该方法中向BufferedImage对象上绘制中文验证码，完成中文验证码的绘制。

（4）面板类ChineseCodePanel的paint()方法代码如下：

```
public void paint(Graphics g) {
  String hanZi= "编程词典集学查用界面设计项目开发等内容于一体";    //定义验证码使用的汉字
  BufferedImage image=new BufferedImage(WIDTH,HEIGHT,BufferedImage.TYPE_INT_RGB);
                                    //实例化BufferedImage
  Graphics gs= image.getGraphics();                                    //获取Graphics类的对象
  if (!num.isEmpty()) {
    num= "";                                                          //清空验证码
  }
  Font font=new Font("黑体",Font.BOLD,20);                            //通过Font构造字体
  gs.setFont(font);
      //设置字体
  gs.fillRect(0,0,WIDTH,HEIGHT);
      //填充一个矩形
  //输出随机的验证文字
```

```
  for (int i = 0; i < 4; i++) {
    int index=
random.nextInt(hanZi.length());                    //随
机获得汉字的索引值
    String ctmp
=hanZi.substring(index,index+1);                   //获得
指定索引处的一个汉字
    num+=
ctmp;                                              //更新验
证码
    Color color=new Color(20+ random.nextInt(120),20+
random.nextInt(120),20+
random.nextInt(120));                  //生成随机颜色
    gs.setColor(color);
      //设置颜色
    Graphics2D gs2d= (Graphics2D)
gs;                              //将文字旋转指定角度
    AffineTransform trans=new
AffineTransform();                     //实例化
AffineTransform
    trans.rotate(random.nextInt(45) * 3.14 / 180,22 *
i+8,7);
    float scaleSize=
random.nextFloat()+0.8f;                           //缩放文字
    if (scaleSize > 1f)
      scaleSize=1f;
    //如果scaleSize大于1，则等于1
```

```
        trans.scale(scaleSize,
    scaleSize);                                    //进行缩放
        gs2d.setTransform(trans);
          //设置AffineTransform对象
        gs.drawString(ctmp,WIDTH / 6 * i+28,HEIGHT /
    2);                  //绘制出验证码
      }
      g.drawImage(image,0,0,null);
          //在面板中绘制出验证码
    }
```

举一反三

根据本实例，读者可以开发以下程序。

生成没有重复文字的验证码。

生成字母、数字验证码。

实例288 图片验证码

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\07\Ex07_288

实例说明

本实例演示了如何实现图片验证码。运行程序，输入正确的用户名mrsoft、密码mrsoft和验证码，即可登录，如图7.43所示。

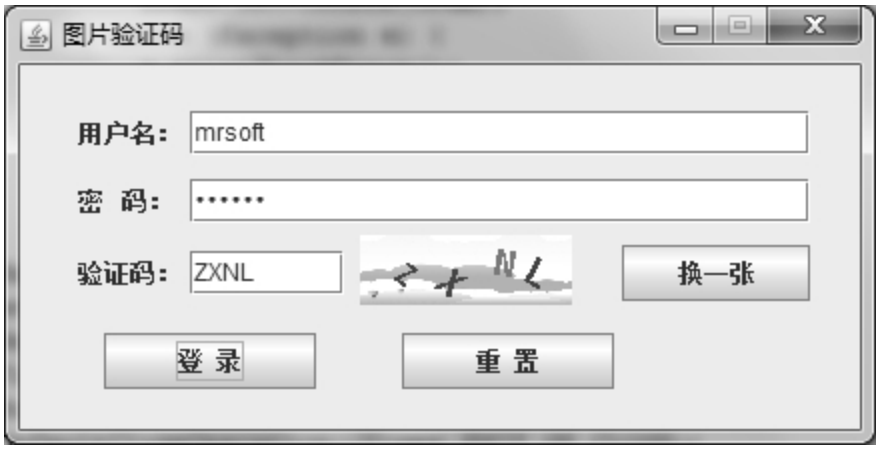


图7.43 图片验证码

技术要点

本实例通过BufferedImage类绘制图像，然后使用Random类随机获得A~Z的大写字母，并绘制到BufferedImage对象上，从而生成图片验证码。

（1）由于大写字母A～Z对应的Unicode字符集的编号为65～90，所以要产生65～90之间的随机整数，可以使用下面的代码实现：

```
Random random = new Random();          //创建Random对象
int code = random.nextInt(26) + 65;          //生成65～90的随机整数
```

（2）通过强制类型转换将65～90的随机整数转换为大写字母，可以使用下面的代码实现：

```
Random random = new Random();          //创建Random对象
int code = random.nextInt(26) + 65;          //生成65～90的随机整数
char letter = (char) code;          //转换为大写字母
```

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 MainFrame 窗体类和一个继承 Jpanel 类的ImageCodePanel面板类。

（3）在面板类 ImageCodePanel 中，重写 JComponent 类的paint()方法，在该方法中向BufferedImage对象上绘制图片和验证码，完成图片验证码的绘制。

（4）面板类ImageCodePanel的paint()方法代码如下：

```
public void paint(Graphics g) {
  BufferedImage image=new
BufferedImage(WIDTH,HEIGHT,BufferedImage.TYPE_INT_RGB);
                //实例化BufferedImage
  Graphics gs=
image.getGraphics();                          //获取
Graphics类的对象
  if (!num.isEmpty()) {
    num= "";                                          //
清空验证码
  }
  Font font=new Font("黑
体",Font.BOLD,20);                    //通过Font构造字体
  gs.setFont(font);
//设置字体
  gs.fillRect(0,0,WIDTH,HEIGHT);
//填充一个矩形
  Image img = null;
  try {
    img= ImageIO.read(new
File("src/img/image.jpg"));          //创建图像对象
  } catch (IOException e) {
    e.printStackTrace();
```

```
    }
    image.getGraphics().drawImage(img,0,0,WIDTH,HEIGHT,null
);  //在缓冲图像对象上绘制图像
    //输出随机的验证文字
    for (int i = 0; i < 4; i++) {
      char ctmp= (char)
  (random.nextInt(26)+65);                    //生成A~Z的字母
      num+= ctmp;                                              //更
  新验证码
      Color color=new Color(20+ random.nextInt(120),20+
  random.nextInt(120),20+
  random.nextInt(120));                  //生成随机颜色
      gs.setColor(color);
  //设置颜色
      Graphics2D gs2d= (Graphics2D)
  gs;                           //将文字旋转指定角度
      AffineTransform trans=new
  AffineTransform();                  //实例化AffineTransform
      trans.rotate(random.nextInt(45) * 3.14 / 180, 22 * i
  + 8, 7);
      float scaleSize=
  random.nextFloat()+0.8f;                      //缩放文字
      if (scaleSize > 1f)
        scaleSize=1f;                                          //如
    果scaleSize大于1，则等于1
      trans.scale(scaleSize,
  scaleSize);                             //进行缩放
```

```
            gs2d.setTransform(trans);                    //设
    置AffineTransform对象
            gs.drawString(String.valueOf(ctmp), WIDTH / 6 * i +
    28, HEIGHT / 2); //绘制出验证码
          }
          g.drawImage(image,0,0,null);                    //
    在面板中绘制出验证码
        }
```

举一反三

根据本实例，读者可以开发以下程序。

绘制图片验证码。

绘制带干扰线的验证码。

实例289 带干扰线的验证码

这是一个可以提高分析能力的实例

实例位置：光盘\mingrisoft\07\Ex07_289

实例说明

本实例演示了如何实现带干扰线的验证码。运行程序，输入正确的用户名 mrsoft、密码mrsoft和验证码，即可登录，如图7.44所示。

图7.44 带干扰线的验证码

技术要点

本实例通过BufferedImage类绘制图像，然后绘制两条首尾相连的线段，最后使用Random类随机获得A~Z的大写字母，并绘制到BufferedImage对象上，从而生成带干扰线的验证码。

（1）使用Graphics类的drawLine()方法绘制干扰线。

（2）使用Graphics类的drawString()方法，绘制随机生成的大写字母，完成带干扰线的验证码的绘制。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 MainFrame 窗体类和一个继承 Jpanel 类的DisturbCodePanel面板类。

（3）在面板类 DisturbCodePanel 中，重写 JComponent 类的paint()方法，使用该方法在BufferedImage对象上绘制图片、干扰线和验证码，完成带干扰线验证码的绘制。

（4）面板类DisturbCodePanel的paint()方法代码如下：

```
public void paint(Graphics g) {
  BufferedImage image=new
BufferedImage(WIDTH,HEIGHT,BufferedImage.TYPE_INT_RGB);
                //实例化BufferedImage
  Graphics gs=
image.getGraphics();                          //获取
Graphics类的对象
  if (!num.isEmpty()) {
    num= "";                                          //
清空验证码
  }
```

```
    Font font=new Font("黑
体",Font.BOLD,20);                    //通过Font构造字体
    gs.setFont(font);
  //设置字体
    gs.fillRect(0,0,WIDTH,HEIGHT);
  //填充一个矩形
    Image img = null;
    try {
      img= ImageIO.read(new
  File("src/img/image.jpg"));          //创建图像对象
    } catch (IOException e) {
      e.printStackTrace();
    }
    image.getGraphics().drawImage(img,0,0,WIDTH,HEIGHT,null
);  //在缓冲图像对象上绘制图像
    int startX1= random.nextInt(20);            //随机获取
第一条干扰线起点的x坐标
    int startY1= random.nextInt(20);            //随机获取
第一条干扰线起点的y坐标
    int startX2= random.nextInt(30)+35;          //随机获取
第一条干扰线终点的x坐标，也是第二条干扰线起点的x坐标
    int startY2= random.nextInt(10)+20;          //随机获取
第一条干扰线终点的y坐标，也是第二条干扰线起点的y坐标
    int startX3= random.nextInt(30)+90;          //随机获取
第二条干扰线终点的x坐标
    int startY3= random.nextInt(10)+5;          //随机获取
第二条干扰线终点的y坐标
```

```
    gs.setColor(Color.RED);
    gs.drawLine(startX1, startY1, startX2, startY2);  //绘制第一条干扰线
    gs.setColor(Color.BLUE);
    gs.drawLine(startX2, startY2, startX3, startY3);  //绘制第二条干扰线
    //输出随机的验证文字
    for (int i = 0; i < 4; i++) {
      char ctmp= (char)(random.nextInt(26)+65);          //生成A~Z的字母
      num+= ctmp;                                        //更新验证码
      Color color=new Color(20+ random.nextInt(120),20+ random.nextInt(120),20+ random.nextInt(120));          //生成随机颜色
      gs.setColor(color);                                //设置颜色
      Graphics2D gs2d= (Graphics2D) gs;                  //将文字旋转指定角度
      AffineTransform trans=new AffineTransform();       //实例化AffineTransform
      trans.rotate(random.nextInt(45) * 3.14 / 180, 22 * i + 8, 7);
      float scaleSize= random.nextFloat()+0.8f;          //缩放文字
      if (scaleSize > 1f)
```

```
        scaleSize=1f;                                    //如
  果scaleSize大于1，则等于1
      trans.scale(scaleSize,
scaleSize);                              //进行缩放
      gs2d.setTransform(trans);
  //设置AffineTransform对象
      gs.drawString(String.valueOf(ctmp),WIDTH / 6 *
i+28,HEIGHT / 2);  //绘制出验证码
  }
  g.drawImage(image,0,0,null);
  //在面板中绘制出验证码
}
```

举一反三

根据本实例，读者可以开发以下程序。

带干扰线的数字验证码。

带干扰线的中文验证码。

7.7 图片特效

实例290 纹理填充特效

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\07\Ex07_290

实例说明

在使用Word文档进行绘图时，可以对图形进行纹理填充，本实例使用Java的绘图技术，实现了图形的纹理填充效果。运行程序，效果如图7.45所示。

技术要点

本实例使用TexturePaint类创建纹理填充对象，然后使用Graphics2D类的setPaint()方法，将该纹理填充对象设置为绘图上下文的Paint属性，这样再绘制填充图形时，将以TexturePaint类指定的纹理进行填充。

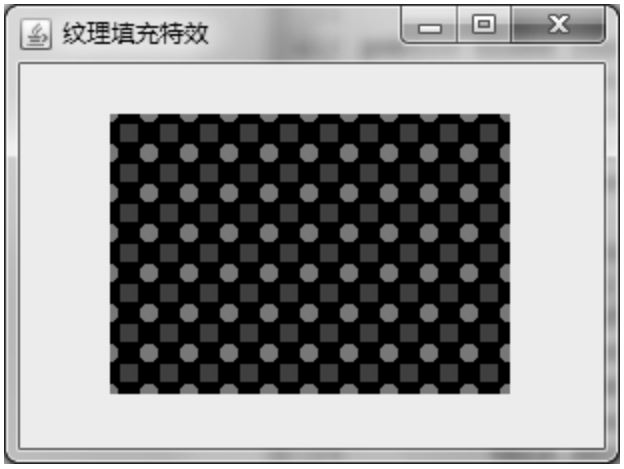


图7.45 纹理填充特效

（1）使用TexturePaint类创建纹理填充对象，该类构造方法的定义如下：

public TexturePaint(BufferedImage txtr, Rectangle2D anchor)

参数说明

- txtr：绘制有纹理的BufferedImage 对象。
- anchor：用于定位和复制纹理的Rectangle2D 对象。

（2）使用Graphics2D类的setPaint()方法，为绘图上下文的Paint属性。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的TextureFillFrame窗体类。

（3）在TextureFillFrame窗体类中，创建内部面板类TextureFillPanel，并重写JComponent类的paint()方法，在该方法中实现图形的纹理填充特效。

（4）将内部面板类TextureFillPanel的实例添加到窗体类TextureFillFrame的内容面板上，用于在窗体上显示填充有纹理的图形，代码如下：

```
class TextureFillPanel extends JPanel {
  public void paint(Graphics g) {
    BufferedImage image=new
BufferedImage(200,200,BufferedImage.TYPE_INT_RGB);
          //创建缓冲图像对象
    Graphics2D g2=
image.createGraphics();                    //获得缓冲图像对象的绘图上下文对象
```

```
        g2.setColor(Color.BLUE);
    //设置颜色
        g2.fillRect(0,0,90,90);
    //绘制填充矩形
        g2.setColor(Color.RED);
//设置颜色
        g2.fillOval(95,95,90,90);
    //绘制带填充色的圆形
        Rectangle2D rect=new
Rectangle2D.Float(10,10,20,20);       //创建Rectangle2D对
象
        TexturePaint textPaint=new
TexturePaint(image,rect);          //创建纹理填充对象
        Graphics2D graphics2=
(Graphics2D)g;                          //转换paint()方法的绘图
上下文对象
        graphics2.setPaint(textPaint);
          //为绘图上下文对象设置纹理填充对象
        Rectangle2D.Float ellipse2=new
Rectangle2D.Float(45,25,200,140);  //创建矩形对象
        graphics2.fill(ellipse2);
        //绘制填充纹理的矩形
    }
}
```

举一反三

根据本实例，读者可以实现以下功能。

改变纹理填充图案的大小。

改变纹理的填充颜色。

实例291 水波效果的图片

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\07\Ex07_291

实例说明

本实例演示在 Java 中绘制图像时，如何实现水波效果的图片特效。运行程序，效果如图7.46所示。

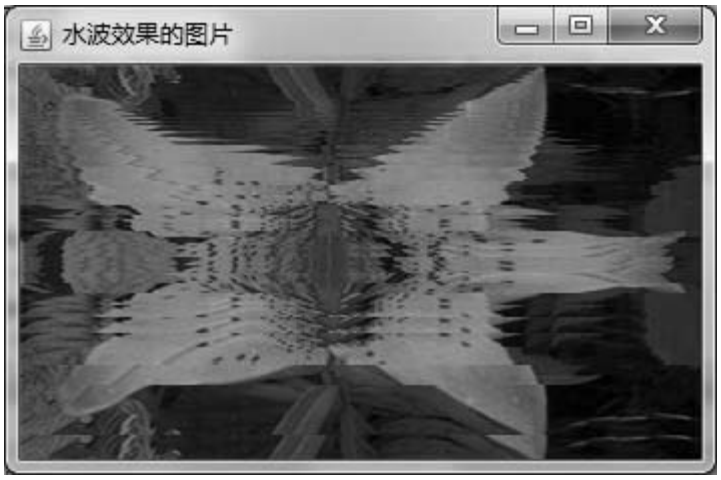


图7.46 水波效果的图片

技术要点

本实例主要是通过在JPanel类的子类中，重写JComponent类的paint()方法，并在该方法中使用Graphics类的copyArea()方法复制图像区域，实现最终的水波效果。

使用Graphics类的copyArea()方法，可以复制图像区域，该方法的定义如下：

```
public abstract void copyArea(int x, int y, int width,
int height, int dx, int dy)
```

参数说明

- x：源矩形的x 坐标。

- y：源矩形的y 坐标。
- width：源矩形的宽度。
- height：源矩形的高度。
- dx：复制像素的水平距离。
- dy：复制像素的垂直距离。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的WaterWavePictureFrame窗体类。

（3）在WaterWavePictureFrame窗体类中，创建内部面板类WaterWavePicturePanel，并重写JComponent类的paint()方法，在该方法中使用Graphics类的copyArea()方法复制图像区域，实现水波效果。

（4）将内部面板类WaterWavePicturePanel的实例添加到窗体类WaterWavePictureFrame的内容面板上，用于在窗体上显示水波效果的图片，代码如下：

```
class WaterWavePicturePanel extends JPanel {
    private Graphics
graphics;                                    //Graphics对象
    private Graphics
waveGraphics;                                //绘制水波的
Graphics对象
    private Image
oldImage;                                     //原图像对象
    private Image
waveImage;                                    //声明表示水波效
果的图像对象
```

```
    private int currentImage, imageWidth, imageHeight;
    private boolean
isImageLoaded;                                  //表示图片是否被
加载的标记
    public void paint(Graphics g) {
      drawWaterWave();
      if (waveImage != null) {
        g.drawImage(waveImage, -currentImage* imageWidth,0,
    this);    //绘制图像
      }
      g.clearRect(imageWidth,0, imageWidth * 4, imageHeight
  * 2);         //清除显示区域右侧的内容
    }
    public void drawWaterWave() {
      currentImage = 0;
      if (!isImageLoaded)
  {                                             //如果未加载图片
        graphics=getGraphics();
        //获得绘图上下文对象
        MediaTracker mediatracker=new
    MediaTracker(this);           //创建媒体跟踪对象
        URL
    imgUrl=WaterWavePictureFrame.class.getResource("/img/im
    age.jpg");                     //获取图片资源的路径
        oldImage=Toolkit.getDefaultToolkit().getImage(imgUr
    l);         //获取图像资源
```

```
        mediatracker.addImage(oldImage,0);
          //添加图片
        try {
          mediatracker.waitForAll();
            //加载图片
          isImageLoaded=
      !mediatracker.isErrorAny();                    //是否有错误
      发生
        } catch (InterruptedException ex) {
        }
        if (!isImageLoaded)
    {                                          //图片加载失败
          graphics.drawString("图片加载错
      误",10,40);                  //绘制错误信息
          return;
        }
        imageWidth=oldImage.getWidth(this);
        //得到图像宽度
        imageHeight=oldImage.getHeight(this);
          //得到图像高度
        createWave();
      //创建水波效果
      }
    }
    public void createWave() {
      Image img= createImage(imageWidth,
  imageHeight);                  //以图像宽度和高度创建图像对象
```

```
    Graphics g = null;
    if (img != null) {
      g=
  img.getGraphics();                                    //
  得到Image对象的Graphics对象
      g.drawImage(oldImage,0,0,
  this);                          //绘制原图像对象
      for (int i = 0; i < imageHeight; i++) {
        g.copyArea(0, imageHeight - 1 - i,
    imageWidth,1,0,-imageHeight+1+ (i *
    2));                      //复制图像区域
      }
    }
    waveImage= createImage(13 * imageWidth,
imageHeight);          //得到水波效果的图像对象
    if (waveImage != null) {
      waveGraphics=waveImage.getGraphics();
      //得到水波效果的绘图上下文对象
      waveGraphics.drawImage(img,12 * imageWidth,0,
  this);        //绘制图像
      int j = 0;
      while (j < 12) {
        simulateWaves(waveGraphics,
    j);                         //调用方法，模拟水波效果
        j++;
      }
    }
```

```
    }
    public void simulateWaves(Graphics g, int i)
{                          //水波效果模拟
      int j= (12 - i) *
  imageWidth;                                    //计算复制像素
  的水平距离
      int waveHeight= imageHeight /
  16;                                 //计算水波高度
      for (int h = 0; h < imageHeight; h++) {
        int k= (int) ((waveHeight * (h+28)
    *Math.sin(waveHeight* (imageHeight - h) / (h+1))) /
    imageHeight);           //计算复制像素的垂直距离
        if (h <-k) {
          g.copyArea(12* imageWidth,h, imageWidth,1, -
      j,0);       //复制图像区域，形成水波
        } else {
          g.copyArea(12 * imageWidth,h+k, imageWidth,1, -j,
      -k);     //复制图像区域，形成水波
        }
      }
    }
  }
```

举一反三

根据本实例，读者可以实现以下功能。

使用MediaTracker类对图像进行跟踪。

把水波效果图片恢复成原始图片。

实例292 局部图像放大

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\07\Ex07_292

实例说明

本实例使用Java的绘图技术，实现图像局部区域的放大。运行程序，拖动鼠标选择图像的局部区域，释放鼠标后，就可看到选择区域的图像被放大了，效果如图7.47所示。

图7.47 局部图像放大的效果

技术要点

本实例主要是通过Robot类的createScreenCapture()方法，使用BasicStroke类创建虚线对象，并结合Graphics类的drawRect()方法绘制选区选择图像，并使用drawImage()方法，将选择区域的图像放大，然后绘制到绘图上下文，从而实现图像局部放大的功能。

（1）使用BasicStroke类重载的构造方法创建笔画对象，并指定笔画为虚线模式。

public BasicStroke(float width, int cap, int join, float miterlimit, float[] dash, float dash_phase)

（2）使用Graphics类的drawRect()方法绘制矩形，以获得选择区域。

（3）使用Robot类的createScreenCapture()方法，用选择区域的图片创建缓冲图像对象，该方法的定义如下：

public BufferedImage createScreenCapture(Rectangle screenRect)

参数说明

- screenRect：屏幕上被截取的矩形区域。
- 返回值：从屏幕上截取的缓冲图像对象。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的PartZoomInImageFrame窗体类。

（3）在 PartZoomInImageFrame 窗体类中，创建内部面板类PartZoomInPanel，并重写JComponent类的paint()方法，在该方法中绘制原始图像、选择区域和局部放大的图像。

（4）将内部面板类PartZoomInPanel的实例添加到窗体类PartZoomInImageFrame的内容面板上，用于在窗体上显示原始图像、选择区域和局部放大的图像，代码如下：

```
class PartZoomInPanel extends JPanel
{                                  //创建绘制原图像的面板类
  public void paint(Graphics g) {
    Graphics2D g2 = (Graphics2D) g;
    g2.drawImage(img, 0, 0, this.getWidth(),
this.getHeight(), this); //绘制图像
    g2.setColor(Color.WHITE);
    if (flag) {
```

```
        float[] arr= {5.0f
    };                                  //创建虚线模式的数组
        BasicStroke stroke=new
    BasicStroke(1,BasicStroke.CAP_BUTT,BasicStroke.JOIN_BEV
    EL,1.0f, arr,0);          //创建宽度是1的平头虚线笔画对象
        g2.setStroke(stroke);
    //设置笔画对象
        g2.drawRect(pressPanelX,pressPanelY, releaseX -
    pressX,releaseY - pressY);                          //绘
    制矩形选区
      }
      if (mouseFlag)
{                                       //条件为真
        int zoomX=pressPanelX - (releaseX - pressX)
    /4;          //放大图像绘制点的x坐标
        int zoomY=pressPanelY - (releaseY - pressY) /
    4;          //放大图像绘制点的y坐标
        if (zoomX <= 0) {
          zoomX=0;                                  //坐标值
    小于等于0，让坐标值为0
        }
        if (zoomY <= 0) {
          zoomY=0;                                  //坐标值
    小于等于0，让坐标值为0
        }
        g.drawImage(buffImage, zoomX, zoomY,(int)
    ((releaseX - pressX) * 1.5f),(int) ((releaseY - pressY)
```

```
    * 1.5f), this);              //绘制放大后的局部图像
      }
    }
  }
```

举一反三

根据本实例，读者可以实现以下功能。

获得鼠标指针在控件和屏幕上的坐标。

局部图像缩小。

实例293 图片半透明特效

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\07\Ex07_293

实例说明

本实例使用Java的绘图技术，实现了图片的半透明效果。运行程序，窗体上显示不透明的图像，单击窗体上的“半透明”按钮，图片将显示为半透明效果，如图7.48所示。

图7.48 图片半透明的效果

技术要点

本实例主要是通过 Graphics2D类的 setComposite()方法，为绘图上下文指定表示透明度的AlphaComposite对象，从而实现了图片的半透明效果。

（1）使用 AlphaComposite 类获得表示透明度的 AlphaComposite 对象，该对象使用AlphaComposite 类的字段 SrcOver 调用 derive()方法，并指定透明度即可获得AlphaComposite对象。

（2）使用 Graphics2D 类的 setComposite()方法，为绘图上下文指定表示透明度的AlphaComposite对象。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的TranslucenceImageFrame窗体类。

（3）在TranslucenceImageFrame窗体类中，创建内部面板类 TranslucenceImagePanel，并重写JComponent类的paint()方法，在该方法中实现图像透明度的调整。

（4）将内部面板类 TranslucenceImagePanel 的实例添加到窗体类 TranslucenceImageFrame的内容面板上，用于在窗体上显示设置了透明度的图像，面板类 TranslucenceImagePanel 的代码如下：

```
class TranslucenceImagePanel extends JPanel {
  public void paint(Graphics g) {
    Graphics2D g2= (Graphics2D) g;                    //获得Graphics2D对象
    g2.clearRect(0, 0,  getWidth(), getHeight());          //清除绘图上下文的内容
    g2.setComposite(alpha);                                  //指定AlphaComposite对象
```

```
        g2.drawImage(img, 0, 0, getWidth(), getHeight(),
    this); //绘制图像
      }
    }
```

（5）“半透明”按钮用于实现图像的半透明效果，其事件代码如下：

```
    button.addActionListener(new ActionListener() {
      public void actionPerformed(final ActionEvent e) {
        alpha=AlphaComposite.SrcOver.derive(0.5f);          /
    /获得表示半透明的AlphaComposite对象
        translucencePanel.repaint();                        /
    /调用paint()方法
      }
    });
```

（6）“不透明”按钮用于实现图像的半透明效果，其事件代码如下：

```
    button_1.addActionListener(new ActionListener() {
      public void actionPerformed(final ActionEvent e) {
        alpha=AlphaComposite.SrcOver.derive(1.0f);          /
    /获得表示不透明的AlphaComposite对象
        translucencePanel.repaint();                        /
    /调用paint()方法
      }
    });
```

举一反三

根据本实例，读者可以实现以下功能。

实现图片水印。

实现混合特效。

实例294 图片融合特效

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\07\Ex07_294

实例说明

本实例使用Java的绘图技术，实现了两幅图片的融合效果。运行程序，窗体上将显示两幅图片的融合特效，改变窗体上滑块的位置，可对两幅图片的融合效果进行调整，效果如图7.49所示。

技术要点

本实例主要是通过 Graphics2D类的 setComposite()方法，为绘图上下文设置表示透明度的AlphaComposite对象，从而实现了两幅图片融合的效果。

（1）使用 AlphaComposite 类获得表示透明度的 AlphaComposite 对象，该对象使用AlphaComposite 类的字段 SrcOver 调用 derive()方法，并指定透明度即可获得AlphaComposite对象。

图7.49 图片融合特效

（2）使用 Graphics2D 类的 setComposite()方法，为绘图上下文设置表示透明度的AlphaComposite对象。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的PictureMixFrame窗体类。

（3）在 PictureMixFrame 窗体类中，创建内部面板类 PictureMixPanel，并重写 JComponent类的paint()方法，在该方法中实现图像的融合效果。

（4）将内部面板类PictureMixPanel的实例添加到窗体类PictureMixFrame的内容面板上，用于在窗体上显示融合效果的图像，面板类PictureMixPanel的代码如下：

```
class PictureMixPanel extends JPanel {
  boolean flag=
true;                                          //定义标记变量，
用于控制x的值
  AlphaComposite
alpha=AlphaComposite.SrcOver.derive(0.5f);      //获得表示
透明度的AlphaComposite对象
  public void paint(Graphics g) {
    Graphics2D g2= (Graphics2D)
g;                              //获得Graphics2D对象
    g2.drawImage(img1,0,0,  getWidth(),getHeight(),
this);      //绘制图像
    float value=
slider.getValue();                              //滑块组件的
取值
```

```
        float alphaValue=value /
    100;                                    //计算透明度的值
        alpha=AlphaComposite.SrcOver.derive(alphaValue);
          //获得表示透明度的AlphaComposite对象
        g2.setComposite(alpha);
    //指定AlphaComposite对象
        g.drawImage(img2,0,0,getWidth(),getHeight(),
    this);          //绘制调整透明度后的图片
      }
    }
```

（5）通过滑块控件，可以调整图片的融合效果，滑块控件的事件代码如下：

```
slider.addChangeListener(new ChangeListener() {
  public void stateChanged(final ChangeEvent e) {
    pictureMixPanel.repaint();
    //重新调用面板类的paint()方法
  }
});
```

举一反三

根据本实例，读者可以实现以下功能。

显示两幅图片的融合效果。

滑块控制透明度。

实例295 以椭圆形显示图像

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\07\Ex07_295

实例说明

在窗体上显示的图像通常都以矩形显示，本实例使用Java的绘图技术，实现以椭圆形显示图像。运行程序，效果如图7.50所示。

图7.50 以椭圆形显示图像

技术要点

本实例主要是通过图形区域的减运算，将矩形区域与椭圆形区域相减，并用运算结果覆盖图像，从而实现以椭圆形显示图像的功能。

（1）使用Area类的构造方法封装图形对象。

（2）使用Area类的subtract()方法对封装的图形对象进行减运算。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的DrawEllipseImageFrame窗体类。

（3）在 DrawEllipseImageFrame 窗体类中，创建内部面板类EllipseImagePanel，并重写JComponent类的paint()方法，在该方法中实现以椭圆形显示图像的功能。

（4）将内部面板类 EllipseImagePanel 的实例添加到窗体类DrawEllipseImageFrame 的内容面板上，用于在窗体上显示椭圆形图

像，代码如下：

```
class EllipseImagePanel extends JPanel {
  public void paint(Graphics g) {
    Graphics2D g2 = (Graphics2D)g;
    g2.drawImage(img,20,20,260,160,
this);                    //绘制图像
    Rectangle2D.Float rectangle = new
Rectangle2D.Float(0, 0, getWidth(),getHeight()); //创建矩
形对象
    Ellipse2D.Float ellipse = new Ellipse2D.Float(20, 20,
260, 160); //创建椭圆形对象
    Area area1=new
Area(rectangle);                          //创建区域矩形
    Area area2=new
Area(ellipse);                            //创建区域椭圆
    area1.subtract(area2);
  //两个区域形状进行减运算
    g2.setColor(getBackground());
  //设置绘图上下文的颜色为面板的背景颜色
    //绘制减运算后的区域形状g2.fill(area1);
  }
}
```

举一反三

根据本实例，读者可以实现以下功能。

以正方形形式显示图像。

以菱形形式显示图像。

实例296 图片百叶窗特效

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\07\Ex07_296

实例说明

本实例演示了如何利用Java的绘图技术实现图片百叶窗特效。运行程序，将在窗体上显示图片百叶窗效果，如图7.51所示。

图7.51 图片百叶窗特效

技术要点

本实例主要是通过在缓冲图像对象上绘制直线，并对缓冲图像对象进行模糊处理实现的，对图像进行模糊处理需要用到Kernel类和ConvolveOp类。

（1）Kernel 类定义了一个矩阵，用于描述指定的像素及其周围像素，如何影响过滤操作输出图像中像素位置的计算值，其构造方法的定义如下：

```
public Kernel(int width, int height, float[] data)
```

参数说明

- width：当前kernel 的宽度。
- height：当前kernel 的高度。

● data：以行优先顺序提供的kernel 数据。

（2）ConvolveOp类实现从源到目标的卷积，是一种通过输入像素来计算输出像素的空间运算，方法是将核与输入像素邻域相乘。这种运算使得直接邻域可按核数学指定的方式影响输出像素，其构造方法定义如下：

public ConvolveOp(Kernel kernel)

参数说明

kernel：指定的Kernel。

（3）ConvolveOp类提供了一个filter()方法，可以对缓冲图像进行过滤，实现对图像的特殊处理，如模糊、照亮边缘等，该方法的定义如下：

public final BufferedImage filter(BufferedImage src, BufferedImage dst)

参数说明

● src：要过滤的源BufferedImage。

● dst：已过滤的src 的目标BufferedImage 或为null。

● 返回值：对缓冲图像进行处理后的新BufferedImage 对象。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的ShutterFrame窗体类。

（3）在ShutterFrame窗体类中，创建内部面板类ShutterPanel，并重写JComponent类的paint()方法，在该方法中进行图片百叶窗效果的绘制。

（4）将内部面板类 ShutterPanel 的实例添加到窗体类 ShutterFrame 的内容面板上，用于在窗体上显示图片百叶窗。面板类ShutterPanel的代码如下：

```
class ShutterPanel extends JPanel {
```

```
    public void paint(Graphics g) {
      Graphics2D g2 = (Graphics2D) g;
      g2.drawImage(img,0,0, this);                          //
绘制图像对象
      int y=5;                                          //直线绘
制点的y坐标
      int space=10;                                      //下一条
直线的偏移量
      Line2D.Float line = null;
    image = new BufferedImage(getWidth() + 10,
getHeight(),BufferedImage.TYPE_INT_ARGB);          //创建缓冲
图像对象
      Graphics2D gs2d= (Graphics2D)
    image.getGraphics();    //获得缓冲图像对象的Graphics2D对
    象
      BasicStroke stroke=new
    BasicStroke(7);              //创建宽度是7的笔画对象
      gs2d.setStroke(stroke);                              //
    设置笔画对象
      gs2d.setColor(Color.WHITE);                        //指
    定颜色
      while (y <= getHeight()) {
        line=new Line2D.Float(0,y,getWidth(),y);        //
      创建直线对象
        gs2d.draw(line);                                 //在缓
      冲图像对象上绘制直线
```

```
        y=y+ space;                                    //计算下一
    条直线的y坐标
      }
      for (int i=0; i<3; i++) {                        //该
  for循环，实现3次模糊
        float[] elements=new float[9];                  //
    定义表示像素分量的数组
        for (int j = 0; j < 9; j++) {
          elements[j]=0.11f;                          //为数组
    赋值
        }
        convolve(elements);                            //调用
    方法，实现模糊功能
      }
      //绘制缓冲图像对象g2.drawImage(image,0,0, this);
    }
  }
```

在面板类ShutterPanel中使用的convolve()方法，用于实现模糊直线的功能，该方法是窗体类ShutterFrame中的成员方法，其代码如下：

```
  private void convolve(float[] elements) {
    Kernel kernel=new Kernel(3,3,
elements);                  //创建 Kernel对象
    ConvolveOp op=new ConvolveOp(kernel);              //创
建ConvolveOp对象
    if (image == null) {
      return;
```

```
        }
        image=op.filter(image,null);                                    /
/过滤缓冲图像对象
        repaint();                                                      //调
用paint()方法
    }
```

举一反三

根据本实例，读者可以实现以下功能。

制作图片渐变特效。

制作图片淡出特效。

实例297 图片马赛克特效

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\07\Ex07_297

实例说明

在进行马赛克处理后，可以使人物或图像变得模糊，对于不想公开的内容部分可以起到保护作用。运行程序，效果如图7.52所示，单击窗体上的“添加马赛克”按钮，将为图片添加马赛克，效果如图7.53所示。

图7.52 原图片效果

图7.53 添加马赛克后的效果

技术要点

本实例主要是通过在缓冲图像对象上绘制渐变的矩形，并对缓冲图像对象进行模糊处理和透明度调整实现的。对图像进行模糊处理需要用到Kernel类和ConvolveOp类。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的MosaicFrame窗体类。

（3）在MosaicFrame窗体类中，创建内部面板类MosaicPanel，并重写JComponent类的paint()方法，在该方法中实现在图像上绘制缓冲图像，完成马赛克效果的绘制。

（4）将内部面板类MosaicPanel的实例添加到窗体类MosaicFrame的内容面板上，用于在窗体上显示图像和在图像上绘制马赛克。面板类MosaicPanel的代码如下：

```
class MosaicPanel extends JPanel {
  public void paint(Graphics g) {
    Graphics2D g2 = (Graphics2D) g;
    g2.drawImage(img,0,0,getWidth(),getHeight(),
this);        //绘制图像对象
    g2.drawImage(image,0,0,
this);                          //绘制缓冲图像对象
  }
}
```

（5）窗体类MosaicFrame上的“添加马赛克”按钮，用于实现为图像添加马赛克的功能。“添加马赛克”按钮的事件代码如下：

```
button.addActionListener(new ActionListener() {
  public void actionPerformed(final ActionEvent e) {
    int x=104;                                          //
矩形绘制点的x坐标
    int y=60;                                          //矩
形绘制点的y坐标
    Rectangle2D.Float rect = null;
    image=new
BufferedImage(getWidth()+10,getHeight(),BufferedImage.TYP
E_INT_ARGB);                //创建缓冲图像对象
    Graphics2D gs2d= (Graphics2D)
image.getGraphics();        //获得缓冲图像对象的
Graphics2D对象
```

```
    AlphaComposite alpha =
AlphaComposite.SrcOver.derive(0.90f);//获得表示透明度的
AlphaComposite对象
    gs2d.setComposite(alpha);
  //设置透明度
    GradientPaint paint = null;
    for (int i = 0; i < 4; i++) {
      for (int j = 0; j < 3; j++) {
        paint=new
    GradientPaint(x,y,Color.white,x+10,y+10,Color.gray,tr
    ue);                    //创建循环渐变的GradientPaint对象
        gs2d.setPaint(paint);                              //
    设置渐变
        rect=new
    Rectangle2D.Float(x,y,20,20);          //创建矩形对象
        gs2d.fill(rect);                                   //
    在缓冲图像对象上绘制矩形
        y=y+20;                                        //计算下
    一个矩形的y坐标
      }
      y=60;                                             //还原
  y坐标
      x=x+20;                                          //计算x
  坐标
    }
    for (int i=0; i<3; i++)
{                                  //该for循环，实现3次模糊
```

```
        float[] elements=new
    float[9];                          //定义表示像素分量的数组
        for (int j = 0; j < 9; j++) {
          elements[j]=0.11f;                                    //
      为数组赋值
        }
        convolve(elements);                                      //
      调用方法，实现模糊功能
      }
      //调用paint()方法mosaicPanel.repaint();
    }
  });
```

（6）在“添加马赛克”按钮的事件中使用了 convolve()方法，用于实现模糊马赛克矩形的功能，该方法是窗体类MosaicFrame中的成员方法，其代码如下：

```
  private void convolve(float[] elements) {
    Kernel kernel=new Kernel(3,3,
elements);                        //创建 Kernel对象
    ConvolveOp op=new
ConvolveOp(kernel);                          //创建ConvolveOp对象
    if (image == null) {
      return;
    }
    image=op.filter(image,null);
    //过滤缓冲图像对象
  }
```

举一反三

根据本实例，读者可以开发以下程序。

为图片添加马赛克。

去除马赛克特效。

实例298 模糊

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\07\Ex07_298

实例说明

本实例使用 Java 绘制技术实现了图片的模糊效果。运行程序，效果如图 7.54 所示，单击窗体上的“模糊”按钮，将对图片进行模糊处理，效果如图7.55所示。

图7.54 图片模糊前的效果

图7.55 图片模糊后的效果

技术要点

本实例主要是通过在缓冲图像对象上绘制图片，并对缓冲图像对象进行模糊处理实现的，对图像进行模糊处理需要用到Kernel类和ConvolveOp类。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的BlurImageFrame窗体类。

（3）在BlurImageFrame窗体类中，创建内部面板类BlurImagePanel，在该类中实现创建缓冲图像对象，并将其绘制到绘图上下文对象上。

（4）将内部面板类BlurImagePanel的实例添加到窗体类BlurImageFrame的内容面板上，用于在窗体上显示原图片和进行模糊处理后的图片。面板类BlurImagePanel的代码如下：

```
class BlurImagePanel extends JPanel {
  public BlurImagePanel(){
    Image
img=null;                                                    //
声明创建图像对象
```

```
    try {
      img= ImageIO.read(new
  File("src/img/imag.jpg"));                    //创建图像
  对象
    } catch (IOException e) {
      e.printStackTrace();
    }
    image = new
BufferedImage(img.getWidth(null),img.getHeight(null),Buff
eredImage.TYPE_INT_RGB);
    //创建缓冲图像对象
    image.getGraphics().drawImage(img,0,0,
null);                          //在缓冲图像对象上绘制图像
  }
  public void paint(Graphics g) {
    if (image != null) {
      g.drawImage(image,0,0,null);
            //绘制缓冲图像对象
    }
  }
}
```

（5）窗体类BlurImageFrame上的“模糊”按钮，用于实现模糊图像的功能，其事件代码如下：

```
button.addActionListener(new ActionListener() {
  public void actionPerformed(final ActionEvent e) {
    float[] elements=new
float[9];                                  //定义表示像素分量
```

的数组

```
    for (int i = 0; i < 9; i++) {
      elements[i]=0.11f;
      //为数组赋值
    }
    convolve(elements);
    //调用方法，实现模糊功能
  }
});
```

（6）在“模糊”按钮的事件中使用了 convolve()方法，用于实现模糊图像的功能，该方法是窗体类BlurImageFrame的成员方法，其代码如下：

```
private void convolve(float[] elements) {
  Kernel kernel=new Kernel(3,3,
elements);                              //创建 Kernel对象
  ConvolveOp op=new
ConvolveOp(kernel);                          //创建ConvolveOp
对象
  if (image == null) {
    return;
  }
  image=op.filter(image,null);
        //过滤缓冲图像对象
  repaint();
    //调用paint()方法
}
```

举一反三

根据本实例，读者可以开发以下程序。

模糊图像的部分区域。

模糊整个图像。

实例299 锐化

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\07\Ex07_299

实例说明

本实例使用 Java 绘制技术实现图片的锐化效果。运行程序，效果如图 7.56 所示，单击窗体上的“锐化”按钮，将对图片进行锐化处理，效果如图7.57所示。

图7.56 图片锐化前的效果

图7.57 图片锐化后的效果

技术要点

本实例主要是通过在缓冲图像对象上绘制图片，并对缓冲图像对象进行锐化处理实现的，对图像进行锐化处理需要用到Kernel类和ConvolveOp类。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的SharpenImageFrame窗体类。

（3）在SharpenImageFrame窗体类中，创建内部面板类SharpenImagePanel，在该类中实现创建缓冲图像对象，并将其绘制到绘图上下文对象上。

（4）将内部面板类SharpenImagePanel的实例添加到窗体类SharpenImageFrame的内容面板上，用于在窗体上显示原图片和进行锐化处理后的图片。面板类SharpenImagePanel的代码如下：

```
class SharpenImagePanel extends JPanel {
  public SharpenImagePanel(){
    Image
img=null;                                                    //
```

```
声明创建图像对象
    try {
      img= ImageIO.read(new
  File("src/img/imag.jpg"));                    //创建图像
  对象
    } catch (IOException e) {
      e.printStackTrace();
    }
    image = new
BufferedImage(img.getWidth(null),img.getHeight(null),Buff
eredImage.TYPE_INT_RGB);
    //创建缓冲图像对象
    image.getGraphics().drawImage(img,0,0,
null);                          //在缓冲图像对象上绘制图像
  }
  public void paint(Graphics g) {
    if (image != null) {
      g.drawImage(image,0,0,null);
            //绘制缓冲图像对象
    }
  }
}
```

（5）窗体类SharpenImageFrame上的“锐化”按钮，用于实现锐化图像的功能，其事件代码如下：

```
button.addActionListener(new ActionListener() {
  public void actionPerformed(final ActionEvent e) {
```

```
        float[] elements=
    {0.0f,-1.0f,0.0f,-1.0f,5.0f,-1.0f,0.0f,-1.0f,0.0f};
          //声明表示像素分量的数组
        convolve(elements);
              //调用方法实现图片锐化功能
      }
    });
```

（6）在“锐化”按钮的事件中使用了 convolve()方法，用于实现锐化图像的功能，该方法是窗体类SharpenImageFrame的成员方法，其代码如下：

```
    private void convolve(float[] elements) {
      Kernel kernel=new Kernel(3,3,
  elements);                                    //创建 Kernel对象
      ConvolveOp op=new
  ConvolveOp(kernel);                                  //创建
  ConvolveOp对象
      if (image == null) {
        return;
      }
      image=op.filter(image,null);
                //过滤缓冲图像对象
      repaint();
              //调用paint()方法
    }
```

举一反三

根据本实例，读者可以开发以下程序。

为图像添加蜕化效果。

去除蜕化效果还原图像。

实例300 照亮边缘

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\07\Ex07_300

实例说明

本实例使用 Java 绘制技术实现图片的照亮边缘效果。运行程序，效果如图 7.58 所示，单击窗体上的“照亮边缘”按钮，将对图片进行照亮边缘处理，效果如图7.59所示。

图7.58 图片照亮边缘前的效果

图7.59 图片照亮边缘后的效果

技术要点

本实例主要是在缓冲图像对象上绘制图片，并对缓冲图像对象进行照亮边缘处理实现的，对图像进行照亮边缘处理需要用到Kernel类和ConvolveOp类。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的EdgeDetectImageFrame窗体类。

（3）在EdgeDetectImageFrame窗体类中，创建内部面板类EdgeDetectImagePanel，在该类中创建缓冲图像对象，并将其绘制到绘图上下文对象上。

（4）将内部面板类EdgeDetectImagePanel的实例添加到窗体类EdgeDetectImageFrame的内容面板上，用于在窗体上显示原图片和进行照亮边缘处理后的图片。面板类EdgeDetectImagePanel的代码如下：

```
class EdgeDetectImagePanel extends JPanel {
  public EdgeDetectImagePanel(){
    Image
img=null;                                                    //
声明创建图像对象
    try {
      img= ImageIO.read(new
  File("src/img/image.jpg"));                    //创建图像对
  象
    } catch (IOException e) {
      e.printStackTrace();
    }
```

```
    image = new
BufferedImage(img.getWidth(null),img.getHeight(null),Buff
eredImage.TYPE_INT_RGB);
    //创建缓冲图像对象
    image.getGraphics().drawImage(img,0,0,
null);                          //在缓冲图像对象上绘制图像
  }
  public void paint(Graphics g) {
    if (image != null) {
      g.drawImage(image,0,0,null);
            //绘制缓冲图像对象
    }
  }
}
```

（5）窗体类EdgeDetectImageFrame上的“照亮边缘”按钮，用于实现图片的照亮边缘处理，其事件代码如下：

```
button.addActionListener(new ActionListener() {
  public void actionPerformed(final ActionEvent e) {
    float[] elements=
{0.0f,-1.0f,0.0f,-1.0f,4.0f,-1.0f,0.0f,-1.0f,0.0f};
      //声明表示像素分量的数组
    convolve(elements);
          //调用方法实现图片锐化功能
  }
});
```

（6）在“照亮边缘”按钮的事件中使用 convolve()方法，实现照亮图片边缘的功能，该方法是窗体类SharpenImageFrame的成员方

法，其代码如下：

```
    private void convolve(float[] elements) {
      Kernel kernel=new Kernel(3,3,
  elements);                                    //创建 Kernel对象
      ConvolveOp op=new
  ConvolveOp(kernel);                                  //创建
  ConvolveOp对象
      if (image == null) {
        return;
      }
      image=op.filter(image,null);
                //过滤缓冲图像对象
      repaint();
              //调用paint()方法
    }
```

举一反三

根据本实例，读者可以开发以下程序。

突出显示图像的边缘。

实例301 反向

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\07\Ex07_301

实例说明

本实例使用 Java 绘制技术实现图片的反向处理。运行程序，效果如图 7.60 所示，单击窗体上的“反向”按钮，将对图片进行反向处理，效果如图7.61所示。

图7.60 图片反向处理前的效果

图7.61 图片反向处理后的效果

技术要点

本实例主要是通过在缓冲图像对象上绘制图片，并对缓冲图像对象进行反向处理实现的，对图像进行反向处理需要用到ShortLookupTable类和LookupOp类。

（1）ShortLookupTable 类定义了一个查找表对象，该对象查找操作的输出被解释为一个无符号short量，查找表包含图像的一个或多个band的short型数据数组，并包含对数组建立索引前从输入值中减掉的偏移量，因此数组应小于为约束输入提供的本地数据的大小。如果查找表中只有一个数组，则将该数组应用于所有band，该类的构造方法定义如下：

public ShortLookupTable(int offset, short[] data)

参数说明

- offset：在为数组建立索引前从输入值中减掉的值。
- data：short 型数据数组。

（2）LookupOp类实现从源到目标的查找操作，其构造方法定义如下：

public LookupOp(LookupTable lookup, RenderingHints hints)

参数说明

- lookup：指定的LookupTable 对象。
- hints：指定的RenderingHints 对象，或者为null。

（3）LookupOp类提供了一个filter()方法，可以对缓冲图像执行查找操作，该方法的定义如下：

public final BufferedImage filter(BufferedImage src, BufferedImage dst)

参数说明

- src：要过滤的源BufferedImage。
- dst：存储过滤操作结果的BufferedImage。
- 返回值：过滤后的BufferedImage。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的NegativeImageFrame窗体类。

（3）在NegativeImageFrame窗体类中，创建内部面板类NegativeImagePanel，在该类中创建缓冲图像对象，并将其绘制到绘图上下文对象上。

（4）将内部面板类NegativeImagePanel的实例添加到窗体类NegativeImageFrame的内容面板上，用于在窗体上显示原图片和进行

反向处理后的图片，面板类NegativeImagePanel的代码如下：

```
class NegativeImagePanel extends JPanel {
  public NegativeImagePanel(){
    Image
img=null;                                                    //
声明创建图像对象
    try {
      img= ImageIO.read(new
  File("src/img/imag.jpg"));                    //创建图像
  对象
    } catch (IOException e) {
      e.printStackTrace();
    }
    image = new
BufferedImage(img.getWidth(null),img.getHeight(null),Buff
eredImage.TYPE_INT_RGB);
    //创建缓冲图像对象
    image.getGraphics().drawImage(img,0,0,
null);                          //在缓冲图像对象上绘制图像
  }
  public void paint(Graphics g) {
    if (image != null) {
      g.drawImage(image,0,0,null);
            //绘制缓冲图像对象
    }
  }
}
```

（5）窗体类NegativeImageFrame上的“反向”按钮，用于实现图片的反向处理，其事件代码如下：

```
button.addActionListener(new ActionListener() {
  public void actionPerformed(final ActionEvent e) {
    short[]negative=new
short[256*1];                    //创建表示颜色反向的分量数组
    for (int i = 0; i<256;i++){
      negative[i]= (short)(255-
  i);                            //为数组赋值
    }
    ShortLookupTable table = new
ShortLookupTable(0,negative); //创建查找表对象
    LookupOp op=new LookupOp(table,null);          //创建实现从源到目标查找操作的LookupOp对象
    image=op.filter(image,null);                    //调用LookupOp对象的filter()方法，实现图像反向功能
    repaint();                                      //调用paint()方法
  }
});
```

举一反三

根据本实例，读者可以实现以下功能。

取消图像的反向效果。

实例302 光栅图像

这是一个可以启发思维的实例

实例位置：光盘\mingrisoft\07\Ex07_302

实例说明

本实例演示如何利用Java的绘图技术，实现光栅图像的绘制。运行程序，将在窗体上显示绘制的光栅图像，效果如图7.62所示。

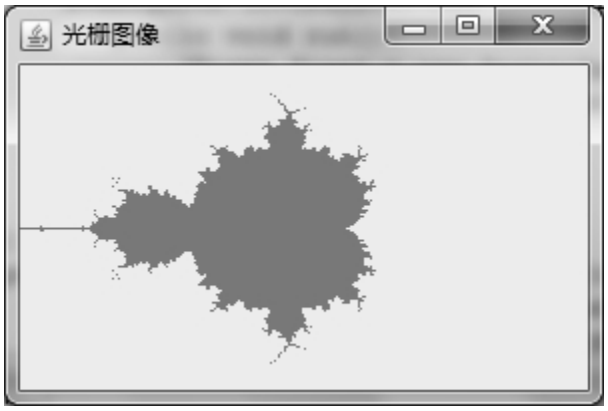


图7.62 光栅图像

技术要点

本实例的实现主要是通过已有数字序列的数学公式，计算数字序列在指定的点(a,b)上是收敛的还是发散的，如果是收敛的，就绘制光栅上的点；如果是发散的，则不进行光栅的绘制。

公式x * x - y * y + a 和2 * x * y + b 分别用于计算点(a,b)在x 轴和y 轴上的坐标值，如果x或者y小于等于2，说明该点是收敛的；否则，说明该点是发散的。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的RasterImageFrame窗体类。

（3）在RasterImageFrame窗体类中，创建isOrNotConvergence()方法，用于判断指定点是收敛的还是发散的，该方法的代码如下：

```
    private boolean isOrNotConvergence(double a,doubleb)
{  //判断数字序列上的点(a,b)是收敛的还是发散的
      double x=0.0D;                                    //如果x大于
  2，数字序列就是发散的
      double y=0.0D;                                    //如果y大于
  2，数字序列也是发散的
      int iterations=0;                                  //循环变
  量
      while (x <= 2 && y <= 2 && iterations < MAX_ITERATIONS)
  {
        double xNew=x * x - y* y+ a;                    //计算每个
    点的x值
        double yNew=2 * x * y+b;                      //计算每个点
    的y值
        x=xNew;                                       //赋值给变量x，
    用于判断是收敛还是发散
        y=yNew;                                       //赋值给变量y，
    用于判断是收敛还是发散
        iterations++;                                  //调整迭代
    器变量的值
      }
      return x>2 ||y>2;                                 //返回
  false表示收敛，则进行绘制；为true表示发散，则透明
    }
```

（4）在窗体类RasterImageFrame中，再创建makeRasterImage()方法，用于获得绘制有光栅的缓冲图像，该方法的代码如下：

```
    private BufferedImage makeRasterImage(int width, int
height) {
        BufferedImage image=new
    BufferedImage(width,height,BufferedImage.TYPE_INT_ARGB);
            //创建缓冲图像对象
        WritableRaster raster= image.getRaster();              //
    获得提供像素写入功能的WritableRaster对象
        ColorModel model= image.getColorModel();          //获得
    缓冲图像的颜色模型
        Color fractalColor=Color.RED;                        //定义表
    示红色的颜色对象
        int argb= fractalColor.getRGB();                       //获
    得表示颜色的RGB值
        Object colorData = model.getDataElements(argb, null);
    //返回ColorModel中指定像素的数组表示形式
        for (int i = 0; i < width; i++) {
          for (int j = 0; j < height; j++) {
            //计算点(i,j)是否导致与点(a,b)上的像素收敛
            double a = XMIN + i * (XMAX - XMIN) / width;
            double b = YMIN + j * (YMAX - YMIN) / height;
            if (!isOrNotConvergence(a, b)) { //如果点(i,j)导致
        与点(a,b)上的像素收敛，escapesToInfinity()方法返回
        false，则进行光栅绘制
              raster.setDataElements(i,j,colorData); //为类型
          TransferType基本数组中的单个像素设置数据
            }
          }
```

```
        }
        return image;                                    //返回绘制
    有光栅图像的缓冲图像对象
    }
```

举一反三

根据本实例，读者可以开发以下程序。

将光栅图像保存为图片。

显示绘制的光栅图像。

实例303 图片倒影效果

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\07\Ex07_303

实例说明

本实例演示如何使用Java绘图技术，实现图片倒影效果。运行程序，效果如图7.63所示。

图7.63 图片倒影效果

技术要点

本实例主要是通过在JPanel类的子类中，重写JComponent类的paint()方法，并在该方法中使用Graphics类的copyArea()方法复制图像区域，实现最终的图片水波倒影效果。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的PictureInvertedFrame窗体类。

（3）在 PictureInvertedFrame 窗体类中，创建内部面板类PictureInvertedPanel，并重写JComponent类的paint()方法，在该方法中使用Graphics类的copyArea()方法复制图像区域，实现图片水波倒影效果。

（4）将内部面板类PictureInvertedPanel的实例添加到窗体类PictureInvertedFrame的内容面板上，用于在窗体上显示图片的水波倒影效果，面板类PictureInvertedPanel的代码如下：

```
  class PictureInvertedPanel extends JPanel {
    private Graphics
graphics;                                          //Graphi
cs对象
    private Graphics
waveGraphics;                                      //绘制水
波的Graphics对象
    private Image
oldImage;                                           //原图
像对象
    private Image
waveImage;                                          //声明
表示水波效果的图像对象
    private int currentImage, imageWidth, imageHeight;
    private boolean
imageLoaded;                                        //表示
图片是否被加载的标记
    public void paint(Graphics g) {
      drawWaterWave();
      if (waveImage != null) {
        g.drawImage(waveImage, -currentImage* imageWidth,
    imageHeight, this);      //绘制图像
      }
```

```
    g.drawImage(oldImage,0,1,
this);                                          //绘制原图片
    g.clearRect(imageWidth,0, imageWidth * 4,
imageHeight*2);                    //清除显示区域右侧的内容
  }
  public void drawWaterWave() {
    currentImage = 0;
    if (!imageLoaded)
{                                                    //如果
未加载图片
      graphics=getGraphics();
                //获得绘图上下文对象
      MediaTracker mediatracker=new
  MediaTracker(this);                    //创建媒体跟踪对
  象
      URL imgUrl =
  PictureInvertedFrame.class.getResource("/img/image.jpg"
  );                                    //获取图片资源的路径
      oldImage=Toolkit.getDefaultToolkit().getImage(imgUr
  l);                    //获取图像资源
      mediatracker.addImage(oldImage,0);
                //添加图片
      try {
        mediatracker.waitForAll();
                  //加载图片
        imageLoaded=
    !mediatracker.isErrorAny();                          //
```

```
    是否有错误发生
      } catch (InterruptedException ex) {
      }
      if (!imageLoaded)
  {                                                    //图片加
  载失败
        graphics.drawString("图片加载错
    误",10,40);                          //绘制错误信息
        return;
      }
      imageWidth=oldImage.getWidth(this);
                //得到图像宽度
      imageHeight=oldImage.getHeight(this);
                  //得到图像高度
      createWave();
              //创建水波效果
    }
  }
  public void createWave() {
    Image img= createImage(imageWidth,
imageHeight);                        //以图像宽度和高度创建图
像对象
    Graphics g = null;
    if (img != null) {
      g=
  img.getGraphics();
    //得到Image对象的Graphics对象
```

```
        g.drawImage(oldImage,0,0,
    this);                                  //绘制原图像对象
        for (int i = 0; i < imageHeight; i++) {
          g.copyArea(0, imageHeight - 1 - i,
      imageWidth,1,0,-imageHeight+1+ (i *
      2));                              //复制图像区域
        }
      }
      waveImage= createImage(13 * imageWidth,
  imageHeight);                   //得到水波效果的图像对象
      if (waveImage != null) {
        waveGraphics=waveImage.getGraphics();
              //得到水波效果的绘图上下文对象
        waveGraphics.drawImage(img,12 * imageWidth,0,
    this);              //绘制图像
        int j = 0;
        while (j < 12) {
          simulateWaves(waveGraphics,
      j);                                 //调用方法，模拟水波效
      果
          j++;
        }
      }
    }
    public void simulateWaves(Graphics g, int i)
{                                 //水波效果模拟
```

```
    int j= (12 - i) *
imageWidth;                                          //计算复
制像素的水平距离
    int waveHeight= imageHeight /
16;                                  //计算水波高度
    for (int l = 0; l < imageHeight; l++) {
      int k= (int) ((waveHeight * (l+28)
  *Math.sin(waveHeight* (imageHeight - l) / (l+1))) /
  imageHeight);                  //计算复制像素的垂直距离
      if (l <-k){
        g.copyArea(12* imageWidth, l, imageWidth,1, -
    j,0);              //复制图像区域，形成水波
      } else {
        g.copyArea(12* imageWidth, l+k, imageWidth,1, -j,
    -k);         //复制图像区域，形成水波
      }
    }
  }
}
```

举一反三

根据本实例，读者可以开发以下程序。

实现动态的图片倒映效果。

实现反向的倒映图片。

7.8 其他

实例304 图片浏览器

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\07\Ex07_304

实例说明

本实例实现了图片浏览器的功能。运行程序，单击“选择”按钮，在弹出的“打开文件”对话框中选择一幅图片，将在窗体上显示该图片，如图7.64所示，选择图片后，就可以通过窗体下方的按钮，浏览该图片所在文件夹中所有相同格式的图片。

图7.64 图片浏览器

技术要点

本实例主要是通过File类的list()方法，将指定文件夹中符合格式要求的图片放到List集合列表中，然后通过索引值对List集合列表中的文件名进行浏览，从而实现图片浏览器的功能。

（1）使用File类的list()方法，可以对指定目录下满足条件的记录进行过滤，返回一个符合条件的所有文件的字符串数组，该方法的定义如下：

```
public String[] list(FilenameFilter filter)
```

参数说明

- filter：文件名过滤器，用于过滤符合条件的文件名。
- 返回值：字符串数组，是符合过滤条件的所有文件名。

（2）将过滤出来的符合条件的文件名添加到List集合列表中，代码如下：

```
for (int i=0;i<fileNames.length;i++)
{                                    //遍历数组中的文件名
  if (fileNames[i].equals(currentFileName))
  {                          //判断是否为当前文件
    currentFileIndex=
i;                                          //记忆当前文件的索引
值
  }
  fileNameList.add(fileNames[i]);
  //将文件添加到集合列表中
}
```

（3）通过索引值对List集合列表中的文件名进行浏览，实现浏览图片功能的代码如下：

```
imgFile = new
File(filePath+"/"+fileNameList.get(currentFileIndex).toString
```

());
//创建当前索引值对应图片的File对象
try{
img=
ImageIO.read(imgFile); //创建
当前图片的图像对象
imgPanel.repaint();
//调用paint()方法，显示图片
} catch (IOException e1) {
e1.printStackTrace();
}
实现过程
（1）新建一个项目。
（2）在项目中创建一个继承JFrame类的BrowerPictureFrame窗体类。
（3）在BrowerPictureFrame窗体类中，创建内部面板类DrawImagePanel，用于在窗体上显示当前图片。
（4）在BrowerPictureFrame窗体类中，为“上一张”按钮添加事件代码，实现向上浏览图片的功能。其事件代码如下：
```
button_2.addActionListener(new ActionListener() {
  public void actionPerformed(final ActionEvent e) {
    currentFileIndex-
-; //调整当前图片
的索引值
    if (currentFileIndex < 0) {
      currentFileIndex= fileNameList.size() -
1; //当前图片索引为最后一张图片的索
```

```
    引
      }
      imgFile = new
    File(filePath+"/"+fileNameList.get(currentFileIndex).toSt
    ring());
      //创建当前索引值对应图片的File对象
      try {
        img=
    ImageIO.read(imgFile);                              //创
    建当前图片的图像对象
        imgPanel.repaint();
        //调用paint()方法，显示图片
      } catch (IOException e1) {
        e1.printStackTrace();
      }
    }
});
```

（5）在BrowerPictureFrame窗体类中，为“下一张”按钮添加事件代码，实现向下浏览图片的功能。其事件代码如下：

```
button_3.addActionListener(new ActionListener() {
  public void actionPerformed(final ActionEvent e) {
    currentFileIndex++;
    //调整当前图片的索引值
    if (currentFileIndex > fileNameList.size() - 1) {
      currentFileIndex=0;
    //当前图片索引为第一张图片的索引
    }
```

```
        imgFile = new
    File(filePath+"/"+fileNameList.get(currentFileIndex).toSt
    ring());
        //创建当前索引值对应图片的File对象
        try {
          img=
      ImageIO.read(imgFile);                              //创
      建当前图片的图像对象
          imgPanel.repaint();
          //调用paint()方法
        } catch (IOException e1) {
          e1.printStackTrace();
        }
      }
    });
```

举一反三

根据本实例，读者可以开发以下程序。

制作旋转图片的浏览器。

制作滚动图片的浏览器。

实例305 转换图片格式

这是一个可以提高分析能力的实例

实例位置：光盘\mingrisoft\07\Ex07_305

实例说明

本实例实现了图片格式的转换。运行程序，单击“选择图片”按钮选择一幅图片，效果如图7.65所示，然后在窗体下方的组合框中，

选择图片转换后的格式，单击“保存”按钮完成图片格式的转换。

图7.65 转换图片格式

技术要点

本实例使用ImageIO类的write()方法，将绘制有原图片的BufferedImage对象，以指定格式存储到磁盘上，实现了转换图片格式的功能。

（1）创建原图片的BufferedImage对象，实现代码如下：

```
buffImage=
ImageIO.read(imgFile);                                    //构造BufferedImage对象
```

（2）以指定格式将原图片的BufferedImage对象存储到磁盘上，实现代码如下：

```
ImageIO.write(buffImage, extName,
file);                                    //将缓冲图像保存到磁盘
```

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的ConversionPictureFormatFrame窗体类，在窗体类中定义一个内部面板类DrawImagePanel，用于在窗体上显示选择的原图片，代码如下：

```
class DrawImagePanel extends JPanel {
  public void paint(Graphics g) {
    g.clearRect(0, 0, getWidth(), getHeight());
    g.drawImage(buffImage,0,0,
this);                          //绘制指定的图片
  }
}
```

（3）在项目中创建一个继承JFrame类的ConversionPictureFormatFrame窗体类，并完成窗体的设计，然后在“保存”按钮的事件中，完成图片格式的转换，其事件代码如下：

```
try {
  String extName=
(String)comboBox.getSelectedItem();          //新格式的扩展名，不含点
  String pathAndName = pathAndFileName.substring(0,
pathAndFileName.lastIndexOf(".") + 1)+extName;
  //转换后的图片完整路径和文件名
  File file=new
File(pathAndName);                          //创建转换后图片的File对象
  ImageIO.write(buffImage, extName,
file);                        //将缓冲图像保存到磁盘
  File oldFile=new
File(pathAndFileName);                      //原图片的File对
```

象

```
    oldFile.delete();
  //删除原图片文件
    JOptionPane.showMessageDialog(null, "文件格式更改成功！");    //显示提示信息
  } catch (IOException e1) {
    JOptionPane.showMessageDialog(null, "保存失败\n"+e1.getMessage());    //显示提示信息
  }
```

举一反三

根据本实例，读者可以开发以下程序。

实现jpg图片转换为png图片。

实现gif图片转换为bmp图片。

实例306 绘制石英钟

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\07\Ex07_306

实例说明

本实例绘制了一个石英钟，在石英钟上显示有时针、分针、秒针、日期和星期。运行程序，效果如图7.66所示。

图7.66 石英钟

技术要点

本实例使用Calendar类获得日历对象，进而获得毫秒值、秒值、分值和小时值，然后根据这几个值计算出秒针、分针和时针的角度，最终计算出秒针、分针和时针指向点的坐标，从而实现了本实例的功能。

（1）通过Calendar日历对象，获得毫秒值、秒值、分值和小时值，代码如下：

```
    Calendar
calendar=Calendar.getInstance();                                    /
/获取日历对象
    int millisecond=
calendar.get(MILLISECOND);                                  //获取毫
秒值
    int sec=
calendar.get(SECOND);                                       //获
取秒值
    int minutes=
calendar.get(MINUTE);                                       //获取
分值
    int hours=
calendar.get(HOUR);                                         //获取
小时值
```

（2）根据毫秒值、秒值、分值和小时值，计算出秒针、分针和时针的角度，代码如下：

```
    double secAngle= (60 - sec) * 6 - (millisecond /
150);                       //秒针角度
    int minutesAngle= (60 -minutes) *
6;                                  //分针角度
```

```
    int hoursAngle= (12 - hours) * 360 / 12 - (minutes /
2);                          //时针角度
```

（3）根据秒针、分针和时针的角度，计算出秒针、分针和时针指向点的坐标，代码如下：

```
    int secX= (int) (secLen
*Math.sin(Math.toRadians(secAngle)));                  //秒针指向
点的x坐标
    int secY= (int) (secLen
*Math.cos(Math.toRadians(secAngle)));                  //秒针指向
点的y坐标
    int minutesX= (int) (minuesLen
*Math.sin(Math.toRadians(minutesAngle)));    //分针指向点的x
坐标
    int minutesY= (int) (minuesLen
*Math.cos(Math.toRadians(minutesAngle)));    //分针指向点的y
坐标
    int hoursX= (int) (hoursLen
*Math.sin(Math.toRadians(hoursAngle)));        //时针指向点的
x坐标
    int hoursY= (int) (hoursLen
*Math.cos(Math.toRadians(hoursAngle)));        //时针指向点的
y坐标
```

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JPanel类的ClockPanel面板类，重写paint()方法实现石英钟的绘制，paint()方法的关键代码如下：

```
    //分别绘制时针、分针、秒针
```

```
    g2.setStroke(HOURS_POINT_WIDTH);
        //设置时针的宽度
    g2.setColor(Color.BLACK);
        //设置时针的颜色
    g2.drawLine(centerX, centerY, centerX - hoursX, centerY -
hoursY);              //绘制时针
    g2.setStroke(MINUETES_POINT_WIDTH);
      //设置分针的宽度
    if (minutesAngle
!=hoursAngle)                                          //分针、时针
重叠变色
      g2.setColor(new
  Color(0x2F2F2F));                                    //设置未重叠
  时的颜色
    else {
      g2.setColor(Color.GREEN);
        //设置重叠时的颜色
    }
    g2.drawLine(centerX, centerY, centerX -minutesX, centerY
-minutesY);         //绘制分针
    g2.setStroke(SEC_POINT_WIDTH);
        //设置秒针的宽度
    if (secAngle !=hoursAngle && secAngle
!=minutesAngle)                    //分针、时针、秒针重叠变色
      g2.setColor(Color.ORANGE);
          //设置未重叠时的颜色
    else {
```

```
        g2.setColor(Color.GREEN);
          //设置重叠时的颜色
    }
    //绘制3个指针的中心圆和秒针
    g2.fillOval(centerX -5, centerY
-5,10,10);                                    //绘制中心圆
    g2.drawLine(centerX, centerY, centerX - secX, centerY -
secY);                  //绘制秒针
    g2.drawLine(centerX+1, centerY+1, centerX - secX+1,
centerY - secY+1);     //绘制秒针
```

（3）创建继承JDialog类的ClockFrame对话框窗体类，在该类的构造方法中取消窗体装饰，使其不显示标题栏和边框，并使其总在最前显示，代码如下：

```
    setUndecorated(true);
        //取消窗体修饰
    setAlwaysOnTop(true);
        //窗体置顶
```

举一反三

根据本实例，读者可以实现以下功能。

照亮石英钟的边缘实现夜光效果。

调整石英钟时间。

实例307 画图程序

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\07\Ex07_307

实例说明

本实例使用Java的绘图技术，实现一个画图程序。运行程序，可以通过菜单和工具栏两种方式，选择背景色、前景色、画笔的粗细，还可以使用橡皮进行擦除，也可以清除整个画板的内容，图7.67所示是使用本实例绘制的图片。

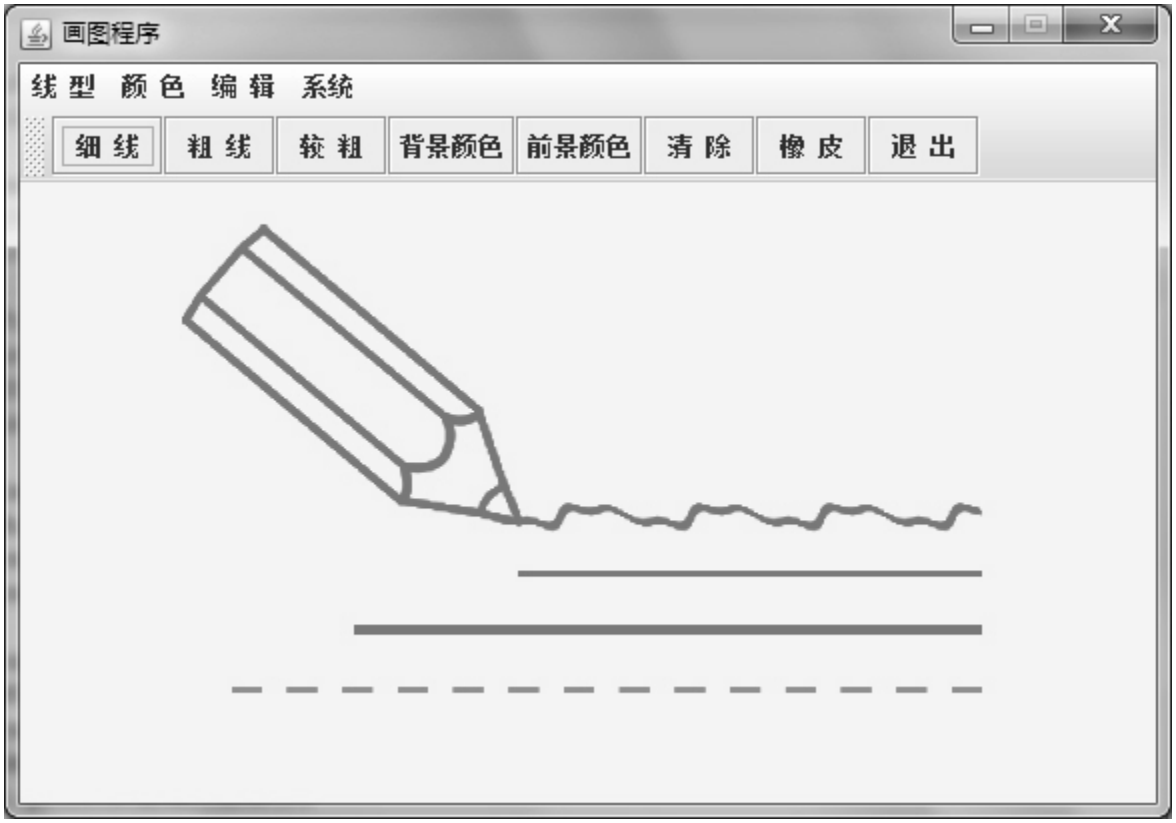


图7.67 画图程序

技术要点

本实例通过BufferedImage对象进行画图，然后在自定义的Canvas对象上显示绘制的内容，从而实现了画图功能。

（1）在BufferedImage对象上画图，是通过鼠标的拖动事件实现的，其中的关键代码如下：

```
if (x > 0 && y > 0) {
  g.drawLine(x, y, e.getX(),
e.getY());                                    //在鼠标经过处画直线
```

```
    }
    x=
e.getX();
//上一次鼠标绘制点的横坐标
    y=
e.getY();
//上一次鼠标绘制点的纵坐标
```

（2）创建Canvas类的子类DrawPictureCanvas，用于显示BufferedImage对象上绘制的内容，其中的关键代码如下：

```
    public void setImage(Image image)
{                          //传递BufferedImage对象
      this.image= image;                                    //为
  成员变量赋值
    }
    public void paint(Graphics g)
{                                //重写paint()方法，在画布上绘制
BufferedImage对象
      g.drawImage(image,0,0,null);                          //
  在画布上绘制图像
    }
```

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承Canvas类的DrawPictureCanvas画布类，用于在其上绘制缓冲图像对象，该缓冲图像对象就是用户绘制的内容，从而实现显示所绘制内容的功能，该类的关键代码如下：

```
    public class DrawPictureCanvas extends Canvas {
```

```
    private Image
image=null;                                          //定义Image
对象的引用
    public void setImage(Image image) {
      this.image=
  image;                                              //为成员变
  量赋值
    }
    public void paint(Graphics g) {
      g.drawImage(image,0,0,null);
        //在画布上绘制图像
    }
  }
```

（3）在项目中创建一个继承 Canvas 类的 DrawPictureFrame 窗体类，用于显示完成绘图的各种操作，其中实现绘图的关键代码，是画布类DrawPictureCanvas的实例canvas的3个鼠标事件，即拖动、移动和释放。

☑拖动事件的代码

```
  public void mouseDragged(final MouseEvent e) {
    if (rubber)
{                                                     //橡皮标识
为true，表示使用橡皮
      if (x > 0 && y > 0) {
        g.setColor(backgroundColor);
          //用背景色设置绘图上下文对象的颜色
```

```
            g.fillRect(x,y,10,10);
              //擦除鼠标指针经过位置的图像
          }
          x=
    e.getX();                                          //获
    得鼠标指针在画布上的横坐标
          y=
    e.getY();                                          //获
    得鼠标指针在画布上的纵坐标
      } else
{                                                      //橡皮
标识为false，表示画图
          if (x > 0 && y > 0) {
            g.drawLine(x, y, e.getX(),
      e.getY());                          //在鼠标指针经过处画
      直线
          }
          x=
    e.getX();                                          //上
    一次鼠标绘制点的横坐标
          y=
    e.getY();                                          //上
    一次鼠标绘制点的纵坐标
      }
      canvas.repaint();
          //更新画布
    }
```

☑移动事件的代码

```
public void mouseMoved(final MouseEvent arg0) {
  if (rubber) {
    setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURS
OR));      //设置鼠标指针的形状
  } else {
    setCursor(Cursor.getPredefinedCursor(Cursor.CROSSHAIR
_CURSOR));//设置鼠标指针的形状
  }
}
```

☑释放事件的代码

```
public void mouseReleased(final MouseEvent arg0) {
  x=
-1;                                                    //上
一次鼠标绘制点的横坐标
  y=
-1;                                                    //上
一次鼠标绘制点的纵坐标
}
```

举一反三

根据本实例，读者可以实现以下功能。

增加刷子工具。

增加几何图形。

实例308 屏幕抓图程序

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\07\Ex07_308

实例说明

本实例实现了屏幕抓图功能。运行程序，可以用鼠标选择屏幕上需要抓取的内容，并以zzkkee.jpg 保存到C 盘根目录。抓取Windows 7 桌面上时钟和日历的效果，如图7.68 所示，如果需要退出程序，只需要在屏幕上单击鼠标右键即可。

图7.68 抓取屏幕上时钟和日历的效果

技术要点

本程序使用窗体透明技术把窗体设置为透明，然后使用Robot类的createScreenCapture()方法捕获屏幕，实现屏幕抓图的功能。

（1）程序首先取消窗体装饰，然后使用 AWTUtilities 类的setWindowOpacity()方法，将窗体的透明度设置为0.01f，即使窗体接近完全透明，可以透出桌面，因此可以响应鼠标事件，并且可以抓取桌面图片。取消窗体装饰和设置窗体透明度的代码如下：

```
setUndecorated(true);
//取消窗体修饰
AWTUtilities.setWindowOpacity(this,0.01f);
    //设置窗体透明
```

（2）使用Robot类的createScreenCapture()方法，可以根据需要捕获屏幕上的指定区域。捕获屏幕指定区域的代码如下：

```
    buffImage=
robot.createScreenCapture(rect);                    //获得
缓冲图像对象
```

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的CaptureScreenImageFrame窗体类，在构造方法中添加取消窗体装饰、设置窗体透明度和捕获桌面图片的功能，主要代码如下：

```
    setUndecorated(true);
    //取消窗体修饰
    AWTUtilities.setWindowOpacity(this,0.01f);
        //设置窗体透明
    getContentPane().add(partZoomInPanel,
BorderLayout.CENTER);
    robot=new
Robot();                                    //创建Robot对
象
    rect=new
Rectangle(0,0,dim.width,dim.height);                //创建
Rectangle对象
    buffImage=
robot.createScreenCapture(rect);                    //获得
缓冲图像对象
```

（3）在CaptureScreenImageFrame窗体类的鼠标释放事件中，完成抓取屏幕图像并保存到磁盘的操作。鼠标释放事件的主要代码如下：

```
    releaseX= e.getXOnScreen() -
1;                          //鼠标释放点在屏幕上的x坐标减1，
即去除选择线
    releaseY= e.getYOnScreen() -
1;                          //鼠标释放点在屏幕上的y坐标减1，
即去除选择线
    try {
      if (releaseX - pressX > 0 && releaseY - pressY > 0) {
        Rectangle rect=new Rectangle(pressX,pressY, releaseX-
    pressX, releaseY - pressY);                    //创建
    Rectangle对象
        buffImage=
    robot.createScreenCapture(rect);               //获得缓
    冲图像对象
        FileOutputStream out=new
    FileOutputStream("c:/zzkkee.jpg");  //保存位置的输出流对
    象
        ImageIO.write(buffImage,
    "jpg",out);                    //写入磁盘
        out.flush();
        out.close();
      }
    } catch (FileNotFoundException e2) {
      e2.printStackTrace();
    } catch (IOException e3) {
      e3.printStackTrace();
    }
```

举一反三

根据本实例，读者可以实现以下功能。

保存抓取的屏幕图像。

美化抓取的图像。

实例309 屏幕放大镜

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\07\Ex07_309

实例说明

本实例实现了一个屏幕放大镜程序。运行程序，通过鼠标选择屏幕上需要放大的内容，即可放大所选择的内容。例如选择Windows 7桌面上的日历，将放大显示该日历，效果如图7.69所示，如果需要退出程序，只需要在屏幕上单击鼠标右键即可。

图7.69 屏幕上的日历放大之前和之后的效果

技术要点

本实例通过Robot类捕获屏幕图像，然后使用绘图上下文的drawImage()方法，对在屏幕上捕获的图像进行放大，并绘制到窗体上。

（1）使用Robot类的createScreenCapture()方法，可以根据需要捕获屏幕上的指定区域，捕获屏幕指定区域的代码如下：

Rectangle rect = new Rectangle(pressX, pressY, releaseX - pressX, releaseY - pressY); //创建Rectangle对象

●oomBuffImage= robot.createScreenCapture(rect); //获得缓冲图像对象

（2）使用绘图上下文的drawImage()方法，对在屏幕上捕获的图像进行放大，并绘制到窗体上，代码如下：

//绘制放大后的内容，即在水平和垂直方向分别放大1.5倍

g2.drawImage(zoomBuffImage, zoomX, zoomY, (int) ((releaseX - pressX) * 1.5f), (int) ((releaseY - pressY) * 1.5f), this);

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的ScreenZoomInFrame窗体类，在该类的成员声明区定义如下变量，用于标识是否显示选择区域的矩形，以及是否放大选择的图像，代码如下：

private boolean flag= false; //声明标记变量，为true时显示选择区域的矩形，否则不显示

private boolean mouseFlag= false; //放大图像的标记变量，为true时进行放大，否则不放大

（3）在ScreenZoomInFrame窗体类中，创建一个内部面板类PartZoomInPanal，用于在屏幕上绘制放大后的图像，该内部类的代码如下：

```
class PartZoomInPanel extends JPanel
{                          //创建面板类
  public void paint(Graphics g) {
    Graphics2D g2 = (Graphics2D) g;
```

```
    g2.drawImage(buffImage,0,0, this);                    //
绘制图像
    g2.setColor(Color.BLACK);
    if (flag) {
      float[] arr= {5.0f };                               //创建
  虚线模式的数组
      BasicStroke stroke=new
  BasicStroke(1,BasicStroke.CAP_BUTT,BasicStroke.JOIN_BEV
  EL,1.0f, arr,0);     //创建宽度是1的平头虚线笔画对象
      g2.setStroke(stroke);                               //设置
  笔画对象
      g2.drawRect(pressPanelX,pressPanelY, releaseX -
  pressX,releaseY - pressY);                    //绘制矩形
  选区
    }
    if (mouseFlag) {                                      //条
件为真
      int zoomX = pressPanelX - (releaseX - pressX) / 4;
  //放大内容绘制点的x坐标
      int zoomY = pressPanelY - (releaseY - pressY) / 4;
  //放大内容绘制点的y坐标
      if (zoomX <= 0) {
        zoomX=0;                                          //坐标值小于
  等于0，让坐标值为0
      }
      if (zoomY <= 0) {
```

```
            zoomY=0;                                    //坐标值小于
        等于0，让坐标值为0
          }
            g2.drawImage(zoomBuffImage, zoomX, zoomY,(int)
       ((releaseX - pressX) * 1.5f),(int) ((releaseY - pressY)
       * 1.5f), this); //绘制放大后的内容
          }
       }
    }
```

（4）在窗体类ScreenZoomInFrame的鼠标释放事件中，捕获在屏幕上选择的图像，该图像就是在内部面板中放大的图像。鼠标释放事件的代码如下：

```
    public void mouseReleased(final MouseEvent e)
{               //鼠标释放事件
      releaseX= e.getXOnScreen() - 1;                          //
  鼠标释放点在屏幕上的x坐标减1，即去除选择线
      releaseY= e.getYOnScreen() - 1;                          //
  鼠标释放点在屏幕上的y坐标减1，即去除选择线
      if (releaseX - pressX > 0 && releaseY - pressY > 0) {
        Rectangle rect=new Rectangle(pressX,pressY, releaseX-
  pressX, releaseY - pressY);                  //创建
  Rectangle对象
        zoomBuffImage=
  robot.createScreenCapture(rect);       //获得缓冲图像对象
      }
      flag= false;                                             //为
  标记变量赋值为false
```

```
    mouseFlag= true;                                    //标
记为true，进行放大
    partZoomInPanel.repaint();                          /
/调用paint()方法，实现放大
  }
```

举一反三

根据本实例，读者可以实现以下功能。

恢复放大后的图片。

实现图片的缩小功能。

第8章 动画

文字动画
图片动画

8.1 文字动画

实例310 文字淡入淡出

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\08\Ex08_310

实例说明

本实例演示如何利用Java的绘图技术，实现文字淡入淡出特效的绘制。运行程序，将在窗体上以淡入淡出的效果显示文字，如图8.1所示。

图8.1 文字淡入淡出特效

技术要点

本实例主要是通过 Graphics2D类的 setComposite()方法，为绘图上下文指定表示透明度的AlphaComposite对象，以及使用从Graphics类继承的drawString()方法绘制文本实现的。

（1）使用 AlphaComposite 类获得表示透明度的AlphaComposite 对象，该对象使用AlphaComposite类的字段SrcOver

调用derive()方法获得，该方法的定义如下：

public AlphaComposite derive(float alpha)

参数说明

● alpha：闭区间0.0f 到1.0f 之间的一个浮点数字，为0.0f 时完全透明，为1.0f 时不透明。

● 返回值：表示透明度的AlphaComposite 对象。

（2）使用 Graphics2D 类的 setComposite()方法，为绘图上下文指定表示透明度的AlphaComposite对象，该方法的定义如下：

public abstract void setComposite(Composite comp)

参数说明

comp：表示透明度的AlphaComposite对象。

（3）使用Graphics类的drawString()方法绘制文本，该方法的定义如下：

public abstract void drawString(String str, int x, int y)

参数说明

● str：绘制的文本内容。

● x：绘制点的x 坐标。

● y：绘制点的y 坐标。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的TextFadeFrame窗体类。

（3）在TextFadeFrame窗体类中创建内部面板类TextFadePanel，该面板类实现了Runnable接口，并重写JComponent类的paint()方法和实现Runnable接口的run()方法，在paint()方法中使用Graphics类的setFont()、setColor()、setComposite()和drawString()方法，完成设置字体、颜色、透明度和绘制文本的操作；在run()方法中实现改变透明度值的操作。

（4）将内部面板类TextFadePanel的实例添加到窗体类TextFadeFrame的内容面板上，用于在窗体上显示淡入淡出效果的文字。内部面板类TextFadePanel的代码如下：

```
class TextFadePanel extends JPanel implements Runnable {
  boolean flag= true;                                    //定义标记变量，用于控制x的值
  float x=0.0f;                                          //定义表示透明度的变量x
  AlphaComposite alpha=AlphaComposite.SrcOver.derive(x);  //获得表示透明度的AlphaComposite对象
  public void paint(Graphics g) {
    Graphics2D g2= (Graphics2D) g;                       //获得Graphics2D对象
    g2.drawImage(img,0,0,  getWidth(),getHeight(), this);  //绘制图像
    Font font=new Font("华文楷体",Font.BOLD,60);          //创建字体对象
    g2.setFont(font);                                    //指定字体
    g2.setColor(Color.RED);                              //指定颜色
    g2.setComposite(alpha);                              //指定AlphaComposite对象
    g2.drawString("编程词典",30,120);                     //绘制文本
  }
```

```
public void run() {
  while (true) {
    if (flag) {                                    // flag
为true时
      x-=0.1f;                                    // x进行减
  0.1计算
      if (x<=0.0f) {                               //x小于等
  于0.0f时
        x=0.0f;                                  // x等于0.0f
        flag= false;                              //为flag赋
    值为false
      }
    } else {                                       // flag为
false时
      x+=0.1f;                                    // x进行加
  0.1计算
      if (x>=1.0f) {                               //x大于等
  于1.0f时
        x=1.0f;                                  // x等于1.0f
        flag= true;                               //为flag赋值
    为true
      }
    }
    alpha=AlphaComposite.SrcOver.derive(x);         //重
新获得表示透明度的AlphaComposite对象
    repaint();                                      //调用
paint()方法
```

```
            try {
              Thread.sleep(150);                    //休眠150
        毫秒
            } catch (InterruptedException e) {
              e.printStackTrace();
            }
          }
        }
      }
```

举一反三

根据本实例，读者可以开发以下程序。

实现文字的淡入淡出。

实现百叶窗的文字效果。

实例311 文字缩放

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\08\Ex08_311

实例说明

本实例演示如何利用Java的绘图技术，实现文字缩放特效的绘制。运行程序，将在窗体上显示缩放的文字，效果如图8.2和图8.3所示。

技术要点

本实例主要是使用从Graphics类继承的setFont()方法改变字体的大小，以及使用drawString()方法绘制文本实现的。

图8.2 文字缩小后的效果

图8.3 文字放大后的效果

（1）使用Font类可以创建字体对象，其中包括字体名称、字形和字体大小，然后使用Graphics类的setFont()方法为给图上下文设置字体，Font类的构造方法的定义如下：

public Font(String name, int style, int size)

参数说明

- name：字体的名称。
- style：字体的字形，也就是字体的样式。
- size：字体的大小。

（2）使用Graphics类的drawString()方法绘制文本。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的TextZoomFrame窗体类。

（3）在TextZoomFrame窗体类中创建内部面板类TextZoomPanel，该面板类实现了Runnable接口，并重写JComponent类的paint()方法和实现Runnable接口的run()方法。在paint()方法中使用Graphics类的setFont()、setColor()和drawString()方法，完成设置字体、字形、字体大小、颜色和绘制文本的操作；在run()方法中实现改变字体大小的操作。

（4）将内部面板类TextZoomPanel的实例添加到窗体类TextZoomFrame的内容面板上，用于在窗体上显示缩放效果的文字。内部面板类TextZoomPanel的代码如下：

```
class TextZoomPanel extends JPanel implements Runnable {
  boolean flag= false;                                    //
定义标记变量，用于控制x的值
   int x=12;                                              //定义
表示字体大小的变量x
   Font font=new Font("华文楷体",Font.BOLD,x);            //
创建字体对象
   public void paint(Graphics g) {
     Graphics2D g2= (Graphics2D) g;                       //获
得Graphics2D对象
      g2.drawImage(img,0,0,getWidth(),getHeight(),
this);    //绘制图像
      g2.setFont(font);                                   //指
定字体
      g2.setColor(Color.WHITE);                           //指
定颜色
```

```
  //绘制文本g2.drawString("编程词典",30,120);
}
public void run() {
  while (true) {
    if (flag) {                                  // flag
为true时
      x-=1;                                     // x进行减1
  计算
      if (x<=12) {                               //x小于等于
  12时
        x=12;                                   // x等于12
        flag= false;                             //为flag赋
    值为false
      }
    } else {                                     // flag为
false时
      x+=1;                                     // x进行加1
  计算
      if (x>=72) {                               //x大于等于
  72时
        x=72;                                   // x等于72
        flag= true;                              //为flag赋值
    为true
      }
    }
    font=new Font("华文楷体",Font.BOLD,x);       //重新
创建字体对象
```

```
            repaint();                                    //调用
        paint()方法
            try {
              Thread.sleep(50);                           //休眠50
          毫秒
            } catch (InterruptedException e) {
              e.printStackTrace();
            }
          }
        }
      }
```

举一反三

根据本实例，读者可以实现以下功能。

对放大后的文字进行还原。

对缩小后的文字进行还原。

实例312 文字跑马灯

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\08\Ex08_312

实例说明

本实例演示如何利用Java的绘图技术和多线程技术，实现文字跑马灯特效。运行程序，将在窗体上以跑马灯效果显示文字信息，效果如图8.4所示。

图8.4 文字跑马灯特效

技术要点

本实例将作为跑马灯的文字转换为字符数组，然后通过另一个 int 型数组来存储字符数组中每个字符的x坐标值，并在线程的run()方法中有规律地调整数组中这些x坐标值，从而实现文字跑马灯的特效。

（1）使用 String 类的 toCharArray()方法，可以将字符串转换为字符数组。toCharArray()方法的定义如下：

```
public char[] toCharArray()
```

参数说明

返回值：一个新分配的字符数组，其长度是此字符串的长度，其内容被初始化为包含此字符串标识的字符序列。

（2）使用 int 型数组来存储跑马灯文字数组中每个字符的 x 位置坐标，当第一次执行线程时，使x坐标值等比递增，以后再执行线程时，则使 x坐标值等差递增，从而实现了文字跑马灯效果。调整x坐标值的代码如下：

```
if (!flag){
  x[i]=x[i]+20*i;          // x坐标进行等比递增
} else {
```

```
        x[i]=x[i]+20;          // x坐标进行等差递增
    }
```

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的HorseRaceLightTextFrame窗体类。

（3）在HorseRaceLightTextFrame窗体类中创建内部面板类HorseRaceLightTextPanel，该面板类实现了Runnable接口，并重写JComponent类的paint()方法和实现Runnable接口的run()方法。在paint()方法中完成跑马灯文字的绘制；在run()方法中实现改变跑马灯文字的x坐标值。

（4）将内部面板类HorseRaceLightTextPanel的实例添加到窗体类HorseRaceLightTextFrame的内容面板上，用于在窗体上显示跑马灯效果的文字。内部面板类HorseRaceLightTextPanel的代码如下：

```
    class HorseRaceLightTextPanel extends JPanel implements Runnable {    //创建内部面板类
        String value= "全能编程词典，我的学习专家。";                //存储绘制的内容
        char[]drawChar=value.toCharArray();
    //将绘制内容转换为字符数组
        int[]x=new int[drawChar.length];                                //存储每个字符绘制点x坐标的数组
        int y=100;                                                      //存储绘制点的y坐标
        public void paint(Graphics g) {
```

```
    g.clearRect(0, 0, getWidth(),
getHeight());                    //清除绘图上下文的内容
    g.drawImage(img,0,0,getWidth(),getHeight(),
this);          //绘制图像
    Font font=new Font("华文楷
体",Font.BOLD,20);          //创建字体对象
    g.setFont(font);
  //指定字体
    g.setColor(Color.RED);
  //指定颜色
    for (int j = drawChar.length-1; j >=0; j--){
      g.drawString(drawChar[drawChar.length-1-j]+"",x[j]
  ,y);    //绘制文本
    }
  }
  public void run() {
    try {
      boolean flag= false;              //为false时表示第
  一次执行，x坐标进行等比递增，否则进行等差递增
      while (true) {                          //读取内容
        Thread.sleep(300);                    //当前线程休
  眠300毫秒
        for (int i = drawChar.length-1; i >=0 ; i--) {
          if (!flag){
            x[i]=x[i]+20*i;              // x坐标进行等比
        递增
          } else {
```

```
                x[i]=x[i]+20;                    // x坐标进行等差递
            增
              }
              if (x[i]>=360 - 20){                    //大于窗体宽
          度减20的值时
                x[i]=0;                        // x坐标为0
              }
            }
            repaint();                              //调用paint()方
        法
            if (!flag) {
              flag= true;                           //赋值为true
            }
          }
        } catch (Exception e) {
          e.printStackTrace();
        }
      }
    }
```

举一反三

根据本实例，读者可以实现以下功能。

在屏幕上显示公司的联系电话和地址。

控制滚动速度。

实例313 字幕显示

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\08\Ex08_313

实例说明

本实例演示如何利用Java的绘图技术，实现字幕显示特效的绘制。运行程序，文字会从窗体的底部向上移动，移动一小段距离后消失，然后其他文字又从窗体底部向上移动，效果如图8.5和图8.6所示。

图8.5 字幕显示特效1

图8.6 字幕显示特效2

技术要点

本实例通过Java绘图技术绘制文本，通过多线程技术改变文本绘制点的y坐标，从而实现字幕显示的特效。

（1）使用Graphics类的drawString()方法完成文本的绘制。

（2）通过实现Runnable接口实现多线程，并在run()方法中改变文本绘制点的y坐标值，代码如下：

```
if (y<=216 - 50) {                                    //如果已经向上移动50像素
    y=216;                                            // y坐标定位到最下方
    if (value.equals("明日编程词典网址")) {
      value= "http://www.mrbccd.com";                 //改变绘制的内容
    } else {
      value= "明日编程词典网址";                      //改变绘制的内容
    }
} else {                                              //如果还没向上移动到50像素
    y -=2;                                            // y坐标上移
}
```

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的CaptionSpecificFrame窗体类。

（3）在CaptionSpecificFrame窗体类中创建内部面板类CaptionSpecificPanel，该面板类实现了Runnable接口，并重写JComponent类的paint()方法和实现Runnable接口的run()方法。在paint()方法中完成字幕文字的绘制；在run()方法中实现改变字幕文字的y坐标值。

（4）将内部面板类CaptionSpecificPanel的实例添加到窗体类CaptionSpecificFrame的内容面板上，用于在窗体上显示字幕效果的文字。内部面板类CaptionSpecificPanel的代码如下：

```
    class CaptionSpecificPanel extends JPanel implements
Runnable {
      int x=50;                                          //存储
  绘制点的x坐标
      int y=216;                                        //存储
  绘制点的y坐标
      String value= "明日编程词典网址";                    //
  存储绘制的内容
      public void paint(Graphics g) {
        g.clearRect(0,0,316,237);                        //清
    除绘图上下文的内容
        g.drawImage(img,0,0,getWidth(),getHeight(),
    this);    //绘制图像
        Font font=new Font("华文楷体",Font.BOLD,20);      //
    创建字体对象
        g.setFont(font);                                 //指
    定字体
        g.setColor(Color.RED);                           //指
    定颜色
        g.drawString(value,x,y);                         //绘
    制文本
      }
      public void run() {
        try {
```

```
        while (true) {                                    //读取
    内容
            Thread.sleep(100);                            //当前线
        程休眠1秒
            if (y<=216 - 50) {                            //如果已
        经向上移动50像素
                y=216;                                 // y坐标定位到
            最下方
                if (value.equals("明日编程词典网址")) {
                    value= "http://www.mrbccd.com";    //改变绘制
                的内容
                } else {
                    value= "明日编程词典网址";       //改变绘制的
                内容
                }
            } else {                                      //如果还没向上
        移动到50像素
                y -=2;                                 // y坐标上移
            }
            repaint();                                    //调用paint()方
        法
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}
```

举一反三

根据本实例，读者可以开发以下程序。

实现从左向右移动的字幕特效。

实现文字在原位显示后消失的特效。

实例314 文字闪现

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\08\Ex08_314

实例说明

本实例使用Java的绘图技术和多线程技术，实现文字闪现的特效。运行程序，窗体上会出现闪现的文字，即窗体上的文字在很短的时间内，交替消失和出现，如图8.7和图8.8所示。

图8.7 无闪现文字的效果

图8.8 有闪现文字的效果

技术要点

本实例通过Java绘图技术绘制文本，通过多线程技术控制文字是否出现，从而实现了文字闪现的特效。

（1）使用Graphics类的drawString()方法，完成文本的绘制。

（2）通过实现 Runnable 接口实现多线程，并在 run()方法中通过定义标记变量，来控制文字的闪现，当需要显示闪现文字时，将文字赋值给变量；当不需要显示闪现文字时，为变量赋值为空字符串。run()方法中，通过标记变量控制闪现文字的代码如下：

```
if (flag) {                                    // flag为true
  flag= false;                                 //赋值为false
  value="明日编程词典";                          //为value赋值
} else {
  flag= true;                                  //赋值为true
  value="";                                    //赋值为空字符串
}
```

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的TextFlashFrame窗体类。

（3）在TextFlashFrame窗体类中创建内部面板类TextFlashPanel，该面板类实现了Runnable接口，并重写JComponent类的paint()方法和实现Runnable接口的run()方法。在paint()方法中完成闪现文字的绘制；在run()方法中控制是否显示闪现文字。

（4）将内部面板类TextFlashPanel的实例添加到窗体类TextFlashFrame的内容面板上，用于在窗体上显示文字闪现特效。内部面板类TextFlashPanel的代码如下：

```
class TextFlashPanel extends JPanel implements Runnable {
  boolean flag= true;                                    //标记变量
  String value= "";                                      //存放绘制内容的变量
  public void paint(Graphics g) {
    g.clearRect(0,0,310,230);                            //清除绘图上下文的内容
    g.drawImage(img,0,0,getWidth(),getHeight(),this);
  //绘制图像
    Font font=new Font("华文楷体",Font.BOLD,42);         //创建字体对象
    g.setFont(font);                                     //指定字体
    g.setColor(Color.RED);                               //指定颜色
    g.drawString(value,10,110);                          //绘制文本
  }
  public void run() {
```

```
    try {
      while (true) {                                    //读取
  内容
        Thread.sleep(150);                              //当前线
      程休眠150毫秒
        if (flag) {                                      // flag
      为true
          flag= false;                                  //赋值为
        false
          value="明日编程词典";                          //为value
        赋值
        } else {
          flag= true;                                   //赋值为true
          value="";                                     //赋值为空字
        符串
        }
        repaint();                                      //调用
      paint()方法
      }
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

举一反三

根据本实例，读者可以实现以下功能。

实现不同文字的交替显示。

延长交替显示的时间。

实例315 滚动广告字幕

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\08\Ex08_315

实例说明

滚动广告字幕是指文字从窗体的一侧向另一侧移动，当所有文字都从一侧移动到窗体以外区域时，再次重复上述移动过程。运行程序，将在窗体上显示滚动广告字幕特效，效果如图8.9所示。

图8.9 滚动广告字幕

技术要点

本实例通过Java绘图技术绘制文本，并通过多线程技术控制广告字幕的滚动显示，从而实现滚动广告字幕特效。

（1）使用Graphics类的drawString()方法，完成文本的绘制。

（2）通过实现Runnable接口实现多线程，并在run()方法中改变文本绘制点的x坐标值。run()方法中，调整x坐标值的代码如下：

```
if (x<= -400) {                                    //该条件可以根据需要自行调整
```

```
    x=316;                                          // x坐标定位
到最右侧
  } else {
    x -=2;                                          // x坐标左
移
  }
```

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的RollAdvertisementFrame窗体类。

（3）在 RollAdvertisementFrame 窗体类中创建内部面板类RollAdvertisementPanel，该面板类实现了Runnable接口，并重写了JComponent类的paint()方法，同时也实现了Runnable接口的run()方法。在paint()方法中完成广告字幕的绘制；在run()方法中控制广告字幕的滚动。

（4）将内部面板类 RollAdvertisementPanel 的实例添加到窗体类 RollAdvertisementFrame的内容面板上，用于在窗体上显示滚动的广告字幕。内部面板类RollAdvertisementPanel的代码如下：

```
  class RollAdvertisementPanel extends JPanel implements
Runnable {
    int
x=316;                                          //存储绘
制点的x坐标
    int
y=190;                                          //存储绘
制点的y坐标
```

```
    String value= "明日编程词典网址：
http://www.mrbccd.com";       //存储绘制的内容
    public void paint(Graphics g) {
      g.clearRect(0,0,316,237);
  //清除绘图上下文的内容
      g.drawImage(img,0,0,getWidth(),getHeight(),
  this);           //绘制图像
      Font font=new Font("华文楷
  体",Font.BOLD,20);           //创建字体对象
      g.setFont(font);
    //指定字体
      g.setColor(Color.RED);
    //指定颜色
      g.drawString(value,x,y);
    //绘制文本
    }
    public void run() {
      try {
        while (true) {                                    //
    读取内容
          Thread.sleep(50);                               //当
      前线程休眠1秒
          if (x<= -400) {                                 //该
      条件可以根据需要自行调整
            x=316;                                    // x坐标
        定位到最右侧
          } else {
```

```
            x -=2;                                          // x坐标
        左移
          }
          repaint();                                        //调
      用paint()方法
        }
      } catch (Exception e) {
        e.printStackTrace();
      }
    }
}
```

举一反三

根据本实例，读者可以实现以下功能。

制作从左向右移动的滚动广告字幕。

制作从右向左移动的滚动广告字幕。

8.2 图片动画

实例316 图片淡入淡出

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\08\Ex08_316

实例说明

本实例演示如何利用Java的绘图技术，实现图片淡入淡出特效的绘制。运行程序，将在窗体上以淡入淡出的效果显示图片，效果如图8.10和图8.11所示。

图8.10 只有背景图片的效果

图8.11 图片淡入淡出的效果

技术要点

本实例主要是通过 Graphics2D类的 setComposite()方法，为绘图上下文指定表示透明度的AlphaComposite对象，以及使用从Graphics类继承的drawImage()方法绘制图像实现的。

（1）使用Composite接口的实现类AlphaComposite，获得表示透明度的AlphaComposite对象，该对象可以使用AlphaComposite类的字段SrcOver，调用derive()方法获得。

（2）使用 Graphics2D 类的 setComposite()方法，为绘图上下文指定表示透明度的AlphaComposite对象。

（3）使用Graphics类的drawImage()方法绘制图片，该方法的定义如下：

public abstract boolean drawImage(Image img, int x, int y, int width, int height, ImageObserver observer)

参数说明

- img：要绘制的图像。
- x：起始点的x 坐标。
- y：起始点的y 坐标。
- width：绘制的宽度。

● height：绘制的高度。

● observer：图像观察者。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的PictureFadeFrame窗体类。

（3）在 PictureFadeFrame 窗体类中创建内部面板类PictureFadePanel，该面板类实现了Runnable接口，并重写了JComponent类的paint()方法，同时也实现了Runnable接口的run()方法，并在paint()方法中使用Graphics类的drawImage()方法绘制图片，然后在run()方法中实现改变透明度值的操作。

（4）将内部面板类PictureFadePanel的实例添加到窗体类PictureFadeFrame的内容面板上，用于在窗体上显示淡入淡出效果的图片。内部面板类PictureFadePanel的代码如下：

```
    class PictureFadePanel extends JPanel implements Runnable
{
        boolean flag=
    true;                                          //定义标记变量，
    用于控制x的值
        float
    x=0.0f;                                          //定义表示
    透明度的变量x
        AlphaComposite
    alpha=AlphaComposite.SrcOver.derive(x);          //获得表示透
    明度的AlphaComposite对象
        public void paint(Graphics g) {
```

```
    Graphics2D g2= (Graphics2D)
g;                              //获得Graphics2D对象
    g2.clearRect( 0, 0, getWidth(),
getHeight());                   //绘制图像
    g2.drawImage(img1,0,0,getWidth(),getHeight(),
this);          //绘制图像
    g2.setComposite(alpha);
//指定AlphaComposite对象
    g.drawImage(img2, 50, 40, getWidth() - 100,
getHeight() - 80, this);
    //绘制调整透明度后的图片，实现图片的淡入淡出特效
  }
  public void run() {
    while (true) {
      if (flag) {                                    //
  flag为true时
        x -=0.1f;                                  // x进
    行减0.1计算
        if (x<=0.0f) {                              //x小
    于等于0.0f时
          x=0.0f;                                 // x等于
      0.0f
        flag= false;                                //为
      flag赋值为false
        }
      } else {                                       //
  flag为false时
```

```
            x+=0.1f;                                    // x进
        行加0.1计算
            if (x>=1.0f) {                              //x大
        于等于1.0f时
                x=1.0f;                                 // x等于
          1.0f
                flag= true;                             //为
          flag赋值为true
            }
          }
          alpha=AlphaComposite.SrcOver.derive(x);
        //重新获得表示透明度的AlphaComposite对象
          repaint();                                    //
      调用paint()方法
          try {
            Thread.sleep(200);                          //
        休眠200毫秒
          } catch (InterruptedException e) {
            e.printStackTrace();
          }
        }
      }
    }
```

举一反三

根据本实例，读者可以实现以下功能。

显示淡入淡出的图片。

显示淡入淡出的文字。

实例317 随鼠标指针移动的图片

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\08\Ex08_317

实例说明

本实例通过鼠标的移动事件，演示了如何使显示有图片的标签随着鼠标指针移动，另外本实例还在显示图片的标签下方，显示一个文本颜色随机变化的标签，这是通过多线程技术实现的。运行程序，显示有图片和文本的标签就会随着鼠标指针移动，效果如图8.12所示。

技术要点

在鼠标移动的过程中，通过MouseEvent类的getX()方法获得鼠标指针在背景面板中的横坐标值，通过MouseEvent类的getY()方法获得鼠标指针在背景面板中的纵坐标值，然后将要移动图形的标签通过setLocation()方法设置到对鼠标指针当前位置坐标进行计算后的位置处，使图形在背景面板上移动。

图8.12 随鼠标指针移动的图片

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的FollowMousePictureFrame窗体类。

（3）将FollowMousePictureFrame窗体类的内容面板设置为绝对布局，然后在内容面板上分别添加名称为lb_move和lb_tip的JLabel标签控件，分别用于显示图片和文本。

（4）在窗体 FollowMousePictureFrame 中创建内部线程类 FlareThread，用于改变标签中文字的颜色。FlareThread类的代码如下：

```
class FlareThread implements Runnable {
  public void run() {
    while (true) {
      int red= (int) (Math.random() * 256);              //
    随机生成RGB颜色中的R，即红色
      int green= (int) (Math.random() * 256);            //
    随机生成RGB颜色中的G，即绿色
      int blue= (int) (Math.random() *
    256);              //随机生成RGB颜色中的B，即蓝色
      lb_tip.setForeground(new Color(red, green,
    blue));  //设置标签上文字的前景色
      try {
        Thread.sleep(500);                              //线程休
    眠500毫秒
      } catch (Exception e) {
      }
    }
  }
}
```

（5）为窗体类FollowMousePictureFrame添加鼠标移动事件，实现在鼠标指针移动时，使显示图片的标签和显示文字的标签随着鼠标

指针移动，该事件的代码如下：

```
addMouseMotionListener(new MouseMotionAdapter() {
  public void mouseMoved(final MouseEvent e) {
    int x= e.getX();                                    //获得鼠标
在窗体容器中的横坐标值
    int y= e.getY();                                    //获得鼠标
在窗体容器中的纵坐标值
    int w= lb_move.getWidth();                          //获得随
鼠标移动的图形所在标签的宽度
    int h= lb_move.getHeight();                         //获得随
鼠标移动的图形所在标签的高度
    int x1=x -w / 2;                                    //计算出随
鼠标移动的图形所在标签的横坐标值
    int y1=y - h / 2;                                   //计算出随
鼠标移动的图形所在标签的纵坐标值
    lb_move.setLocation(x1,y1);                         //设置随
鼠标移动的图形所在标签的显示位置
    int x2=x - 52;                                      //计算显示
文字的标签的横坐标值
    int y2=y1+120;                                      //计算显示文
字的标签的纵坐标值
    lb_tip.setLocation(x2,y2);                          //设置
显示文字的标签的显示位置
  }
});
```

举一反三

根据本实例，读者可以实现以下功能。

任意移动容器中的控件。

设置显示图片的初始位置。

实例318 通过键盘移动图片

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\08\Ex08_318

实例说明

本实例演示如何利用键盘移动图片。运行程序，可以通过键盘上的上、下、左、右方向键移动窗体中显示有图片的标签，效果如图8.13所示。

图8.13 通过键盘移动图片

技术要点

本实例通过键盘事件类 KeyEvent 的 getKeyCode()方法，获得按键的整数代码，并使用 if语句判断是否与所按方向键的整数代码相同。

（1）KeyEvent类的getKeyCode()方法用于获得键盘上实际按键的整数代码，该方法的定义如下：

```
public int getKeyCode()
```

参数说明

返回值：键盘上实际按键的整数代码。

（2）键盘上非数字键上与上、下、左、右方向键相对应的整数代码分别为 VK_UP、VK_DOWN、VK_LEFT、VK_RIGHT，可以使用KeyEvent类进行调用，使用方法如下：

□ KeyEvent.VK_UP：表示非数字键上的向上方向键。

□ KeyEvent.VK_DOWN：表示非数字键上的向下方向键。

□ KeyEvent.VK_LEFT：表示非数字键上的向左方向键。

□ KeyEvent.VK_RIGHT：表示非数字键上的向右方向键。

（3）使用if语句判断是否按了非数字键上的向上方向键，可以使用如下代码实现：

```
if (e.getKeyCode()==KeyEvent.VK_UP)                    //如果按了非数字键上的向上方向键，则条件成立
```

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承自JFrame类的窗体类KeyboardMovePictureFrame。

（3）将KeyboardMovePictureFrame窗体类的内容面板设置为绝对布局，然后在内容面板上添加一个名称为lb_move的JLabel标签控件，用于显示图片。

（4）为窗体类KeyboardMovePictureFrame的内容面板添加键盘按键事件，实现通过键盘上非数字键上的方向键，移动窗体上显示图片的标签控件。窗体类内容面板的键盘按键事件的代码如下：

```
getContentPane().addKeyListener(new KeyAdapter() {
  public void keyPressed(final KeyEvent e) {
    int x= lb_move.getLocation().x;                    //获得移动标签的x坐标
```

```
    int y= lb_move.getLocation().y;                    //获得移
动标签的y坐标
    if (e.getKeyCode() == KeyEvent.VK_LEFT) {
      lb_move.setLocation(x - 10,y);                   //向
  左移动，x坐标减小
    } else if (e.getKeyCode() == KeyEvent.VK_UP) {
      lb_move.setLocation(x,y - 10);                   //向
  上移动，y坐标减小
    } else if (e.getKeyCode() == KeyEvent.VK_RIGHT) {
      lb_move.setLocation(x+10,y);                     //向右
  移动，x坐标增加
    } else if (e.getKeyCode() == KeyEvent.VK_DOWN) {
      lb_move.setLocation(x,y+10);                     //向下
  移动，y坐标增加
    }
  }
});
```

举一反三

根据本实例，读者可以实现以下功能。

用键盘移动文字。

锁定键盘阻止移动。

实例319 图片动态拉伸

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\08\Ex08_319

实例说明

本实例通过Java的绘图技术，实现了使图片动态拉伸的功能。运行程序，窗体上的图片将动态放大和缩小，并不断重复该过程，效果如图8.14和图8.15所示。

图8.14 图片缩小的效果

图8.15 图片放大的效果

技术要点

本实例通过 Graphics 类的 drawImage()方法绘制图片，并使用线程动态改变所绘制图片的宽度和高度，从而实现了图片动态拉伸的动画效果。

（1）使用Graphics类的drawImage()方法，可以在指定位置绘制指定大小的图片。

（2）在实现Runnable接口的run()方法中，使用标记变量确定是放大还是缩小图片，并在图片放大和缩小到一定程度时，改变标记变量的值，以防止图片的大小超出窗体的范围，代码如下：

```
if (flag) {
  width+=2;                          //调整宽度值
  height++;                          //调整高度值
  if (width >= getWidth() || height >= getHeight()) {
    flag= false;                     //达到图像的宽度或高度时，
改变标记变量的值
  }
} else {
  width -=2;                         //调整宽度值
  height--;                          //调整高度值
  if (width <= 0 || height <= 0) {
    flag= true;                      //图像的宽度或高度小于等于
0时，改变标记变量的值
  }
}
```

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承自JFrame类的DynamicFlexImageFrame窗体类。

（3）在DynamicFlexImageFrame窗体类中创建内部面板类DynamicFlexPanel，该面板类实现了Runnable接口，并重写了JComponent类的paint()方法，同时也实现了Runnable接口的run()方法，并在paint()方法中使用Graphics类的drawImage()方法绘制图片，然后在run()方法中实现改变图片大小的操作。

（4）将内部面板类DynamicFlexPanel的实例添加到窗体类DynamicFlexImageFrame的内容面板上，用于在窗体上显示动态拉伸的图片。内部面板类DynamicFlexPanel的代码如下：

```
class DynamicFlexPanel extends JPanel implements Runnable
{
    private boolean flag= true;                    //标记变量
    int width=0;                                  //调整图像宽度的
变量
    int height=0;                                  //调整图像高度的
变量
    public void paint(Graphics g) {
      g.clearRect(0, 0, getWidth(), getHeight());  //清除绘
图上下文的内容
      g.drawImage(img,0,0,width,height, this);  //绘制指定
大小的图片
    }
    public void run() {
      while (true) {
        if (flag) {
          width+=2;                          //调整宽度值
          height++;                            //调整高度值
          if (width >= getWidth() || height >= getHeight())
        {
            flag= false;                    //达到图像的宽度或高
        度时，改变标记变量的值
          }
        } else {
          width -=2;                          //调整宽度值
          height--;                            //调整高度值
          if (width <= 0 || height <= 0) {
```

```
                flag= true;                    //图像的宽度或高度小于
            等于0时，改变标记变量的值
              }
            }
            repaint();                          //调用paint()方法
            try {
              Thread.sleep(20);                 //线程休眠20毫秒
            } catch (InterruptedException e) {
              e.printStackTrace();
            }
          }
        }
      }
```

举一反三

根据本实例，读者可以实现以下功能。

实现图片的放射效果。

实现图片的动态还原。

实例320 桌面弹球

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\08\Ex08_320

实例说明

本实例实现了桌面弹球动画。运行程序，小球将在窗体中移动，当遇到窗体边缘时，小球会改变运动轨迹，效果如图8.16所示。

技术要点

本实例通过自定义标签实现，该自定义标签继承自JLabel类，同时实现了Runnable接口，并重写了控件的paintComponent()方法，该方法是从Container类继承来的，用于在自定义标签上绘制小球，并在自定义标签类的构造方法中指定标签的首选大小，使其正好与小球的大小相同，最后在run()方法中，对绘制有小球的自定义标签在父级容器中的运动轨迹上进行控制，这主要是通过从Component类继承的方法来实现的。

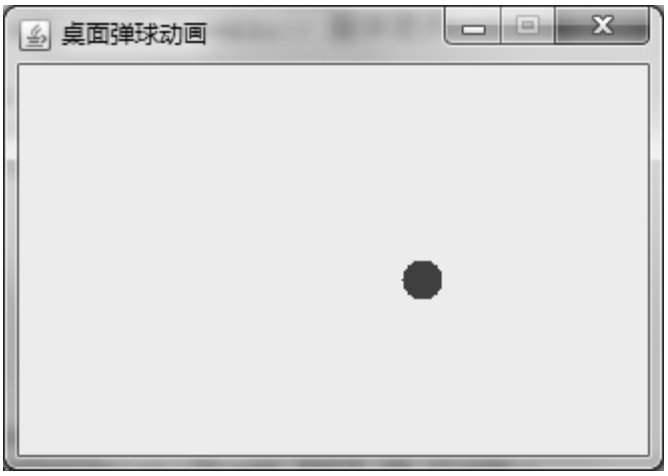


图8.16 桌面弹球动画

（1）在自定义标签中重写控件的 paintComponent()方法，用于在标签上绘制小球，该方法的代码如下：

```
@Override
protected void paintComponent(Graphics g) {
  super.paintComponent(g);                          //调用父类的方法
  g.setColor(ballColor);                            //设置默认颜色
  g.fillOval(0,0,width,height);                      //在标签上绘制球体
}
```

（2）通过getParent()方法，可以获得当前控件所在父级容器的对象，该方法是从Component类继承来的，其定义如下：

public Container getParent()

参数说明

返回值：当前控件的父级容器对象。

（3）通过 getLocation()方法，可以获得当前控件所在父级容器的位置对象，该方法也是从Component类继承来的，其定义如下：

public Point getLocation()

参数说明

返回值：一个Point对象，表示当前控件左上角在父级容器中的坐标。

（4）通过 setLocation()方法，可以设置当前控件在父级容器中的位置，该方法也是从Component类继承来的，其定义如下：

public void setLocation(int x, int y)

参数说明

- x：指定当前控件左上角在父级容器中新位置的x 坐标。
- y：指定当前控件左上角在父级容器中新位置的y 坐标。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 BallFrame 窗体类和一个继承 JLabel 类并实现Runnable接口的自定义标签类BallPanel。

（3）在标签类BallPanel中，重写控件的paintComponent()方法，用于在标签上绘制小球，在实现 Runnable 接口的 run()方法中，每隔一小段时间就对自定义标签在父级容器中的位置进行改变，从而实现桌面弹球的动画效果。

（4）自定义标签类BallPanel的代码如下：

```
package com.zzk;
```

```
import java.awt.*;
import javax.swing.JLabel;
/**
 *@author 张振坤
 */
public class BallPanel extends JLabel implements Runnable
{
    private int r=10;                                    //小
球半径
    private int width= r * 2;                             //
球宽度
    private int height= r *
2;                              //球高度
    private Color ballColor=Color.BLUE;                   //
默认颜色
    public BallPanel() {
      setPreferredSize(new
  Dimension(width,height));        //初始化大小
      new Thread(this).start();                           //
  启动小球跳跃线程
    }
    @Override
    protected void paintComponent(Graphics g) {
      super.paintComponent(g);
      g.setColor(ballColor);                             //
  设置默认颜色
```

```
    g.fillOval(0,0,width,height);                          //
在标签上绘制球体
  }
  @Override
  public void run() {
    Container parent=getParent();                          //获
得当前标签的父级容器对象
    Point myPoint=getLocation();                           //获
取初始位置
    while (true) {                                         //循
环读取父容器对象
      if (parent == null) {
        try {
          Thread.sleep(50);                                //线程休眠
        } catch (InterruptedException e) {
          e.printStackTrace();
        }
        myPoint=getLocation();                             //获取初
    始位置
        parent=getParent();                                //获得当
    前标签的父级容器对象
      } else {                                             //如果已
  经获取到父容器
        break;                                             //跳出循环
      }
    }
```

```
    int sx=myPoint.x;                                    // x坐
标
    int sy=myPoint.y;                                    //y坐
标
    int step= (int) (Math.random() * 10)%
3+1;          //移动步进
    int dx= (Math.random() * 100)>=50 ? step : -
step;      //水平步进值
    step= (int) (Math.random() * 10)% 3+1;              //
移动步进
    int dy= (Math.random() * 100)>=50 ? step : -
step;      //垂直步进值
    int stime= (int) (Math.random() *
80+10);            //随机移动速度
    while (parent.isVisible()) {
      int parentWidth=parent.getWidth();                //
  容器宽度
      int parentHeight=parent.getHeight();              //
  容器高度
      setLocation(sx, sy);                               //指
  定小球的位置
      try {
        Thread.sleep(stime);                            //通过
  休眠改变移动速度
      } catch (InterruptedException e) {
        e.printStackTrace();
      }
```

```
            sx= sx+dx * 5;                                    //水平坐
        标偏移5个像素
            sy= sy+dy* 5;                                    //垂直坐标
        偏移5个像素
            if (sy>parentHeight - height || sy<0)                    //
        检测垂直边界
                dy= -dy;                                    //防止坐标
          超出垂直边界
            if (sx>parentWidth -width || sx<0)                    //
        检测水平边界
                dx= -dx;                                    //防止坐标
          超出水平边界
          }
        }
    }
```

举一反三

根据本实例，读者可以实现以下功能。

在窗体同时进行多球弹动。

设置弹动的开始与停止。

实例321 循环滚动图片

这是一个可以用来提高基础性能的实例

实例位置：光盘\mingrisoft\08\Ex08_321

实例说明

本实例使用Java的绘图技术和多线程技术，实现在窗体上循环滚动图片的功能。运行程序，图片将从窗体的左侧向右侧移动，当移动

到右侧边缘时，返回，即向左侧移动，并重复上述过程循环滚动，效果如图8.17所示。

图8.17 循环滚动图片

技术要点

在循环移动图片时，当图片右侧边缘移动到所在面板右侧边缘时，图片不再向右移动，这可以通过判断图片左侧边缘的 x 坐标值是否大于等于所在面板的宽度减去图片的宽度进行控制；当图片左侧边缘移动到所在面板的左侧边缘时，图片不再向左移动，这可以通过判断图片左侧边缘的x坐标值是否小于等于零进行控制。

（1）当图片向右移动时，为了使图片右侧边缘移动到所在面板右侧边缘时，不再向右移动，可以用如下代码来实现：

```
if (x >= getWidth() - img.getWidth(this)) { //图片的x坐标值大于等于面板与图片宽度的差
    x = getWidth() - img.getWidth(this); //图片的x坐标值等于面板与图片宽度的差
}
```

（2）当图片向左移动时，为了使图片左侧边缘移动到所在面板左侧边缘时，不再向左移动，可以用如下代码来实现：

```
if ( x<=0 ) {                          //图片的x坐标值小于等于0，也就是超出了面板的左侧边缘
```

```
    x=0;                                    //图片的x坐标值等于0
  }
```

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的CircularRollPictureFrame窗体类。

（3）在CircularRollPictureFrame窗体类中，创建内部面板类CircularRollPicturePanel，并重写JComponent类的paint()方法，实现Runnable接口中的run()方法。

（4）将内部面板类 CircularRollPicturePanel 的实例添加到窗体类 CircularRollPictureFrame的内容面板上，用于在窗体上显示循环滚动的图片。内部面板类 CircularRollPicturePanel 的代码如下：

```
  private class CircularRollPicturePanel extends JPanel
implements Runnable {
    int
  x=0;
  //定义图片移动位置的x坐标
    int
  y=30;
  //定义图片移动位置的y坐标
    URL
  imgUrl=CircularRollPictureFrame.class.getResource("/image/p
  icture.png");  //获取图片资源的路径
    Image
  img=Toolkit.getDefaultToolkit().getImage(imgUrl);
          //获取图像对象
```

```
  public void paint(Graphics g){
    g.clearRect(0, 0, getWidth(),
getHeight());                              //清除面板上的内
容
    //在面板的指定位置绘制图像g.drawImage(img,x,y, this);
  }
  public void run() {
    boolean flag=
true;                                      //声明标记变量
    while (true){
      if (flag) {                                          //
  标记变量为true
        x=x+10;                                    //图片x
    坐标值加10
        if (x>=getWidth() - img.getWidth(this))
    {          //图片的x坐标值大于等于面板与图片宽度的差
          x=getWidth() -
      img.getWidth(this);            //图片的x坐标值等于
      面板与图片宽度的差
          flag= false;                                 //设
      置标记变量为false
        }
      }else {                                           //标
  记变量为false
        x=x - 10;                                  //图片
    x坐标值减10
```

```
            if ( x<=0 ) {                                    //图
        片的x坐标值小于等于0
                x=0;                                        //图片的x
            坐标值等于0
                flag= true;                                 //设置
            标记变量为true
              }
            }
            try {
              Thread.sleep(200);                             //
        休眠200毫秒
            } catch (InterruptedException e) {
              e.printStackTrace();
            }
            repaint();                                       //
          调用paint()方法
          }
        }
}
```

举一反三

根据本实例，读者可以实现以下功能。

制作图片在窗体内任意位置移动的动画效果。

循环滚动文字。

实例322 撞球动画

这是一个可以提高基础性能的实例

实例位置：光盘\mingrisoft\08\Ex08_322

实例说明

本实例实现了一个撞球动画，并模拟了两球相撞时会产生挤压变形的变化。运行程序，两个小球在窗体上水平相向运动，相撞后，又向相反的方向运动，两球没相撞与相撞时的效果分别如图8.18和图8.19所示。

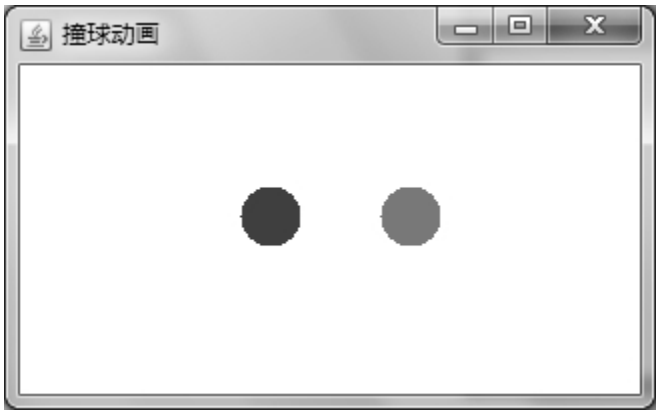


图8.18 两球没相撞的效果

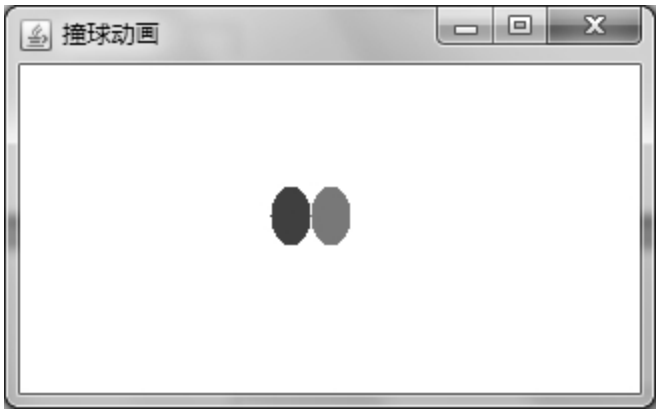


图8.19 两球相撞时的效果

技术要点

本实例主要实现的是在两球相遇时，使两球的宽度变小，这样就产生了挤压的效果，并通过两个标记变量分别控制两个球的运行方向。

（1）为了使两球相遇时，使两球的宽度变小，需要判断两个球是否相撞，代码如下：

```
    if (x1+width>=x2+1)
{                                                    //两球相撞
      x1=x1+5;
    //计算第1个球的x坐标
      width=width -
  10;                                                //球的宽度减
  10
      x2=x1+width;
      //计算第2个球的x坐标
      flag1=
  false;                                             //设置第1个
  球的标记变量为false
      flag2=
  false;                                             //设置第2个
  球的标记变量为false
      repaint();
    //调用paint()方法
    }
```

（2）使用标记变量flag1控制第1个球的运行方向，为true时第1个球右移，为false时第1个球左移，代码如下：

```
    if (flag1) {                                     //标记变量为true，
第1个球右移
      x1=x1+10;                                  //图片x坐标值加10，即第1
  个球右移
      width=30;                                  //球的宽度
    } else {                                         //标记变量为false，
第1个球左移
```

```
    x1=x1 - 10;                          //图片x坐标值减10，即
  第1个球左移
    width=30;                           //球的宽度
    if (x1<=0) {                         //图片的x坐标值小于等
  于0
      x1=0;                             //图片的x坐标值等于0
      flag1= true;                      //设置标记变量为true
    }
  }
```

（3）使用标记变量flag2控制第二个球的运行方向，为true时第2个球左移，为false时第2个球右移，代码如下：

```
  if (flag2) {                           //标记变量为true，
第2个球左移
    x2=x2 - 10;                          //图片x坐标值减10，即
  第2个球左移
    width = 30;
  } else {                               //标记变量为false，
第2个球右移
    x2=x2+10;                           //图片x坐标值加10，即第2
  个球右移
    width=30;                           //球的宽度
    if (x2>=getWidth() -width) {         //图片的x坐标值大于
  等于面板的宽度与球的宽度之差
      x2=getWidth() -width;              //图片的x坐标值等于面
  板的宽度与球的宽度之差
      flag2= true;                      //设置标记变量为true
    }
```

```
}
```

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的BilliardBallFrame窗体类。

（3）在 BilliardBallFrame 窗体类中，创建内部面板类BilliardBallPanel，重写 JComponent类的paint()方法，并实现Runnable接口中的run()方法，完成撞球动画。

（4）将内部面板类BilliardBallPanel的实例添加到窗体类BilliardBallFrame的内容面板上，用于在窗体上显示撞球动画。内部面板类BilliardBallPanel的代码如下：

```
private class BilliardBallPanel extends JPanel implements Runnable {
    int x1=0;                                    //定义第1个球移动位置的x坐标
    int y1=60;                                   //定义第1个球移动位置的y坐标
    int x2=326 - 30;                             //定义第2个球移动位置的初始x坐标为窗体宽度减球的宽度
    int y2=60;                                   //定义第2个球移动位置的y坐标
    int width=30;                                //定义球的宽度
    int height=30;                               //定义球的高度
    public void paint(Graphics g) {
      g.clearRect(0, 0, getWidth(), getHeight());  //清除面板上的内容
      g.setColor(Color.BLUE);                    //设置颜色
```

```
    g.fillOval(x1,y1,width,height);          //绘制第1个球
    g.setColor(Color.RED);                   //设置颜色
    g.fillOval(x2,y2,width,height);          //绘制第2个球
  }
  public void run() {
    boolean flag1= true;                     //声明第1个球的标
记变量
    boolean flag2= true;                     //声明第2个球的标
记变量
    while (true) {
      //第1个球的x坐标值加上球的宽度大于等于第2个球的x坐
  标值加1，表示两球相遇
      if (x1+width>=x2+1) {          //两球相撞
        x1=x1+5;                     //计算第1个球的x坐标
        width=width - 10;            //球的宽度减10
        x2=x1+width;                 //计算第2个球的x坐标
        flag1= false;                  //设置第1个球的标记
    变量为false
        flag2= false;                  //设置第2个球的标记
    变量为false
        repaint();                     //调用paint()方法
      } else {                         //两球没相撞
        if (flag1) {                           //标记变量为
    true，第1个球右移
          x1=x1+10;                    //图片x坐标值加10，
      即第1个球右移
          width=30;                      //球的宽度
```

```
    } else {                                    //标记变量为
false，第1个球左移
      x1=x1 - 10;                               //图片x坐标值减
  10，即第1个球左移
      width=30;                                 //球的宽度
      if (x1<=0) {                               //图片的x坐标值小
  于等于0
        x1=0;                                  //图片的x坐标值等于0
        flag1= true;                            //设置标记变量为
    true
      }
    }
    if (flag2) {                                //标记变量为
true，第2个球左移
      x2=x2 - 10;                               //图片x坐标值减
  10，即第2个球左移
      width = 30;
    } else {                                    //标记变量为
false，第2个球右移
      x2=x2+10;                                 //图片x坐标值加10，
  即第2个球右移
      width=30;                                 //球的宽度
      if (x2>=getWidth() -width) {       //图片的x坐标
  值大于等于面板的宽度与球的宽度之差
        x2=getWidth() -width ;       //图片的x坐标值等
    于面板的宽度与球的宽度之差
```

```
                flag2= true;                    //设置标记变量为
            true
              }
            }
          }
          try {
            Thread.sleep(200);                  //休眠200毫秒
          } catch (InterruptedException e) {
            e.printStackTrace();
          }
          repaint();                            //调用paint()
        方法
          }
        }
      }
```

举一反三

根据本实例，读者可以实现以下功能。

实现两球相撞后粘在一起运动的效果。

实现大小两球相撞后小球穿过大球的效果。

实例323 电影胶片特效

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\08\Ex08_323

实例说明

本实例使用Java的多线程技术，实现电影胶片特效。运行程序，窗体上显示5个带有图片的标签，并将依次从窗体的右侧向左侧运动，

就像放电影时胶片的运动一样，而且，每个标签上显示的图片都是在两个图片之间变换的，效果如图8.20所示。

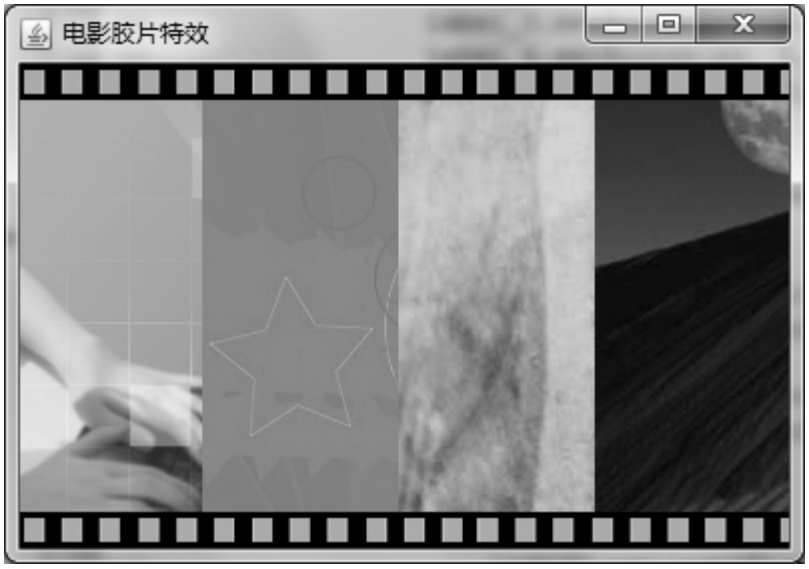


图8.20 电影胶片特效

技术要点

根据图片的宽度和高度，创建5个从左向右运动的标签，并将其中4个标签放到绝对布局的面板容器中，另一个则在移动的过程中进行填补，使图片能够连贯地移动，并使用控件的setBounds()方法，指定每个控件的位置和大小，从而实现电影胶片特效。

（1）根据图片的宽度创建5个标签，然后计算出每个标签左上角的x坐标，代码如下：

```
    int
x1=0;                                          //第1个
标签显示位置的变量
    int
x2=98;                                          //第2个
标签显示位置的变量
    int
x3=196;                                        //第3个
```

标签显示位置的变量

```
    int
x4=294;                                          //第4个
```

标签显示位置的变量

```
    int
x5=392;                                          //第5个
```

标签显示位置的变量

（2）为了连贯地移动图片，需要使每个控件的x坐标值不断变化，代码如下：

```
    x1=x1 -
7;                                               //第1个标签
```

的左边界减7，使其左移

```
    x2=x2 -
7;                                               //第2个标签
```

的左边界减7，使其左移

```
    x3=x3 -
7;                                               //第3个标签
```

的左边界减7，使其左移

```
    x4=x4 -
7;                                               //第4个标签
```

的左边界减7，使其左移

```
    x5=x5 -
7;                                               //第5个标签
```

的左边界减7，使其左移

（3）使用控件的setBounds()方法，设置每个标签控件的位置，该方法的定义如下：

public void setBounds(int x, int y, int width, int height)

参数说明

- x：控件左上角的x 坐标。
- y：控件左上角的y 坐标。
- width：控件的宽度。
- height：控件的高度。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的CinefilmEffectFrame窗体类。

（3）在CinefilmEffectFrame窗体类中，创建实现了Runnable接口的内部类CinefilmThread，用于实现电影胶片的动画效果。内部类CinefilmThread的代码如下：

```
private class CinefileThread implements Runnable {
  public void run() {
    while (true) {
      x1=x1 - 7;                                    //第1个标签的左边界减7，使其左移
      x2=x2 - 7;                                    //第2个标签的左边界减7，使其左移
      x3=x3 - 7;                                    //第3个标签的左边界减7，使其左移
      x4=x4 - 7;                                    //第4个标签的左边界减7，使其左移
      x5=x5 - 7;                                    //第5个标签的左边界减7，使其左移
```

```
    label_1.setBounds(x1,0,98,210);
//设置第1个标签的显示位置
    label_2.setBounds(x2,0,98,210);
//设置第1个标签的显示位置
    label_3.setBounds(x3,0,98,210);
//设置第1个标签的显示位置
    label_4.setBounds(x4,0,98,210);
//设置第1个标签的显示位置
    label_5.setBounds(x5,0,98,210);
//设置第1个标签的显示位置
    if (x1== -98) {                                    //
当第1个标签的显示位置是-98时执行
      indexFlag= !indexFlag;                           //
  改变indexFlag的值
      x1=392;                                          //设置第
  1个标签的显示位置
      if (indexFlag) {
        label_1.setIcon(SwingResourceManager.getIcon(Ci
    nefilmEffectFrame.class, "/image/6.jpg"));    //
    indexFlag为true时改变的图片
      } else {
        label_1.setIcon(SwingResourceManager.getIcon(Ci
    nefilmEffectFrame.class, "/image/1.jpg"));    //
    indexFlag为false时改变的图片
      }
    }
    ……//省略了其他4个标签的代码
```

```
            try {
              Thread.sleep(150);                                    //
        线程睡眠150毫秒
            } catch (Exception ex) {
            }
        }
      }
    }
```

举一反三

根据本实例，读者可以实现以下功能。

将标签内的图片内容自定义为其他图片。

实现电影播放功能。

实例324 随机移动的图片

这是一个可以启发思维的实例

实例位置：光盘\mingrisoft\08\Ex08_324

实例说明

本实例使用Java的绘图技术和多线程技术，实现随机移动图片的功能。运行程序，图片将在窗体上随机移动，但是不会移出窗体，效果如图8.21所示。

图8.21 随机移动的图片

技术要点

本实例使用 Random 类获得随机数，随机生成图片显示位置的坐标，这需要使用 Random类的构造方法创建对象，然后使用nextInt()方法获得随机整数。

（1）使用Random类的构造方法创建对象，用于创建一个新的随机数生成器。

该类的构造方法定义如下：

```
public Random()
```

（2）使用Random类的对象，调用nextInt()方法生成随机整数，该方法的定义如下：

```
public int nextInt(int n)
```

参数说明

● n：要返回的随机数的范围，必须为正数。

● 返回值：下一个伪随机数，是在此随机数生成器序列中 0（包括）和 n（不包括）之间均匀分布的整数值。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的RandomMovePictureFrame窗体类。

（3）在 RandomMovePictureFrame 窗体类中，创建内部面板类RandomMovePicturePanel，重写JComponent类的paint()方法，并实现Runnable接口的run()方法，从而实现随机移动图片的功能。

（4）将内部面板类 RandomMovePicturePanel 的实例添加到窗体类 RandomMovePicture Frame的内容面板上，用于在窗体上显示随机移动的图片。面板类RandomMovePicturePanel的代码如下：

```
    private class RandomMovePicturePanel extends JPanel
implements Runnable {
        Random random=new Random();                          //创建
    Random对象
        int x=0;                                             //定义
    图片移动位置的x坐标
        int y=0;                                             //定义
    图片移动位置的y坐标
        URL
    imgUrl=RandomMovePictureFrame.class.getResource("/image/pic
    ture.png");                  //获取图片资源的路径
        Image
    img=Toolkit.getDefaultToolkit().getImage(imgUrl);    //获取
    图像对象
        public void paint(Graphics g) {
          g.clearRect(0, 0, getWidth(),
      getHeight());              //清除面板上的内容
          g.drawImage(img,x,y, this);                        //在
      面板的指定位置绘制图像
        }
        public void run() {
```

```
    while (true) {
      x= random.nextInt(winWIDTH - 110);              //随
  机获得图片移动位置的x坐标
      y= random.nextInt(winHEIGHT - 140);            //随机
  获得图片移动位置的y坐标
      try {
        Thread.sleep(200);                          //休眠200
    毫秒
      } catch (InterruptedException e) {
        e.printStackTrace();
      }
      repaint();                                      //调用
  paint()方法
    }
  }
}
```

举一反三

根据本实例，读者可以实现以下功能。

在窗体中左右循环移动图片。

实现暂停功能。

实例325 雪花飘落动画

本实例可以提高工作效率

实例位置：光盘\mingrisoft\08\Ex08_325

实例说明

本实例使用Java的绘图技术，实现雪花飘落动画。运行程序，效果如图8.22所示。

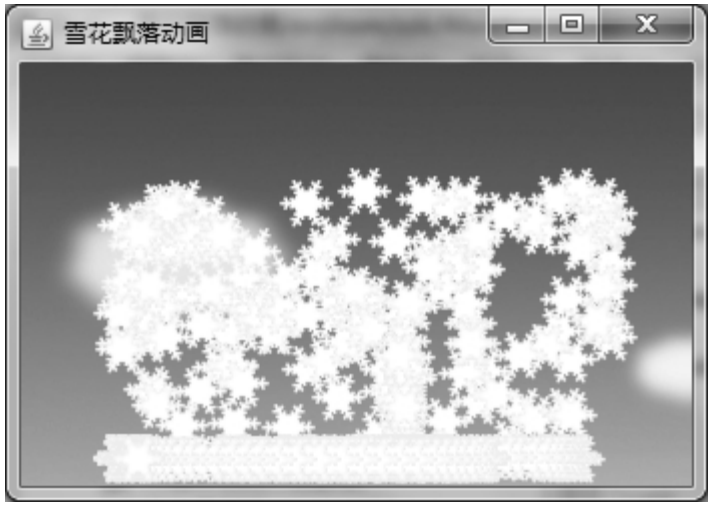


图8.22 雪花飘落动画

技术要点

本实例通过创建自定义标签，该自定义标签继承自JLabel类，同时实现了Runnable接口，并在run()方法中实现在父级容器中放置显示有雪花图片的自定义标签对象。

（1）通过getParent()方法，可以获得当前控件所在父级容器的对象。

（2）通过getLocation()方法，可以获得当前控件所在父级容器的位置对象。

（3）通过setLocation()方法，可以设置当前控件在父级容器中的位置。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 MainFrame 窗体类、一个继承 JLabel 类并实现Runnable接口的自定义标签类SnowFlakeLabel，以及一个背景面板类BackgroundPanel。

（3）在标签类SnowFlakeLabel中的实现Runnable接口的run()方法中，每隔一小段时间就对自定义标签在父级容器中的位置进行改变，从而实现雪花飘落的效果。

（4）自定义标签类SnowFlakeLabel的代码如下：

```
package com.zzk;
import java.awt.*;
import javax.swing.*;
/**
  *@author 张振坤
  */
public class SnowFlakeLabel extends JLabel implements Runnable {
    private final static ImageIcon snow = new ImageIcon(SnowFlakeLabel.class.getResource("/image/snowflake.png"));
    private int width= snow.getIconWidth();              //雪花的宽度
    private int height= snow.getIconHeight();            //雪花的高度
    /**
      *构造方法
      */
    public SnowFlakeLabel() {
      setSize(new Dimension(width,height));              //雪花的初始化大小
      setIcon(snow);                                     //指定图标
```

```
    new Thread(this).start();                    //创建并
启动线程
  }
  public void run() {
    Container parent=getParent();                //获取父
容器对象
    Point myPoint=getLocation();                 //获取初
始位置
    while (true) {                               //循环读取
父容器对象
      if (parent == null) {
        try {
          Thread.sleep(50);                      //线程休眠50毫秒
        } catch (InterruptedException e) {
          e.printStackTrace();
        }
        myPoint=getLocation();                   //获取初始位置
        parent=getParent();                      //获取父容器对
    象
      } else {                                   //如果已经获
  取到父容器
        break;                                   //跳出循环
      }
    }
    int sx=myPoint.x;                            // x坐标
    int sy=myPoint.y;                            //y坐标
```

```
    int stime= (int) (Math.random() * 30+10);       //随机
移动速度
    int parentHeight=parent.getHeight();            //容器
高度
    while (parent.isVisible() && sy < parentHeight -
height) {
      setLocation(sx, sy);                          //指定位置
      try {
        Thread.sleep(stime);                        //线程休眠
      } catch (InterruptedException e) {
        e.printStackTrace();
      }
      sy+=2;                                        //垂直偏移2个像素
    }
  }
}
```

（5）在MainFrame窗体类中，为背景面板类BackgroundPanel的实例添加鼠标移动事件，用于向背景面板中添加自定义雪花标签对象，代码如下：

```
backgroundPanel.addMouseMotionListener(new MouseAdapter()
{
  public void mouseMoved(MouseEvent e) {           //鼠标
移动事件
    SnowFlakeLabel snow=new SnowFlakeLabel();  //创建雪花
飘落标签
    Point point= e.getPoint();                      //获得
鼠标位置
```

```
        snow.setLocation(point);                    //指定雪
花在背景面板上的位置
        backgroundPanel.add(snow);                  //将雪花
添加到背景面板上
    }
});
```

举一反三

根据本实例，读者可以实现以下功能。

利用update()方法消除动画闪烁。

实现飘落的花瓣动画。

实例326 图片旋转动画

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\08\Ex08_326

实例说明

本实例实现图片旋转动画。运行程序，效果如图8.23所示。

图8.23 图片旋转动画

技术要点

本实例是使用Graphics2D类的translate()和rotate()方法实现的，其中translate()方法用于平移坐标轴，rotate()方法用于旋转绘图上下文。

（1）Graphics2D类的translate()方法用于平移坐标轴，该方法的定义如下：

public abstract void translate(int x, int y)

参数说明

- x：指定的x 坐标。
- y：指定的y 坐标。

（2）Graphics2D类的rotate()方法用于旋转绘图上下文，该方法的定义如下：

参数说明

public abstract void rotate(double theta)

theta：以弧度为单位的旋转角度。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的RotateImageFrame窗体类。

（3）在RotateImageFrame窗体类中，创建内部面板类RotatePanel，重写JComponent类的paint()方法，用于绘制旋转的图片，并使该类实现Runnable接口中的run()方法，用于改变图片的旋转角度。

（4）将内部面板类RotatePanel的实例添加到窗体类RotateImageFrame的内容面板上，用于在窗体上显示旋转的图片。内部面板类RotatePanel的代码如下：

```
class RotatePanel extends JPanel implements Runnable {
```

```
    int angle=0;                                    //初始旋转
角度
    public void paint(Graphics g) {
      Graphics2D g2= (Graphics2D) g;                //获得
  Graphics2D对象
      g2.translate(190,120);                        //平移坐
  标轴
      g2.clearRect(-190, -120,getWidth(),getHeight());  //
  清除画布上的内容
      g2.rotate(Math.toRadians(angle));             //旋转
  画布
      g2.drawImage(img, -95, -80,190,160, this);      //绘
  制指定大小的图片
    }
    public void run() {
      while (true) {
        angle= (angle+10)% 360;                     //计算旋转角度
        try {
          Thread.sleep(200);                        //休眠200毫秒
        } catch (InterruptedException e) {
          e.printStackTrace();
        }
        repaint();                                  //调用paint()
    方法
      }
    }
  }
```

举一反三

根据本实例，读者可以实现以下功能。

图片计时旋转。

图片的逆时针旋转。

实例327 图片闪现动画

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\08\Ex08_327

实例说明

本实例使用Java的绘图和多线程技术，实现图片闪现动画。运行程序，窗体上会出现闪现的图片，即窗体上的图片在很短的时间内，交替消失和出现，效果如图8.24和图8.25所示。

图8.24 无闪现图片的效果

图8.25 有闪现图片的效果

技术要点

本实例通过Java绘图技术绘制图片，通过多线程技术控制图片是否出现，从而实现了图片闪现的动画特效。

（1）使用Graphics类的drawImage()方法，完成图片的绘制。

（2）通过实现 Runnable 接口实现多线程，并在 run()方法中通过定义标记变量来控制图片的闪现，当需要显示闪现图片时，就通过指定图片的URL创建图像对象；当不需要显示闪现的图片时，就通过空字符串创建一个空的图像对象。run()方法中通过标记变量控制闪现图片的代码如下：

```
    if (flag)
{                                                    // flag为
true
      flag=
  false;                                             //赋值为
  false
      fadeImage=Toolkit.getDefaultToolkit().getImage(imgUrl);
          //通过指定图片的URL获取图像对象
    } else {
```

```
        flag=
    true;                                          //赋值为
    true
        fadeImage=Toolkit.getDefaultToolkit().getImage("");
            //获取空的图像对象
      }
```

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的PictureFadeFrame窗体类。

（3）在 PictureFadeFrame 窗体类中创建内部面板类PictureFadePanel，该面板类实现了Runnable接口，并重写JComponent类的paint()方法和实现Runnable接口的run()方法。在paint()方法中完成绘制闪现图片的功能；在run()方法中控制是否显示闪现图片。

（4）将内部面板类PictureFadePanel的实例添加到窗体类PictureFadeFrame的内容面板上，用于在窗体上显示图片闪现特效。内部面板类PictureFadePanel的代码如下：

```
    class TextFadePanel extends JPanel implements Runnable {
        boolean flag=
    true;                                          //标记
    变量
        String value=
    "";                                            //存放
    绘制内容的变量
        public void paint(Graphics g) {
```

```
    g.clearRect(0, 0,
getWidth(),getHeight());                                    //清
除绘图上下文的内容
    g.drawImage(img,0,0,getWidth(),getHeight(),this);
                    //绘制图像
    g.drawImage(fadeImage,10,10,getWidth()-20,getHeight()
-20,this);            //绘制闪现的图像对象
  }
  public void run() {
    try {
      while (true)
  {                                                  //读取内容
        Thread.sleep(200);
            //当前线程休眠200毫秒
        if (flag)
    {                                                  // flag
    为true
          flag=
      false;                                           //赋
      值为false
          fadeImage=Toolkit.getDefaultToolkit().getImage(
      imgUrl);    //获取图像对象
        } else {
          flag=
      true;                                            //赋值
      为true
```

```
                fadeImage=Toolkit.getDefaultToolkit().getImage(
            "");          //获取图像对象
              }
              repaint();
                //调用paint()方法
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

举一反三

根据本实例，读者可以实现以下功能。

实现不同图片的交替显示。

控制图片的闪现时间。

实例328 帧动画效果

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\08\Ex08_328

实例说明

本实例利用多线程技术实现帧动画效果。运行程序，将显示卡通小人吃饼干的全过程，效果如图8.26所示。单击窗体上的“暂停”按钮，可以暂停动画的播放；单击“继续”按钮，可以继续播放动画；单击“停止”按钮，将停止动画的播放；单击“播放”按钮，又开始重新播放动画。

技术要点

本实例通过将作为帧动画的图片文件，按数字1、2、3等为主文件名顺序进行命名，然后使用多线程技术，每隔一段时间就动态生成一幅图片的完整路径，并使用JLabel标签的setIcon()方法为标签指定图片，从而实现帧动画效果。

图8.26 帧动画效果

（1）本实例需要8幅图片，分别按顺序命名为1.jpg、2.jpg、3.jpg、4.jpg、5.jpg、6.jpg、7.jpg和8.jpg，这样就可以定义一个int型变量index，使其初始值为1，然后与字符串“.jpg”连接形成图片文件的完整名称1.jpg，并在一段时间间隔后，使index加1，重复上述过程，即可得到其他图片文件的名称。当index达到8时，为index赋值为1，这样就可以循环获得图片文件的名称了。

（2）在线程的run()方法中，生成每幅图片的完整路径，并使用JLabel类的setIcon()方法为标签指定图片，主要代码如下：

```
    String picture=
"/image/";                                          //创建图片存放
位置和文件名的变量
    index++;
  //图片的主文件名索引加1
    if (index <= 8) {
```

```
    picture=picture+ index+
".jpg";                                    //通过索引获得图片的位置
和文件名
  } else {
    index=1;                                                    //
图片的主文件名索引赋值为1
    picture=picture+ index+
".jpg";                                    //通过索引获得图片的位置
和文件名
  }
  label.setIcon(SwingResourceManager.getIcon(FrameActionFra
me.class, picture)); //改变标签上显示的图片，实现动画效果
```

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的FrameActionFrame窗体类。

（3）在FrameActionFrame窗体类中，创建内部线程类ActionThread，用于完成图片文件的切换，改变标签上的图标实现帧动画效果。线程类ActionThread的代码如下：

```
  private class ActionThread implements Runnable {
    private int
index=1;                                          //用于控制图形
文件的主文件名
    public void run() {
      while (true) {
        if (stop) {
```

```
    thread=null;                                    //销
毁线程
    break;                                         //跳出
循环，结束线程的执行
  }
  if (flag) {
    String picture=
"/image/";                              //创建图片存放位置和
文件名的变量
    index++;
    if (index <= 8) {
      picture=picture+ index+ ".jpg";                //
  通过索引获得图片的位置和文件名
    } else {
      index = 1;
      picture=picture+ index+ ".jpg";                //
  通过索引获得图片的位置和文件名
    }
    label.setIcon(SwingResourceManager.getIcon(FrameA
ctionFrame.class,picture));          //改变标签上显示的
图片实现动画效果
    try {
      Thread.sleep(200);                            //线
  程睡眠200毫秒
    } catch (Exception ex) {
    }
  }
```

```
    }
  }
}
```

（4）在窗体类FrameActionFrame中，定义两个boolean型的成员变量flag和stop，其中flag用于标识动画播放、暂停和继续，stop是用于标识动画播放和停止的成员变量，代码如下：

```
boolean flag= true;                                    //用于标识动画播放、暂停和继续的成员变量
boolean stop=
false;                                                 //用于标识动画播放和停止的成员变量
```

（5）窗体类FrameActionFrame上的“播放”按钮，用于实现播放帧动画，其事件代码如下：

```
button.addActionListener(new ActionListener() {
  public void actionPerformed(final ActionEvent arg0) {
    if (thread == null) {
      thread=new Thread(new ActionThread());    //如果线程为空，则创建线程对象
    }
    if (!thread.isAlive()) {                          //如果线程不是活动线程则执行下面的代码，启动线程的执行
      // stop为false，flag为true表示执行帧动画
      stop = false;
      flag = true;
      thread.start();                                 //启动线程的执行
    }
```

```
    }
});
```

（6）窗体类FrameActionFrame上的“暂停”按钮，用于实现暂停帧动画的播放，其事件代码如下：

```
button_1.addActionListener(new ActionListener() {
    public void actionPerformed(final ActionEvent arg0) {
        flag= false;          //暂停动画播放
    }
});
```

（7）窗体类FrameActionFrame上的“继续”按钮，用于实现继续播放暂停后的帧动画，其事件代码如下：

```
button_3.addActionListener(new ActionListener() {
    public void actionPerformed(final ActionEvent arg0) {
        flag= true;          //继续播放动画
    }
});
```

（8）窗体类FrameActionFrame上的“停止”按钮，用于实现停止帧动画的播放，其事件代码如下：

```
button_2.addActionListener(new ActionListener() {
    public void actionPerformed(final ActionEvent arg0) {
        stop= true;          //停止播放动画
    }
});
```

举一反三

根据本实例，读者可以实现以下功能。

实现小鸡吃米粒的动画效果。

实现走路的小人动画效果。

实例329 水波动画

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\08\Ex08_329

实例说明

本实例实现图片的水波动画。运行程序，将在窗体上显示图片的水波动画，就像风吹水面产生波纹的效果，如图8.27所示。

技术要点

本实例主要是使用Graphics类的copyArea()方法复制图像区域，并通过多线程技术在指定的时间间隔内，使所复制图片区域的位置发生改变，从而实现最终的水波动画效果。

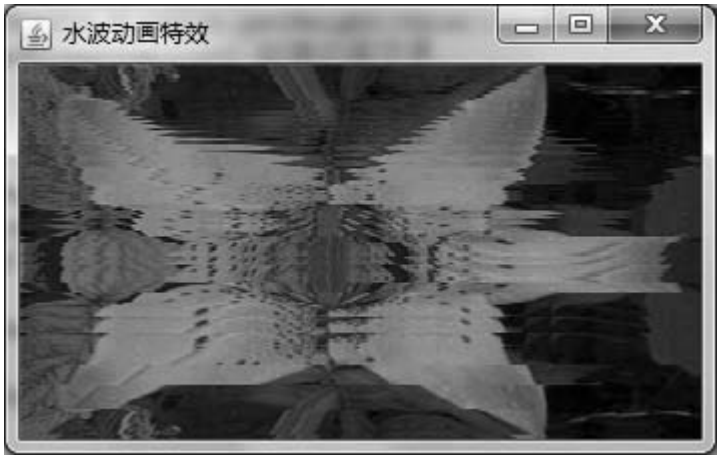


图8.27 水波动画特效

使用Graphics类的copyArea()方法，可以复制图像区域，该方法的定义如下：

```
public abstract void copyArea(int x, int y, int width, int height, int dx, int dy)
```

参数说明

- x：源矩形的x 坐标。
- y：源矩形的y 坐标。
- width：源矩形的宽度。

- height：源矩形的高度。
- dx：复制像素的水平距离。
- dy：复制像素的垂直距离。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的WaterWaveActionFrame窗体类。

（3）在WaterWaveActionFrame窗体类中，创建内部面板类WaterWaveActionPanel，并实现Runnable接口，用于实现水波动画效果。

（4）将内部面板类WaterWaveActionPanel的实例添加到窗体类WaterWaveActionFrame的内容面板上，用于在窗体上显示水波动画效果的图片。面板类WaterWaveActionPanel的代码如下：

```
class WaterWaveActionPanel extends JPanel implements Runnable {
    private Graphics graphics;                          //Graphics对象
    private Graphics waveGraphics;                      //倒影的Graphics对象
    private Image image;                                //原Image对象
    private Image waveImage;                            //表示倒影的Image对象
```

```
    private int
currentIndex;                                    //当前图
像索引
    private int
imageWidth;                                      //图像的
宽度
    private int
imageHeight;                                     //图像的
高度
    private boolean
imageLoaded;                                     //表示图片是
否被加载的标记
```

（5）调用paint()方法，用于在面板上绘制水波动画效果的图片，代码如下：

```
  public void paint(Graphics g) {
    if (waveImage != null) {
      g.drawImage(waveImage, -currentIndex * imageWidth,0,
  this);    //绘制图像
    }
    g.clearRect(imageWidth,0, imageWidth * 4,
  imageHeight);          //清除显示区域右侧的内容
  }
```

（6）使用多线程对复制的图像区域进行动态调整，以实现水波动画特效。代码如下：

```
  public void run() {
    currentIndex = 0;
```

```
    while (!imageLoaded)
{                                          //如果图片未加载
      repaint();
    //重绘屏幕
      graphics=getGraphics();
    //获得Graphics对象
      MediaTracker mediatracker=new
  MediaTracker(this);           //创建媒体跟踪对象
      URL
  imgUrl=WaterWaveActionFrame.class.getResource("/img/image
  .jpg");                       //获取图片资源的路径
      image=Toolkit.getDefaultToolkit().getImage(imgUrl);
          //获取图像资源
      mediatracker.addImage(image,0);
      //添加图片
      try {
        mediatracker.waitForAll();
        //加载图片
        imageLoaded=
    !mediatracker.isErrorAny();              //是否有错误发
    生
      } catch (InterruptedException ex) {
      }
      if (!imageLoaded) {                                  //加
  载图片失败
        graphics.drawString("加载图片错
    误",10,40);          //输出错误信息
```

```
        continue;
      }
      imageWidth= image.getWidth(this);                    //
得到图像宽度
      imageHeight=
image.getHeight(this);                    //得到图像高度
      createWave();                                        //调用
方法，实现动画效果
      break;
    }
    try {
      while (true) {
        repaint();                                         //重绘
    屏幕
        currentIndex++;                                    //调整
    当前图像索引
        if (currentIndex==12) {                            //如
    果当前图像索引为12
          currentIndex=0;                                  //设置当
    前图像索引为0
        }
        Thread.sleep(50);                                  //线程
    休眠50毫秒
      }
    } catch (InterruptedException ex) {
    }
}
```

（7）创建水波效果的图片，代码如下：

```
  public void createWave() {
    Image img= createImage(imageWidth,
imageHeight);          //以图像高度创建Image实例
    Graphics g = null;
    if (img != null) {
      g= img.getGraphics();                              //得
  到Image对象的Graphics对象
      g.drawImage(image,0,0, this);                       //
  绘制Image
      for (int i = 0; i < imageHeight; i++) {
        g.copyArea(0, imageHeight - 1 - i, imageWidth,1,0,-
    imageHeight+1+ (i * 2));              //拷贝图像区域
      }
    }
    waveImage= createImage(13 * imageWidth,
imageHeight);      //得到波浪效果的Image实例
    if (waveImage != null) {
      waveGraphics=waveImage.getGraphics();               //
  得到波浪效果的Graphics实例
      waveGraphics.drawImage(img,12 * imageWidth,0,
  this);  //绘制图像
      int j = 0;
      while (j < 12) {
        simulateWaves(waveGraphics, j);                   //
    调用方法
        j++;
```

```
        }
        //绘制图像g.drawImage(image,0,0, this);
      }
    }
```

（8）模拟波浪效果，代码如下：

```
    public void simulateWaves(Graphics g, int i)
{                          //波浪效果模拟
        double d = (6.0 * i) / 12;
        int j= (12 - i) *
  imageWidth;                                    //计算水平移动
  的距离
        int waveHeight= imageHeight /
  16;                                  //用于计算水波的高度
        for (int m = 0; m < imageHeight; m++) {
          int k= (int) ((waveHeight * (m+28)
    *Math.sin(waveHeight* (imageHeight -m) / (m+1)+d)) /
    imageHeight);       //用于控制要拷贝矩形区域的宽度
          if (m <-k)
            g.copyArea(12* imageWidth,m, imageWidth,1, -
        j,0);       //拷贝图像区域，形成波浪
          else
            g.copyArea(12 * imageWidth,m+k, imageWidth,1, -j,
        -k);  //拷贝图像区域，形成波浪
        }
      }
    }
```

举一反三

根据本实例，读者可以实现以下功能。
使用repaint()方法刷新页面效果。
实现动态水波。

第9章 文件操作技术

文件与数据库
操作磁盘文件夹
文件的读取与写入
文件控制
文件批量操作
RAR文件压缩
数据压缩的网络应用

9.1 文件与数据库

实例330 提取数据库内容到文件

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\09\Ex09_330

实例说明

数据的备份功能是数据管理系统的重点，它主要涉及两种技术：首先是从数据库中读取需要备份的数据，通常是一个表格；其次是将数据写入本地文件。本实例通过程序编码也实现了该功能：利用树结构选择需要备份的表，利用输出流将数据写入本地文件。实例运行效果如图9.1所示。

图9.1 实例运行效果

技术要点

在本实例中，需要写入文本文件的内容是字符，因此使用FileWriter 类。它是写入字符文件的便捷类。本实例使用到的方法如表9.1所示。

表9.1 FileWriter类的常用方法

方 法 名	作 用
FileWriter(File file)	根据给定的 File 对象构造一个 FileWriter 对象
write(String str)	将字符串 str 写入到文件中
flush()	在释放 FileWriter 对象之前将缓冲中的数据写入文件中
close()	释放 FileWriter 对象占用的资源

实现过程

（1）继承JFrame编写一个窗体类，名称为DataOutputFrame。

（2）设计DataOutputFrame窗体类时用到的控件及说明如表9.2所示。

表9.2 窗体的控件及说明

控 件 类 型	控 件 命 名	控 件 用 途
JTree	tree	显示数据库中所有表的名称
JTable	table	显示用户选择的表的信息
JButton	button	实现提取数据的功能

（3）编写监听单击按钮事件的方法do_button_actionPerformed()，它实现了让用户选择数据写入的文件和写入数据的功能。代码如下：

```
    protected void do_button_actionPerformed(ActionEvent
arg0) {
        JFileChooser fileChooser = new JFileChooser();
        fileChooser.setFileFilter(new
  FileNameExtensionFilter("文本文件", "txt"));
        fileChooser.setFileSelectionMode(JFileChooser.FILES_ONL
  Y);
```

```
  fileChooser.setMultiSelectionEnabled(false);
            //设置成单选
  int result = fileChooser.showSaveDialog(tree);
  if (result == JFileChooser.APPROVE_OPTION) {
    File file=
fileChooser.getSelectedFile();                          //
获得用户选择的文件
    FileWriter fileWriter=null;                          //
在trycatch块外面创建FileWriter对象，方便释放资源
    try {
      fileWriter=new
  FileWriter(file);                          //利用用户选
  择的文件创建FileWriter对象
      //获得用户选择的表格的数据
      Vector tableBody = ((DefaultTableModel)
  table.getModel()).getDataVector();
      //利用StringBuilder对象来存储表格的数据
      StringBuilder builder = new StringBuilder();
      for (int i=0; i< tableBody.size(); i++)
  {                    //遍历表格的所有行
        Vector row = (Vector) tableBody.elementAt(i);
        for (int j=0; j< row.size(); j++)
    {                       //遍历一行的所有列
          builder.append(row.elementAt(j)+
      "\t");               //各列之间用制表符分隔
        }
```

```
            builder.append("\r\n");
            //各行之间用换行符分隔
        }
        fileWriter.write(builder.toString());
              //将数据写入到文件
        fileWriter.flush();
            //将缓冲中的数据写入到文件中
        JOptionPane.showMessageDialog(this, "数据导出成
    功");          //提示用户数据导出成功
      } catch (IOException e)
{                                        //捕获IO异常
        e.printStackTrace();
      } finally {
        if (fileWriter !=null)
    {                                  //如果FileWriter对象非空就
    释放它占用的资源
          try {
            fileWriter.close();
          } catch (IOException e) {
            e.printStackTrace();
          }
        }
      }
    }
}
```

举一反三

根据本实例，读者可以开发以下程序。

将表中的数据写入文件中。

用输入流写入文件。

实例331 提取文本文件的内容到MySQL数据库

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\09\Ex09_331

实例说明

在使用数据库管理系统的过程中，会遇到将本地文本文件导入数据库表格的操作。这个过程可以分成两步：首先是读取本地文件到内存中；其次将内存中的数据按行和列分割，插入数据库中。本实例也实现了类似的功能：在树结构中选择要插入数据的表格，在表格结构中显示插入数据的结果。实例运行效果如图9.2所示。

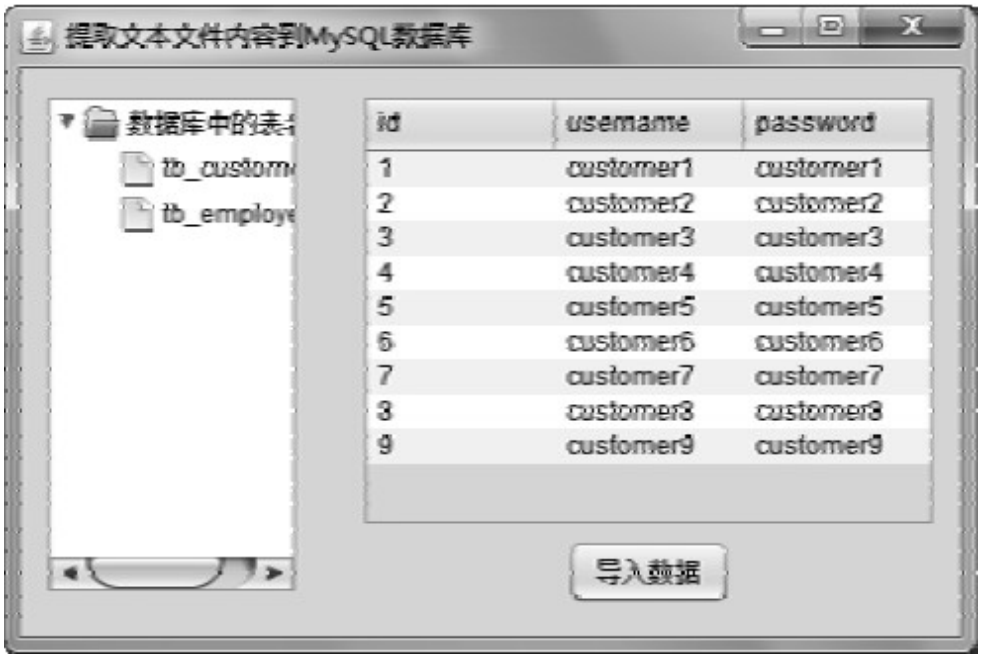


图9.2 实例运行效果

技术要点

在读取本地文本文件的过程中，可以使用BufferedReader类作为缓冲，以提高效率。本实例使用的方法如表9.3所示。

表9.3 BufferedReader类的常用方法

方 法 名	作 用
BufferedReader(Reader in)	创建一个使用默认大小输入缓冲区的缓冲字符输入流
close()	关闭该流并且释放与之关联的所有资源
readLine()	读取一个文本行

实现过程

（1）继承JFrame编写一个窗体类，名称为DataInputFrame。

（2）设计DataInputFrame窗体类时用到的控件及说明如表9.4所示。

表9.4 窗体的控件及说明

控 件 类 型	控 件 命 名	控 件 用 途
JTree	tree	显示数据库中所有表的名称
JTable	table	显示用户选择的表的信息
JButton	button	实现向数据库中保存数据的功能

（3）编写监听单击按钮事件的方法do_button_actionPerformed()，它实现了读入文本文件中的数据并将其保存到数据库中的功能。代码如下：

```
    protected void do_button_actionPerformed(ActionEvent
arg0) {
      JFileChooser fileChooser = new JFileChooser();
      fileChooser.setFileFilter(new
  FileNameExtensionFilter("文本文件", "txt"));
      fileChooser.setFileSelectionMode(JFileChooser.FILES_ONL
  Y);
      fileChooser.setMultiSelectionEnabled(false);
      int result = fileChooser.showOpenDialog(tree);
      if (result == JFileChooser.APPROVE_OPTION) {
        File file=
    fileChooser.getSelectedFile();                          //
```

获得用户选择的文件

```
    FileReader fileReader = null;
    BufferedReader bufferedReader = null;
    try {
      fileReader=new
  FileReader(file);                          //利用用户选择
  的文件创建FileReader对象
      bufferedReader=new
  BufferedReader(fileReader);              //创建
  BufferedReader对象
      //利用StringBuilder对象保存用户选择的文件中的数据
      StringBuilder builder = new StringBuilder();
      String temp = null;
      while ((temp=bufferedReader.readLine()) !=null)
  {                  //读入用户选择的文件
        builder.append(temp);
            //保存读入的一行数据
        builder.append("\n");
            //利用换行符分隔读入的各行
      }
      String[]
  rows=builder.toString().split("\n");
      //利用换行符分割各行
      if (tableName==null) {              //如果用户没有在
  树中选择要保存到的表名，则设置保存到第一个表
        tableName = (String)
    DBHelper.getTableNames().get(0);
```

```
        }
        for (String row : rows) {
          DBHelper.insertData(tableName,
      row.split("\t"));                    //利用工具类实现保存
      数据功能
        }
        JOptionPane.showMessageDialog(this, "数据导入成
    功");              //提示用户数据导入成功
      } catch (IOException e)
{                                          //捕获IO异常
        e.printStackTrace();
      }
      finally
{                                                  //释
放资源
        if (bufferedReader != null) {
          try {
            bufferedReader.close();
          } catch (IOException e) {
            e.printStackTrace();
          }
        }
        if (fileReader != null) {
          try {
            fileReader.close();
          } catch (IOException e) {
            e.printStackTrace();
```

```
            }
        }
    }
  }
}
```

举一反三

根据本实例，读者可以实现以下功能。

将本地文件导入数据库的表格中。

修改数据库中的文件。

实例332 将图片文件保存到 SQLServer数据库

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\09\Ex09_332

实例说明

在向数据库保存数据的过程中，可能会遇到如图片、声音等字节文件。通常将这类文件存入数据库中有两种方法，本实例演示第一种，即将图片转换成字节流插入表格对应的列中。利用这种方式可以实现对图片安全性的保护。实例运行效果如图9.3所示。

图9.3 实例运行效果

技术要点

FileInputStream 可以从文件系统中的某个文件中获得输入字节流。它用于读入诸如图像数据之类的原始字节流。本实例使用的方法如表9.5所示。

表9.5 FileInputStream类的常用方法

方 法 名	作 用
FileInputStream(File file)	通过打开一个到实际文件的连接来创建一个 FileInputStream,该文件通过文件系统中的 File 对象 file 指定
close()	关闭此文件输入流并释放与此流有关的所有系统资源

实现过程

（1）继承JFrame编写一个窗体类，名称为ImageSaveFrame。

（2）设计ImageSaveFrame窗体类时用到的控件及说明如表9.6所示。

表9.6 窗体的控件及说明

控 件 类 型	控 件 命 名	控 件 用 途
JTextField	pathTextField	显示用户选择的文件绝对路径
	nameTextField	显示用户输入的文件名，默认是用户选择的文件的名字

续表

控 件 类 型	控 件 命 名	控 件 用 途
JButton	chooseButton	实现让用户选择文件的功能
	saveButton	实现存储用户选择文件的功能

（3）编写存储图片的方法saveImage()，参数name是用户输入的图片的名字，type是用户选择的图片的类型，image是用户选择的图片对象。代码如下：

```
public static void saveImage(String name, String type,
File image) {
    FileInputStream in = null;
    Connection conn = null;
    PreparedStatement ps = null;
```

```
    try {
      in=new
  FileInputStream(image);                          //利用
  image参数创建FileInputStream对象
      Class.forName(DRIVER);
      //加载驱动程序
      conn=DriverManager.getConnection(URL,USERNAME,PASSWOR
  D);  //建立连接
      ps = conn.prepareStatement("insert into images values
  (?, ?, ?);");
      ps.setString(1,name);
        //保存图片名称
      ps.setString(2,
  type);                                           //保存图片类
  型
      ps.setBinaryStream(3, in, (int)
  image.length());                        //保存图片
      ps.executeUpdate();
        //执行保存
    } catch (Exception e)
{                                                  //捕获异常
      e.printStackTrace();
    } finally
{                                                        //释放资
源
      if (ps != null) {
        try {
```

```
            ps.close();
          } catch (SQLException e) {
            e.printStackTrace();
          }
        }
        if (conn != null) {
          try {
            conn.close();
          } catch (SQLException e) {
            e.printStackTrace();
          }
        }
        if (in != null) {
          try {
            in.close();
          } catch (IOException e) {
            e.printStackTrace();
          }
        }
      }
}
```

举一反三

根据本实例，读者可以实现以下功能。

将图片字节文件保存到数据库。

将声音文件保存到数据库。

实例333 显示数据库中的图片信息

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\09\Ex09_333

实例说明

在使用流的方式将图片存入数据库中之后，该怎样利用存储的数据来还原成原来的图片呢？本实例将向读者展示这个过程。实例运行效果如图9.4所示。

图9.4 实例运行效果

技术要点

将字节流还原成图片是一个复杂的过程，主要涉及的类（接口）和方法如下。

ResultSet表示数据库结果集的数据表，通常通过执行查询数据库的语句生成。它包含了很多常用的方法，其中获得字节流的方法如下：

InputStream getBinaryStream(int columnIndex)throws SQLException

参数说明

columnIndex：表示含有字节流的列号。利用它就可以获得数据库中指定列的字节流。

ImageIO包含一些用来查找ImageReader和ImageWriter以及执行简单编码和解码的静态便捷方法。它的静态方法read()可以将输入的字节流转换成BufferedImage对象。该方法的声明如下：

public static BufferedImage read(InputStream input)throws IOException

□ 参数说明

□ input：代表需要进行编码的字节流。

BufferedImage类描述具有可访问图像数据缓冲区的Image。使用该类的getSource()方法可以获得ImageProducer对象。该方法的声明如下：

public ImageProducer getSource()

□ 该方法返回生成该图像像素的对象。

Toolkit的子类被用于将各种控件绑定到特定本机工具包中实现。这里使用它的createImage()方法来生成需要的图像。该方法的声明如下：

public abstract Image createImage(ImageProducer producer)

□ 参数说明

producer：要使用的图像生成器。

实现过程

（1）继承JFrame编写一个窗体类，名称为ImageOpenFrame。

（2）设计ImageOpenFrame窗体类时用到的控件及说明如表9.7所示。

表9.7 窗体的控件及说明

控 件 类 型	控 件 命 名	控 件 用 途
JTable	table	显示表格中存储的图片信息
JButton	button	实现显示图片的功能

（3）编写getImage()方法，参数sql指明要显示数据库表格中的哪一行，columnLabel指明存储图片的列名。代码如下：

```
public static Image getImage(String sql, String columnLabel) {
    Image image = null;
    try {
      rs=getConnection().createStatement().executeQuery(sql);
                    //获得结果集
      rs.next();
            //将结果集游标指向第一条记录
      InputStream in = rs.getBinaryStream(columnLabel);
      BufferedImage bi = ImageIO.read(in);
      ImageProducer ip = bi.getSource();
      Toolkit tk = Toolkit.getDefaultToolkit();
      image = tk.createImage(ip);
    } catch (SQLException e) {
      e.printStackTrace();
    } catch (IOException e) {
      e.printStackTrace();
    }
    return image;
}
```

举一反三

根据本实例，读者可以开发以下程序。

显示数据表中的图片信息。

显示数据库中的文字信息。

实例334 在数据库中建立磁盘文件索引

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\09\Ex09_334

实例说明

在用户使用电脑的过程中，一定用过查找功能，根据输入的关键字不同，操作系统会在用户指定的文件夹下进行相应查找，并返回查找的结果。本实例利用了File类将用户指定的文件夹下（包括子文件夹）所有文件的绝对路径存储到数据库中，再使用 MySQL 的索引功能，方便用户对指定文件进行查询。实例运行效果如图9.5所示。

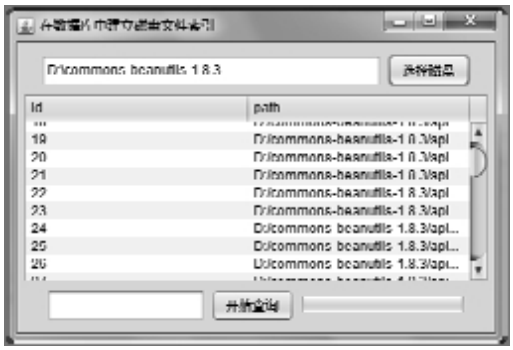


图9.5 实例运行效果

技术要点

File类包括了与文件管理系统相关的大量方法。本实例用到的方法如表9.8所示。

表9.8 File类的常用方法

方 法 名	作 用
File(String pathname)	通过将给定路径名字符串转换为抽象路径名来创建一个新 File 实例
getAbsoluteFile()	返回此抽象路径名的绝对路径名形式
getAbsolutePath()	以字符串的形式返回该 File 对象的绝对路径
isDirectory()	测试该 File 对象是不是一个文件夹，是则返回 true
listFiles()	如果给定的 File 对象是一个文件夹，则将其转换成 File 数组，数组中包括该文件夹中的文件和子文件夹；否则抛出 NullPointerException

实现过程

（1）继承JFrame编写一个窗体类，名称为IndexFrame。

（2）设计IndexFrame窗体类时用到的控件及说明如表9.9所示。

表9.9 窗体的控件及说明

控 件 类 型	控 件 命 名	控 件 用 途
JTextField	chooseTextField	显示用户选择的文件路径
	keyTextField	显示用户输入的文件名
JButton	chooseButton	实现让用户选择文件路径的功能
	searchButton	实现查询功能
JTable	table	显示数据库中保存的索引
JProgressBar	progressBar	提示用户正在创建索引

（3）创建工具方法getFilePath()，参数list用来保存迭代出来的路径，rootFile用来指明迭代开始的文件夹。代码如下：

```
    private static List<String> getFilePath(List<String>
list, File rootFile) {
        File[] files=
    rootFile.listFiles();                          //列出用户
    选择的文件夹下的所有文件（夹）
        if
    (files==null)                                  //如果
    是空文件夹，则直接返回
          return list;
        for (File file : files)
    {                                    //遍历用户选择的文件夹
    下所有的文件（夹）
          if (file.isDirectory())
    {                                  //如果是一个文件夹，则进
    行迭代
            getFilePath(list, file);
          } else {
```

```
            list.add(file.getAbsolutePath().replace("\\",
        "/"));          //否则保存路径
          }
        }
        return list;
    }
```

举一反三

根据本实例，读者可以实现以下功能。

实现文件的查找功能。

创建一个新的文件。

9.2 操作磁盘文件夹

实例335 以树结构显示文件路径

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\09\Ex09_335

实例说明

Java的JTree树控件用于显示多层次结构的数据，这类似于文件夹的上下层关系，而且各个操作系统也都因为这个相似之处，采用树控件来显示文件夹的层次结构。本实例也结合了文件夹数据与树控件来显示计算机中的文件与文件夹信息。实例运行效果如图9.6所示。

技术要点

展开树的指定节点，JTree树控件中包含了一个特殊的方法，利用该方法可以展开指定的树节点。该方法的声明如下：

```
public void expandPath(TreePath path)
```

该方法确保指定路径标识的节点展开，并且可查看。如果路径中的最后一项是叶节点，则此方法无效。

参数说明

path：标识节点的TreePath。

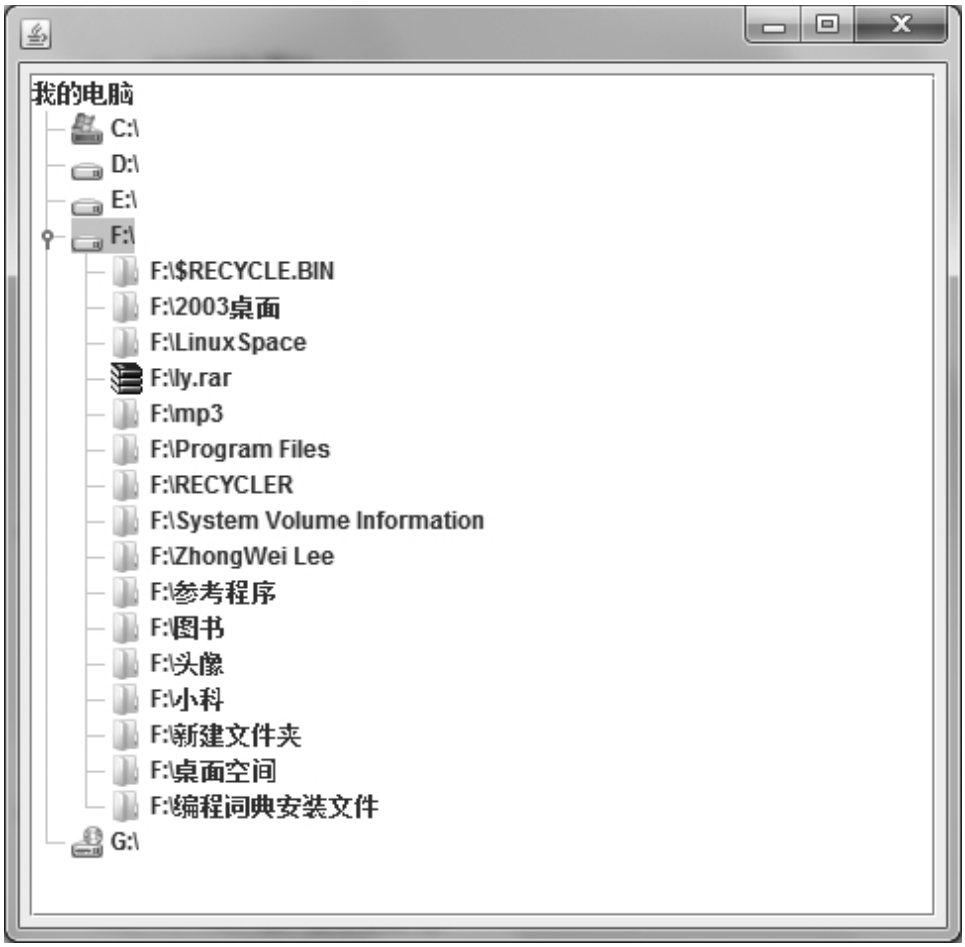


图9.6 实例运行效果

在本实例中使用如下代码来展开根节点。

```
tree.expandPath(new
TreePath(rootNode));                                    //
//展开根节点
```

这个方法接收一个TreePath类型的参数，其作用是指明树节点的路径。创建该类型对象的常用构造方法有两个，分别介绍如下：

TreePath(Object singlePath)

该方法构造仅包含单个元素的TreePath，这个单元元素可以是节点对象。

TreePath(Object[] path)

该方法根据 Objects 的数组构造路径，并根据树的数据模型的返回情况，唯一地标识树的根到指定节点的路径。

例如，本实例中是这样创建根节点的TreePath对象的。方法如下：

```
tree.expandPath(new
TreePath(rootNode));                                    /
/展开根节点
```

实现过程

（1）继承JFrame编写一个窗体类，名称为DiskTree。

（2）创建 JTree 树控件并布局到窗体的中间位置。这个树控件在窗体激活时要读取计算机根节点的数据，即一个磁盘名称。

（3）编写窗体激活事件监听器调用的do_this_windowActivated()方法，该方法是在类中自定义的，主要用途是获取计算机中的磁盘列表，并将其添加到树控件中。关键代码如下：

```
protected void do_this_windowActivated(WindowEvent e) {
  File[]disks=File.listRoots();
            //获取磁盘列表
  for (File file :disks)
{                                                    //遍历列表
    //使用文件对象创建树节点
    DefaultMutableTreeNode node = new
DefaultMutableTreeNode(file);
    rootNode.add(node);
          //添加节点到树控件的根节点
  }
```

```
    tree.expandPath(new
TreePath(rootNode));                                    //展开根
节点
  }
```

（4）在树节点的选择事件监听器中调用do_tree_valueChanged()方法，这个方法将获取选择的树节点，并读取节点中封装的文件对象。如果该文件对象是文件夹，则遍历文件夹中的文件列表，并将其封装到树节点对象中作为树控件的显示内容。关键代码如下：

```
  protected void do_tree_valueChanged(TreeSelectionEvent e)
{
    TreePath path=
e.getPath();                                            //获取树选
择路径
    //获取选择路径中的节点
    DefaultMutableTreeNode node = (DefaultMutableTreeNode)
path.getLastPathComponent();
    //获取节点中的用户对象
    Object userObject = node.getUserObject();
    if (!(userObject instanceof File)) {
      return;
    }
    File folder= (File)
userObject;                                             //把用户对象转
换为文件对象
    if
(!folder.isDirectory())
    //过滤非文件夹的选择操作
```

```
        return;
      File[] files=
folder.listFiles();                                    //获
取文件夹中的文件列表
      for (File file : files)
{                                                      //遍历文件列表数
组
      //使用文件做用户对象创建节点
        node.add(new DefaultMutableTreeNode(file));
      }
  }
```

举一反三

根据本实例，读者可以实现以下功能。

用JTable和JTree控件来显示表结构的数据。

用JTable和JTree控件来显示树结构的数据。

实例336 窗体动态加载磁盘文件

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\09\Ex09_336

实例说明

在使用图形界面操作系统时，当打开一个文件夹系统时会自动列出该文件夹下的所有文件及子文件夹。本实例实现了类似的功能：首先让用户选择一个文件夹，程序会动态列出该文件夹下的所有文件；如果该文件是隐藏文件，就在属性栏中显示“隐藏文件”。实例运行效果如图9.7所示。

图9.7 实例运行效果

技术要点

Java API 中的File 类提供了很多与文件属性相关的方法。本实例使用到的方法如表9.10所示。

表9.10 File类的常用方法

方 法 名	作 用
getAbsolutePath()	以字符串的形式返回该 File 对象的绝对路径
isFile()	测试该 File 对象是否是一个文件，是则返回 true
isHidden()	测试该 File 对象是否是一个隐藏文件，是则返回 true
listFiles()	如果给定的 File 对象是一个文件夹，则将其转换成 File 数组，数组中包括该文件夹中的文件和子文件夹；否则抛出 NullPointerException

实现过程

（1）继承JFrame编写一个窗体类，名称为FileListFrame。

（2）设计FileListFrame窗体类时用到的控件及说明如表9.11所示。

表9.11 窗体的控件及说明

控 件 类 型	控 件 命 名	控 件 用 途
JTextField	chooseTextField	显示用户选择的文件夹的绝对路径
JButton	chooseButton	实现动态加载磁盘文件的功能
JTable	table	显示用户选择的文件夹中文件的信息

（3）编写按钮激活事件监听器调用的do_chooseButton_actionPerformed()方法，该方法是在类中自定义的，主要用途是实现动态加载磁盘文件的功能。关键代码如下：

```
    protected void
do_chooseButton_actionPerformed(ActionEvent arg0) {
        JFileChooser fileChooser = new JFileChooser();
        fileChooser.setFileSelectionMode(JFileChooser.DIRECTORI
  ES_ONLY);
        fileChooser.setMultiSelectionEnabled(false);
        int result = fileChooser.showOpenDialog(this);
        if (result == JFileChooser.APPROVE_OPTION) {
          chooseFile=
    fileChooser.getSelectedFile();
    //获得用户选择的文件夹
          chooseTextField.setText(chooseFile.getAbsolutePath())
    ;                    //显示用户选择的文件夹
          progressBar.setIndeterminate(true);
                    //设置滚动条开始滚动
          finalFile[] subFiles=
    chooseFile.listFiles();                    //获得用户选择的文
    件夹中的所有文件（夹）
          final DefaultTableModel model = (DefaultTableModel)
    table.getModel();
```

```
    model.setRowCount(0);
          //清空表格
    new Thread()
{                                                    //开始新
的线程
      public void run() {
        for (int i=0; i< subFiles.length; i++)
    {                        //遍历用户选择的文件夹
          if (subFiles[i].isFile())
      {                                  //判断是否是一个文件
            Object[] property = new Object[3];
            property[0]=
        i+1;                                  //保存序号
            property[1]=
        subFiles[i].getName();                      //保存文
        件名
            property[2] ="";
            if (subFiles[i].isHidden())
        {                             //判断是否是一个隐藏文件
              property[2] ="隐藏文件";
            }
            model.addRow(property);
            //向表格中添加记录
            table.setModel(model);
              //更新表格
          }
          try {
```

```
                Thread.sleep(100);
                //线程休眠100毫秒实现动态加载
              } catch (InterruptedException e) {
                e.printStackTrace();
              }
            }
            progressBar.setIndeterminate(false);
                      //停止进度条滚动
          };
        }.start();
      }
    }
```

举一反三

根据本实例，读者可以实现以下功能。

测试文件是否是隐藏文件。

实例337 删除文件夹中所有文件

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\09\Ex09_337

实例说明

删除文件是对文件的常用操作之一，操作系统可以根据用户的选择，删除文件或者删除文件夹。本实例可以根据用户指定的文件夹删除该文件夹中的所有文件，包括子文件夹和隐藏文件，但是保留用户选择的文件夹。实例运行效果如图9.8所示。

图9.8 实例运行效果

技术要点

Java SE API 中的File 类提供了很多与文件管理相关的方法。本实例使用到的方法如表9.12所示。

表9.12 File类的常用方法

方 法 名	作 用
delete()	如果该 File 对象是一个文件或空文件夹就将其删除
getAbsolutePath()	以字符串的形式返回该 File 对象的绝对路径
isFile()	测试该 File 对象是否是一个文件，是则返回 true
isHidden()	测试该 File 对象是否是一个隐藏文件，是则返回 true
listFiles()	如果给定的 File 对象是一个文件夹，则将其转换成 File 数组，数组中包括该文件夹中的文件和子文件夹；否则抛出 NullPointerException

实现过程

（1）继承JFrame编写一个窗体类，名称为FileDeleteFrame。

（2）设计FileDeleteFrame窗体类时用到的控件及说明如表9.13所示。

表9.13 窗体的控件及说明

控 件 类 型	控 件 命 名	控 件 用 途
JTextField	chooseTextField	显示用户选择的文件夹的绝对路径
JButton	chooseButton	实现让用户选择要删除的文件夹
	deleteButton	实现删除文件的功能
JTextArea	resultTextArea	显示被用户删除的文件

（3）编写方法deleteDirectories()来实现删除文件夹及其中内容的功能，参数rootFile代表用户想删除的文件夹。代码如下：

```
public static void deleteDirectories(File rootFile) {
```

```
    if (rootFile.isFile()) {
      rootFile.delete();
          //如果给定的File对象是文件就直接删除
    } else
{                                                    //如果
是一个文件夹就将其转换成File数组
      File[] files = rootFile.listFiles();
      for (File file : files) {
        deleteDirectories(file);
            //如果不是空文件夹就迭代deleteDirectories()方法
      }
      rootFile.delete();
          //如果是空文件夹就直接删除
    }
  }
```

（4）编写方法deleteFiles()来实现删除文件夹下的所有内容，但保留给定文件夹的功能，参数rootFile代表用户想删除的文件夹。关键代码如下：

```
  public static void deleteFiles(File rootFile) {
    if (rootFile.listFiles().length==0)
{                                  //如果用户给定的是空文件夹
就退出方法
      return;
    } else {
      File[] files=
  rootFile.listFiles();                                  //将非
  空文件夹转换成File数组
```

```
    for (File file : files) {
      if (file.isFile()) {
        file.delete();
        //删除指定文件夹下的所有文件
      } else {
        if (file.listFiles().length == 0) {
          file.delete();
        //删除指定文件夹下的所有空文件夹
        } else {
          deleteDirectories(file);
          //删除指定文件夹下的所有非空文件夹
        }
      }
    }
  }
}
```

举一反三

根据本实例，读者可以实现以下功能。

用File 类修改文件。

用File类删除文件。

删除文件夹操作。

实例338 创建磁盘索引文件

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\09\Ex09_338

实例说明

为了提高对磁盘文件的搜索效率，可以创建一个磁盘索引文件，将磁盘中所有文件的路径都保存到该文件中，当需要查找时，在该文件中查找即可。实例运行效果如图9.9所示。

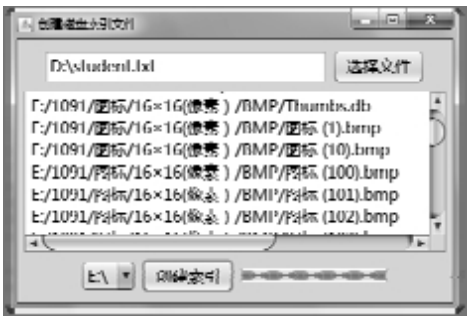


图9.9 实例运行效果

技术要点

磁盘索引文件继承JFrame。

实现过程

（1）继承JFrame编写一个窗体类，名称为IndexFileFrame。

（2）设计IndexFileFrame窗体类时用到的控件及说明如表9.14所示。

表9.14 窗体的控件及说明

控 件 类 型	控 件 命 名	控 件 用 途
JTextField	chooseTextField	显示用户选择的索引文件的绝对路径
JButton	chooseButton	实现让用户选择保存索引的文件
	createButton	实现创建索引文件的功能
JComboBox	comboBox	显示用户电脑上所有可用磁盘
JTextArea	resultTextArea	显示创建的索引
JProgressBar	progressBar	显示创建索引文件的进度

（3）编写监听单击按钮事件的方法do_createButton_actionPerformed()，它实现了创建索引并显示创建结果的功能。代码如下：

```
    protected void
do_createButton_actionPerformed(ActionEvent arg0) {
        if (chooseFile == null) {
```

```
        JOptionPane.showMessageDialog(this, "请选择保存索引的
    文件", null, JOptionPane.WARNING_MESSAGE);
        return;
      }
      String disc=
comboBox.getSelectedItem().toString();                    //
获得用户选择的磁盘
      comboBox.setSelectedItem(disc);
          //设置JComboBox显示用户选择的磁盘
      final List<String> list=new ArrayList<String>
();                    //用list保存索引
      final File rootFile=new
File(disc);                                  //利用用户选择的
磁盘创建File对象
      final StringBuilder sb=new
StringBuilder();                           //利用StringBuilder
对象保存写入的索引
      progressBar.setIndeterminate(true);
              //设置滚动条开始滚动
      new Thread() {                                        //在
一个新的线程中处理创建索引和写入索引的操作
        @Override
        public void run() {
          getFilePath(list,
      rootFile);                                  //获得磁盘上所
      有文件的路径
```

```
    Iterator<String> iterator=
list.iterator();                          //创建迭代器
    while (iterator.hasNext())
{                                  //遍历list
      sb.append(iterator.next());
      sb.append("\r\n");
      try {
        Thread.sleep(100);
      //线程休眠100毫秒
      } catch (InterruptedException e) {
        e.printStackTrace();
      }
      textArea.setText(sb.toString());
          //在文本域中显示文件路径
    }
    FileWriter fileWriter = null;
    try {
      fileWriter = new FileWriter(chooseFile);
      fileWriter.write(textArea.getText());
          //向用户选择的文本文件写入数据
      fileWriter.flush();
    } catch (IOException e) {
      e.printStackTrace();
    }
    //省略释放资源的代码
    progressBar.setIndeterminate(false);
          //停止进度条的滚动
```

```
            JOptionPane.showMessageDialog(null, "索引创建成
    功");          //提示用户索引创建成功
      };
    }.start();
  }
```

举一反三

根据本实例，读者可以实现以下功能。

磁盘索引文件的创建。

修改磁盘索引文件。

实例339 快速全盘查找文件

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\09\Ex09_339

实例说明

在磁盘上进行文件查找有两种方式。第一种是遍历整个磁盘，获得各个文件的路径。如果路径中含有用户指定的关键字，就保存该路径，最后将结果显示给用户。第二种是使用已经建立好的磁盘索引文件，直接在该文件中进行查找。显然第二种方法速度较快，本实例就是采用这种方法来实现对某一特定文件的查找功能。实例运行效果如图9.10所示。

图9.10 实例运行效果

技术要点

本实例主要是读取磁盘索引文件，再利用String类的contains()方法找出与用户输入的关键字匹配的结果。当且仅当此字符串包含指定的char值序列时，返回true。该方法的声明如下：

public boolean contains(CharSequence s)

□ 参数说明

s：要搜索的序列。

实现过程

（1）继承JFrame编写一个窗体类，名称为FileFindFrame。

（2）设计FileFindFrame窗体类时用到的控件及说明如表9.15所示。

表9.15 窗体的控件及说明

控 件 类 型	控 件 命 名	控 件 用 途
JTextField	chooseTextField	显示用户选择的索引文件的绝对路径
	searchTextField	获得用户输入的关键字
JButton	chooseButton	让用户选择保存索引的文件
	searchButton	实现查找功能
JTextArea	resultTextArea	显示查找的结果

（3）编写监听单击按钮事件的方法do_searchButton_actionPerformed()，它实现了让用户选择数据写入的文件和写入数据的功能。代码如下：

```
    protected void
do_searchButton_actionPerformed(ActionEvent arg0) {
      if (chooseFile == null) {
        JOptionPane.showMessageDialog(this, "请选择索引文件",
    null,JOptionPane.WARNING_MESSAGE);
        return;
      }
      String keyword=
  searchTextField.getText();                          //获得
  用户输入的关键字
      if (keyword.length() == 0) {
        JOptionPane.showMessageDialog(this, "请输入关键字",
    null,JOptionPane.WARNING_MESSAGE);
        return;
      }
      FileReader fileReader = null;
      BufferedReader bufferedReader = null;
      try {
        fileReader=new
    FileReader(chooseFile);                           //利用用户
    选择的文件创建FileReader对象
        bufferedReader = new BufferedReader(fileReader);
        StringBuilder builder=new
    StringBuilder();                        //利用StringBuilder
```

对象保存索引

```
    String temp = null;
    while ((temp=bufferedReader.readLine()) !=null)
{                   //读入文本文件
      builder.append(temp);
      builder.append("\n");
      //在每一行的末尾添加一个分隔符
    }
    String[]
rows=builder.toString().split("\n");
  //将索引按换行符分割
    resultTextArea.setText("");
        //清空文本域
    for (String row : rows)
{                                          //遍历读入的文本文件
      if (row.contains(keyword))
  {                           //判断读入的文本文件是否包含指
  定的关键字
        resultTextArea.append(row+
    "\n");                        //返回结果
      }
    }
    if (resultTextArea.getText().length() == 0) {
      JOptionPane.showMessageDialog(this, "没有找到您需要
  的文件", null,JOptionPane.WARNING_MESSAGE);
      return;
    }
```

```
    } catch (IOException e) {
      e.printStackTrace();
    } finally
{                                                    //释放资
源
      if (bufferedReader != null) {
        try {
          bufferedReader.close();
        } catch (IOException e) {
          e.printStackTrace();
        }
      }
      if (fileReader != null) {
        try {
          fileReader.close();
        } catch (IOException e) {
          e.printStackTrace();
        }
      }
    }
  }
```

举一反三

根据本实例，读者可以实现以下功能。

利用索引文件查找用户指定的文件。

显示查找结果。

实例340 获取磁盘所有文本文件

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\09\Ex09_340

实例说明

本实例和实例339类似，都是利用已经创建好的索引文件进行查找。不过由用户指定要查找的文件变成了查找所有文本文件。实例运行效果如图9.11所示。

图9.11 实例运行效果

技术要点

本实例主要是读取磁盘索引文件，再利用String类的endsWith()方法找出所有以.txt结尾的路径。该方法的声明如下：

public boolean endsWith(String suffix)

□ 该方法用来测试该字符串是否以指定的后缀结束。

□ 参数说明

suffix：指定的后缀。

实现过程

（1）继承JFrame编写一个窗体类，名称为TextFileFindFrame。

（2）设计TextFileFindFrame窗体类时用到的控件及说明如表9.16所示。

表9.16 窗体的控件及说明

控 件 类 型	控 件 命 名	控 件 用 途
JTextField	chooseTextField	显示用户选择的索引文件的绝对路径
JButton	chooseButton	实现让用户选择保存索引的文件
	searchButton	实现查找功能
JTextArea	resultTextArea	显示查询的结果

（3）编写按钮单击事件监听器调用的do_searchButton_actionPerformed()方法，该方法是在类中自定义的，主要用途是根据用户指定的索引文件查找文本文件，如果有就在文本域中显示查询结果，如果没有就显示“没有找到您要的文件”。代码如下：

```
    protected void
do_searchButton_actionPerformed(ActionEvent arg0) {
      if (chooseFile == null) {
        JOptionPane.showMessageDialog(this, "请选择索引文件",
    null,JOptionPane.WARNING_MESSAGE);
        return;
      }
      String keyword=
  ".txt";                                    //将关键字指定为文
  本文件的后缀
      FileReader fileReader = null;
      BufferedReader bufferedReader = null;
      try {
        fileReader=new
    FileReader(chooseFile);                  //利用用户选择的
    文件创建FileReader对象
        bufferedReader = new BufferedReader(fileReader);
```

```
    StringBuilder builder=new
StringBuilder();                    //利用StringBuilder对象保
存索引
    String temp = null;
    while ((temp=bufferedReader.readLine()) !=null)
{           //读入文本文件
      builder.append(temp);
      builder.append("\n");
  //在每一行的末尾添加一个分隔符
    }
    String[]
rows=builder.toString().split("\n");                    //将
索引按换行符分割
    resultTextArea.setText("");
    //清空文本域
    for (String row : rows)
{                                   //遍历读入的文本文件
      if (row.endsWith(keyword))
  {                           //判断读入的文本文件是否包含指
  定的关键字
        resultTextArea.append(row+
    "\n");                          //返回结果
      }
    }
    if (resultTextArea.getText().length() == 0) {
      JOptionPane.showMessageDialog(this, "没有找到您需要
  的文件", null,JOptionPane.WARNING_MESSAGE);
```

```
            return;
        }
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (bufferedReader != null) {
            try {
                bufferedReader.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        if (fileReader != null) {
            try {
                fileReader.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
```

举一反三

根据本实例，读者可以实现以下功能。

利用索引文件查找特定类型的文件。

显示查找结果。

实例341 将某文件夹中的文件进行分类存储

这是一个可以用来提高基础性能的实例

实例位置：光盘\mingrisoft\09\Ex09_341

实例说明

随着信息技术的高速发展，人们在计算机中存储的文件越来越多，有时由于时间的问题，没有注意文件的存放方式，这样长期下去，计算机中的文件会显得比较凌乱。可以自己开发一个小程序，实现文件的分类存储，将具有相同格式的文件存储在同一个文件夹下，以方便查询。实例运行效果如图9.12所示。

图9.12 实例运行效果

技术要点

本实例实现文件分类存储的关键是获取某文件夹下的文件，并通过字符串截取的方式提取文件的格式，再根据获取的文件格式创建文件夹。提取文件格式使用的是String类的substring()方法，该方法可在指定的字符串中截取子字符串，语法格式如下：

substring(int beginIndex,int endIndex)

参数说明

● beginIndex：要截取字符串的开始索引位置，包括该索引位置处的字符。

● endIndex：要截取字符串的结束索引，不包括该索引位置处的字符。

实现过程

（1）继承JFrame编写一个窗体类，名称为SortFrame。

（2）设计SortFrame窗体类时用到的控件及说明如表9.17所示。

表9.17 窗体的控件及说明

控 件 类 型	控 件 命 名	控 件 用 途
JTextField	pathTextField	显示要分类的文件夹地址
JButton	choicButton	为用户提供选择要分类的文件夹的按钮
	sortButton	显示“确定分类”按钮

（3）创建工具类SortUtil，该类定义了获取文件夹下所有文件的方法getList()和复制文件的方法 copyFile() ，读者可参考光盘中的源程序，这里不再赘述。还定义了新建文件夹方法createFolder()，该方法有一个 String 类型的参数，用于定义新建文件夹的保存地址。具体代码如下：

```
public void createFolder(String strPath) {
  try {
    FilemyFilePath=new
File(strPath);                          //根据文件地址创建
File对象
    if (!myFilePath.exists())
{                                  //如果指定的File对象不存在
      myFilePath.mkdir();                                    //
  创建目录
    }
  } catch (Exception e) {
    System.out.println("新建文件夹操作出错");
    e.printStackTrace();
  }
}
```

（4）当用户单击“确定分类”按钮后，系统会调用相应方法，实现文件分类存储。具体代码如下：

```
    protected void do_sortButton_actionPerformed(ActionEvent arg0) {
        SortUtil sortUtil = new SortUtil();
        List list=
    sortUtil.getList(pathTextField.getText());                    //
    获取用户选择文件夹中的所有文件集合
        for(int i=0;i<list.size();i++)
    {                                    //循环遍历该文件集合
          String strFile = list.get(i).toString();
          int index = strFile.lastIndexOf(".");
          if(index !=-1){
            //对文件夹进行截取，获取文件扩展名
            String strN =
        strFile.substring(index+1,strFile.length());
            int ind = strFile.lastIndexOf("\\");
            String strFileName = strFile.substring(ind, index);
            //调用创建文件夹的方法，新建文件夹
            sortUtil.createFolder(pathTextField.getText()+"\\"+
        "分类");
            sortUtil.createFolder(pathTextField.getText()+"\\"+
        "分类"+"\\"+strN);
            if(strFile.endsWith(strN)){
            //将文件集合中与文件夹名称相同的文件复制到相应的文
        件夹中
```

```
            sortUtil.copyFile(strFile,pathTextField.getText()
        +"\\"+"分
        类"+"\\"+strN+"\\"+strFileName+strFile.substring(inde
        x,strFile.length()));
          }
        }
      }
      JOptionPane.showMessageDialog(getContentPane(),
            //给出用户分类完成提示框
        "文件分类成功！", "信息提示框",
JOptionPane.WARNING_MESSAGE);
}
```

举一反三

根据本实例，读者可以实现以下功能。

实现用户信息提示对话框。

9.3 文件的读取与写入

实例342 键盘录入内容保存到文本文件

这是一个可以提高基础性能的实例

实例位置：光盘\mingrisoft\09\Ex09_342

实例说明

本节为读者介绍文件读取和写入相关的实例。本实例实现的是将键盘录入的内容保存到文本文件中，通过本实例相信读者会对文件的写入有更多的了解。实例运行效果如图9.13所示。

运行本实例后，会将用户输入的内容写入到与项目同一目录的Example8.txt文本文件中，文本文件中的内容如图9.14所示。

图9.13 实例运行效果

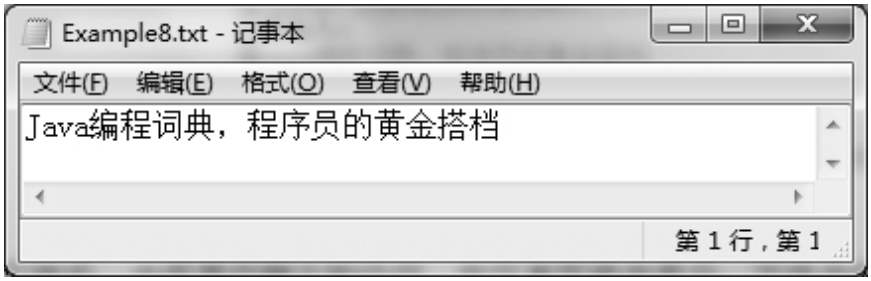


图9.14 文本文件中的内容

技术要点

本实例首先通过文件输入流类InputStreamReader实现读取用户写入的内容，再通过文件输出流类FileWriter，将读取的内容写入磁盘

文件中。InputStreamReader类提供了read()方法可实现数据的读取，该方法有两种重载形式，分别介绍如下。

□ read()：读取单个字符。

□ read(char[] cbuf, int offset, int length)：将字符读入数组中的某一部分。

参数说明

● cbuf：目标缓冲区。

● offset：开始存储字符的偏移量。

● length：要读取的最大字符数。

BufferedWriter类提供了write()方法，实现向文件中写数据，该方法提供了3种重载形式，分别介绍如下。

□ write(int c)：写入单个字符。

参数说明

c：指定要写入字符的int值。

□ write(char[] cbuf, int off, int len)：写入字符数组的某一部分。

参数说明

● cbuf：字符数组。

● off：开始读取字符处的偏移量。

● len：要写入的字符数。

□ write(String s, int off, int len)：写入字符串的某一部分。

参数说明

● s：要写入的字符串。

● off：开始读取字符处的偏移量。

● len：要写入的字符数。

实现过程

在项目中创建类Employ，在该类的主方法中实现将用于输入的数据写入文件中。具体代码如下：

```
public static void main(String args[]) {
  File file = new File("Example8.txt");
  try {
    if
(!file.exists())
  //如果文件不存在
      file.createNewFile();
      //创建新文件
    InputStreamReader isr=new
InputStreamReader(System.in);          //定义输入流对象
    BufferedReader br = new BufferedReader(isr);
    System.out.println("请输入：");
    String
str=br.readLine();                                    //读
取用户输入的信息
    System.out.println("您输入的内容是："+ str);
    FileWriter fos=new FileWriter(file,
true);                        //创建文件输出流
    BufferedWriter bw = new BufferedWriter(fos);
    bw.write(str);
      //向文件写入信息
    br.close();
    bw.close();
  } catch (IOException e) {
    e.printStackTrace();
```

```
    }
}
```

举一反三

根据本实例，读者可以实现以下功能。

读取文本文件内容。

对文本文件内容的格式转换。

实例343 将数组写入文件中并逆序输出

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\09\Ex09_343

实例说明

实现将一个数组写入文件中是一种很常用的形式，本实例实现将数组顺序写入文件中并实现逆序输出。实例运行效果如图9.15所示。

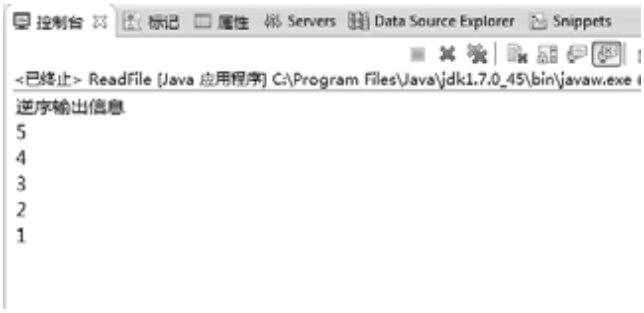


图9.15 实例运行效果

技术要点

本实例使用RandomAccessFile类实现了数据的读取和写入。此类的实例支持对随机访问文件的读取和写入。该类的构造方法介绍如下。

RandomAccessFile(File file, String mode)：创建从中读取和向其中写入（可选）的随机访问文件流，该文件由File参数指定。

参数说明

● file：文件对象。

● mode：访问模式。可选项为“r”，以只读方式打开。“rw”打开以便读取和写入，如果该文件尚不存在，则尝试创建该文件。“rws”打开以便读取和写入，对于“rw”，还要求对文件的内容或元数据的每个更新都同步写入底层存储设备。“rwd”打开以便读取和写入，对于“rw”，还要求对文件内容的每个更新都同步写入底层存储设备。

RandomAccessFile(String name, String mode)：创建从中读取和向其中写入（可选）的随机访问文件流，该文件具有指定名称。

参数说明

● name：取决于系统的文件名。

● mode：访问模式。

实现过程

在项目中创建类ReadFile，用于实现数组的写入和反向读取。该类主方法中的代码如下：

```
public static void main(String args[]){
  int bytes[]=
{1,2,3,4,5};                                    //定义写入
文件的数组
  try {
    //创建RandomAccessFile类的对象
    File file = new File("Example9.txt");
    if(!file.exists())
{                                               //判断该文件是否存
在
      file.createNewFile();
  //新建文件
```

```
    }
    RandomAccessFile raf=new
RandomAccessFile(file,"rw");      //定义RandomAccessFile
对象
    for(int i=0;i<bytes.length;i++)
{                              //循环遍历数组
      raf.writeInt(bytes[i]);
    //将数组写入文件
    }
    System.out.println("逆序输出信息");
    for(int i=bytes.length-1;i>=0;i--)
{                             //反向遍历数组
      raf.seek(i*4);                                    /
  /int型数据占4个字节
      System.out.println(+raf.readInt());
    }
    raf.close();
  //关闭流
  } catch (Exception e) {
    e.printStackTrace();
  }
}
```

举一反三

根据本实例，读者可以实现以下功能。

顺序输出文件中的内容。

将文件中的内容排序后输出。

实例344 利用StringBuffer避免文件的多次写入

这是一个可以启发思维的实例

实例位置：光盘\mingrisoft\09\Ex09_344

实例说明

为了避免多次重复地向磁盘文件写数据，可以使用字符串可变的字符串序列StringBuffer。将要写入文件的内容确定后，再使用FileOutputStream 类对象向文件写入信息。本实例实现的是将用户选择的“个人爱好”写入到磁盘文件。实例运行效果如图9.16所示。

图9.16 实例运行效果

技术要点

本实例向磁盘文件中写信息应用的是文件输出流FileOutputStream对象，通过该类的write()方法可实现向文件中写数据。该方法有多种重载形式。

□ 语法一：指定参数byte 型数组，向文件中写数据。语法如下：

write(byte[] b)

该语法将b.length个字节从指定byte数组写入此文件输出流中。

□ 语法二：将指定字节写入此文件输出流。语法如下：

write(int b)

□ 语法三：将指定byte 数组的一部分写入输出流。语法如下：

write(byte[] b , int off , int len)

参数说明

● b：要写入流的数据。

● off：数组中要写入流的开始索引位置。

● len：要写入流的字节数组数。

实现过程

（1）在项目中创建类UseStringBufferFrame，该类继承JFrame类，实现窗体类。

（2）向窗体中添加控件。实现窗体布局，该窗体中的主要控件及说明如表9.18所示。

表9.18 窗体中的主要控件及说明

控 件 类 型	控 件 命 名	控 件 用 途
JCheckBox	checkBox	显示“游泳”的复选框
JButton	saveButton	显示“写入文件”的按钮控件

（3）在“写入文件”按钮的单击事件中，实现将用户选择的“爱好”写入磁盘文件中。关键代码如下：

```
protected void do_button_actionPerformed(ActionEvent
arg0) {
    if (checkBox.isSelected())
{                                    //判断指定的复选框checkBox
是否被选中
      buffer.append(checkBox.getText()+ "
");                    //追加信息
    }
    …//省略了判断其他爱好信息的代码
    File file=new
File("C://w.txt");                              //根据指定文
```

件创建File对象

```
    try {
      FileOutputStream out=new
  FileOutputStream(file);          //创建FileOutputStream实
  例
      String
  str=buffer.toString();                              //将可
  变的字符序列转换为字符串对象
      out.write(str.getBytes());
      //向输出流中写数据
      JOptionPane.showMessageDialog(getContentPane(), "信息
  写入完成！", "信息提示
  框",JOptionPane.WARNING_MESSAGE);                  //给用户提
  供提示信息对话框
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
```

举一反三

根据本实例，读者可以实现以下功能。

向可变的字符串序列中插入字符。

从StringBuffer类对象中删除字符。

实例345 合并多个TXT文件

本实例可以提高工作效率

实例位置：光盘\mingrisoft\09\Ex09_345

实例说明

本实例实现的是将任意个TXT文件合并为一个文件。通过IO流可以实现文件的合并，当然可以对任意格式的文件进行合并，本实例以合并TXT文件为例，来介绍如何实现文件合并。实例运行效果如图9.17所示。

图9.17 实例运行效果

技术要点

本实例实现的文件合并主要通过 FileInputStream 类实现读取文件，通过 FileOutputStream类实现向文件中写入内容。在对文件读取的过程中，本实例应用了FileInputStream类的一个很重要的方法available()，来获取可读的有效字节数。该方法的语法格式如下：

```
int available()
```

可以通过FileInputStream类对象调用该方法。该方法的返回值是可以从输入流中读取的字节数。

实现过程

（1）在项目中创建类UniteFile，在该类中定义writeFiles()方法，该方法包含List对象与String类型对象，分别表示要进行合并的文件对象和合并后文件的保存地址。具体代码如下：

```
public void writeFiles(List<File> files, String fileName)
{
    try { //根据文件保存地址创建FileOutputStream对象
```

```
        fo = new FileOutputStream(fileName, true);
        for (int i=0; i< files.size(); i++)
    {                          //循环遍历要复制的文件集合
          File file=
      files.get(i);                                  //获取集合中的
      文件对象
          fi1=new
      FileInputStream(file);                          //创建
      FileInputStream对象
          b1=new
      byte[fi1.available()];                          //从流中获
      取字节数
          fi1.read(b1);                                        //
      读取数据
          fo.write(b1);                                        //
      向文件中写数据
        }
      } catch (Exception e) {
        e.printStackTrace();
      }
    }
```

（2）创建类UniteFrame，该类继承自JFrame类，实现窗体类。向窗体中添加控件，主要控件及说明如表9.19所示。

表9.19 窗体的主要控件及说明

控 件 类 型	控 件 命 名	控 件 用 途
JList	fileList	显示要合并文件的列表控件
JTextField	savePathtextField	显示用户选择的保存地址的文本框控件
JButton	submitButton	显示“确定合并”的按钮控件
	choiceButton	显示“选择合并文件”的按钮控件
	saveButton	显示“保存地址”的按钮控件

（3）当用户单击“选择合并文件”按钮时，系统会对用户选择的文件进行过滤，把 TXT文件添加到列表控件中。代码如下：

```
    protected void
do_choiceButton_actionPerformed(ActionEvent arg0) {
        java.awt.FileDialog fd=new
  FileDialog(this);                          //创建选择文件对话
  框
        fd.setVisible(true);
              //设置窗体为可视状态
        String filePath= fd.getDirectory()+
  fd.getFile();                          //获取用户选择的文件路径
        if (filePath.endsWith(".txt"))
  {                                          //判断用户选择的是否为
  txt文件
          list.addElement(fd.getDirectory()+
    fd.getFile());                       //将用户选择的文件添加到
    列表中
        //将用户选择的文件名添加到集合对象中
        listFile.add(new File((fd.getDirectory() +
    fd.getFile())));
      }
    }
```

（4）在“确定合并”按钮的单击事件中调用文件合并方法，将用户选择的文件进行合并。具体代码如下：

```
    protected void do_button_actionPerformed(ActionEvent
arg0) {
      UniteFile unitFile=new
  UniteFile();                                    //创建UniteFile对
  象
      unitFile.writeFiles(listFile,savePathtextField.getText(
  ));                    //调用合并文件的方法
      JOptionPane.showMessageDialog(getContentPane(),"文件合
  并成功！", "信息提示框", JOptionPane.WARNING_MESSAGE);
    }
```

举一反三

根据本实例，读者可以实现以下功能。

将多个文件合并成一个文件。

合并其他格式的文件。

实例346 对大文件实现分割处理

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\09\Ex09_346

实例说明

大的文件在传输时不太方便，为了便于携带，很多软件都提供了将大的文件进行分割的功能。这样就可以实现将一个较大的文件分割成若干个小的文件，方便携带。如果要实现该文件，可以通过相应的工具，将分割后的文件进行合并，这样不会耽误实现。本实例显示用

户选择的文件，并按指定的大小进行分割，实例运行效果如图9.18所示。

图9.18 实例运行效果

技术要点

实现本实例的关键是通过输入流读取要分割的文件，再分别从流中读取相应的字节数，将其写入以tem为后缀的文件中。通过FileInputStream类的read()方法可实现读取文件。

□ 语法一：以byte 数组为参数。表示从输入流中将数组长度个字节读取到byte 数组中。该方法的语法格式如下：

```
int read(byte[] b)
```

□ 语法二：从输入流中读取指定的字节数到数组中。其语法格式如下：

```
int read(byte[] b,int off.int len)
```

参数说明

- b：存储读取数据的字节数组。
- off：目标数组b 中的开始偏移量。
- len：读取的最大字节数。

实现过程

（1）创建CominuteFrame类，该类继承自JFrame类，实现窗体类。

（2）向该窗体中添加控件，主要控件及说明如表9.20所示。

表9.20 窗体的主要控件及说明

控 件 类 型	控 件 命 名	控 件 用 途
JTextField	sourceTextField	显示要进行分割的文件地址文本框
	sizeTextField	显示分割文件大小的文本框控件
JButton	sourceButton	显示“选择”按钮控件
	cominButton	显示“分割”按钮控件
	close	显示“退出”按钮控件

（3）编写工具类 CominuteUtil，在该类中定义实现文件分割的方法，该类包含两个 String类型的参数（分别用于指定分割文件的地址与分割后文件的保存地址）和一个int类型的参数（用于指定分割文件的大小）。代码如下：

```
    public void fenGe(File commFile, File untieFile, int
filesize) {
      FileInputStream fis = null;
      long size=1024 *
  1024;                                    //用来指定分割文件以
  MB为单位
      try {
        if (!untieFile.isDirectory())
    {                                  //如果要保存分割文件的地址不
    是路径
          untieFile.mkdirs();
      //创建该路径
        }
        size = size * filesize;
        long length= (int)
    commFile.length();                     //获取文件大小
```

```
    int num=(int)( length /
size);                                  //获取文件大小除以MB的
得数
    int yu= (int)(length%
size);                                  //获取文件大小与MB相除
的余数
    String newfengeFile=
commFile.getAbsolutePath();             //获取保存文件的完整
路径信息
    int fileNew = newfengeFile.lastIndexOf(".");
    String
strNew=newfengeFile.substring(fileNew,newfengeFile.length
());                                    //截取字符串
    fis=new
FileInputStream(commFile);                          //创建
FileInputStream类对象
    File[] fl=new
File[num+1];                            //创建文件数组
    long begin = 0;
    for (int i=0; i<num; i++)
{                                  //循环遍历数组
      fl[i]=new File(untieFile.getAbsolutePath()+ "\\"+
  (i+1)+ strNew+ ".tem");                           //指定分
  割后小文件的文件名
      if (!fl[i].isFile()) {
        fl[i].createNewFile();                          /
    /创建该文件
```

```
        }
        FileOutputStream fos = new FileOutputStream(fl[i]);
        byte[] bl = new byte[(int)size];
        fis.read(bl);                                          //
    读取分割后的小文件
        fos.write(bl);                                          /
    /写文件
        begin = begin + size * 1024 * 1024;
        fos.close();                                          //
    关闭流
      }
      if (yu !=0)
{                                              //文件大小与指定
文件分割大小相除的余数不为0
        fl[num]=new File(untieFile.getAbsolutePath()+ "\\"+
    (num+1)+ strNew+ ".tem");                  //指定文件分割后
    数组中的最后一个文件名
        if (!fl[num].isFile()) {
          fl[num].createNewFile();                              /
      /新建文件
        }
        FileOutputStream fyu = new
    FileOutputStream(fl[num]);
        byte[] byt = new byte[yu];
        fis.read(byt);
        fyu.write(byt);
        fyu.close();
```

```
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

举一反三

根据本实例，读者可以实现以下功能。

按指定大小分割文件。

将分割后的文件进行合并。

实例347 将分割后的文件重新合并

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\09\Ex09_347

实例说明

在实例346中，为大家介绍了如何实现将较大的文件进行分割，分割后的文件是不能运行的。如果想运行分割后的文件，就需要通过程序对相应的文件进行重新合并。本实例实现的是文件合并，实例运行效果如图9.19所示。

图9.19 实例运行效果

技术要点

本实例实现文件合并，仍然是通过文件字节输入/输出流。在进行文件合并时，需要将要进行合并的所有文件全部读取之后，再写入新文件中。

实现过程

（1）创建窗体类UniteFrame，该类继承自JFrame类。

（2）向该窗体中添加控件，主要控件及说明如表9.21所示。

表9.21 窗体的主要控件及说明

控 件 类 型	控 件 命 名	控 件 用 途
JList	fileList	显示要进行合并的文件列表控件
JButton	openButton	显示“打开”按钮控件
	uniteButton	显示“合并”按钮控件
	closeButton	显示“退出”按钮控件

（3）编写工具类 UniteUtil，在该类中定义文件合并方法heBing()，该方法中包含一个 File类型数组参数，用于指定要合并的文件数组；一个File对象，用于指定要合并后文件的保存地址；还有一个String类型参数，用于指定合并后文件的格式。该方法的具体代码如下：

```
public void heBing(File[] file, File cunDir, String hz) {
  try
{                                                    //指
定分割后文件的文件名
    File heBingFile = new File(cunDir.getAbsoluteFile()
+"\\UNTIE"+ hz);
    if (!heBingFile.isFile()) {
      heBingFile.createNewFile();
    }
    //创建FileOutputStream对象
```

```
    FileOutputStream fos = new
FileOutputStream(heBingFile);
    for (int i=0; i< file.length; i++)
{                                   //循环遍历要进行合并的文件数
组对象
      FileInputStream fis = new FileInputStream(file[i]);
      int len= (int)
  file[i].length();                                  //获取文件
  长度
      byte[] bRead = new byte[len];
      fis.read(bRead);
      //读取文件
      fos.write(bRead);
      //写入文件
      fis.close();
      //将流关闭
    }
    fos.close();
  } catch (Exception e) {
    e.printStackTrace();
  }
}
```

举一反三

根据本实例，读者可以实现以下功能。

合并分割的文件。

将合并后的文件重新写入新文件中。

实例348 在复制文件时使用进度条

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\09\Ex09_348

实例说明

在对大文件操作时，可能会需要些时间，此时为用户提供进度条提示是非常常见的一项功能，这样用户就可以了解操作文件需要的时间信息。本实例为大家介绍了在复制大的文件时使用的进度条提示，需要注意的是，只有在读取文件超过两秒时，才会显示进度条。实例运行效果如图9.20所示。

图9.20 实例运行效果

技术要点

本实例实现在读取文件时显示进度条使用的是ProgressMonitorInputStream类，该类提供了自动弹出进度窗口和事件机制。该类的构造方法语法如下：

ProgressMonitorInputStream(Component parentComponent, Object message, InputStream in)

参数说明

- parentComponent：触发被监视操作的组件。
- message：要在对话框中放置的描述性文本。
- in：要监视的输入流。

实现过程

（1）创建窗体类UserMonitorFrame，该类继承自JFrame类。

（2）在该类中添加控件，实现窗体布局，该窗体中的主要控件及说明如表9.22所示。

表9.22 窗体的主要控件及说明

控 件 类 型	控 件 命 名	控 件 用 途
JTextField	pathTextField	显示要复制的文件地址
	saveTextField	显示复制后文件的保存地址
JButton	pathButton	显示“选择文件”的按钮控件
	saveButton	显示“选择地址”的按钮控件
	copyButton	显示“确定复制”的按钮控件

（3）编写工具类 ProgressMonitorTest ，在该类中定义复制文件时显示进度条的方法useProgressMonitor()。该方法包含一个JFrame类型的参数，用于指定显示进度条所依赖的窗体；还有两个String类型的参数，分别用于指定要复制的文件以及复制后的文件保存地址。具体代码如下：

```
    public void useProgressMonitor(JFrame frame,String
copyPath, String newPath){
      try {
        File file=new
    File(copyPath);                                    //根据要复
    制的文件创建File对象
        File newFile=new
    File(newPath);                                     //根据复制后文
    件的保存地址创建File对象
        FileOutputStream fop=new
    FileOutputStream(newFile);              //创建
    FileOutputStream对象
        InputStream in = new FileInputStream(file);
```

```
    //读取文件，如果总耗时超过2秒，将会自动弹出一个进度监视窗口
    ProgressMonitorInputStream pm = new ProgressMonitorInputStream(frame, "文件读取中，请稍后...", in);
    int c = 0;
    byte[]bytes=new byte[1024];                    //定义byte数组
    while ((c=pm.read(bytes)) != -1)
{                                  //循环读取文件
      fop.write(bytes,0,c);
        //通过流写数据
    }
    fop.close();
      //关闭输出流
    pm.close();
    //关闭输入流
  } catch (Exception ex) {
    ex.printStackTrace();
  }
}
```

举一反三

根据本实例，读者可以实现以下功能。

进度条显示已完成的进度。

进度条加载完成后自动进入程序。

9.4 文件控制

实例349 利用StreamTokenizer统计文件的字符数

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\09\Ex09_349

实例说明

在常见的文本编辑器中，有一些提供了对字数的统计，如Word。但有些文本编辑器是不提供字数统计的，如记事本工具。为了方便使用记事本的用户可以快速地统计记事本文件中的字符数，可以开发专门的小工具，Java中的StreamTokenizer类可以实现该功能。本实例显示统计记事本文件的字符数，实例运行效果如图9.21所示。

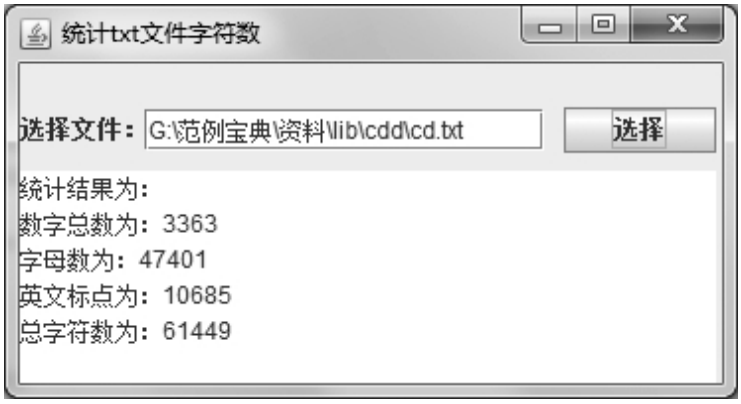


图9.21 实例运行效果

技术要点

java.io 包中的 StreamTokenizer 类可以获取输入流并将其解析为标记，可以通过该类的nextToken()方法读取下一个标记。该类的构造方法的语法如下：

StreamTokenizer(Reader r)

参数说明

r：提供输入流的Reader对象。

该类有几个非常重要的常量来标记读取文件的内容。这些常量与含义如表9.23所示。

表9.23 常量与含义

常 量 名	常 量 说 明
TT_EOF	表示读取到文件末尾
TT_WORD	指示读到一个文字标记的常量
TT_NUMBER	表示已读到一个数字标记的常量

实现过程

（1）继承JFrame编写一个窗体类，名称为StatFrame。

（2）设计StatFrame窗体类时用到的控件及说明如表9.24所示。

表9.24 窗体的控件及说明

控 件 类 型	控 件 命 名	控 件 用 途
JTextField	pathTextField	显示要统计的文本文件地址的文本框控件
JButton	chooseButton	为用户提供选择文件的按钮控件
JTextArea	resultTextArea	为用户提供显示统计结果的文本框控件

（3）定义工具类StatUtil，在该类中定义了statis()方法，获取读取文件的字符数组。该类有一个String类型的参数，用于指定文件地址，返回值为保存读取结果的int数组。关键代码如下：

```
public static int[] statis(String fileName) {
FileReader fileReader = null;
try {
  fileReader=new
FileReader(fileName);                          //创建
FileReader对象
   StreamTokenizer stokenizer=new StreamTokenizer(new
BufferedReader(fileReader));
        //创建StreamTokenizer对象
```

```
    stokenizer.ordinaryChar('\'');
        //将单引号当作普通字符
    stokenizer.ordinaryChar('\"');
        //将双引号当作普通字符
    stokenizer.ordinaryChar('/');
        //将“/”当作普通字符
    int[] length=new
int[4];                                    //定义保存计算结
果的int型数组
    String str;
    int
numberSum=0;                                    //定义保
存数字的变量
    int
symbolSum=0;                                     //定义
保存英文标点数的变量
    int wordSum = 0;
    int
sum=0;                                    //定义保
存总字符数的变量
    while (stokenizer.nextToken() !=StreamTokenizer.TT_EOF)
{       //如果没有读到文件的末尾
      switch (stokenizer.ttype)
  {                                //判断读取标记的类型
        case
    StreamTokenizer.TT_NUMBER:                    //如果用
    户读取的是一个数字标记
```

```
        str=String.valueOf(stokenizer.nval);
        //获取读取的数字值
        numberSum+= str.length();                    //
    计算读取的数字长度
        length[0]=numberSum;                         //设
    置数组中的元素
        break;                                       //退
    出语句
      case
StreamTokenizer.TT_WORD:                             //如果读
取的是文字标记
        str=
    stokenizer.sval;                                 //获取该
    标记
        wordSum+= str.length();                      //
    计算该文字的长度
        length[1] = wordSum;
        break;
      default:                                       //
如果读取的是其他标记
        str=String.valueOf((char)
    stokenizer.ttype);          //读取该标记
        symbolSum+= str.length();                    //
    计算该标记的长度
        length[2]= symbolSum;                        //
    设置int数组中的元素
    }
```

```
        }
        sum=
    symbolSum+numberSum+wordSum;                    //获取总
    字符数
        length[3] = sum;
        return length;
      } catch (Exception e) {
        e.printStackTrace();
      return null;
      }
    }
```

举一反三

根据本实例，读者可以实现以下功能。

通过StreamTokenizer类的ordinaryChar()方法将单引号和双引号当作普通字符处理。

统计文件的字符数。

实例350 在指定目录下搜索文件

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\09\Ex09_350

实例说明

Windows操作系统下的文件搜索功能大家都很熟悉，通过该功能，用户可以在指定的范围内搜索相关的文件。本实例模拟该功能开发一个小型的文件搜索工具，通过该工具可以将相关文件名称显示在窗体中，本实例支持星号“*”表示任意多个字符，支持问号“？”表示任意一个字符。实例运行效果如图9.22所示。

图9.22 实例运行效果

技术要点

本实例的实现首先要获取指定目录下的文件数组，再从数组中查询满足条件的文件。获取指定目录下的文件数组，可以通过File类的listFiles()方法，具体语法如下：

```
File[] listFiles()
```

该方法返回一个抽象路径名数组，表示此抽象路径名表示的目录中的文件。

实现过程

（1）继承JFrame编写一个窗体类，名称为SearchFrame。

（2）设计SearchFrame窗体类时用到的控件及说明如表9.25所示。

表9.25 窗体的控件及说明

控 件 类 型	控 件 命 名	控 件 用 途
JTextField	pathTextField	显示要进行搜索的文件地址文本框控件
	nameTextField	显示要搜索的文件名文本框控件
JComboBox	postfixComboBox	显示要搜索的文件名后缀的下拉列表控件
JList	resultList	显示搜索出满足条件的文件名的列表控件

（3）编写工具类FileSearch，在该类中定义findName()方法，用于查找匹配的文件。如果要查找的文件名与搜索模式匹配，则返回true；如果不匹配则返回false。具体代码如下：

```
    public static boolean findName(String pattern, String
str) {
        int
    patternLength=pattern.length();                    //获取参
    数字符串的长度
        int strLength = str.length();
        int strIndex = 0;
        char eachCh;
        for (int i=0; i<patternLength; i++)
    {                    //循环字符参数字符串中的每个字符
          eachCh=pattern.charAt(i);                          //获
      取字符串中每个索引位置的字符
        if (eachCh== '*') {                                  //如
      果这个字符是一个星号
          while (strIndex < strLength) {
            if (findName(pattern.substring(i+1),
        str.substring(strIndex))) {              //如果文件名与
        搜索模式匹配
              return true;
            }
            strIndex++;
          }
        } else if (eachCh== '?')
    {                           //如果包含问号
```

```
            strIndex++;
            if (strIndex> strLength) {                          //
        如果str中没有字符可以匹配“?”号
                return false;
            }
        } else {                                               //如果
    要寻找的是普通的文件
        //如果没有查找到匹配的文件
            if ((strIndex >= strLength) || (eachCh !=
        str.charAt(strIndex))) {
                return false;
            }
            strIndex++;
        }
    }
    return (strIndex == strLength);
}
```

（4）定义findFiles()方法，实现文件搜索功能。该方法有两个String类型的参数，分别用于指定要搜索目录的地址以及要搜索文件的名称。具体代码如下：

```
public static List findFiles(String baseDirName, String
targetFileName) {
    List fileList=new
  ArrayList();                                    //定义保存返回值
  的List对象
    File baseDir=new
  File(baseDirName);                              //根据参数创建
```

```
File对象
    if (!baseDir.exists() || !baseDir.isDirectory())
{                   //如果该File对象不存在或者不是一个目录
      return
  fileList;                                          //返回List
  对象
    }
    String tempName = null;
    File[]
files=baseDir.listFiles();                                    //
获取参数目录下的文件数组
    for (int i=0; i< files.length; i++)
{                           //循环遍历文件数组
      if (!files[i].isDirectory())
  {                                 //如果数组中的文件不是一个
  目录
        tempName=
    files[i].getName();                          //获取该数组的
    名称
        if (FileSearch.findName(targetFileName, tempName))
    {    //调用文件匹配方法
          fileList.add(files[i].getAbsoluteFile());
            //将指定的文件名添加到集合中
        }
      }
    }
    return fileList;
```

}

举一反三

根据本实例，读者可以实现以下功能。

在指定条件下搜索文件。

在指定范围内搜索文件。

实例351 文件锁定

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\09\Ex09_351

实例说明

在操作文件时，可能会遇到这样的问题。当打开一个文件时会遇到“该文件已经被另一个程序占用，打开失败”的问题。这是因为另一个程序正在编辑该文件，在这个过程中，不允许其他程序修改这个程序。这样就是文件的锁定。本实例通过 Java 程序实现将 C 盘的count.txt文件锁定1分钟，当对该文件进行编辑保存时，会出现图9.23所示的结果。

技术要点

本实例通过使用FileLock类文件进行文件锁定。文件锁定可阻止其他并发运行的程序获取重叠的独占锁定。文件锁定可以是独占，也可以是共享。该类的主要方法介绍如下。

图9.23 实例运行效果

isShared()方法：判断此锁定是否为共享的。

isValid()方法：判断此锁定是否有效。

release()方法：释放锁定。

实现过程

（1）在项目中创建类EncryptInput，在该类中定义锁定文件方法fileLock()，该方法有一个String类型的参数，用于指定锁定文件的地址。具体代码如下：

```
    public static void fileLock(String file) {
      FileOutputStream
  fous=null;                                    //创建
  FileOutputStream对象
      FileLock
  lock=null;                                        //创建
  FileLock对象
      try {
        fous=new
    FileOutputStream(file);                          //实例化
    FileOutputStream对象
```

```
        lock=
    fous.getChannel().tryLock();                    //获取
    文件锁定
        Thread.sleep(60 *
    1000);                                  //线程锁定1分钟
      } catch (Exception e) {
        e.printStackTrace();
      }
    }
```

（2）在该类的主方法中调用锁定文件方法，实现将C盘的count.txt文件进行锁定。具体代码如下：

```
    public static void main(String[] args) {
      String file=
  "C://count.txt";                                  //创建文件
  对象
      fileLock(file);
    //调用文件锁定方法
    }
```

举一反三

根据本实例，读者可以开发以下程序。

对部分文件锁定。

对全部文件锁定。

9.5 文件批量操作

实例352 文件批量重命名

这是一个可以启发思维的实例

实例位置：光盘\mingrisoft\09\Ex09_352

实例说明

Windows 操作系统可以实现重命名文件操作，却不能实现批量重命名。本实例实现了批量重命名功能，可以将一个文件夹内同一类型的文件按照一定的规则批量重命名。用户可以给出重命名模板，程序将根据模板对相应的文件进行重命名。除此之外，还可以在重命名模板中添加特殊符号，程序会将这些特殊符号替换成重命名后的文件编号。实例运行效果如图9.24所示。

图9.24 实例运行效果

技术要点

本实例主要应用了String字符串的格式化方法，该方法可以将指定对象按特定的格式生成字符串，本实例格式化的目的是为新文件名称做递增编号的同时保留指定位数的前导数字 0。例如，3位编号的1应该为001。字符串类的格式化方法声明如下：

```
public static String format(String format, Object... args)
```

该方法的作用是使用指定的格式字符串和参数返回一个格式化字符串。

参数说明

- format：格式化字符串。
- args：格式化字符串中由格式说明符引用的参数。

例如，以下代码可以格式化并返回字符串photo025。

String fileName = String.format("photo%04d", 25);

实现过程

（1）继承JFrame编写一个窗体类，名称为RenameFiles。

（2）设计RenameFiles窗体类时用到的主要控件及说明如表9.26所示。

表9.26 窗体的主要控件及说明

控件类型	控件命名	控件用途
JSpinner	startSpinner	设置起始编号
JTextField	forderField	显示要处理的文件夹
	templetField	新名称的模板字符串
	extNameField	指定文件扩展名，程序将针对这些文件进行改名
JButton	button	“浏览”按钮
	startButton	“开始”按钮
JTable	table	显示文件改名记录

（3）编写“浏览”按钮的事件处理方法。在该方法中创建一个文件选择器，并设置其只对文件夹生效，然后把选择的文件夹保存为类的成员变量，最后把选择的文件夹信息显示在文本框中。关键代码如下：

```
protected void do_button_actionPerformed(ActionEvent e) {
    JFileChooser chooser=new
JFileChooser();                          //创建文件选择器
    //设置只选择文件夹
    chooser.setFileSelectionMode(JFileChooser.DIRECTORIES_O
NLY);
    int option=
chooser.showOpenDialog(this);                          //显示
打开对话框
    if (option == JFileChooser.APPROVE_OPTION) {
```

```
        dir=
    chooser.getSelectedFile();                                    /
    /获取选择的文件夹
      } else {
        dir = null;
      }
      forderField.setText(dir+"");
              //显示文件夹信息
    }
```

（4）编写“开始”按钮的事件处理方法，在这个方法中将完善模板字符串，并利用过滤器来提取指定扩展名类型的文件列表，在遍历文件列表的过程中对文件进行改名，并把改名记录保存到表格中。关键代码如下：

```
    protected void do_startButton_actionPerformed(ActionEvent
e) {
      String templet=
  templetField.getText();                                  //获取模
  板字符串
      if (templet.isEmpty()) {
        JOptionPane.showMessageDialog(this, "请确定重命名模
    板", "信息对话框",JOptionPane.WARNING_MESSAGE);
        return;
      }
      //获取表格数据模型
      DefaultTableModel model = (DefaultTableModel)
  table.getModel();
```

```
    model.setRowCount(0);
      //清除表格数据
    int bi= (Integer)
startSpinner.getValue();                                    //获取
起始编号
    int index=
templet.indexOf("#");                                       //获取
第一个“#”的索引
    String code=
templet.substring(index);                                   //获取
模板中数字占位字符串
    //把模板中数字占位字符串替换为指定格式
    templet = templet.replace(code, "%0"+ code.length()
+"d");
    String extName = extNameField.getText().toLowerCase();
    if (extName.indexOf(".") ==-1)
      extName ="."+ extName;
    //获取文件中的文件列表数组
    File[] files = dir.listFiles(new
ExtNameFileFilter(extName));
    for (File file : files)
{                                                           //变量文件数组
      //格式化每个文件名称
      String name = templet.format(templet, bi++) +
  extName;
      //把文件的旧名称与新名称添加到表格的数据模型
      model.addRow(new String[]{file.getName(),name});
```

```
        File parentFile=
    file.getParentFile();                          //获取文件
    所在文件夹对象
        File newFile = new File(parentFile, name);
        file.renameTo(newFile);
          //文件重命名
      }
    }
```

（5）编写文件过滤器，这个过滤器的任务是只允许获取指定扩展名的文件对象，它将被应用到遍历文件夹所有文件的listFiles()方法中。关键代码如下：

```
    private final class ExtNameFileFilter implements
FileFilter {
      private String extName;
      public ExtNameFileFilter(String extName) {
        this.extName=
    extName;                                       //保存文件扩展
    名
      }
      @Override
      public boolean accept(File pathname) {
        //过滤文件扩展名
        if
    (pathname.getName().toUpperCase().endsWith(extName.toUppe
    rCase()))
          return true;
        return false;
```

```
    }
}
```

举一反三

根据本实例，读者可以实现以下功能。

对文件批量重命名。

批量复制文件。

实例353 快速批量移动文件

这是一个可以美化界面的实例

实例位置：光盘\mingrisoft\09\Ex09_353

实例说明

文件移动是计算机资源管理常用的一个操作，这在操作系统中可以通过文件的剪切与复制来实现，也可以通过鼠标的拖动来实现。但是在Java语言的编程实现中，大多是以复制文件到目的地，再删除原有文件来实现的。这对于小文件来说看不出什么弊端，但是如果移动几个大的文件就会看出操作缓慢并且浪费系统资源。本实例将通过File类的API方法直接实现文件的快速移动，哪怕是几GB的大文件也不会造成严重延时。实例运行效果如图9.25所示。

图9.25 实例运行效果

技术要点

File类位于Java.io类包中，它提供了多种获取文件属性的方法，其中renameTo()方法可用于实现文件的重新命名，但是本实例利用该方法对文件路径进行修改，从而实现文件的快速移动。该方法的声明如下：

public boolean renameTo(File dest)

参数说明

dest：指定文件的新抽象路径名。

对于参数dest，可以设置不同路径的文件对象，这样就可以实现文件的快速移动。

实现过程

（1）继承JFrame编写一个窗体类，名称为QuickMoveFiles。

（2）设计QuickMoveFiles窗体类时用到的主要控件及说明如表9.27所示。

表9.27 窗体的主要控件及说明

控 件 类 型	控 件 命 名	控 件 用 途
JTextField	sourceFolderField	显示选择的源文件列表信息
	targetFolderField	显示要移动到的目标文件夹路径
JTextArea	infoArea	显示文件操作记录
JButton	browserButton1	浏览源文件的按钮
	browserButton2	浏览目标文件夹的按钮

（3）首先实现选择源文件的“浏览”按钮的事件处理方法，在该方法中使用文件选择器来获取用户选择的多个文件，并把选择的文件以数组保存为类的成员变量，同时把所有选择的文件名称显示到文本框中。程序主要代码如下：

```
    protected void
do_browserButton1_actionPerformed(ActionEvent e) {
        JFileChooser chooser=new
  JFileChooser();                                    //创建文件
```

选择器

```
    chooser.setMultiSelectionEnabled(true);
                        //设置文件多选
    int option=
chooser.showOpenDialog(this);
    //显示文件打开对话框
    if (option == JFileChooser.APPROVE_OPTION) {
      files=
  chooser.getSelectedFiles();
            //获取选择的文件数组
      sourceFolderField.setText("");
                        //清空文本框
      StringBuilder filesStr = new StringBuilder();
      for (File file : files)
  {                                                    //遍历文
  件数组
        filesStr.append("、"+
    file.getName());                                    //连接
    文件名称
      }
      String str=
  filesStr.substring(1);
        //获取所有文件名称的字符串
      sourceFolderField.setText(str);
                        //设置文件名称信息到文本框
    } else {
      files = new File[0];
```

```
        sourceFolderField.setText("");
                        //清空文本框
        }
    }
```

（4）编写选择目标文件夹的“浏览”按钮的事件处理方法，该方法将创建文件选择器，并设置其只针对文件夹有效，也就是说只能选择文件夹的文件选择器。在获取用户选择文件夹的同时也把该文件夹的信息显示到文本框中。关键代码如下：

```
    protected void
do_browserButton2_actionPerformed(ActionEvent e) {
        JFileChooser chooser=new
  JFileChooser();                                    //创建文件
  选择器
        //设置选择器只针对文件夹生效
        chooser.setFileSelectionMode(JFileChooser.DIRECTORIES_O
  NLY);
        int option=
  chooser.showOpenDialog(this);
        //显示文件打开对话框
        if (option == JFileChooser.APPROVE_OPTION) {
          dir=
    chooser.getSelectedFile();
                //获取选择的文件夹
          targetFolderField.setText(dir.toString());
                              //显示文件夹到文本框
        } else {
          dir = null;
```

```
        targetFolderField.setText("");
      }
    }
```

（5）编写“移动”按钮的事件处理方法，在该方法中利用之前用户选取的文件数组和目标文件夹实现文件移动操作，并把操作记录显示到JTextArea文本域控件中。关键代码如下：

```
    protected void do_moveButton_actionPerformed(ActionEvent
e) {
      if
  (files.length<=0)
            //判断文件数组有无元素
        return;
      for (File file : files)
  {                                                    //遍历
  文件数组
        File newFile=new File(dir,
    file.getName());                                   //创建移
    动目标文件
        infoArea.append(file.getName()+ "\t移动到
    \t"+dir);                          //显示移动记录
        file.renameTo(newFile);
                    //文件移动
        infoArea.append("------完成
    \n");                                              //显示移动完
    成信息
      }
      //显示操作完成
```

```
    infoArea.append("####################操作完成
####################\n");
    }
```

举一反三

根据本实例，读者可以实现以下功能。

通过文件选择器来获取目标文件夹的绝对路径。

批量删除文件。

实例354 删除磁盘中所有的.tmp临时文件

这是一个可以美化界面的实例

实例位置：光盘\mingrisoft\09\Ex09_354

实例说明

在操作系统中有很多程序建立了太多的.tmp临时文件来为程序提供数据缓冲。这样的文件有的在程序关闭时会自动清理，有的则一直存在。另外，程序在运行期间被非法终止也会导致临时文件的冗余。本实例实现了搜索指定磁盘的.tmp临时文件并进行清理的功能。实例运行效果如图9.26所示。

图9.26 实例运行效果

技术要点

JProgressBar滚动条控件可以通过对Value值的修改来控制界面进度条的滚动，但是对于本实例这一类遍历未知深度与搜索进度的工作，无法为进度条指定滚动范围值。针对这类问题， JProgressBar控件提供了不确定进度的滚动显示方式，它会在界面上循环滚动直到关闭该功能。有关进度条以不确定方式运行的关键方法如下：

public void setIndeterminate(boolean newValue)

参数说明

newValue：如果进度条更改为不确定模式，则为true；如果转换回常规模式，则为false。

此方法设置进度条的indeterminate属性，该属性确定进度条处于确定模式还是不确定模式中。不确定模式的进度条连续地显示动画，指示发生未知长度的操作。默认情况下，此属性为false。有些外观可能不支持不确定进度条，它们将忽略此属性。

实现过程

（1）继承JFrame编写一个窗体类，名称为DeleteAllTempFile。

（2）设计DeleteAllTempFile窗体类时用到的主要控件及说明如表9.28所示。

表9.28 窗体的主要控件及说明

控 件 类 型	控 件 命 名	控 件 用 途
JButton	searchButton	搜索临时文件的按钮
	clearButton	清除临时文件的按钮
JProgressBar	progressBar	显示搜索进度的进度条
JList	driverList	显示驱动器列表
JTable	table	显示搜索到的文件列表

（3）编写窗体激活事件处理方法，这个方法在窗体激活状态下加载计算机的磁盘列表，它们也是File文件对象，然后遍历这些文件对象，把它们添加到JList控件的数据模型中，从而显示在界面中。关键代码如下：

```
  protected void do_this_windowActivated(WindowEvent e) {
    DefaultListModel model = new DefaultListModel();
    File[]
roots=File.listRoots();
//获取计算机磁盘列表
    for (File file : roots)
{                                                    //遍历磁盘列表
      model.addElement(file);
        //添加磁盘到JList控件的模型
    }
    driverList.setModel(model);
        //设置列表控件的模型
  }
```

（4）编写“搜索”按钮的事件处理方法，在该方法中，获取用户在列表控件选择的磁盘对象，然后创建搜索线程对象，并启动这个搜索线程，由该线程去完成临时文件的搜索。关键代码如下：

```
    protected void
do_searchButton_actionPerformed(ActionEvent e) {
      //获取用户在列表控件选择的磁盘对象
      final File driver = (File)
  driverList.getSelectedValue();
      if (searchThread !=null)
  {                                          //如果搜索线程已经初
  始化
        searchThread.setSearching(false);
            //停止该线程
      }
```

```
    //获取表格对象的数据模型
    DefaultTableModel model = (DefaultTableModel)
table.getModel();
    //创建新的搜索线程
    searchThread = new SearchThread(driver, model,
progressBar);
    searchThread.start();
        //启动搜索线程
  }
```

（5）编写“清理”按钮的事件处理方法，这个方法对JTable表格控件中保存的已搜索到的文件进行删除，并更新该文件的处理结果。关键代码如下：

```
   protected void do_clearButton_actionPerformed(ActionEvent
e) {
    //获取表格控件的数据模型
    DefaultTableModel model = (DefaultTableModel)
table.getModel();
    int
rowCount=model.getRowCount();                          //
获取模型中表格数据的行数
    for (int i=0; i< rowCount; i++)
{                                  //变量模型指定行数的数据
      File file= (File)model.getValueAt(i,
1);                          //获取指定行的文件对象
      if
(file.exists())
//判断文件存在
```

```
            file.delete();
              //删除.tmp临时文件
          model.setValueAt("处理完成",
    i,3);                                    //更新模型中对该文件的处
    理结果
      }
    }
```

（6）编写搜索文件的线程类SearchThread。该类继承Thread类并且重写run()方法。在这个方法中调用递归方法listTempFiles()来遍历指定磁盘下的所有子文件夹和临时文件。该递归方法为搜索线程的核心方法，其关键代码如下：

```
    private void listTempFiles(File driver) {
      //获取指定磁盘或文件夹的子列表
      File[] files = driver.listFiles(tempFileFilter);
      if (files == null)
        return;
      progressBar.setIndeterminate(true);
              //设置进度条以不确定方式滚动
      for (File file : files)
  {                                                  //遍历文件数组
        progressBar.setString(file.toString());
                //进度条显示搜索文件夹
        if (file.isFile()&& searching)
    {                                  //处理文件
          tableModel.addRow(new Object[] { file.getName(),
      file,file.length(), "未处理"
      });                              //添加文件信息到表格控件
```

```
        } else if (file.isDirectory()&& searching)
    {                          //处理文件夹
          listTempFiles(file);
            //用递归方法遍历文件夹
        }
      }
      progressBar.setIndeterminate(false);
              //停止进度条
      progressBar.setString("搜索完
  成");                                    //提示搜索完成
    }
```

举一反三

根据本实例，读者可以实现以下功能。

移动所有后缀名为.jpg的文件。

批量删除磁盘中的临时文件。

实例355 批量复制指定扩展名的文件

这是一个可以美化界面的实例

实例位置：光盘\mingrisoft\09\Ex09_355

实例说明

在Windows操作系统下可以很轻松地实现复制文件，但是如果要实现批量复制某个类型的文件，就不是很轻松。本实例实现一个小工具，通过本实例可以实现将某文件夹下指定格式的文件复制到相应的文件夹下。实例运行效果如图9.27所示。

图9.27 实例运行效果

技术要点

实现本实例需要遍历指定文件夹下的文件，之后将满足条件的文件复制到相应的地址下。通过File类的listFiles()方法可以获取指定路径下的文件集合。该方法的语法如下：

File[] listFiles()

该方法返回的是 File 数组。要获取数组中每个 File 文件的绝对路径，可以使用 File 类的getAbsolutePath()方法，该方法以String形式返回文件的绝对路径。该方法的语法如下：

String getAbsolutePath()

实现过程

（1）创建CopyFileFrame类，该类继承自JFrame类，实现窗体类。

（2）向窗体中添加控件，实现窗体布局，主要控件及说明如表9.29所示。

表9.29 窗体的主要控件及说明

控 件 类 型	控 件 命 名	控 件 用 途
JTextField	filePathTextField	显示要复制文件地址的文本框控件
	saveTextField	显示要复制文件的保存路径的文本框控件
JComboBox	typeComboBox	显示要复制文件的文件类型
JButton	choiceButton	为用户提供要进行复制的文件对话框按钮
	saveButton	为用户提供保存复制后的文件地址对话框按钮
	copyButton	显示“复制”按钮

（3）创建工具类CopyUtil，在该类中定义getList()方法，用于获取某文件夹下的文件集合。该方法有一个String类型的参数，用于要查询的文件夹。具体代码如下：

```
  public List getList(String path) {
    LinkedList<File> list=new LinkedList<File>
();                              //定义保存目录的集合对象
    ArrayList<String> listPath=new ArrayList<String>
();                          //定义文件地址的集合对象
    File dir=new
File(path);                                          //根
据文件地址创建File对象
    File
file[]=dir.listFiles();
        //获取文件夹下的文件数组
    for (int i=0; i< file.length; i++)
{                                       //循环遍历数组
      if
  (file[i].isDirectory())
          //判断文件是否是一个目录
        list.add(file[i]);
                //向集合中添加元素
      else {
        listPath.add(file[i].getAbsolutePath());
                    //将文件路径添加到集合中
      }
    }
    File tmp;
```

```
    while (!list.isEmpty())
{                                                    //如果保存文件
路径的集合不为空
      tmp=
  list.removeFirst();
    //移除并返回集合中第一项
      if (tmp.isDirectory()) {
        file = tmp.listFiles();
        if (file == null)
          continue;
        for (int i=0; i< file.length; i++)
    {                                  //循环遍历数组
          if
      (file[i].isDirectory())
          //如果文件表示一个目录
            list.add(file[i]);
          else
      {                                                  //如
      果为一个文件对象
            listPath.add(file[i].getAbsolutePath());
          }
        }
      }
    }
    return listPath;
  }
```

（4）在该CopyUtil类中定义复制文件的方法copyFile()，该方法包含两个String类型的参数，分别用于指定复制文件的路径与复制后文件的保存路径。具体代码如下：

```
public void copyFile(String oldPath, String newPath) {
  try {
    int bytesum = 0;
    int byteread = 0;
    File oldfile = new File(oldPath);
    if (oldfile.exists())
{                                                    //文件存在时
      InputStream inStream=new
  FileInputStream(oldPath);                  //读入源文件
      FileOutputStream fs = new
  FileOutputStream(newPath);
      byte[] buffer = new byte[1444];
      while ((byteread= inStream.read(buffer)) != -1)
  {                      //循环读取文件
        bytesum+=byteread;
          //获取文件大小
        fs.write(buffer,0,byteread);
                //向文件中写数据
      }
      inStream.close();
    }
  } catch (Exception e) {
    e.printStackTrace();
  }
```

}

举一反三

根据本实例，读者可以开发以下程序。

批量复制某文件夹下的所有文件。

批量复制多个文件夹下的指定文件。

9.6 RAR文件压缩

实例356 文件压缩为RAR文档

这是一个可以提高分析能力的实例

实例位置：光盘\mingrisoft\09\Ex09_356

实例说明

文件压缩是对数据的一种紧凑存储格式，通过压缩能够使文件更小，占用更少的磁盘空间，同时也可以减少网络传输的时间。本实例实现文件到RAR文档的压缩，实例运行效果如图9.28所示。

图9.28 实例运行效果

技术要点

Runtime类是每个Java程序都内置的一个运行时对象。通过这个对象可以执行外部命令，这样就可以执行RAR的压缩、解压缩、添加注释等各种命令。但是这个类不能直接创建对象，需要使用静态方法来获取实例对象并且调用对象的方法来执行外部命令。

（1）获取Runtime实例对象

```
public static Runtime getRuntime()
```

该方法返回与当前Java应用程序相关的运行时对象。Runtime类的大多数方法是实例方法，并且必须根据当前的运行时对象对其进行调用。

（2）执行外部命令

public Process exec(String command) throws IOException

该方法在单独的进程中执行指定的字符串命令，并返回该命令的进程对象。

参数说明

command：一条指定的系统命令。

返回一个新的 Process 对象，用于管理子进程。

实现过程

（1）创建CompressTxtToRAR类，该类需要继承JFrame类成为窗体。

（2）设计CompressTxtToRAR窗体类时用到的主要控件及说明如表9.30所示。

表9.30 窗体的主要控件及说明

控件类型	控件命名	控件用途
JButton	addButton	添加待压缩文件的按钮
	removeButton	从表格控件中移除文件的按钮
	compressButton	“压缩”按钮，用于执行文件压缩命令
	stopButton	用于停止压缩任务
	browseButton	选择保存压缩文档 RAR 文件的浏览按钮
JTable	table	显示选择待压缩文件的表格
JTextField	compressFileField	显示 RAR 压缩文档路径的文本框
JProgressBar	progressBar	显示压缩进度的进度条控件

（3）编写“增加”按钮的事件处理方法，在该方法中创建文件选择器控件，让用户通过该控件选择要压缩的文件，然后把文件添加到表格控件中。关键代码如下：

```
    protected void do_addButton_actionPerformed(ActionEvent
arg0) {
        JFileChooser chooser=new
    JFileChooser();                                    //创建文件选择
    器
        chooser.setAcceptAllFileFilterUsed(false);
        chooser.setMultiSelectionEnabled(true);
                    //设置允许文件多选
        int option=
    chooser.showOpenDialog(this);
    //显示文件打开对话框
        if (option != JFileChooser.APPROVE_OPTION)
          return;
        File[] files=
    chooser.getSelectedFiles();
      //获取用户选择文件数组
        //获取表格控件的数据模型
        DefaultTableModel model = (DefaultTableModel)
    table.getModel();
        for (File file : files)
    {                                                  //遍历用户选
    择的文件数组
          //把文件信息添加到表格控件的模型中
          model.addRow(new Object[] { file.getName(),
      file.length(), file });
        }
      }
```

（4）用同样的方法编写“浏览”按钮的事件处理方法，在该方法中完成指定压缩 RAR 文档名称的业务。关键代码如下：

```
    protected void
do_browseButton_actionPerformed(ActionEvent arg0) {
        JFileChooser chooser=new
    JFileChooser();                                    //创建文件选择
    器
        //设置选择文件类型为RAR
        chooser.setFileFilter(new FileNameExtensionFilter("RAR
    压缩文档", "rar"));
        chooser.setAcceptAllFileFilterUsed(false);
        int option=
    chooser.showSaveDialog(this);
    //显示保存对话框
        if (option != JFileChooser.APPROVE_OPTION)
          return;
        rarFile=
    chooser.getSelectedFile();
        //获取用户定制的RAR文件
        compressFileField.setText(rarFile.getPath());
                          //显示RAR文件路径信息
    }
```

（5）编写“压缩”按钮的事件处理方法，这个方法将执行压缩任务，把表格控件中保存的文件对象压缩为指定的RAR文档。这需要利用线程来完成，避免造成GUI界面假死，所以在该方法中应该创建并启用压缩线程。关键代码如下：

```
    protected void
do_compressButton_actionPerformed(ActionEvent arg0) {
      if (rarFile == null) {
        browseButton.doClick();
        if (rarFile == null)
          return;
      }
      progressBar.setVisible(true);
      CompressThread compressThread=new
  CompressThread();                    //创建压缩线程
      compressThread.start();
            //启动线程
    }
```

（6）编写压缩线程CompressThread类，该类继承Thread类成为线程类，在run()核心方法中编写处理文件压缩的核心代码。这需要获取JTable表格控件中的所有文件数据并创建相应的文件列表文件，再在压缩命令中把列表文件作为压缩命令的参数，最后读取压缩进度并控制窗体上滚动条的显示。关键代码如下：

```
    private final class CompressThread extends Thread {
      public void run() {
        try {
          //获取表格控件的数据模型
          DefaultTableModel model = (DefaultTableModel)
      table.getModel();
          int
      rowCount=model.getRowCount();                    //获
      取数据模型中表格的行数
```

```
    StringBuilder fileList = new StringBuilder();
    for (int i=0; i< rowCount; i++)
{                              //遍历数据表格模型中的文件对象
      File file = (File) model.getValueAt(i, 2);
      fileList.append(file.getPath()+
  "\n");                       //把文件路径保存到字符串构
  建器中
    }
    //创建临时文件，用于保存压缩文件列表
    File listFile = File.createTempFile("fileList",
".tmp");
    FileOutputStream fout = new
FileOutputStream(listFile);
    fout.write(fileList.toString().getBytes());
            //保存字符串构建器数据到临时文件
    fout.close();
    //创建压缩命令字符串
    final String command ="rar a "+ rarFile.getPath()
+"@"+ listFile.getPath();
    Runtime
runtime=Runtime.getRuntime();                    //获取
Runtime对象
    progress= runtime.exec(command.toString()+
"\n");              //执行压缩命令
    progress.getOutputStream().close();
        //关闭进程输出流
```

```
    progressBar.setString(null);
        //初始化进度条控件
    progressBar.setValue(0);
    //获取进程输入流
    Scanner scan = new
Scanner(progress.getInputStream());
    while (scan.hasNext()) {
      String line=
  scan.nextLine();                          //获取进程
  提示单行信息
      //获取提示信息的进度百分比的索引位置
      int index = line.lastIndexOf("%") - 3;
      if (index <= 0)
        continue;
      //获取进度百分比字符串
      String substring = line.substring(index, index +
  3);
      //获取整数的百分比数值
      int percent = Integer.parseInt(substring.trim());
      progressBar.setValue(percent);
        //在进度条控件显示的百分比
    }
    progressBar.setString("完成");
    scan.close();
  } catch (IOException e) {
    e.printStackTrace();
  }
```

```
    }
}
```

举一反三

根据本实例，读者可以开发以下程序。

将文件压缩成RAR文件。

解压缩RAR文件。

实例357 解压缩RAR压缩包

这是一个可以启发思维的实例

实例位置：光盘\mingrisoft\09\Ex09_357

实例说明

文件解压缩是最常用的数据操作，目前大多数资料和软件都采用RAR格式进行压缩并在网站上提供下载，经过压缩的资源占用空间更小，在网络中的传输速度更快。用户从网站上下载该文件之后，需要使用RAR软件进行解压缩才能获取自己想要的资源。本程序实现对RAR压缩包的解压缩功能，可以针对指定的压缩包文件定制解压的目标文件夹。实例运行效果如图9.29所示。

图9.29 实例运行效果

技术要点

本节所介绍的所有实例都是通过调用 rar.exe 命令来执行的，如果读者的计算机中安装了WinRAR，那么在该软件的安装文件夹中会包

含rar.exe命令文件，本节所有实例需要这个命令才能实现RAR文档的操作。下面介绍使用该命令的方法。

□ 复制RAR命令文件到项目文件夹

可以找到WinRar软件的安装文件夹，把rar.exe文件复制到自己的项目中，如在Eclipse项目中，把rar.exe文件复制到根目录，即与src文件夹同级，这样程序代码就可以直接调用该命令，也可以把rar.exe文件复制到某个文件夹，然后在程序代码中使用绝对路径来调用该命令，但是那样不利于软件的复制与传播。

□ 设置path 环境变量

另一种方法是复制WinRar软件的安装文件夹路径，然后添加到系统的path环境变量中。例如，笔者计算机中的安装路径是“C:\Program Files\WinRAR”。把这个文件夹路径作为值添加到path环境变量中，如果其左右都有其他变量值，那么使用英文的“;”分号进行分割。例如：

; C:\Program Files\WinRAR;

把上面的字符串添加到path环境变量中，然后重新运行Eclipse以使用新的系统环境变量。

实现过程

（1）继承JFrame编写一个窗体类，名称为DeCompressRAR。

（2）设计DeCompressRAR窗体类时用到的主要控件及说明如表9.31所示。

表9.31 窗体的主要控件及说明

控 件 类 型	控 件 命 名	控 件 用 途
JButton	browseButton	用于选择压缩文档
	pathButton	选择压缩文档解压路径的按钮
	deCompressButton	执行解压缩的按钮
JTextField	compressFileField	显示选择的压缩文档
	pathField	显示用户选择的解压缩路径
JProgressBar	progressBar	显示解压缩进度的百分比

（3）编写选择压缩文件的“浏览”按钮的事件处理方法。该方法的关键代码如下：

```
    protected void
do_browseButton_actionPerformed(ActionEvent arg0) {
        JFileChooser chooser=new
  JFileChooser();                                    //创建文件选择
  器
        //设置选择文件类型为rar
        chooser.setFileFilter(new FileNameExtensionFilter("RAR
  压缩文档", "rar"));
        chooser.setAcceptAllFileFilterUsed(false);
        chooser.setDialogTitle("选择RAR压缩文
  件");                                    //设置对话框标题
        int option=
  chooser.showOpenDialog(this);
  //显示保存对话框
        if (option != JFileChooser.APPROVE_OPTION)
          return;
        rarFile=
  chooser.getSelectedFile();
        //获取用户定制的RAR文件
        compressFileField.setText(rarFile.getPath());
                        //显示RAR文件的路径信息
    }
```

（4）编写选择解压缩文件夹的“路径”按钮的事件处理方法。该方法的关键代码如下：

```
    protected void do_pathButton_actionPerformed(ActionEvent
arg0) {
        JFileChooser chooser=new
    JFileChooser();                                    //创建文件选择
    器
        chooser.setDialogTitle("选择解压缩文件
    夹");                                    //设置对话框标题
        chooser.setAcceptAllFileFilterUsed(false);
        //选择解压缩文件夹
        chooser.setFileSelectionMode(JFileChooser.DIRECTORIES_O
    NLY);
        int option=
    chooser.showOpenDialog(this);
    //显示文件打开对话框
        if (option != JFileChooser.APPROVE_OPTION)
          return;
        dir=
    chooser.getSelectedFile();
            //获取选择的文件夹
        pathField.setText(dir.toString());
                    //把文件夹路径更新到文本框
    }
```

（5）“解压”按钮的事件处理方法将确认需要的压缩文档和解压缩文件夹两个参数的完整性，只有这两个参数都是非NULL值的情况下，才能执行解压缩。解压缩的操作是由单独的线程对象完成的，该按钮的事件处理方法只需要创建并启动该线程。关键代码如下：

```
    protected void
do_deCompressButton_actionPerformed(ActionEvent e) {
        if
    (rarFile==null)                                          //如果
    未选择压缩文档
          browseButton.doClick();
      //执行选择压缩文件按钮的单击操作
        if
    (dir==null)                                              //如果
    未选择解压缩文件夹
          pathButton.doClick();
      //执行选择解压缩文件夹的路径按钮的单击操作
        if (rarFile==null
    ||dir==null)                                  //如果参数不全，
    则终止本方法
          return;
        //创建命令字符串
        final String command ="rar x "+ rarFile +""+ dir +"/y";
        //让用户确认是否覆盖目标文件夹同名文件
        int option = JOptionPane.showConfirmDialog(null, "此操
    作会覆盖目标文件夹同名文件，是否继续");
        if (option != JOptionPane.YES_OPTION)
          return;                                             //不
      覆盖目标文件夹内容则不执行解压缩
        new
    DeCompressThread(command).start();                        //创
    建并启动解压缩线程
```

```
}
```

（6）编写处理解压缩业务的线程类，该类继承 Thread 类，并重写其 run()方法，在该方法中执行解压缩的命令，并把进度显示到窗体中的进度条控件中。关键代码如下：

```
private final class DeCompressThread extends Thread {
  private final String command;
  private DeCompressThread(String command) {
    this.command = command;
  }
  public void run() {
    try {
      final Process process =
  Runtime.getRuntime().exec(command);
      process.getOutputStream().close();
      final Scanner scan = new
  Scanner(process.getInputStream());
      progressBar.setString(null);
              //初始化进度条控件
      progressBar.setValue(0);
      while (scan.hasNext()) {
        String line=
    scan.nextLine();                                    //获取
    进程提示单行信息
        //获取提示信息的进度百分比的索引位置
        int index = line.lastIndexOf("%") - 3;
        if (index <= 0)
          continue;
```

```
                //获取进度百分比字符串
                String substring = line.substring(index, index +
        3);
                //获取整数的百分比数值
                int percent = Integer.parseInt(substring.trim());
                progressBar.setValue(percent+1);
                      //在进度条控件中显示百分比
              }
              progressBar.setString("完成");
              process.getInputStream().close();
            } catch (IOException e1) {
              e1.printStackTrace();
            }
          }
        }
```

举一反三

根据本实例，读者可以实现以下程序。

完成文件解压缩。

将文件压缩到目标文件夹。

实例358 文件分卷压缩

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\09\Ex09_358

实例说明

把多个文件压缩成一个压缩文档方便文件的保存与传输，经过压缩后文件内容不变，但是占用磁盘空间更小，这样有利于Internet传

输，也可以利用移动设备在多个计算机中进行复制。有时对于太大或者太多的文件进行压缩后体积仍然很大，这种情况下可以把压缩后的文件进行分卷，也就是说压缩后的 RAR 文档不再是一个文件，而是多个指定大小的压缩分卷文件，这样可以分别传输部分分卷文件，然后在目的地把各分卷文件解压缩成原有资源文件。本实例就利用RAR实现了压缩文档的分卷功能。实例运行效果如图9.30所示。

图9.30 实例运行效果

技术要点

JFormattedTextField类是一个文本框控件，但是它不同于JTextField控件，它是一个格式文本框控件。该控件可以设置文本框的值类型，并对用户输入进行校验，从而获取指定类型的值。下面介绍该控件的简单使用方法。

（1）创建指定类型格式文本框控件

创建JFormattedTextField控件的构造方法有很多，本实例使用了其中的一个，该构造方法的声明如下：

public JFormattedTextField(Format format)

参数说明

format：用于查找AbstractFormatter的Format，该参数是一个格式抽象类的对象。

可以为该构造方法传递任何格式对象，这样创建后的控件会以该格式对象创建格式化工厂，并对用户输入进行验证与格式化。例如，本实例使用数字格式化抽象类的静态方法创建了整型数字格式化对象，并传递给该控件的构造方法，这样该控件就只接收整数输入。关键代码如下：

```
new
JFormattedTextField(NumberFormat.getIntegerInstance());
```

（2）设置并获取控件的整型数值

通过该控件的 getValue()和 setValue()方法可以设置并获取控件中符合格式规范的数值，但是用户输入与实际想获取的数值类型可能会有一些出入，所以需要对其进行一些转换。如setValue()方法为控件设置整型数值，关键代码如下：

```
volumetField.setValue(1024);
```

立刻从该控件中获取值以后，其类型与设置该值的类型完全相同，但是当用户在控件中输入新的数字，也就是对控件内容进行编辑之后，其值的类型会修改为控件默认数值类型。例如，整型格式化后的默认值类型是Long长整型，而这时如果将值赋值给Integer整型变量，会出现类型转换异常。解决方法是使用该数值类型的超类的静态方法来获取指定类型的值，例如：

```
volumetField.setValue(1024);
//用户编辑文本框内容以后，获取目标类型的数值
int volumeSize = ((Number)
volumetField.getValue()).intValue();
```

实现过程

（1）创建VolumeCompress类，该类需要继承JFrame类成为窗体。

（2）设计VolumeCompress窗体类时用到的主要控件及说明如表9.32所示。

表9.32 窗体的主要控件及说明

控件类型	控件命名	控件用途
JButton	addButton	添加待压缩文件的按钮
	removeButton	从表格控件中移除文件的按钮
	compressButton	压缩按钮，用于执行文件压缩命令
	stopButton	用于停止压缩任务
	browseButton	选择保存压缩文档 RAR 文件的浏览按钮
JTable	table	显示选择待压缩文件的表格
JTextField	compressFileField	显示 RAR 压缩文档路径的文本框
JProgressBar	progressBar	显示压缩进度的进度条控件
JFormattedTextField	volumetField	设置分卷压缩大小的文本框

（3）本程序的核心在于压缩线程CompressThread类的创建，这个线程类从界面的表格控件中读取需要压缩的文件并生成文件列表文件，把该列表文件作为压缩命令的参数。为压缩命令传参的同时设置压缩文件每个分卷文档的大小，单位为KB。关键代码如下：

```
private final class CompressThread extends Thread {
  public void run() {
    try {
      //获取表格控件的数据模型
      DefaultTableModel model = (DefaultTableModel)
table.getModel();
      int
rowCount=model.getRowCount();                    //获
取数据模型中表格的行数
      StringBuilder fileList = new StringBuilder();
      for (int i=0; i< rowCount; i++)
{                          //遍历数据表格模型中的文件对象
        File file = (File) model.getValueAt(i, 2);
```

```
      fileList.append(file.getPath()+
  "\n");                    //把文件路径保存到字符串构
  建器中
    }
    //创建临时文件，用于保存压缩文件列表
    File listFile = File.createTempFile("fileList",
".tmp");
    FileOutputStream fout = new
FileOutputStream(listFile);
    fout.write(fileList.toString().getBytes());
            //保存字符串构建器数据到临时文件
    fout.close();
    int volumeSize = ((Number)
volumetField.getValue()).intValue();
    //创建压缩命令字符串
    final String command ="rar a -v"+volumeSize+"k "+
rarFile.getPath() +"@"+ listFile.getPath();
    Runtime
runtime=Runtime.getRuntime();                    //获取
Runtime对象
    progress= runtime.exec(command.toString()+
"\n");            //执行压缩命令
    progress.getOutputStream().close();
        //关闭进程输出流
    progressBar.setString(null);
        //初始化进度条控件
    progressBar.setValue(0);
```

```
        //获取进程输入流
        Scanner scan = new
    Scanner(progress.getInputStream());
        while (scan.hasNext()) {
          String line=
      scan.nextLine();                              //获取进程
      提示单行信息
          //获取提示信息的进度百分比的索引位置
          int index = line.lastIndexOf("%") - 3;
          if (index <= 0)
            continue;
          //获取进度百分比字符串
          String substring = line.substring(index, index +
      3);
          //获取整数的百分比数值
          int percent = Integer.parseInt(substring.trim());
          progressBar.setValue(percent);
            //在进度条控件中显示百分比
        }
        progressBar.setString("完成");
        scan.close();
      } catch (IOException e) {
        e.printStackTrace();
      }
    }
}
```

举一反三

根据本实例，读者可以实现以下程序。

实现文件的分卷压缩。

将分卷压缩文件压缩到指定文件夹。

实例359 从RAR压缩包中删除文件

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\09\Ex09_359

实例说明

RAR压缩文档对于少量文件的压缩与解压缩速度很快，但是如果文件数量和数据太多，使用任何优化技术都无法大幅度提高压缩速度，因为没有数据可以忽略与抛弃。而对于这样大的RAR压缩包要删除其中的某个或某些文件，重新解压、整理再压缩是不现实的。为此，本程序通过对RAR命令的操作，实现了直接删除RAR压缩包中部分文件的功能。实例运行效果如图9.31所示。

技术要点

本实例使用 RAR 的命令实现从 RAR 压缩包中删除指定名称的文件。实例中用到的 RAR完整命令格式如下：

```
rar d  -c- "rarfile" deleteFile
```

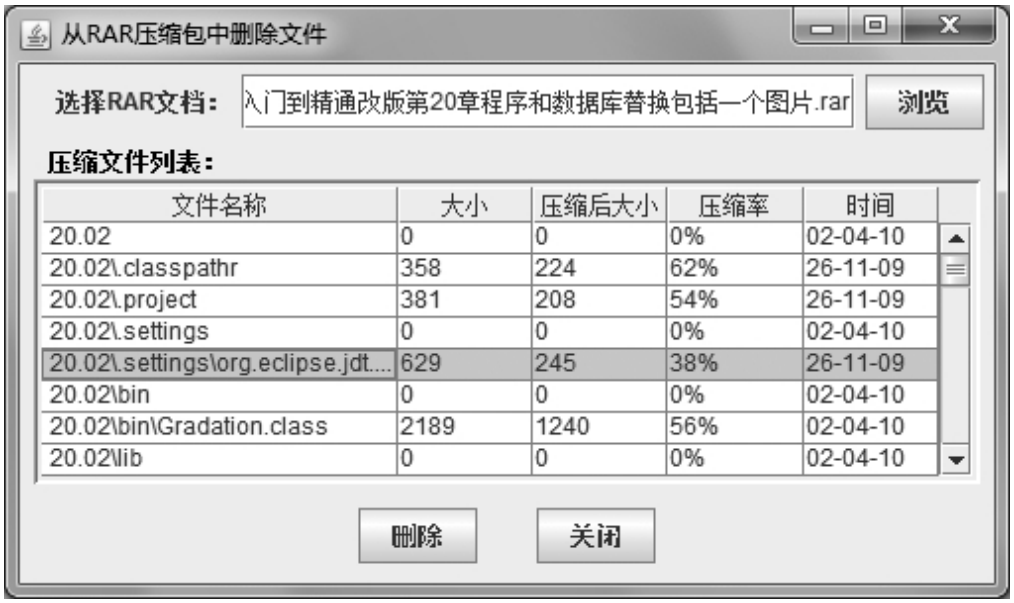


图9.31 实例运行效果

参数说明

- rarfile：一个RAR 压缩文档文件。
- deleteFile：将要从RAR 压缩文件中删除的文件。

实现过程

（1）继承JFrame编写一个窗体类，名称为DeleteFileFromRAR。

（2）设计DeleteFileFromRAR窗体类时用到的主要控件及说明如表9.33所示。

表9.33 窗体的主要控件及说明

控 件 类 型	控 件 命 名	控 件 用 途
JButton	browseButton	选择 RAR 文件的浏览按钮
	delButton	用于执行压缩包文件删除命令
	closeButton	用于关闭当前窗体
JTextField	rarFileField	显示 RAR 文件信息的文本框
JTable	table	显示 RAR 压缩文件列表

（3）本实例的核心在于“删除”按钮的事件处理方法，在该方法中首先获取表格控件的数据模型和表格当前选择行号，然后从RAR压缩包中删除表格指定行对应的文件。关键代码如下：

```
    protected void do_delButton_actionPerformed(ActionEvent
e) {
      //获取表格数据模型
      DefaultTableModel model = (DefaultTableModel)
  table.getModel();
      int selectedRow=
  table.getSelectedRow();                              //获取表格
  当前选择行
      if(selectedRow<0)return;
      //获取选择行中的文件名
      String path = model.getValueAt(selectedRow,
  0).toString();
      try {
        //执行RAR删除命令
        Process exec = getRuntime().exec("rar d -c-\""+
    rarFile +"\""+ path);
        //创建进程输入流
        Scanner scan = new Scanner(exec.getInputStream());
        while (scan.hasNext())
    {                                          //遍历输入流内容
          scan.nextLine();
          //清空输入流数据
        }
        scan.close();
        //关闭输入流
      } catch (IOException e1) {
        e1.printStackTrace();
```

```
    }
    resolveFileList();
        //重载表格中的文件列表数据
}
```

举一反三

根据本实例，读者可以开发以下程序。

从压缩包中删除文件。

从压缩包中移动文件。

实例360 在压缩文件中查找字符串

这是一个可以提高分析能力的实例

实例位置：光盘\mingrisoft\09\Ex09_360

实例说明

计算机中有很多数据文件都使用 RAR 或其他压缩格式进行备份，在需要时来做恢复。作为备份的压缩文件随着资料的积累，压缩内容会越来越多，本实例实现了在压缩文件中搜索字符串的功能，通过这个搜索可以确认某个要查找的资料位于压缩包中的哪个文件中。实例运行效果如图9.32所示。

图9.32 实例运行效果

技术要点

本实例使用RAR的命令实现从RAR压缩包中搜索指定的字符串，并确定该字符串位于某个文件中。本实例中用到的RAR完整命令格式如下：

```
rar i[i|c]="sText"-c-"rarFile" extName
```

参数说明

- sText：要搜索的文本字符串。
- rarFile：一个RAR 压缩文档文件。
- extName：要搜索的文件类型，也就是文件的扩展名，可以使用通配符，例如：

```
rar ii="test"-c-"D:\资料.rar"*.txt
```

实现过程

（1）继承JFrame编写一个窗体类，名称为FindRARString。

（2）设计FindRARString窗体类时用到的主要控件及说明如表9.34所示。

表9.34 窗体的主要控件及说明

控 件 类 型	控 件 命 名	控 件 用 途
JButton	browseButton	选择 RAR 文件的浏览按钮
	searchButton	执行搜索任务的按钮
JTextField	rarFileField	显示 RAR 文件信息的文本框
	searchStringField	接收用户输入的搜索字符串
	extNameField	输入搜索的文件扩展名称
JRadioButton	radioButton1	选择搜索文本区分大小写
	radioButton2	选择搜索文本不区分大小写
JTextArea	infoArea	显示搜索结果

（3）程序主要代码如下：

```
protected void
do_searchButton_actionPerformed(ActionEvent e) {
    String searchText = searchStringField.getText();
    if (searchText.isEmpty() || rarFile == null) {
      getToolkit().beep();//发出提示声音
      return;
    }
    //获取区分大小写的标记
    String arg = group.getSelection().getActionCommand();
    int count = 0;
    try {
      System.out.println("rar i"+ arg +"=\""+ searchText
  +"\"-c-\""+ rarFile +"\""+ extNameField.getText());
      //执行RAR命令
      Process process = getRuntime().exec("rar i"+ arg
  +"=\""+ searchText +"\"-c-\""+ rarFile +"\""+
  extNameField.getText());
      //获取进程的输入流扫描器
      Scanner scan = new Scanner(process.getInputStream());
```

```
    infoArea.setText("");
    while (scan.hasNext())
{                                          //遍历进程执行结果
      String line=
  scan.nextLine();                                  //获取单行
  信息
      if (line.isEmpty()) count++;
      if
  (count<2)                                         //过滤
  非查询结果
        continue;
      infoArea.append(line+
  "\n");                                 //将查询结果添加到文
  本域控件
    }
  } catch (IOException e1) {
    e1.printStackTrace();
  }
}
```

举一反三

根据本实例，读者可以开发以下程序。

在压缩文件中查找字符串。

在压缩文件中查找指定文件。

实例361 重命名RAR压缩包中的文件

这是一个可以启发思维的实例

实例位置：光盘\mingrisoft\09\Ex09_361

实例说明

RAR压缩文档可以保存很多资料文件的备份，并且节省磁盘空间，也利于网络传输。但是一个包含上百个资源文件的 RAR 压缩包，经常包含需要修改的内容，如某个文件的名称需要修改，如果把所有文件解压缩、改名再压缩会非常烦琐，本实例通过 RAR 的命令实现了压缩包中单个文件的改名操作。实例运行效果如图9.33所示。

图9.33 实例运行效果

技术要点

本实例使用RAR的命令实现从RAR压缩包中搜索指定的字符串，并确定该字符串位于某个文件中。本实例中用到的RAR完整命令格式如下：

rar rn <rarFile><sourceFile1><target1>

参数说明

● rarFile：一个RAR 压缩文档文件。

● sourceFile1：要修改的位于压缩包中的文件名称。

● target1：新的文件名称，源文件将使用这个新名称命名。

实现过程

（1）继承JFrame编写一个窗体类，名称为RenameFileFromRAR。

（2）设计RenameFileFromRAR窗体类时用到的主要控件及说明如表9.35所示。

表9.35 窗体的主要控件及说明

控 件 类 型	控 件 命 名	控 件 用 途
JButton	browseButton	选择 RAR 文件的浏览按钮
	renameButton	用于执行压缩包文件重命名的命令
	closeButton	用于关闭当前窗体
JTextField	rarFileField	显示 RAR 文件信息的文本框
JTable	table	显示 RAR 压缩文件列表

（3）程序的主要代码如下：

```
protected void
do_renameButton_actionPerformed(ActionEvent e) {
    //获取表格数据模型
    DefaultTableModel model = (DefaultTableModel)
  table.getModel();
    int selectedRow=
  table.getSelectedRow();                                    //获
  取表格当前选择行
    if (selectedRow < 0)
      return;
    //获取选择行中的文件名
    String path = model.getValueAt(selectedRow,
  0).toString();
```

```
    String
newFile=newFileField.getText();
    //获取新文件名称
    try {
      //执行RAR改名命令
      Process exec = getRuntime().exec("rar rn -c-\""+
  rarFile +"\""+ path +""+ newFile);
      //创建进程输入流
      Scanner scan = new Scanner(exec.getInputStream());
      while (scan.hasNext())
  {                                                    //遍历输入流内容
        scan.nextLine();
              //清空输入流数据
      }
      scan.close();
          //关闭输入流
    } catch (IOException e1) {
      e1.printStackTrace();
    }
    resolveFileList();
            //重载表格中的文件列表数据
  }
```

举一反三

根据本实例，读者可以开发以下程序。

将RAR包中的文件重命名。

批量命名RAR包中的文件。

实例362 创建自解压RAR压缩包

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\09\Ex09_362

实例说明

RAR可以把多个资料文件压缩成一个压缩包文件，这样就可以把大量的资料文件压缩在一起，然后复制到另一个工作室或者其他工作人员那里进行交流，但是如果对方计算机没有安装相应的软件进行解压缩，那就麻烦了。本实例实现RAR文件的改装，将一个RAR压缩文件添加自解压模块，这样它就可以自己运行来解压缩数据，而不需要专门的软件。实例运行效果如图9.34所示。

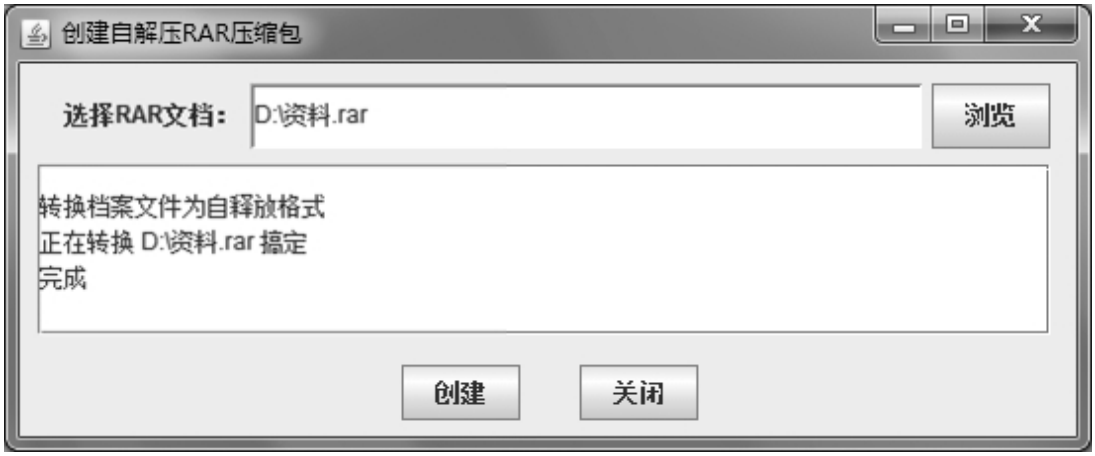


图9.34 实例运行效果

技术要点

本实例使用RAR的命令实现为指定RAR压缩包添加自解压模块的功能，这将使目标RAR压缩文件拥有自行解压缩的能力，而不需要其他软件来实现解压缩。本实例中用到的 RAR 完整命令格式如下：

rar s <rarFile>

参数说明

rarFile：一个RAR压缩文档文件。

例如：

rar s -c--y 资料.rar

这个命令是把名称为“资料.rar”的文件生成可以自动解压缩的“资料.exe”文件。

实现过程

（1）继承JFrame编写一个窗体类，名称为SFXRAR。

（2）设计SFXRAR窗体类时用到的主要控件及说明如表9.36所示。

表9.36 窗体的主要控件及说明

控件类型	控件命名	控件用途
JButton	browseButton	选择指定的 RAR 压缩文件
	createButton	创建指定 RAR 文件的自解压可执行文件
	closeButton	关闭当前窗体
JTextArea	infoArea	显示执行结果

（3）程序的主要代码如下：

```
protected void
do_createButton_actionPerformed(ActionEvent e) {
      if
  (rarFile==null)                                          //
  验证用户是否选择了RAR文件
        return;
      try {
        //执行RAR命令
        Process process = getRuntime().exec("rar s -y -c-"+
    rarFile);
        Scanner scan = new Scanner(process.getInputStream());
        infoArea.setText("");
          //清空文本域控件的内容
        int count = 0;
```

```
    while (scan.hasNext())
{                                          //遍历进程执行结果
      String line=
  scan.nextLine();                                  //获取单行
  信息
      if (line.isEmpty())
        count++;
      if
  (count<2)                                         //过滤
  非查询结果
        continue;
      infoArea.append(line+
  "\n");                                 //将查询结果添加到文
  本域控件
    }
  } catch (IOException e1) {
    e1.printStackTrace();
  }
}
```

举一反三

根据本实例，读者可以开发以下程序。

创建自解压压缩包。

批量解压缩。

实例363 设置RAR压缩包密码

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\09\Ex09_363

实例说明

RAR作为目前最流行的压缩文档之一，经常被用来打包资源并在网络中传输，从网络上下载的资源、软件、音频和视频等很多都是经过RAR 压缩后供用户下载的。但是有些资源需要保密，只有知道密码的人才能够获取压缩包中的文件。RAR支持对压缩包设置密码的功能，本实例在RAR命令的基础上实现了图形化操作的加密程序。实例运行效果如图9.35所示。

图9.35 实例运行效果

技术要点

本实例使用RAR的命令把用户选定的资源文件压缩为RAR压缩包并支持密码设置功能，设置密码以后只有通过合法的密码才能解压缩这个RAR压缩包。本实例中用到的RAR完整命令格式如下：

```
rar a -p["password"] <rarFile>
```

参数说明

- password：要设置的压缩密码。

● rarFile：一个RAR 压缩文档文件。

例如：

rar a -p"mrsoft"-y 资料.rar *.*

这个命令是把当前文件夹中的所有文件压缩成名称为“资料.rar”的压缩文件。同时设置该压缩文件的密码为“mrsoft”。

实现过程

（1）继承JFrame编写一个窗体类，名称为CompressFileWithPassword。

（2）设计CompressFileWithPassword窗体类时用到的主要控件及说明如表9.37所示。

表9.37 窗体的主要控件及说明

控件类型	控件命名	控件用途
JButton	addButton	添加待压缩文件的按钮
	removeButton	从表格控件中移除文件的按钮
	compressButton	压缩按钮，用于执行文件压缩命令
	stopButton	用于停止压缩任务
	browseButton	选择保存压缩文档 RAR 文件的浏览按钮
JTable	table	显示选择待压缩文件的表格
JTextField	compressFileField	显示 RAR 压缩文档路径的文本框
JProgressBar	progressBar	显示压缩进度的进度条控件
JPasswordField	passwordField1	接收用户输入密码
	passwordField2	用于确认用户输入密码的正确性

（3）程序的主要代码如下：

```
private final class CompressThread extends Thread {
  public void run() {
    try {
      progressBar.setString(null);
                    //初始化进度条控件
      progressBar.setValue(0);
      //获取密码
```

```
    String pass1 =
String.valueOf(passwordField1.getPassword());
    //获取确认密码
    String pass2 =
String.valueOf(passwordField2.getPassword());
    String passCommand=
"";                                    //设置密码命令字符串
    if (pass1 != null) {
      if (pass1.equals(pass2))
  {                                  //判断两次密码是否相同
        passCommand= "-p\""+pass1+ "\"
    ";              //完成密码命令
      }else{                                          /
  /如果两次密码不一样则终止当前命令
        JOptionPane.showMessageDialog(null, "两次输入密
    码不一致");
        return;
      }
    }
    //获取表格控件的数据模型
    DefaultTableModel model = (DefaultTableModel)
table.getModel();
    int
rowCount=model.getRowCount();                          //获
取数据模型中表格的行数
    StringBuilder fileList = new StringBuilder();
```

```
    for (int i=0; i< rowCount; i++)
{                           //遍历数据表格模型中的文件对象
      File file = (File) model.getValueAt(i, 2);
      fileList.append(file.getPath()+
  "\n");                    //把文件路径保存到字符串构
  建器中
    }
    //创建临时文件，用于保存压缩文件列表
    File listFile = File.createTempFile("fileList",
".tmp");
    FileOutputStream fout = new
FileOutputStream(listFile);
    fout.write(fileList.toString().getBytes());
            //保存字符串构建器数据到临时文件
    fout.close();
    //创建压缩命令字符串
    final String command ="rar a "+ passCommand+
rarFile.getPath() +"@"+ listFile.getPath();
    Runtime
runtime=Runtime.getRuntime();                    //获取
Runtime对象
    progress= runtime.exec(command.toString()+
"\n");              //执行压缩命令
    progress.getOutputStream().close();
        //关闭进程输出流
    //获取进程输入流
```

```
        Scanner scan = new
    Scanner(progress.getInputStream());
        while (scan.hasNext()) {
          String line=
      scan.nextLine();                                    //获取
      进程提示单行信息
          //获取提示信息的进度百分比的索引位置
          int index = line.lastIndexOf("%") - 3;
          if (index <= 0)
            continue;
          //获取进度百分比字符串
          String substring = line.substring(index, index +
      3);
          //获取整数的百分比数值
          int percent = Integer.parseInt(substring.trim());
          progressBar.setValue(percent);
                  //在进度条控件中显示百分比
        }
        progressBar.setString("完成");
        scan.close();
      } catch (IOException e) {
        e.printStackTrace();
      }
    }
}
```

举一反三

根据本实例，读者可以实现以下功能。

为压缩包设置解压密码。
为压缩后的文件设置只读属性。

9.7 数据压缩的网络应用

实例364 以压缩格式传输网络数据

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\09\Ex09_364

实例说明

现代人的生活已经离不开网络，网络可以为人们带来最新的信息、新闻、技术、影视等很多资源。通过网络数据传输与信息收发，可以让远隔千里的人互相通信、传送文件等。这些都依赖于网络传输速度，为节省网络带宽并提高数据传输速度，程序开发人员经常把数据进行特殊的压缩处理然后在网络中进行传输。本实例通过Java提供的ZIP输出流实现了以压缩格式传输网络数据的功能，利用最少的网络占用实现最快的数据传输。实例运行效果如图9.36所示。

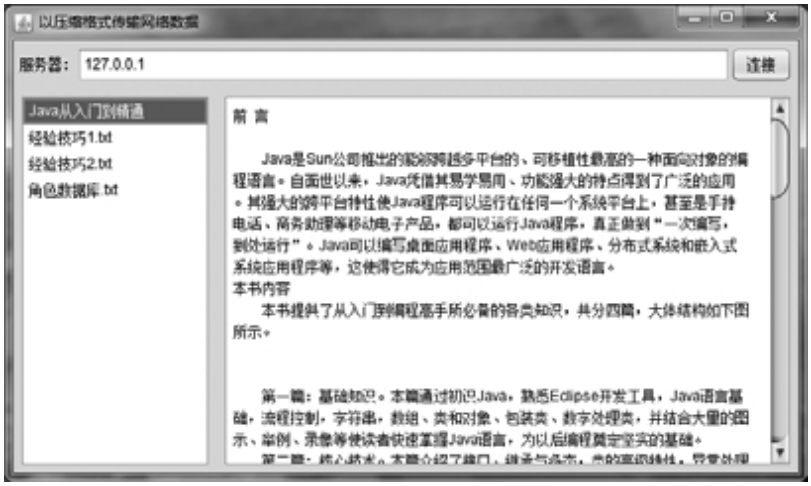


图9.36 实例运行效果

技术要点

本实例使用ZIP数据流在网络中传输数据，这期间使用的当然是二进制流，但是用于传送文本很容易导致两次读取过程中把双字节的汉字编码拆分，从而造成数据中文乱码的情况。为避免该问题的出现，本实例通过InputStreamReader类把二进制流转换为字符流，这样就可以正确地读取每一个字符，包括中文汉字。本实例中把ZIP二进制数据流转换为字符数据流的关键代码如下：

```
    ZipInputStream zis = new
ZipInputStream(socket.getInputStream());
    char[]data=new
char[1024];
//缓冲数组
    int readNum;
    zis.getNextEntry();
                    //读取下一个ZIP条目
    //把ZIP二进制输入流转换为字符输入流
    InputStreamReader ir=new
InputStreamReader(zis);                                    //二进
制到字符数据流的转换
    while ((readNum= ir.read(data))>0)
{                                                  //读取数据
      //数据操作代码
    }
```

实现过程

（1）编写InfoServer类。该类需要实现信息服务器，这个服务器的主方法将创建接收用户连接的ServerSocket类，通过该类来获取用户连接的Socket对象，并把这个Socket对象传递给一个线程对象。关键代码如下：

```
    public static void main(String[] args) throws IOException
{
      ServerSocket ss=new
  ServerSocket(1598);                                    //创
  建socket服务器对象
      System.out.println("服务器已经启
  动。。。。");                                          //输出提示信
  息
      while (!ss.isClosed()) {
        final Socket socket = ss.accept();
        new SocketThread(socket).start();
      }
    }
```

（2）内部类SocketThread是用于处理用户Socket连接的线程类，在这个线程类中接收用户消息，再把与消息同名的文件内容以ZIP格式压缩并传送到客户端。关键代码如下：

```
    private static final class SocketThread extends Thread {
      private static final String TEXT_FILE_PATH
  ="/com/textFile/";
      private final Socket socket;
      private SocketThread(Socket socket) {
        this.socket = socket;
      }
      public void run() {
        try{
          //创建Socket输入流扫描器
```

```
    final Scanner scanner = new
Scanner(socket.getInputStream());
    //创建存放文本文件的文件夹对象
    File dir = new
File(getClass().getResource(TEXT_FILE_PATH).toURI());
    String[]
files=dir.list();
//获取文件列表数组
    ObjectOutputStream dout=new
ObjectOutputStream(socket.getOutputStream());
                        //创建对象输出流
    dout.writeObject(files);
            //把文件列表数组输出到socket
    while (scanner.hasNext())
{                                       //遍历socket输入流
的扫描器数据
      String line=
  scanner.nextLine();                                  //读取
  一行文本
      InputStream
  is=getClass().getResourceAsStream(TEXT_FILE_PATH+
  line);                          //加载文本文件输入流
      ZipOutputStream zout=new
  ZipOutputStream(socket.getOutputStream());
                        //创建socket的ZIP输出流
      byte[]data=new
  byte[1024];                                  //创建数据缓
```

```
冲
        int readNum;
        //为ZIP输出流添加一个压缩条目
        zout.putNextEntry(new ZipEntry("one"));
        while (is != null && (readNum = is.read(data)) >
    0) {
          zout.write(data,0,
      readNum);                          //向ZIP流写数据
        }
        zout.closeEntry();
            //关闭压缩条目
        is.close();
          //关闭文件输入流
      }
      scanner.close();
            //关闭输入流扫描器
      socket.close();
          //关闭socket
    } catch (IOException e) {
      e.printStackTrace();
    } catch (URISyntaxException e) {
      e.printStackTrace();
    }
  }
}
```

（3）继承JFrame编写一个窗体类，名称为ClientFrame。设计ClientFrame窗体类时用到的主要控件及说明如表9.38所示。

表9.38 窗体的主要控件及说明

控 件 类 型	控 件 命 名	控 件 用 途
JButton	linkButton	根据用户输入的主机名或地址去连接服务器
JTextField	hostField	接收用户输入的服务器名称或 IP 地址
JTextArea	infoArea	显示从服务器接收的压缩流中的数据
JList	list	接收服务器传送过来的文件列表数组

（4）“连接”按钮的事件处理方法将获取用户输入的主机名称或IP地址，然后通过该地址来创建Socket连接，从而向服务器发送消息并接收服务器传递过来的数据。程序关键代码如下：

```
    protected void do_linkButton_actionPerformed(ActionEvent
e) {
      try {
        String
    host=hostField.getText();
            //获取输入的主机名或地址
        socket=new
    Socket(host,1598);
      //创建socket连接
        final ObjectInputStream ois=new
    ObjectInputStream(socket.getInputStream());
                                  //获取socket的对象输入流
        list.setModel(new AbstractListModel()
    {                                       //设置JList的数据模型
          //获取socket传递的数组对象作为列表控件的数据
          String[] values = (String[]) ois.readObject();
          public int getSize() {
            return values.length;
          }
```

```
        public Object getElementAt(int index) {
          return values[index];
        }
      });
    } catch (UnknownHostException e1)
{                                          //捕获未知的主机异常
      JOptionPane.showMessageDialog(null, "输入的主机无法连
  接");
      return;
    } catch (SocketException e1)
{                                              //捕获socket异
常
      JOptionPane.showMessageDialog(null, "输入的主机无法连
  接");
      return;
    } catch (IOException e11)
{                                                //捕获输入输
出异常
      e11.printStackTrace();
    } catch (ClassNotFoundException e1) {
      e1.printStackTrace();
    }
  }
```

（5）列表控件的事件处理方法将获取用户选择的列表项，把该选择转换为字符串传递给服务器，随后从服务器获取以压缩流传递过来的数据，并从压缩条目中获取它，然后显示在infoArea文本域中。程序关键代码如下：

```
    protected void do_list_valueChanged(ListSelectionEvent e)
{
      if (e.getValueIsAdjusting())
        return;
      Object value = list.getSelectedValue();
      if (value == null)
        return;
      String bookName = value.toString();
      infoArea.setText("");
      try {
        //获取socket的输出流
        OutputStream outputStream = socket.getOutputStream();
        //向socket发送信息
        outputStream.write((bookName +"\n").getBytes());
        //创建ZIP输入流
        ZipInputStream zis = new
    ZipInputStream(socket.getInputStream());
        char[]data=new
    char[1024];                                    //缓冲数组
        int readNum;
        zis.getNextEntry();
              //读取下一个ZIP条目
        //把ZIP二进制输入流转换为字符输入流
        InputStreamReader ir = new InputStreamReader(zis);
        while ((readNum= ir.read(data))>0)
    {                                  //读取数据
          //把数据添加到文本域控件中
```

```
            infoArea.append(new String(data, 0, readNum));
        }
        infoArea.select(0, 0);
    } catch (IOException e1) {
        e1.printStackTrace();
    }
}
```

举一反三

根据本实例，读者可以实现以下功能。

以压缩格式传输文件。

以压缩格式传输图片。

实例365 压缩远程文件夹

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\09\Ex09_365

实例说明

在前面的范例中，实现了备份网络文件夹的功能。原理是直接将网络文件夹复制到本地文件夹。这样虽然能实现需求，但是效果并不理想。如果能在传输的过程中使用压缩技术会更好，这样能提高文件的下载速度。本实例就实现了这样的功能。实例运行效果如图9.371所示。

图9.37 实例运行效果

技术要点

压缩文件和复制文件类似，只是用另外一种格式来保存输入流。本实例使用到了ZipOutputStream，它以ZIP文件格式写入文件实现输出流过滤器。本实例使用的方法如表9.39所示。

表9.39 ZipOutputStream的常用方法

方 法 名	作 用
ZipOutputStream(OutputStream out)	创建新的 ZIP 输出流
putNextEntry(ZipEntry e)	开始写入新的 ZIP 文件条目并将流定位到条目数据的开始处

实现过程

（1）继承JFrame编写一个窗体类，名称为FileCopyFrame。

（2）设计FileCopyFrame窗体类时用到的主要控件及说明如表9.40所示。

表9.40 窗体的主要控件及说明

控 件 类 型	控 件 命 名	控 件 用 途
JButton	targetButton	让用户选择保存压缩文件的文件夹
	downloadButton	实现备份和压缩功能
JTextField	sourceTextField	显示用户输入的 URI 地址
	targetTextField	显示用户选择的文件夹绝对路径

（3）编写监听单击“开始备份”按钮事件的方法do_downloadButton_actionPerformed()，它实现了将远程文件夹压缩到本地文件夹的功能。核心代码如下：

```
    protected void
do_downloadButton_actionPerformed(ActionEvent arg0) {
        String uri=
  sourceTextField.getText();                                    //
  获得用户输入的URI地址
        String target=
  targetTextField.getText();                                   //获
  得用户选择的保存压缩文件的路径
        try {
```

```
    File sourceFile=new File(new
URI(uri));                          //根据用户输入的URI创建
File对象
    //省略校验代码
    List<String>path=new ArrayList<String>
();                          //用列表保存网络文件夹中文件的地址
    getPath(sourceFile,
path);                                  //获得文件地址
    //根据用户选择的文件夹和网络文件夹的名字创建压缩文件
    File targetFile = new File(target +
sourceFile.getName() +".zip");
    zipFile(path, targetFile,
sourceFile.getAbsolutePath());              //实现压缩功
能
    JOptionPane.showMessageDialog(this, "文件夹压缩成
功");          //提示用户压缩成功
  } catch (URISyntaxException e) {
    e.printStackTrace();
  } catch (IOException e) {
    e.printStackTrace();
  }
}
```

举一反三

根据本实例，读者可以开发以下程序。

为远程压缩文件加密。

解压远程文件夹。

实例366 压缩存储网页

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\09\Ex09_366

实例说明

在前面的范例中，实现了下载单个网页的功能，本实例使用压缩技术来下载网页。对于由大量文本组成的网页，压缩技术能大幅度减少下载的流量，提高了网速。实例运行效果如图9.38所示。

图9.38 实例运行效果

技术要点

抽象类URLConnection是所有代表应用程序和URL之间的通信链接类的超类。此类的实例可用于读取和写入此URL引用的资源。通常，创建一个到URL的连接需要以下几个步骤：

（1）通过在URL上调用openConnection方法创建连接对象。

（2）处理设置参数和一般请求属性。

（3）使用connect方法建立到远程对象的实际连接。

（4）远程对象变为可用。远程对象的头字段和内容变为可访问。

为了获得需要下载的数据，需要使用getInputStream()方法，该方法的声明如下：

```
public InputStream getInputStream()throws IOException
```

实现过程

（1）继承JFrame编写一个窗体类，名称为DownloadFrame。

（2）设计DownloadFrame窗体类时用到的主要控件及说明如表9.417所示。

表9.41 窗体的主要控件及说明

控 件 类 型	控 件 命 名	控 件 用 途
JButton	chooseButton	实现选择压缩文件夹的功能
	button	实现下载和压缩功能
JTextField	urlTextField	显示用户输入的网址
	pathTextField	显示用户选择的文件夹绝对路径

（3）实现下载和压缩网页的方法zipWebPage()，参数url代表用户输入的网址，savePath代表用户选择的文件夹。代码如下：

```
    private static void zipWebPage(String url, String
savePath) throws IOException {
        URLConnection conn=new
    URL(url).openConnection();                    //利用用户输入的网
    址创建URL连接对象
        InputStream in=
    conn.getInputStream();                                  //获得输
    入流
        FileOutputStream fos = new FileOutputStream(savePath
    +"download.zip");
        ZipOutputStream zos = new ZipOutputStream(fos);
        byte[] buffer = new byte[1024];
        ZipEntry entry=new
    ZipEntry("download.html");                        //创建名为
    download.html的压缩条目
        zos.putNextEntry(entry);
        int read = 0;
```

```
    while ((read= in.read(buffer)) != -1)
{                                   //写入数据
      zos.write(buffer, 0, read);
    }
    zos.closeEntry();
    in.close();
      //释放资源
    zos.close();
    fos.close();
  }
```

举一反三

根据本实例，读者可以开发以下程序。

实现单个网页的下载功能。

批量压缩存储网页。

第10章 操作办公文档

操作Word

操作Excel

10.1 操作Word

实例367 把文本文件导入Word中

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\10\Ex10_367

实例说明

jacob是java-com bridge的缩写，它在Java与微软的com组件之间构建一座桥梁。使用jacob自带的DLL动态链接库，并通过JNI的方式实现了Java平台上对com程序的调用。jacob功能较强大，很多操作办公文档的程序员都使用 jacob 组件，本实例实现将文本文件中的内容导入Word文档中。实例运行效果如图10.1所示。

图10.1 实例运行效果

技术要点

jacob是一个开源项目，可以到网址http://sourceforge.net下载最近的jacob，本章中应用jacob的版本是1.9。下载jacob_1.9.1.zip后，进行解压，将得到的jacob.jar文件添加到项目的构建路径中，将jacob.dll放置在JDK的bin路径下，或者是系统的System32文件夹下，就做好了准备工作。

使用jacob组件新建Word文档，可分为以下几个主要步骤。

（1）通过创建Word运行程序启动对象ActiveXComponent来启动Word。实例代码如下：

```
ActiveXComponent word = new
ActiveXComponent("Word.Application");
```

上句代码为构建 ActiveX 组件实例，其中参数与需要调用的Activex 控制有关。常用的参数如表10.1所示。

表10.1 ActiveXComponent构造方法中的常用参数列表

MS 控制名	对应的参数值
InternetExplorer	InternetExplorer.Application
Excel	Excel.Application
Word	Word.Application
outlook	Outlook.Application
visio	Visio.Application

（2）设置Word文档为可视状态。通过调用ActiveXComponent对象的setProperty()方法实现，代码如下：

```
word.setProperty("Visible", new Variant(true));
```

其中参数Variant用于映射com的数据类型，提供Java和com的数据交互。

（3）调用Dispatch类的call()方法访问com/dll。call()方法提供了多种重载形式，来满足程序开发的需要，在本实例中通过给定参数 Add 创建一个新的 Word 文档，该方法的返回值为Variant对象。代码如下：

```
doc = Dispatch.call(documents, "Add").toDispatch();
```

（4）调用Dispatch类的get()方法，读取com对象的属性值。代码如下：

```
selection = Dispatch.get(word, "Selection").toDispatch();
```

实现过程

（1）新建项目，并在项目中创建类WordBean。

（2）在该类中定义Word文档对象、Word运行程序对象和Dispatch对象的代码如下：

```
    private Dispatch
doc;                                                    //Word文档
    private ActiveXComponent
word;                                                  //Word运行程序对象
    private Dispatch
documents;                                              //所有Word
文档集合
    private Dispatch
selection;                                              //选定的范
围或插入点
```

（3）在WordBean类的构造方法中启动Word，设置Word为可视状态，并读取文档的属性值。代码如下：

```
    public WordBean() {
      if (word == null) {
        word=new
    ActiveXComponent("Word.Application");                  //启动
    Word
        word.setProperty("Visible",new
    Variant(true));                        //设置Word为可视状态
      }
      if (documents == null)
        documents=word.getProperty("Documents").toDispatch();
              //读取属性值
    }
```

（4）在WordBean类中创建方法createNewDocument()，实现新建Word文档，代码如下：

```
public void createNewDocument() {
  doc=Dispatch.call(documents,
"Add").toDispatch();                    //调用com/dll对象
  selection=Dispatch.get(word,
"Selection").toDispatch();                  //读取com对象的属性值
  }
```

（5）在WordBean类中创建insertText()方法，实现向Word文档中添加新的字符串。该方法有一个String类型的参数。代码如下：

```
public void insertText(String newText) {
  Dispatch.put(selection,
"Text",newText);                          //设置属性值
  }
```

（6）在WordBean类中创建save()方法，用于将新建的Word文档保存到其他地址中，该方法包含String类型的参数，用于指定文件的保存地址。代码如下：

```
public void save(String savePath) {
  Dispatch.call((Dispatch) Dispatch.call(word,
"WordBasic").getDispatch(), "FileSaveAs", savePath);
  }
```

举一反三

根据本实例，读者可以开发以下程序。

将文本文件导入Word中。

将Word中的内容导入文本文件。

实例368 浏览本地Word文件

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\10\Ex10_368

实例说明

jacob组件是操作办公软件很强大的工具，本实例为大家介绍的是，使用jacob组件实现打开本地磁盘中的 Word 文档。在本实例中为大家提供了可浏览本地系统的树，用户可选择打开本地系统中任意的Word文档。实例运行效果如图10.2所示。

图10.2 实例运行效果

技术要点

本实例实现的是使用jacob技术来打开Word文档，具体地说是通过使用Dispatch类的call()方法和给定命令Open实现打开Word文档。call()方法的返回值为Dispatch对象，实例代码如下：

```
doc=Dispatch.call(documents,
"Open",docPath).toDispatch();                    //调用打开Word文档命令
```

参数说明

● documents：Dispatch 对象。

● docPath：打开文件的地址。

例如，要打开C盘根目录下的w.doc文件，代码如下：

```
doc=Dispatch.call(documents, "Open",
"c://w.doc").toDispatch();                    //调用打开
Word文档命令
```

实现过程

（1）创建类FileHeald，在该类中保存有打开Word文档的方法。首先在该类中定义Dispatch类、ActiveXComponent类对象。代码如下：

```
    private Dispatch doc;
    private ActiveXComponent
word;                                          //Word运行程序对
象
    private Dispatch
documents;                                          //所有
Word文档集合
```

（2）在FileHeald类的构造方法中，对创建的Dispatch对象、ActiveXComponent类进行实例化。代码如下：

```
    public FileHeald() {
      if (word == null) {
        word=new
    ActiveXComponent("Word.Application");                  //
    启动Word
        word.setProperty("Visible",new
    Variant(true));                          //设置Word为可视状
    态
```

```
    }
    if (documents == null)
      documents=word.getProperty("Documents").toDispatch();
                    //读取属性值
  }
```

（3）在FileHeald类中创建方法openDocument()，打开Word文档，该方法有一个String类型的参数，用于指定要打开的文档的地址。代码如下：

```
  public void openDocument(String docPath) {
    doc=Dispatch.call(documents,
"Open",docPath).toDispatch();                    //调用打开Word
文档命令
  }
```

（4）在项目中创建类OpenWord，该类继承JFrame类且是一个窗体类。在该类中添加用于显示本地磁盘结构的树控件、列表控件与按钮控件，如表10.2所示。

表10.2 窗体中的控件

控 件 类 型	控 件 命 名	控 件 用 途
JTree	jtree	用于显示本地磁盘结构的树控件
JList	list	用于显示 doc 文档地址的列表控件
JButton	open	显示“打开”的按钮控件

（5）在“打开”按钮的单击事件中，调用FileHeald类的打开Word文档的方法，并以用户选择的文件列表的文件地址作为该方法的参数。代码如下：

```
  open.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
      String path=
  list.getSelectedValue().toString();
```

```
    //获取用户选择的列表项内容
      FileHeald fileHeald=new
FileHeald();                                    //创建FileHeald
对象
      fileHeald.openDocument(path);
              //调用打开文件的方法
    }
});
```

举一反三

根据本实例，读者可以实现以下功能。

显示本地磁盘的结构。

显示.doc文档地址的列表。

实例369 将员工表插入Word文档中

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\10\Ex10_369

实例说明

一些软件可能会要求程序员将相应的数据内容以表格的形式添加到 Word 文档中，来方便数据的移植。本实例是使用jacob组件来实现以表格的形式向Word文档中插入内容。实现该实例时需要注意，要向表格中的指定列与指定行分别添加数据。通过本实例向 Word 中添加的表格内容如图10.3所示。

图10.3 向Word 中添加的表格内容

技术要点

实现本实例仍然是通过Dispatch类的get()方法与call()方法。get()方法获取属性信息，该方法包含多个重载形式，本实例用到的该方法的语法格式如下：

```
Dispatch cols = Dispatch.get(this.table,
"columns").toDispatch();
```

其中的columns 就是属性名称，类似于VB 的cols = table.columns。

call()方法用于访问com/dll对象，本实例应用的该方法的语法格式如下：

```
Dispatch table = Dispatch.call(tables, "add".new
Variant(1));
```

其中第一个参数是对象名；第二个字符串参数就是方法名字，实现向表格中添加内容。

实现过程

（1）创建类 Inerttable，在该类的主方法中启动 Word，代码参考光盘，篇幅有限，这里不再赘述。

（2）在该类中创建createTable()方法，该方法为创建表格方法。该方法有两个int类型的参数，分别用于指定创建表所拥有的行数与列数。代码如下：

```
public void createTable(int numCols, int numRows) {
  Dispatch tables=Dispatch.get(doc,
"Tables").toDispatch();                    //获取表格属性
  Dispatch range=Dispatch.get(selection,
"Range").toDispatch();                     //获取表格行列属性
  Dispatch newTable=Dispatch.call(tables, "Add",
range,new Variant(numRows),new
Variant(numCols)).toDispatch();            //向表格中添加内容
  Dispatch.call(selection, "MoveRight");
}
```

（3）在该类中编写putTxtToCell()方法，实现向表格中添加内容。代码如下：

```
public void putTxtToCell(int tableIndex, int cellRowIdx,
int cellColIdx, String txt) {
  Dispatch tables=Dispatch.get(doc,
"Tables").toDispatch();                    //获取表格属性
  Dispatch table=Dispatch.call(tables, "Item",new
Variant(tableIndex)).toDispatch();
                                           //要填充的表格
```

```
    Dispatch cell = Dispatch.call(table, "Cell", new
  Variant(cellRowIdx),new Variant(cellColIdx)).toDispatch();
    Dispatch.call(cell, "Select");
    Dispatch.put(selection, "Text",
  txt);                                          //使用put()
  方法设置表格内容
  }
```

（4）在类的主方法中调用 createTable()方法与 putTxtToCell()方法，实现创建表格，并向表格中添加内容。代码如下：

```
    Inerttable msWordManager=new
Inerttable();                                    //创建本类对
象
    try {
      msWordManager.createNewDocument();
                  //新建文档
      msWordManager.createTable(5,5);
                  //创建5行5列的表格
      msWordManager.putTxtToCell(1,1,1, "编
  号");                                  //向第1行第1列中添加内
  容
      msWordManager.putTxtToCell(1,2,1,
  "1");                                      //向第2行第1列中添
  加内容
      msWordManager.putTxtToCell(1, 1, 2, "姓名");
      msWordManager.putTxtToCell(1, 2, 2, "李四");
      msWordManager.putTxtToCell(1, 1, 3, "年龄");
```

```
        msWordManager.putTxtToCell(1, 2, 3, "30");
        msWordManager.putTxtToCell(1, 1, 4, "性别");
        msWordManager.putTxtToCell(1, 2, 4, "男");
        msWordManager.putTxtToCell(1, 1, 5, "学历");
        msWordManager.putTxtToCell(1, 2, 5, "本科");
        msWordManager.save("c:\\向word中绘制表
格.doc");                                    //调用保存文档的方法
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        msWordManager.close();
    }
```

举一反三

根据本实例，读者可以实现以下功能。

将表中信息插入Word文档中。

将相片插入Word文档中。

实例370 将员工照片插入Word简历

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\10\Ex10_370

实例说明

包含图片的 Word 文档是非常常见的，如带有图片的员工表。本实例实现向员工表中插入图片信息，在插入图片时需要注意，默认的 Word 文档的插入点在文档的开始部分，要根据表格的情况相应地移动插入点。实例运行效果如图10.4所示。

技术要点

实现本实例仍然是通过 Dispatch 类的 get()与 call()方法。通过给定参数可实现向文档中插入图片。代码如下：

```
Dispatch.call(Dispatch.get(selection,
"InLineShapes").toDispatch(), "AddPicture", imagePath);
```

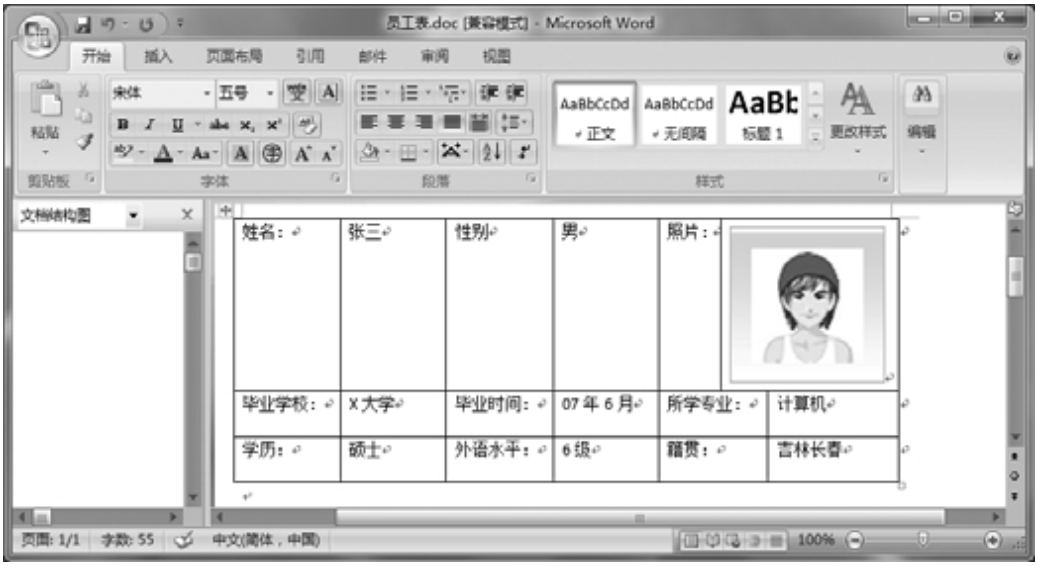


图10.4 向Word 中添加图片

参数说明

●selection：Dispatch对象。

●imagePath：要向文档中插入图片的地址。

实现过程

（1）创建类WordBean，在该类中包含新建Word文档、保存Word文档等方法。在前面的实例中，已经向大家做了介绍，这里不再赘述。

（2）在该类中包含一个向当前插入点插入图片的方法，该类包含两个String类型参数，分别用于指定要插入文档的图片的地址与插入文档的地址；一个 int 类型的参数，用于指定插入点向右移动的位置。具体代码如下：

```
public void insertImage(String imagePath, String docPath,
int pos) {
```

```
    doc=Dispatch.call(documents,
"Open",docPath).toDispatch();                    //打开相应的
Word文档
    selection = Dispatch.get(word,
"Selection").toDispatch();
    for (int i = 0; i < pos; i++)
      Dispatch.call(selection,
  "MoveRight");                                  //将插入点向右
  移动相应的位置
    Dispatch.call(Dispatch.get(selection,
"InLineShapes").toDispatch(),"AddPicture",
imagePath);                                      //向文档中插
入图片
  }
```

举一反三

根据本实例，读者可以开发以下程序。

将多张图片批量插入Word文档中。

将Word文档中的信息插入Excel表格中。

实例371 将Word文档保存为HTML格式

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\10\Ex10_371

实例说明

笔者曾遇到过一个挺麻烦的问题，程序运行后生成了Word文档，但是有的客户的机器上没有安装Word，无法打开程序生成的Word。无奈之下，编写了一个将Word文档转换为HTML格式的文件解决了这一问

题。本实例实现将实例369生成的Word文件转换为HTML格式，转换后的结果可通过IE浏览器打开，如图10.5所示。

技术要点

实现本实例主要是通过Dispatch类的invoke()方法，该方法的语法如下：

Dispatch.invoke(a1,"a3",a4,a5,a6).toDispatch();

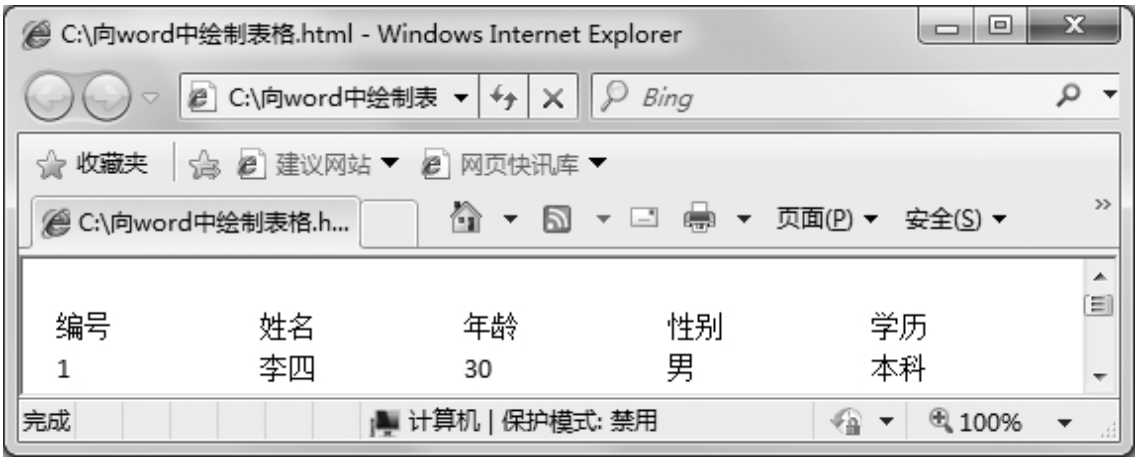


图10.5 实例运行效果

该方法实现了功能调用，作用为对于Dispatch对象a1的a3属性执行a4，类似于Dispatch类的get()方法，执行操作后a3 的值为a5，a6为错误参数码，通常将其定义为“new int[1]”。

实现过程

（1）创建类WordToHtml，在该类中定义wordToHtml()方法，实现将Word文档转换为HTML格式。该方法包含两个String类型的参数，分别指定Word文档的存放路径与转换后HTML文件的存放路径。代码如下：

```
    public void wordToHtml(String docfilePath, String
htmlfilePath) {
        ActiveXComponent app=new
  ActiveXComponent("Word.Application");          //启动Word
        try {
```

```
        app.setProperty("Visible",new
    Variant(false));                          //设置Word为不可
    见
        Dispatch dispatch=
    app.getProperty("Documents").toDispatch();          //
    读取文档属性值
        Dispatch doc =
    Dispatch.invoke(dispatch,"Open",Dispatch.Method, new
    Object[] { docfilePath, new Variant(false),new
    Variant(true) },new
    int[1]).toDispatch();                     //功能调用
        Dispatch.invoke(doc, "SaveAs", Dispatch.Method, new
    Object[] {htmlfilePath,new Variant(8) },new
    int[1]);                        //以HTML格式保存到临时文件
        Variant f = new Variant(false);
        Dispatch.call(doc, "Close",
    f);                                       //将文档关闭，并将
    其设置为不可见
      } catch (Exception e) {
        e.printStackTrace();
      }
    }
```

（2）在该类的主方法中调用wordToHtml()方法，实现指定磁盘文件转换为HTML格式。代码如下：

```
    public static void main(String[] args) {
      WordToHtml wth=new
  WordToHtml();                                   //创建本类对象
```

```
wth.wordToHtml("c:\\向word中绘制表格.doc", "c:\\向word中绘制表格.html");
}
```

举一反三

根据本实例，读者可以实现以下功能。

将Word文档保存成Excel文件格式。

将Word文档保存成TXT文件格式。

10.2 操作Excel

实例372 将员工信息保存到Excel表中

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\10\Ex10_372

实例说明

相信读者对Excel（电子表格）一定不会陌生，但是使用Java语言操纵Excel文件并不是一件容易的事。需要应用第三方组件，如有的系统客户要求通过程序向 Excel 表中插入数据。本实例实现的是将用户添加的信息保存到Excel表中，实例运行效果如图10.6所示。

图10.6 新建的Excel 文件

技术要点

POI组件的首页是http://Jakarta.apache.org。读者可以自行下载最高版本。解压缩*.zip，把poi.jar文件放置在项目的构建路径

中，即可实现POI组件操作Excel。POI项目实现的Excel文件格式称为HSSF，是Horrible SpreadSheet Format 的缩写。可以使用纯Java 代码读取、写入、修改Excel文件。POI项目操作Excel需要以下几个重要的类。

□ HSSFWorkbook 类

该类表示Excel工作簿类。创建该类对象，表示新建Excel工作簿。

□ HSSFSheet 类

该类表示 Excel 工作表，可通过为构造方法指定参数来创建指定名称的工作表。例如，创建名称为工资表的工作表代码如下：

```
HSSFSheet sheet = excelbook.createSheet("工资表");
```

□ Cells 类

该类表示Excel文件中的一个单元格。

实现过程

（1）在项目中创建类InsertExcelFrame，该类继承JFrame类，实现窗体类。

（2）向窗体中添加控件，实现窗体布局。该窗体中的主要控件及说明如表10.3所示。

表10.3 窗体中的主要控件及说明

控 件 类 型	控 件 命 名	控 件 用 途
JTextField	nameTextField	为用户提供添加“姓名”的文本框
	sexTextField	为用户提供添加“性别”的文本框
	ageTextField	为用户提供添加“年龄”的文本框
	deptTextField	为用户提供添加“部门”的文本框
	jobTextField	为用户提供添加“职位”的文本框
	laborageTextField	为用户提供添加“工资”的文本框
JButton	insertButton	为用户提供“添加到 Excel”的按钮

（3）创建工具类CreateXL，该类定义了操作Excel表的相关方法。其中新建Excel表的方法是createExcel()，关键代码如下：

```
public void createExcel() {
  try {
    excelbook=new HSSFWorkbook();
    sheet= excelbook.createSheet("工资
表");                          //在索引0的位置创建行（最顶
端的行）
    HSSFRow row = sheet.createRow((short) 0);
    HSSFCell monadism= row.createCell((short)
0);                    //在索引0的位置创建单元格（左上端）
    monadism.setCellType(HSSFCell.CELL_TYPE_STRING);
        //定义单元格为字符串类型
    monadism.setCellValue("姓
名");                                  //在单元格中输入一些内
容
    row.createCell((short) 1).setCellValue("性
别");                      //在第1行第2列添加内容
    row.createCell((short) 2).setCellValue("年龄");
    row.createCell((short) 3).setCellValue("部门");
    row.createCell((short) 4).setCellValue("职位");
    row.createCell((short) 5).setCellValue("工资信息");
    FileOutputStream out=new
FileOutputStream(outputFile);          //新建输出文件流
    excelbook.write(out);
      //把相应的Excel工作簿存盘
    out.flush();
    out.close();
      //操作结束，关闭文件
```

```
            System.out.println("文件创建成功！！！");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

（4）在CreateXL类中定义insertvalue()方法，用于向Excel表中插入数据，该方法包含有String类型的参数，用于指定插入表的内容。关键代码如下：

```
    public void insertvalue(String name, String sex, String age, String dept,String job, String laborage) {
        try {
            excelbook=new HSSFWorkbook(new FileInputStream(outputFile));    //定义Excel表对象
            HSSFSheet sheet= excelbook.getSheet("工资表");                //获取指定的工作表
            int count= sheet.getPhysicalNumberOfRows();                    //获取工作表中总的行数
            HSSFRow row= sheet.createRow((short) count);                    //新建一行
            row.createCell((short) 0).setCellValue(name);                    //在索引0的位置创建单元格（左上端）
            row.createCell((short) 1).setCellValue(sex);
            row.createCell((short) 2).setCellValue(age);
            row.createCell((short) 3).setCellValue(dept);
            row.createCell((short) 4).setCellValue(job);
```

```
        row.createCell((short) 5).setCellValue(laborage);
        FileOutputStream
out;                                                    //新建输出文件
流
        out = new FileOutputStream(outputFile);
        excelbook.write(out);
          //把相应的Excel工作簿存盘
        out.flush();
      } catch (Exception e) {
        e.printStackTrace();
      }
}
```

举一反三

根据本实例，读者可以实现以下功能。

批量将员工信息保存到Excel表中。

修改Excel表中的数据。

实例373 通过Excel公式计算出商品表中的总售价

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\10\Ex10_373

实例说明

对 Excel 熟悉的人都知道，Excel 中有大量的公式来满足用户的需要。那么如何通过 Java程序在Excel表中使用公式呢？通过POI组件实现向Excel中写文件，可以应用公式。这样可以满足很多程序的需求。本实例通过Java程序实现在磁盘中写数据，并使用公式。实例运行效果如图10.7所示。

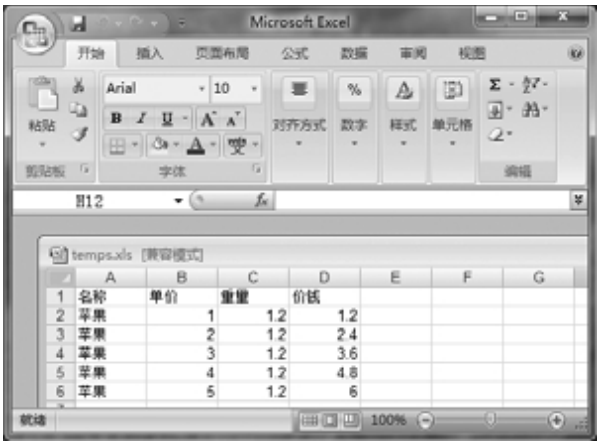


图10.7 将数据库表内容写入Excel 中

技术要点

在Excel表中的指定单元格内使用表格，可以使用setCellFormula()方法。代码如下：

```
HSSFCell cell = row.createCell((short)4);
Cell.setCellFormula("sum(B1:B2)");
```

这样设置之后，得到的cell单元格中的值就为B1和B2的列值相加。

实现过程

创建类 CreateExcelUseformula，在该类中定义创建 Excel 表格，并在插入表格数据时使用公式方法。具体代码如下：

```
public static void main(String[] args) {
  try {
    /** Excel 文件要存放的位置，假定在C盘根目录下*/
    String outputFile ="c://temps.xls";
    //创建新的Excel 工作簿
    HSSFWorkbook excelbook = new HSSFWorkbook();
    //如要新建一名为“工资表”的工作表，其语句为：
    HSSFSheet sheet = excelbook.createSheet("工资表");
    //在索引0的位置创建行（最顶端的行）
```

```
    HSSFRow row = sheet.createRow((short) 0);
    //在索引0的位置创建单元格（左上端）
    HSSFCell monadism = row.createCell(0);
    //定义单元格为字符串类型
    monadism.setCellType(HSSFCell.CELL_TYPE_STRING);
    //在单元格中输入一些内容
    monadism.setCellValue("名称");
    //在第1行第2列添加内容
    HSSFCell cell1 = row.createCell(1);
    cell1.setCellValue("单价");
    row.createCell(2).setCellValue("重量");
    row.createCell(3).setCellValue("价钱");
    for (int i=1; i<=5; i++)
{                                          //通过for循环创建
表格
      HSSFRow row2=
  sheet.createRow(i);                              //在工作
  簿中新建一行
      row2.createCell(0).setCellValue("苹
  果");                          //在工作簿中新建一列
      row2.createCell(1).setCellValue(i);
                //设置单元格值
      row2.createCell(2).setCellValue(1.2);
      row2.createCell(3).setCellFormula("B"+ (i+1)+ "*C"+
  (i+1)+ "");                        //为单元格添加公式
    }
```

```
        FileOutputStream out=new
    FileOutputStream(outputFile);                    //新建输出文
    件流
        excelbook.write(out);
              //把相应的Excel工作簿存盘
        out.flush();
        out.close();
              //操作结束，关闭文件
        System.out.println("文件创建成功！！！");
      } catch (Exception e) {
        e.printStackTrace();
      }
    }
```

举一反三

根据本实例，读者可以实现以下功能。

通过Excel公式计算单个商品的总价。

通过Excel公式计算商品的销量。

实例374 将数据库表中的内容写入到Excel

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\10\Ex10_374

实例说明

数据库表与 Excel 的存储形式类似，都是以表格的形式存储的。很多软件要求将数据库表与 Excel 进行交互。本实例实现的是将数据库表中的内容写入到 Excel 中，仍然使用的是 POI组件，写入后Excel文件中的内容如图10.8所示。

图10.8 将数据库表中的内容写入到Excel

技术要点

实现本实例，需要根据数据库表格式创建 Excel 文件，再将从数据库表中读取的内容写入到Excel表中，向Excel表中写内容主要通过HSSFCell类的setCellValue()方法。具体语法如下：

cell.setCellValue(value)

参数说明

● cell：HSSFCell 对象。

● value：向Excel 单元格中插入的内容。

实现过程

（1）创建ExcelWriter类，在该类的构造方法中创建一个Excel文件写入器，将所有数据写入指定的Excel文件中。具体代码如下：

```
/**
 * . *创建一个Excel文件写入器，所有数据将写入指定的.xls文件中；
 * . *写入日期时以指定格式形式写入
 * . *
 * . *@param xlsFile 指定目标文件位置，原位置已存在同名文件将被覆盖
```

```
    * . *@param dateFormat 日期格式，如果为null则按本地日期格式
    * .
    */
  public ExcelWriter(File xlsFile, String dateFormat) {
    this.xlsFile = xlsFile;
    workbook = new HSSFWorkbook();
    try {
      FileOutputStream fileoUut = new
  FileOutputStream(xlsFile);
      workbook.write(fileoUut);
      fileoUut.close();
    } catch (Exception e) {
      e.printStackTrace();
  }
  if(dateFormat!=null){
    dateFormatter=new SimpleDateFormat(dateFormat);
  }else{
    dateFormatter=new SimpleDateFormat();
  }
  }
```

（2）在该类中定义写入数据库结果集的列名及数据库到指定工作簿方法writeSheet()，该类有一个File类型的参数，用于指定写入的Excel表格；一个ResultSet类型的参数，用于指定数据库结果集。具体代码如下：

```
  private void writeSheet(File file, ResultSet resultSet)
throws SQLException {
```

```
    HSSFWorkbook book=new
HSSFWorkbook();                          //定义工作簿对象
    HSSFSheet sheet=book.createSheet("员工
表");                                    //创建工作表
    ResultSetMetaData metaData = resultSet.getMetaData();//
获取关于 ResultSet对象中列的类型和属性信息的对象
    int rowNum = 0;
    HSSFRow header=
sheet.createRow(rowNum);                          //写入列名
    int
colCount=metaData.getColumnCount();
//获取数据库表中共有几列
    for (int i=0; i< colCount; i++)
{                                        //循环遍历数据表列名
      HSSFCell
  cell=header.createCell(i);                                  //根
  据数据库内容创建单元格
      writeCell(cell,metaData.getColumnLabel(i+1));
            //将数据库中的内容写入单元格内
    }
    while (resultSet.next())
{                                        //循环遍历查询结果集
      rowNum++;
      HSSFRow row=
  sheet.createRow(rowNum);                        //创建一行
      for (int i = 0; i < colCount; i++) {
```

```
            HSSFCell cell=
        row.createCell(i);                          //新建单元格
            writeCell(cell,
        resultSet.getObject(i+1));                  //将结果集中的
        内容写入单元格中
          }
        }
        try {
          FileOutputStream file0=new
      FileOutputStream(file);          //创建FileOutputStream对象
          book.write(file0);
          file0.close();
        } catch (Exception e) {
          e.printStackTrace();
        }
      }
```

（3）在该类中定义writeCell()方法，用于将数据写入指定的单元格。具体代码如下：

```
    private void writeCell(HSSFCell hssFcell, Object object)
{
    if (object instanceof Date)
{                                   //判断要写入的数值是否为日期
类型
      Date d = (Date) object;
      hssFcell.setCellValue(new
  HSSFRichTextString(dateFormatter.format(d)));
                        //日期以文本形式写入
```

```
    } else if (object instanceof Boolean)
{                          //判断要写入的数值是否为布尔类型
      boolean b = (Boolean) object;
      hssFcell.setCellValue(b);
  //向表格中写入数据
    } else if (object instanceof Number)
{                          //判断要写入的数据是否为数值类型
      double d = ((Number) object).doubleValue();
      hssFcell.setCellValue(d);
  //向表格中写入数据
    } else {
      String s = (String) object;
      hssFcell.setCellValue(new HSSFRichTextString(s));
    }
    }
```

（4）在该类中定义查询数据库，获取数据库结果集方法getRest()。具体代码如下：

```
    public ResultSet getRest() {
      try {
        Class.forName("net.sourceforge.jtds.jdbc.Driver");
      } catch (ClassNotFoundException e) {
        e.printStackTrace();
      }
    Connection conn = null;
      //定义连接数据库的url
      String url
  ="jdbc:jtds:sqlserver://localhost:1433;DatabaseName=db_data
```

```
base21";
    String userName=
"sa";                                    //连接数据库的用户名
    String passWord=
"";                                      //连接数据库的密码
    try {
      conn=DriverManager.getConnection(url,userName,passWor
  d);      //获取数据库连接
    } catch (SQLException e) {
      e.printStackTrace();
    }
    ResultSet rest = null;
    String sql= "select* from
tb_emp";                                 //定义查询的SQL语句
    Statement statement;
    try {
      statement=
  conn.createStatement();                                //创建
  Statement实例
      rest=
  statement.executeQuery(sql);                                /
  /执行SQL语句
    } catch (SQLException e) {
      e.printStackTrace();
    }
    return rest;
  }
```

举一反三

根据本实例，读者可以实现以下功能。

将Excel表中的内容保存到数据库。

修改数据库表中的信息。

实例375 将Excel表中的内容保存到数据库

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\10\Ex10_375

实例说明

本实例实现的是通过POI组件读取Excel表内容，再将读取的内容写入数据库中。本实例将实例374中生成的Excel文件写入数据库表tb_empTable中。写入后该数据库表中的数据如图10.9所示。

id	name	sex	age	laborage
16	张三	男	30.0	5000.0
17	李四	男	26.0	3500.0
18	孙宇	女	25.0	2800.0
19	杜军	男	29.0	3500.0

图10.9 将Excel 表中的内容保存到数据库

技术要点

HSSF为读取操作提供了两类API，即usermodel（用户模型）和eventusermodel（事件用户模型）。Usermodel包把Excel文件映射成熟悉的结构，如WorkBook、Sheet、Row、Cell等，把整体结构以一组对象的形式保存在内存中。

实现过程

（1）定义JDBCUtil类，在该类中定义获取数据库连接的方法，读者可参考光盘中的源程序。除此之外，还有向数据库表中添加数据的方法insertEmp()。该方法的具体代码如下：

```
public void insertEmp(String[] str) {
```

```
    JDBCUtil iteacher=new
JDBCUtil();                                    //创建本类对象
    Connection conn=
iteacher.getConn();                                    //调用
获取数据库连接的方法
    //定义向数据库插入数据的SQL语句
    String sql ="insert into tb_empTable values('"+ str[0]
+"','"+ str[1] +"','"+ str[2] +"','"+ str[3] +"')";
    try {
      Statement statement = conn.createStatement();
      statement.executeUpdate(sql);
              //执行插入的SQL语句
    } catch (SQLException e) {
      e.printStackTrace();
    }
  }
```

（2）创建ReadToDateBase类，该类实现将C盘的temp.xls读取到数据库表中。具体代码如下：

```
  String fileToBeRead ="c:\\temp.xls";
  try {//创建对Excel工作簿文件的引用
    HSSFWorkbook workbook = new HSSFWorkbook(new
FileInputStream(fileToBeRead));
    HSSFSheet sheet=workbook.getSheet("员工
表");                          //创建对工作表的引用
    int rows=
sheet.getPhysicalNumberOfRows();
  //获取表格的行数
```

```
    for (int r=1; r< rows; r++)
{                                                    //循环遍历表格的行
      String
  value="";                                                        //
  定义保存读取内容的String对象
      HSSFRow row=
  sheet.getRow(r);                                          //获取
  单元格中指定的行对象
      if (row != null) {
        int  cells=
    row.getPhysicalNumberOfCells();                              //
    获取单元格中指定的列对象
          for (short c=1; c< cells; c++)
      {                                  //循环遍历单元格中的列
            HSSFCell cell= row.getCell((short)
        c);                          //获取指定单元格中的列
            if (cell != null) {
              //判断单元格的值是否为字符串类型
              if (cell.getCellType() ==
          HSSFCell.CELL_TYPE_STRING) {
                value += cell.getStringCellValue()+",";
              //判断单元格的值是否为数字类型
              } else if (cell.getCellType() ==
          HSSFCell.CELL_TYPE_NUMERIC) {
                value += cell.getNumericCellValue()+",";
              //判断单元格的值是否为布尔类型
```

```
            } else if(cell.getCellType() ==
          HSSFCell.CELL_TYPE_BOOLEAN){
                value += cell.getStringCellValue()+",";
            }
          }
        }
      }
      String []
str=value.split(",");
  //将字符串进行分割
      util.insertEmp(str);
              //调用向数据库插入数据的方法
  }
} catch (Exception e) {
  e.printStackTrace();
}
```

举一反三

根据本实例，读者可以实现以下功能。

实现批量保存。

修改表里单元格中的数据。

实例376 将Excel文件转换为HTML格式

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\10\Ex10_376

实例说明

如果在电脑上没有安装Office，就无法浏览Excel文件，将Excel文件转换成HTML文件会方便查阅。本实例实现将Excel文件转换为HTML格式，转换后的文件可以使用IE浏览器打开，如图10.10所示。

图10.10 将Excel 文件转换为HTML 格式

技术要点

本实例使用的是 jacob 组件实现将 Excel 文件转换为 HTML 格式，具体地说，使用的是Dispatch类的invoke()方法。

实现过程

（1）在项目中创建类ExcelToHtml，在该类中定义excelToHtml()方法，该方法用于将Excel文件转换为HTML格式。该方法包含两个String类型的参数，分别用于指定转换前Excel文件的保存地址和转换后HTML文件的保存地址。代码如下：

```
    public void excelToHtml(String xlsfilePath, String
htmlfilePath) {
      ActiveXComponent app=new
  ActiveXComponent("Excel.Application");        //启动Excel
      try {
        app.setProperty("Visible",new
    Variant(false));                            //设置Excel对象为
```

```
不可见
    Dispatch excels =
app.getProperty("Workbooks").toDispatch();
    Dispatch excel = Dispatch.invoke(
      excels, "Open", Dispatch.Method,
new Object[] { xlsfilePath, new Variant(false),new
Variant(true) },new int[1]).toDispatch();          //功能
调用
    Dispatch.invoke(excel, "SaveAs", Dispatch.Method, new
Object[] {
    htmlfilePath,new Variant(44) },new
int[1]);                          //以HTML格式保存到临时
文件
    Variant f = new Variant(false);
    Dispatch.call(excel, "Close",
f);                                    //关闭Excel文件
  } catch (Exception e) {
    e.printStackTrace();
  }
}
```

（2）在该类的主方法中调用excelToHtml()方法，实现将指定的Excel文件转换为HTML格式。具体代码如下：

```
public static void main(String[] args) {
  ExcelToHtml eth=new
ExcelToHtml();                          //创建本类对象
  eth.excelToHtml("c:\\JAVA 周计划及完成情况报表.xls",
"c:\\JAVA 周计划及完成情况报表.html");
```

```
  //调用将Excel转换为HTML格式的方法
}
```

举一反三

根据本实例，读者可以实现以下功能。

将图片转换成字节流写入数据库。

将图片所在的路径写入数据库。

第11章 JFreeChart图表

绘制柱形图
绘制饼图
绘制折线图
绘制时序图

11.1 绘制柱形图

实例377 绘制简单柱形图

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\11\Ex11_377

实例说明

本实例使用JFreeChart的简单柱形图绘制2010年上半年的销售情况，x轴（即横轴）表示销售月份，y轴（即竖轴）表示销售数量，运行效果如图11.1所示。

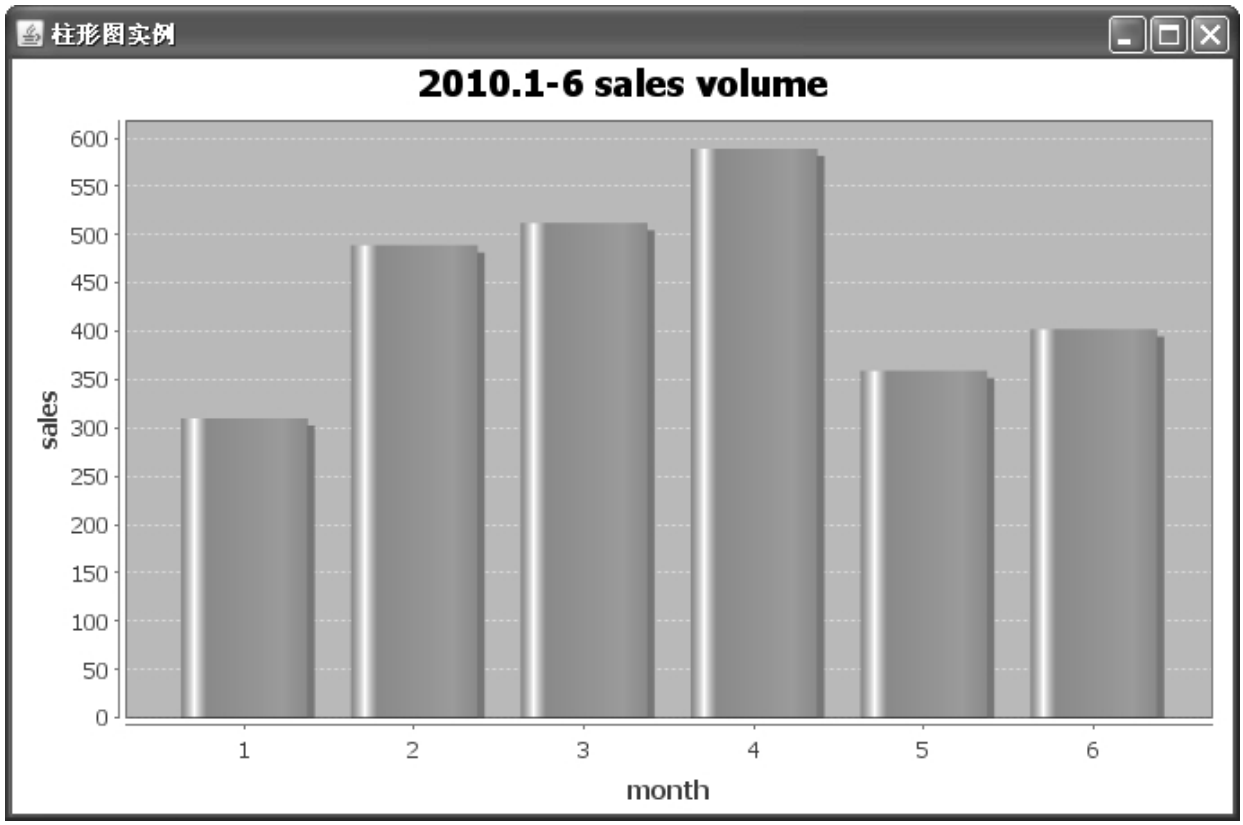


图11.1 简单柱形图

技术要点

（1）DatasetUtilities类可以创建一个简单的柱形图数据集，其方法如下：

public static CategoryDataset createCategoryDataset(Comparable rowKey, KeyedValues rowData)

参数说明

- rowKey：x 轴的含义，可以作为图示。
- rowData：数据集的内容，可以使用DefaultKeyedValues 类创建数据集。

（2）DefaultKeyedValues 类可以为柱形图提供一个数据集的操作，为数据集添加数据的方法如下：

addValue(Comparable key, double value)

参数说明

- key：x 轴的名称。
- value：与x 轴名称相对应的数值。

（3）JFreeChart 提供了创建普通柱形图的方法，创建完成后会返回一个 JFreeChart 对象， createBarChart()方法的语法如下：

public static JFreeChart createBarChart(String title, String categoryAxisLabel, String valueAxisLabel, CategoryDataset dataset, PlotOrientation orientation, boolean legend, boolean tooltips, boolean urls)

参数说明

- title：柱形图的标题。
- categoryAxisLabel：柱形图类别标签，即x 轴的名称。
- valueAxisLabel：柱形图数据标签，即y 轴的名称。
- dataset：柱形图的数据集合。
- orientation：柱形图的图表方向。
- legend：表示是否使用图示。

- tooltips：表示是否生成工具栏提示。
- urls：表示是否生成URL 链接。

实现过程

（1）新建一个JAVA文件，同时继承JFreeChart的ApplicationFrame类。

（2）创建 getCategoryDataset()方法，在方法内部创建一个适合普通柱形图的数据集合，代码如下：

```
private CategoryDataset getCategoryDataset() {
  DefaultKeyedValues keyedValues = new
DefaultKeyedValues();
  //添加数据
  keyedValues.addValue("1", 310);
  keyedValues.addValue("2", 489);
  keyedValues.addValue("3", 512);
  keyedValues.addValue("4", 589);
  keyedValues.addValue("5", 359);
  keyedValues.addValue("6", 402);
  //创建数据集合实例
  CategoryDataset dataset =
DatasetUtilities.createCategoryDataset("java book",
keyedValues);
  return dataset;
}
```

（3）创建getJFreeChart()方法，在方法里获取数据集，通过数据集创建柱形图的JfreeChart对象，代码如下：

```
private JFreeChart getJFreeChart() {
  CategoryDataset dataset = getCategoryDataset();
```

```
    JFreeChart chart=ChartFactory.createBarChart("2010.1-6
  sales volume",        //图表标题
      "month",                                    // x轴标签
      "sales",                                    // y轴标签
      dataset,                                    //数据集
      PlotOrientation.VERTICAL,                   //图表方
    向：垂直
      false,                                      //是否显示图
    例（对于简单的柱状图必须是false）
      false,                                      //是否生成工
    具
      false                                      //是否生成URL
    链接
      );
    return chart;
  }
```

（4）创建createPlot()方法，在方法里获取JFreeChart对象，调用父类的方法setContentPane()把JFreeChart对象保存在窗体面板里，代码如下：

```
  public void createPlot() {
    JFreeChart chart = getJFreeChart();
    //把JFreeChart面板保存在窗体里
    setContentPane(new ChartPanel(chart));
  }
```

（5）在main()方法中调用createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
  public static void main(String[] args) {
```

```
    BarDemo1 barDemo = new BarDemo1("柱形图实例");
    barDemo.createPlot();
    barDemo.pack();
    //把窗体显示到显示器中央
    RefineryUtilities.centerFrameOnScreen(barDemo);
    //设置可以显示
    barDemo.setVisible(true);
}
```

举一反三

根据本实例，读者可以开发以下程序。

绘制所有商品销售数量柱形图。

为柱形图添加图例说明。

实例378 绘制自定义颜色的柱形图

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\11\Ex11_378

实例说明

JFreeChart 柱形图中柱形的颜色是可以设置的。本实例演示如何设置柱形图的柱形颜色，运行效果如图11.2所示。

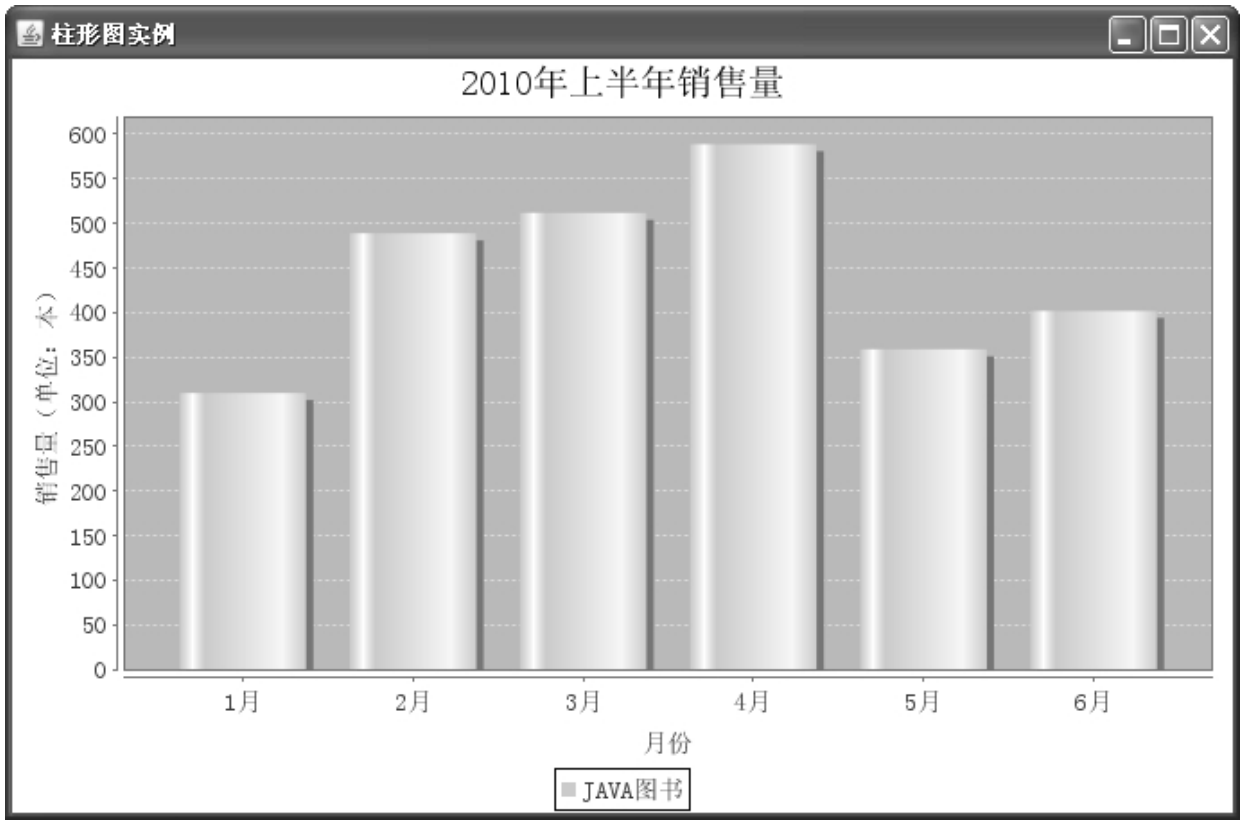


图11.2 设置柱形图的柱形颜色

技术要点

使用AbstractRenderer的setSeriesPaint()方法可以设置柱形的颜色，语法如下：

```
public void setSeriesPaint(int series, Paint paint)
```

参数说明

- series：表示要设置的柱形颜色的系列。
- paint：表示要设置的柱形的颜色。

实现过程

（1）新建一个JAVA文件，同时继承JFreeChart的ApplicationFrame类。

（2）创建getCategoryDataset()方法，在方法内部创建一个适合普通柱形图的数据集合。

（3）创建 getJFreeChart()方法，在方法里获取数据集，通过数据集创建一个柱形图的JFreeChart对象。

（4）创建updateFont()方法，在方法里修改标题、x轴、y轴、x轴标签、y轴标签的字体为“宋体”。

（5）创建updatePlot()方法，在方法中获取CategoryPlot的对象，通过CategoryPlot的对象获取BarRenderer对象，然后为柱形设置颜色，代码如下：

```
private void updatePlot(JFreeChart chart) {
  //图表
  CategoryPlot categoryPlot = chart.getCategoryPlot();
  BarRenderer renderer = (BarRenderer)
categoryPlot.getRenderer();
  //柱图颜色
  renderer.setSeriesPaint(0, Color.orange);
}
```

（6）创建createPlot()方法，在方法里获取JFreeChart对象并且调用updateFont()方法修改字体，再调用updatePlot()方法修改图表设置，然后调用父类的方法setContentPane()把JFreeChart对象保存在窗体面板里。

（7）在main()方法中调用createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
  BarDemo44 barDemo = new BarDemo44("柱形图实例");
  barDemo.createPlot();
  barDemo.pack();
  //把窗体显示到显示器中央
  RefineryUtilities.centerFrameOnScreen(barDemo);
```

```
    //设置可以显示
    barDemo.setVisible(true);
}
```

举一反三

根据本实例，读者可以实现以下功能。

绘制柱形图颜色。

绘制图例，根据颜色区分商品。

实例379 绘制多系列3D柱形图

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\11\Ex11_379

实例说明

多系列的柱形图也可以绘制为3D效果。本实例演示如何绘制一个多系列的3D柱形图，运行效果如图11.3所示。

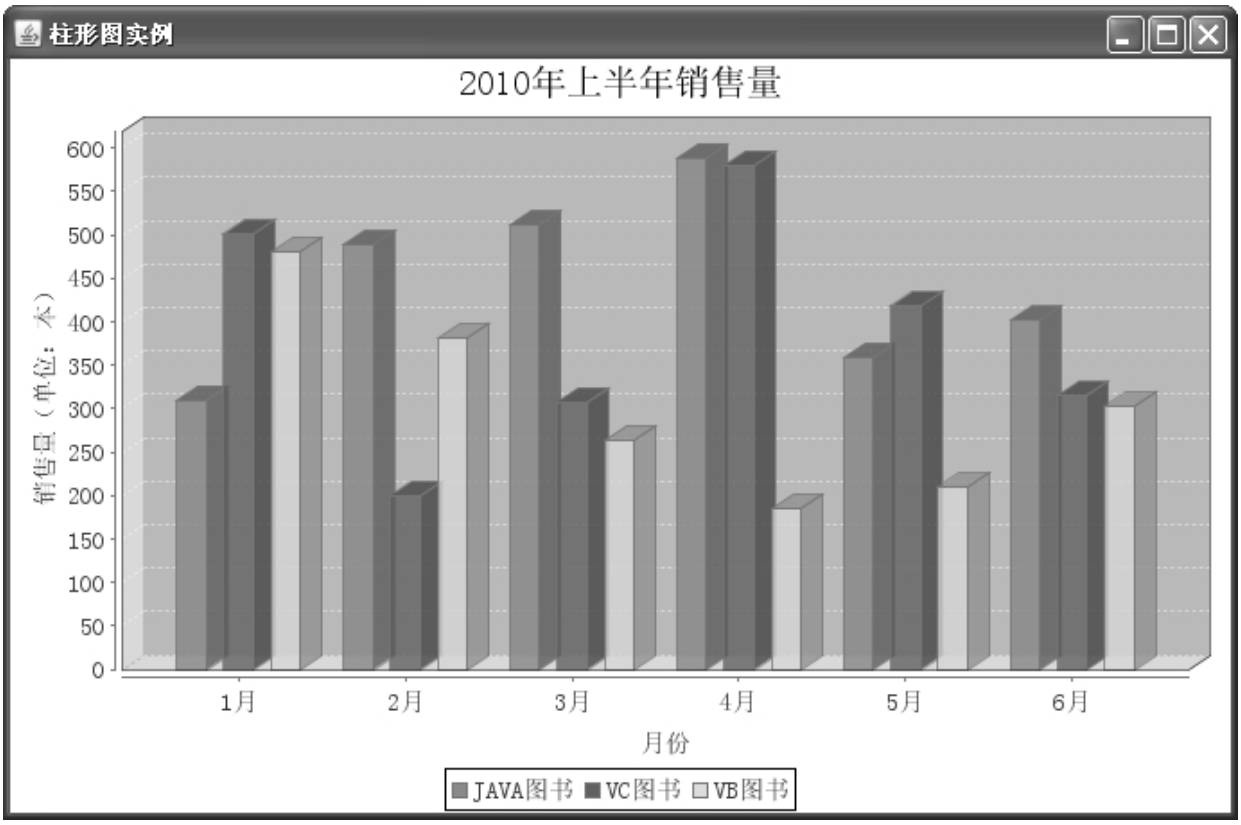


图11.3 多系列3D 柱形图

技术要点

在绘制多系列的3D柱形图时，还可以设置渲染效果。使用CategoryPlot类的getRenderer()方法可以获取多系列3D的渲染效果实例，语法如下：

public CategoryItemRenderer getRenderer()

实现过程

（1）新建一个JAVA文件，同时继承JFreeChart的ApplicationFrame类。

（2）创建getCategoryDataset()方法，在方法内部创建多系列的分类数据集合。

（3）创建getJFreeChart()方法，在方法里获取数据集，通过数据集创建柱形图的JFreeChart对象。

（4）创建updateFont()方法，在方法里重新设置图表标题、标签等字体，代码见实例211。

（5）创建updatePlot()方法，在方法里获取3D的渲染效果实例，设置柱形图显示边线，代码如下：

```
private void updatePlot(JFreeChart chart) {
  //分类图表
  CategoryPlot categoryPlot = chart.getCategoryPlot();
  BarRenderer3D renderer = (BarRenderer3D)
categoryPlot.getRenderer();
  //显示边线
  renderer.setDrawBarOutline(true);
}
```

（6）创建createPlot()方法，在方法里获取JFreeChart对象并且调用updateFont()方法修改字体，调用 updatePlot()方法更新图表的相关设置，然后调用父类的方法 setContentPane()把JFreeChart对象保存在窗体面板里。

（7）在main()方法中调用createPlot()方法，并把窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
  BarDemo48 barDemo = new BarDemo48("柱形图实例");
  barDemo.createPlot();
  barDemo.pack();
  //把窗体显示到显示器中央
  RefineryUtilities.centerFrameOnScreen(barDemo);
  //设置可以显示
  barDemo.setVisible(true);
}
```

举一反三

根据本实例，读者可以实现以下功能。

设置多系列的3D柱形图的边线。

设置柱形图的渲染效果。

11.2 绘制饼图

实例380 绘制椭圆形饼图

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\11\Ex11_380

实例说明

一般的饼图是正圆形的，但有时也会使用椭圆形的饼图。JFreeChart 为我们提供了这样一个功能，可以很方便地绘制一个椭圆形的饼图，运行效果如图11.4所示。

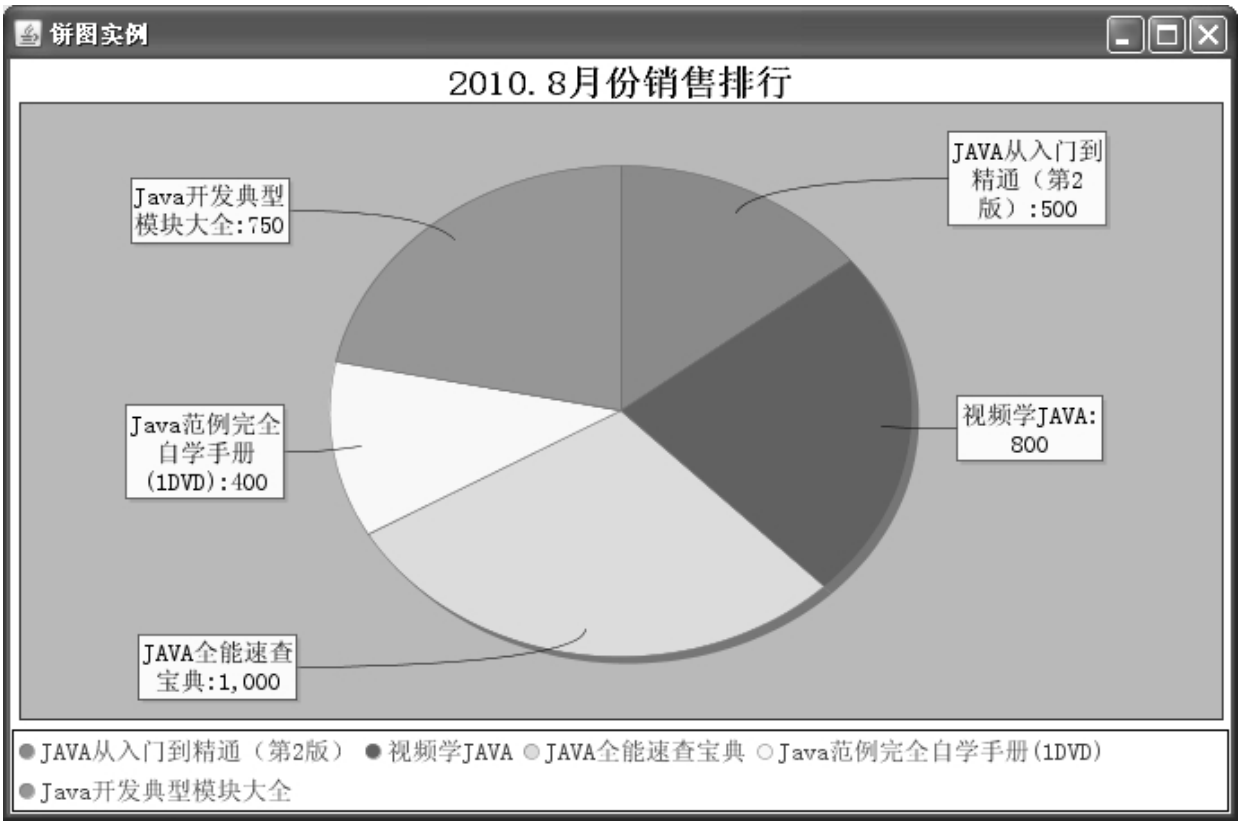


图11.4 椭圆形饼图

技术要点

PiePlot类提供的方法setCircular()可以指定是否绘制圆形，其语法如下：

public void setCircular(boolean flag)

参数说明

flag：表示要绘制的是否为正圆形。当flag为true时，表示要绘制的图表为正圆形；为false时，表示要绘制的图表为椭圆形。

实现过程

（1）新建一个JAVA文件，在创建文件时继承org.jfree.ui.ApplicationFrame类。

（2）获取 DefaultPieDataset 创建的饼图的数据集合，然后使用 ChartFactory 类根据饼图的数据集合创建一个JFreeChart对象，再修改饼图字体。代码如下：

```
private JFreeChart getJFreeChart() {
  //获取数据集
  PieDataset dataset = getPieDataset();
  //生成JFreeChart对象
  JFreeChart chart = ChartFactory.createPieChart("2010.8
月份销售排行", dataset,true, true, false);
  //设置图表字体
  setPiePoltFont(chart);
  return chart;
}
```

（3）创建createPiePlot()方法，在该方法中使用PiePlot的setCircular()方法，设置图表为椭圆形。代码如下：

```
public void createPiePlot() {
  JFreeChart chart = getJFreeChart();
  PiePlot piePlot = (PiePlot) chart.getPlot();
```

```
    //是否椭圆
    piePlot.setCircular(false);
    //把JFreeChart面板保存在窗体里
    setContentPane(new ChartPanel(chart));
  }
```

（4） 在 main()方法中调用 createPiePlot()方法创建图表，然后使用 RefineryUtilities 类的centerFrameOnScreen()方法把图表窗体显示到屏幕中央。代码如下：

```
  public static void main(String[] args) {
    PieDemo2 demo = new PieDemo2("饼图实例");
    demo.createPiePlot();
    demo.pack();
    //把窗体显示到显示器中央
    RefineryUtilities.centerFrameOnScreen(demo);
    demo.setVisible(true);
  }
```

举一反三

根据本实例，读者可以开发以下程序。

用JfreeChart进行图形设置。

创建立体饼图。

实例381 创建3D饼图

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\11\Ex11_381

实例说明

普通的饼图都是平面的，可能会给人一种死板的感觉，JFreeChart 提供了一个很方便地创建3D饼图的功能，让图表更生动。本实例实现的3D饼图运行效果如图11.5所示。

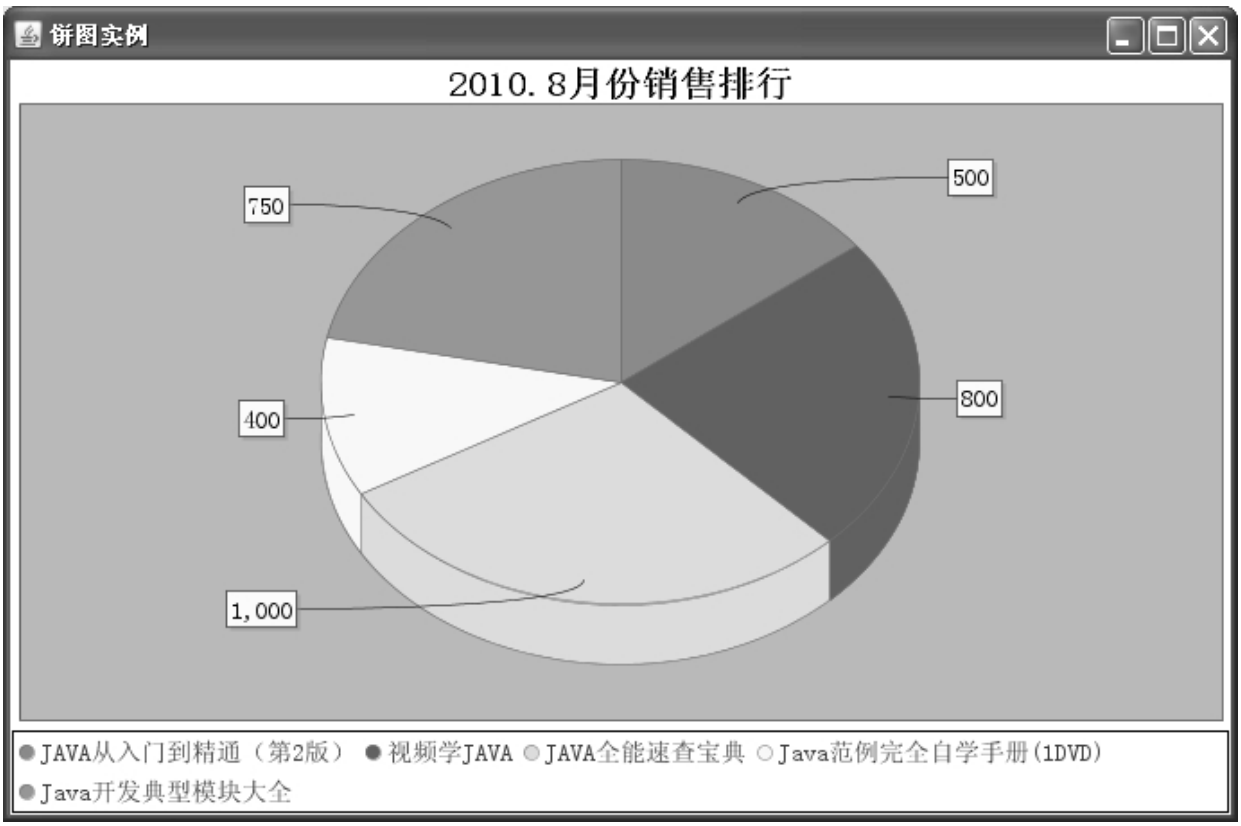


图11.5 3D 饼图

技术要点

使用ChartFactory的createPieChart3D()方法可以创建3D饼图，语法如下：

public static JFreeChart createPieChart3D(String title, PieDataset dataset, boolean legend, boolean tooltips, boolean urls)

参数说明

- title：表示饼图的标题。
- dataset：表示饼图的数据集合。
- legend：表示是否使用图示。

● tooltips：表示是否生成工具栏提示。

● urls：表示是否生成URL 链接。

实现过程

（1）新建一个JAVA文件，在创建文件时继承JFreeChart的ApplicationFrame类。

（2）使用DefaultPieDataset创建一个饼图的数据集合。代码如下：

```
private PieDataset getPieDataset() {
  //创建数据集合实例
  DefaultPieDataset dataset = new DefaultPieDataset();
  //向数据集合添加数据
  dataset.setValue("JAVA从入门到精通（第2版）", 500);
  dataset.setValue("视频学JAVA", 800);
  dataset.setValue("JAVA全能速查宝典", 1,000);
  dataset.setValue("Java范例完全自学手册(1DVD)", 400);
  dataset.setValue("Java开发典型模块大全", 750);
  return dataset;
}
```

（3）使用ChartFactory类根据饼图的数据集合创建有3D效果的JFreeChart对象。代码如下：

```
private JFreeChart getJFreeChart() {
  PieDataset dataset = getPieDataset();
  JFreeChart chart =
ChartFactory.createPieChart3D("2010.8月份销售排行",
dataset,true, true, false);
  //设置饼图使用的字体
  setPiePoltFont(chart);
```

```
    return chart;
  }
```

（4）创建createPiePlot()方法，在方法里把JFreeChart对象保存到面板中。代码如下：

```
  public void createPiePlot() {
    JFreeChart chart = getJFreeChart();
    //把JFreeChart对象保存到面板中
    setContentPane(new ChartPanel(chart));
  }
```

（5）在main()方法中使用RefineryUtilities类的centerFrameOnScreen()方法把窗体显示到屏幕中央。

举一反三

根据本实例，读者可以实现以下功能。

绘制分离饼图。

旋转饼图初始角度。

实例382 绘制3D多饼图

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\11\Ex11_382

实例说明

多饼图同普通饼图一样，也可以绘制成3D的效果。本实例绘制一个具有3D效果的多饼图，运行效果如图11.6所示。

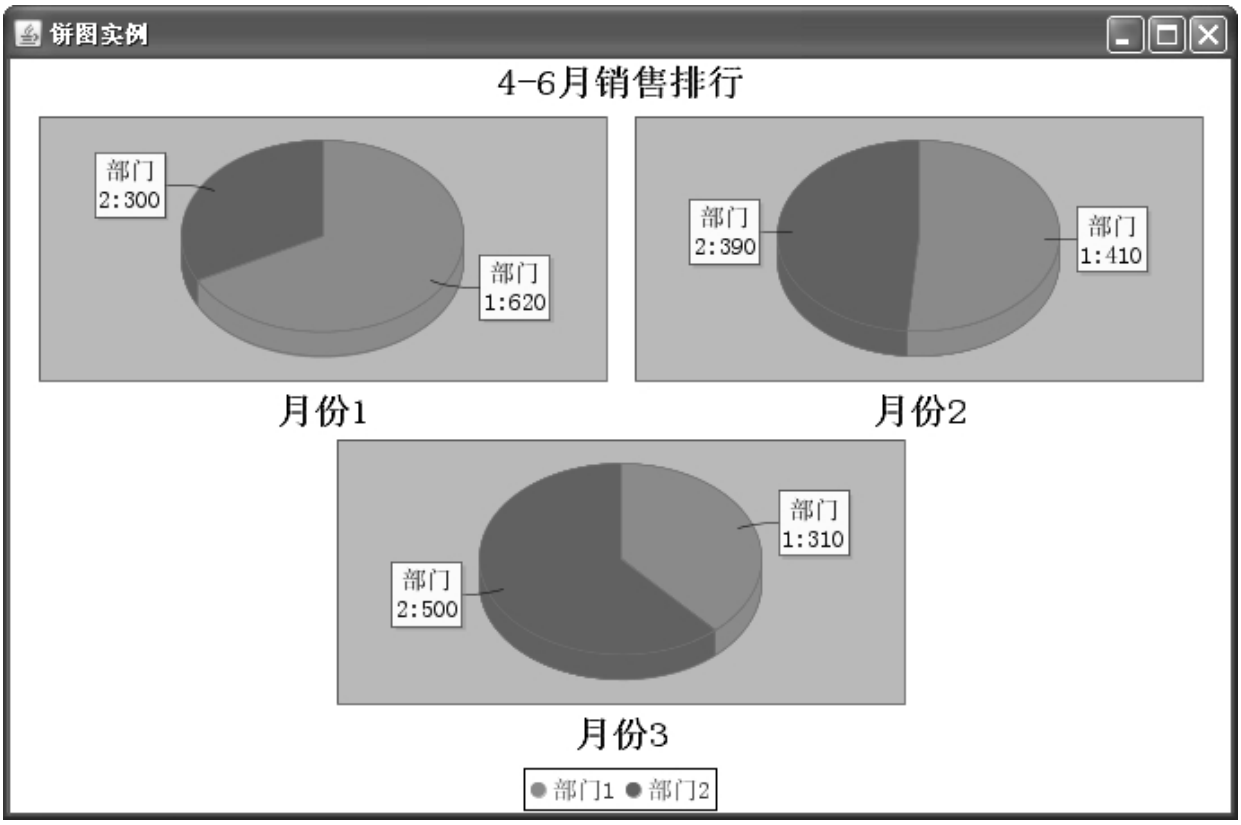


图11.6 3D 多饼图

技术要点

JFreeChart已经在ChartFactory类中创建了可以绘制3D多饼图的方法，该方法的语法如下：

public static JFreeChart createMultiplePieChart3D(String title,CategoryDataset dataset, TableOrder order, boolean legend, boolean tooltips, boolean urls)

参数说明

- title：3D 多饼图的窗体的标题。
- dataset：3D 多饼图的数据集合。
- order：设置3D 多饼图的显示方式。
- legend：表示是否使用图示。
- tooltips：表示是否生成工具栏提示。
- urls：表示是否生成URL 链接。

实现过程

（1）新建一个JAVA文件，然后继承JFreeChart的ApplicationFrame类。

（2）创建 createDataset()方法，在方法里设置图表数据，创建一个 CategoryDataset对象。

（3）创建getJFreeChart()方法，在方法里根据数据集生成多饼图的JFreeChart对象，并且指定多饼图排序方法为列排列。

（4）创建createPiePlot()方法，在方法中获取JFreeChart的实例，使用JFreeChart的实例分别获取MultiplePiePlot、JFreeChart和PiePlot实例。通过修改相应的字体来解决乱码问题。

（5）创建main()方法，使用RefineryUtilities类的centerFrameOnScreen()方法把窗体显示到屏幕中央。

举一反三

根据本实例，读者可以实现以下功能。

设置3D多饼图的z轴高度。

设置3D多饼图的显示方式。

11.3 绘制折线图

实例383 绘制基本折线图

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\11\Ex11_383

实例说明

折线图通过数据的线形走势来体现情况的变化趋势，x轴表示分类情况，y轴表示具体数值。本实例演示如何创建一个基本的折线图，运行效果如图11.7所示。

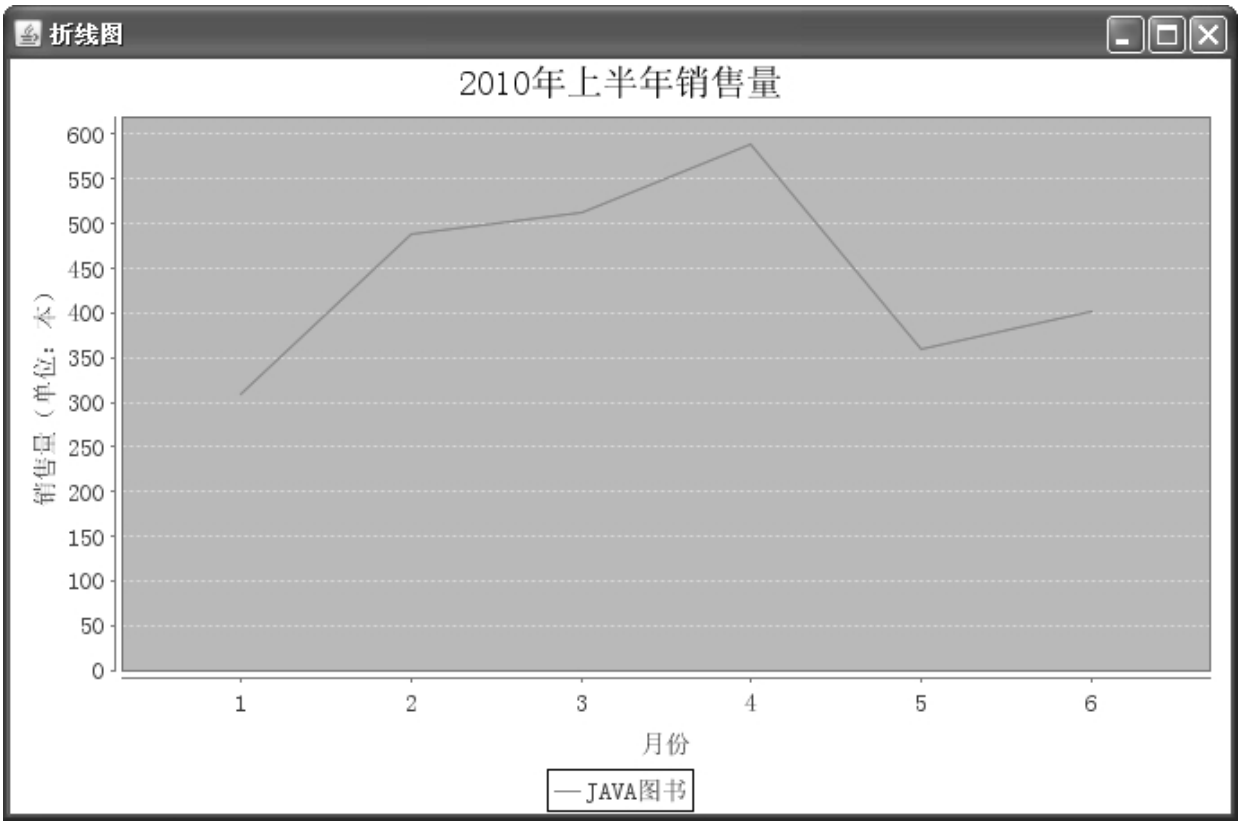


图11.7 基本折线图

技术要点

ChartFactory类的createLineChart()方法提供了创建基本折线图的方法，创建完成以后会返回一个JFreeChart对象，语法如下：

createLineChart (String title, String categoryAxisLabel, String valueAxisLabel, CategoryDataset dataset, PlotOrientation orientation, boolean legend, boolean tooltips, boolean urls)

参数说明

- title：图表的标题。
- categoryAxisLabel：图表类别标签，即x 轴的名称。
- valueAxisLabel：图表数据标签，即y 轴的名称。
- dataset：折线图的数据集合。
- orientation：图表的显示方向。
- legend：表示是否使用图示。
- tooltips：表示是否生成工具栏提示。
- urls：表示是否生成URL 链接。

实现过程

（1）新建一个JAVA文件，在创建文件时继承JFreeChart的ApplicationFrame类。

（2）创建 getCategoryDataset()方法，向 DefaultKeyedValues 类的实例里面添加数据内容，然后通过 DatasetUtilities 类的 createCategoryDataset()方法创建一个数据集，代码如下：

```
private CategoryDataset getCategoryDataset() {
  DefaultKeyedValues keyedValues = new
DefaultKeyedValues();
  keyedValues.addValue("1", 310);
  keyedValues.addValue("2", 489);
  keyedValues.addValue("3", 512);
```

```
        keyedValues.addValue("4", 589);
        keyedValues.addValue("5", 359);
        keyedValues.addValue("6", 402);
        CategoryDataset dataset =
    DatasetUtilities.createCategoryDataset("JAVA图书",
    keyedValues);
        return dataset;
    }
```

（3）创建 getJFreeChart()方法，在方法中获取数据集合，再使用 ChartFactory 类的createLineChart()方法根据数据集合生成一个JFreeChart对象，代码如下：

```
    private JFreeChart getJFreeChart() {
        CategoryDataset dataset = getCategoryDataset();
        JFreeChart chart = ChartFactory.createLineChart("2010年
    上半年销售量",
        //图表标题
        "月份",                                          //
    x轴标签
        "销售量（单位：
    本）",                                    //y轴标签
        dataset,                                          /
    /数据集
        PlotOrientation.VERTICAL,
    //图表方向：垂直
        true,                                             //
    显示图例
```

```
        false,                                                    //
    //不生成工具栏提示
        false                                                     //
    不生成URL链接
        );
        return chart;
    }
```

（4）创建updateFont()方法，在方法中修改标题、x轴、y轴等标签字体，代码如下：

```
    private void updateFont(JFreeChart chart) {
        //标题
        TextTitle textTitle = chart.getTitle();
        textTitle.setFont(new Font("宋体", Font.PLAIN, 20));
        LegendTitle legendTitle = chart.getLegend();
        legendTitle.setItemFont(new Font("宋体", Font.PLAIN,
    14));
        //图表
        CategoryPlot categoryPlot = chart.getCategoryPlot();
        CategoryAxis categoryAxis =
    categoryPlot.getDomainAxis();
        // X轴字体
        categoryAxis.setTickLabelFont(new Font("宋体",
    Font.PLAIN, 14));
        // X轴标签字体
        categoryAxis.setLabelFont(new Font("宋体", Font.PLAIN,
    14));
        ValueAxis valueAxis = categoryPlot.getRangeAxis();
```

```
        // Y轴字体
        valueAxis.setTickLabelFont(new Font("宋体", Font.PLAIN,
    14));
        // Y轴标签字体
        valueAxis.setLabelFont(new Font("宋体", Font.PLAIN,
    14));
      }
```

（5）创建createPlot()方法，在方法中获取一个JFreeChart对象，然后调用updateFont()方法修改字体，最后调用父类的setContentPane()方法把JFreeChart对象保存在窗体面板里，代码如下：

```
    public void createPlot() {
      JFreeChart chart = getJFreeChart();
      //修改字体
      updateFont(chart);
      setContentPane(new ChartPanel(chart));
    }
```

（6）在main()方法中调用createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
    public static void main(String[] args) {
      LineDemo1 demo = new LineDemo1("折线图");
      demo.createPlot();
      demo.pack();
      //把窗体显示到屏幕中央
      RefineryUtilities.centerFrameOnScreen(demo);
      //设置可以显示
      demo.setVisible(true);
```

}

举一反三

根据本实例，读者可以实现以下功能。

在折线图中设置x轴字体。

在折线图中设置y轴字体。

实例384 绘制多条彩色折线图

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\11\Ex11_384

实例说明

折线图的颜色很丰富，用户可以根据实际情况修改折线颜色。本实例演示如何修改折线图的默认颜色，运行效果如图11.8所示。

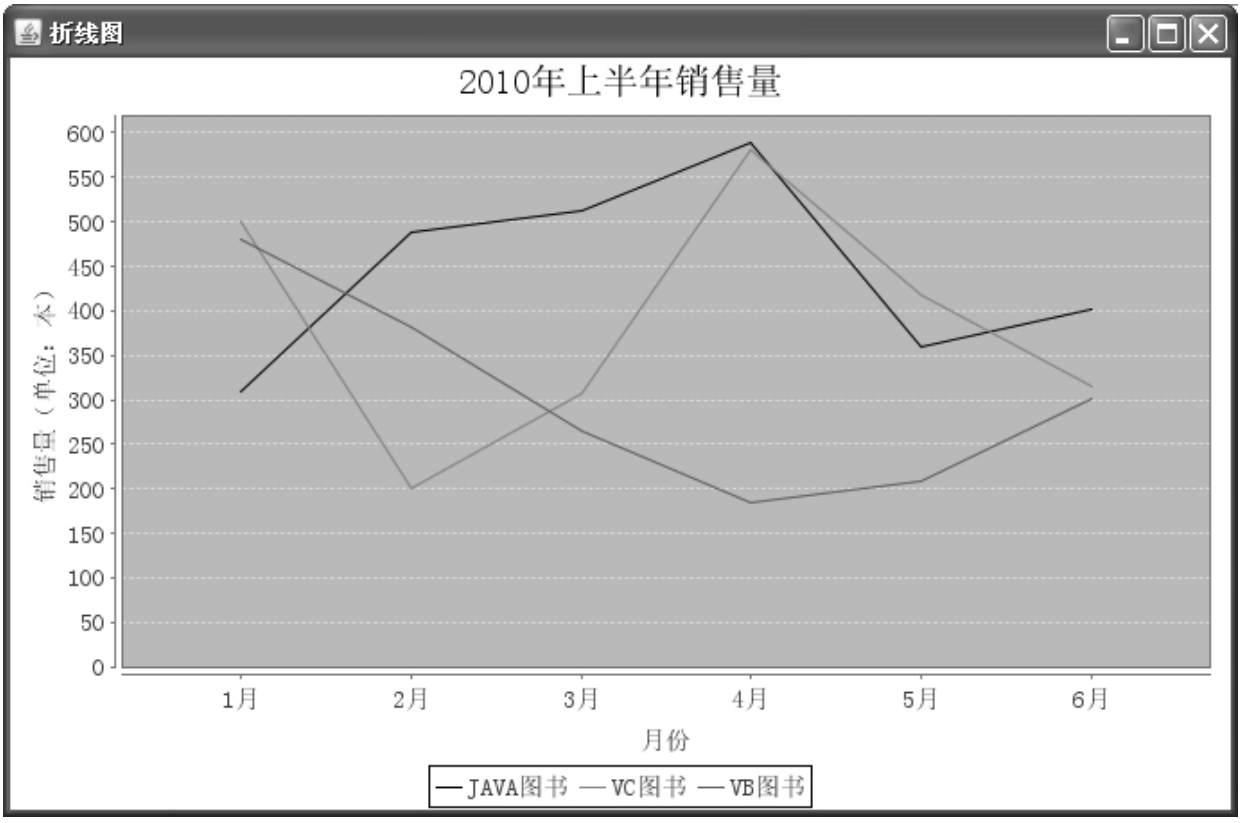


图11.8 设置折线颜色

技术要点

使用LineAndShapeRenderer类的setSeriesPaint()方法可以修改折线图的颜色，语法如下：

setSeriesPaint(int series, Paint paint)

参数说明

- series：折线图当前的系列索引。
- paint：准备设置系列的颜色。

实现过程

（1）新建一个JAVA文件，在创建文件时继承JFreeChart的ApplicationFrame类。

（2）创建getCategoryDataset()方法，向DefaultCategoryDataset类的实例中添加数据内容，代码见实例265。

（3）创建 getJFreeChart()方法，在方法中获取数据集合，再使用 ChartFactory 类的createLineChart()方法根据数据集合生成一个JFreeChart对象，代码见实例264。

（4）创建updateFont()方法，在方法中修改标题、x轴、y轴等标签字体，代码见实例264。

（5）创建updatePlot()方法，在方法中获取LineAndShapeRenderer实例，通过LineAndShapeRenderer实例设置第一个索引的折线为黑色，代码如下：

```
private void updatePlot(JFreeChart chart) {
  //分类图表
  LineAndShapeRenderer renderer = (LineAndShapeRenderer)
chart.getCategoryPlot().getRenderer();
  //设置显示颜色
  renderer.setSeriesPaint(0, Color.black);
}
```

（6）创建createPlot()方法，在方法中获取一个JFreeChart对象，然后修改字体与图表设置，最后调用父类的setContentPane()方法把JFreeChart对象保存在窗体面板里，代码见实例267。

（7）在main()方法中调用createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
  LineDemo9 demo = new LineDemo9("折线图");
  demo.createPlot();
  demo.pack();
  //把窗体显示到屏幕中央
  RefineryUtilities.centerFrameOnScreen(demo);
  //设置可以显示
  demo.setVisible(true);
}
```

举一反三

根据本实例，读者可以实现以下功能。

修改折线图的颜色。

绘制折线的走势。

实例385 绘制排序折线图

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\11\Ex11_385

实例说明

在绘制折线图时，首先要准备数据集填充折线图表，很多时候需要对数据集进行排序显示。本实例演示如何对x轴进行升序排列，运行效果如图11.9所示。

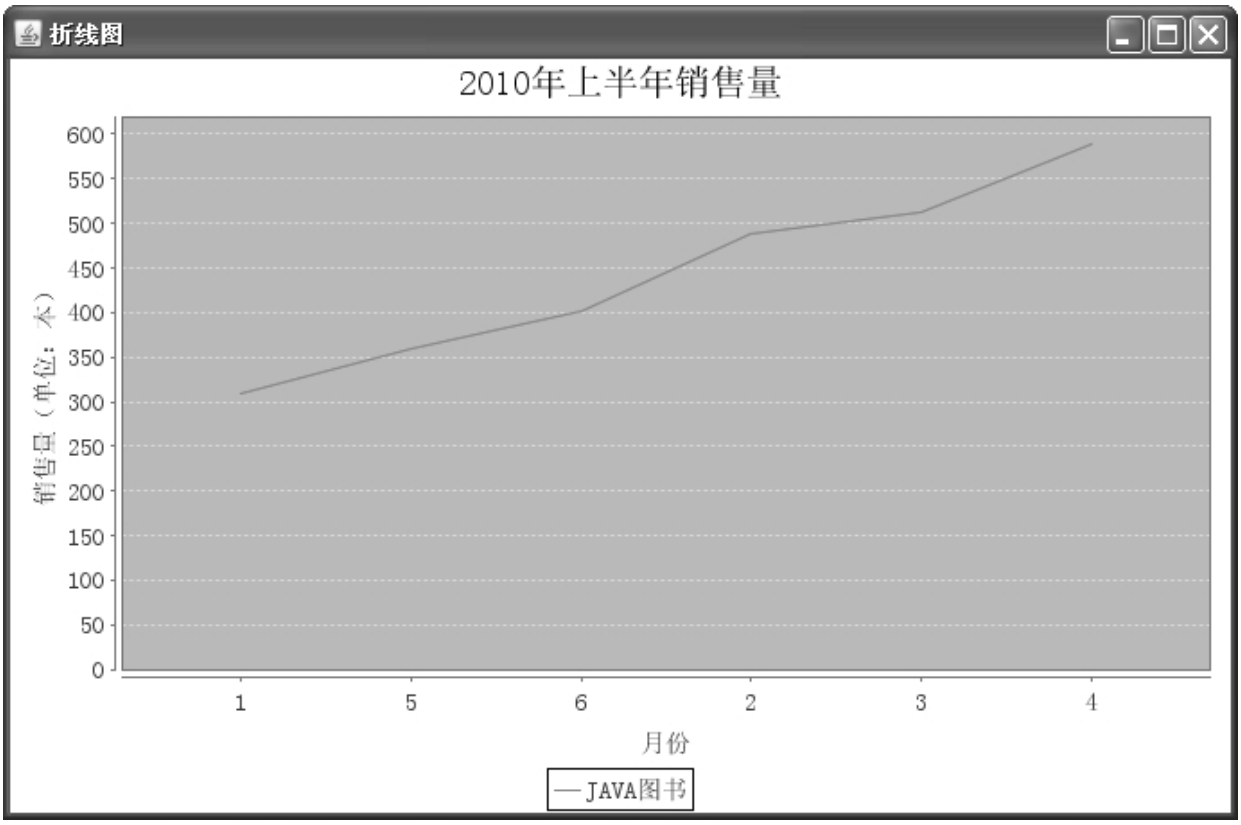


图11.9 排序折线图

技术要点

使用DefaultKeyedValues类的sortByValues()方法可以对数据集进行排序，语法如下：

sortByValues(SortOrder order)

参数说明

order：表示DefaultKeyedValues数据内容的排序方式。

实现过程

（1）新建一个JAVA文件，在创建文件时继承JFreeChart的ApplicationFrame类。

（2）创建getCategoryDataset()方法，使用DefaultKeyedValues类创建分类数据集，代码如下：

```
private CategoryDataset getCategoryDataset() {
```

```
        DefaultKeyedValues keyedValues = new
    DefaultKeyedValues();
        keyedValues.addValue("1", 310);
        keyedValues.addValue("2", 489);
        keyedValues.addValue("3", 512);
        keyedValues.addValue("4", 589);
        keyedValues.addValue("5", 359);
        keyedValues.addValue("6", 402);
        //排序
        keyedValues.sortByValues(SortOrder.ASCENDING);
        CategoryDataset dataset =
    DatasetUtilities.createCategoryDataset("JAVA图书",
    keyedValues);
        return dataset;
      }
```

（3）创建 getJFreeChart()方法，在方法中获取 CategoryDataset 数据集合，再使用 ChartFactory类的createLineChart()方法生成JFreeChart对象。

（4）创建updateFont()方法，在方法中修改标题、x轴、y轴等标签字体。

（5）创建createPlot()方法，在方法中获取一个JFreeChart对象修改字体，然后调用父类的setContentPane()方法把JFreeChart对象保存在窗体面板里。

（6）在main()方法中调用createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
    public static void main(String[] args) {
      LineDemo12 demo = new LineDemo12("折线图");
```

```
    demo.createPlot();
    demo.pack();
    //把窗体显示到屏幕中央
    RefineryUtilities.centerFrameOnScreen(demo);
    //设置可以显示
    demo.setVisible(true);
}
```

举一反三

根据本实例，读者可以实现以下功能。

使用sortByValues方法对数据集进行排序。

绘制排序折线图。

11.4 绘制时序图

实例386 绘制基本时序图

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\11\Ex11_386

实例说明

时序图中时间刻度的样式可以根据需求进行调整。本实例演示如何格式化时序图中的时间样式，运行效果如图11.10所示。

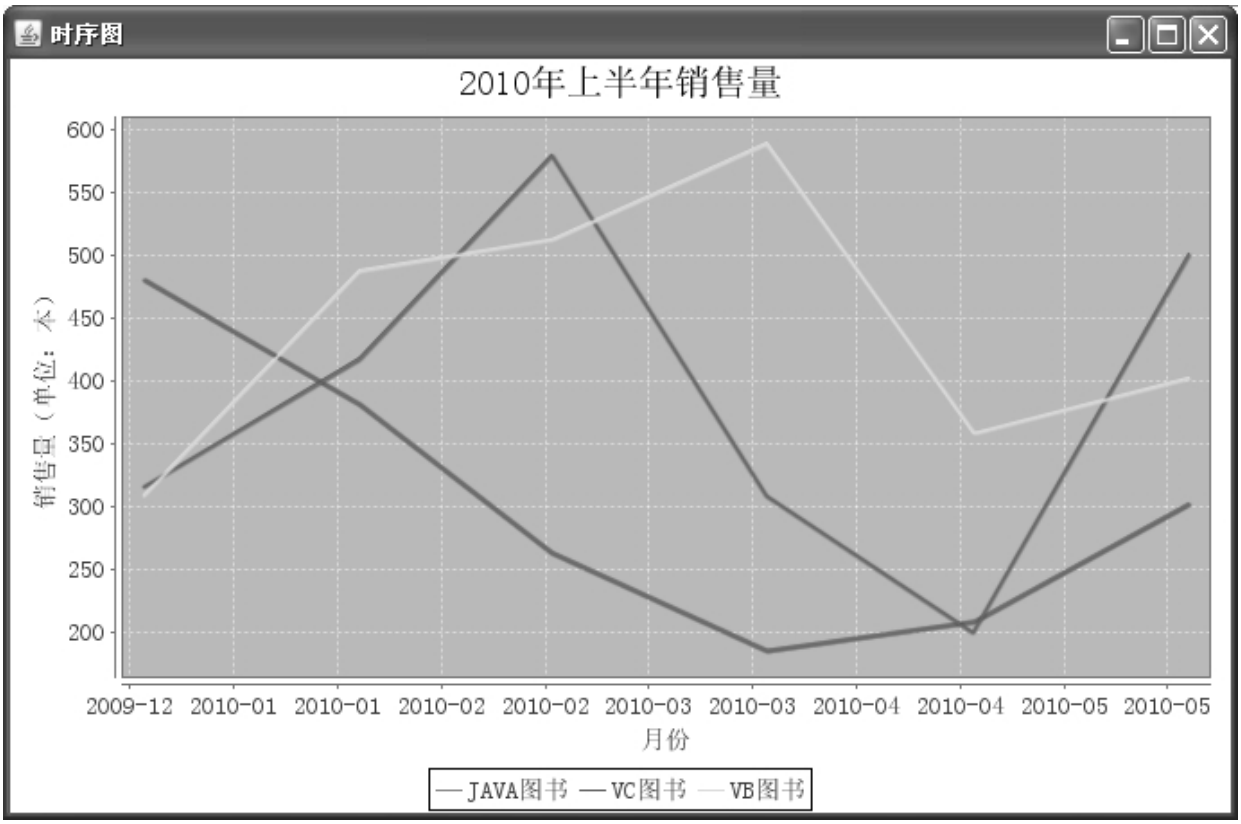


图11.10 设置时间格式

技术要点

使用 DateAxis 类的 setDateFormatOverride()方法可以格式化时间轴显示格式，创建完成以后会返回一个JFreeChart对象，语法如下：

setDateFormatOverride(DateFormat formatter)

参数说明

formatter：表示时序图时间轴的显示格式。

实现过程

（1）新建一个JAVA文件，在创建文件时继承JFreeChart的ApplicationFrame类。

（2）创建getDataset()方法，在方法中使用TimeSeriesCollection类创建一个数据集。代码如下：

```
private XYDataset getDataset() {
  final TimeSeries s1 = new TimeSeries("JAVA图书");
  s1.add(new Month(1, 2010), 480);
  s1.add(new Month(2, 2010), 381);
  s1.add(new Month(3, 2010), 264);
  s1.add(new Month(4, 2010), 185);
  s1.add(new Month(5, 2010), 209);
  s1.add(new Month(6, 2010), 302);
  final TimeSeries s2 = new TimeSeries("VC图书");
  s2.add(new Month(1, 2010), 315);
  s2.add(new Month(2, 2010), 418);
  s2.add(new Month(3, 2010), 580);
  s2.add(new Month(4, 2010), 308);
  s2.add(new Month(5, 2010), 200);
  s2.add(new Month(6, 2010), 501);
  final TimeSeries s3 = new TimeSeries("VB图书");
```

```
        s3.add(new Month(1, 2010), 310);
        s3.add(new Month(2, 2010), 489);
        s3.add(new Month(3, 2010), 512);
        s3.add(new Month(4, 2010), 589);
        s3.add(new Month(5, 2010), 359);
        s3.add(new Month(6, 2010), 402);
        final TimeSeriesCollection dataset = new
    TimeSeriesCollection();
        dataset.addSeries(s1);
        dataset.addSeries(s2);
        dataset.addSeries(s3);
        return dataset;
      }
```

（3）创建 getJFreeChart()方法，在方法中使用 getDataset()方法获取数据集合，再使用ChartFactory类的createTimeSeriesChart()方法生成一个JFreeChart对象，代码如下：

```
      private JFreeChart getJFreeChart() {
        XYDataset dataset = getDataset();
        JFreeChart
    chart=ChartFactory.createTimeSeriesChart("2010年上半年销售
    量",    //图表标题
        "月
    份",
    //x轴标签
        "销售量（单位：
    本）",                                          //y轴标签
```

```
        dataset,
          //数据集
        true,
        //是否显示图例
        false,
          //是否生成工具
        false
        //是否生成URL链接
        );
        return chart;
    }
```

（4）创建updateFont()方法，在方法中修改图表、标题、图示、坐标轴等字体，代码如下：

```
    private void updateFont(JFreeChart chart) {
        //标题
        TextTitle textTitle = chart.getTitle();
        textTitle.setFont(new Font("宋体", Font.PLAIN, 20));
        LegendTitle legendTitle = chart.getLegend();
        legendTitle.setItemFont(new Font("宋体", Font.PLAIN,
  14));
        //图表
        XYPlot xyPlot = chart.getXYPlot();
        ValueAxis domainyAxis = xyPlot.getDomainAxis();
        // x轴字体
        domainyAxis.setTickLabelFont(new Font("宋体",
  Font.PLAIN, 14));
        // x轴标签字体
```

```
    domainyAxis.setLabelFont(new Font("宋体", Font.PLAIN,
14));
    ValueAxis rangeAxis = xyPlot.getRangeAxis();
    // y轴字体
    rangeAxis.setTickLabelFont(new Font("宋体", Font.PLAIN,
14));
    // y轴标签字体
    rangeAxis.setLabelFont(new Font("宋体", Font.PLAIN,
14));
  }
```

（5）创建updatePlot()方法，在方法中获取一个XYPlot实例，然后重新格式化时间格式。代码如下：

```
  private void updatePlot(JFreeChart chart) {
    //分类图表
    XYPlot xyPlot = chart.getXYPlot();
    DateAxis dateAxis = (DateAxis) xyPlot.getDomainAxis();
    //设置显示格式
    dateAxis.setDateFormatOverride(new
SimpleDateFormat("yyyy-MM"));
    //添加时间轴
    DateAxis dateAxis1 = new DateAxis();
    //添加时间范围
    dateAxis1.setRange(new Month(1, 2010).getStart(), new
Month(7, 2010).getEnd());
    //设置时间表格
    dateAxis1.setDateFormatOverride(new
SimpleDateFormat("yyyy-MMM"));
```

```
    xyPlot.setDomainAxis(1,dateAxis1);
  }
```

（6）创建createPlot()方法，在方法中获取一个JFreeChart对象，然后修改图表字体与图表设置，最后调用父类的setContentPane()方法把JFreeChart对象保存在窗体面板里，代码如下：

```
  public void createPlot() {
    JFreeChart chart = getJFreeChart();
    //修改字体
    updateFont(chart);
    //修改图表
    updatePlot(chart);
    setContentPane(new ChartPanel(chart));
  }
```

（7）在main()方法中调用createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
  public static void main(String[] args) {
    TimeSeriesDemo2 demo = new TimeSeriesDemo2("时序图");
    demo.createPlot();
    demo.pack();
    //把窗体显示到屏幕中央
    RefineryUtilities.centerFrameOnScreen(demo);
    //设置可以显示
    demo.setVisible(true);
  }
```

举一反三

根据本实例，读者可以实现以下功能。

绘制时序图。

为时序图添加图例。

实例387 绘制双时间轴的时序图

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\11\Ex11_387

实例说明

JFreeChart 可以新增时间轴，新增的时间轴位置可以根据需要进行设置。本实例演示如何将两个时间轴都显示在图表的底部，运行效果如图11.11所示。

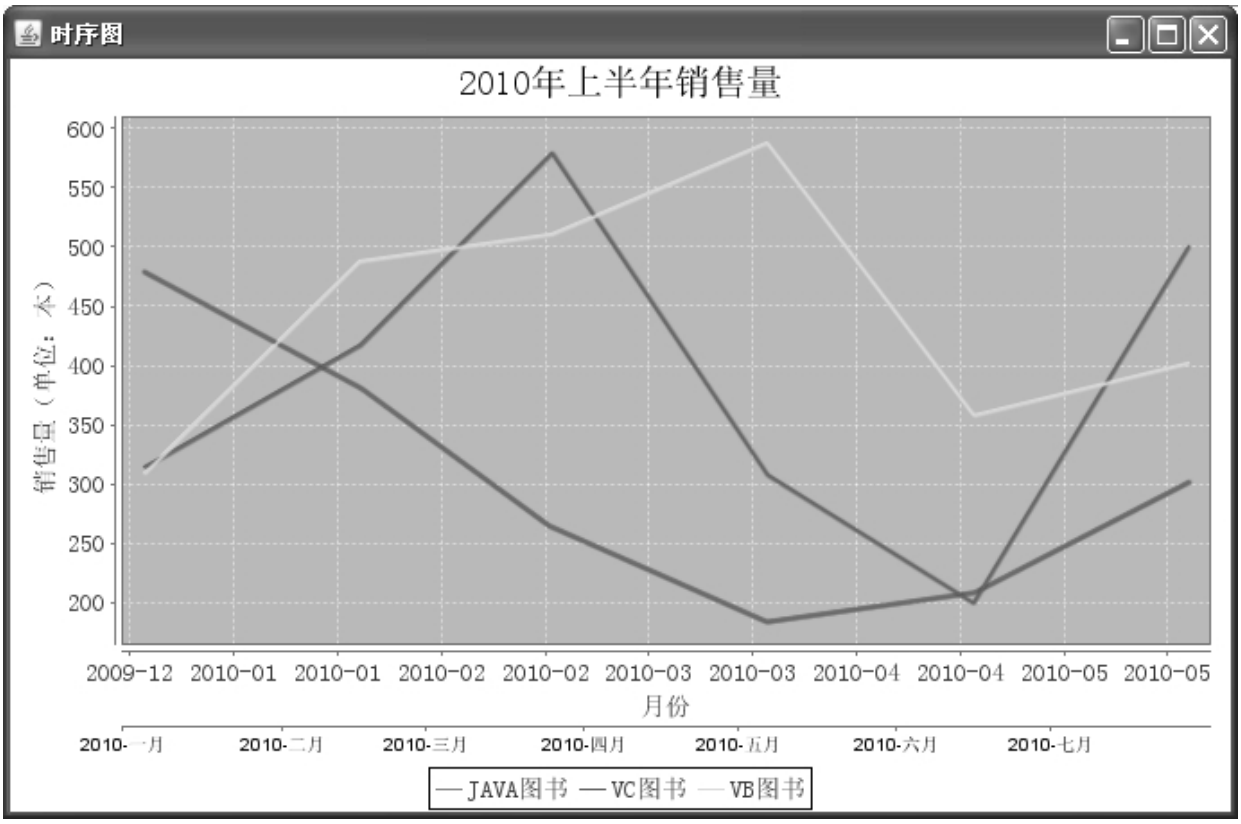


图11.11 设置时间轴位置

技术要点

使用XYPlot类的setDomainAxisLocation()方法可以为新增的时间轴设置显示位置，语法如下：

setDomainAxisLocation(int index, AxisLocation location)

参数说明

- index：时间轴的索引。
- location：要设置时间轴的显示位置。

实现过程

（1）新建一个JAVA文件，在创建文件时继承JFreeChart的ApplicationFrame类。

（2）创建getDataset()方法，在方法中使用TimeSeriesCollection类创建一个数据集。

（3）创建 getJFreeChart()方法，在方法中使用 getDataset()方法获取数据集合，再使用ChartFactory类的createTimeSeriesChart()方法生成一个JFreeChart对象。

（4）创建updateFont()方法，在方法中修改图表、标题、图示、坐标轴等字体。

（5）创建updatePlot()方法，在方法中获取一个XYPlot实例，为图表新增一个时间轴，并且设置时间轴在图表底部或者左侧，代码如下：

```
private void updatePlot(JFreeChart chart) {
  //分类图表
  XYPlot xyPlot = chart.getXYPlot();
  DateAxis dateAxis = (DateAxis) xyPlot.getDomainAxis();
  //设置显示格式
  dateAxis.setDateFormatOverride(new
SimpleDateFormat("yyyy-MM"));
  //添加时间轴
```

```
    DateAxis dateAxis1 = new DateAxis();
    dateAxis1.setRange(new Month(1, 2010).getStart(), new
  Month(7, 2010).getEnd());
    dateAxis1.setDateFormatOverride(new
  SimpleDateFormat("yyyy-MMM"));
    xyPlot.setDomainAxis(1,dateAxis1);
    //设置时间轴位置
    xyPlot.setDomainAxisLocation(1,
  AxisLocation.BOTTOM_OR_LEFT);
  }
```

（6）创建createPlot()方法，在方法中获取一个JFreeChart对象，然后修改图表字体与图表设置，最后调用父类的setContentPane()方法把JFreeChart对象保存在窗体面板里。

（7）在main()方法中调用createPlot()方法创建图表，并把图表的窗体显示在屏幕中央，代码如下：

```
public static void main(String[] args) {
  TimeSeriesDemo4 demo = new TimeSeriesDemo4("时序图");
  demo.createPlot();
  demo.pack();
  //把窗体显示到屏幕中央
  RefineryUtilities.centerFrameOnScreen(demo);
  //设置可以显示
  demo.setVisible(true);
}
```

举一反三

根据本实例，读者可以实现以下功能。

绘制双时间轴。

设置时间轴的位置。

第12章 报表打印

打印的控制

打印的应用

12.1 打印的控制

实例388 打印对话框

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\12\Ex12_388

实例说明

在打印时需要使用“打印”对话框对打印页进行设置，如纸张大小、横向、纵向、打印范围和打印份数等，本实例演示了如何在Java应用程序中打开“打印”对话框。运行程序，将打开“打印”对话框，效果如图12.1所示，单击“属性”按钮，将打开打印属性对话框，可以对纸张大小、横向、纵向、纸张来源和纸张类型等属性进行设置，效果如图12.2所示。

图12.1 “打印”对话框

图12.2 打印属性对话框

技术要点

本实例主要是通过PrinterJob类的静态方法getPrinterJob()，获得PrinterJob对象，然后使用PrinterJob对象调用printDialog()方法打开“打印”对话框。

（1）使用PrinterJob类的静态方法getPrinterJob()，可以获得PrinterJob对象，该方法的定义如下：

public static PrinterJob getPrinterJob()

参数说明

返回值：获得一个新的PrinterJob对象。

（2）使用PrinterJob对象调用printDialog()方法，可以打开“打印”对话框，该方法的定义如下：

public abstract boolean printDialog()

参数说明

返回值：如果用户不取消该对话框，则返回true；否则返回false。

实现过程

（1）新建一个项目。

（2）在项目中创建一个PrintDialogDemo类，在该类的主方法中实现打开“打印”对话框、设置打印任务的名称等操作，该类的代码如下：

```
public class PrintDialogDemo {
  public static void main(String[] args) {
    PrinterJob job = PrinterJob.getPrinterJob(); //获得打印对象
    if (!job.printDialog()) {                    //打开“打印”对话框
      return;                  //单击“打印”对话框的“取消”按钮或关闭“打印”对话框，结束程序的执行
    }
    job.setJobName("测试打印对话框");        //设置打印任务的名称
    String jobName= job.getJobName();        //获得打印任务的名称
    System.out.println("打印任务的名称是： "+ jobName);
  }
}
```

举一反三

根据本实例，读者可以开发以下程序。

设置打印的纸张大小。

设置打印范围。

实例389 实现打印

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\12\Ex12_389

实例说明

本实例演示了如何在Java中实现打印。运行程序，效果如图12.3所示，单击窗体上的“打印”按钮，然后单击“打印”对话框中的“确定”按钮，将开始打印唐诗，效果如图12.4所示。

图12.3 本实例的运行效果

图12.4 正在执行打印任务

技术要点

本实例主要是使用PrinterJob类的setPrintable()方法，并为其传递一个实现了Printable接口的实例，该实例实现了接口中的print()方法，该方法用于绘制打印内容，从而实现打印的功能，该方法的定义如下：

int print(Graphics graphics, PageFormat pageFormat, int pageIndex) throws PrinterException

参数说明

- graphics：用来绘制页面的上下文。
- pageFormat：要绘制的页面的大小和方向。
- pageIndex：要绘制的页面从0 开始的索引。
- 返回值：如果成功呈现该页面，则返回PAGE_EXISTS；如果pageIndex 指定不存在的页面，则返回NO_SUCH_PAGE。
- 异常处理：打印作业被终止时将抛出PrinterException 异常。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的PrintControlFrame窗体类，该类中的“打印”按钮用于实现打印功能，其事件代码如下：

```
button.addActionListener(new ActionListener() {
  public void actionPerformed(final ActionEvent e) {
    try {
      if (!job.printDialog())                    //打开“打印”对话框
        return;                                  //单击“打印”对话框中的“取消”按钮或关闭“打印”对话框结束程序的执行
      job.setPrintable(new Printable() {
        //实现print()方法，绘制打印内容
        public int print(Graphics graphics,PageFormat pageFormat, int pageIndex)throws PrinterException {
          if (pageIndex > 0)
```

```
        return Printable.NO_SUCH_PAGE;          //返回
    该值表示没有打印页
        Graphics2D g2= (Graphics2D) graphics;          //
获得图形上下文对象
        g2.setColor(Color.BLUE);                    //设
置当前绘图颜色
        Font font = new Font("宋体", Font.BOLD, 24);
        g2.setFont(font);                           //设
置字体
        g2.drawString("静夜思",80,40);               //
绘制文本
        font = new Font("宋体", Font.BOLD |
Font.ITALIC, 18);
        g2.setFont(font);                           //设
置字体
        g2.drawString("李白",130,70);                //绘
制文本
        font = new Font("宋体", Font.PLAIN, 14);
        g2.setFont(font);                           //设
置字体
        g2.drawString("床前明月光，",40,100);          //
绘制文本
        g2.drawString("疑是地上霜。",120,100);        //
绘制文本
        g2.drawString("举头望明月，",40,120);          //
绘制文本
```

```
                g2.drawString("低头思故乡。",120,120);        //
            绘制文本
                return Printable.PAGE_EXISTS;                 //返
            回该值表示存在打印页
              }
            });
            job.setJobName("打印唐诗《静夜
        思》");                  //设置打印任务的名称
            job.print();                                          //
        调用print()方法开始打印
          } catch (PrinterException ee) {
            ee.printStackTrace();
          }
        }
    });
```

举一反三

根据本实例，读者可以实现以下功能。

打印文本。

打印图片。

实例390 打印图形

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\12\Ex12_390

实例说明

本实例演示了如何在 Java 应用程序中实现图形的打印。运行程序，效果如图 12.5 所示，单击窗体上的“打印图形”按钮，将通过

打印机打印出五环图形。

图12.5 打印图形

技术要点

本实例主要是通过 Graphics类的 drawOval()方法绘制椭圆，使用 Graphics类的 setColor()方法指定颜色，以及使用Graphics2D类的setStroke()方法设置笔画对象实现的。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的PrintShapeFrame窗体类，该类中的“打印图形”按钮用于实现图形的打印，其事件代码如下：

```
button.addActionListener(new ActionListener() {
  public void actionPerformed(final ActionEvent e) {
    try {
      if (!job.printDialog())     //打开“打印”对话框
        return;              //单击“打印”对话框的“取消”按钮或关闭“打印”对话框结束程序的执行
      job.setPrintable(new Printable() {
        //实现print()方法，绘制打印内容
        public int print(Graphics graphics,PageFormat pageFormat, int pageIndex)throws PrinterException {
```

```
    if (pageIndex > 0)
      return Printable.NO_SUCH_PAGE;
    Graphics2D g2= (Graphics2D)graphics;    //获得Graphics2D对象
    BasicStroke stroke=new BasicStroke(3);  //创建宽度是3的笔画对象
    g2.setStroke(stroke);                   //设置笔画对象
    Color color=new Color(0,162,232);       //创建颜色对象
    g2.setColor(color);                     //设置颜色
    g2.drawOval(30,40,60,60);               //绘制第1个圆
    color=new Color(233,123,16);            //创建新的颜色对象
    g2.setColor(color);                     //设置颜色
    g2.drawOval(60,70,60,60);               //绘制第2个圆
    color=new Color(28,20,100);             //创建新的颜色对象
    g2.setColor(color);                     //设置颜色
    g2.drawOval(92,40,60,60);               //绘制第3个圆
    color=new Color(0,255,0);               //创建新的颜色对象
    g2.setColor(color);                     //设置颜色
```

```
            g2.drawOval(122,70,60,60);              //绘制第4个
        圆
            color=new Color(255,0,0);               //创建新的
        颜色对象
            g2.setColor(color);                     //设置颜色
            g2.drawOval(154,40,60,60);              //绘制第5个
        圆
            return Printable.PAGE_EXISTS;
          }
        });
        job.setJobName("打印五环图形");                  //设置
    打印任务的名称
        job.print();                                //调用
    print()方法开始打印
      } catch (PrinterException ee) {
        ee.printStackTrace();
      }
    }
});
```

举一反三

根据本实例，读者可以实现以下功能。

打印未填充的扇形图。

打印填充的扇形图。

实例391 打印图片

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\12\Ex12_391

实例说明

本实例利用 Java 的打印技术，实现了图片的打印功能。运行程序，效果如图 12.6 所示，单击“选择图片”按钮，选择需要打印的图片（选择图片后的效果如图 12.7 所示），单击“打印”按钮即可打印该图片。

图12.6 选择图片之前的效果

图12.7 选择图片之后的效果

技术要点

本实例通过文件选择器选择需要打印的图片，然后通过从Graphics 类继承的 drawImage()方法绘制需要打印的图片，最终完成图片的打印功能。

（1）通过文件选择器JFileChooser类的showOpenDialog()方法，可以打开文件选择对话框，该方法的定义如下：

public int showOpenDialog(Component parent)

参数说明

- parent：该对话框的父组件，可以为null。
- 返回值：单击对话框的“打开”按钮，返回JFileChooser.APPROVE_OPTION，否则返回JfileChooser.CANCEL_OPTION。

（2）打开文件选择对话框后，可以通过 getSelectedFile()方法获得所选文件的 File 对象， getSelectedFile()方法的定义如下：

public File getSelectedFile()

参数说明

返回值：返回所选择文件的File对象。

（3）使用Graphics类的drawImage()方法绘制图像。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JPanel类的PrintPicturePanel面板类，实现Printable接口，并在实现的print()方法中绘制打印的内容，print()方法的代码如下：

```
public int print(Graphics graphics, PageFormat
pageFormat, int pageIndex)
    throws PrinterException {
    if (pageIndex > 0)
      returnPrintable.NO_SUCH_PAGE;            //没有打印页
```

```
    int x= (int) pageFormat.getImageableX();              //
获得可打印区域的x坐标
    int y= (int) pageFormat.getImageableY();              //
获得可打印区域的y坐标
    Graphics2D g2 = (Graphics2D) graphics;
    int ableW= (int) pageFormat.getImageableWidth();    //
获得可打印区域的宽度
    int ableH= (int) pageFormat.getImageableHeight();    //
获得可打印区域的高度
    int imgW=0;                                          //定义打印图
片的宽度
    int imgH=0;                                          //定义打印图
片的高度
    if (src != null) {
      imgW= src.getWidth();                               //获得图片
  的宽度
      imgH= src.getHeight();                               //获得图
  片的高度
      if (imgW> ableW) {                                  //图片宽度
  大于打印区域的宽度
        imgW= ableW;                                     //图片宽度为打
  印区域的宽度
      }
      if (imgH> ableH) {                                  //图片高度
  大于打印区域的高度
        imgH= ableH;                                     //图片高度为打
  印区域的高度
```

```
        }
      }
      g2.drawImage(src,x,y, imgW, imgH, this);          //绘制打印内容
      return Printable.PAGE_EXISTS;                    //返回存在打印内容的信息
    }
```

（3）面板类PrintPicturePanel中的“打印”按钮，用于实现打印所选图片的功能，其关键代码如下：

```
    try{
      job.setPrintable(PrintPicturePanel.this);       //为打印对象指定Printable对象
      job.setJobName("打印图片");                      //设置打印任务的名称
      job.print();                                    //执行打印
    } catch (PrinterException e1) {
      e1.printStackTrace();
    }
```

举一反三

根据本实例，读者可以开发以下程序。

打印指定的图片。

实现打印预览功能。

实例392 打印预览

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\12\Ex12_392

实例说明

为了在打印之前能够清晰地看到打印内容，可以通过打印预览来实现，本实例实现了打印预览功能。运行程序，选择需要打印的图片文件，单击“打印预览”按钮，将显示打印效果，如图12.8所示。

图12.8 打印预览的效果

技术要点

本实例通过Canvas画布对象绘制打印图像的预览界面，并通过PageFormat类的方法获得可打印区域的坐标和大小，从而实现打印预览效果。

（1）Canvas画布用于表示屏幕上一个空白矩形区域，应用程序可以在该区域内绘图或者捕获用户的输入事件，应用程序必须为Canvas类创建子类，以获得有用的功能，并要重写paint()方法，以便在Canvas画布上执行自定义图形，paint()方法的定义如下：

```
public void paint(Graphics g)
```

参数说明

g：用于在画布上绘制图像的绘图上下文对象。

（2）使用PageFormat类的方法，可以获得可打印区域的坐标和大小，这些方法如下：

□ getImageableX()方法

PageFormat类的getImageableX()方法，用于获得可打印区域左上角的x坐标，该方法的定义如下：

public double getImageableX()

参数说明

返回值：可打印区域左上角的x坐标。

□ getImageableY()方法

PageFormat类的getImageableY()方法，用于获得可打印区域左上角的y坐标，该方法的定义如下：

public double getImageableY()

参数说明

返回值：可打印区域左上角的y坐标。

□ getImageableWidth()方法

PageFormat类的getImageableWidth()方法，用于获得可打印区域的宽度，该方法的定义如下：

public double getImageableWidth()

参数说明

返回值：可打印区域的宽度。

□ getImageableHeight()方法

PageFormat类的getImageableHeight()方法，用于获得可打印区域的高度，该方法的定义如下：

public double getImageableHeight()

参数说明

返回值：可打印区域的高度。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的MainFrame窗体类。

（3）在MainFrame窗体类中，创建继承Canvas类的内部画布类PreviewCanvas，用于绘制打印预览的图像，该类的定义如下：

```
class PreviewCanvas extends Canvas {
  public void paint(Graphics g) {
    try {
      super.paint(g);
      Graphics2D g2 = (Graphics2D) g;
      g2.translate(10,10);                                    //平移绘图上下文
      int x= (int) (pf.getImageableX() - 1);                  //获得可打印区域的x坐标偏左，用于绘制虚线
      int y= (int) (pf.getImageableY() - 1);                  //获得可打印区域的y坐标偏上，用于绘制虚线
      int width= (int) (pf.getImageableWidth()+1);            //获得可打印区域的宽度偏右，用于绘制虚线
      int height= (int) (pf.getImageableHeight()+1);          //获得可打印区域的高度偏下，用于绘制虚线
      int mw= (int) pf.getWidth();                            //获得打印页的宽
```

```
度
    int mh= (int)
pf.getHeight();                          //获得打印页的高
度
    g2.drawRect(0,0,mw,mh);                          //绘
制实线外框
    g2.setColor(new
Color(255,253,234));                  //设置前景色
    g2.fillRect(1,1,mw -1,mh
-1);                          //绘制填充区域
    g2.setStroke(new
BasicStroke(1f,BasicStroke.CAP_ROUND,BasicStroke.JOIN_R
OUND,10f,new float[] {5,5 },0f));       //指定虚线模式
    g2.setColor(Color.BLACK);
        //设置前景色
    g2.drawRect(x,y,width,height);
            //绘制虚线内框
    g2.setColor(Color.WHITE);
        //设置前景色
    g2.fillRect(x+1,y+1,width -1,height
-1);                          //绘制填充区域
    MainFrame.this.print(g2,pf,0);
      //调用print()方法
  } catch (PrinterException e) {
    e.printStackTrace();
  }
}
```

}

举一反三

根据本实例，读者可以实现以下功能。

绘制专业的打印预览效果。

绘制不同的矩形（颜色、实线或虚线）、填充矩形。

实例393 倒序打印

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\12\Ex12_393

实例说明

本实例利用Java的绘图技术和打印技术，实现了图片的倒序打印。运行程序，选择图片文件后，单击“打印预览”按钮，进行打印预览，效果如图12.9所示，单击“开始打印”按钮即可实现倒序打印。

图12.9 倒序打印的预览效果

技术要点

本实例通过Graphics类的drawImage()方法实现倒序绘制图像，并通过Canvas画布对象绘制打印图像的预览界面。

（1）Canvas画布用于表示屏幕上一个空白矩形区域，应用程序可以在该区域内绘图或者捕获用户的输入事件，应用程序必须为Canvas类创建子类，以获得有用的功能，并要重写paint()方法，以便在Canvas画布上绘制自定义图形。

（2）使用Graphics类的drawImage()方法缩放图像。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的MainFrame窗体类。

（3）在MainFrame窗体类中创建实现倒序绘制图像的printPicture()方法，该方法的代码如下：

```
    public void printPicture(Graphics graphics, PageFormat
pageFormat,int pageIndex) {
      int x= (int)
  pageFormat.getImageableX();                    //获取可打印区域
  的x坐标
      int y= (int)
  pageFormat.getImageableY();                    //获取可打印区域
  的y坐标
      Graphics2D g2 = (Graphics2D) graphics;
      if (imgFile != null && isPreview) {
        try {
          src= ImageIO.read(imgFile);                    //构造
      BufferedImage对象
        } catch (IOException e) {
          e.printStackTrace();
        }
        double imgWidth= src.getWidth(this);                //
    获得图像的宽度
        double imgHeight=
    src.getHeight(this);                //获得图像的高度
        double imgWidthS= imgWidth;                    //存储
    图像的宽度
```

```
    double imgHeightS= imgHeight;                    //存储
图像的高度
    int mw= (int) pf.getWidth();                      //
获得打印页的宽度
    int mh= (int) pf.getHeight();                     //
获得打印页的高度
    if (imgWidth>mw) {                               //如果
宽大于可打印区域
      imgWidth=mw - x;                              //设置新
  宽度值
    }
    if (imgHeight>mh) {                              //如果
高大于可打印区域
      imgHeight=mh - y;                             //设置新
  的高度值
    }
    g2.drawImage(src,x,y, (int) imgWidth, (int)
imgHeight,x,(int) imgHeightS, (int) imgWidthS, y,
this);    //绘制倒序图像
  }
  isPreview= false;                                  //设
置不可以打印
}
```

举一反三

根据本实例，读者可以实现以下功能。

实现图像的缩放、旋转和倾斜。

实现图像的阴影效应。

实例394 为打印内容添加水印

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\12\Ex12_394

实例说明

本实例利用Java的绘图技术和打印技术，实现了为打印内容添加水印的功能。运行程序，选择图片文件后，单击“打印预览”按钮，在打开的对话框中进行打印设置后，将在打印页添加水印文字“明日科技”，效果如图12.10所示，单击“开始打印”按钮可以打印带有水印的内容。

技术要点

本实例主要是通过 Graphics2D类的 setComposite()方法，为绘图上下文指定表示透明度的AlphaComposite对象，以及使用从Graphics类继承的drawString()方法绘制文本，从而实现了为打印内容添加水印。

图12.10 为打印内容添加水印的效果

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的MainFrame窗体类。

（3）在MainFrame窗体类中创建为打印内容添加水印的printPicture()方法，该方法中实现添加水印文字的代码如下：

```
/************************ 添加水印文字
*****************************/
    Graphics2D g=
src.createGraphics();                              //获取图片绘
```

图上下文

```
    Font font=new Font("黑
体",Font.BOLD,wordSize);                    //创建字体对象
    g.setFont(font);
    //设置绘图字体
    g.setPaint(Color.RED);
    //设置绘图颜色
    Rectangle2D rec=
font.getStringBounds(watermarkWord,g.getFontRenderContext());
                                    //获取文字占用的像素区域
    double pw=
rec.getWidth();                                    //获取水印文
字占用的像素宽度
    double ph=
rec.getHeight();                                    //获取水印
文字占用的像素高度
    g.rotate(Math.toRadians(30),wordSize+pw / 2,wordSize+ph /
2);      //转换角度
    g.setComposite(AlphaComposite.SrcOver.derive(0.4f));
          //设置水印透明合成规则
    g.drawString(watermarkWord,wordSize* 2+ (int) pw /
2,wordSize* 2+ (int) ph /
2);                                    //绘制水印文字
    /**************************************************************
*******/
```

举一反三

根据本实例，读者可以实现以下功能。

在打印页中添加多处水印。

在打印页中局部添加水印。

12.2 打印的应用

实例395 打印快递单

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\12\Ex12_395

实例说明

本实例演示了如何实现快递单的打印。运行程序，在快递单图片上需要添加文字的位置单击鼠标右键，即可添加一个文本框，输入文字后按Enter键确认，可以隐藏文本框的边框，用鼠标单击该文本框，又可以显示其边框，重复上述过程可以实现快递单文字的添加，效果如图12.11所示。

图12.11 打印快递单界面

技术要点

本实例根据快递单图片背景面板的位置和大小，创建Rectangle对象，然后将该对象传递给Robot类的createScreenCapture()方法，捕获窗体上快递单图片的图像缓冲对象，并将其绘制到打印页面上，从而实现了快递单的打印。

（1）获得窗体上快递单图片背景面板的位置和大小，并创建Rectangle对象，代码如下：

```
    int x= (int)
(ExpressPrintFrame.this.getBounds().getX())+8;
        //背景面板在屏幕上的x坐标
    int y= (int)
(ExpressPrintFrame.this.getBounds().getY())+30;
        //背景面板在屏幕上的y坐标
    int w=
(int)backPanel.getBounds().getWidth();
          //背景面板的宽度
    int h=
(int)backPanel.getBounds().getHeight();
          //背景面板的高度
    Rectangle rect=new
Rectangle(x,y,w,h);                                    //创建
Rectangle对象
```

（2）使用Robot类的createScreenCapture()方法，捕获窗体上快递单图片的图像缓冲对象，并将其绘制到打印页，关键代码如下：

```
    buffImage=
robot.createScreenCapture(rect);
```

```
//获得缓冲图像对象
//省略了部分代码
g2.drawImage(buffImage,printX,printY, imgWidth,
imgHeight,ExpressPrintFrame.this);       //将缓冲图像绘制到打印页
```

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承自JTextField类的文本框类TargetTextField，以实现用鼠标改变文本框在父级容器中的位置和大小。

（3）在项目中创建一个继承自JFrame类的窗体类ExpressPrintFrame，以实现快递单的打印功能，其中打印文字可以通过在背景面板上单击鼠标右键添加文本框实现。背景面板的鼠标右键事件代码如下：

```
backPanel.addMouseListener(new MouseAdapter() {
  public void mouseClicked(final MouseEvent e) {
    if (e.getButton()==MouseEvent.BUTTON3)
{                          //单击鼠标右键
      int x=
  e.getX();                                                    //
  获得鼠标位置的x坐标
      int y=
  e.getY();                                                    //
  获得鼠标位置的y坐标
      TargetTextField tf=new
  TargetTextField();                          //创建自定义
  文本框的实例
```

```
            tf.addMouseListener(tf);
                    //添加鼠标监听器
            tf.addMouseMotionListener(tf);
                    //添加鼠标监听器
            tf.addActionListener(tf);
                    //添加动作监听器
            tf.setBounds(x,y,147,22);
                  //指定文本框的位置和大小
            backPanel.add(tf);
                  //添加到背景面板上
            tf.requestFocus();
                  //使文本框获得焦点
        }
    }
});
```

举一反三

根据本实例，读者可以实现以下功能。

实现快递单文字的添加。

打印快递单。

实例396 打印报表

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\12\Ex12_396

实例说明

本实例通过Java的绘图技术和打印技术，实现了打印报表的功能。运行程序，单击窗体上的“预览报表”按钮，在弹出的对话框中

进行打印设置后，将把数据库中的信息显示在预览界面中，效果如图12.12所示，单击“打印报表”按钮，将把这些信息打印出来。

1	张**	男	50	吉林省长春市	13**00	130**1233122
2	赵**	女	26	吉林省吉林市	13**00	130**2222222
3	金**	女	32	吉林省长春市	13**56	135**1255252
4	王**	男	30	吉林省通化市	13**12	133**5445122
5	张**	女	42	吉林省通化市	13**55	139**4555222
6	刘**	男	40	吉林省长春市	13**00	136**1545666
7	李**	男	18	吉林省吉林市	13**15	135**1256885
8	孙**	女	20	吉林省吉林市	13**33	136**5656565
9	安**	女	30	吉林省通化市	13**59	133**4567989

图12.12 打印报表的预览效果

技术要点

本实例使用JDBC技术从数据库表中获得结果集，然后通过Graphics类的drawString()方法绘制文本，使用drawLine()方法绘制报表的表格，从而实现了打印报表的功能。

（1）为了获得数据库表中的数据记录，本实例定义了一个QueryResultSet类，在该类中定义了一个gainRecord()方法，用于获得数据库表的结果集对象，该方法的代码如下：

```
public static ResultSet gainRecord() {
  Connection
conn=null;                                    //声明连接
```

```
    Statement
st=null;                                          //声明
Statement对象
    ResultSet
rs=null;                                          //声明结果集
对象
    try {
      String url=
  "jdbc:jtds:sqlserver://localhost:1433/db_database";  //数
  据库db_database的URL
      String username=
  "sa";                                  //数据库的用户名
      String password=
  "";                                   //数据库密码
      conn=DriverManager.getConnection(url,username,passwor
  d);  //建立连接
      st=
  conn.createStatement();                                 //创建
  Statement对象
      String sql= "select* from
  tb_employee";                  //定义SQL查询语句
      rs=
  st.executeQuery(sql);                                   //执行
  SQL语句获得结果集
      return
  rs;                                                     //返回结果集
    } catch (SQLException e) {
```

```
        JOptionPane.showMessageDialog(null, "数据库异常：\n"+
    e.getMessage());
        return null;
      }
    }
```

（2）使用Graphics类的drawString()方法，绘制结果集中的记录数据，代码如下：

```
    g2.drawString(String.valueOf(rs.getInt(1)), x+30,
y);                    //绘制报表内容
    g2.drawString(rs.getString(2),x+60,y);
        //绘制报表内容
    //省略了部分代码
```

（3）使用Graphics类的drawLine()方法，绘制报表的表格，代码如下：

```
    g2.drawLine(x+20,y1+10,x+20,y+10);                    /
/绘制垂直直线
    g2.drawLine(x+50,y1+10,x+50,y+10);                    /
/绘制垂直直线
    //省略了部分代码
```

实现过程

（1）新建一个项目。

（2）在项目中创建一个 QueryResultSet 类，用于加载数据库驱动、连接数据库以及获得数据库表中的结果集。

（3）在项目中创建一个继承自 JFrame 类的 MainFrame 窗体类，在该类中定义一个printPicture()方法，用于绘制打印的报表内容，该方法的关键代码如下：

```
    try{
```

```
    y=y+80;                                                    //调
整打印位置的y值
    int y1=y;                                                   //
保存调整后打印位置的y值
    while (rs.next()) {
      y=y+30;                                                  //调整
  打印位置的y值
      g2.drawLine(x+20,y - 20,x+w - 20,y -
  20);                  //绘制水平直线
      g2.drawString(String.valueOf(rs.getInt(1)), x+30,
  y);            //绘制报表内容
      g2.drawString(rs.getString(2),x+60,y);
      //绘制报表内容
      g2.drawString(rs.getString(3),x+120,y);
      //绘制报表内容
      g2.drawString(String.valueOf(rs.getInt(4)),x+170,y);
          //绘制报表内容
      g2.drawString(rs.getString(5),x+220,y);
      //绘制报表内容
      g2.drawString(rs.getString(6),x+420,y);
      //绘制报表内容
      g2.drawString(rs.getString(7),x+480,y);
      //绘制报表内容
    }
    g2.drawLine(x+20,y1+10,x+20,y+10);                         //
/绘制垂直直线
```

```
    g2.drawLine(x+50,y1+10,x+50,y+10);                    //绘制垂直直线
    g2.drawLine(x+110,y1+10,x+110,y+10);                  //绘制垂直直线
    g2.drawLine(x+160,y1+10,x+160,y+10);                  //绘制垂直直线
    g2.drawLine(x+210,y1+10,x+210,y+10);                  //绘制垂直直线
    g2.drawLine(x+410,y1+10,x+410,y+10);                  //绘制垂直直线
    g2.drawLine(x+470,y1+10,x+470,y+10);                  //绘制垂直直线
    g2.drawLine(x+w - 20,y1+10,x+w - 20,y+10);            //绘制垂直直线
    g2.drawLine(x+20,y+30 - 20,x+w - 20,y+30 - 20);       //绘制水平直线
    rs.close();                                           //关闭结果集
  } catch (Exception ex) {
    ex.printStackTrace();
  }
```

举一反三

根据本实例，读者可以实现以下功能。

绘制报表表头。

打印报表。

实例397 打印桌面图片

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\12\Ex12_397

实例说明

本实例实现了打印桌面图片的功能。运行程序，单击窗体上的“获取桌面”按钮，将隐藏打印桌面图片窗体，并抓取屏幕图片，效果如图 12.13 所示，用户还可以单击“页面设置”按钮进行页面设置，然后单击“打印”按钮就可以打印桌面图片。

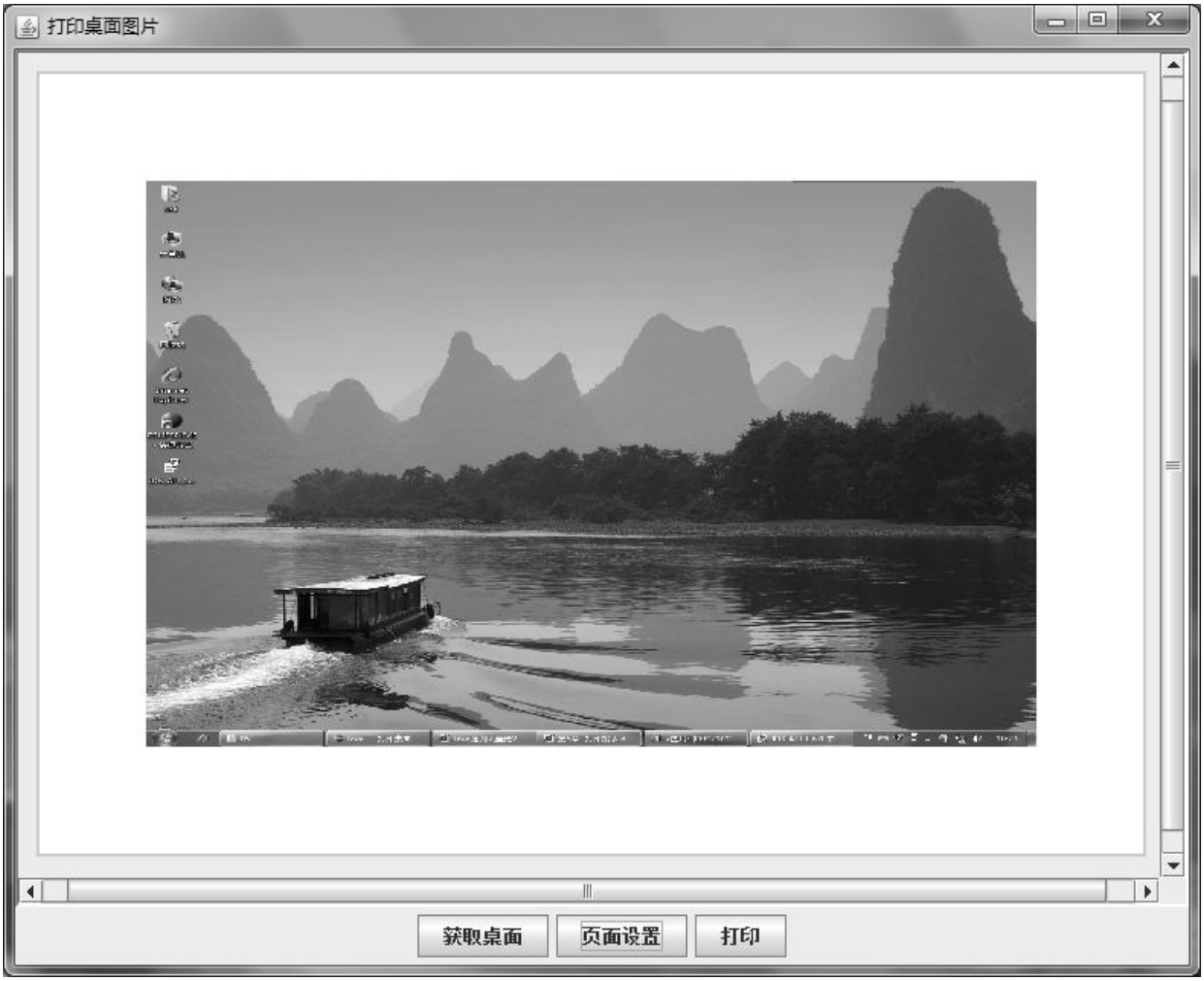


图12.13 获取桌面图片的效果

技术要点

本实例是在单击窗体上的“获取桌面”按钮时，隐藏当前窗体，然后使用 Robot 类的createScreenCapture()方法来捕获桌面图片，

然后再显示当前窗体，从而可以使用Java的打印技术实现打印桌面图片的功能。

（1）使用JFrame类从Window类继承的setVisible()方法可以隐藏和显示窗体，该方法的定义如下：

public void setVisible(boolean b)

参数说明

b：布尔型值，如果为true，则显示窗体；如果为false，则隐藏窗体。

（2）使用Robot类的createScreenCapture()方法，可以捕获桌面图片，该方法的定义如下：

public BufferedImage createScreenCapture(Rectangle screenRect)

参数说明

- screenRect：在屏幕中捕获的矩形区域的位置和大小。
- 返回值：存储桌面图片的缓冲图像对象。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JPanel类并实现Printable接口的自定义面板类PrintPanel，在实现的print()方法中完成打印内容的绘制，该方法的代码如下：

```
public int print(Graphics g1, PageFormat pageFormat, int pageIndex)throws PrinterException {
    Graphics2D g = (Graphics2D) g1;
    px= (int) pageFormat.getImageableX();                    //获取可打印区域的x坐标
```

```
    py= (int)
pageFormat.getImageableY();                    //获取可打
印区域的y坐标
    pwidth= (int)
pageFormat.getImageableWidth();                //获取可打印
的宽度
    pheight= (int)
pageFormat.getImageableHeight();               //获取可打
印的高度
    int pageWidth= (int)
pageFormat.getWidth();                    //获取打印页面宽度
    int pageHeight= (int)
pageFormat.getHeight();                    //获取打印页面高度
    Dimension preferredSize = new Dimension(pageWidth,
pageHeight);
    setPreferredSize(preferredSize);
      //设置面板大小
    getParent().doLayout();
  //重写布局父容器
    g.setColor(Color.WHITE);
  //设置前景色为白色
    g.fill3DRect(0,0,pageWidth,pageHeight,
true);                  //绘制与打印页面相同大小的矩形
    if (pageIndex<1)
{                                          //如果当前打印页数小
于1
```

```
        g.drawImage(image,px,py,pwidth,pheight,
    this);              //绘制打印内容
        return
    Printable.PAGE_EXISTS;                          //返回可打
    印标识
      }
  else                                              //否则
        return
    Printable.NO_SUCH_PAGE;                         //返回不支持
    打印标识
    }
```

（3）在项目中创建一个继承JFrame类的PrintDesktop窗体类，其中的“获取桌面”按钮，用于捕获桌面图片，其事件代码如下：

```
    getDesktopBtn.addActionListener(new ActionListener() {
      public void actionPerformed(final ActionEvent e) {
        try {
          Robot robot=new
      Robot();                                  //创建Robot类的实
      例对象
          Dimension
      size=getToolkit().getScreenSize();                //获
      取屏幕大小
          Rectangle screenRect=new Rectangle(0,0,
      size.width,size.height);
          //创建屏幕大小的矩形对象
          setVisible(false);
            //隐藏窗体
```

```
            Thread.sleep(1000);
            //休眠1秒钟
            BufferedImage image=
        robot.createScreenCapture(screenRect);     //抓取屏幕图
        像
            setVisible(true);
            //显示窗体
            printPanel.setImage(image);
              //为打印面板设置图像
            printPanel.repaint();
              //重新绘制打印面板
          } catch (AWTException e1) {
            e1.printStackTrace();
          } catch (InterruptedException e1) {
            e1.printStackTrace();
          }
        }
});
```

举一反三

根据本实例，读者可以实现以下功能。

截取桌面图片并打印。

实现图片打印预览功能。

实例398 导出报表到Excel表格

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\12\Ex12_398

实例说明

本实例实现了将报表导出到 Excel 表格的功能。运行程序，将在窗体上显示数据库表中的数据信息，效果如图12.14所示，单击窗体上的“导出到Excel”按钮，可以将报表中的数据导出到C盘名为report.xls的Excel文件中。

图12.14 导出报表到Excel 表格

技术要点

本实例通过POI实现了将报表导出到Excel表格的功能，其中，关键技术是工作簿、工作表、工作表中的行和单元格的创建，以及向单元格中添加数据值，使用POI实现这些功能的代码如下：

```
HSSFWorkbook workbook=new HSSFWorkbook();                    //创建工作簿对象
HSSFSheet sheet=workbook.createSheet("ReportToExcel");       //创建名称为ReportToExcel的工作表
HSSFRow row=sheet.createRow(0);                              //在工作表中创建行，其行索引为0
```

```
    HSSFCell cell=null;                                    //声
明单元格
    //导出的报表在Excel中的标题
    for (int i = 0; i < title.length; i++) {
      cell= row.createCell((short)
  i);                          //创建单元格
      cell.setCellType(HSSFCell.CELL_TYPE_STRING);        //
  设置单元格类型
      cell.setCellValue(title[i]);
    //设置单元格的内容
    }
```

实现过程

（1）新建一个项目。

（2）在项目中创建一个QueryResultSet类，用于加载数据库驱动、连接数据库以及获得数据库表中的数据信息。

（3）在项目中创建一个继承自 JFrame 类的 ReportToExcelFrame 窗体类，在该类中创建writeExcel()方法，用于将报表导出到Excel文件中，该方法的代码如下：

```
    @SuppressWarnings("deprecation")
    public void writeExcel(String filename) throws
IOException {
      HSSFWorkbook workbook=new
  HSSFWorkbook();              //创建工作簿对象
      HSSFSheet
  sheet=workbook.createSheet("ReportToExcel");        //创建
  名称为ReportToExcel的工作表
```

```
    HSSFRow row=
sheet.createRow(0);                          //在工作表中创建
行，其行索引为0
    HSSFCell
cell=null;                                   //声明单元格
    //导出的报表在Excel中的标题
    for (int i = 0; i < title.length; i++) {
      cell= row.createCell((short)
  i);                              //创建单元格
      cell.setCellType(HSSFCell.CELL_TYPE_STRING);
    //设置单元格类型
      cell.setCellValue(title[i]);
        //设置单元格的内容
    }
    //导出的报表在Excel中的内容
    ResultSet
rs=QueryResultSet.gainRecord();              //获得数
据库表的结果集对象
    int
rowIndex=1;                                  //单元格
内容的行索引值
    try {
      while (rs.next())
  {                                          //遍历结果集
        row = sheet.createRow(rowIndex);
        for (int i = 0; i < title.length; i++) {
```

```
        String cellContent= "";                          //
    定义存放单元格内容的变量
        if (i == 0 || i == 3) {
          cellContent=String.valueOf(rs.getInt(i+1));
      //从记录集获得单元格内容
        } else {
          cellContent= rs.getString(i+1);                //
      从记录集获得单元格内容
        }
        cell= row.createCell((short)
    i);                    //创建单元格对象
        cell.setCellType(HSSFCell.CELL_TYPE_STRING);    /
    /设置单元格类型
        cell.setCellValue(cellContent);
      //指定单元格的值
      }
      rowIndex++;                                        //调
  整单元格内容的行索引值
    }
    FileOutputStream fout=new
FileOutputStream(filename);      //创建导出Excel的输出流
    workbook.write(fout);
//将工作簿写入输出流
    fout.flush();
//刷新输出流并强制写出所有缓冲的输出字节
    fout.close();
//关闭输出流对象
```

```
        JOptionPane.showMessageDialog(null, "已经成功地将报表
    \n导出到“C:/report.xls”中。");
      } catch (SQLException e) {
        e.printStackTrace();
      }
    }
```

举一反三

根据本实例，读者可以实现以下功能。

设置导出到Excel表格中的数据类型。

将导出的报表保存到磁盘上。

实例399 相册特效打印程序

这是一个可以用来提高基础性能的实例

实例位置：光盘\mingrisoft\12\Ex12_399

实例说明

本实例实现了相册特效的打印程序。运行程序，单击窗体上的“选择文件”按钮选择一幅图片，然后用鼠标单击边框样式，将显示相册特效的图片，效果如图 12.15 所示，并且还可以通过“页面设置”按钮进行页面设置，单击“开始打印”按钮，可以打印相册特效图片。

图12.15 相册特效打印程序的预览效果

技术要点

本实例通过使用透明背景的图片（该图片中心是空白的，没有任何内容，只是图片的边缘绘制有图像），实现作为相册特效图片的边框，然后将其绘制到其他图片之上，就仿佛为图片添加了相册边框，其中绘制相册效果边框的代码如下：

```
g2.drawImage(src,x,y, (int) imgWidth, (int) imgHeight, this);                    //绘制正常图像，如图片、照片等
/********************** 绘制边框**************************/
if (!border.equals("")) {
```

```
    ImageIcon icon =
SwingResourceManager.getIcon(MainFrame.class, "/com/zzk/"+
border +".png");
    //获取ImageIcon对象
    Image borderImg=
icon.getImage();                                    //获取用于绘
制边框的Image对象
    g2.drawImage(borderImg,x,y, (int) imgWidth, (int)
imgHeight, this);      //绘制边框
  }
  /*************************************************/
```

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承自 JLabel 类的 ImgLabel 标签类，用于显示相册效果的边框样式，该类的代码如下：

```
  public class ImgLabel extends JLabel {
    public ImgLabel() {
      super();
      //调用超类的构造方法
    }
    @Override
    public void paint(Graphics g) {
      try {
        int width=
    this.getWidth();                                    //获取图
    片宽度
```

```
        int height=
    this.getHeight();                              //获取图片
    高度
        ImageIcon icon= (ImageIcon)
    getIcon();                          //获取ImageIcon对象
        if (icon !=null)
    {                                              //图片不为空
          g.drawImage(icon.getImage(),0,0,width,height,
      this);       //绘制图片
        }
      } catch (Exception e) {
        e.printStackTrace();
          //输出异常信息
      }
    }
  }
```

（3）在项目中创建一个继承JFrame类的MainFrame窗体类，在该类中定义一个printPicture()方法，用于绘制原图片或照片，以及在图片或照片上绘制相册效果的边框，该方法的代码如下：

```
  public void printPicture(Graphics graphics, PageFormat
pageFormat, int pageIndex) {
      int x= (int)
  pageFormat.getImageableX();                          //获取
  可打印区域的x坐标
      int y= (int)
  pageFormat.getImageableY();                          //获取
  可打印区域的y坐标
```

```
  Graphics2D g2 = (Graphics2D) graphics;
  if (imgFile != null && isPreview) {
    try {
      src=
  ImageIO.read(imgFile);                              //构
  造Image对象
    } catch (IOException e) {
      e.printStackTrace();
    }
    double imgWidth=
src.getWidth(this);                                  //获取图片的宽
    double imgHeight=
src.getHeight(this);                                  //获取图片的
高
    double percent= imgWidth /
imgHeight;                                 //宽高比例
    int mw= (int) pf.getWidth() - x *
2;                                  //计算可打印区域的宽
    int mh= (int) pf.getHeight() - y*
2;                                  //计算可打印区域的高
    if (imgWidth>mw)
{                                              //如果宽度大于可打
印区域
      imgWidth=mw;                                                 /
  /宽度等于可打印区域的宽度
      imgHeight=mw*
  percent;                                    //通过百分比计算
```

```
  高度
    }
    if (imgHeight>mh)
{                                          //如果高度大于可打
印区域
      imgHeight=mh;
  //高度等于可打印区域的高度
      imgWidth=mh *
  percent;                                  //通过百分比计算
  宽度
    }
    g2.drawImage(src,x,y, (int) imgWidth, (int)
imgHeight, this);          //绘制正常图像，如图片、照片等
    /**********************绘制边框
************************/
    if (!border.equals("")) {
      ImageIcon
  icon=SwingResourceManager.getIcon(MainFrame.class,"/com
  /zzk/"+border+ ".png");                    //获取
  ImageIcon对象
      Image borderImg=
  icon.getImage();                           //获取用于绘制
  边框的Image对象
      g2.drawImage(borderImg,x,y, (int) imgWidth, (int)
  imgHeight, this);    //绘制边框
    }
```

```
        /**************************************************
    */
      }
      isPreview=
  false;                                          //设置不
  可以打印
    }
```

举一反三

根据本实例，读者可以实现以下功能。

在打印的图片上绘制商标。

在打印的图片上绘制注册信息。

实例400 镜面效果文本打印

这是一个可以提高基础性能的实例

实例位置：光盘\mingrisoft\12\Ex12_400

实例说明

本实例实现了镜面效果的文本打印。运行程序，效果如图 12.16 所示，然后单击窗体上的“镜面效果/还原”按钮，将水平翻转文字，就像在镜子里的成像效果，如图 12.17 所示，此时如果再次单击“镜面效果/还原”按钮，又将恢复为图12.16的效果。

图12.16 实现镜面效果之前的内容

图12.17 实现镜面效果之后的内容

技术要点

本实例通过将文本域绘制到缓冲图像对象上，然后使用Java的绘图技术水平翻转缓冲图像对象，从而实现了文本的镜面效果打印。

（1）将文本域绘制到缓冲图像对象上，是通过从 JComponent 类继承的 print()方法， JComponent类的print()方法定义如下：

public void print(Graphics g)

参数说明

g：绘图上下文对象。

（2）使用Graphics类的drawImage()方法，实现图像的翻转，从而实现镜面效果文本的绘制。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承自JPanel类，并实现Printable接口的PrintPanel打印面板类，在该类中实现Printable接口中的print()方法，在该方法中完成打印内容的绘制，其中实现镜面效果文本的代码如下：

```
if (pageIndex<1&& textArea !=null && reverse)
{            //如果当前打印页数小于1并且开启镜面效果
    BufferedImage image=new BufferedImage(pwidth -
px,pheight -
py,BufferedImage.TYPE_INT_RGB);                    //创建缓冲图像对象
    Graphics2D graphics=
image.createGraphics();              //获取图片对象的绘图上下文
    graphics.setColor(Color.WHITE);                              //设置前景色为白色
```

```
        graphics.fillRect(0, 0, image.getWidth(),
    image.getHeight());     //使用白色填充界面
        graphics.setColor(Color.BLACK);                          //
    设置前景色为黑色
        Font font=
    textArea.getFont();                          //获取文本域组件
    的字体对象
        graphics.setFont(font);                                  //
    把字体对象设置给图片的绘图上下文
        textArea.print(graphics);                                //
    把文本域界面绘制到缓冲图像对象上
        image.flush();                                           //刷
    新图片绘图缓冲区
        g.drawImage(image,px,py,pwidth,pheight,
    image.getWidth(),0,0,image.getHeight(),
    this);                          //反向绘制打印内容，实现镜面效
    果
        return Printable.PAGE_EXISTS;                            //
    返回可打印标识
      } else {                                                   //否
则
        return Printable.NO_SUCH_PAGE;                           //返
    回不支持打印标识
      }
```

（3）在项目中创建一个继承JFrame类的PrintReverseTextFrame窗体类，该窗体类中的“镜面效果/还原”按钮事件，用于实现原文本与镜面效果文本的转换，其代码如下：

```
reverseBtn.addActionListener(new ActionListener() {
  public void actionPerformed(final ActionEvent e) {
    boolean reverse=
reverseBtn.isSelected();              //获得按钮的按下状态
    printPanel.setReverse(reverse);                    //
重新绘制打印面板
  }
});
```

举一反三

根据本实例，读者可以实现以下功能。

在原有图片的右侧绘制对原有图片进行水平翻转的图片。

在原有图片的下方绘制对原有图片进行垂直翻转的图片。

实例401 透明的打印预览对话框

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\12\Ex12_401

实例说明

本实例实现了一个透明的打印预览对话框。运行程序，在窗体上可以透出后面的内容，单击窗体上的“选择文件”按钮选择图片，然后单击“打印预览”按钮进行页面打印预览，效果如图12.18所示。

图12.18 透明的打印预览对话框效果

技术要点

本实例通过使用 AWTUtilities 类的 setWindowOpacity()方法，改变窗体的透明度使窗体透明，从而实现了透明的打印预览对话框。

从JDK 6.0 开始，可以使用AWTUtilities 类的setWindowOpacity()方法设置窗体透明度，从而实现窗体的透明效果，该方法的定义如下：

public static void setWindowOpacity(Window win, float opacity)

参数说明

● win：需要调整透明度的窗体控件。

● opacity：窗体的透明度，为1.0f 时不透明；为0.0f 时完全透明；在0.0f~1.0f 为不完全透明状态。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承自JFrame类，并实现Printable接口的MainFrame窗体类，在该类的main()主方法中，添加实现窗体透明的效果，代码如下：

```
public static void main(String args[]) {
  MainFrame frame=new MainFrame();          //创建窗体对象
  AWTUtilities.setWindowOpacity(frame,0.8f);    //设置窗体80%透明
  frame.setVisible(true);                   //显示窗体
}
```

（3）在MainFrame窗体类中，实现Printable接口中的print()方法，在该方法中完成打印内容的绘制，代码如下：

```
public int print(Graphics graphics, PageFormat pageFormat, int pageIndex) throws PrinterException {
  printPicture(graphics,pageFormat,pageIndex);  //绘制打印内容
  return Printable.PAGE_EXISTS;                 //返回PAGE_EXISTS
}
```

举一反三

根据本实例，读者可以实现以下功能。

显示窗体的标题栏和边框。

实现透明打印。

第13章 操作PDF

创建PDF文档

读取PDF文档

绘制PDF图形和图像

13.1 创建PDF文档

实例402 创建PDF文档

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\13\Ex13_402

实例说明

本实例演示了如何通过Java应用程序创建PDF文档。运行程序，将在本地系统的C盘根目录下，创建一个名为“创建第一个PDF文档.pdf”的文件，双击该文件可以看到所创建PDF文档的显示效果，如图13.1所示。

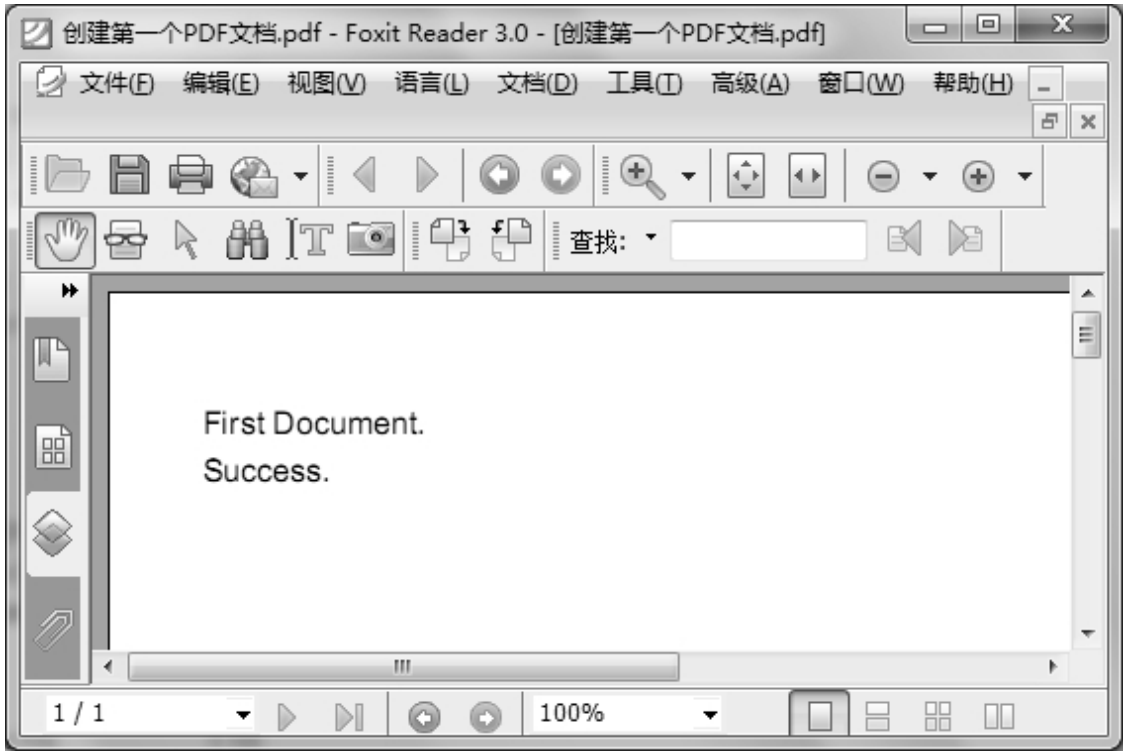


图13.1 创建的PDF 文档

技术要点

本实例主要是通过Document类创建文档对象，然后使用PdfWriter类的getInstance()方法将该文档对象与磁盘文件关联，再使用Document 类的 open()方法打开文档对象，然后再通过Document类的add()方法向文档中添加段落文本，最后使用Document类的close()方法关闭文档对象，完成PDF文档的创建。

实现过程

（1）新建一个项目。

（2）在项目中创建一个 CreateDocumentDemo 类，该类中只有一个 main()主方法，在应用程序启动后，将执行该方法的代码，完成文档的创建。CreateDocumentDemo类的完整代码如下：

```
public class CreateDocumentDemo {
  public static void main(String[] args) {
    try {
      Document document=new
  Document();                                    //创建文
  档对象
      PdfWriter.getInstance(document, new
  FileOutputStream("c:\\创建第一个PDF文档.pdf"));
      //关联文档对象与输出流
      document.open();
                    //打开文档
      document.add(new Paragraph("First
  Document."));                                  //向文档中添加
  内容
      document.add(new
  Paragraph("Success."));
    //向文档中添加内容
```

```
            document.close();
                                //关闭文档
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (DocumentException e) {
            e.printStackTrace();
        }
            }
}
```

举一反三

根据本实例，读者可以开发以下程序。

在文档中插入内容。

实例403 为PDF文档添加水印

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\13\Ex13_403

实例说明

PDF文档创建后，除了可以对页面进行修饰外，还可以为其添加水印，并将添加有水印的内容存储为一个新的PDF文档。运行程序，可以看到添加水印之后的效果，如图13.2所示。

图13.2 文档添加水印之后的效果

技术要点

本实例主要是创建一个空白PDF文档的临时文件，然后使用PdfReader对象，将已有的PDF空白文档的内容与输出流对象关联后创建PdfStamper对象，再使用该对象的getUnderContent()方法获得要为其添加水印内容的PdfContentByte对象，并添加水印图片，然后向文档中添加内容，最后将空白的PDF临时文件删除，从而实现为PDF文档添加水印的功能。创建空白PDF文档的实现代码如下：

```
Document document=new
Document();                                    //创建文档对象
    PdfWriter.getInstance(document, new
FileOutputStream("c:\\tempWatermark.pdf"));
    //关联文档对象与临时文件的输出流
    document.open();
      //打开文档
    document.add(new Paragraph("
"));                                    //向文档中添加内容
```

```
  document.close();
```

实现过程

（1）创建一个Java工程。

（2）向工程中添加一个AddWatermark类，向该类中添加main()方法。主要代码如下：

```
public static void main(String[] args) {
  Document document=new
Document();                          //创建文档对象
  try{
    PdfWriter.getInstance(document,new
FileOutputStream("c:\\tempWatermark.pdf"));
      //关联文档对象与临时文件的输出流
    document.open();                                    //
打开文档
    document.add(new Paragraph("
"));                          //向文档中添加内容
    document.close();                                    //
关闭文档对象
    PdfReader reader = new
PdfReader("c:\\tempWatermark.pdf"); //创建
tempWatermark.pdf的PdfReader对象
    PdfStamper stamp = new PdfStamper(reader, new
FileOutputStream("c:\\添加水印.pdf"));
    //创建PdfStamper对象
    Image img=
Image.getInstance("image/watermark.jpg");      //创建图像
对象
```

```
    img.setAbsolutePosition(50,385);
  //定位图片对象
    PdfContentByte under=
stamp.getUnderContent(1);          //获得第一页的内容
    //添加图片，完成水印功能under.addImage(img);
    BaseFont chinese=BaseFont.createFont("STSong-
Light","UniGB-UCS2-H",BaseFont.NOT_EMBEDDED);        //定
义字体
    under.beginText();
      //标记文本开始
    under.setFontAndSize(chinese,42);
      //设置字体和字号
    under.setTextMatrix(70,550);
        //设置添加内容的显示位置
    under.showText("下面是添加的水印图
片.");                        //添加内容
    under.endText();
    //标记文本结束
    stamp.close();    //PdfStamper对象，将从
tempWatermark.pdf中读取的文档添加水印后写入“添加水
印.pdf”
    File file=new
File("c:\\tempWatermark.pdf");                    //创建临
时文件的File对象
    file.delete();
      //删除临时文件
  } catch (FileNotFoundException e) {
```

```
            e.printStackTrace();
        } catch (DocumentException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

举一反三

根据本实例，读者可以实现以下功能。

为文档的多页添加水印。

为文档的多页添加文本内容。

实例404 在PDF文档中显示中文

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\13\Ex13_404

实例说明

开发人员在使用iText开发包创建PDF文档向文档中输入数据时会发现，英文可以正常显示，但是中文却显示不出来。本实例实现了在PDF文档中显示中文的功能，效果如图13.3所示。

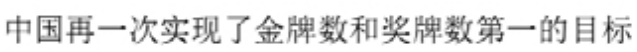


图13.3 在PDF 文档中显示中文

技术要点

在PDF文档中显示中文主要是在设置文本时指定一个支持中文的字体。例如，在定义字体时使用STSong- Light 为字体名称，采用UniGB-UCS2-H 字符编码，就可以显示中文了。代码如下：

```
    BaseFont Chinese = BaseFont.createFont("STSong-Light",
"UniGB-UCS2-H", BaseFont.NOT_EMBEDDED);
    //定义基础字体
    Font FontChinese=new
Font(Chinese,20,Font.NORMAL);              //实例化字体类与设置
字体大小属性
```

实现过程

（1）创建一个Java工程。

（2）向工程中添加一个HandleChineseErrorCode类，向该类中添加main()方法。主要代码如下：

```
    public static void main(String[] args){
      Document document=new
  Document();                          //创建文档对象
      try{
        PdfWriter.getInstance(document,newFileOutputStream("C
    :\\亚运速递.pdf"));
        //关联文档对象与输出流
        document.open();                                    //
    打开文档
        BaseFont Chinese=BaseFont.createFont("STSong-Light",
    "UniGB-UCS2-
    H",BaseFont.NOT_EMBEDDED);
                    //定义基础字体
        Font FontChinese = new Font(Chinese, 20,
    Font.NORMAL); //实例化字体类与设置字体大小属性
        document.add(new Paragraph("中国再一次实现了金牌数和
    奖牌数第一的目标", FontChinese));
```

```
            //向文档中添加内容并指定中文
            document.close();
            //关闭文档
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (DocumentException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

举一反三

根据本实例，读者可以实现以下功能。

在PDF文档中显示中文。

设置中文的显示样式。

实例405 为PDF文档添加章节

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\13\Ex13_405

实例说明

为了方便对PDF文档的阅读，可以为PDF文档添加章节信息，同样也可以在章节中添加小节信息，本实例实现了在章节中添加小节信息的功能，效果如图13.4所示。

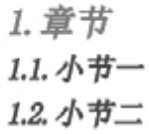


图13.4 在章节中添加小节后的效果

技术要点

本实例主要使用Chapter类创建章节对象，然后使用Chapter类的addSection()方法向章节中添加小节，最后再将章节对象添加到Document对象中。向章节对象中添加小节的代码如下：

```
paragraph=new Paragraph("小节
一",fontChinese2);              //创建段落对象
chapter.addSection(paragraph);
//在章节中添加小节
```

实现过程

（1）创建一个Java工程。

（2）向工程中添加一个AddSectionInChapter类，向该类中添加main()方法。主要代码如下：

```
public static void main(String[] args){
  Document document=new
Document();                              //创建文档对象
  try{
    PdfWriter.getInstance(document,newFileOutputStream("C
:\\在章节中添加小节.pdf"));
    //关联文档对象与输出流
    document.open();
    //打开文档
```

```
    BaseFont Chinese = BaseFont.createFont("STSong-
Light", "UniGB-UCS2-H", BaseFont.NOT_EMBEDDED);
    //定义基础字体
    Font fontChinese1 = new Font(Chinese, 18,
Font.BOLDITALIC,BaseColor.RED);
    //实例化字体类、设置字体大小和颜色
    Font fontChinese2 = new Font(Chinese, 15,
Font.BOLDITALIC,BaseColor.BLUE);
    //实例化字体类、设置字体大小和颜色
    Paragraph paragraph=new Paragraph("章
节",fontChinese1);        //创建段落对象
    Chapter chapter=new
Chapter(paragraph,1);                    //创建章节对象
    paragraph=new Paragraph("小节
一",fontChinese2);              //创建段落对象
    chapter.addSection(paragraph);
        //添加小节
    paragraph=new Paragraph("小节
二",fontChinese2);              //创建段落对象
    chapter.addSection(paragraph);
        //添加小节
    document.add(chapter);
      //向文档添加章节
    document.close();
    //关闭文档
  } catch (FileNotFoundException e) {
    e.printStackTrace();
```

```
    } catch (DocumentException e) {
      e.printStackTrace();
    } catch (IOException e) {
      e.printStackTrace();
    }
}
```

举一反三

根据本实例，读者可以开发以下程序。

添加多个章节并为每个章节添加小节。

设置各章节的字体颜色。

13.2 读取PDF文档

实例406 读取普通PDF文档

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\13\Ex13_406

实例说明

PDF 文档创建后，可以通过程序读取其中的文本信息。本实例实现了通过程序读取 PDF文档中文本信息的功能。运行程序，在控制台显示读取到的文本内容，如图13.5所示。

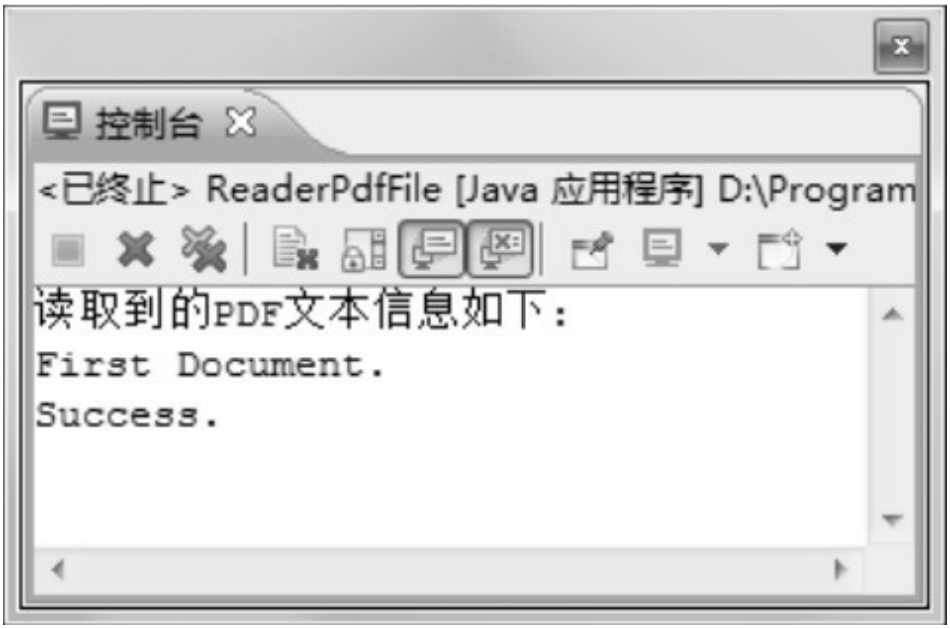


图13.5 从PDF 文档中读取的文本内容

技术要点

本实例主要通过PDFBox开源工具，实现了对PDF文件中文本内容的读取，关键代码如下：

```
    PDFParser parser=new
PDFParser(in);                              //创建PDF解析器
    parser.parse();
    //解析PDF文档
    PDDocument
pdfdocument=parser.getPDDocument();                  //获得解析
后的PDF文档
    PDFTextStripper stripper=new
PDFTextStripper();                      //创建PDF文本剥离器
    String msg=
stripper.getText(pdfdocument);                          //使用
剥离器从PDF文档中剥离文本信息
```

实现过程

（1）创建一个Java工程。

（2）向工程中添加一个ReaderPdfFile类，向该类中添加main()方法。主要代码如下：

```
    public static void main(String[] args) throws
MalformedURLException {
      Document document=new
  Document();                           //创建文档对象
      File file=new File("c:\\创建第一个PDF文
  档.pdf");                 //创建File对象
      try{
        FileInputStream in=new
    FileInputStream(file);              //创建输入流对象
        PDFParser parser=new
    PDFParser(in);                      //创建PDF解析器
```

```
    parser.parse();
//解析PDF文档
    PDDocument
pdfdocument=parser.getPDDocument();          //获得解析后的
PDF文档
    PDFTextStripper stripper=new
PDFTextStripper();              //创建PDF文本剥离器
    String msg=
stripper.getText(pdfdocument);                    //使用剥离
器从PDF文档中剥离文本信息
    System.out.println("读取到的PDF文本信息如
下:\n"+msg);     //输出信息
    in.close();                                          //
关闭输入流对象
  } catch (Exception e) {
    e.printStackTrace();
  }
  //关闭文档document.close();
}
```

举一反三

根据本实例，读者可以实现以下功能。

为PDF文档加密。

读取PDF文档。

实例407 读取加密的PDF文档

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\13\Ex13_407

实例说明

如何通过程序读取加密的PDF文档呢？本实例实现了该功能。运行程序，在控制台显示从加密的PDF文档中读取的文本内容，效果如图13.6所示。

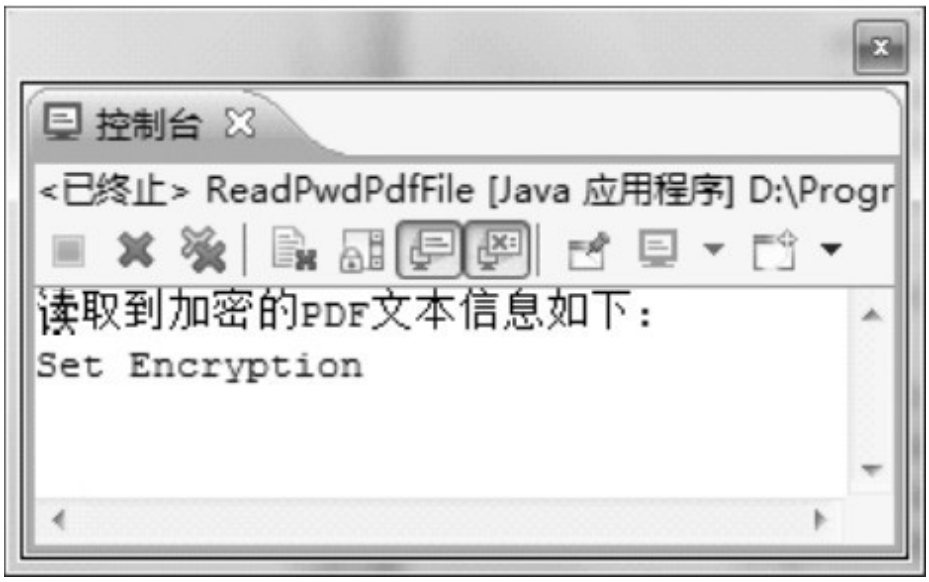


图13.6 读取加密的PDF 文档内容

技术要点

本实例主要通过PdfReader类对加密的PDF文档进行解密，然后将解密后的内容存储到一个临时的PDF文件中，并使用PDFBox开源工具，从这个临时的PDF文件中读取文本内容。

使用PDFReader类将加密的PDF文档读取到临时文件中的代码如下：

```
PdfReader reader=new PdfReader("c:\\设置密码.pdf",
"123".getBytes());                    //创建“水印.pdf”的PdfReader对象
PdfStamper stamp=new PdfStamper(reader, new
FileOutputStream("c:\\TempFile.pdf"));      //创建PdfStamper对象
```

stamp.close(); //关闭PdfStamper对象，并将从“设置密码.pdf”中读取的内容写入TempFile.pdf

实现过程

（1）创建一个Java工程。

（2）向工程中添加一个ReadPwdPdfFile类，向该类中添加main()方法。主要代码如下：

```
public static void main(String[] args) throws MalformedURLException {
  try{
    PdfReaderreader=newPdfReader("c:\\设置密码.pdf","123".getBytes());
    //创建“水印.pdf”的PdfReader对象
    PdfStamper stamp = new PdfStamper(reader, new FileOutputStream("c:\\TempFile.pdf"));
    //创建PdfStamper对象
    stamp.close();
      //关闭PdfStamper对象，并将从“设置密码.pdf”中读取的内容写入TempFile.pdf
    Document document=new Document(); //创建文档对象
    File file=new File("c:\\TempFile.pdf"); //创建File对象
    try{
      FileInputStream in=newFileInputStream(file); //创建输入流对象
```

```
    PDFParser
parser=newPDFParser(in);                    //创建PDF
    parser.parse();
  //解析PDF文档
    PDDocument
pdfdocument=parser.getPDDocument();         //获得解析后
的PDF文档
    PDFTextStripper
stripper=newPDFTextStripper();              //创建PDF文本剥
离器
    Stringmsg=
stripper.getText(pdfdocument);              //使用剥离器
从PDF文档中剥离文本信息
    System.out.println("读取到加密的PDF文本信息如
下:\n"+ msg);//输出信息
    in.close();
      //关闭输入流对象
  } catch (Exception e) {
    e.printStackTrace();
  }
  document.close();
      //关闭文档
  file.delete();
        //删除TempFile.pdf
} catch (FileNotFoundException e) {
  e.printStackTrace();
} catch (DocumentException e) {
```

```
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

举一反三

根据本实例，读者可以实现以下功能。

解密PDF文档。

实例408 编辑PDF文档

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\13\Ex13_408

实例说明

在PDF文档创建以后，如果对文档内容不满意，可以对文档进行编辑，比如添加页码、文本内容等。本实例实现了编辑PDF文档的功能，图13.7和图13.8所示分别是原文档的内容和对原文档进行编辑后的内容。

图13.7 原文档编辑前的内容

First

新添加的内容1

图13.8 原文档编辑后的内容

技术要点

本实例将原文件中的内容保存到临时文件中，然后对临时文件进行编辑，编辑后删除原文件，并将临时文件重命名为原文件的名称，从而实现了对原文件进行编辑的功能。其中，删除原文件，并将临时文件重命名的实现代码如下：

```
File oldFile=new File("c:\\原文档.pdf");                    //创建原文件的File对象
oldFile.delete();
        //删除原文件
File tempFile=new File("c:\\编辑后文档的临时文件.pdf");          //创建临时文件的File对象
tempFile.renameTo(oldFile);
        //重命名临时文件为原文件名
tempFile.delete();
        //删除临时文件
```

实现过程

（1）创建一个Java工程。

（2）向工程中添加一个EditPdfDocumentDemo类，在该类中创建一个createOldFile()方法，用于创建“原文档.pdf”文件，其代码如下：

```
    public static void createOldFile()
{                                          //创建原文件的方法
      Document document=new
  Document();                              //创建文档对象
      try {
        PdfWriter
          .getInstance(document, new FileOutputStream("c:\\原
      文档.pdf"));//关联文档对象与输出流
        document.open();
        //打开文档
        document.add(new
    Paragraph("First"));                   //向文档中添
    加内容
        document.newPage();
        document.add(new
    Paragraph("Second"));                  //向文档中添加
    内容
        document.newPage();
        document.add(new
    Paragraph("Third"));                   //向文档中添
    加内容
        document.close();
        //关闭文档对象
      } catch (FileNotFoundException e) {
        e.printStackTrace();
      } catch (DocumentException e) {
        e.printStackTrace();
```

```
    }
  }
```

（3）在EditPdfDocumentDemo类中再创建一个editOldFile()方法，用于对“原文档.pdf”文件进行编辑，本实例向“原文档.pdf”文件中添加了页码和文本内容，其代码如下：

```
  public static void editOldFile()
{                                          //编辑原文件的方法
    try {
      PdfReader reader=new PdfReader("c:\\原文
  档.pdf");              //创建“原文档.pdf”的PdfReader对
  象
      int totalPages=
  reader.getNumberOfPages();                    //获得总页
  数
      PdfStamper stamp=new PdfStamper(reader,new
  FileOutputStream("c:\\编辑后文档的临时文
  件.pdf"));                  //创建PdfStamper对象
      BaseFont chinese=BaseFont.createFont("STSong-
  Light","UniGB-UCS2-H",BaseFont.NOT_EMBEDDED);        //定
  义字体
      PdfContentByte under = null;
      for (int i = 1; i <= totalPages; i++) {
        under=
    stamp.getUnderContent(i);                          //获得
    每一页的内容
        under.beginText();
        //标记文本开始
```

```
        under.setFontAndSize(chinese,18);
        //设置字体和字号
        under.setTextMatrix(200,810);
        //设置页码的显示位置
        under.showText("第"+ i+
    "页");                          //添加页脚
        under.endText();
      //标记文本结束
        under.beginText();
        //标记文本开始
        under.setFontAndSize(chinese,32);
        //设置字体和字号
        under.setTextMatrix(100,750);
        //设置文本的显示位置
        under.showText("新添加的内容"+
    i);                         //添加新文本
        under.endText();
      //标记文本结束
      }
      stamp.close();     //PdfStamper对象，将从“原文
档.pdf”中读取的文档添加页码后写入“编辑后文档的临时文
件.pdf”
      File oldFile=new File("c:\\原文
档.pdf");                          //创建原文件的File对象
      oldFile.delete();
        //删除原文件
```

```
        File tempFile=new File("c:\\编辑后文档的临时文
    件.pdf");          //创建临时文件的File对象
        tempFile.renameTo(oldFile);
          //重命名临时文件为原文件名
        tempFile.delete();
          //删除临时文件
      } catch (FileNotFoundException e) {
        e.printStackTrace();
      } catch (DocumentException e) {
        e.printStackTrace();
      } catch (IOException e) {
        e.printStackTrace();
      }
    }
```

举一反三

根据本实例，读者可以实现以下功能。

在PDF中添加页码。

编辑文本内容。

实例409 导入并添加水印

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\13\Ex13_409

实例说明

PDF文档创建后，为了对页面进行修饰，还可以为其添加水印，然后将添加有水印的内容存储为一个新的PDF文档。运行程序，可以看到

添加水印之前的原文档和导入并添加水印之后新文档的显示效果，如图13.9和图13.10所示。

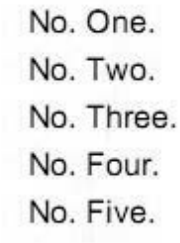


图13.9 添加水印之前的效果

图13.10 添加水印之后的效果

技术要点

本实例主要是使用 PdfReader 对象，将已有 PDF 文档的内容与输出流对象关联后创建PdfStamper 对象，然后使用该对象的getUnderContent()方法获得要为其添加水印内容的PdfContentByte对象，从而实现为PDF文档添加水印的功能。

为PDF文档添加水印的实现代码如下：

```
    PdfReader reader=new PdfReader("c:\\水
印.pdf");                                    //创建“水印.pdf”的
PdfReader对象
    PdfStamper stamp=new PdfStamper(reader,new
FileOutputStream("c:\\添加水印.pdf"));       //创建PdfStamper
对象
    Image img=
Image.getInstance("image/watermark.jpg");
          //写上内容
    img.setAbsolutePosition(30,385);
                        //定位图片对象
    PdfContentByte under=
stamp.getUnderContent(1);                                    //获
得第一页的内容
    under.addImage(img);
                  //添加图片，完成水印功能
    stamp.close();                  //PdfStamper对象，将从“水
印.pdf”中读取的文档添加水印后写入“添加水印.pdf”
```

实现过程

（1）创建一个Java工程。

（2）向工程中添加一个ImportAndAddWatermark类，向该类中添加main()方法。主要代码如下：

```
    public static void main(String[] args) {
      Document document=new
  Document();                                    //创建文档对象
      try{
```

```
    PdfWriter.getInstance(document,new
FileOutputStream("c:\\水印.pdf"));    //关联文档对象与输
出流
    document.open();                                    //
打开文档
    document.add(new
Paragraph("No.One."));                                  //向文档
中添加内容
    document.add(new
Paragraph("No.Two."));                                  //向文档
中添加内容
    document.add(new
Paragraph("No.Three."));                                //向文档
中添加内容
    document.add(new
Paragraph("No.Four."));                                  //向文档
中添加内容
    document.add(new
Paragraph("No.Five."));                                  //向文档
中添加内容
    document.close();
        //关闭文档对象
    PdfReader reader=new PdfReader("c:\\水
印.pdf");                  //创建“水印.pdf”的PdfReader对
象
    PdfStamper stamp = new PdfStamper(reader, new
FileOutputStream("c:\\添加水印.pdf"));
```

```
    //创建PdfStamper对象
    Image img=
Image.getInstance("image/watermark.jpg");          //写
上内容
    img.setAbsolutePosition(30,385);
        //定位图片对象
    PdfContentByte under=
stamp.getUnderContent(1);              //获得第1页的内容
    under.addImage(img);
    //添加图片，完成水印功能
    stamp.close();
      // PdfStamper对象，将从“水印.pdf”中读取的文档添加
  水印后写入“添加水印.pdf”
  } catch (FileNotFoundException e) {
    e.printStackTrace();
  } catch (DocumentException e) {
    e.printStackTrace();
  } catch (IOException e) {
    e.printStackTrace();
  }
}
```

举一反三

根据本实例，读者可以实现以下功能。

添加水印。

设置水印图片的旋转角度。

实例410 拆分PDF文档

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\13\Ex13_410

实例说明

创建完PDF文档后，还可以对PDF文档进行拆分。本实例实现了拆分PDF文档的功能，运行程序，将原来的文档拆分成多个新的子文档，图13.11和图13.12所示分别是原文档拆分后生成的两个新文档。

图13.11 拆分后生成的第1 个文档

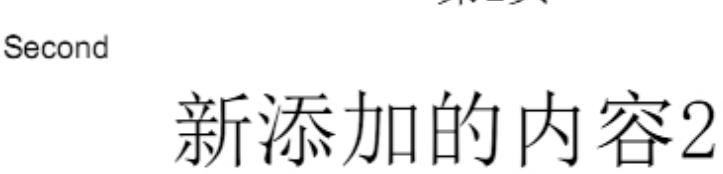


图13.12 拆分后生成的第2 个文档

技术要点

本实例主要是使用PdfReader类和PdfCopy类实现将原文档拆分成多个子文档的功能，其中新文档中的数据通过PdfReader对象获得，子文档中的内容通过PdfCopy对象的addPage()方法进行添加。实现拆分PDF文档功能的关键代码如下：

```
copy=new PdfCopy(document,new
FileOutputStream(fileName));                    //创建复本并关联文档
与输出流对象
document.open();
    //打开文档
```

```
    copy.addPage(copy.getImportedPage(reader,
i+1));                          //根据获得的页创建新文档
    document.close();
        //关闭文档
```

实现过程

（1）创建一个Java工程。

（2）向工程中添加一个SplitPDFDocument类，向该类中添加main()方法。主要代码如下：

```
    public static void main(String[] args) {
      String filePathFile= "c:\\原文档.pdf
  ";                            //需要拆分的原文档
      PdfReader
  reader=null;                                  //声明
  PdfReader对象
      try{
        reader=new
  PdfReader(filePathFile);                      //创建
  PdfReader对象
      } catch (IOException e) {
        e.printStackTrace();
      }
      int pageN=
  reader.getNumberOfPages();                    //获取文件
  内的页数
    for (int i=0; i<pageN; i++)
  {                                 //循环向外拆分页
```

```
    Document document=new
Document(reader.getPageSizeWithRotation(i+1));
    //创建文档，同时获得前面循环的页
    PdfCopy copy = null;
    try {
      int len=
  filePathFile.length();                    //获得文件
  完整路径的长度
      String noExt= filePathFile.substring(0, len -
  5);        //去除文件扩展名后的路径
      String fileName=noExt+ "-"+ (i+1)+
  ".pdf";        //拆分后生成的文件名称
      copy = new PdfCopy(document, new
  FileOutputStream(fileName));
      //创建复本并关联文档与输出流对象
      document.open();                          //打
  开文档
      copy.addPage(copy.getImportedPage(reader,
  i+1));      //根据获得的页创建新文档
      document.close();                          //关
  闭文档
    } catch (FileNotFoundException e) {
      e.printStackTrace();
    } catch (DocumentException e) {
      e.printStackTrace();
    } catch (IOException e) {
      e.printStackTrace();
```

```
        }
    }
}
```

举一反三

根据本实例，读者可以实现以下功能。

使拆分后生成的子文档中包含多页内容。

将PDF分档拆分为多个子文档。

实例411 合并PDF文档

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\13\Ex13_411

实例说明

如果创建了多个PDF文档，可以将这些PDF文档合并成一个，这样就可以通过一个文档查看多个PDF文档中的内容了。本实例实现了将多个文档合并成一个文档的功能，打开合并后的文档，可以看到第2页和第3页的内容，分别如图13.13和图13.14所示。

图13.13 合并后文档第2 页的内容

图13.14 合并后文档第3 页的内容

技术要点

本实例主要是通过数组对需要合并的PDF文档进行存储，然后使用PdfCopy类和for循环语句实现了将多个PDF文档合并成一个大的PDF文档。实现合并文档的关键代码如下：

```
    for (int i=0; i< subFiles.length; i++)
{                                          //做循环，获取待合并文件长度
      PdfReader reader=new
  PdfReader(subFiles[i]);                          //读取待合并文
  件长度
      int totalPages=
  reader.getNumberOfPages();                          //获得每
  个子文档的总页数
      for (int p=1; p<= totalPages; p++)
  {                                        //遍历子文档的每一页
        copy.addPage(copy.getImportedPage(reader,p));
              //将子文档的每一页添加到新文档中
      }
    }
```

实现过程

（1）创建一个Java工程。

（2）向工程中添加一个UnitePDFDocument类，向该类中添加main()方法。主要代码如下：

```
    public static void main(String[] args) {
      String[] subFiles ={"c:\\原文档-1.pdf", "c:\\原文
  档-2.pdf", "c:\\原文档-3.pdf"}; //待合并的PDF文档
```

```
    String newFile= "C:\\合并结
果.pdf";                                    //合并后的新文档
    Document document=new
Document();                                  //创建文本文档
    try{
      PdfCopycopy=newPdfCopy(document,newFileOutputStream(n
  ewFile));
      //创建copy对象关联文档与输出流
      document.open();
      //打开文档
      for (int i=0; i< subFiles.length; i++)
  {                          //做循环，获取待合并文件长度
        PdfReader reader=new
      PdfReader(subFiles[i]);              //读取待合并文件长
      度
        int totalPages=
      reader.getNumberOfPages();                  //获得每个子
      文档的总页数
        for (int p=1; p<= totalPages; p++)
      {                      //遍历子文档的每一页
          copy.addPage(copy.getImportedPage(reader,p));
      //将子文档的每一页添加到新文档中
        }
      }
      document.close();
      //关闭文档
    } catch (FileNotFoundException e) {
```

```
            e.printStackTrace();
        } catch (DocumentException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

举一反三

根据本实例，读者可以实现以下功能。

选择需要合并的PDF文档。

合并指定的PDF文档。

13.3 绘制PDF图形和图像

实例412 使用Graphics2D绘制图形

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\13\Ex13_412

实例说明

在绘制PDF文档时，可以通过多种方式在PDF文档中绘制图形。本实例将使用Graphics2D类在PDF文档中绘制图形，效果如图13.15所示。

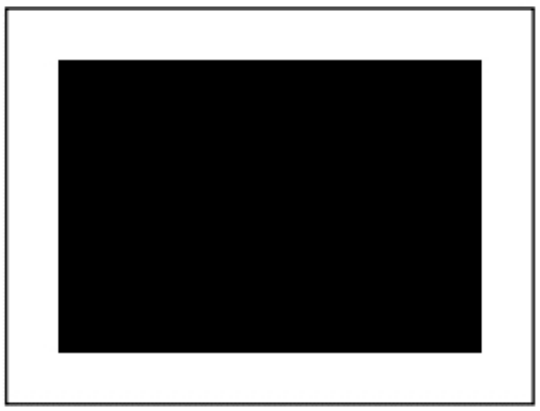


图13.15 使用Graphics2D 在PDF 中绘制的图形

技术要点

本实例主要是通过PdfContentByte类的createGraphics()方法获得指定大小的Graphics2D对象，然后使用Graphics2D类的方法绘制图形。在PDF文档中使用Graphics2D类绘制图形的代码如下：

```
Rectangle2D rect1=new
Rectangle2D.Double(50,50,200,150);                    //创建矩
```

形对象

```
    g.draw(rect1);
        //绘制矩形
```

实现过程

（1）新建一个项目。

（2）在项目中创建一个UseGraphics2D类，该类中只有一个main()主方法，在应用程序启动后将执行该方法的代码，完成在PDF文档中使用Graphics2D绘制图形的操作，主方法的代码如下：

```
    public static void main(String[] args) throws
MalformedURLException {
      Document document=new
  Document();                                    //创建文档对象
      try {
        PdfWriter writer=PdfWriter.getInstance(document,new
    FileOutputStream("C:\\使用Graphics2D绘制图
    形.pdf"));       //关联文档对象与输出流
        document.open();
              //打开文档
        PdfContentByte
    cb=writer.getDirectContent();                          //获
    取文档内容
        Graphics2D g=
    cb.createGraphics(850,850);                            //创
    建Graphics和坐标
        Rectangle2D rect1=new
    Rectangle2D.Double(50,50,200,150);              //创建矩形
    对象
```

```
        g.draw(rect1);
                //绘制矩形
        Rectangle2D rect2=new
Rectangle2D.Double(70,70,160,110);                    //创建矩形
对象
        g.fill(rect2);
                  //绘制填充矩形
        g.dispose();
                //部署
        cb.stroke();
                //确认绘制的图形
        document.close();
                //关闭文档
    } catch (DocumentException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

举一反三

根据本实例，读者可以实现以下功能。

在PDF文档中绘制多边形。

在PDF文档中为多边形添加颜色。

实例413 使用PdfGraphics2D绘制文本

这是一个可以用来提高基础性能的实例

实例位置：光盘\mingrisoft\13\Ex13_413

实例说明

在绘制PDF文档时，除了可以用前面讲的内容绘制图形以外，还可以使用PdfGraphics2D类实现文本和图形的绘制。本实例使用PdfGraphics2D实现了在PDF文档中绘制文本的功能，效果如图13.16所示。

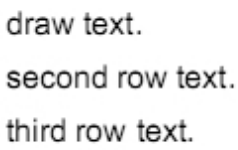


图13.16 使用PdfGraphics2D 绘制文本的效果

技术要点

本实例主要是通过PdfContentByte类的createGraphics()方法获得指定大小的Graphics2D对象，然后将该Graphics2D对象强制转换为PdfGraphics2D对象，并通过PdfGraphics2D对象的drawString()方法完成文本的绘制。在PDF文档中使用PdfGraphics2D类绘制文本的代码如下：

```
PdfGraphics2D g= (PdfGraphics2D)
cb.createGraphics(700,800);          //获得PdfGraphics2D对象
g.drawString("draw text.
",54,10);                                    //绘制文本
```

实现过程

（1）新建一个项目。

（2）在项目中创建一个PdfGraphics2DText类，该类中只有一个main()主方法，在应用程序启动后将执行该方法的代码，完成在PDF文档中使用PdfGraphics2D绘制文本的操作，主方法的代码如下：

```
    public static void main(String[] args) throws
MalformedURLException {
      Document document=new
  Document();                                    //创建文档对象
      try {
        PdfWriter writer=PdfWriter.getInstance(document,new
    FileOutputStream("C:\\使用PdfGraphics2D绘制文
    本.pdf"));    //关联文档对象与输出流
        document.open();
              //打开文档
        PdfContentByte
    cb=writer.getDirectContent();                          //获
    取文档内容
        PdfGraphics2D g= (PdfGraphics2D)
    cb.createGraphics(700,800);          //获得PdfGraphics2D
    对象
        g.drawString("draw text.
    ",54,10);                                  //绘制文本
        g.drawString("second row text.
    ",54,30);                              //绘制文本
        g.drawString("third row text.
    ",54,50);                                //绘制文本
        g.dispose();
              //部署
        cb.stroke();
              //确认绘制的内容
```

```
            document.close();
                  //关闭文档
        } catch (DocumentException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

举一反三

根据本实例，读者可以实现以下功能。

使用PdfGraphics2D绘制图形。

使用PdfGraphics2D绘制彩色字体。

实例414 使用PdfGraphics2D绘制图形

这是一个可以提高基础性能的实例

实例位置：光盘\mingrisoft\13\Ex13_414

实例说明

在绘制 PDF 文档时，还可以使用 PdfGraphics2D 类实现图形的绘制。本实例使用PdfGraphics2D实现了在PDF文档中绘制图形的功能，效果如图13.17所示。

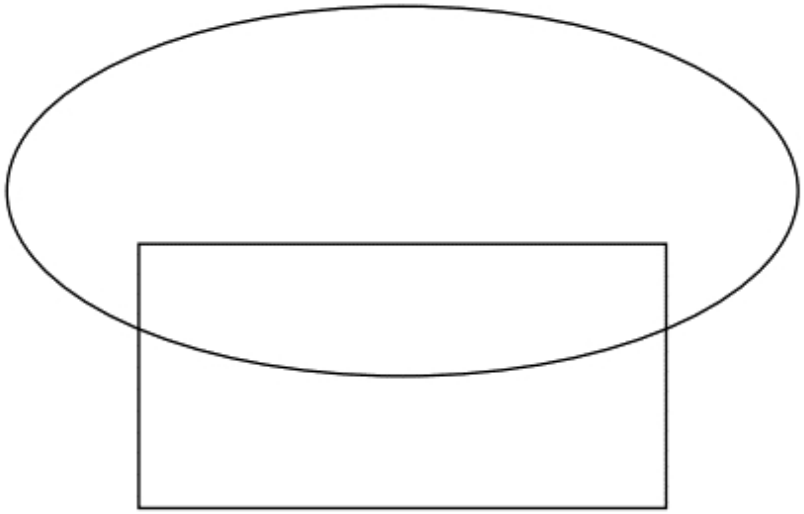


图13.17 使用PdfGraphics2D 绘制图形的效果

技术要点

本实例主要是通过PdfContentByte类的createGraphics()方法获得指定大小的Graphics2D对象，然后将该Graphics2D对象强制转换为PdfGraphics2D对象，并通过PdfGraphics2D对象的draw()方法完成图形的绘制。在PDF文档中使用PdfGraphics2D类绘制图形的代码如下：

```
    Rectangle2D rect=new
Rectangle2D.Double(120,100,200,100);          //创建矩形对象
    Ellipse2D circle=new
Ellipse2D.Double();                                  //创建椭圆对象
    circle.setFrameFromCenter(220,80,370,150);
        //设置椭圆的中心点坐标和角点坐标
    g.draw(rect);
  //绘制矩形对象
    g.draw(circle);
  //绘制圆形对象
```

实现过程

（1）新建一个项目。

（2）在项目中创建一个PdfGraphics2DShape类，该类中只有一个main()主方法，在应用程序启动后将执行该方法的代码，完成在PDF文档中使用PdfGraphics2D绘制图形的操作，主方法的代码如下：

```
    public static void main(String[] args) throws
MalformedURLException {
      Document document=new
  Document();                                    //创建文档对象
      try {
        PdfWriter writer=PdfWriter.getInstance(document,new
    FileOutputStream("C:\\使用PdfGraphics2D绘制图
    形.pdf"));    //关联文档对象与输出流
        document.open();
              //打开文档
        PdfContentByte
    cb=writer.getDirectContent();                          //获
    取文档内容
        PdfGraphics2D g= (PdfGraphics2D)
    cb.createGraphics(700,800);          //创建Graphics和坐标
        Rectangle2D rect=new
    Rectangle2D.Double(120,100,200,100);          //创建矩形
    对象
        Ellipse2D
    circle=newEllipse2D.Double();                          //
    创建椭圆对象
        circle.setFrameFromCenter(220,80,370,150);
                  //设置椭圆的中心点坐标和角点坐标
```

```
            g.draw(rect);
                    //绘制矩形对象
            g.draw(circle);
                    //绘制圆形对象
            g.dispose();
                    //部署
            cb.stroke();
                    //确认绘制的内容
            document.close();
                    //关闭文档
        } catch (DocumentException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

举一反三

根据本实例，读者可以实现以下功能。

使用PdfGraphics2D类的方法绘制长方形。

使用PdfGraphics2D类的方法绘制矩形。

实例415 在PDF文档中对齐图片

本实例可以美化界面、简化操作

实例位置：光盘\mingrisoft\13\Ex13_415

实例说明

在 PDF 文档中绘制图片以后，可以根据需要调整图片的对齐方式。本实例实现了在 PDF文档中调整图片对齐方式的功能，效果如图13.18所示。

技术要点

本实例主要使用 Image 类创建指定图片的图像对象，然后使用 Image 类的 setAlignment()方法设置图片的对齐方式。例如下面是在PDF文档中设置图片左对齐的代码。

```
Image image=
Image.getInstance("image/picture.jpg");          //创建图像对象
image.setAlignment(Image.LEFT);
//设置图片居左
```

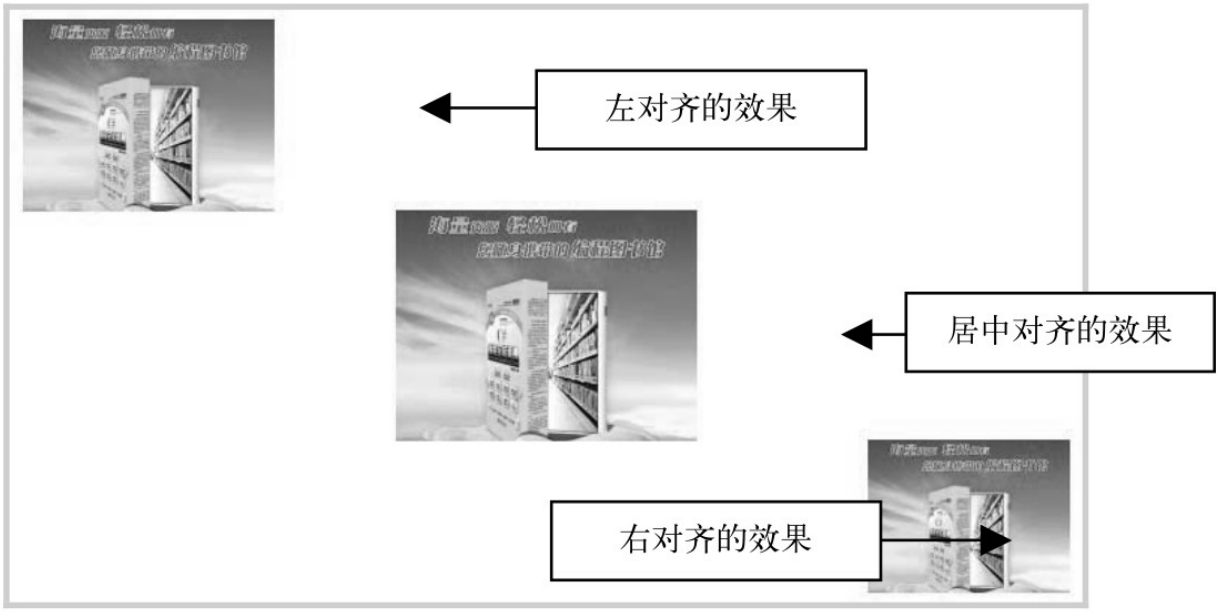


图13.18 在PDF 文档中设置图片的对齐方式

实现过程

（1）新建一个项目。

（2）在项目中创建一个 PictureAlignment 类，该类中只有一个main()主方法，在应用程序启动后执行该方法，完成在PDF文档中对齐

图片的功能，主方法的代码如下：

```
    public static void main(String[] args) {
      Document document=new
  Document();                          //创建文档对象
      try {
        PdfWriter.getInstance(document,new
    FileOutputStream("C:\\设置图片对齐方
    式.pdf"));                          //关联文档对象与输出流
        document.open();                                    /
    /打开文档
        Image image=
    Image.getInstance("image/picture.jpg");          //创建图像
    对象
        image.setAlignment(Image.LEFT);
      //设置图片居左
        image.scalePercent(25);
      //设置原图像的比例
        document.add(image);                                /
    /向文档添加图片
        image=
    Image.getInstance("image/picture.jpg");                //创
    建图像对象
        image.setAlignment(Image.MIDDLE);
    //设置图片居中
        image.scalePercent(30);
      //设置原图像的比例
```

```
    document.add(image);                                    /
/向文档添加图片
    image=
Image.getInstance("image/picture.jpg");                 //创
建图像对象
    image.setAlignment(Image.RIGHT);
  //设置图片居右
    image.scalePercent(20);
  //设置原图像的比例
    document.add(image);                                    /
/向文档添加图片
    document.close();
//关闭文档
  } catch (FileNotFoundException e) {
    e.printStackTrace();
  } catch (DocumentException e) {
    e.printStackTrace();
  } catch (MalformedURLException e) {
    e.printStackTrace();
  } catch (IOException e) {
    e.printStackTrace();
  }
}
```

举一反三

根据本实例，读者可以实现以下功能。

设置图片居中对齐。

设置图片靠右对齐。

实例416 在PDF文档中旋转图片

这是一个可以启发思维的实例

实例位置：光盘\mingrisoft\13\Ex13_416

实例说明

在PDF文档中绘制图片时，有时需要对图片进行旋转，使其以一定的旋转角度显示。本实例演示如何在PDF文档中调整图片的旋转角度，效果如图13.19所示。

图13.19 旋转图片的效果

技术要点

本实例主要使用Image类创建指定图片的图像对象，然后使用Image类的setRotation()方法设置图片的旋转角度。调整图片旋转角度的代码如下：

```
Image image=
Image.getInstance("image/picture.jpg");          //创建图像对象
image.setRotation(320);
//设置旋转弧度
```

实现过程

（1）新建一个项目。

（2）在项目中创建一个PictureRotate类，该类中只有一个main()主方法，在应用程序启动后执行该方法，完成调整PDF文档中图片旋转角度的功能，主方法的代码如下：

```
public static void main(String[] args) {
  Document document=new Document();                    //创建文档对象
    try {
      PdfWriter.getInstance(document,new FileOutputStream("C:\\旋转图片.pdf"));    //关联文档对象与输出流
      document.open();                                 //打开文档
      Image image=Image.getInstance("image/picture.jpg");    //创建图像对象
      image.setRotation(320);                          //设置旋转弧度
      document.add(image);                             //向文档添加图片
      document.close();                                //关闭文档
    } catch (FileNotFoundException e) {
      e.printStackTrace();
    } catch (DocumentException e) {
      e.printStackTrace();
    } catch (MalformedURLException e) {
```

```
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

举一反三

根据本实例，读者可以实现以下功能。

设置图片的旋转角度。

设置图片的旋转方向。

第14章 解析XML文件

使用SAX解析XML
使用DOM解析XML
使用DOM操作XML

14.1 使用SAX解析XML

实例417 解析XML元素名称和内容（SAX）

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\14\Ex14_417

实例说明

解析XML元素名称和内容的关键是当SAX解析XML时，把元素的名称和内容及时保存起来，同时 XML 中可能会有很多同名的元素，要把元素名称和内容对应起来。本实例在解析XML时，在endElement()方法中把元素和内容拼成一个字符串，然后把字符串保存在List中。有了元素和内容的 List，想实现其他业务逻辑就不难了，在这里我们只是简单地输出，效果如图14.1所示。

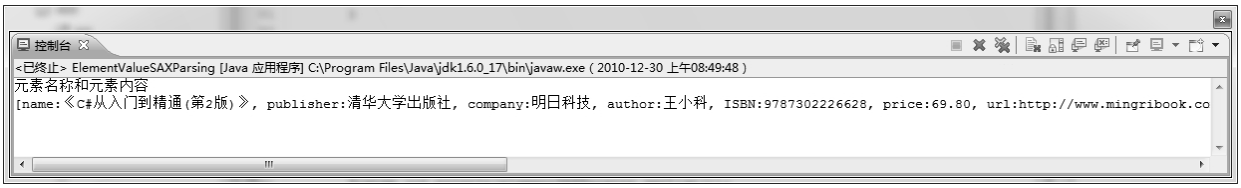


图14.1 解析XML 元素名称和内容

技术要点

（1）使用ElementValueSAXParsing类继承DefaultHandler类可以实现characters()方法，通过实现该方法，可以获取XML元素名称和内容，语法如下：

public void characters(char[] ch, int start, int length) throws SAXException

参数说明

● ch：表示XML 的内容，其内容是整个XML 文档。

- start：表示当前元素在整个XML 文档中开始的字节数。
- length：表示当前元素本身的字节长度。

（2）SAX 解析 XML 时，ElementValueSAXParsing 继承了DefaultHandler 类，并重写endElement()方法。以后在每次调用SAXParser的parse()方法时，向parse()里传入XML文件和ElementValueSAXParsing的实例。SAX读取每个元素内容时，characters()方法就会被执行，读取XML元素结束时，endElement()方法就会被调用。代码如下：

```
package com.mingrisoft.SAX_demo;
import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;
public class ElementValueSAXParsing extends
DefaultHandler {
    private List<String> list = new ArrayList<String>();
    private String value;
    /**
    *读取当前元素的内容，过滤制表符、空格符、回车符和换行符
    */
    @Override
```

```
    public void characters(char[] ch, int start, int
length)throws SAXException {
      // TODO Auto-generated method stub
      value = String.valueOf(ch, start, length);
      value = value.replace("\t", "");
      value = value.replace("", "");
      value = value.replace("\n", "");
      value = value.replace("\r", "");
    }
    /**
    *读取元素结束，把元素名称和元素内容保存在Map中
    */
    @Override
    public void endElement(String uri, String localName,
String qName)throws SAXException {
      // TODO Auto-generated method stub
      list.add(localName +":"+ value);
    }
    public List<String> getList() {
      return this.list;
    }
    /**
    *通过文件读取XML
    *
    *@param pathname
    *文件路径
    */
```

```
  public void parseReadFile(String pathname) {
    SAXParser parser;
    SAXParserFactory factory =
SAXParserFactory.newInstance();
    try{
      factory.setValidating(true);
      factory.setNamespaceAware(true);
      parser = factory.newSAXParser();
      File file = new File(pathname);
      parser.parse(file, this);
    } catch (ParserConfigurationException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
    } catch (SAXException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
    } catch (IOException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
    }
  }
  public static void main(String[] arg) {
    String pathname ="xmldemo/books.xml";
    ElementValueSAXParsing elementSAXParsing = new
ElementValueSAXParsing();
    elementSAXParsing.parseReadFile(pathname);
    System.out.println("元素名称和元素内容");
```

```
        System.out.println(elementSAXParsing.getList());
      }
    }
```

实现过程

（1）创建一个XML文档用于SAX进行解析。

（2）创建ElementNameSAXParsing类，然后继承DefaultHandler接口。在JAVA文件中创建parseReadFile()方法用于解析XML。

（3）实现characters()方法，获取当前元素的内容，把它保存在临时变量value里。代码如下：

```
    public void characters(char[] ch, int start, int length) throws SAXException {
      //读取当前元素的内容，过滤制表符、空格符、回车符和换行符
      value = String.valueOf(ch, start, length);
      value = value.replace("\t", "");
      value = value.replace("", "");
      value = value.replace("\n", "");
      value = value.replace("\r", "");
    }
```

（4）实现endElement()方法，在方法中获取localName和临时变量value值，并把元素的名称和内容一块保存在List中。代码如下：

```
    public void endElement(String uri, String localName, String qName)throws SAXException {
      //读取元素结束，把元素名称和内容保存在list中
      list.add(localName +":"+ value);
    }
```

（5）创建main()方法，在方法里使用ElementValueSAXParsing解析books.xml文件，输出元素名称到控制台。

举一反三

根据本实例，读者可以开发以下程序。

解析XML的名称。

解析XML的内容。

实例418 解析XML元素属性和属性值（SAX）

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\14\Ex14_418

实例说明

XML的属性包含属性名称和属性值，在XML中，每个元素都可能含有属性，属性是针对元素而言的。本实例的XML文档有两本图书，每个图书都有自己的价格，也就是每个book元素里面都包含一个price元素，但是它们的内容可能又是不一样的。在price元素中包含unit和unitType两个属性，每个price元素都可以有同样的属性，但是不同的price的属性值可能是不一样的。本实例讲解如何获取XML元素的属性和属性值，如图14.2所示。

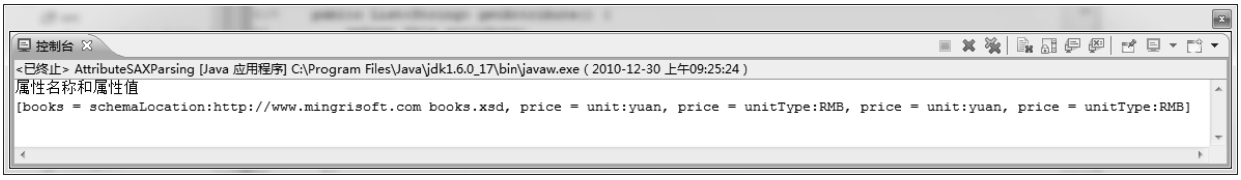


图14.2 解析XML 属性

技术要点

SAX每次开始读取XML元素时，startElement()方法都会被执行，使用AttributeSAXParsing类重写DefaultHandler的startElement()方法可以获取元素属性和属性值，语法如下：

public void startElement(String uri, String localName, String qName,Attributes attributes) throws SAXException

参数说明

● uri：表示XML 元素命名空间，在这里就是http://www.mingrisoft.com。

● localName：表示XML 元素的本地标示符，在这里就是name、publisher、company、author和ISBN等。

● qName：表示元素在XML 文件中使用的名称，在这里就是book:name、book:publisher、book:company、book:author和book:ISBN等。

● attributes：表示当前元素的属性集合。

实现过程

（1）创建一个XML文档用于SAX进行解析。

（2）创建AttributeSAXParsing类继承DefaultHandler类，重写startElement()方法读取属性和属性值，并把它们合并为一个字符串保存在 List 里。一个元素可能有多个属性，所以使用attributes 中的 getLength()方法，获取当前元素属性的个数，在这个属性集合中，使用getLocalName()方法能获取当前元素第几个属性的名称，getValue()方法中参数是几就表示当前元素的第几个属性值。代码如下：

```
public void startElement(String uri, String localName, String qName, Attributes attributes) throws SAXException {
    //读取属性名称和属性值保存在List中
    for (int i = 0; i < attributes.getLength(); i++) {
      attribute.add(localName +"="+ attributes.getLocalName(i) +":"+ attributes.getValue(i));
    }
```

```
    }
```

（3）创建parseReadFile()方法把AttributeSAXParsing实例传入解析器，实现XML的解析。代码如下：

```
    public void parseReadFile(String pathname) {
      SAXParser parser;
      //获取SAXParserFactory实例
      SAXParserFactory factory =
  SAXParserFactory.newInstance();
      try{
        factory.setValidating(true);
        factory.setNamespaceAware(true);
        //获取SAXParser实例
        parser = factory.newSAXParser();
        //获取XML文件
        File file = new File(pathname);
        parser.parse(file, this);
      } catch (ParserConfigurationException e) {
        e.printStackTrace();
      } catch (SAXException e) {
        e.printStackTrace();
      } catch (IOException e) {
        e.printStackTrace();
      }
    }
```

（4）创建main()方法，使用AttributeSAXParsing解析books.xml文件，输出属性名称和属性值到控制台。

举一反三

根据本实例，读者可以实现以下功能。

使用startElement()方法获取属性的名称和值。

获取当前元素的所有属性集合和属性相关信息。

实例419 使用VO解析XML元素

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\14\Ex14_419

实例说明

XML 元素使用DTD 或者XML Schema 限定以后，结构一般是不会变的，可以根据XML的结构定义一个VO，解析XML时，借助VO保存元素内容，使程序操作起来更方便，结构更简单。本实例讲解如何使用VO解析XML元素，如图14.3所示。

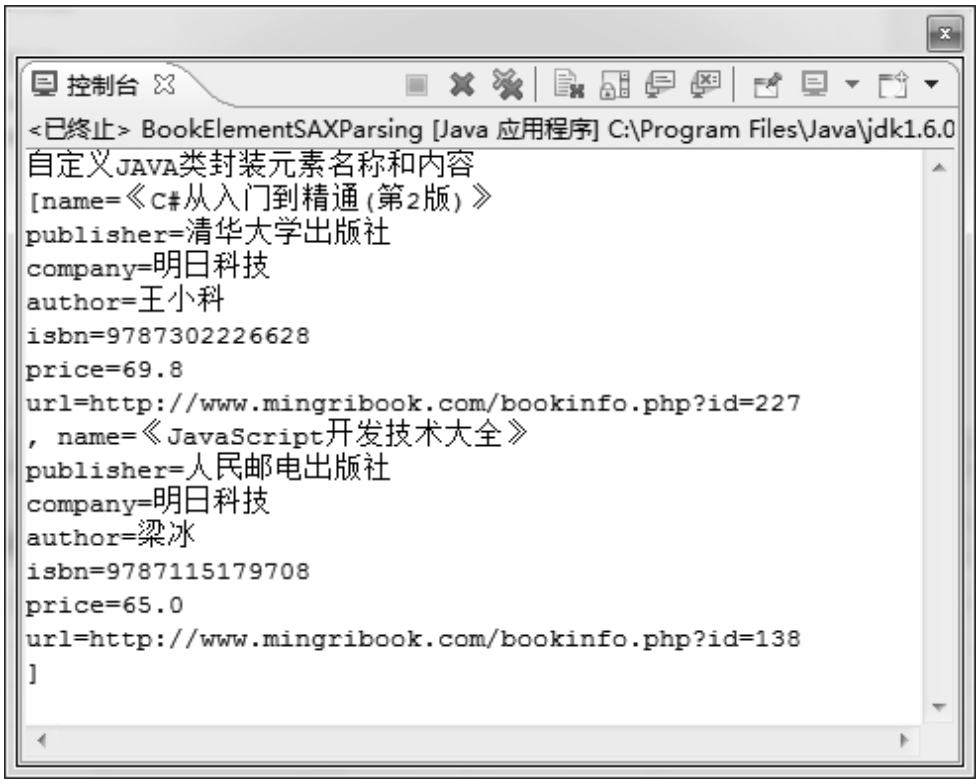


图14.3 使用VO 解析XML 元素

技术要点

SAX 解析 XML 时，建立 BookElementSAXParsing 继承 DefaultHandler 类，并在BookElementSAXParsing里重写 startElement()方法。在startElement()方法内，每次SAX检测到book元素都重新创建一个BookElement实例，示例如下：

```
//每次读取book元素的时候重新初始化bookElement
if (bookElement == null ||"book".equals(localName)) {
  bookElement = new BookElement();
}
```

在endElement()方法中，SAX检测到相应的元素就把元素内容置到BookElement的实例里，当检测到book元素时，就把BookElement实例保存在List里，这样，在XML里有几个book元素，在List里就会有几个BookElement实例。示例如下：

```
//检测当前元素，把元素内容保存在相应的实例属性里
if ("name".equals(localName)) {
  bookElement.setName(elementValue);
} else if ("publisher".equals(localName)) {
  bookElement.setPublisher(elementValue);
} else if ("company".equals(localName)) {
  bookElement.setCompany(elementValue);
} else if ("author".equals(localName)) {
  bookElement.setAuthor(elementValue);
} else if ("ISBN".equals(localName)) {
  bookElement.setIsbn(elementValue);
} else if ("price".equals(localName)) {
  bookElement.setPrice(Double.valueOf(elementValue));
} else if ("url".equals(localName)) {
  try{
```

```
        bookElement.setUrl(newURL(elementValue));
      } catch (MalformedURLException e) {
        e.printStackTrace();
      }
    } else if ("book".equals(localName)) {
      bookList.add(bookElement);
    }
```

实现过程

（1）创建一个XML文档用于SAX进行解析。

（2）根据XML文档结构创建BookElement类。

（3）创建BookElementSAXParsing类继承DefaultHandler类，然后实现startElement()方法，在方法中根据元素名称创建VO实体，代码如下：

```
    public void startElement(String uri, String localName,
String qName,Attributes attributes) throws SAXException {
      //每次读取book元素的时候重新初始化bookElement
      if (bookElement == null ||"book".equals(localName)) {
        bookElement = new BookElement();
      }
    }
```

（4）创建characters()方法，在方法中读取、解析XML元素内容。代码如下：

```
    public void characters(char[] ch, int start, int length)
throws SAXException {
      //解析元素内容
      elementValue = elementValue.valueOf(ch, start, length);
      elementValue = elementValue.replace("\t", "");
```

```
        elementValue = elementValue.replace("", "");
        elementValue = elementValue.replace("\n", "");
        elementValue = elementValue.replace("\r", "");
    }
```

（5）创建endElement()方法，把元素内容保存在BookElement实例中，再把BookElement实例保存在List里。

（6）创建parseReadFile()方法，把BookElementSAXParsing实例传入解析器，实现XML的解析。

（7）创建main()方法，在方法里使用BookElementSAXParsing解析books.xml文件，输出BookElement的List到控制台。

举一反三

根据本实例，读者可以实现以下功能。

使用VO解析XML元素属性。

使用VO解析XML元素属性值。

实例420 使用VO解析XML元素和属性（SAX）

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\14\Ex14_420

实例说明

在前面的实例中，解析XML元素时使用VO保存元素内容，这样让程序操作起来更方便。在本实例中，使用VO实现对XML元素和属性的解析，如图14.4所示。

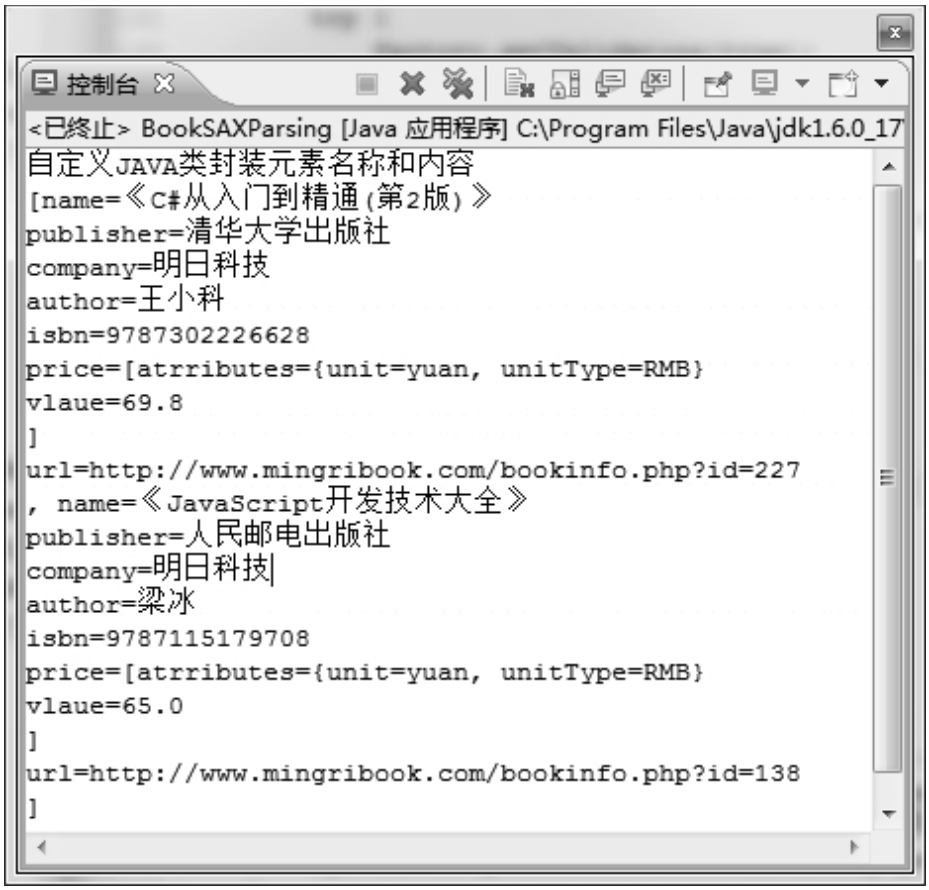


图14.4 使用VO 解析XML 元素和属性

技术要点

SAX解析XML时，创建BookSAXParsing类并继承DefaultHandler，在BookSAXParsing类中重写startElement()方法，并使用该方法读取price元素的属性，保存在bookPrice中。代码如下：

```
//读取元素、属性，把元素、属性名称保存在List中
if (bookPrice == null ||"price".equals(localName)) {
  bookPrice = new BookPrice();
  //读取属性名称和值
  Map<String, String> attributeMap = new HashMap<String,
String>();
  for (int i = 0; i < attributes.getLength(); i++) {
    //把属性名称和值保存在Map中
```

```
        attributeMap.put(attributes.getLocalName(i),attributes.getValue(i));
      }
      if (!attributeMap.isEmpty()) {
        bookPrice.setAttributeMap(attributeMap);
      }
    }
```

实现过程

（1）创建一个XML文档用于SAX进行解析。

（2）根据XML文档结构创建Book类和BookPrice类。

（3）创建 BookSAXParsing 类继承 DefaultHandler ，然后实现startElement()方法，在startElement()方法中根据元素名称创建VO实体，同时解析XML元素属性，把属性名称和属性值保存在VO中，代码如下：

```
    public void startElement(String uri, String localName, String qName, Attributes attributes) throws SAXException {
      if (book == null ||"book".equals(localName)) {
        book = new Book();
      }
      //读取元素、属性，把元素、属性名称保存在List中
      if (bookPrice == null ||"price".equals(localName)) {
        bookPrice = new BookPrice();
        //读取属性名称和值
        Map<String, String> attributeMap = new HashMap<String, String>();
        for (int i = 0; i < attributes.getLength(); i++) {
          //把属性名称和值保存在Map中
```

```
            attributeMap.put(attributes.getLocalName(i),attribu
        tes.getValue(i));
          }
          if (!attributeMap.isEmpty()) {
            bookPrice.setAttributeMap(attributeMap);
          }
        }
      }
```

（4）创建characters()方法，在方法中读取、解析XML元素内容。代码如下：

```
    public void characters(char[] ch, int start, int length)
throws SAXException {
        //解析元素内容
        elementValue = elementValue.valueOf(ch, start, length);
        elementValue = elementValue.replace("\t", "");
        elementValue = elementValue.replace("", "");
        elementValue = elementValue.replace("\n", "");
        elementValue = elementValue.replace("\r", "");
    }
```

（5）创建endElement()方法，把元素内容保存在Book实例中，再把Book实例保存在List里。代码如下：

```
    public void endElement(String uri, String localName,
String qName) throws SAXException {
        if ("name".equals(localName)) {
          book.setName(elementValue);
        } else if ("publisher".equals(localName)) {
          book.setPublisher(elementValue);
```

```
        } else if ("company".equals(localName)) {
          book.setCompany(elementValue);
        } else if ("author".equals(localName)) {
          book.setAuthor(elementValue);
        } else if ("ISBN".equals(localName)) {
          book.setIsbn(elementValue);
        } else if ("price".equals(localName)) {
          bookPrice.setValue(Double.valueOf(elementValue));
          book.setPrice(bookPrice);
        } else if ("url".equals(localName)) {
          try{
            book.setUrl(newURL(elementValue));
          } catch (MalformedURLException e) {
            e.printStackTrace();
          }
        } else if ("book".equals(localName)) {
          bookList.add(book);
        }
    }
```

（6）创建parseReadFile()方法，把BookSAXParsing实例传入解析器，实现XML的解析。

（7）创建main()方法，在方法里使用BookSAXParsing解析books.xml文件，输出Book的List到控制台。

举一反三

根据本实例，读者可以开发以下程序。

使用startElement()、endElement()和characters()方法获取元素、属性。

使用startDocument()方法解析XML。

实例421 使用SAX验证DTD

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\14\Ex14_421

实例说明

使用SAX也可以验证XML是否符合DTD的规范。首先XML要引用DTD文件，然后通过SAX对当前的XML进行解析，解析时需要设置factory.setValidating(true)。本实例要解析XML和DTD，为了看到验证效果，将XML元素中的price元素改成prices，解析XML时就会出现图14.5所示的错误信息。

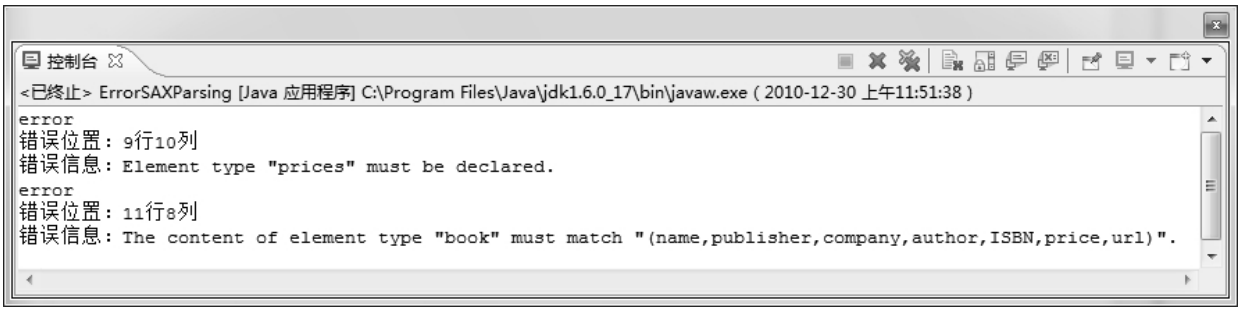


图14.5 验证XML 内容

技术要点

SAX解析XML时，建立ErrorSAXParsing继承DefaultHandler类，并在ErrorSAXParsing上重写warning()、error()和fatalError()方法。3个错误处理表示3个不同程度的错误，验证XML是否符合DTD规范需要覆盖error()方法。error()方法的语法如下：

public void error(SAXParseException exception) throws SAXException

参数说明

exception：解析XML时发生的异常处理。

实现过程

（1）创建一个DTD文档，用来定义XML文档的格式，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT book
(name,publisher,company,author,ISBN,price,url)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT publisher (#PCDATA)>
<!ELEMENT company (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT ISBN (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ELEMENT url (#PCDATA)>
<!ATTLIST price unit CDATA "RMB">
```

（2）根据XSD文档的结构定义XML文档，代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE book SYSTEM "books.dtd">
<book>
  <name>《C#从入门到精通(第2版)》</name>
  <publisher>清华大学出版社</publisher>
  <company>明日科技</company>
  <author>王小科</author>
  <ISBN>9787302226628</ISBN>
  <price>69.80</price>
  <url><![CDATA[http://www.mingribook.com/bookinfo.php?
id=227&sid=4]]></url>
</book>
```

（3）创建ErrorSAXParsing类继承DefaultHandler，实现error()、warning()和fatalError()方法。在3个方法里向控制台输出错误位置和错误信息。代码如下：

```
    public void error(SAXParseException exception) throws
SAXException {
       System.out.println("error");
       System.out.println("错误位置："+
  exception.getLineNumber() +"行"+
  exception.getColumnNumber() +"列");
       System.out.println("错误信息："+
  exception.getMessage());
    }
```

（4）创建parseReadFile()方法，把ErrorSAXParsing实例传入解析器，实现XML的解析。

（5）创建main()方法，在方法里使用ErrorSAXParsing解析books.xml文件，输出错误信息到控制台。

举一反三

根据本实例，读者可以实现以下功能。

使用SAXParseException类的getLineNumber()方法可以获取错误发生的行号。

使用SAXParseException类的getColumnNumber()方法获取错误发生的列号。

14.2 使用DOM解析XML

实例422 解析XML元素名称和内容（DOM）

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\14\Ex14_422

实例说明

XML元素的名称和内容都可以使用DOM解析出来。本实例演示如何使用DOM解析XML元素名称和内容，然后把它们输出到控制台，如图14.6所示。

图14.6 DOM 解析XML 元素名称和内容

技术要点

（1）使用Node类的getNodeName()方法可以获取元素节点的名称，语法如下：

```
public String getNodeName()
```

（2）使用Node类的getTextContent()方法可以获取元素内容，语法如下：

public String getTextContent() throws DOMException;

实现过程

（1）创建一个XML文档用于DOM解析。

（2）创建ElementValueDOMParserFile类，在类中创建parseReadFile()方法用于读取XML文件，同时返回一个Document。

（3）建立一个List变量，用于保存XML文件的元素和内容，代码如下：

```
private List<String> elementList = new ArrayList<String>();
```

（4）创建一个 getElementName()方法，在方法内部读取 XML 元素和内容，然后拼成一个字符串，保存到List的变量里。代码如下：

```
public void getElementName(Node parentNode) {
  if (parentNode.hasChildNodes()) {
    NodeList nodeList = parentNode.getChildNodes();
    for (int i = 0; i < nodeList.getLength(); i++) {
      Node node = nodeList.item(i);
      //判断是否有子节点
      if (node.hasChildNodes()) {
        getElementName(node);
        //拼成字符串保存在List中
        elementList.add(node.getNodeName() +"-"+
    node.getTextContent());
      }
    }
  }
}
```

（5）创建getElementList()方法，返回保存在List变量里的结果，代码如下：

```
public List<String> getElementList() {
  return this.elementList;
}
```

（6）创建main()方法，在内部调用getElementList()方法，最后把结果输出到控制台。

举一反三

根据本实例，读者可以实现以下功能。

通过DOM获取父节点的内容。

通过DOM解析XML文件。

实例423 解析XML元素属性和属性值（DOM）

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\14\Ex14_423

实例说明

DOM解析完XML以后，每个元素都会转换成一个Node对象，Node内不但存储着元素内容，同时还保存着元素的属性。使用getAttributes()方法可以获取当前元素的所有属性。在XML中，不同父节点的同名子节点的属性有可能相同，如本实例中的两个book:book 元素，它们都有book:price子节点，两个子节点的属性都是unit="yuan"和unitType="RMB"，使用DOM解析XML属性如图14.7所示。

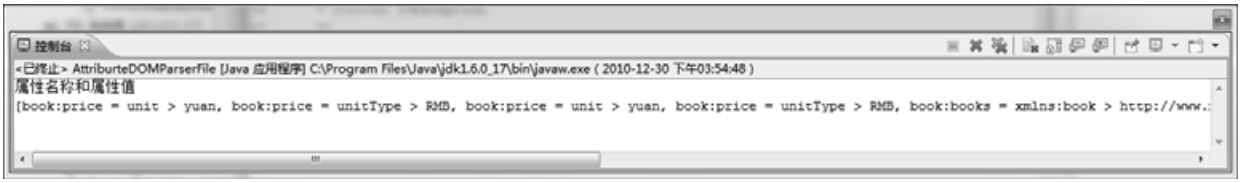


图14.7 DOM 解析XML 属性

技术要点

（1）使用Node类的getAttributes()方法可以获取属性列表，语法如下：

public NamedNodeMap getAttributes()

（2）使用NamedNodeMap类的item()方法可以获取指定的节点，语法如下：

public Node item(int index)

参数说明

index：表示当前节点的索引。

实现过程

（1）创建一个XML文档用于DOM解析。

（2）创建AttriburteDOMParserFile类，在类中创建parseReadFile()方法，用于读取XML文件，同时返回一个Document。

（3）建立一个List变量，用于保存XML文件的元素和内容，代码见实例422。

（4）创建一个 getElementName()方法，在方法内部读取 XML 文件元素、属性和属性值，然后拼成一个字符串，保存到List的变量里。代码如下：

```
public void getElementName(Node parentNode) {
  if (parentNode.hasChildNodes()) {
    NodeList nodeList = parentNode.getChildNodes();
    for (int i = 0; i < nodeList.getLength(); i++) {
      Node node = nodeList.item(i);
      if (node.hasChildNodes()) {
        getElementName(node);
        //获取属性列表
```

```
            NamedNodeMap namedNodeMap = node.getAttributes();
            for (int j = 0; j < namedNodeMap.getLength();
        j++) {
              Node node2 = namedNodeMap.item(j);
              //获取属性值
              elementList.add(node.getNodeName() +"="+
          node2.getNodeName() +">"+ node2.get NodeValue());
            }
          }
        }
      }
    }
```

（5）创建getElementList()方法，返回保存在List变量里的结果，代码如下：

```
    public List<String> getElementList() {
      return this.elementList;
    }
```

（6）创建 main()方法，在内部调用 AttriburteDOMParserFile 的 parseReadFile()、getElementName()和 getElementList()方法，然后把结果输出到控制台。

举一反三

根据本实例，读者可以实现以下功能。

使用Node的getNodeValue()方法获取属性的值。

使用getTextContent()方法获取属性值。

实例424 使用VO解析XML元素和属性（DOM）

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\14\Ex14_424

实例说明

本实例需要使用VO解析XML元素和属性，我们可以根据XML元素和属性的特点建立一个VO类，使每一个元素都对应VO的一个属性，并把它们输出到控制台，效果如图14.8所示。

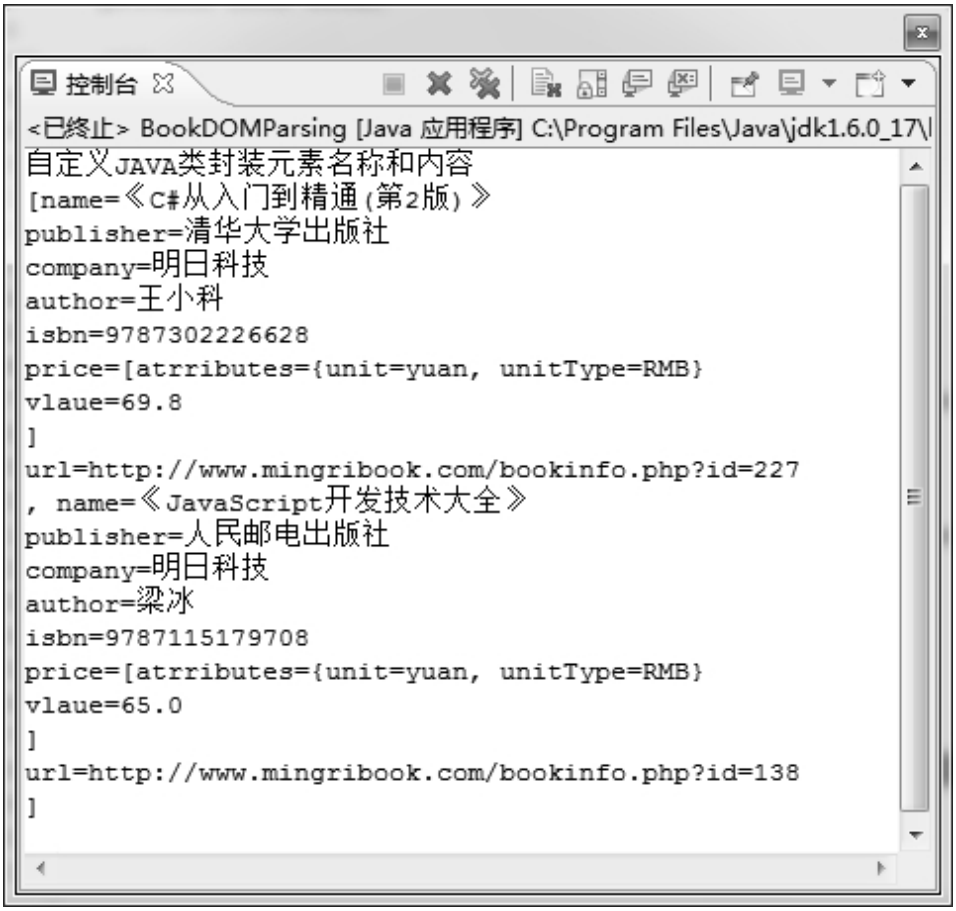


图14.8 使用VO 解析XML 元素和属性

技术要点

获取XML元素的属性比较繁琐，特别是元素有多个属性时。先根据getAttributes()方法获取属性列表，然后把属性内容保存在Map中，等所有属性解析完成后再把内容保存在VO中，代码如下：

```
//获取价格
BookPrice bookPrice = new BookPrice();
```

```
    Map<String, String> attributeMap = new HashMap<String,
String>();
    NamedNodeMap namedNodeMap =
price.item(i).getAttributes();
    for (int j = 0; j < namedNodeMap.getLength(); j++) {
      //获取价格属性
      Node node = namedNodeMap.item(j);
      attributeMap.put(node.getNodeName(),
  node.getNodeValue());
    }
    bookPrice.setAttributeMap(attributeMap);
    bookPrice.setValue(new
Double(price.item(i).getTextContent()));
    book.setPrice(bookPrice);
```

实现过程

（1）创建一个XML文档用于DOM解析。

（2）分别创建Book类和BookPrice类用于保存XML元素和属性。

（3）创建BookDOMParsing类，在类中创建parseReadFile()方法，用于读取XML文件，代码如下：

```
    public Document parseReadFile(String path) {
      //创建DocumentBuilderFactory对象
      DocumentBuilderFactory documentBuilderFactory =
  DocumentBuilderFactory.newInstance();
      DocumentBuilder dombuilder = null;
      try{
        //创建DocumentBuilder对象
```

```
        dombuilder =
    documentBuilderFactory.newDocumentBuilder();
      } catch (ParserConfigurationException e) {
        e.printStackTrace();
      }
      File file = new File(path);
      try{
        returndombuilder.parse(file);
      } catch (SAXException e) {
        e.printStackTrace();
      } catch (IOException e) {
        e.printStackTrace();
      }
      return null;
    }
```

（4）创建getBook()方法，把读取的XML文件元素和内容保存在Book中。对于book:price元素要特殊处理，先把book:price元素属性取出来保存在BookPrice类的attributeMap中，然后把值也保存在BookPrice中，再把BookPrice对象存放在Book中，最后把遍历的每个Book保存在一个List变量中，代码如下：

```
    public List<Book> getBook(Element element) {
      NodeList list =
  element.getElementsByTagName("book:book");
      for (int i = 0; i < list.getLength(); i++) {
        book = new Book();
        NodeList name =
    element.getElementsByTagName("book:name");
```

```
    NodeList publisher =
element.getElementsByTagName("book:publisher");
    NodeList company =
element.getElementsByTagName("book:company");
    NodeList author =
element.getElementsByTagName("book:author");
    NodeList ISBN =
element.getElementsByTagName("book:ISBN");
    NodeList price =
element.getElementsByTagName("book:price");
    NodeList url =
element.getElementsByTagName("book:url");
    book.setName(name.item(i).getTextContent());
    book.setPublisher(publisher.item(i).getTextContent())
;
    book.setCompany(company.item(i).getTextContent());
    book.setAuthor(author.item(i).getTextContent());
    book.setIsbn(ISBN.item(i).getTextContent());
    //获取价格
    BookPrice bookPrice = new BookPrice();
    Map<String, String> attributeMap = new
HashMap<String, String>();
    NamedNodeMap namedNodeMap =
price.item(i).getAttributes();
    for (int j = 0; j < namedNodeMap.getLength(); j++) {
      //获取价格属性
      Node node = namedNodeMap.item(j);
```

```
            attributeMap.put(node.getNodeName(),
        node.getNodeValue());
            }
            bookPrice.setAttributeMap(attributeMap);
            bookPrice.setValue(new
      Double(price.item(i).getTextContent()));
            book.setPrice(bookPrice);
            //获取URL
            try {
              book.setUrl(new URL(url.item(i).getTextContent()));
            } catch (MalformedURLException e) {
              e.printStackTrace();
            } catch (DOMException e) {
              e.printStackTrace();
            }
            bookList.add(book);
          }
          return bookList;
      }
```

（5）返回一个List<Book>，在main()方法中把刚才保存的结果输出到控制台，代码如下：

```
      public static void main(String[] arg) {
          String pathname ="xmldemo/books.xml";
          BookDOMParsing elementSAXParsing = new
    BookDOMParsing();
          Document document = null;
          document = elementSAXParsing.parseReadFile(pathname);
```

```
        List<Book> bookElements =
    elementSAXParsing.getBook(document.getDocumentElement());
        System.out.println("自定义JAVA类封装元素名称和内容");
        System.out.println(bookElements);
    }
```

举一反三

根据本实例，读者可以实现以下功能。

使用startElement()、endElement()和characters()方法获取元素、属性。

使用startDocument()方法解析XML。

14.3 使用DOM操作XML

实例425 创建基本的XML文件

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\14\Ex14_425

实例说明

本实例通过DOM创建了一个基本的XML文件，在XML中含有元素、元素的级别和元素的内容。通过浏览器查看XML文档的结构如图14.9所示。

图14.9 通过DOM 创建的基本的XML

技术要点

通过DocumentBuilderFactory和DocumentBuilder类可以创建一个新的Document，再使用Document 的 createElement()方法创建相应的元素 book:books、book:book、book:name、book:publisher、book:company、book:author、book:ISBN、book:price和book:url，然后根据XML元素的对应关系，使用appendChild()方法添加XML节点，最后返回document。代码如下：

```
    DocumentBuilderFactory documentBuilderFactory =
DocumentBuilderFactory.newInstance();
    DocumentBuilder dombuilder = null;
    try {
      dombuilder =
  documentBuilderFactory.newDocumentBuilder();
    } catch (ParserConfigurationException e) {
      e.printStackTrace();
    }
    Document document = dombuilder.newDocument();
    Element books = document.createElement("book:books");
    books.setAttribute("xmlns:book",
"http://www.mingrisoft.com");
    Element book = document.createElement("book:book");
    Element name = document.createElement("book:name");
    ….
    name.setTextContent("《C#从入门到精通(第2版)》");
    ….
    book.appendChild(name);
    ….
    document.appendChild(books);
```

实现过程

（1）创建一个JAVA文件ElementDOMBulid。

（2）在 ElementDOMBulid 中创建 bulid()方法，在方法内部构建一个符合 DOM 规范的Document对象。先使用createElement()方法生成元素，再为生成的元素设置相应的内容或者属性，如name.setTextContent("《C#从入门到精通(第2版)》");，然后根据元素之间的父子关系添加元素节点。代码如下：

```
public Document bulid() {
  DocumentBuilderFactory documentBuilderFactory =
DocumentBuilderFactory.newInstance();
  DocumentBuilder dombuilder = null;
  try{
    dombuilder=documentBuilderFactory.newDocumentBuilder(
);
  } catch (ParserConfigurationException e) {
    e.printStackTrace();
  }
  Document document = dombuilder.newDocument();
  Element books = document.createElement("book:books");
  books.setAttribute("xmlns:book",
"http://www.mingrisoft.com");
  Element book = document.createElement("book:book");
  Element name = document.createElement("book:name");
  Element publisher =
document.createElement("book:publisher");
  Element company =
document.createElement("book:company");
```

```
        Element author = document.createElement("book:author");
        Element isbn = document.createElement("book:ISBN");
        Element price = document.createElement("book:price");
        Element url = document.createElement("book:url");
        name.setTextContent("《C#从入门到精通(第2版)》");
        publisher.setTextContent("清华大学出版社");
        company.setTextContent("明日科技");
        author.setTextContent("王小科");
        isbn.setTextContent("9787302226628");
        price.setTextContent("69.80");
        url.setTextContent("http://www.mingribook.com/bookinfo.
    php?id=227");
        book.appendChild(name);
        book.appendChild(publisher);
        book.appendChild(company);
        book.appendChild(author);
        book.appendChild(isbn);
        book.appendChild(price);
        book.appendChild(url);
        books.appendChild(book);
        document.appendChild(books);
        return document;
    }
```

（3）创建writeFile()方法，获取Node对象和XML文件的生成路径，生成XML文档。代码如下：

```
    public void writeFile(Node node, String url) {
```

```
    TransformerFactory transformerFactory =
  TransformerFactory .newInstance();
    DOMSource domSource = new DOMSource(node);
    StreamResult streamResult = new StreamResult(new
  File(url));
    try {
      Transformer transformer =
    transformerFactory.newTransformer();
      transformer.transform(domSource, streamResult);
    } catch (TransformerConfigurationException e) {
      e.printStackTrace();
    } catch (TransformerException e) {
      e.printStackTrace();
    }
  }
```

（4）创建main()方法，设置XML生成的路径，调用bulid()和writeFile()方法完成XML文档的创建。

举一反三

根据本实例，读者可以实现以下功能。

创建XML文档。

为XML文件添加内容。

实例426 使用VO创建XML文件

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\14\Ex14_426

实例说明

为了更方便地创建XML文件，可以根据要创建的XML文件的格式，建立符合XML结构特性的VO，让VO中的每个属性都对应XML的一个元素，同时为复杂的元素创建单独的VO。创建的XML文档的运行效果如图14.10所示。

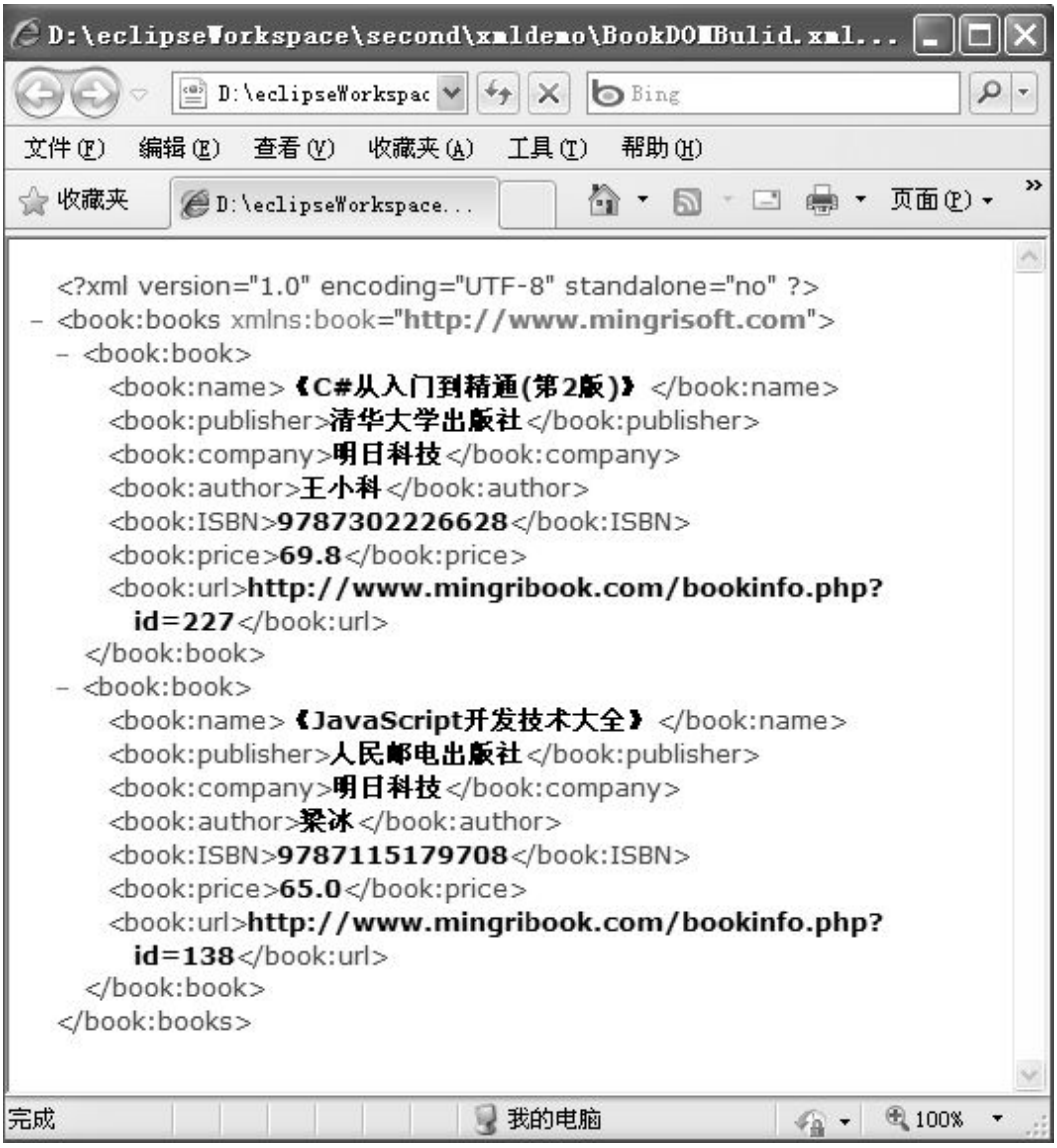


图14.10 使用VO 生成XML 文档

技术要点

使用TransformerFactory类的newInstance()方法可以创建TransformerFactory的一个实例，用于创建XML文件时产生一个转换器，语法如下：

public static TransformerFactory newInstance() throws TransformerFactoryConfigurationError

实现过程

（1）创建initData()方法，在方法内部生成一组数据，数据通过Book、BookPrice 等VO 存储在List中。代码如下：

```
protected List<Book> initData() {
  List<Book> bookList = new ArrayList<Book>();
  Book book;
  BookPrice bookPrice;
  Map<String, String> attributeMap;
  //第一本书
  book = new Book();
  bookPrice = new BookPrice();
  book.setName("《C#从入门到精通(第2版)》");
  book.setPublisher("清华大学出版社");
  book.setCompany("明日科技");
  book.setAuthor("王小科");
  book.setIsbn("9787302226628");
  attributeMap = new HashMap<String, String>();
  attributeMap.put("unit", "yuan");
  attributeMap.put("unitType", "RMB");
  bookPrice.setAttributeMap(attributeMap);
  bookPrice.setValue(Double.parseDouble("69.80"));
  book.setPrice(bookPrice);
  try{
    book.setUrl(newURL("http://www.mingribook.com/bookinf
o.php?id=227"));
```

```
    } catch (MalformedURLException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
    }
    bookList.add(book);
    //第二本书
    book = new Book();
    bookPrice = new BookPrice();
    book.setName("《JavaScript开发技术大全》");
    book.setPublisher("人民邮电出版社");
    book.setCompany("明日科技");
    book.setAuthor("梁冰");
    book.setIsbn("9787115179708");
    attributeMap = new HashMap<String, String>();
    attributeMap.put("unit", "yuan");
    attributeMap.put("unitType", "RMB");
    bookPrice.setAttributeMap(attributeMap);
    bookPrice.setValue(Double.parseDouble("65.00"));
    book.setPrice(bookPrice);
    try{
      book.setUrl(newURL("http://www.mingribook.com/bookinf
o.php?id=138"));
    } catch (MalformedURLException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
    }
    bookList.add(book);
```

```
    return bookList;
  }
```

（2）创建bulid()方法，在方法内调用initData()方法产生的数据，解析初始化数据的同时，可以转换成DOM的规范格式存储在Document中。代码如下：

```
  public Document bulid() {
    //创建DocumentBuilderFactory对象
    DocumentBuilderFactory documentBuilderFactory =
DocumentBuilderFactory.newInstance();
    DocumentBuilder dombuilder = null;
    Document document = null;
    try{
      //创建DocumentBuilder对象
      dombuilder =
  documentBuilderFactory.newDocumentBuilder();
    } catch (ParserConfigurationException e) {
      e.printStackTrace();
    }
    List<Book> bookList =initData();
    System.out.println(bookList);
    if (bookList != null) {
      document = dombuilder.newDocument();
      Element books = document.createElement("book:books");
      books.setAttribute("xmlns:book",
  "http://www.mingrisoft.com");
      for (Iterator iterator = bookList.iterator();
  iterator.hasNext();) {
```

```
    Book bookEntity = (Book) iterator.next();
    Element book = document.createElement("book:book");
    Element name = document.createElement("book:name");
    Element publisher =
document.createElement("book:publisher");
    Element company =
document.createElement("book:company");
    Element author =
document.createElement("book:author");
    Element isbn = document.createElement("book:ISBN");
    Element price =
document.createElement("book:price");
    Element url = document.createElement("book:url");
    //设置XML内容
    name.setTextContent(bookEntity.getName());
    publisher.setTextContent(bookEntity.getPublisher())
;
    company.setTextContent(bookEntity.getCompany());
    author.setTextContent(bookEntity.getAuthor());
    isbn.setTextContent(bookEntity.getIsbn());
    price.setTextContent(bookEntity.getPrice().getValue
().toString());
    url.setTextContent(bookEntity.getUrl().toString());
    //添加XML节点
    book.appendChild(name);
    book.appendChild(publisher);
    book.appendChild(company);
```

```
            book.appendChild(author);
            book.appendChild(isbn);
            book.appendChild(price);
            book.appendChild(url);
            books.appendChild(book);
        }
        document.appendChild(books);
    }
    return document;
}
```

（3）使用writeFile()方法接收bulid()方法产生的Document，然后将其转换成XML文件，根据参数传递的XML文件路径写在指定的位置上。

（4）在main()方法中设置XML生成的路径，调用bulid()和writeFile()方法完成XML文档的创建。

举一反三

根据本实例，读者可以实现以下功能。

对XML文档进行解析。

为XML文档添加元素。

实例427 使用DOM添加XML元素

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\14\Ex14_427

实例说明

在操作XML时，会经常需要为XML添加元素。在本实例中，原始XML文件只有两本图书内容，如图14.11所示。下面通过DOM为XML添加一个

book:book元素，产生新的XML文件，运行效果如图14.12所示。

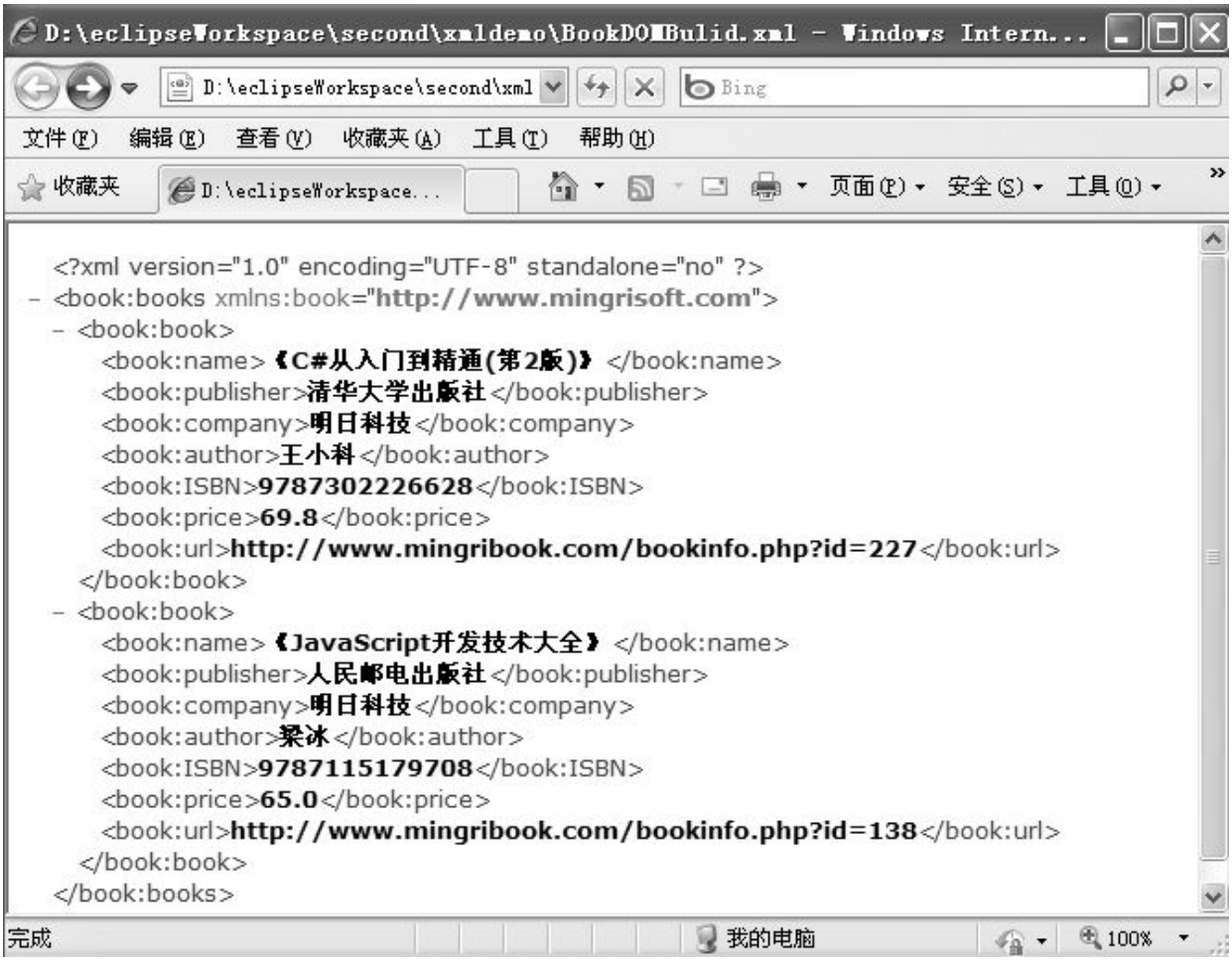


图14.11 原始XML 文件

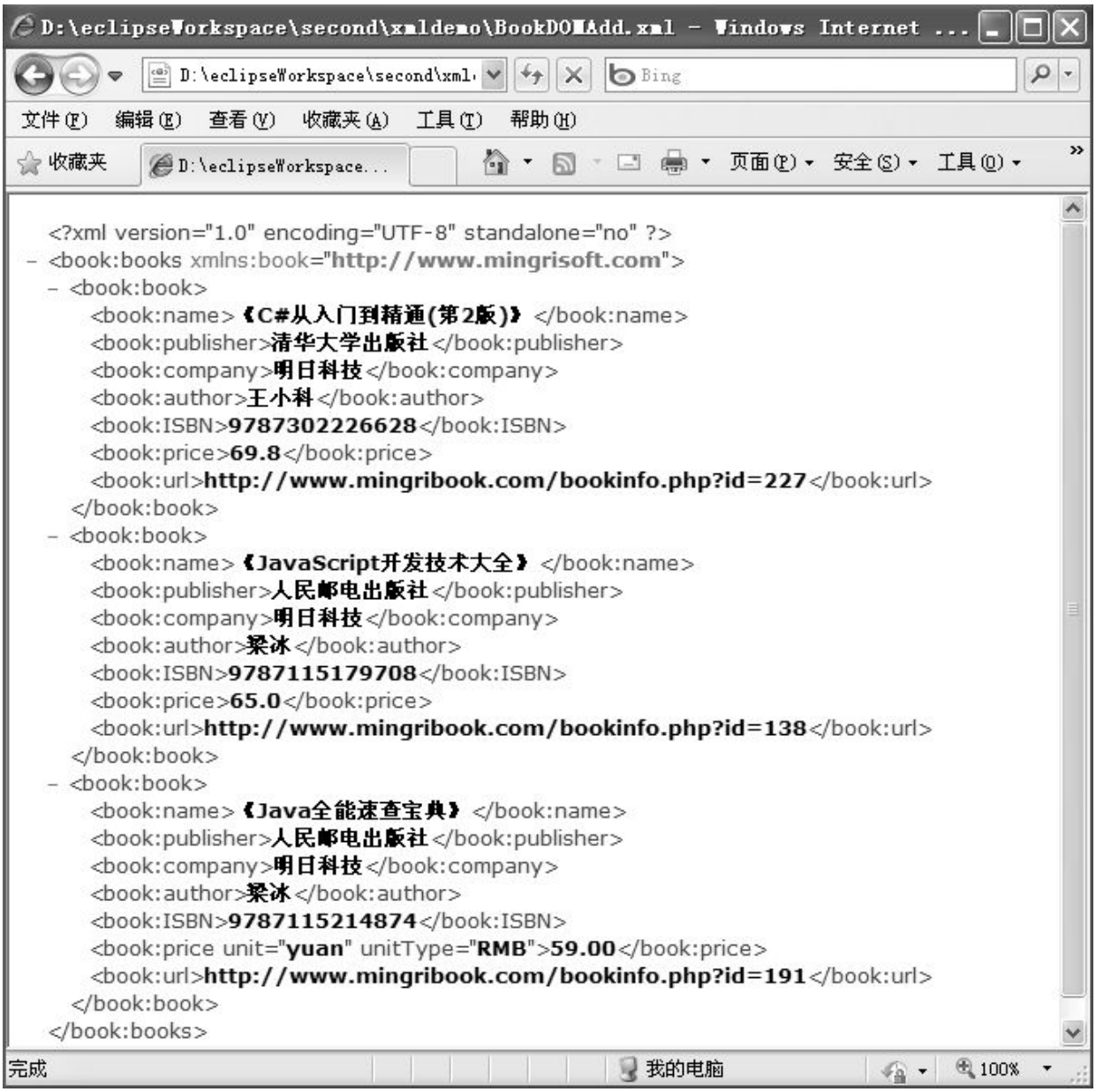


图14.12 添加元素以后的XML 文件

通过图14.11和图14.12可以很清楚地看到，最下面的XML元素是新增的book:book元素，该元素含有元素名称、元素内容和属性的相关信息。

技术要点

使用DOMSource类的构造方法，可以创建一个DOM的数据源，生成XML文件时，从该数据源获取数据，语法如下：

public DOMSource(Node n)

参数说明

n：表示DOM的XML节点。

实现过程

（1）创建BookDOMAdd类，然后创建parseReadFile()方法解析XML文件，文件解析完以后存储在DOM的document对象里，代码如下：

```
private void parseReadFile() {
  //创建DocumentBuilderFactory对象
  DocumentBuilderFactory documentBuilderFactory =
DocumentBuilderFactory.newInstance();
  DocumentBuilder dombuilder = null;
  try{
    //创建DocumentBuilder对象
    dombuilder =
documentBuilderFactory.newDocumentBuilder();
  } catch (ParserConfigurationException e) {
    e.printStackTrace();
  }
  File file = new File(path);
  try{
    //解析XML
    document = dombuilder.parse(file);
  } catch (SAXException e) {
    e.printStackTrace();
  } catch (IOException e) {
    e.printStackTrace();
  }
```

```
    }
```

（2）新建addBook()方法，创建新的book:book、book:name、book:publisher、book:company、book:author、book:isbn、book:price、book:url等元素，然后为book添加子元素name、publisher、company、author、isbn、price和url，再把book作为一个子元素添加到document里，代码如下：

```
    public void addBook() {
      parseReadFile();
      Element book = document.createElement("book:book");
      Element name = document.createElement("book:name");
      Element publisher =
  document.createElement("book:publisher");
      Element company =
  document.createElement("book:company");
      Element author = document.createElement("book:author");
      Element isbn = document.createElement("book:ISBN");
      Element price = document.createElement("book:price");
      Element url = document.createElement("book:url");
      name.setTextContent("《Java全能速查宝典》");
      publisher.setTextContent("人民邮电出版社");
      company.setTextContent("明日科技");
      author.setTextContent("梁冰");
      isbn.setTextContent("9787115213794");
      price.setTextContent("59.00");
      price.setAttribute("unit", "yuan");
      price.setAttribute("unitType", "RMB");
```

```
        url.setTextContent("http://www.mingribook.com/bookinfo.
    php?id=191");
        book.appendChild(name);
        book.appendChild(publisher);
        book.appendChild(company);
        book.appendChild(author);
        book.appendChild(isbn);
        book.appendChild(price);
        book.appendChild(url);
        document.getDocumentElement().appendChild(book);
    }
```

（3）创建writeFile()方法，把document中存储的内容保存在XML文件中。

举一反三

根据本实例，读者可以实现以下功能。

使用setTextContent()方法为指定的XML元素添加内容。

使用setAttribute()方法添加属性名称和属性值。

实例428 使用DOM修改XML元素

这是一个可以用来提高基础性能的实例

实例位置：光盘\mingrisoft\14\Ex14_428

实例说明

在操作XML时，如果XML元素的内容存储错误或者由于其他原因需要更改时，也可以使用DOM对XML进行操作。如有一个原始的XML，如图14.11所示，现需要将作者是王小科的图书名称由《C#从入门到精通（第2版）》改成《C#从入门到精通（第1版）》，这就需要对

book:name中的内容修改。修改完内容以后的运行效果如图14.13所示。

技术要点

使用 StreamResult 类的构造函数可以创建一个 StreamResult 实例，通过该实例能把一个XML文件转换成数据流的结果集，语法如下：

public StreamResult(File f)

参数说明

f：表示XML文件的实例。

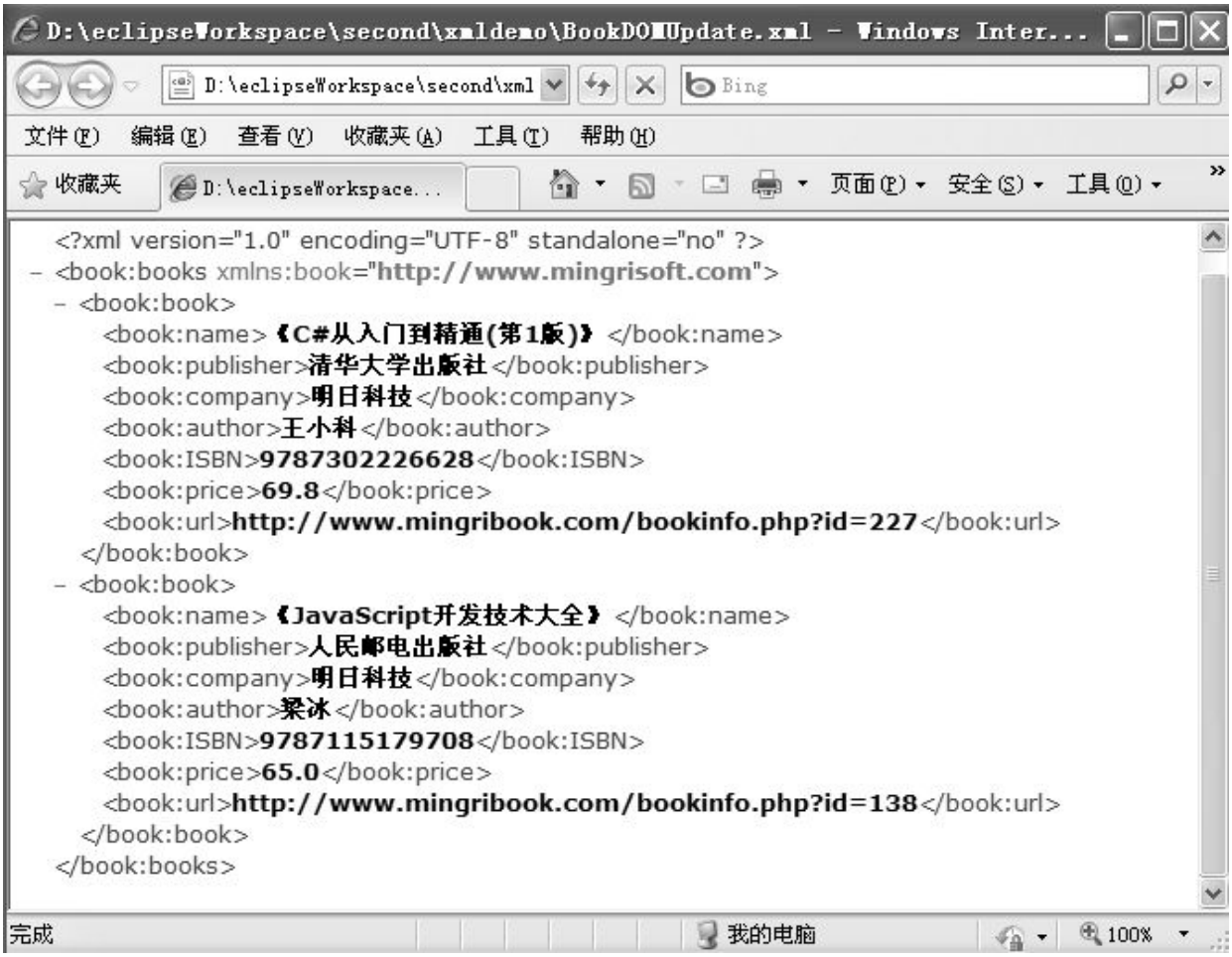


图14.13 修改book:name 以后的XML

实现过程

（1）创建BookDOMUpdate类，在类中创建parseReadFile()方法解析XML文件，文件解析完以后存储在DOM的document对象里，代码如下：

```
    private void parseReadFile() {
      DocumentBuilderFactory documentBuilderFactory =
  DocumentBuilderFactory.newInstance();
      DocumentBuilder dombuilder = null;
      try{
        dombuilder=documentBuilderFactory.newDocumentBuilder(
    );
      } catch (ParserConfigurationException e) {
        e.printStackTrace();
      }
      File file = new File(path);
      try{
        document=dombuilder.parse(file);
      } catch (SAXException e) {
        e.printStackTrace();
      } catch (IOException e) {
        e.printStackTrace();
      }
    }
```

（2）新建updateBook()方法，在方法中修改XML元素。使用DOM修改XML元素的内容前，先把XML文件读取出来保存在document中，根据条件查找document内部需要修改的XML元素，然后再修改元素内容。根据本实例的 XML 的结构，需要逐级锁定要找的元素，即先找到

book:book级别的元素，再找出book:book内的子元素名称，才能做修改操作。

book:book元素在document中可能存在多个，所以查找结果是一个NodeList列表，使用for语句对list列表循环搜索，获取的每个Node就是一个book:book元素。

因为book:book元素还有子节点，所以使用bookNode.getChildNodes获取book:book的子元素列表，再对子元素的列表list1循环搜索，获取的node1就是book:book的子元素。然后根据指定的条件node1.getNodeName().equals(nodeName)和node1.getTextContent().equals(text)进行判断，如果循环取得的元素名称和元素内容与nodeName、text中的一致，就设置变量flag=true。如果falg=true，就锁定了我们要找的book:book元素，可以针对当前的book:book进行修改操作。在修改之前，还要对list1 循环搜索，若node1.getNodeName().equals (replNodeName)判读成立，表示当前的node1就是我们想要修改的元素，然后使用setTextContent(repltext)设置修改以后的元素内容。具体代码如下：

```
/**
*根据book子节点指定的条件更新元素的内容
*@param nodeName 指定条件元素名称
*@param text 指定条件元素内容
*@param replNodeName 指定更新的节点
*@param repltext 设置更新的内容
*/
public void updateBook(String nodeName, String text,
String replNodeName,String repltext) {
  parseReadFile();
```

```
    NodeList list =
document.getElementsByTagName("book:book");
    for (int i = 0; i < list.getLength(); i++) {
      boolean flag = false;
      Node bookNode = list.item(i);
      NodeList list1 = bookNode.getChildNodes();
      //查询出需要修改的book节点
      for (int j = 0; j < list1.getLength(); j++) {
        Node node1 = list1.item(j);
        if (node1.getNodeName().equals(nodeName)&&
    node1.getTextContent().equals(text)) {
          flag = true;
        }
      }
      //查询到需要的book节点，设置指定的元素内容
      if (flag) {
        for (int j = 0; j < list1.getLength(); j++) {
          Node node1 = list1.item(j);
          //查询出需要修改的节点
          if (node1.getNodeName().equals(replNodeName)) {
            node1.setTextContent(repltext);
          }
        }
        flag = false;
      }
    }
  }
```

（3）创建writeFile()方法，把修改后的document内容保存在XML文件中。

举一反三

根据本实例，读者可以实现以下功能。

修改XML元素中的临时变量。

修改XML中的元素内容。

实例429 使用DOM删除XML元素

这是一个可以提高基础性能的实例

实例位置：光盘\mingrisoft\14\Ex14_429

实例说明

XML文件的内容是不断变化的，有时需要把XML中不需要的元素删除，使用DOM删除元素时，也同样要先把 XML文件读取到 DOM对象中，删除以后再把 DOM中的内容保存到XML文件。现有原始的XML文件（见图14.11），本实例要删除book:name是《C#从入门到精通(第2版)》的book:book元素，删除完以后XML的运行效果如图14.14所示。

技术要点

使用Node类的removeChild()方法可以删除不需要的节点，语法如下：

```
public Node removeChild(Node oldChild) throws
DOMException
```

参数说明

oldChild：表示希望删除的节点。

图14.14 删除元素以后的XML

实现过程

（1）创建BookDOMDelete类，然后创建parseReadFile()方法解析XML文件，文件解析完以后存储在DOM的document对象里，代码如下：

```
    private void parseReadFile() {
      //创建DocumentBuilderFactory对象
      DocumentBuilderFactory documentBuilderFactory =
  DocumentBuilderFactory.newInstance();
      DocumentBuilder dombuilder = null;
      try{
        //创建DocumentBuilder对象
        dombuilder =
    documentBuilderFactory.newDocumentBuilder();
      } catch (ParserConfigurationException e) {
        e.printStackTrace();
      }
```

```
        File file = new File(path);
        try{
          //解析XML
          document = dombuilder.parse(file);
        } catch (SAXException e) {
          e.printStackTrace();
        } catch (IOException e) {
          e.printStackTrace();
        }
    }
```

（2）在deleteBook()方法内部找到要删除的XML元素，并且存放在临时变量里，然后使用removeChild()方法删除临时变量里的元素值。deleteBook()方法有nodeName和text两个参数，根据这两个参数传递的值可以删除指定的 book:book 元素。在 deleteBook()方法内部使用getElementsByTagName()方法获取NodeList，循环list变量可以获取每个book:book元素，将其保存在node1里，再获取nod1.getChildNodes循环list1的内容，遍历出book:book下面的所有子元素。

在list1的循环里根据元素名称和节点锁定这个元素的父节点book:book元素，把book:book的对象赋值给node，然后等循环结束使用根元素的removeChild方法删除node。代码如下：

```
    //根据book子节点的条件删除book节点
    public void deleteBook(String nodeName, String text) {
      parseReadFile();
      Element books = document.getDocumentElement();
      NodeList list =
  document.getElementsByTagName("book:book");
```

```
    Node node = null;
    for (int i = 0; i < list.getLength(); i++) {
      NodeList list1 = list.item(i).getChildNodes();
      for (int j = 0; j < list1.getLength(); j++) {
        Node node1 = list1.item(j);
        //根据条件查询要删除的节点
        if (node1.getNodeName().equals(nodeName)&&
    node1.getTextContent().equals(text)) {
          node = list.item(i);
        }
      }
    }
    books.removeChild(node);
  }
```

（3）删除元素后，document中的内容发生了变化，需要使用writeFile()方法把document存储在XML中。

举一反三

根据本实例，读者可以实现以下功能。

使用removeChild()方法删除当前元素的下一级元素。

删除XML元素并保存。

第15章 网络技术

网络资源管理

TCP网络通信

TCP实用程序

15.1 网络资源管理

实例430 网络资源的单线程下载

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\15\Ex15_430

实例说明

本实例实现了网络资源的单线程下载，也就是在一个线程中完成网络资源的下载。运行程序，输入下载资源的网址，单击窗体上的“单击开始下载”按钮，将下载网络资源，并在下载完毕后进行提示，效果如图15.1和图15.2所示。

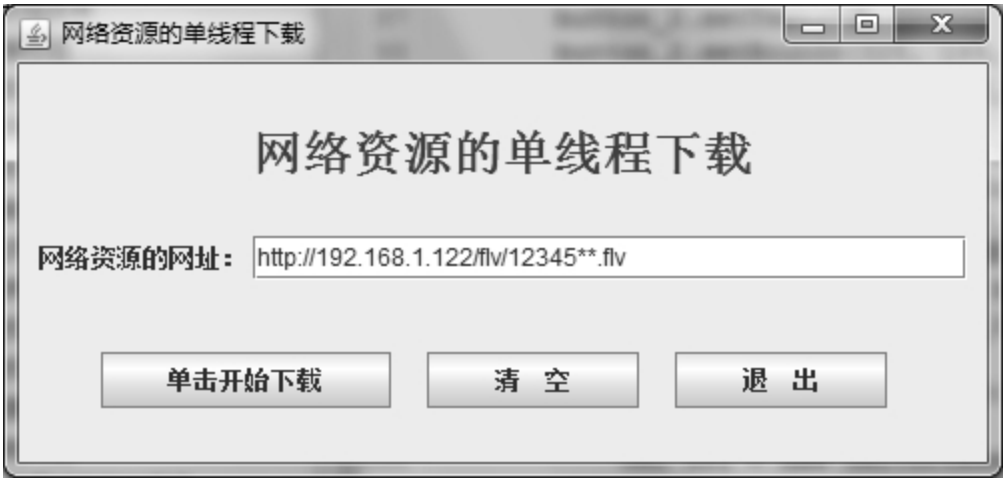


图15.1 网络资源的单线程下载界面

图15.2 提示下载完毕的消息框

技术要点

本实例主要是在主线程中，利用URLConnection对象的getInputStream()方法，获得网络资源的输入流对象，并将读取的信息通过输出流保存到用户主机上，从而实现了网络资源的单线程下载。

（1）通过URLConnection对象的getInputStream()方法，获得网络资源的输入流对象，用于读取网络资源的信息。

（2）使用 FileOutputStream 类创建输出流对象，然后使用该类的 write()方法，将从输入流获得的网络资源保存到磁盘上，实现网络资源的单线程下载，代码如下：

```
FileOutputStream out=new
FileOutputStream("C:/"+fileName);    //创建输出流对象
byte[]bytes=new byte[1024];                                    //
声明存放下载内容的字节数组
int len=
in.read(bytes);                                                //从输入流中
读取内容
while (len !=-1){
  out.write(bytes,0,len);
//将读取的内容写到输出流
  len= in.read(bytes);                                         //
继续从输入流中读取内容
}
```

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的SingleThreadDownloadFrame窗体类。

（3）在 SingleThreadDownloadFrame 窗体类的内容面板上添加相应控件，完成窗体界面的设计。

（4）在SingleThreadDownloadFrame窗体类中定义download()方法，用于根据参数urlAddr指定的地址，完成网络资源的单线程下载，该方法的代码如下：

```
public void download(String urlAddr)
{                              //从指定网址下载文件
  try {
    URL url=new URL(urlAddr);                              //
创建URL对象
    URLConnection
urlConn=url.openConnection();              //获得连接对象
    urlConn.connect();                                      /
/打开到url引用资源的通信链接
    InputStream in=urlConn.getInputStream()
;                 //获得输入流对象
    String
filePath=url.getFile();                                 //获得完
整路径
    int pos=
filePath.lastIndexOf("/");                               //获得
路径中最后一个斜杠的位置
    String fileName=
filePath.substring(pos+1);                   //截取文件名
    FileOutputStream out=new
FileOutputStream("C:/"+fileName);  //创建输出流对象
```

```
    byte[]bytes=new
byte[1024];                          //声明存放下载内容的
字节数组
    int len=
in.read(bytes);                                  //从输入流
中读取内容
    while (len !=-1){
      out.write(bytes,0,len);
  //将读取的内容写到输出流
      len=
  in.read(bytes);                                //继续从输
  入流中读取内容
    }
    out.close();                                        /
/关闭输出流
    in.close();
//关闭输入流
    JOptionPane.showMessageDialog(null, "下载完毕");
  } catch (Exception e) {
    e.printStackTrace();
  }
}
```

（5）在 SingleThreadDownloadFrame 窗体上，为“单击开始下载”按钮添加事件代码，用于调用download()方法，该按钮的事件代码如下：

```
button.addActionListener(new ActionListener() {
  public void actionPerformed(final ActionEvent e) {
```

```
        String address=
    tf_address.getText().trim();                    //获得网址
        download(address);                                          /
    /下载文件
      }
    });
```

举一反三

根据本实例，读者可以开发以下程序。

下载文本文件。

下载图片。

实例431 网络资源的多线程下载

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\15\Ex15_431

实例说明

本实例实现了网络资源的多线程下载，也就是主线程启动后，再通过单独的线程完成网络资源的下载。运行程序，输入下载资源的网址，单击窗体上的“下载”按钮，将完成网络资源下载，并在下载完毕后进行提示，效果如图15.3和图15.4所示。

图15.3 网络资源的多线程下载界面

图15.4 提示完成下载消息框

技术要点

本实例主要是通过创建线程类，然后在下载时指定线程数，并通过RandomAccessFile类的seek()方法定位下一个写入点，同时通过write()方法写入文件，从而完成网络资源的多线程下载。

（1）通过实现Runnable接口创建DownMultiThread类，在该类的成员声明区定义网络资源地址、需要写入的目标文件对象、写入的开始位置和结束位置4个成员变量，并通过构造方法为其赋值，代码如下：

```
public class DownMultiThread implements Runnable{
  private String sUrl=
"";                                    //网络资源地址
  private File desFile;                                    //
需要写入的目标文件对象
  private long
startPos;                                  //写入的开始位置
  private long endPos;                                     //
写入的结束位置
  /**
  *@param sUrl 网络资源地址
  *@param file 需要写入的目标文件对象
  *@param startPos 写入的开始位置
  *@param endPos 写入的结束位置
```

```
     */
     public DownMultiThread(String sUrl,File desFile,long
  startPos,long endPos) {
       this.sUrl = sUrl;
       this.desFile = desFile;
       this.startPos = startPos;
       this.endPos = endPos;
     }
     //省略了其他代码
   }
```

（2）使用RandomAccessFile类的seek()方法定位下一个写入点，并通过write()方法将网络资源写入文件，实现代码如下：

```
   RandomAccessFile out= new RandomAccessFile(desFile,
"rw");    //创建可读写的流对象
   out.seek(startPos);
//指定读写的开始标记
   out.write(buff,0,len);
  //写入磁盘文件
```

实现过程

（1）新建一个项目。

（2）在项目中创建一个实现Runnable接口的线程类DownMultiThread，用于实现网络资源的下载，该线程类中run()方法的关键代码如下：

```
   RandomAccessFile out=new RandomAccessFile(desFile,
"rw");              //创建可读写的流对象
   out.seek(startPos);
        //指定读写的开始标记
```

```
    InputStream in=
conn.getInputStream();                                    //获得
网络资源的输入流对象
    BufferedInputStream bin=new
BufferedInputStream(in);                          //创建输入缓冲流对
象
    byte[]buff=new
byte[2048];                                               //创建字节数
组
    int len=
-1;                                                       //声明
存放读取字节数的变量
    len=bin.read(buff);
        //读取到内容并添加到字节数组
    while (len!=-1){
      out.write(buff,0,len);
            //写入磁盘文件
      len=bin.read(buff);
          //读取到内容并添加到字节数组
    }
```

（3）在项目中创建一个继承JFrame类的MultiThreadDownFrame窗体类，在该类中定义一个download()方法，用于实现通过线程类DownMultiThread下载网络资源，该方法的代码如下：

```
    public void download(String url, String dest, int
threadNum)throws Exception {
      URL downURL=new
  URL(url);                                               //创建网络资
```

```
源的URL
    HttpURLConnection conn= (HttpURLConnection)
downURL.openConnection();  //打开网络连接
    long fileLength=
-1;                                                    //用于存储
文件长度的变量
    int stateFlagCode=
conn.getResponseCode();                                 //获得连
接状态标记代码
    if (stateFlagCode==200)
{                                                  //网络连接正常
      fileLength=
  conn.getContentLength();                                  //
  获得文件的长度
      conn.disconnect();
              //取消网络连接
    }
    if (fileLength > 0) {
      long byteCounts= fileLength /
  threadNum+1;                          //计算每个线程的字节
  数
      File file=new
  File(dest);                                           //创建
  目标文件的File对象
      int i = 0;
      while (i < threadNum) {
```

```
        long startPosition=byteCounts *
    i;                                  //定义开始位置
        long endPosition=byteCounts *
    (i+1);                              //定义结束位置
        if (i == threadNum - 1) {
          DownMultiThread fileThread=new
      DownMultiThread(url,
      file,startPosition,0);                                    //
      创建DownMultiThread线程的实例
          new
      Thread(fileThread).start();                               //
      启动线程对象
        } else {
          DownMultiThread fileThread=new
      DownMultiThread(url, file,startPosition,
      endPosition);                       //创建
      DownMultiThread线程的实例
          new
      Thread(fileThread).start();                               //
      启动线程对象
        }
        i++;
      }
      JOptionPane.showMessageDialog(null, "完成网络资源的下
载。");
    }
}
```

（4）在MultiThreadDownFrame窗体类的“下载”按钮事件中，添加调用download()方法的代码，完成网络资源的多线程下载，该按钮的事件代码如下：

```
String address= tf_address.getText();                    //获得网络资源地址
download(address, "c:\\01.flv",2);                       //调用download()方法，将下载的网络资源保存到磁盘
```

举一反三

根据本实例，读者可以实现以下功能。

多线程下载图片及文件。

多线程下载音频文件。

实例432 下载网络资源的断点继传

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\15\Ex15_432

实例说明

本实例实现了网络资源的断点继传功能。运行程序，输入下载资源的网址，然后按 Enter键，将显示网络资源的大小、上次读取到的字节位置以及未读取的字节数；输入下载的起始位置和结束位置，单击窗体上的“开始下载”按钮，开始下载网络资源，如果没有下载完成，可以接着上次的下载位置继续下载，直到全部资源下载完成后，弹出消息框并退出系统，效果如图15.5和图15.6所示。

图15.5 网络资源的多线程下载界面

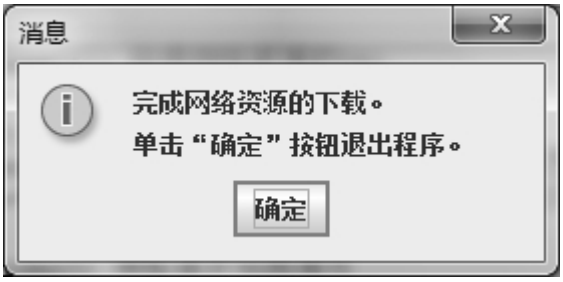


图15.6 显示完成下载消息框

技术要点

本实例主要是通过设置请求参数RANGE实现的，通过该参数，可以指定下载网络资源的字节区间，从而实现每次下载部分网络资源的功能。

例如，将该参数设置为RANGE: bytes = 0-1024，就表示将网络资源中0~1024 之间的内容下载到客户机；将该参数设置为RANGE: bytes =1024-，就表示将网络资源中从1024 到结束位置的内容全部下载到客户机。要设置RANGE属性可以通过如下代码实现：

```
connection.setRequestProperty("User-Agent",
"NetFox");                    //设置请求属性
String rangeProperty= "bytes="+ startPosition+ "-
";                            //定义请求范围属性
if (endPosition > 0) {
```

```
      rangeProperty+=
  endPosition;                                    //调整请求范围属
  性
    }
    connection.setRequestProperty("RANGE",
rangeProperty);              //设置请求范围属性
```

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的BreakPointSuperveneFrame窗体类。

（3）在BreakPointSuperveneFrame窗体类的内容面板上添加相应控件，完成窗体界面的设计。

（4）在 BreakPointSuperveneFrame 窗体类中定义 download() 方法，用于实现网络资源的断点继传，该方法的代码如下：

```
    public void download(long startPosition, long
endPosition) {
      try {
        URL url=new
    URL(urlAddress);                              //获得网络资
    源的URL
        HttpURLConnection connection= (HttpURLConnection)
    url.openConnection();                                  //获
    得连接对象
        connection.setRequestProperty("User-Agent",
    "NetFox");              //设置请求属性
        String rangeProperty= "bytes="+ startPosition+ "-
    ";                 //定义请求范围属性
```

```
    if (endPosition > 0) {
      rangeProperty+=
  endPosition;                              //调整请求范围
  属性
    }
    connection.setRequestProperty("RANGE",
rangeProperty);          //设置请求范围属性
    connection.connect();
      //连接网络资源
    InputStream in=
connection.getInputStream();                    //获得输
入流对象
    String
file=url.getFile();                                 //
获得文件对象
    String name=
file.substring(file.lastIndexOf('/')+1);             /
/获得文件名
    FileOutputStream out=new FileOutputStream("c:/"+name,
true);    //创建输出流对象，保存下载的资源
    byte[]buff=new
byte[2048];                             //创建字节数组
    int
len=0;                                          //定
义存储读取内容长度的变量
    len=
in.read(buff);                                      //
```

读取内容

```
    while (len !=-1) {
      out.write(buff,0,
  len);                                          //写入磁盘
      len=
  in.read(buff);                                          //读
  取内容
    }
    out.close();
      //关闭流
    in.close();
    //关闭流
    connection.disconnect();
        //断开连接
    if (readToPos > 0 && readToPos == totalLength) {
      JOptionPane.showMessageDialog(null, "完成网络资源的
  下载。\n单击“确定”按钮退出程序。");
      System.exit(0);
    }
  } catch (Exception e) {
    e.printStackTrace();
  }
}
```

（5）在 BreakPointSuperveneFrame 窗体上，为“网络资源的地址”文本框添加动作事件，用于获得文件大小和初始化窗体上的控件，该事件的代码如下：

```
tf_address.addActionListener(new ActionListener() {
```

```
public void actionPerformed(final ActionEvent e) {
  try {
    urlAddress = tf_address.getText().trim();
    URL url=new
URL(urlAddress);                              //获得网络资
源的URL
    HttpURLConnection connection= (HttpURLConnection)
url.openConnection();                              //获得
连接对象
    connection.connect();
    //连接网络资源
    totalLength=
connection.getContentLength();                  //获得网
络资源的长度
    connection.disconnect();
      //断开连接
    tf_totalLength.setText(String.valueOf(totalLength))
;            //显示总长度
    tf_readToPos.setText("0");
      //显示上次读取到的位置
    residuaryLength=
totalLength;                                  //未读内容为文
件总长度
    tf_residuaryLength.setText(String.valueOf(residuary
Length));      //显示未读内容
  } catch (MalformedURLException e1) {
    e1.printStackTrace();
```

```
      } catch (IOException e2) {
        e2.printStackTrace();
      }
    }
  });
```

（6）在 BreakPointSuperveneFrame 窗体上，为“开始下载”按钮添加动作事件，用于根据输入的起始位置和结束位置完成网络资源的断点继传，该事件的代码如下：

```
  button.addActionListener(new ActionListener() {
    public void actionPerformed(final ActionEvent e) {
      if (totalLength == 0) {
        JOptionPane.showMessageDialog(null,"没有网络资源。
    \n\n请输入正确的网址，然后回车。");
        return;
      }
      long
  startPos=0;                                    //起始位置
      long endPos=0;                                  //
  结束位置
      try {
        startPos=Long.parseLong(tf_startPos.getText().trim(
    ));    //起始位置
        endPos=Long.parseLong(tf_endPos.getText().trim());
          //结束位置
      } catch (Exception ex) {
        JOptionPane.showMessageDialog(null, "输入的起始位置
    或结束位置不正确。");
```

```
        return;
    }
    readToPos=
endPos;                                    //记录读取到的位
置
    residuaryLength= totalLength -
readToPos;                  //记录未读内容的大小
    tf_readToPos.setText(String.valueOf(readToPos));
        //显示读取到的位置
    tf_residuaryLength.setText(String.valueOf(residuaryLe
ngth));    //显示未读字节数
    tf_startPos.setText(String.valueOf(readToPos));
        //设置下一个读取点的开始位置
    tf_endPos.setText(String.valueOf(totalLength));
        //设置下一个读取点的结束位置
    tf_endPos.requestFocus();
  //使结束位置文本框获得焦点
    tf_endPos.selectAll();                          //选择结
束位置文本框中的全部内容，方便输入结束位置值
    download(startPos,
endPos);                              //调用方法进行下载
  }
});
```

举一反三

根据本实例，读者可以实现以下功能。

利用断点续传功能下载文件。

利用断点续传功能下载大型数据。

15.2 TCP网络通信

实例433 使用Socket通信

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\15\Ex15_433

实例说明

在使用套接字进行网络编程时，使用Socket传递汉字有时会出现乱码，本实例将讲解如何防止出现汉字乱码。运行程序，效果如图15.7和图15.8所示。

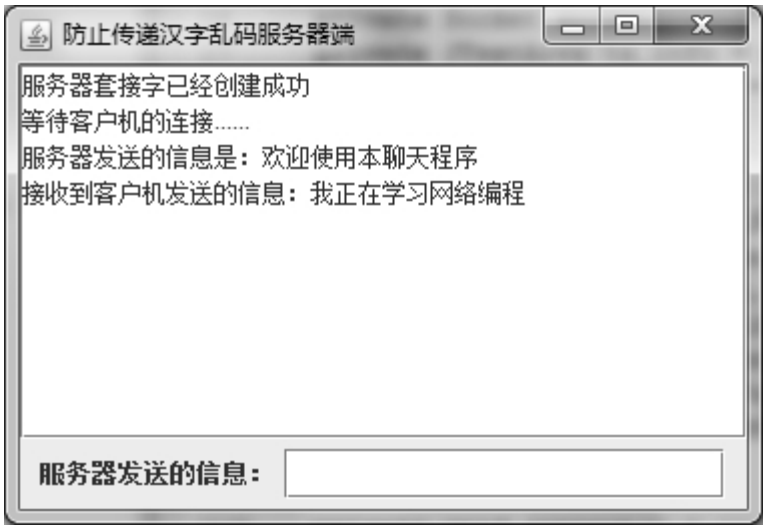


图15.7 防止传递汉字乱码服务器端

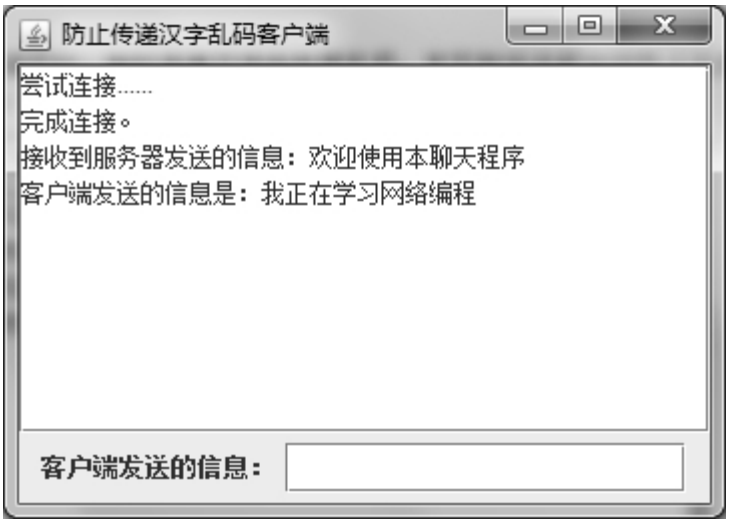


图15.8 防止传递汉字乱码客户端

技术要点

之所以会在传递中出现汉字乱码，是因为发送数据和接收数据所用的编码不同，因此，要解决传递汉字乱码问题，只需要在创建输入流和输出流对象时，使用相同的编码，在使用OutputStreamWriter类和InputStreamReader类创建对象时，在构造方法中指定相同的编码。

（1）OutputStreamWriter类是字节流通向字符流的桥梁，该类的构造方法定义如下：

public OutputStreamWriter(OutputStream out, String charsetName) throws UnsupportedEncodingException

参数说明

- out：字节输出流对象。
- charsetName：受支持的字符集名称。
- 使用该方法需要处理UnsupportedEncodingException 异常。

（2）InputStreamReader类是字节流通向字符流的桥梁，该类的构造方法定义如下：

public InputStreamReader(InputStream in, String charsetName) throws UnsupportedEncodingException

参数说明

- in：字节输入流对象。
- charsetName：受支持的字符集名称。
- 使用该方法需要处理UnsupportedEncodingException 异常。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承自JFrame类的服务器窗体类ServerSocketFrame和一个客户端窗体类ClientSocketFrame。

（3）在服务器窗体类ServerSocketFrame 的内容面板中央，添加一个 JScrollPane滚动面板控件，然后在滚动面板的视图上放置一个JTextArea 文本域控件，并命名为 ta_info，用于显示客户端与服务器的连接信息以及接收到的客户端发送的信息；再在服务器窗体的内容面板上部位置，添加一个JPanel面板控件，并在面板上添加一个JLabel标签和一个JTextField文本框，文本框的名称为tf_send，用于向客户端发送信息。

（4）在客户端窗体类ClientSocketFrame的内容面板中部位置，添加一个JScrollPane滚动面板控件，然后在滚动面板的视图上放置一个 JTextArea 文本域控件，并命名为 ta_info，用于显示连接信息以及客户端本身发送的信息；再在客户端窗体的内容面板上部位置，添加一个JPanel面板控件，并在面板上添加一个JLabel标签和一个JTextField文本框，文本框的名称为tf_send，用于向用服务器端发送信息。

（5）在服务器窗体类ServerSocketFrame中，定义getServer()方法用于创建服务器端套接字、监听客户端程序，以及创建向客户端发送信息的输出流对象和用于接收客户端发送信息的输入流对象，该方法的代码如下：

```
public void getserver() {
```

```
  try {
    server=new
ServerSocket(1978);                                    //实
例化Socket对象
    ta_info.append("服务器套接字已经创建成功
\n");                          //输出信息
    while (true)
{                                                      //如果套
接字是连接状态
      ta_info.append("等待客户机的连
  接......\n");                          //输出信息
      socket=
  server.accept();                                     /
  /实例化Socket对象
      reader=new BufferedReader(new
  InputStreamReader(socket.getInputStream(), "UTF-
  8"));                          //实例化BufferedReader对
  象
      out=new
  OutputStreamWriter(socket.getOutputStream(), "UTF-
  8");    //实例化OutputStreamWriter对象
      writer=new PrintWriter(out,
  true);                         //实例化PrintWriter对
  象
      getClientInfo();
            //调用getClientInfo()方法
    }
```

```
    } catch (Exception e) {
      e.printStackTrace();
              //输出异常信息
    }
  }
```

（6）在客户端窗体类ClientSocketFrame中，定义connect()方法用于创建套接字对象输入流和输出流对象以及在文本域中显示连接信息和接收服务器端发送的信息，该方法的代码如下：

```
  private void connect()
{                                                    //连接套接字
方法
    ta_info.append("尝试连
  接......\n");                                      //文本域中
  信息
    try
  {
  //捕捉异常
      socket=new
    Socket("192.168.1.122",1978);                    //
    实例化Socket对象
      while (true) {
        out=new
    OutputStreamWriter(socket.getOutputStream(), "UTF-
    8");    //实例化OutputStreamWriter对象
      writer=new PrintWriter(out,
    true);                              //实例化PrintWriter对
    象
```

```
            reader=new BufferedReader(new
        InputStreamReader(socket.getInputStream(), "UTF-
        8"));                              //实例化BufferedReader对
        象
            ta_info.append("完成连接。
        \n");                                  //文本域中提示信息
            getClientInfo();
                  //调用方法
          }
        } catch (Exception e) {
          e.printStackTrace();
                  //输出异常信息
        }
    }
```

举一反三

根据本实例，读者可以开发以下程序。

使用Socket传递文件。

使用Socket传递视频。

实例434 使用Socket传输图片

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\15\Ex15_434

实例说明

在使用套接字进行网络编程时，有时需要通过Socket传输图片，本实例讲解如何通过套接字传输图片。运行程序，在服务器端选择图片，单击“发送”按钮，就会将图片发送到客户端，效果如图15.9和

图15.10所示，也可以在客户端选择图片，单击“发送”按钮，向服务器端发送图片。

图15.9 服务器端选择图片并发送

图15.10 客户端显示接收到的图片

技术要点

本实例通过使用DataInputStream类的read()方法，将图片文件读取到字节数组，然后使用DataOutputStream类从DataOutput类继承的write()方法输出字节数组，从而实现了使用Socket传输图片的功能。

（1）使用DataInputStream类的read()方法，可以将文件中的信息读取到字节数组，该方法的定义如下：

public final int read(byte[] b) throws IOException

参数说明

- b：存储读取信息的字节数组。
- 返回值：读取到的字节总数，如果已经是文件尾，则返回-1。
- 异常处理：使用该方法需要处理IOException 异常。

（2）使用DataOutputStream类从DataOutput类继承的write()方法，可以输出字节数组，该方法的定义如下：

void write(byte[] b) throws IOException

参数说明

- b：需要写入输出流中的字节数组。
- 异常处理：使用该方法需要处理IOException 异常。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承自JFrame类的服务器窗体类ServerSocketFrame和一个客户端窗体类ClientSocketFrame，并完成服务器和客户端窗体界面的设计。

（3）在服务器窗体类ServerSocketFrame中，定义getServer()方法用于创建服务器端套接字、监听客户端程序，以及创建向客户端发送信息的输出流对象和用于接收客户端发送信息的输入流对象，该方法的关键代码如下：

```
server=new
ServerSocket(1978);                                    //实例化
Socket对象
while (true)
{                                                      //如果套接字是
连接状态
```

```
    socket=
  server.accept();                                    //实例化
  Socket对象
    out=new
  DataOutputStream(socket.getOutputStream());         //获得
  输出流对象
    in=new
  DataInputStream(socket.getInputStream());           //获
  得输入流对象
    getClientInfo();
   //调用getClientInfo()方法
   }
```

（4）在服务器窗体类 ServerSocketFrame 中，定义 getClientInfo()方法用于接收客户端发送的图片，该方法的关键代码如下：

```
    long lengths=
in.readLong();                                        //读取图片文
件的长度
    byte[]bt=new byte[(int)
lengths];                                   //创建字节数组
    for (int i = 0; i < bt.length; i++) {
      bt[i]=
  in.readByte();                                      //读取字
  节信息并存储到字节数组
    }
    receiveImg=new
ImageIcon(bt).getImage();                             //创建图像对象
```

```
    receiveImagePanel.repaint();
      //重新绘制图像
```

（5）在客户端窗体类 ClientSocketFrame 中，“发送”按钮的事件用于向服务器传输图片，其事件代码如下：

```
    DataInputStream
inStream=null;                                    //定义数据输入
流对象
    if (imgFile != null) {
      lengths=
  imgFile.length();                                  //获得选择
  图片的大小
      inStream=new DataInputStream(new
  FileInputStream(imgFile));    //创建输入流对象
    } else {
      JOptionPane.showMessageDialog(null, "还没有选择图片文
  件。");
      return;
    }
    out.writeLong(lengths);
    //将文件的大小写入输出流
    byte[]bt=new byte[(int)
lengths];                                  //创建字节数组
    int len =-1;
    while ((len= inStream.read(bt)) != -1)
{                              //将图片文件读取到字节数组
      out.write(bt);
    //将字节数组写入输出流
```

}

举一反三

根据本实例，读者可以实现以下功能。

使用Socket接收图片并解决中文乱码问题。

使用Socket发送图片。

实例435 使用Socket传输视频

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\15\Ex15_435

实例说明

在使用套接字进行网络编程时，有时需要通过Socket传输视频文件，本实例讲解如何通过套接字传输视频文件。运行程序，在服务器端或客户端选择视频文件，单击“发送”按钮，就会将视频文件发送到对方，并弹出“保存”对话框，然后由用户指定文件的保存位置和文件名，单击对话框中的“保存”按钮，完成视频文件的传输，如图15.11所示。

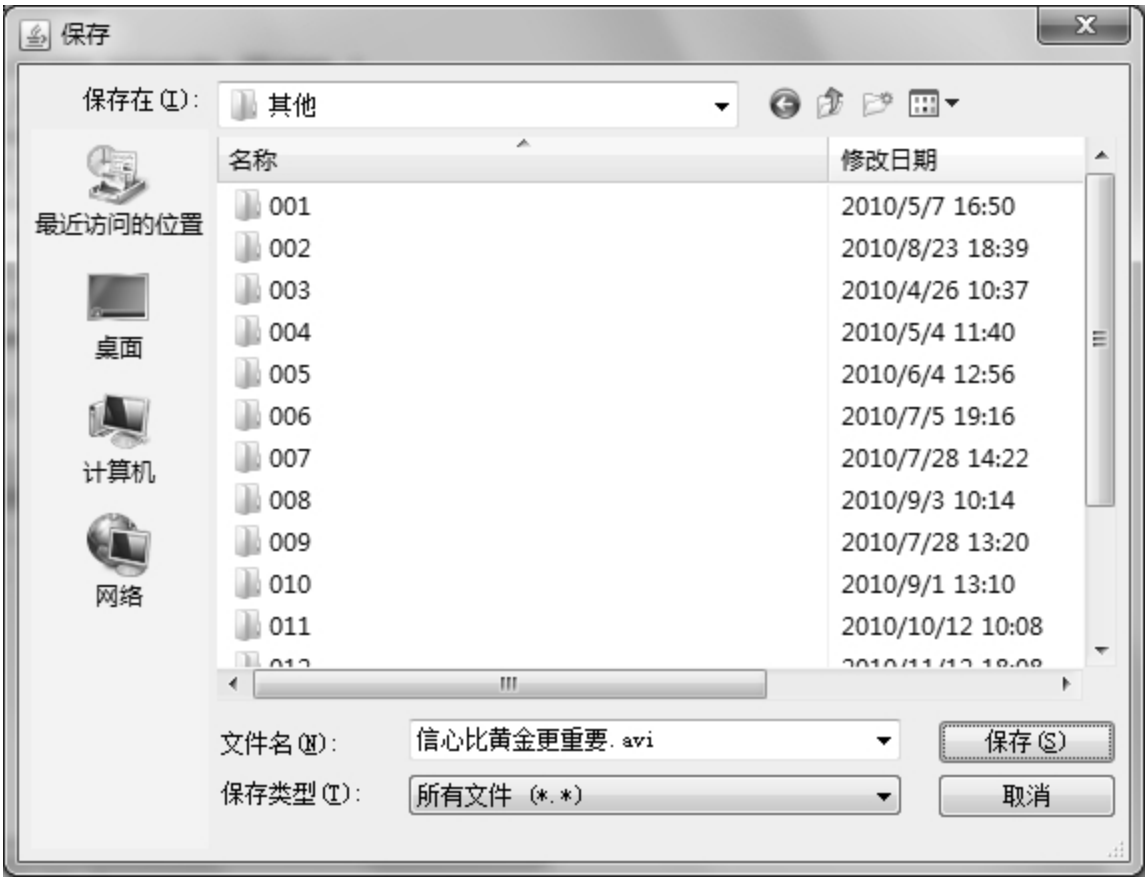


图15.11 用于保存接收到的视频文件的“保存”对话框

技术要点

本实例是在文件选择对话框中指定用于发送的文件类型，为了方便用户操作，在发送不同类型的文件时，为文件选择对话框指定相应的文件类型，下面是为文件选择对话框指定音频和视频的代码。

（1）为文件选择对话框指定音频类型的文件，代码如下：

```
JFileChooser fileChooser=new
JFileChooser();                                    //创建文件选择器
FileFilter filter = new FileNameExtensionFilter("音频文件
（WAV/MIDI/MP3/AU）", "WAV", "MID", "MP3", "AU");
//创建音频过滤器
fileChooser.setFileFilter(filter);
                    //设置过滤器
```

```
    int flag=
fileChooser.showOpenDialog(null);
    //显示打开对话框
```

（2）为文件选择对话框指定视频类型的文件，代码如下：

```
    JFileChooser fileChooser=new
JFileChooser();                                    //创建文件选择器
    FileFilter filter = new FileNameExtensionFilter("视频文件
（AVI/MPG/DAT/RM）", "AVI", "MPG", "DAT", "RM");
    //创建视频过滤器
    fileChooser.setFileFilter(filter);
                    //设置过滤器
    int flag=
fileChooser.showOpenDialog(null);
    //显示打开对话框
```

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承自JFrame类的服务器窗体类ServerSocketFrame和一个客户端窗体类ClientSocketFrame，并完成服务器和客户端窗体界面的设计。

（3）在服务器窗体类ServerSocketFrame中，定义getServer()方法用于创建服务器端套接字、监听客户端程序，以及创建向客户端发送信息的输出流对象和用于接收客户端发送信息的输入流对象，并显示相应的信息。

（4）在服务器窗体类 ServerSocketFrame 中，定义getClientInfo()方法用于接收客户端发送的视频文件，并弹出“保存”对话框，保存接收到的视频文件。

（5）在客户端窗体类ClientSocketFrame中，定义连接服务器的connect()方法，接收服务器信息的getServerInfo()方法，以及在“发送”按钮的事件中添加向服务器传输视频文件的代码，完成客户端程序的功能。

举一反三

根据本实例，读者可以实现以下功能。

使用UDP提高数据传输速度。

实现批量传输视频。

实例436 一个服务器与一个客户端通信

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\15\Ex15_436

实例说明

在使用套接字进行网络编程时，需要在服务器和客户端之间进行通信，本实例讲解如何实现一个服务器与一个客户端进行通信。运行程序，效果如图15.12和图15.13所示。

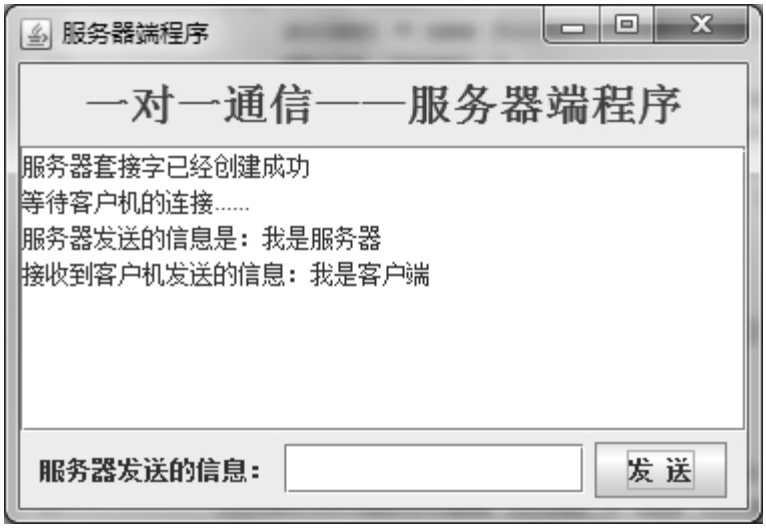


图15.12 服务器端发送和接收信息的效果

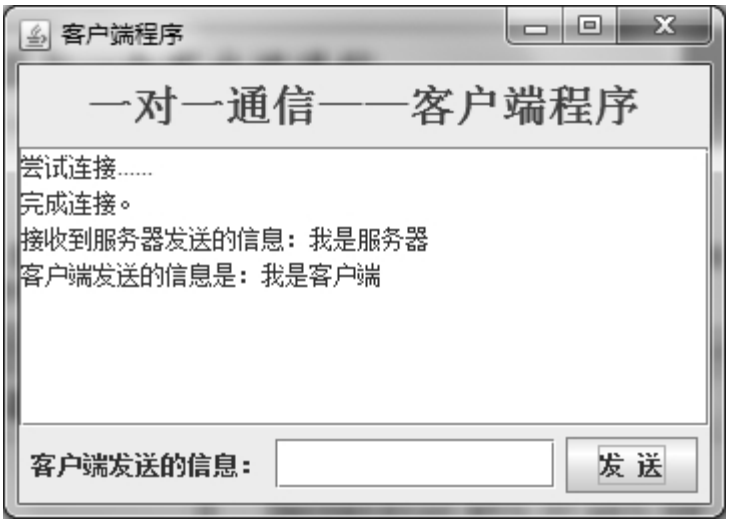


图15.13 客户端接收和发送信息的效果

技术要点

本实例在服务器端和客户端没有使用线程来处理接收到的信息，所以当有多个客户连接到服务器时，只有第一个客户能够与服务器进行通信，而其他客户必须等待，只有第一个客户退出，服务器才能与下一个客户进行通信，以此类推。

（1）服务器端通过getClientInfo()方法来接收客户端发送的信息，该方法的关键代码如下：

```
private void getClientInfo() {
  try{
    while(true){
      String line=
  reader.readLine();                                    //读
  取客户端发送的信息
      if (line != null)
        ta_info.append("接收到客户机发送的信息："+ line+
    "\n");       //显示客户端发送的信息
    }
```

```
    } catch (Exception e) {
      ta_info.append("客户端已退出。
\n");                                    //输出异常信息
    }
  }
```

（2）客户端通过getServerInfo()方法来接收服务器端发送的信息，该方法的关键代码如下：

```
  private void getServerInfo() {
    try{
      while(true){
        if (reader != null) {
          String line=
        reader.readLine();                                    //读
        取服务器端发送的信息
            if (line != null)
              ta_info.append("接收到服务器发送的信息："+
        line+ "\n");  //显示服务器端发送的信息
        }
      }
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
```

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承自 JFrame类的服务器窗体类ServerSocketFrame和一个客户端窗体类 ClientSocketFrame。

（3）在服务器窗体类ServerSocketFrame中定义getServer()方法，用于创建服务器端套接字、监听客户端程序、创建向客户端发送信息的输出流对象和用于接收客户端发送信息的输入流对象，该方法的关键代码如下：

```
    server=new
ServerSocket(1978);                                    //实例化
Socket对象
    ta_info.append("服务器套接字已经创建成功
\n");                    //输出信息
    while (true)
{                                                      //如果套接字是
连接状态
      ta_info.append("等待客户机的连
  接......\n");                        //输出信息
      socket=
  server.accept();                                     //实例化
  Socket对象
      reader=new BufferedReader(new
  InputStreamReader(socket.getInputStream()));
                  //实例化BufferedReader对象
      writer=new PrintWriter(socket.getOutputStream(),
  true);          //实例化PrintWriter对象
      getClientInfo();
    //调用getClientInfo()方法
    }
```

（4）在服务器窗体类ServerSocketFrame中，“发送”按钮用于向客户端发送信息，其事件代码如下：

```
    writer.println(tf_send.getText());
          //将文本框中信息写入流
    ta_info.append("服务器发送的信息是："+ tf_send.getText()+
"\n");    //将文本框中信息显示在文本域中
    tf_send.setText("");
      //将文本框清空
```

（5）在客户端窗体类ClientSocketFrame中定义connect()方法，用于创建与服务器连接的套接字对象、输入流对象和输出流对象，以及在文本域中显示与服务器的连接信息和接收到服务器端发送的信息，该方法的关键代码如下：

```
    socket=new
Socket("192.168.1.122",1978);                    //实例化
Socket对象
    while (true) {
      writer=new PrintWriter(socket.getOutputStream(),
  true);          //实例化PrintWriter对象
      reader=new BufferedReader(new
  InputStreamReader(socket.getInputStream()));
                  //实例化BufferedReader对象
      ta_info.append("完成连接。
  \n");                          //文本域中提示信息
      getServerInfo();
    }
```

（6）在客户端窗体类ServerSocketFrame中，“发送”按钮用于向服务器端发送信息，其事件代码如下：

```
    writer.println(tf_send.getText());
          //将文本框中信息写入流
```

```
ta_info.append("客户端发送的信息是："+ tf_send.getText()+"\n");     //将文本框中信息显示在文本域中
tf_send.setText("");
  //将文本框清空
```

举一反三

根据本实例，读者可以实现以下功能。

实现服务器与客户端一对一通信。

实现一个服务器与多个客户端通信。

实例437 一个服务器与多个客户端通信

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\15\Ex15_437

实例说明

在使用套接字进行网络编程时，需要在服务器和客户端之间进行通信，本实例讲解如何通过一个服务器与多个客户端进行通信。运行程序，服务器启动后，启动两个客户端程序，然后通过服务器向客户端发送信息，两个客户端都会收到服务器发送的信息，效果如图15.14和图15.15所示。

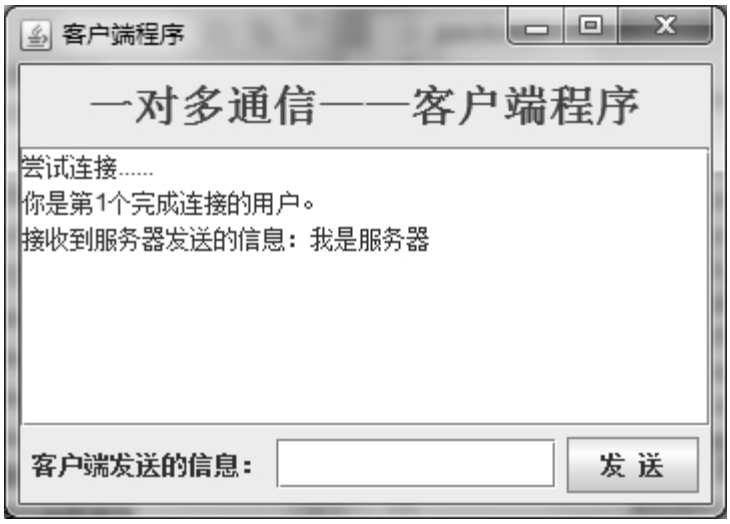


图15.14 第一个客户端接收到的服务器信息

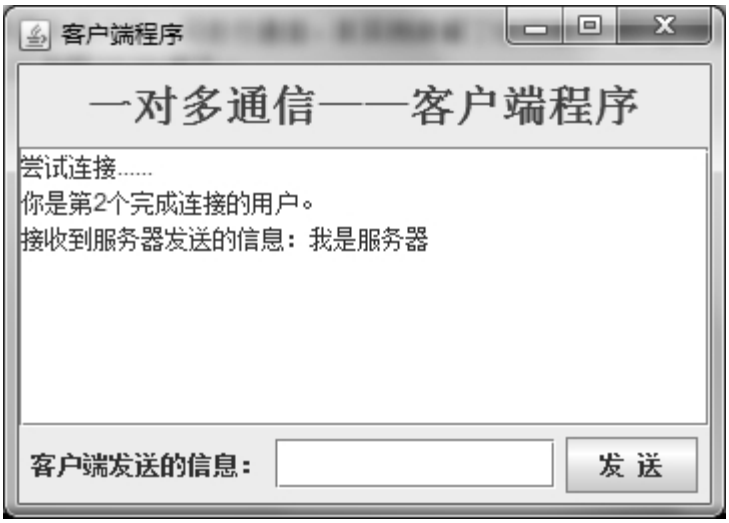


图15.15 第二个客户端接收到的服务器信息

技术要点

本实例在服务器端通过线程来处理不同客户发送的信息，所以当有多个客户连接到服务器时，服务器会为每个客户建立一个线程来处理接收到的信息，而不会产生阻塞，从而实现了一个服务器与多个客户端通信，并且在关闭客户端窗体时，会向服务器端发送退出客户的索引。（1）在服务器端创建线程类 ServerThread，用于接收客户端发送的信息以及处理客户端的退出信息，该线程类中run()方法的关键代码如下：

```
public void run() {
  try{
    if(socket!=null){
      reader=newBufferedReader(newInputStreamReader(socke
  t.getInputStream()));
      //实例化BufferedReader对象
      int index=
  -1;                                                    //
```

```
存储退出的客户端索引值
    try{
      while (true)
  {                                                    //如果
  套接字是连接状态
        String line=
  reader.readLine();                                    //
  读取客户端信息
        try{
          index=
  Integer.parseInt(line);
          //获得退出的客户端索引值
        } catch (Exception ex) {
          index =-1;
        }
        if (line != null) {
          ta_info.append("接收到客户机发送的信息："+
  line+ "\n");    //获得客户端信息
        }
      }
    } catch (Exception e) {
      if (index !=-1) {
        vector.set(index,null);
            //将退出的客户端套接字设置为null
        ta_info.append("第"+ (index+1)+ "个客户端已经退
  出。\n");       //输出异常信息
      }
```

```
        }
      }
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
```

（2）客户端窗体的关闭事件，用于向服务器发送退出客户的索引值，客户端窗体的关闭事件代码如下：

```
  addWindowListener(new WindowAdapter() {
    public void windowClosing(final WindowEvent e)
{                          //窗体关闭事件
      writer.println(String.valueOf(index));
              //向服务器端发送退出客户的索引值
    }
  });
```

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承自 JFrame类的服务器窗体类ServerSocketFrame和一个客户端窗体类 ClientSocketFrame。

（3）在服务器窗体类ServerSocketFrame中定义getServer()方法，用于创建服务器端套接字、监听客户端程序、创建向客户端发送信息的输出流对象，并向客户端发送连接用户的套接字索引以及创建并启动线程对象，用于接收客户端发送的信息，该方法的关键代码如下：

```
  server=new
ServerSocket(1978);                                    //实例
化Socket对象
```

```
    ta_info.append("服务器套接字已经创建成功
\n");                              //输出信息
    while (true)
{                                                          //如果套接
字是连接状态
      socket=
  server.accept();                                          //
  实例化Socket对象
      counts++;
      //计算连接客户总数
      ta_info.append("第"+ counts+ "个客户连接成功
  \n");                    //输出信息
      PrintWriter out=new
  PrintWriter(socket.getOutputStream(), true);          //创建
  输出流对象
      out.println(String.valueOf(counts -
  1));                                //向客户端发送套接字索引
      vector.add(socket);
          //存储客户端套接字对象
      new
  ServerThread(socket).start();
    //创建并启动线程对象
    }
```

（4）在客户端窗体类ClientSocketFrame中定义connect()方法，用于创建与服务器连接的套接字对象、输入流对象和输出流对象，从服务器读取客户端的索引以及在文本域中显示是第几个与服务器连接的用户和接收服务器端发送的信息，该方法的关键代码如下：

```
    socket=new
Socket("192.168.1.122",1978);                                    //实
例化Socket对象
    while (true) {
      writer=new PrintWriter(socket.getOutputStream(),
  true);                          //创建输出流对象
      reader=new BufferedReader(new
  InputStreamReader(socket.getInputStream()));  //实例化
  BufferedReader对象
      index=
  Integer.parseInt(reader.readLine());
      //获得客户登录服务器的索引值
      ta_info.append("你是第"+(index+1)+"个完成连接的用户。
  \n");          //文本域中提示信息
      getServerInfo();
          //调用接收服务器信息的方法
    }
```

举一反三

根据本实例，读者可以实现以下功能。

服务器与客户端之间通信。

15.3 TCP实用程序

实例438 聊天室服务器端

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\15\Ex15_438

实例说明

本实例实现了聊天室服务器端的功能。运行程序，服务器端等待客户端的连接，并显示客户端的连接信息，图 15.16 所示为有 3 个客户端连接到服务器，然后有一个客户端退出的效果。

图15.16 聊天室服务器端

技术要点

本实例使用 Hashtable 类来存储连接到服务器的用户名和套接字对象，并使用 String 类的startWith()方法判断客户端发送信息的类型，从而实现了向服务器端添加登录用户、发送退出信息、通过服务

器转发客户端发送的信息等功能，最终完成了聊天室服务器端程序的开发。

（1）Hashtable类实现了一个哈希表，用于存储键值映射关系，任何非null对象都可以用作键和值，在本实例中用到该类的构造方法定义如下：

public Hashtable()

参数说明

使用默认的初始容量11和加载因子0.75创建一个新的空哈希表。

（2）String类的startWith()方法，用于判断当前字符串是否以参数指定的字符串为前缀，该方法的定义如下：

public boolean startsWith(String prefix)

参数说明

● prefix：指定的前缀字符串。

● 返回值：如果该方法以指定的前缀开始，则返回true；否则返回false。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承自JFrame类的服务器窗体类ChatServerFrame，然后在窗体上添加一个滚动面板，并在滚动面板上添加一个文本域控件，完成窗体界面的设计。

（3）在服务器窗体类ChatServerFrame的成员声明区定义一个Hashtable对象，用于存储登录用户的用户名和套接字对象，代码如下：

```
private Hashtable<String, Socket> map = new Hashtable<String, Socket>(); //用于存储连接到服务器的用户和客户端套接字对象
```

（4）在服务器窗体类 ChatServerFrame 中，定义 createSocket()方法，用于创建服务器套接字对象、获得连接到服务器的客户端套接字对象以及启动线程对象对客户端发送的信息进行处理，该方法的代码如下：

```
public void createSocket() {
  try {
    server=new
ServerSocket(1982);                                    //创
建服务器套接字对象
    while (true) {
      ta_info.append("等待新客户连接......\n");
      socket=
  server.accept();                                      /
  /获得套接字对象
      ta_info.append("客户端连接成功。"+socket+"\n");
      new
  ServerThread(socket).start();
      //创建并启动线程对象
    }
  } catch (IOException e) {
    e.printStackTrace();
  }
}
```

（5）在服务器窗体类ChatServerFrame中，定义内部线程类 ServerThread，用于对客户端的连接信息以及发送的信息进行处理和转发，该类的定义如下：

```
class ServerThread extends Thread {
```

```
//省略了部分代码
public void run() {
  try {
    ObjectInputStream ins=new
ObjectInputStream(socket.getInputStream());      //获得
输入流对象
    while (true) {
      //省略了部分代码
      if (v != null && v.size() > 0) {
        for (int i = 0; i < v.size(); i++) {
          String info= (String)
      v.get(i);                          //读取信息
          String key ="";
          if (info.startsWith("用户：))
      {                          //添加登录用户到客户端列表
          //省略了部分代码
          } else if (info.startsWith("退出：")) {
          //省略了部分代码
          } else
      {                                    //转发接收
      的消息
            key= info.substring(info.indexOf("：发送
        给：")+5,info.indexOf("：的信息是："));    //获
        得接收方的key值，即接收方的用户名
          String sendUser= info.substring(0,
        info.indexOf("：发送给："));          //获得发
        送方的key值，即发送方的用户名
```

```
            Set<String> set=map.keySet();              //
        获得集合中所有键的Set视图
            Iterator<String>keyIt=
        set.iterator();        //获得所有键的迭代器
            while (keyIt.hasNext()) {
              String receiveKey=keyIt.next();         //
          获得表示接收信息的键
              if (key.equals(receiveKey) &&
          !sendUser.equals(receiveKey)) {
              //与接受用户相同，但不是发送用户
                Socket s=map.get(receiveKey);    //获得
          与该键对应的套接字对象
                PrintWriter out = new
          PrintWriter(s.getOutputStream(), true);
                //创建输出流对象
                out.println("MSG:"+ info);         //发
          送信息
                out.flush();                       //刷新
          输出缓冲区
              }
            }
          }
        }
      }
    } catch (IOException e) {
      ta_info.append(socket +"已经退出。\n");
```

```
        }
      }
    }
```

举一反三

根据本实例，读者可以实现以下功能。

使聊天室服务器端发送信息。

实现聊天功能。

实例439 聊天室客户端

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\15\Ex15_439

实例说明

本实例实现了聊天室客户端。运行程序，用户登录服务器后，可以从用户列表中选择单个用户进行聊天，也可以选择多个用户进行聊天，其中选择单个用户进行聊天的效果如图 15.17和图15.18所示。

技术要点

通过线程对接收到的信息进行处理分为3种情况：第1种是接收到的是登录用户；第2种是接收到的是退出提示；第3种是接收到的是消息。实现这3种功能的关键代码如下：

```
    String info=
in.readLine().trim();                                    //读取信息
    if (!info.startsWith("MSG:"))
{                                          //接收到的不是消息
      if (info.startsWith("退出："))
  {                                      //接收到的是退出消息
```

```
        model.removeElement(info.substring(3));
        //从用户列表中移除用户
      } else
  {                                                    //接收到的是
  登录用户
        boolean itemFlag= false;                       //标记是
    否为列表框添加列表项，为true不添加，为false添加
        for (int i=0; i<model.getSize(); i++)
    {                    //对用户列表进行遍历
          if (info.equals((String)model.getElementAt(i)))
      {         //如果用户列表中存在该用户名
            itemFlag= true;                                //设
        置为true，表示不添加到用户列表
            break;                                         //结束
        for循环
          }
        }
        if (!itemFlag) {
          model.addElement(info);                          //将
      登录用户添加到用户列表
        }
      }
    } else {                                               //
如果获得的是消息，则在文本域中显示接收到的消息
      DateFormat
  df=DateFormat.getDateInstance();                         //获得
  DateFormat实例
```

```
    String dateString=df.format(new
Date());                              //格式化为日期
    df=DateFormat.getTimeInstance(DateFormat.MEDIUM);
        //获得DateFormat实例
    String timeString=df.format(new
Date());                              //格式化为时间
    String sendUser= info.substring(4,info.indexOf("：发送
给："));          //获得发送信息的用户
    String receiveInfo= info.substring(info.indexOf("：的信
息是：")+6);      //获得接收到的信息
    ta_info.append(" "+sendUser +" "+dateString+"
"+timeString+"\n "+receiveInfo+"\n");
    //在文本域中显示信息
    ta_info.setSelectionStart(ta_info.getText().length()-1)
;                  //设置选择的起始位置
    ta_info.setSelectionEnd(ta_info.getText().length());
                  //设置选择的结束位置
    tf_send.requestFocus();
        //使发送信息文本框获得焦点
  }
```

图15.17 阳光**与开心**的聊天记录

图15.18 开心**与阳光**的聊天记录

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承自 JFrame 类的客户端窗体类ChatClientFrame，用于进行用户登录、发送聊天信息和显示聊天信息，在该类中完成窗体界面的设计。

（3）在客户端窗体类ChatClientFrame中定义createClientSocket()方法，用于创建套接字对象、输出流对象以及

启动线程对象对服务器转发的信息进行处理，该方法的代码如下：

```
public void createClientSocket() {
  try {
    Socket socket=new
Socket("192.168.1.122",1982);          //创建套接字对象
    out=new
ObjectOutputStream(socket.getOutputStream());      //创建
输出流对象
    new
ClientThread(socket).start();                          //创
建并启动线程对象
  } catch (UnknownHostException e) {
    e.printStackTrace();
  } catch (IOException e) {
    e.printStackTrace();
  }
}
```

（4）在客户端窗体类ChatClientFrame中定义内部线程类ClientThread，用于对服务器端转发的信息进行处理，并显示在相应的控件中，该类的关键代码已在关键技术中给出。

（5）在客户端窗体类ChatClientFrame中，为“登录”按钮添加实现用户登录功能的代码，“登录”按钮的事件代码如下：

```
if (loginFlag)
{                                          //已登录标记为true
  JOptionPane.showMessageDialog(null, "在同一窗口只能登录
一次。");
  return;
```

```
    }
    String userName=
tf_newUser.getText().trim();                          //获得
登录用户名
    Vector v=new
Vector();                                             //定义向
量，用于存储登录用户
    v.add("用
户："+userName);                                       //添
加登录用户
    try {
      out.writeObject(v);
          //将用户向量发送到服务器
      out.flush();
        //刷新输出缓冲区
    } catch (IOException ex) {
      ex.printStackTrace();
    }
    tf_newUser.setEnabled(false);
          //禁用用户文本框
    loginFlag=
true;                                                 //将已登
录标记设置为true
```

（6）在客户端窗体类ChatClientFrame中，定义发送聊天信息的send()方法，该方法的代码如下：

```
    private void send() {
      if (!loginFlag) {
```

```
      JOptionPane.showMessageDialog(null, "请先登录。");
      return;
    //如果用户没登录则返回
    }
    String sendUserName=
tf_newUser.getText().trim();                    //获得登录用
户名
    String info=
tf_send.getText();                                  //获得
输入的发送信息
    if (info.equals("")) {
      return;
    //如果没输入信息则返回，即不发送
    }
    Vector<String>v=new Vector<String>
();                          //创建向量对象，用于存储发送的
消息
    Object[]
receiveUserNames=user_list.getSelectedValues();
  //获得选择的用户数组
    if (receiveUserNames.length <= 0) {
      return;
    //如果没选择用户则返回
    }
    for (int i = 0; i < receiveUserNames.length; i++) {
      String msg= sendUserName+ "：发送给："+ (String)
  receiveUserNames[i]+ "：的信息是： "+
```

```
  info;                                    //定义发送的信息
      v.add(msg);
    //将信息添加到向量
    }
    try{
      out.writeObject(v);
        //将向量写入输出流，完成信息的发送
      out.flush();
        //刷新输出缓冲区
    } catch (IOException e) {
      e.printStackTrace();
    }
    DateFormat
df=DateFormat.getDateInstance();                    //获
得DateFormat实例
    String
dateString=df.format(newDate());                        /
/格式化为日期
    df=DateFormat.getTimeInstance(DateFormat.MEDIUM);
        //获得DateFormat实例
    String timeString=df.format(new
Date());                          //格式化为时间
    String sendUser=
tf_newUser.getText().trim();                        //获得
发送信息的用户
    String receiveInfo=
tf_send.getText().trim();                          //显示发送
```

```
的信息
    ta_info.append("  "+sendUser+
"  "+dateString+"  "+timeString+"\n  "+receiveInfo+"\n");
//在文本域中显示信息
    tf_send.setText(null);
          //清空文本框
    ta_info.setSelectionStart(ta_info.getText().length()-1)
;                    //设置选择的起始位置
    ta_info.setSelectionEnd(ta_info.getText().length());
                   //设置选择的结束位置
    tf_send.requestFocus();
        //使发送信息文本框获得焦点
  }
```

举一反三

根据本实例，读者可以实现以下功能。

使用文件记录聊天信息。

删除聊天信息。

第16章 邮件收发技术

简单邮件

复杂邮件

16.1 简单邮件

实例440 发送邮件

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\16\Ex16_440

实例说明

本实例实现了邮件的发送。运行程序，输入收件人地址、发件人地址、主题和正文，单击“发送”按钮，如果发送成功，将显示消息框进行提示，效果如图16.1和图16.2所示。

图16.1 发送邮件窗体

图16.2 提示邮件发送成功的消息框

技术要点

本实例的实现主要是通过Message类将接收方地址、发送方地址和要发送的信息封装起来，然后使用Transport类的send()方法完成信息的发送。

（1）使用Message类将接收方地址、发送方地址和要发送的信息封装起来，实现代码如下：

```
Message msg=new MimeMessage(session);                    //创建Message对象
InternetAddress[] toAddrs=InternetAddress.parse(toAddr,false);    //创建接收方的InternetAddress对象
msg.setRecipients(Message.RecipientType.TO, toAddrs);          //指定接收方
msg.setSentDate(new Date());                                   //指定发送日期
msg.setSubject(title);                                         //设置主题
msg.setFrom(new InternetAddress(fromAddr));                    //指定发送者
msg.setText(text);                                             //指定发送内容
```

（2）使用Transport类的send()方法，可以将Message对象封装的信息发送给接收方，实现代码如下：

```
Transport.send(msg);                                           //发送邮件
```

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承自JFrame类的窗体类SendMailFrame。

（3）将SendMailFrame窗体的内容面板设置为绝对布局，然后放置相应的控件完成窗体界面的设计。

（4）在SendMailFrame窗体类中，定义一个init()方法，用于完成属性设置和Session对象的创建，该方法的代码如下：

```
public void init() throws Exception {
    Properties props=new
Properties();                              //创建属性对象
    props.put("mail.transport.protocol",
sendProtocol);                  //指定邮件传输协议
    props.put("mail.smtp.class",
"com.sun.mail.smtp.SMTPTransport");  //指定传输协议使用的类
    props.put("mail.smtp.host",
sendHost);                              //定义发送邮件的主机
    session=Session.getDefaultInstance(props);
        //创建Session对象
}
```

（5）在SendMailFrame窗体类中，定义一个sendMessage()方法，用于完成邮件的发送，该方法的代码如下：

```
/**
*@param fromAddr 发送者
*@param toAddr 接收者
*@param title 主题
*@param text 内容
*@throws Exception 异常
```

```
    */
    public void sendMessage(String fromAddr,String
toAddr,String title,String text) throws Exception {
      Message msg=new
  MimeMessage(session);                    //创建Message对象
      InternetAddress[] toAddrs=
  InternetAddress.parse(toAddr,false);    //创建接收方的
  InternetAddress对象
      msg.setRecipients(Message.RecipientType.TO,
  toAddrs);          //指定接收方
      msg.setSentDate(new
  Date());                                 //指定发送日期
      msg.setSubject(title);
      //设置主题
      msg.setFrom(new
  InternetAddress(fromAddr));                  //指定发送者
      msg.setText(text);
      //指定发送内容
      Transport.send(msg);
    //发送邮件
      JOptionPane.showMessageDialog(null, "邮件发送成功。");
    }
```

举一反三

根据本实例，读者可以开发以下程序。

发送纯文本邮件。

发送带图片的邮件。

实例441 接收邮件

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\16\Ex16_441

实例说明

本实例实现了简单邮件的接收，即只能接收文本信息的邮件。运行程序，单击窗体上的“接收邮件”按钮，可以接收到已经发送的邮件，效果如图16.3所示。

图16.3 接收邮件窗体

技术要点

本实例主要是通过为Store对象指定接收邮件协议和连接接收服务器，然后通过Store对象的getFolder()方法获得inbox邮件夹的Folder对象，进而获得接收到的邮件信息。

（1）使用Store类指定接收邮件协议和连接服务器，可以通过如下代码实现：

```
Properties props=new
Properties();                          //声明Properties对象
```

```
    props.put("mail.store.protocol",
receiveProtocol);                    //指定接收协议
    props.put("mail.imap.class",
"com.sun.mail.imap.IMAPStore");     //指定使用Store进行接收
    session=Session.getDefaultInstance(props);
    //获得Session对象
    store=
session.getStore(receiveProtocol);                          //获得
Store对象
    store.connect(receiveHost,username,password);
    //连接接收服务器
```

（2）使用Store对象的getFolder()方法获得inbox邮件夹的Folder对象，可以通过如下代码实现：

```
    Folder folder=
store.getFolder("inbox");                          //获得inbox邮件
夹的Folder对象
```

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承自JFrame类的窗体类ReceiveMailFrame。

（3）在ReceiveMailFrame窗体的内容面板中部添加一个滚动面板，再在滚动面板上添加一个文本域控件，然后在窗体的底部添加一个面板，并在面板上添加两个命令按钮，完成窗体界面的设计。

（4）在ReceiveMailFrame窗体类中，定义一个init()方法，用于完成属性设置和Store对象的创建，该方法的代码如下：

```
    public void init() throws Exception {
```

```
    Properties props=new
  Properties();                                    //声明Properties
  对象
    props.put("mail.store.protocol",
  receiveProtocol);                              //指定接收协议
    props.put("mail.imap.class",
  "com.sun.mail.imap.IMAPStore");                //指定使用Store
  进行接收
    session=Session.getDefaultInstance(props);
          //获得Session对象
    store=
  session.getStore(receiveProtocol);
  //获得Store对象
    store.connect(receiveHost,username,password);
          //连接接收服务器
  }
```

（5）在ReceiveMailFrame窗体类中，定义一个receiveMessage()方法，用于完成邮件的接收，该方法的代码如下：

```
  public void receiveMessage() throws Exception {
    Folder folder=
  store.getFolder("inbox");                          //获得
  inbox邮件夹的Folder对象
    if (folder == null) {
      throw new Exception("不存在inbox邮件夹。");
    }
    folder.open(Folder.READ_ONLY);
        //以只读方式打开邮件夹
```

```
    ta_receive.append("您共收
到"+folder.getMessageCount()+"个电子邮件。\n\n");
    Message[]messages=
folder.getMessages();                          //获得邮件夹中
的所有邮件
    for (int i = 0;i<messages.length;i++){
      ta_receive.append("----第"+(i+1)+"个邮件----\n");
      ta_receive.append("主
  题："+folder.getMessage(i+1).getSubject()+"\n");      //
  主题
      ta_receive.append("正
  文："+folder.getMessage(i+1).getContent()+"\n");      //
  正文
      ta_receive.append("发送日
  期："+folder.getMessage(i+1).getSentDate()+"\n");  //发送
  日期
      Address[] ias=
  folder.getMessage(i+1).getFrom();                      //
  发件人地址
      ta_receive.append("发件人："+ias[0]+"\n");
      Address[] iasTo=
  folder.getMessage(i+1).getAllRecipients();              /
  /收件人地址
      ta_receive.append("收件人："+iasTo[0]+"\n\n");
    }
    folder.close(false);
              //关闭邮件夹，但不删除邮件
```

```
  store.close();
          //关闭Store对象
}
```

举一反三

根据本实例，读者可以实现以下功能。

实现离线接收功能。

16.2 复杂邮件

实例442 发送带附件的邮件

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\16\Ex16_442

实例说明

在发送邮件时，除了发送简单的文本信息外，还可以发送文件，即常说的附件，本实例实现了带附件的邮件的发送。运行程序，输入收件人地址、发件人地址、主题和正文后，如果需要发送附件，可以单击“添加附件”按钮，然后从打开的对话框中选择需要发送的附件，将其添加到附件列表中，然后单击“发送”按钮，如果邮件发送成功将显示消息框进行提示，效果如图16.4和图16.5所示。

图16.4 发送带附件的邮件

图16.5 提示附件发送成功

技术要点

本实例主要是通过向Multipart对象添加多个MimeBodyPart对象（每个MimeBodyPart对象代表一部分内容，可以是文本内容，也可以是附件），再将Multipart对象作为Message对象的setContent()方法的参数，从而实现了带附件邮件的发送。

发送带文本和附件的邮件，可以通过如下代码实现：

```
    Multipart multipart=new
MimeMultipart();                                    //可以添加复杂内
容的Multipart对象
    MimeBodyPart mimeBodyPartText=new
MimeBodyPart();                        //添加正文的MimeBodyPart对象
    mimeBodyPartText.setText(text);
          //指定正文
    multipart.addBodyPart(mimeBodyPartText);
            //添加到Multipart对象上
    if (filePathAndName!=null && !filePathAndName.equals(""))
{
      MimeBodyPart mimeBodyPartAdjunct=new
  MimeBodyPart();              //添加附件的MimeBodyPart对象
      FileDataSource fileDataSource=new
  FileDataSource(filePathAndName);     //创建附件的
```

```
FileDataSource对象
    mimeBodyPartAdjunct.setDataHandler(new
DataHandler(fileDataSource));  //指定数据
    mimeBodyPartAdjunct.setDisposition(Part.ATTACHMENT);
          //指定添加的内容是附件
    String name = fileDataSource.getName();
    mimeBodyPartAdjunct.setFileName(MimeUtility.encodeText(
name, "GBK", null));  //指定附件文件的名称
    //添加到Multipart对象上
multipart.addBodyPart(mimeBodyPartAdjunct);
  }
  msg.setContent(multipart);
          //设置邮件内容
  Transport.send(msg);
        //发送邮件
```

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承自JFrame类的窗体类SendAttachmentMailFrame。

（3）将 SendAttachmentMailFrame 窗体的内容面板设置为绝对布局，然后在内容面板的合适位置放置标签、文本框、文本域和命令按钮等控件，完成窗体界面的设置。

（4）在SendAttachmentMailFrame窗体类中定义sendMessage()方法，用于完成带附件邮件的发送，该方法的代码如下：

```
/**
*@param fromAddr 发送方地址
*@param toAddr 接收方地址
```

```
    *@param title 主题
    *@param text 文本内容
    *@throws Exception 异常
    */
    public void sendMessage(String fromAddr,String
toAddr,String title,String text) throws Exception {
      Message msg=new
  MimeMessage(session);                          //创建Message
  对象
      InternetAddress[] toAddrs=
  InternetAddress.parse(toAddr,false);           //接收方地址
      msg.setRecipients(Message.RecipientType.TO,
  toAddrs);                  //指定接收方
      msg.setSentDate(new
  Date());                                       //设置发送日期
      msg.setSubject(title);
            //设置主题
      msg.setFrom(new
  InternetAddress(fromAddr));                          //设置发送
  地址
      //这里省略了关键技术中给出的代码
      filePathAndName = null;
      JOptionPane.showMessageDialog(null, "邮件发送成功。");
    }
```

举一反三

根据本实例，读者可以实现以下功能。

实现多个附件的发送。

实现选择磁盘上的内容并发送。

实例443 接收带附件的邮件

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\16\Ex16_443

实例说明

接收邮件时，除了可以接收只有文本信息的邮件，还可以接收带有附件的邮件，本实例实现了带附件邮件的接收。运行程序，单击窗体上的“接收邮件并下载附件”按钮，将在文本域中显示所有已接收邮件的内容，如果有附件还会显示“保存”对话框实现附件文件的保存，效果如图16.6和图16.7所示。

图16.6 接收邮件窗体

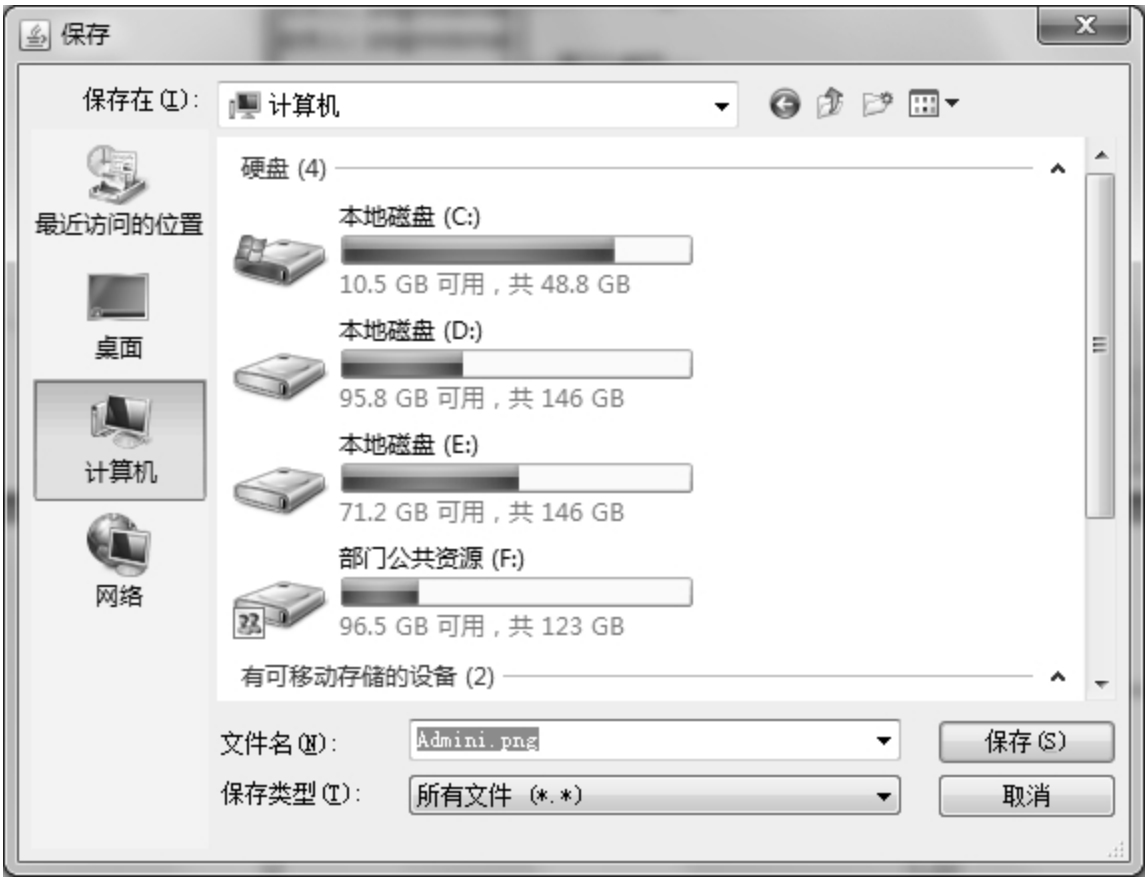


图16.7 用于保存附件的对话框

技术要点

本实例的实现主要是通过对接收内容进行分类处理，对于文本内容直接显示在文本域中；对于有附件的内容，除了在文本域中显示文本，还显示附件的文件名称，同时显示“保存”对话框保存接收到的附件文件。

对接收到的内容进行分类处理，可以通过如下代码实现：

```
Message[]messages= folder.getMessages();          //获得邮件夹中的所有邮件
for (int i=0; i<messages.length; i++) {           //遍历所有邮件
    Object content=messages[i].getContent();      //获得邮件内容
```

```
    if (content instanceof Multipart) {              //邮件内
容是Multipart的实例，说明有附件，否则说明没有附件
      //有附件执行的代码
    } else {
      //没有附件执行的代码
    }
  }
```

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承自JFrame类的窗体类ReceiveMailFrame。

（3）在ReceiveMailFrame窗体的内容面板中部添加一个滚动面板，再在滚动面板上添加一个文本域控件，然后在窗体的底部添加一个面板，并在面板上添加两个命令按钮，完成窗体界面的设计。

（4）在ReceiveMailFrame窗体类中定义一个receiveMessage()方法，实现对有附件邮件和无附件邮件的接收，该方法的代码如下：

```
  public void receiveMessage() throws Exception {
    Folder folder=
store.getFolder("inbox");                                    //
获得inbox邮件夹的Folder对象
    if (folder == null) {
      throw new Exception("不存在inbox邮件夹。");
    }
    folder.open(Folder.READ_ONLY);
                    //以只读方式打开邮件夹
    ta_receive.append("您共收到"+ folder.getMessageCount()
+"个电子邮件。\n\n");
```

```
    Message[]messages=
folder.getMessages();                                    //获
得邮件夹中的所有邮件
    for (int i = 0; i < messages.length; i++) {
      Object
  content=messages[i].getContent();
          //获得邮件内容
      if (content instanceof Multipart)
  {                                          //有附件执行的代码
        ta_receive.append("----第"+ (i + 1) +"个邮件----
    \n");
        ta_receive.append("主题："+
    folder.getMessage(i+1).getSubject()+ "\n");    //主题
        Multipart mPart= (Multipart)
    content;                                    //转换为
    Multipart对象
        ta_receive.append("正
    文："+mPart.getBodyPart(0).getContent()+
    "\n");              //正文内容
        ta_receive.append("发送日期："+
    folder.getMessage(i+1).getSentDate()+ "\n");      //发
    送日期
        Address[] ias=
    folder.getMessage(i+1).getFrom();
          //发件人地址
        ta_receive.append("发件人："+ ias[0] +"\n");
```

```
    Address[] iasTo=
folder.getMessage(i+1).getAllRecipients();
        //收件人地址
    ta_receive.append("收件人："+ iasTo[0] +"\n\n");
    try {
      String
  fileName=mPart.getBodyPart(1).getFileName();
          //获得文件名
      ta_receive.append("接收到一个名为
  “"+MimeUtility.decodeText(fileName)+ "” 的附件\n");
      InputStream in =
  mPart.getBodyPart(1).getInputStream();
      FileDialog dialog=new
  FileDialog(ReceiveMailFrame.this, "保存");      //创
  建对话框
      dialog.setMode(FileDialog.SAVE);
                  //设置对话框为保存对话框
      dialog.setFile(MimeUtility.decodeText(fileName));
      dialog.setVisible(true);
                  //显示保存对话框
      String
  path=dialog.getDirectory();
      //获得文件的保存路径
      String
  saveFileName=dialog.getFile();
        //获得保存的文件名
      if (path == null || saveFileName == null) {
```

```
        return;
      }
      OutputStream out = new BufferedOutputStream(new
  FileOutputStream(path +"/"+ saveFileName));
      //保存附件文件的输出流
      int len =-1;
      while ((len = in.read()) !=-1) {
        out.write(len);
              //向输出流写入数据
      }
      out.close();
              //关闭输出流对象
      in.close();
              //关闭输入流对象
    } catch (Exception ex) {
    }
  } else {    //没有附件执行的代码
    ta_receive.append("----第"+ (i + 1) +"个邮件----
\n");
    ta_receive.append("主题："+
folder.getMessage(i+1).getSubject()+ "\n");          //
主题
    ta_receive.append("正文："+
folder.getMessage(i+1).getContent()+ "\n");          //
正文
    ta_receive.append("发送日期："+
folder.getMessage(i+1).getSentDate()+ "\n");      //发
```

```
  送日期
      Address[] ias=
  folder.getMessage(i+1).getFrom();
        //发件人地址
      ta_receive.append("发件人："+ ias[0] +"\n");
      Address[] iasTo=
  folder.getMessage(i+1).getAllRecipients();
          //收件人地址
      ta_receive.append("收件人："+ iasTo[0] +"\n\n");
    }
  }
  folder.close(false);
  //关闭邮件夹，但不删除邮件
  store.close();
//关闭Store对象
}
```

举一反三

根据本实例，读者可以开发以下程序。

根据需要选择要下载的附件。

实现离线接收功能。

实例444 发送邮件时进行身份验证

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\16\Ex16_444

实例说明

本实例演示如何在发送邮件时，对发送者的身份进行验证。运行程序，输入收件人地址、发件人地址、主题和正文后，如果需要发送附件，可以单击“添加附件”按钮，并从打开的对话框中选择需要发送的附件，将其添加到附件列表中，然后单击“发送”按钮，将弹出“身份验证对话框”要求输入用户名和密码，效果如图16.8和图16.9所示。

技术要点

本实例将SMTP邮件服务器的mail.smtp.auth属性设置为true，这样当用户使用SMTP邮件服务器发送邮件时，就需要对其身份进行验证，然后创建一个 Authenticator 类的子类CheckAuthenticator，在该子类中定义一个对话框类和一个返回值类型是PasswordAuthentication的 getPasswordAuthentication()方法，再创建一个 CheckAuthenticator 对象，并将该对象传递给Session对象的getDefaultInstance()方法，这样在使用Transport类的send()方法发送信息时，就会显示进行身份验证的对话框。

图16.8 发送邮件窗体

图16.9 对发送者进行身份验证

设置SMTP邮件服务器的mail.smtp.auth属性、将Authenticator类的子类CheckAuthenticator的对象传递给Session对象的getDefaultInstance()方法，可以通过如下代码实现：

```
Properties props=new
Properties();                              //创建属性对象
props.put("mail.smtp.auth",
"true");                                   //设置发邮件时需要身份验证
Authenticator check=new
CheckAuthenticator();                      //创建Authenticator实例
session=Session.getDefaultInstance(props,check);
  //创建Session对象
```

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承 Authenticator 类的 CheckAuthenticator 类，该类用于弹出身份验证对话框，对发送邮件的用户进行身份验证，该类的关键代码如下：

```
public class CheckAuthenticator extends Authenticator {
  class UserPassDialog extends JDialog
{                          //用于显示对话框的内部类
```

```
    private JPasswordField pf_pwd;                    //
密码文本框
    private JTextField tf_user;                        //
用户名文本框
    public UserPassDialog() {
      super();
      //省略了初始化对话框的代码
    }
  }
  public PasswordAuthentication
getPasswordAuthentication() {
    UserPassDialog dialog=new UserPassDialog();        //
创建对话框
    dialog.setModal(true);                            //设
置为模式对话框
    dialog.setVisible(true);                           //
显示对话框
    String
user=dialog.tf_user.getText().trim();              //获得用
户名
    String pwd=new
String(dialog.pf_pwd.getPassword());  //获得密码
    dialog.tf_user.setText(null);
//清空用户名文本框
    dialog.pf_pwd.setText(null);                       //
清空密码文本框
```

```
        return new
    PasswordAuthentication(user,pwd);        //返回
    PasswordAuthentication对象
      }
    }
```

（3）在项目中创建一个继承JFrame类的SendAttachmentMailFrame窗体类，将该窗体的内容面板设置为绝对布局，然后放置相应的控件完成窗体界面的设计。

（4）在SendAttachmentMailFrame窗体类中，定义一个init()方法，用于设置SMTP邮件服务器的mail.smtp.auth属性、将Authenticator类的子类CheckAuthenticator的对象传递给Session对象的getDefaultInstance()方法，这样当用户发送邮件时就需要进行身份验证，该方法的代码如下：

```
    public void init() throws Exception {
      Properties props=new
  Properties();                          //创建属性对象
      props.put("mail.transport.protocol",
  sendProtocol);                  //指定邮件传输协议
      props.put("mail.smtp.class",
  "com.sun.mail.smtp.SMTPTransport");  //指定传输协议使用的类
      props.put("mail.smtp.host",
  sendHost);                          //定义发送邮件的主机
      props.put("mail.smtp.auth",
  "true");                            //设置发邮件时需要身份验
  证
      Authenticator check=new
  CheckAuthenticator();                 //创建Authenticator实
```

例

```
        session=Session.getDefaultInstance(props,check);
            //创建Session对象
    }
```

（5）在SendAttachmentMailFrame窗体类中，定义一个sendMessage()方法，用于完成邮件的发送，并且在发送邮件时会显示身份验证对话框，该方法的代码与实例442中的sendMessage()方法相同，这里不再重复给出，如果需要请查看源程序代码。

举一反三

根据本实例，读者可以实现以下功能。

修改间隔验证时间。

添加动态验证码。

实例445 接收邮件时进行身份验证

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\16\Ex16_445

实例说明

本实例演示了如何在接收邮件时，对用户的身份进行验证。运行程序，将弹出“身份验证对话框”，要求用户输入用户名和密码，输入正确，单击对话框中的“确定”按钮，将显示接收邮件窗口，单击该窗口中的“接收邮件并下载附件”按钮，将显示接收到的邮件信息，效果如图16.10和图16.11所示。

图16.10 对收件者进行身份验证的对话框

图16.11 接收邮件窗体

技术要点

由于在使用IMAP服务器接收邮件时，需要使用Store对象的connect()方法，而该方法需要指定收件者的用户名和密码，如果将该方法的用户名和密码设置为 null ，并创建一个Authenticator类的子类CheckAuthenticator，在该子类中定义一个对话框类和一个返回值类型是PasswordAuthentication的getPasswordAuthentication方法，然后创建一个CheckAuthenticator对象，并将该对象传递给Session对象的getDefaultInstance()方法，这样在使用Store对象的connect()方法连接邮件服务器时，就会显示进行身份验证的对话框。

对接收邮件者进行身份验证，可以通过如下代码实现：

```
    Properties props=new
Properties();                          //声明Properties对象
    Authenticator check=new
CheckAuthenticator();                  //创建Authenticator实例
    session=Session.getDefaultInstance(props,check);
      //获得Session对象
    store=
session.getStore(receiveProtocol);                    //获得
Store对象
    store.connect(receiveHost,null,null);
    //连接接收服务器，并进行身份验证
```

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承 Authenticator 类的 CheckAuthenticator 类，该类用于弹出身份验证对话框，对接收邮件的用户进行身份验证，该类的实现代码与实例444中的代码相同，这里不再重复给出，如果需要请查看源程序代码。

（3）在项目中创建一个继承 JFrame类的 ReceiveMailFrame 窗体类，在该窗体内容面板的中部添加一个滚动面板，再在滚动面板上添加一个文本域控件，然后在窗体的底部添加一个面板，并在面板上添加两个命令按钮，完成窗体界面的设计。

（4）在ReceiveMailFrame窗体类中，定义一个init()方法，用于设置接收邮件使用的协议，以及将Authenticator类的子类 CheckAuthenticator的对象传递给Session对象的 getDefaultInstance()方法，获得 Store 对象并连接到接收服务器，这样当用户接收邮件时就需要进行身份验证，该方法的代码如下：

```
    public void init() throws Exception {
```

```
    Properties props=new
Properties();                              //声明Properties对象
    props.put("mail.store.protocol",
receiveProtocol);                  //指定接收协议
    props.put("mail.imap.class",
"com.sun.mail.imap.IMAPStore");       //指定使用Store进行接
收
    Authenticator check=new
CheckAuthenticator();                  //创建Authenticator实
例
    session=Session.getDefaultInstance(props,check);
        //获得Session对象
    store=
session.getStore(receiveProtocol);                          //
获得Store对象
    store.connect(receiveHost,null,null);
      //连接接收服务器，并进行身份验证
  }
```

（5）在ReceiveMailFrame窗体类中，定义一个receiveMessage()方法，用于完成邮件的接收，该方法的代码与实例443中的receiveMessage()方法相同，这里不再重复给出，如果需要请查看源程序代码。

举一反三

根据本实例，读者可以实现以下功能。

验证接收者的身份。

等待超时重新验证接收者身份。

实例446 显示未读邮件

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\16\Ex16_446

实例说明

本实例实现了显示未读邮件的功能。随着邮件数量的不断增加，查看邮件会比较耗时费力，为此可以直接将未读邮件读取出来，以减少查看邮件的时间。运行程序，单击窗体上的“接收邮件并下载附件”按钮，可以将未读邮件的内容显示在文本域中，效果如图16.12所示。

图16.12 显示未读邮件的效果

技术要点

本实例主要是通过获得邮件的 UID 来实现显示未读邮件的功能的，方法是将已读邮件的UID保存到文件中，下次读取时，判断读取的UID是否在文件中，如果不存在，则表示是未读邮件，并读取邮件的内容。

获得邮件的UID可以通过IMAPFolder类的getUID()方法，该方法需要一个Message类型的参数。获得邮件UID的代码如下：

```
Message[]messages=
folder.getMessages();                                    //获得邮件夹中的所有邮件
for (int i = 0; i < messages.length; i++) {
  long uid=
folder.getUID(messages[i]);                              //获得邮件的UID
}
```

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的ReceiveMailFrame窗体类。

（3）在ReceiveMailFrame窗体的内容面板中部添加一个滚动面板，再在滚动面板上添加一个文本域控件，然后在窗体的底部添加一个面板，并在面板上添加两个命令按钮，完成窗体界面的设计。

（4）在 ReceiveMailFrame 窗体类中定义一个receiveMessage()方法，实现通过邮件的 UID判断未读邮件并读取未读邮件内容的功能，该方法的代码如下：

```
public void receiveMessage() throws Exception {
  IMAPFolder folder= (IMAPFolder)
store.getFolder("inbox");              //获得inbox邮件夹的IMAPFolder对象
  if (folder == null) {
    throw new Exception("不存在inbox邮件夹。");
  }
```

```
    folder.open(Folder.READ_ONLY);
        //以只读方式打开邮件夹
    Message[]messages=
folder.getMessages();                          //获得邮件夹中
的所有邮件
    for (int i = 0; i < messages.length; i++) {
      long uid=
  folder.getUID(messages[i]);                          //获
  得邮件的UID
      try {
        objectIn = new ObjectInputStream(new
    FileInputStream("c:/readedEmail.txt"));
        //创建输入流对象
        Object
    readObj=objectIn.readObject();                      //
    读取信息
        if (readObj==null) {                          //文件
    中未存储已读文件的UID，说明是未读邮件
          readedEmailUIDVector.add(String.valueOf(uid));
            //将读取邮件的UID添加到向量中
          objectOut = new ObjectOutputStream(new
      FileOutputStream("c:/readedEmail.txt"));
          //创建输出流对象
          objectOut.writeObject(readedEmailUIDVector);
            //将已读邮件UID的向量写入磁盘
          //读取邮件信息
          noReadNums++;
```

```
    readMessage(messages[i], folder,
i);                    //调用readMessage()方法，读取
邮件信息
  } else {
    boolean readedFlag=
false;                          //为false表示未读
    readedEmailUIDVector = (Vector<String>) readObj;
    for (int j = 0; j < readedEmailUIDVector.size();
j++) {
      if
  (String.valueOf(uid).equals(readedEmailUIDVector.ge
  t(j))) {
        readedFlag= true;                          //
    标记为true，表示已读
        break;
      }
    }
    if (readedFlag) {
      continue;
    } else {
      readedEmailUIDVector.add(String.valueOf(uid));
              //将读取邮件的UID添加到向量中
      objectOut = new ObjectOutputStream(new
  FileOutputStream("c:/readedEmail.txt"));
      //创建输出流对象
      objectOut.writeObject(readedEmailUIDVector);
        //将已读邮件UID的向量写入磁盘
```

```
            //读取邮件信息
            noReadNums++;
            readMessage(messages[i], folder,
        i);                    //调用readMessage()方法，读取邮
        件信息
          }
        }
        ///
      } catch (Exception ex) {
        readedEmailUIDVector.add(String.valueOf(uid));
              //将读取邮件的UID添加到向量中
        objectOut = new ObjectOutputStream(new
    FileOutputStream("c:/readedEmail.txt"));
        //创建输出流对象
        objectOut.writeObject(readedEmailUIDVector);
              //将已读邮件UID的向量写入磁盘
        //读取邮件信息
        noReadNums++;
        //调用readMessage()方法，读取邮件信息
    readMessage(messages[i], folder, i);
      }
    }
    ta_receive.append("您共收到"+ folder.getMessageCount()
+"个邮件。\n");
    if (noReadNums > 0) {
      ta_receive.append("其中上述"+ noReadNums +"个是新邮
  件。\n\n");
```

```
    } else {
      ta_receive.append("没有未读邮件。\n\n");
    }
    folder.close(false);
          //关闭邮件夹，但不删除邮件
    store.close();
        //关闭Store对象
    if (objectOut != null)
      objectOut.close();
    if (objectIn != null)
      objectIn.close();
    }
```

（5）在 ReceiveMailFrame 窗体类中定义 readMessage()方法，实现读取邮件内容的功能，该方法的代码如下：

```
    public void readMessage(Message message, IMAPFolder
folder, int i) throws Exception {
      Object
  content=message.getContent();
      //获得邮件内容
      if (content instanceof Multipart)
  {                                          //有附件执行的代码
        ta_receive.append("----第"+ (i + 1) +"个邮件----\n");
        ta_receive.append("主题："+
    folder.getMessage(i+1).getSubject()+ "\n");        //主题
        Multipart mPart= (Multipart)
    content;                                          //创建
    Multipart对象
```

```
    ta_receive.append("正
文："+mPart.getBodyPart(0).getContent()+
"\n");          //正文
    ta_receive.append("发送日期："+
folder.getMessage(i+1).getSentDate()+ "\n");    //发送日
期
    Address[] ias=
folder.getMessage(i+1).getFrom();
  //发件人地址
    ta_receive.append("发件人："+ ias[0] +"\n");
    Address[] iasTo=
folder.getMessage(i+1).getAllRecipients();
    //收件人地址
    ta_receive.append("收件人："+ iasTo[0] +"\n\n");
    try {
      String
  fileName=mPart.getBodyPart(1).getFileName();
            //获得文件名
      ta_receive.append("接收到一个名为“"+
  MimeUtility.decodeText(fileName) +"”的附件\n\n");
      //获得附件文件名
      InputStream
  in=mPart.getBodyPart(1).getInputStream();
      //获得输入流对象
      FileDialog dialog=new
  FileDialog(ReceiveMailFrame.this, "保存");          //
  创建对话框
```

```
    dialog.setMode(FileDialog.SAVE);
            //设置对话框为保存对话框
    dialog.setFile(MimeUtility.decodeText(fileName));
                //设置对话框显示的文件名
    dialog.setVisible(true);
            //显示保存对话框
    String
path=dialog.getDirectory();
  //获得文件的保存路径
    String
saveFileName=dialog.getFile();
  //获得保存的文件名
    if (path == null || saveFileName == null) {
      return;
    }
    OutputStream out = new BufferedOutputStream(new
FileOutputStream(path +"/"+ saveFileName));
    //创建输出流对象
    int len =-1;
    while ((len= in.read()) != -1)
{                       //读取内容，如果没到文件尾则执行
循环体
      out.write(len);
          //写入文件
    }
    out.close();
    in.close();
```

```
        } catch (Exception ex) {
        }
    } else
{                                                          //
没有附件执行的代码
        ta_receive.append("----第"+ (i + 1) +"个邮件----\n");
        ta_receive.append("主题："+
  folder.getMessage(i+1).getSubject()+ "\n");          //主题
        ta_receive.append("正文："+
  folder.getMessage(i+1).getContent()+ "\n");          //正文
        ta_receive.append("发送日期："+
  folder.getMessage(i+1).getSentDate()+ "\n");     //发送日
  期
        Address[] ias=
  folder.getMessage(i+1).getFrom();
    //发件人地址
        ta_receive.append("发件人："+ ias[0] +"\n");
        Address[] iasTo=
  folder.getMessage(i+1).getAllRecipients();
        //收件人地址
        ta_receive.append("收件人："+ iasTo[0] +"\n\n");
    }
  }
```

举一反三

根据本实例，读者可以实现以下功能。

通过窗体界面选择需要的未读邮件。

显示未读邮件的条数。

实例447 显示已读邮件

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\16\Ex16_447

实例说明

本实例实现了显示已读邮件的功能。在邮件读取完之后，由于某种原因可能还需要对已读邮件进行查看，为此可以通过查看已读邮件快速找到需要的邮件，而不必查看所有邮件。运行程序，单击窗体上的“接收邮件并下载附件”按钮，可以将已读邮件的内容显示在文本域中，效果如图16.13所示。

图16.13 读取已读邮件的效果

技术要点

本实例主要是通过获得邮件的UID，从而实现显示已读邮件的功能，方法是在读取邮件时，首先获得邮件的UID，然后判断该邮件的UID是否与文件中存储的已读邮件的UID相同，如果相同，则表示是已读邮件。

判断邮件为已读邮件可以通过如下代码实现：

```
Message[]messages=
folder.getMessages();                                    //获得
邮件夹中的所有邮件
for (int i = 0; i < messages.length; i++) {
  long uid=
folder.getUID(messages[i]);
//获得邮件的UID
  try{
    objectIn=new ObjectInputStream(new
FileInputStream("c:/readedEmail.txt"));      //创建输入流
对象
    Object
readObj=objectIn.readObject();                          //从输
入流读取信息
    if (readObj==null)
{                                  //文件中未存储已读文件的
UID
      JOptionPane.showMessageDialog(null, "没有已读邮
  件。");
      return;
    } else {
      readedEmailUIDVector = (Vector<String>) readObj;
      //将从文件中读取的内容转换为Vector对象
      for (int j=0; j< readedEmailUIDVector.size(); j++)
  {                          //遍历向量对象
```

```
            if
        (String.valueOf(uid).equals(readedEmailUIDVector.get(
        j))) {          //是已读邮件
              ReadedNums++;
                //已读邮件数加1
              readMessage(messages[i], folder,
          i);             //调用readMessage()方法读取已读邮件
          内容
            }
          }
        }
      } catch (Exception ex) {
        ex.printStackTrace();
      }
    }
```

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的ReceiveMailFrame窗体类。

（3）在ReceiveMailFrame窗体的内容面板中部添加一个滚动面板，再在滚动面板上添加一个文本域控件，然后在窗体的底部添加一个面板，并在面板上添加两个命令按钮，完成窗体界面的设计。

（4）在 ReceiveMailFrame 窗体类中定义一个receiveMessage()方法，实现通过邮件的 UID判断已读邮件并读取已读邮件内容的功能，该方法的代码如下：

```
    public void receiveMessage() throws Exception {
```

```
    IMAPFolder folder= (IMAPFolder)
store.getFolder("inbox");                    //获得inbox邮件
夹
    if (folder == null) {
      throw new Exception("不存在inbox邮件夹。");
    }
    folder.open(Folder.READ_ONLY);
              //以只读方式打开邮件夹
    Message[]messages=
folder.getMessages();                                  //获得邮
件夹中的所有邮件
    for (int i = 0; i < messages.length; i++) {
      long uid=
  folder.getUID(messages[i]);
  //获得邮件的UID
      try {
        objectIn=new ObjectInputStream(new
    FileInputStream("c:/readedEmail.txt"));  //创建输入流对
    象
        Object
    readObj=objectIn.readObject();
      //从输入流读取信息
        if (readObj==null)
    {                                          //文件中未存储
    已读文件的UID
          JOptionPane.showMessageDialog(null, "没有已读邮
      件。");
```

```
            return;
          } else {
            readedEmailUIDVector = (Vector<String>) readObj;
            //将从文件中读取的内容转换为Vector对象
            for (int j=0; j< readedEmailUIDVector.size();
        j++) {              //遍历向量对象
              if
            (String.valueOf(uid).equals(readedEmailUIDVector.ge
            t(j))) {  //是已读邮件
                ReadedNums++;
            //已读邮件数加1
                readMessage(messages[i], folder,
            i);          //调用readMessage()方法读取已读邮件内
            容
              }
            }
          }
        } catch (Exception ex) {
          ex.printStackTrace();
        }
      }
      ta_receive.append("您共收到"+ folder.getMessageCount()+
"个邮件。\n");              //显示收到的邮件总数
      if (ReadedNums > 0) {
        if (ReadedNums == folder.getMessageCount()) {
          ta_receive.append("全部已读。\n\n");
        } else {
```

```
        ta_receive.append("其中上述"+ ReadedNums +"个是已读
    邮件。\n\n");
      }
    } else {
      ta_receive.append("没有已读邮件。\n\n");
    }
    folder.close(false);
              //关闭邮件夹，但不删除邮件
    store.close();
            //关闭Store对象
    if (objectOut != null)
      objectOut.close();
    if (objectIn != null)
      objectIn.close();
    }
```

（5）在ReceiveMailFrame窗体类中定义一个readMessage()方法，实现读取邮件的内容，该方法的代码与实例446中相同，这里不再重复给出。

举一反三

根据本实例，读者可以实现以下功能。

显示已读邮件的条数。

显示邮件的总条数。

第17章 Java安全

Java对称加密

Java非对称加密

Java单项加密

17.1 Java对称加密

实例448 使用BASE64加密

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\17\Ex17_448

实例说明

在 Java加密技术中，BASE64是一种最简单、最基本的加密技术。本实例使用 BASE64对字符串“明日科技”进行加密，使其变成一个不可以直接识别的字符串，运行效果如图17.1所示。

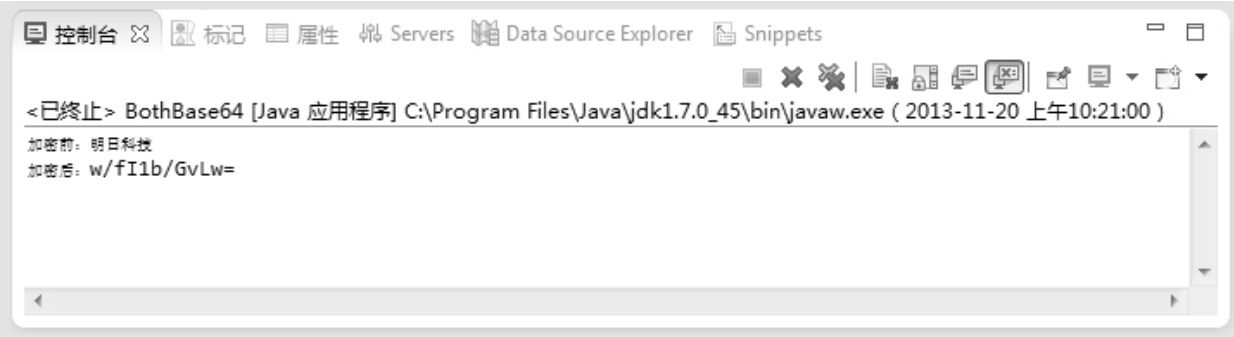


图17.1 使用BASE64 加密

技术要点

使用BASE64加密时，使用BASE64Encoder类的encodeBuffer()方法，该方法会产生一个String类型的返回值，即加密以后的字符。语法如下：

```
public static String encodeBuffer(byte[] arg0)
```

参数说明

arg0：表示要加密的数据。

实现过程

（1）新建一个JAVA文件。

（2）创建 encryptBASE64()方法，在方法内部创建一个BASE64Encoder 的对象，使用BASE64Encoder的encodeBuffer()方法加密数据，代码如下：

```
public static String encryptBASE64(byte[] data) {
  //加密数据
  return (new BASE64Encoder()).encodeBuffer(data);
}
```

（3）对“明日科技”字符串进行加密，先将其转成byte[]类型，再传到encryptBASE64()方法里，就可以得到加密后的数据，代码如下：

```
String data ="明日科技";
System.out.println("加密前："+ data);
String data1 = BothBase64.encryptBASE64(data.getBytes());
System.out.println("加密后："+ data1);
```

举一反三

根据本实例，读者可以开发以下程序。

对加密后的字符串进行解密。

实例449 使用BASE64解密

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\17\Ex17_449

实例说明

BASE64 的加密算法是固定的，对一个明文加密，无论加密多少次，每次产生的密文都是一样的，如使用BASE64对“明日科技”加密以后的密文是w/fI1b/GvLw=。

BASE64的加密方式是可逆的，本实例直接对密文w/fI1b/GvLw=进行解密，即可得到明文。运行效果如图17.2所示。

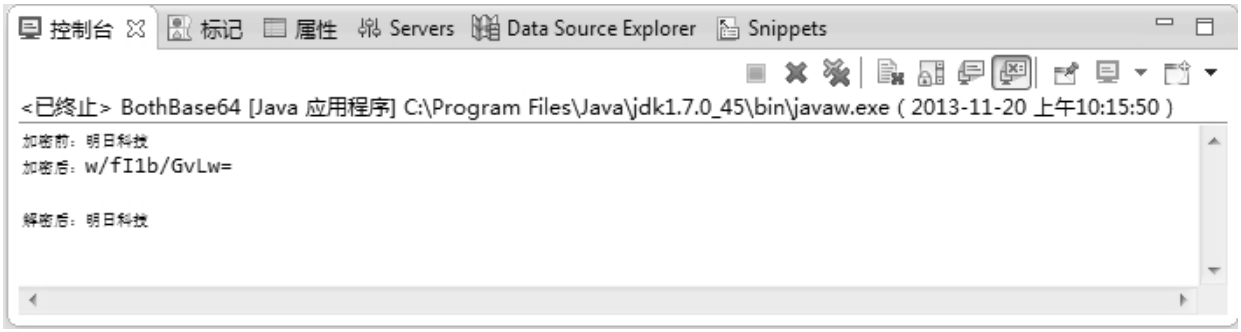


图17.2 使用BASE64 解密

技术要点

使用BASE64解密时，使用BASE64Decoder类的decodeBuffer()方法，该方法返回byte[]类型的数据，也即解密以后的数据。语法如下：

```
public static byte[] decodeBuffer(String data)
```

参数说明

data：表示要解密的数据。

实现过程

（1）新建一个JAVA文件。

（2）创建 encryptBASE64()方法，在方法内部创建一个BASE64Encoder 的对象，使用BASE64Encoder的encodeBuffer()方法加密数据。代码如下：

```
public static String encryptBASE64(byte[] data) {
  //加密数据
  return (new BASE64Encoder()).encodeBuffer(data);
}
```

（3）创建 decryptBASE64()方法，在方法内部创建一个BASE64Encoder 的对象，使用BASE64Encoder的decodeBuffer()方法解密数据。代码如下：

```
    public static byte[] decryptBASE64(String data) throws
IOException {
      //解密数据
      return (new BASE64Decoder()).decodeBuffer(data);
    }
```

（4）对“明日科技”字符串进行加密，先转成 byte[]类型再传到encryptBASE64()方法里，即可得到加密以后的数据。再使用decryptBASE64()方法对加密的数据进行解密，该方法的返回值是byte[]类型，把byte[]转换成String类型就可以得到“明日科技”。代码如下：

```
    public static void main(String[] avg) throws IOException
{
      String data ="明日科技";
      System.out.println("加密前："+ data);
      String data1 =
  BothBase64.encryptBASE64(data.getBytes());
      System.out.println("加密后："+ data1);
      byte[] data2 = BothBase64.decryptBASE64(data1);
      System.out.println("解密后："+ new String(data2));
    }
```

举一反三

根据本实例，读者可以实现以下功能。

使用BASE64加密文件。

实例450 使用DES加密

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\17\Ex17_450

实例说明

DES的密钥生成后，就可以对“明日科技”字符串加密了，由于加密前密钥保存在keyData.dat文件里，先使用readFile()方法读取密钥文件，再把文件内容转换成SecretKey对象，然后对字符串进行加密，加密后会产生一个byte[]类型的数据，将其保存在fileData.dat文件中。生成文件如图17.3所示。

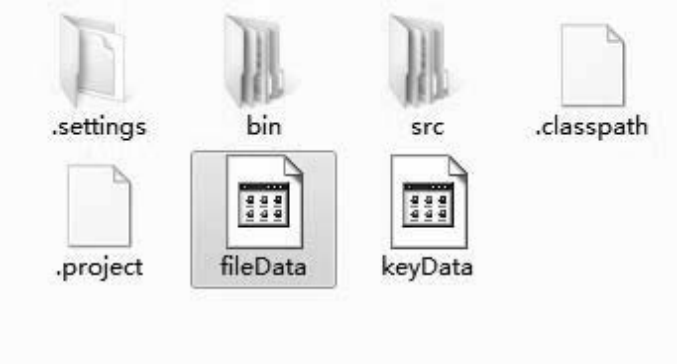


图17.3 生成加密文件

技术要点

使用Cipher 类的getInstance()方法可以返回实现指定转换的Cipher 对象，语法如下：

```
public static final Cipher getInstance(String
transformation) throws NoSuchAlgorithmException,
NoSuchPaddingException
```

参数说明

transformation：表示要转换的加密、解密算法。

实现过程

（1）新建一个JAVA文件。

（2）使用readFile()方法读取密钥数据或者密文数据，代码如下：

```
public byte[] readFile(String fileName) {
```

```
    try{
      //创建文件
      File file = new File(fileName);
      //创建文件输入流
      FileInputStream fileInputStream = new
  FileInputStream(file);
      byte[] data = new byte[(int) file.length()];
      //读取文件输入流
      fileInputStream.read(data);
      fileInputStream.close();
      return data;
    } catch (FileNotFoundException e1) {
      // TODO Auto-generated catch block
      e1.printStackTrace();
    } catch (IOException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
    }
    return null;
  }
```

（3）创建 toKey()方法，使用 readFile()方法读取密钥文件，使用 DESKeySpec 和SecretKeyFactory类把密钥数据转换成密钥对象，代码如下：

```
  private Key toKey() throws InvalidKeyException,
NoSuchAlgorithmException,InvalidKeySpecException {
    byte[] key = readFile(keyFile);
    DESKeySpec keySpec = new DESKeySpec(key);
```

```
        SecretKeyFactory factory =
    SecretKeyFactory.getInstance(algorithm);
        //将密钥转化成Key进行加密解密
        SecretKey secretKey = factory.generateSecret(keySpec);
        return secretKey;
    }
```

（4）创建 encrypt()方法，在方法中先调用 toKey()方法获取密钥，再使用Cipher.ENCRYPT_MODE常量和Key初始化Cipher，并对数据进行加密，然后把加密数据保存到文件中，代码如下：

```
    public void encrypt(byte[] data) throws
InvalidKeyException,NoSuchAlgorithmException,
InvalidKeySpecException,NoSuchPaddingException,
IllegalBlockSizeException,BadPaddingException {
        Key key = toKey();
        //使用Cipher实际完成加密操作
        Cipher cipher = Cipher.getInstance(algorithm);
        //使用密钥初始化Cipher
        cipher.init(Cipher.ENCRYPT_MODE, key);
        byte[] f = cipher.doFinal(data);
        writeFile(f, dataFile);
    }
```

举一反三

根据本实例，读者可以实现以下功能。

使用writeFile()方法把密文保存在.dat文件里。

使用writeFile()方法把密钥保存在.dat文件里。

实例451 使用DES解密

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\17\Ex17_451

实例说明

密钥保存在keyData.dat文件中，密文保存在fileData.dat文件中，只要解密方得到加密方传来的这两个文件，就可以对其进行解密，看到最后的明文，运行效果如图17.4所示。

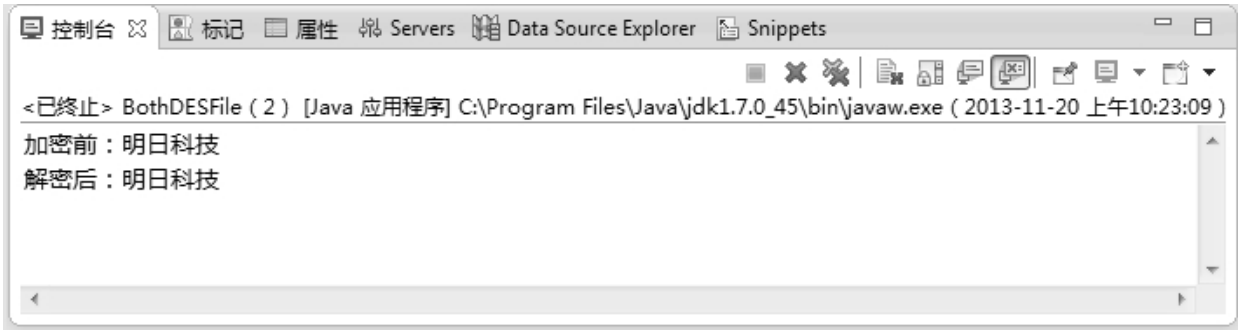


图17.4 使用DES 解密

这种应用是很常见的，如某软件公司把自己的软件发布到网上供大家下载，该软件就类似于实例中的fileData.dat文件，但是软件的下载者如果没有keyData.dat文件，就无法使用该软件，所以需要向软件公司支付一定的费用来获取keyData.dat文件。

技术要点

使用Cipher类的init()方法，可以用密钥初始化该Cipher，语法如下：

```
public final void init(int opmode, Key key) throws
InvalidKeyException
```

参数说明

transformation：表示要转换的加密、解密算法。

实现过程

（1）新建一个JAVA文件。

（2）使用readFile()方法读取密钥数据或者密文数据。代码见实例450。

（3）创建 toKey()方法，使用 readFile()方法读取密钥文件或者加密文件，使用 DESKeySpec 和SecretKeyFactory类把密钥数据转换成密钥对象，代码见实例450。

（4）创建decrypt()方法，在方法中调用toKey()方法获取密钥对象，然后创建一个DES的Cipher 实例，使用Cipher 类的init()方法进行初始化，使用常量Cipher. DECRYPT _MODE 指定Cipher为解密模式。接着通过readFile()方法从fileData.dat文件里获取密文，再使用Cipher类的doFinal()方法把密文数据转化成明文，代码如下：

```
public String decrypt() throws
InvalidKeyException,NoSuchAlgorithmException,
InvalidKeySpecException,NoSuchPaddingException,
IllegalBlockSizeException,BadPaddingException {
    Key key = toKey();
    Cipher cipher = Cipher.getInstance(algorithm);
    //指定解密
    cipher.init(Cipher.DECRYPT_MODE, key);
    byte[] f = readFile(dataFile);
    return new String(cipher.doFinal(f));
}
```

举一反三

根据本实例，读者可以开发以下程序。

使用Cipher.ENCRYPT_MODE常量加密。

使用Cipher.DECRYPT_MODE常量解密。

实例452 使用PBE加密

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\17\Ex17_452

实例说明

PBE的加密需要有盐值和密码，盐是随机生成的8位的byte[]类型，密码是事先设置好用来生成密钥的。本实例把加密后的数据保存在fileData.dat文件里，如图17.5所示。

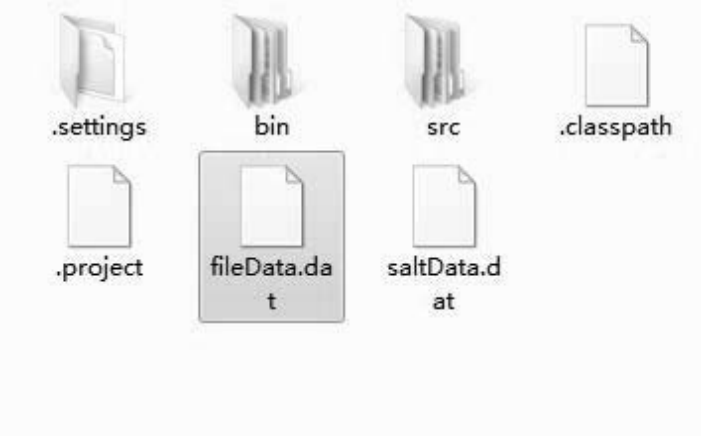


图17.5 生成密钥文件

技术要点

使用PBEParameterSpec的构造方法可以创建一个PBEParameterSpec实例，语法如下：

public PBEParameterSpec(byte[] salt, int iterationCount)

参数说明

- salt：表示PBE 加密时使用的盐值。
- iterationCount：表示PBE 加密时迭代的次数。

实现过程

（1）新建一个JAVA文件。

（2）使用readFile()方法从saltData.dat文件中获取盐值。

（3）创建encrypt()方法使用盐值和数值100生成PBEParameterSpec的对象。创建PBE的Cipher的实例，使用Cipher.ENCRYPT_MODE、key和paramSpec初始化cipher。然后使用

doFinal()方法把明文加密成密文，再使用writeFile()方法把密文数据保存到fileData.dat文件里。代码如下：

```
    public void encrypt(byte[] data, String password)throws
NoSuchAlgorithmException, InvalidKeySpecException,
NoSuchPaddingException,
InvalidKeyException,InvalidAlgorithmParameterException,
IllegalBlockSizeException, BadPaddingException {
        //生成密钥
        Key key = toKey(password);
        //获取盐值
        byte[] salt = readFile(saltFile);
        PBEParameterSpec paramSpec = new PBEParameterSpec(salt,
  100);
        Cipher cipher = Cipher.getInstance(algorithm);
        //加密数据
        cipher.init(Cipher.ENCRYPT_MODE, key, paramSpec);
        writeFile(cipher.doFinal(data), dataFile);
    }
```

举一反三

根据本实例，读者可以实现以下功能。

使用PBE加密文件。

生成加密密钥。

实例453 使用PBE解密

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\17\Ex17_453

实例说明

加密方确定密码以后就可以把密码告诉解密方，待加密完以后，再把盐值和加密数据传给解密方。解密方通过密码生成密钥，再使用密钥和盐对密文进行解密。本实例中，盐值和密文分别保存在saltData.dat 和 fileData.dat文件中。解密后即可看见明文数据，运行效果如图 17.6所示。

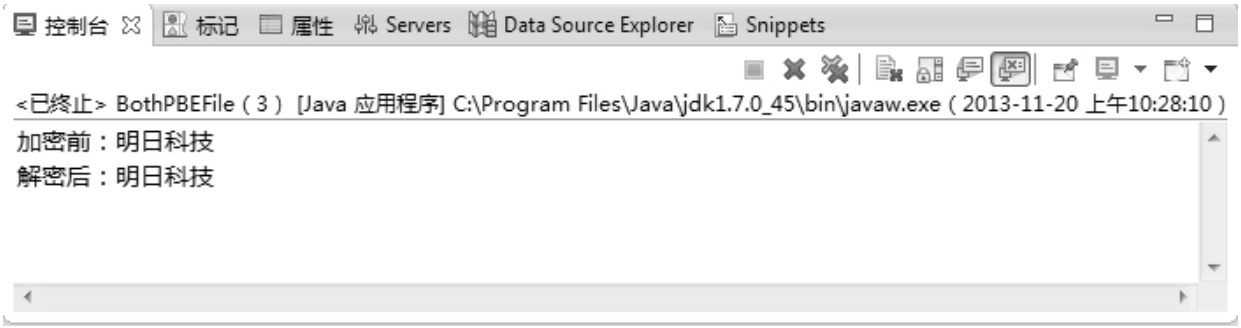


图17.6 使用PBE 解密

技术要点

使用Cipher类的doFinal()方法可以按单部分操作加密或解密数据，或者结束一个多部分操作，数据将被加密或解密，语法如下：

public final byte[] doFinal(byte[] input) throws IllegalBlockSizeException,BadPaddingException

参数说明

input：表示PBE加密的输入缓冲区。

实现过程

（1）新建一个JAVA文件。

（2）使用readFile()方法从saltData.dat文件中获取盐值，从fileData.dat文件中获取密文。

（3）使用盐值和数值100生成PBEParameterSpec的对象，然后创建PBE的Cipher的实例，并使用Cipher.DECRYPT_MODE、key和paramSpec初始化cipher，最后使用doFinal()方法将密文解密成明文，完成解密工作。代码如下：

```
    public String decrypt(String password) throws
NoSuchAlgorithmException, InvalidKeySpecException,
NoSuchPaddingException,InvalidKeyException,
InvalidAlgorithmParameterException,
IllegalBlockSizeException, BadPaddingException {
        //获取密钥
        Key key = toKey(password);
        //读取盐值
        byte[] salt = readFile(saltFile);
        //读取加密数据
        byte[] data = readFile(dataFile);
        PBEParameterSpec paramSpec = new PBEParameterSpec(salt,
  100);
        Cipher cipher = Cipher.getInstance(algorithm);
        cipher.init(Cipher.DECRYPT_MODE, key, paramSpec);
        //进行解密
        return new String(cipher.doFinal(data));
    }
```

举一反三

根据本实例，读者可以实现以下功能。

使用盐值和加密数据解密文件。

17.2 Java非对称加密

实例454 RSA服务端加密

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\17\Ex17_454

实例说明

本实例使用RSA算法在服务端把“客户端你好，我是服务端”字符串加密以后传输给客户端，服务端要分别生成公钥、密文和数字签名，并把这3种数据分别传输给客户端。本实例把公钥保存在keyPublicData.dat文件中，把密文保存在fileServerData.dat文件中，把数字签名保存在fileSignData.dat文件中，当客户端得到这3个文件以后才能正常对密文解密，解密以后得到的明文就是字符串“客户端你好，我是服务端”。运行效果如图17.7所示。

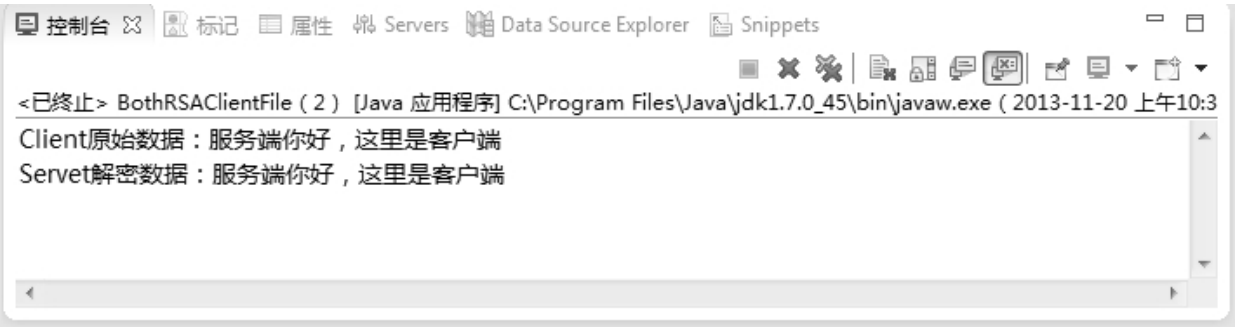


图17.7 服务端加密数据以后客户端解密

技术要点

使用KeyFactory类的generatePrivate()方法可以获取私钥对象，语法如下：

public final PrivateKey generatePrivate(KeySpec keySpec) throws InvalidKeySpecException

参数说明

keySpec：表示根据提供的密钥材料生成私钥对象。

实现过程

（1）新建两个JAVA文件，一个用于服务端加密数据，另一个用于客户端解密数据。

（2）在服务端生成公钥和私钥，公钥保存在keyPublicData.dat文件中，并传送给客户端使用；私钥保存在keyPrivateData.dat文件中，留在本地使用。

（3）服务端使用私钥数据对“客户端你好，我是服务端”加密，并且生成密文保存在fileServerData.dat 文件中。然后使用密文和keyPrivateData.dat文件中的私钥生成签名，把签名保存在fileSignData.dat文件中。

（4）服务端将密文数据fileServerData.dat文件和签名数据fileSignData.dat文件发送给客户端，此时服务端的操作就完成了。

（5）客户端分两次得到服务端传送来的数据，第一次接收到的是公钥文件keyPublicData.dat；第二次接收到的是密文数据fileServerData.dat文件和签名数据fileSignData.dat文件。当客户端接收到文件以后，首先进行签名验证。

（6）客户端签名验证通过后，使用公钥和密文进行解密，最后得到明文“客户端你好，我是服务端”。代码如下：

```
String data ="客户端你好，我是服务端";
//服务端操作
BothRSAServerFile bothRSAServerFile = new BothRSAServerFile();
//生成密钥对
bothRSAServerFile.generateKeyFile();
//加密明文
```

```
    bothRSAServerFile.encryptByPrivateKey(data.getBytes());
    //生成签名
    bothRSAServerFile.generateSign();
    //客户端操作
    BothRSAClientFile bothRSAClientFile = new
BothRSAClientFile();
    byte[] data1 =null;
    if(bothRSAClientFile.verifySign()){
      data1= bothRSAClientFile.decryptByPublicKey();
    }
    System.out.println("Servet原始数据："+data);
    System.out.println("Client解密数据："+new String
(data1));
```

举一反三

根据本实例，读者可以实现以下功能。

使用公钥和密文进行解密。

使用RSA算法传输文件。

实例455 RSA客户端加密

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\17\Ex17_455

实例说明

RSA服务端加密以后再把数据传送给客户端解密是一个比较烦琐的过程，需要生成数字签名、验证数字签名等，客户端如果回传服务端的数据，则较简单。本实例演示如何由客户端加密数据、服务端解密数据，运行效果如图17.8所示。

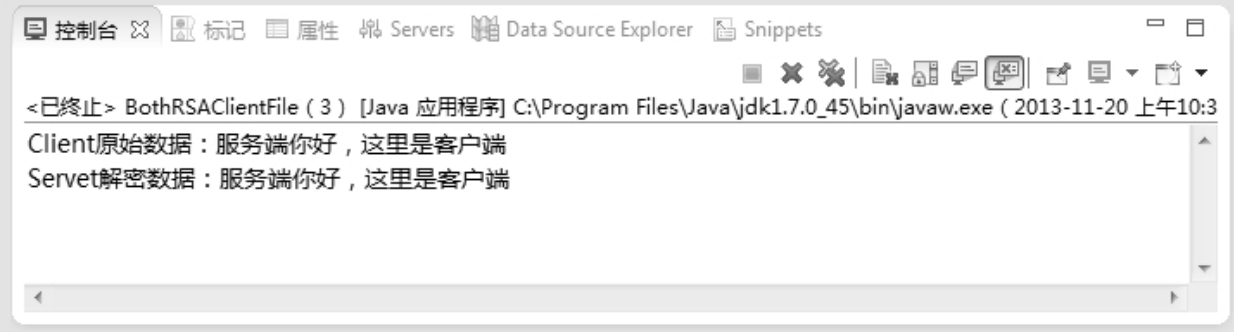


图17.8 客户端加密、服务端解密

技术要点

使用KeyFactory类的generatePublic()方法可以获取公钥对象，语法如下：

public final PublicKey generatePublic(KeySpec keySpec) throws InvalidKeySpecException

参数说明

keySpec：表示根据提供的密钥材料生成公钥对象。

实现过程

（1）新建两个JAVA文件，一个用于客户端加密数据，另一个用于服务端解密数据。

（2）客户端用服务端传来的公钥对“服务端你好，这里是客户端”进行加密，并把密文保存在fileClientData.dat文件中。服务端接收到fileClientData.dat文件后，根据私钥解密，最后得到明文“服务端你好，这里是客户端”。代码如下：

```
//客户端操作
BothRSAClientFile bothRSAClientFile = new BothRSAClientFile();
BothRSAServerFile bothRSAServerFile = new BothRSAServerFile();
String cdata ="服务端你好，这里是客户端";
//获取密文数据
```

```
bothRSAClientFile.encryptByPublicKey(cdata.getBytes());
//客户端解密数据
byte [] cdata1 = bothRSAServerFile.decryptByPrivateKey();
System.out.println("Client原始数据："+cdata);
System.out.println("Servet解密数据："+new String
(cdata1));
```

举一反三

根据本实例，读者可以实现以下功能。

生成数字签名。

验证数字签名。

实例456 DH服务端加密

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\17\Ex17_456

实例说明

DH 加密要先由服务端生成密钥对，并把公钥传给客户端，客户端根据服务端的公钥再生成一对密钥，然后把客户端的公钥传给服务端。这时服务端根据客户端的公钥和自己的私钥生成另一个密钥，这里称为机密密钥。服务端根据机密密钥对明文进行加密，然后把密文传给客户端。客户端根据服务端的密钥和自己的私钥生成客户端的机密密钥，然后再根据自己的机密密钥进行解密。本实例使用DH加密算法，在服务端把字符串“客户端你好，我是服务端”加密以后保存在fileServerData.dat文件中发送给客户端，然后由客户端解密，在控制台打印出来。运行效果如图17.9所示。

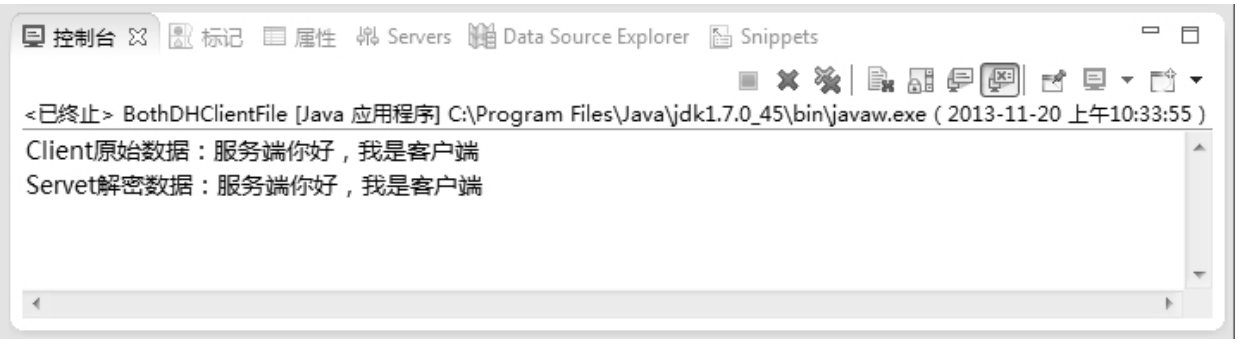


图17.9 服务端加密客户端解密

技术要点

使用KeyAgreement类的init()方法可以用给定的密钥初始化密钥协定，语法如下：

```
public final void init(Key key) throws
InvalidKeyException
```

参数说明

key：表示参与者的私有信息。

实现过程

（1）新建两个JAVA文件，一个用于服务端加密数据，另一个用于客户端解密数据。

（2）在服务端创建 generateServerKeyFile()方法生成密钥对，公钥保存在 keyServerPublic Data.dat文件中，私钥保存在keyServerPrivateData.dat文件中。代码如下：

```
public void generateServerKeyFile() {
  KeyPairGenerator keyPairGen = null;
  try{
    //生成服务端密钥对
    keyPairGen =
KeyPairGenerator.getInstance(keyAlgorithm);
  } catch (NoSuchAlgorithmException e) {
    e.printStackTrace();
```

```
    }
    KeyPair keyPair = keyPairGen.generateKeyPair();
    //公钥
    PublicKey publicKey = keyPair.getPublic();
    writeFile(publicKey.getEncoded(), publicServerkeyFile);
    //私钥
    PrivateKey privateKey = keyPair.getPrivate();
    writeFile(privateKey.getEncoded(),
  privateServerkeyFile);
  }
```

（3）服务端把 keyServerPublicData.dat 文件传给客户端，客户端生成公钥后保存在keyClientPublicData.dat文件中，再把客户端公钥传给服务端。代码如下：

```
  public void generateClientKeyFile() {
    KeyPairGenerator keyPairGen = null;
    try{
      //获取服务户端公钥
      byte[] publicServerkey =
  readFile(publicServerkeyFile);
      X509EncodedKeySpec x509EncodedKeySpec = new
  X509EncodedKeySpec(publicServerkey);
      KeyFactory keyFactory =
  KeyFactory.getInstance(keyAlgorithm);
      PublicKey publicKey =
  keyFactory.generatePublic(x509EncodedKeySpec);
      DHParameterSpec dhParameterSpec = ((DHPublicKey)
  publicKey).getParams();
```

```
            //生成客户端密钥对
            keyPairGen =
        KeyPairGenerator.getInstance(keyFactory.getAlgorithm());
            keyPairGen.initialize(dhParameterSpec);
          } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
          } catch (InvalidKeySpecException e) {
            e.printStackTrace();
          } catch (InvalidAlgorithmParameterException e) {
            e.printStackTrace();
          }
          KeyPair keyPair = keyPairGen.generateKeyPair();
          //公钥
          PublicKey publicKey = keyPair.getPublic();
          writeFile(publicKey.getEncoded(), publicClientkeyFile);
          //私钥
          PrivateKey privateKey = keyPair.getPrivate();
          writeFile(privateKey.getEncoded(),
      privateClientkeyFile);
        }
```

（4）在服务端创建getServerSecretKey()方法。因为DH服务端的加密并不只是使用自己的私钥，而是根据自己的私钥和客户端的公钥生成一个机密密钥，再使用该机密密钥进行加密，所以DH服务端获取到自己的私钥和客户端公钥的二进制文件以后，使用客户端公钥构造一个X509EncodedKeySpec 对象，KeyFactory 密钥工厂可以根据该对象生成一个公钥对象，然后使用自己的私钥构造一个

PKCS8EncodedKeySpec 对象，KeyFactory 密钥工厂可以根据该对象生成一个私钥对象。

生成私钥对象和公钥对象后，就要根据这两个密钥生成一个机密密钥。生成机密密钥前，先使用 DH 算法创建一个 KeyAgreement 密钥协定，然后使用私钥初始化该密钥协定，再把公钥传递给密钥协定的doPhase()方法，最后调用KeyAgreement的generateSecret()方法生成机密密钥。代码如下：

```
    private SecretKey getServerSecretKey() {
      byte[] privateServerKey =
  readFile(privateServerkeyFile);
      byte[] publicClientKey = readFile(publicClientkeyFile);
      X509EncodedKeySpec x509KeySpec = new
  X509EncodedKeySpec(publicClientKey);
      PKCS8EncodedKeySpec pkcs8KeySpec = new
  PKCS8EncodedKeySpec(privateServerKey);
      Key publicKey = null;
      KeyFactory keyFactory = null;
      Key privateKey = null;
      KeyAgreement keyAgree = null;
      try{
        keyFactory=KeyFactory.getInstance(keyAlgorithm);
        publicKey = keyFactory.generatePublic(x509KeySpec);
        privateKey =
    keyFactory.generatePrivate(pkcs8KeySpec);
        //创建密钥协议
        keyAgree =
    KeyAgreement.getInstance(keyFactory.getAlgorithm());
```

```
        //初始化密钥
        keyAgree.init(privateKey);
        keyAgree.doPhase(publicKey, true);
        //生成服务端机密密钥
        return keyAgree.generateSecret(secretAlgorithm);
      } catch (InvalidKeyException e) {
        e.printStackTrace();
      } catch (IllegalStateException e) {
        e.printStackTrace();
      } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
      } catch (InvalidKeySpecException e) {
        e.printStackTrace();
      }
      return null;
    }
```

（5）服务端创建 encryptForServer()方法，根据机密密钥把数据进行加密，生成密文文件fileServerData.dat并传给客户端。代码如下：

```
    public void encryptForServer(byte[] data) {
      //获取机密密钥
      SecretKey secretKey = getServerSecretKey();
      try{
        //加密数据，然后保存到文件中
        Cipher cipher =
    Cipher.getInstance(secretKey.getAlgorithm());
        cipher.init(Cipher.ENCRYPT_MODE, secretKey);
```

```
            writeFile(cipher.doFinal(data), serverdataFile);
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        } catch (NoSuchPaddingException e) {
            e.printStackTrace();
        } catch (InvalidKeyException e) {
            e.printStackTrace();
        } catch (IllegalBlockSizeException e) {
            e.printStackTrace();
        } catch (BadPaddingException e) {
            e.printStackTrace();
        }
    }
```

（6）客户端创建 getClientSecretKey()方法，使用服务端的公钥和自己的私钥生成客户端的机密密钥，并使用该机密密钥进行解密。代码如下：

```
    public byte[] decryptForClient() {
        //获取机密密钥
        SecretKey secretKey = getClientSecretKey();
        try{
            //读取文件，然后解密数据
            byte[] data = readFile(serverdataFile);
            Cipher cipher =
    Cipher.getInstance(secretKey.getAlgorithm());
            cipher.init(Cipher.DECRYPT_MODE, secretKey);
            return cipher.doFinal(data);
        } catch (NoSuchAlgorithmException e) {
```

```
            e.printStackTrace();
        } catch (NoSuchPaddingException e) {
            e.printStackTrace();
        } catch (InvalidKeyException e) {
            e.printStackTrace();
        } catch (IllegalBlockSizeException e) {
            e.printStackTrace();
        } catch (BadPaddingException e) {
            e.printStackTrace();
        }
        return null;
    }
```

举一反三

根据本实例，读者可以实现以下功能。

用DH在服务端进行加密。

客户端实现解密。

实例457 DH客户端加密

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\17\Ex17_457

实例说明

DH 的加密算法里，客户端也要生成密钥对，然后通过服务端传来的公钥与自己的私钥共同生成客户端的机密密钥，再使用机密密钥对明文进行加密。本实例中，客户端把字符串“服务端你好，我是客户端”加密以后保存在fileClientData.dat文件中发送给服务端，然后由服务端解密以后在控制台打印出来。运行效果如图17.10所示。

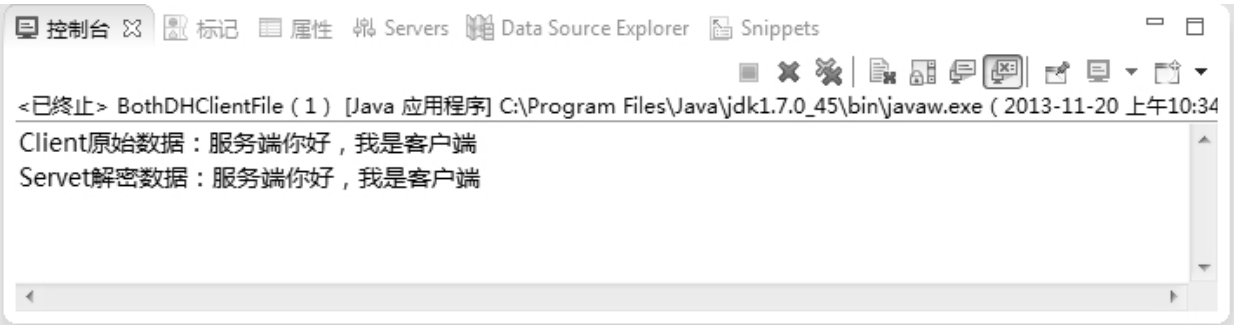


图17.10 客户端加密服务端解密

技术要点

使用KeyAgreement类的doPhase()方法可以给定密钥执行此密钥协定的下一个阶段，语法如下：

public final Key doPhase(Key key, boolean lastPhase) throws InvalidKeyException,

参数说明

- key：表示此阶段的密钥。
- lastPhase：表示是否是此密钥协定最后阶段的标志。

实现过程

（1）新建两个JAVA文件，一个用于客户端加密数据，一个用于服务端解密数据。

（2）DH 算法中，客户端的机密密钥、数据加密、数据解密都与服务端原理相同，唯一不同的是DH的客户端生成密钥比较特别，需要先获取服务端的公钥，用服务端的公钥生成客户端的密钥对。

创建generateClientKeyFile()方法从服务端处获取公钥文件keyServerPublicData.dat，然后通过服务端的公钥生成客户端的密钥对。生成客户端密钥对时，获取服务端公钥以后使用X509EncodedKeySpec规范和KeyFactory生成PublicKey，再把PublicKey转换成DHPublicKey对象，然后通过 DHPublicKey 的getParams()方法得到 DHParameterSpec 实例。密钥对使用KeyPairGenerator 生成，客户端的密钥对在生成前使用

DHParameterSpec 实例初始化密钥生成器，再分别使用密钥生成器的getPublic()和 getPrivate()方法得到密钥对。最后把公钥保存在keyClientPublicData.dat文件中，把私钥保存在keyClientPrivateData.dat文件中。代码如下：

```
    public void generateClientKeyFile() {
      KeyPairGenerator keyPairGen = null;
      try{
        //读取服务端公钥
        byte[] publicServerkey =
    readFile(publicServerkeyFile);
        X509EncodedKeySpec x509EncodedKeySpec = new
    X509EncodedKeySpec(publicServerkey);
        KeyFactory keyFactory =
    KeyFactory.getInstance(keyAlgorithm);
        //生成公钥对象
        PublicKey publicKey =
    keyFactory.generatePublic(x509EncodedKeySpec);
        DHParameterSpec dhParameterSpec = ((DHPublicKey)
    publicKey).getParams();
        keyPairGen =
    KeyPairGenerator.getInstance(keyFactory.getAlgorithm());
        keyPairGen.initialize(dhParameterSpec);
      } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
      } catch (InvalidKeySpecException e) {
        e.printStackTrace();
      } catch (InvalidAlgorithmParameterException e) {
```

```
            e.printStackTrace();
        }
        //生成客户端密钥对
        KeyPair keyPair = keyPairGen.generateKeyPair();
        //公钥
        PublicKey publicKey = keyPair.getPublic();
        writeFile(publicKey.getEncoded(), publicClientkeyFile);
        //私钥
        PrivateKey privateKey = keyPair.getPrivate();
        writeFile(privateKey.getEncoded(),
    privateClientkeyFile);
      }
```

（3）创建 getClientSecretKey()方法，使用服务端的公钥与自己的私钥生成客户端的机密密钥。代码如下：

```
    private SecretKey getClientSecretKey() {
        byte[] privateClientKey =
    readFile(privateClientkeyFile);
        byte[] publicServerKey = readFile(publicServerkeyFile);
        //创建X509EncodedKeySpec实例
        X509EncodedKeySpec x509KeySpec = new
    X509EncodedKeySpec(publicServerKey);
        //创建PKCS8EncodedKeySpec实例
        PKCS8EncodedKeySpec pkcs8KeySpec = new
    PKCS8EncodedKeySpec(privateClientKey);
        PublicKey publicKey = null;
        KeyFactory keyFactory = null;
        Key privateKey = null;
```

```
    KeyAgreement keyAgree = null;
    try{
      keyFactory=KeyFactory.getInstance(keyAlgorithm);
      publicKey = keyFactory.generatePublic(x509KeySpec);
      privateKey =
  keyFactory.generatePrivate(pkcs8KeySpec);
      //创建密钥协议，生成客户端机密密钥
      keyAgree =
  KeyAgreement.getInstance(keyFactory.getAlgorithm());
      keyAgree.init(privateKey);
      keyAgree.doPhase(publicKey, true);
      return keyAgree.generateSecret(secretAlgorithm);
    } catch (InvalidKeyException e) {
      e.printStackTrace();
    } catch (IllegalStateException e) {
      e.printStackTrace();
    } catch (NoSuchAlgorithmException e) {
      e.printStackTrace();
    } catch (InvalidKeySpecException e) {
      e.printStackTrace();
    }
    return null;
  }
```

（4）客户端使用本地的机密密钥进行加密，并把密文保存在fileClientData.dat文件中传给服务端。代码如下：

```
  public void encryptForClient(byte[] data) {
    //获取机密密钥
```

```
    SecretKey secretKey = getClientSecretKey();
    try{
      //客户端数据加密
      Cipher cipher =
  Cipher.getInstance(secretKey.getAlgorithm());
      cipher.init(Cipher.ENCRYPT_MODE, secretKey);
      writeFile(cipher.doFinal(data), clientdataFile);
    } catch (NoSuchAlgorithmException e) {
      e.printStackTrace();
    } catch (NoSuchPaddingException e) {
      e.printStackTrace();
    } catch (InvalidKeyException e) {
      e.printStackTrace();
    } catch (IllegalBlockSizeException e) {
      e.printStackTrace();
    } catch (BadPaddingException e) {
      e.printStackTrace();
    }
  }
```

（5）服务端接收到 fileClientData.dat 文件后，使用自己的机密密钥进行解密，并把解密内容打印在控制台上。代码如下：

```
  public byte[] decryptForServer() {
    //获取机密密钥
    SecretKey secretKey = getServerSecretKey();
    try{
      //读取密文
      byte[] data = readFile(clientdataFile);
```

```
    //服务端数据解密
    Cipher cipher =
Cipher.getInstance(secretKey.getAlgorithm());
    cipher.init(Cipher.DECRYPT_MODE, secretKey);
    return cipher.doFinal(data);
  } catch (NoSuchAlgorithmException e) {
    e.printStackTrace();
  } catch (NoSuchPaddingException e) {
    e.printStackTrace();
  } catch (InvalidKeyException e) {
    e.printStackTrace();
  } catch (IllegalBlockSizeException e) {
    e.printStackTrace();
  } catch (BadPaddingException e) {
    e.printStackTrace();
  }
  return null;
}
```

举一反三

根据本实例，读者可以实现以下功能。

在客户端实现本地加密。

在服务端进行解密。

17.3 Java单项加密

实例458 使用MD5加密

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\17\Ex17_458

实例说明

MD5加密是一种单项加密技术，加密以后生成固定的byte[]类型的数据，这种数据不能使用某一种方法恢复，可以用于用户的密码加密，如一般软件的登录用户的密码都是使用 MD5加密的，用户注册时，把用户密码进行加密后保存起来，每次用户登录时输入的密码都会被加密一次，再把加密后的密码与保存在数据库中的密码进行比较验证，通过验证才允许登录，这样把byte[]类型的数据转换成16进制的字符串更方便操作，如使用MD5对“明日科技”加密以后，密文为ff1b82e719afc4f874b9f47a0b52b666，运行效果如图17.11所示。

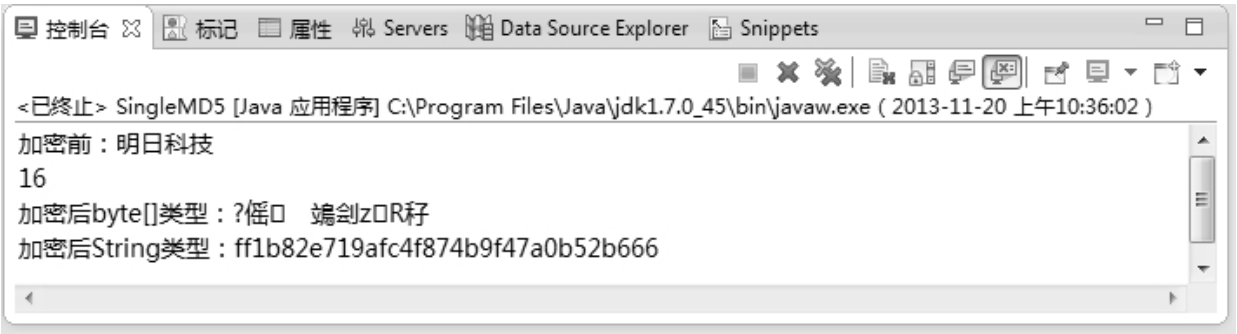


图17.11 使用MD5 对“明日科技”加密

技术要点

使用MessageDigest类的getInstance()方法可以创建一个MessageDigest实例，语法如下：

public static MessageDigest getInstance(String algorithm) throws NoSuchAlgorithmException

参数说明

algorithm：表示指定算法名称的信息摘要。

实现过程

（1）新建一个JAVA文件。

（2）创建encryptMD5()方法，指定加密算法为MD5加密，对明文数据进行加密，代码如下：

```
public static byte[] encryptMD5(byte[] data) throws NoSuchAlgorithmException {
    //指定加密算法
    MessageDigest digest = MessageDigest.getInstance(algorithm);
    //进行加密
    digest.update(data);
    return digest.digest();
}
```

（3）创建encryptMD5toString()方法把byte[]类型的数据转换成16进制的字符串，具体代码如下：

```
public static String encryptMD5toString(byte[] data) throws NoSuchAlgorithmException {
    String str ="";
    String str16;
    for (int i = 0; i < data.length; i++) {
      //转换为16进制数据
      str16 = Integer.toHexString(0xFF & data[i]);
      //如果长度为1，前位补0
```

```
        if (str16.length() == 1) {
          str = str +"0"+ str16;
        } else {
          str = str + str16;
        }
      }
      return str;
    }
```

举一反三

根据本实例，读者可以实现以下功能。

使用toHexString()方法转换字符。

使用MD5加密整个文件。

实例459 使用Hmac加密

这是一个可以用来提高基础性能的实例

实例位置：光盘\mingrisoft\17\Ex17_459

实例说明

Hmac 加密可以用于数据传输的确认，如用户需要做一次投票，在投票之前服务端先生成一个密钥，然后把密钥发送给客户端，与此同时服务端使用该密钥对投票的题目进行加密，生成一个信息摘要。当客户端得到密钥以后也使用这个密钥对投票的题目进行加密，生成信息摘要，并把这个信息摘要和投票结果发给服务端。服务端收到以后，把客户端的信息摘要与服务端的信息摘要进行比较，如果两者一致则认为是有效的投票。

本实例中，使用两个类模拟这种客户端与服务端的情况，都对“明日科技”进行加密，如果经比较一致，则输出“验证通过”；否

则，输出“验证不通过”，运行效果如图17.12所示。

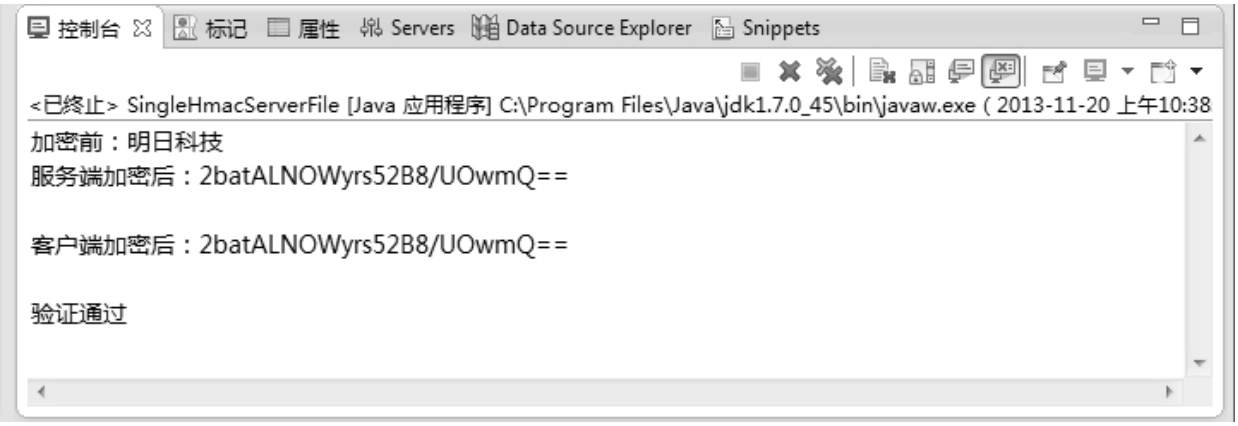


图17.12 比较Hmac 摘要信息

技术要点

使用SecretKeySpec类的构造函数可以根据密钥文件生成实例，语法如下：

```
public SecretKeySpec(byte[] key, String algorithm)
```

参数说明

- key：密钥的内容，复制该数组的内容来防止后续修改。
- algorithm：表示密钥内容相关联的密钥算法的名称。

实现过程

（1）新建两个JAVA文件。一个文件用于服务器端加密，另一个文件用于客户端加密。

（2）在服务端创建initMacKey()方法生成随机密钥，然后把密钥保存在keyData.bat文件里。代码如下：

```
public void initMacKey() throws NoSuchAlgorithmException
{
    //生成密钥生成器
    KeyGenerator generator =
  KeyGenerator.getInstance(algorithm);
    //生成密钥
```

```
        SecretKey key = generator.generateKey();
        writeFile(key.getEncoded(), keyFile);
    }
```

（3）分别在服务端和客户端创建encryptHMAC()方法加密原文，然后生成byte[]类型的数据摘要。代码如下：

```
    public byte[] encryptHMAC(byte[] data) throws
NoSuchAlgorithmException,InvalidKeyException {
        //读取密钥
        byte key[] 文件= readFile(keyFile);
        SecretKey secretKey = new SecretKeySpec(key,
  algorithm);
        //进行加密操作
        Mac mac = Mac.getInstance(secretKey.getAlgorithm());
        mac.init(secretKey);
        return mac.doFinal();
    }
```

（4）对服务端和客户端产生的数据摘要使用BASE64进行加密转成String类型，然后对二者进行比较，如果相同，则打印“验证通过”；否则，打印“验证不通过”。代码如下：

```
    public static void main(String[] avg) throws
NoSuchAlgorithmException, InvalidKeyException {
        SingleHmacServerFile singleHmacServerFile = new
  SingleHmacServerFile();
        SingleHmacClientFile singleHmacClientFile = new
  SingleHmacClientFile();
        String data ="明日科技";
        System.out.println("加密前：" + data);
```

```
    String strData = null;
    String strDataClient = null;
    //生成密钥
    singleHmacServerFile.initMacKey();
    //服务器端加密
    strData =
BothBase64.encryptBASE64(singleHmacServerFile.encryptHMAC(d
ata.getBytes()));
    //客户端加密
    strDataClient =
BothBase64.encryptBASE64(singleHmacClientFile.encryptHMAC(d
ata.getBytes()));
    System.out.println("服务端加密后："+ strData);
    System.out.println("客户端加密后："+ strDataClient);
    //比较加密结果
    if (strData.equals(strDataClient)) {
      System.out.println("验证通过");
    } else {
      System.out.println("验证不通过");
    }
  }
```

举一反三

根据本实例，读者可以实现以下功能。

Hmac加密数据传输。

检测是否有Hmac加密。

实例460 使用DSA加密

这是一个可以提高基础性能的实例

实例位置：光盘\mingrisoft\17\Ex17_460

实例说明

DSA的加密适合验证在从服务端向客户端传递数据的过程中，数据是否被第三者修改过。如果把服务端比作老师，把客户端比作家长，老师要把学生成绩单拿给家长看，那么成绩单就可以看作是服务端向客户端传递的数据。老师一般不会直接把成绩单交给家长，而是要通过学生转交，成绩单是公开的，任何人都可以看但不能修改，当家长看到成绩单时，不知道学生传递过程中有没有修改过，这就需要使用一种办法来验证，这就是DSA加密。本实例对字符串“明日科技”进行加密，然后验证其有效性，运行效果如图17.13所示。

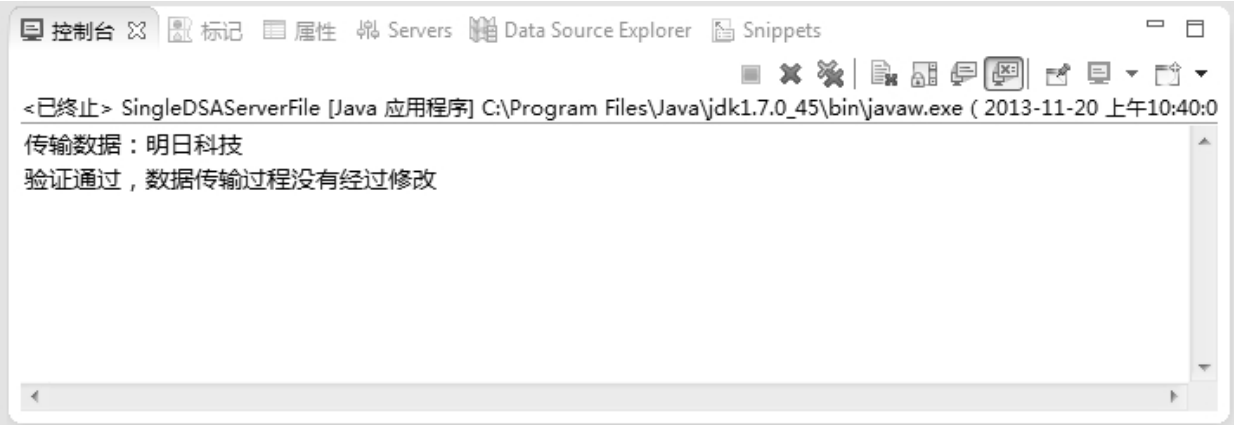


图17.13 DAS 验证数据传输

技术要点

使用 Signature 类的 verify()方法可以验证签名字节及传输数据是否经过修改，如果没有经过修改，方法返回true，表示验证通过；如果经过修改，则返回false，表示验证不通过。语法如下：

public final boolean verify(byte[] signature) throws SignatureException

参数说明

signature：表示要验证的签名字节。

实现过程

（1）新建两个JAVA文件。一个用于服务端加密，一个用于客户端验证。

（2）DSA加密需要生成密钥对，其中，公钥传给客户端，私钥在服务端用来生成数字签名。在服务端生成密钥对后，把公钥保存在keyPublicData.dat 文件中传给客户端，私钥保存在keyPrivateData.dat文件中自己使用。

（3）在服务端创建 generatorSign()方法，使用公钥对要传送的数据生成签名。生成数字签名需要使用私钥数据和要传输的数据。使用PKCS8EncodedKeySpec类和KeyFactory类把二进制私钥数据转化成私钥对象，然后使用私钥对象对Signature进行初始化，使用sign()方法得到数字签名，并把数字签名保存在fileSignData.dat文件中，最后把签名文件和字符串“明日科技”直接发送给客户端，代码如下：

```
    public void generatorSign(byte[] data) throws
NoSuchAlgorithmException,
InvalidKeySpecException,InvalidKeyException,
SignatureException {
        //读取私钥
        byte[] privateKey = readFile(privatekeyFile);
        PKCS8EncodedKeySpec pkcs8KeySpec = new
  PKCS8EncodedKeySpec(privateKey);
        // algorithm 指定的加密算法
        KeyFactory keyFactory = null;
        PrivateKey priKey = null;
        keyFactory = KeyFactory.getInstance(algorithm);
        priKey = keyFactory.generatePrivate(pkcs8KeySpec);
        //生成数字签名
```

```
        Signature signature =
    Signature.getInstance(keyFactory.getAlgorithm());
        signature.initSign(priKey);
        signature.update(data);
        writeFile(signature.sign(), signdataFile);
      }
```

（4）把要传输的数据和数字签名发送给客户端，在客户端创建verifySign()方法，先使用X509EncodedKeySpec和KeyFactory把公钥的二进制数据转换成公钥对象，再使用公钥对象初始化Signature 的验证，然后对接收到的字符串“明日科技”进行验证，如果Signature类的verify()方法返回true，表示数据没有被修改过；否则数据被修改过，代码如下：

```
    public boolean verifySign(byte[] data) {
        //获取公钥数据
        byte[] key = readFile(publickeyFile);
        //获取签名
        byte[] sign = readFile(signdataFile);
        X509EncodedKeySpec keySpec = new
    X509EncodedKeySpec(key);
        KeyFactory keyFactory = null;
        PublicKey publicKey = null;
        try{
          //获取公钥
          keyFactory = KeyFactory.getInstance(algorithm);
          publicKey = keyFactory.generatePublic(keySpec);
        } catch (InvalidKeySpecException e) {
          e.printStackTrace();
```

```
    } catch (NoSuchAlgorithmException e) {
      e.printStackTrace();
    }
    try{
      //验证数字签名
      Signature signature =
Signature.getInstance(keyFactory.getAlgorithm());
      signature.initVerify(publicKey);
      signature.update(data);
      return signature.verify(sign);
    } catch (NoSuchAlgorithmException e) {
      e.printStackTrace();
    } catch (InvalidKeyException e) {
      e.printStackTrace();
    } catch (SignatureException e) {
      e.printStackTrace();
    }
    return false;
}
```

举一反三

根据本实例，读者可以实现以下功能。

使用DSA加密验证文件。

检测是否有DSA加密。

第18章 游戏开发

益智小游戏

休闲小游戏

其他

18.1 益智小游戏

实例461 图片配对游戏

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\18\Ex18_461

实例说明

本实例通过为标签控件添加图标以及鼠标事件，完成图片配对游戏。运行程序，初始显示效果如图18.1所示，用鼠标拖动窗体上显示图标的标签到相应的文字标签上，如果配对正确，则下面的文字标签显示匹配成功的信息和图片，效果如图18.2所示。

图18.1 图片配对之前的效果

图18.2 图片配对成功的效果

技术要点

本实例通过玻璃面板、鼠标事件以及判断控件是否在其他控件的范围内，完成图片配对游戏的开发。

（1）玻璃面板位于 JRootPane 中所有其他组件之上，这为在所有其他组件上绘图和截取鼠标事件提供了方便，对拖动和绘图都非常有用。因此开发人员可在玻璃面板上使用 setVisible()方法，控制玻璃面板在所有其他子级控件上是否显示，默认情况下，玻璃面板是不可见的，可以通过JFrame类的getGlassPane()方法获得玻璃面板对象，该方法的定义如下：

public Component getGlassPane()

参数说明

返回值：此窗体的玻璃面板对象，该对象是Component类型的容器。

（2）通过使窗体类实现MouseListener和MouseMotionListener接口，实现鼠标事件，完成鼠标按下、释放和拖动等事件的处理。

（3）本实例定义了一个 checkPosition()方法，用于检查所有拖动的控件是否匹配成功，以及是否在下面显示文字的控件内，该方法的代码如下：

```
    private boolean checkPosition()
{                                    //检查配对是否正确
      boolean result = true;
      for (int i = 0; i < 3; i++) {
        Point location=
    img[i].getLocationOnScreen();              //获取每个图像
    标签的位置
        Point seat=
    targets[i].getLocationOnScreen();                  //获取每
    个对应位置的坐标
        targets[i].setBackground(Color.GREEN);
        //设置匹配后的颜色
        //如果配对错误
        if (location.x < seat.x || location.y < seat.y||
    location.x > seat.x + 80 || location.y > seat.y + 80) {
          targets[i].setBackground(Color.ORANGE);
        //回复对应位置的颜色
          result=
      false;                                    //检测结果为
      false
        }
      }
      return
  result;                                       //返回检测
```

结果

}

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 PictureMatchingFrame 窗体类，该类实现了MouseListener和MouseMotionListener接口，因此可以响应鼠标事件。

（3）在PictureMatchingFrame窗体类的声明区，声明如下成员：

```
private JLabel img[]=new JLabel[3];                    //显示图标的标签
private JLabel targets[]=new JLabel[3];                //窗体下面显示文字的标签
private Point pressPoint;                              //鼠标按下时的起始坐标
```

（4）在PictureMatchingFrame窗体类的构造方法中，初始化界面，关键代码如下：

```
for (int i = 0; i < 3; i++) {
  img[i]=new JLabel(icon[i]);                          //创建图像标签
  img[i].setSize(50, 50);                              //设置标签大小
  img[i].setBorder(new LineBorder(Color.GRAY));        //设置线性边框
  int x= (int) (Math.random() * (getWidth() - 50));    //随机生成X坐标
```

```
    int y= (int) (Math.random() * (getHeight() -
150));                //随机生成Y坐标
    img[i].setLocation(x,y);
    //设置随机坐标
    img[i].addMouseListener(this);
      //为每个图像标签添加鼠标事件监听器
    img[i].addMouseMotionListener(this);
    imagePanel.add(img[i]);
  //添加图像标签到图像面板
    targets[i]=new
JLabel();                                        //创建匹配位置标
签
    targets[i].setOpaque(true);
    //使标签不透明，以设置背景色
    targets[i].setBackground(Color.ORANGE);
      //设置标签背景色
    targets[i].setHorizontalTextPosition(SwingConstants.CEN
TER);       //设置文本与图像水平居中
    targets[i].setVerticalTextPosition(SwingConstants.BOTTO
M);         //设置文本显示在图像下方
    targets[i].setPreferredSize(newDimension(80,80));
        //设置标签首选大小
    targets[i].setHorizontalAlignment(SwingConstants.CENTER
);         //文字居中对齐
    bottomPanel.add(targets[i]);
      //添加标签到底部面板
  }
```

```
    targets[0].setText("显示
器");                                    //设置匹配位置的文本
    targets[1].setText("衣服");
    targets[2].setText("自行车");
```

（5）鼠标按下事件用于获得拖放图片标签时的起始坐标，代码如下：

```
    public void mousePressed(MouseEvent e) {
      pressPoint=
  e.getPoint();                                  //保存拖放图
  片标签时的起始坐标
    }
```

（6）鼠标释放事件用于检查图片配对是否正确，如果正确，则隐藏玻璃面板，并在下面的文字标签上显示图片和匹配成功的文字，代码如下：

```
    public void mouseReleased(MouseEvent e) {
      if (checkPosition())
  {                                          //如果配对正确
        getGlassPane().setVisible(false);
        for (int i=0; i<3; i++)
    {                                  //遍历所有匹配位置的标签
          targets[i].setText("匹配成
      功");                          //设置正确提示
          targets[i].setIcon(img[i].getIcon());
            //设置匹配的图标
        }
      }
    }
```

（7）鼠标拖动事件用于设置控件新的位置，代码如下：

```
  public void mouseDragged(MouseEvent e) {
    JLabel source= (JLabel)
e.getSource();                              //获取事件源控件
    Point imgPoint=
source.getLocation();                              //获取控件坐标
    Point point=
e.getPoint();                                  //获取鼠标坐
标
    source.setLocation(imgPoint.x+point.x - pressPoint.x,
imgPoint.y+point.y -
pressPoint.y);                              //设置控件新坐标
  }
```

举一反三

根据本实例，读者可以开发以下程序。

游戏推箱子。

实例462 拼图游戏

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\18\Ex18_462

实例说明

本实例实现了拼图游戏的开发。运行程序，单击“开始”按钮将打乱图片的位置，效果如图18.3所示，然后通过鼠标单击图片进行移动，直到将所有图片都移动到正确位置，游戏过关，过关后的效果如图18.4所示。

图18.3 打乱图片位置的效果

图18.4 图片移动到正确位置的效果

技术要点

本程序主要通过Swing与枚举类实现，程序将一幅完整的图片平均分成9部分，每一部分为一个正方形，并将最后一个图片修改为空白图片，作为游戏中的一个空位置。对于每一个图片部分，程序封装了一个按钮对象进行装载，当该按钮对象被单击后，程序将调换该按钮与装载空白图片的按钮，其关键技术是使用枚举类控制方向，以及使用setLocation()方法设置按钮的位置。

（1）本实例通过枚举类定义了图片移动的4个方向，分别为上、下、左、右，其定义方式与定义一个类相似，但定义枚举类使用关键字enmu，枚举类Direction的定义如下：

```
public enum Direction {
  UP,                                                    //
上
  DOWN,                                                  //下
  LEFT,                                                  //
左
  RIGHT                                                  //右
}
```

（2）当图片移动后，按钮的坐标发生改变，此操作通过setLocation()方法实现。setLocation()方法是从Component类继承的，其定义如下：

```
public void setLocation(int x, int y)
```

参数说明

- x：当前控件左上角在父级坐标空间中新位置的x 坐标。
- y：当前控件左上角在父级坐标空间中新位置的y 坐标。

实现过程

（1）新建一个项目。

（2）在项目中创建一个名称为Direction的枚举类，用于定义图片移动的4个方向，其关键代码见关键技术部分。

（3）创建名称为Cell的类，用于封装一个单元图片对象，此类继承JButton对象，并对JButton按钮组件进行重写，其关键代码如下：

```
public class Cell extends JButton {
  private static final long serialVersionUID =
7186231146506578l9L;
  public static final int
IMAGEWIDTH=117;                    //图片宽度
  private int
place;                                          //图片位置
  public Cell(Icon icon, int place) {
    this.setSize(IMAGEWIDTH,
IMAGEWIDTH);                //单元图片的大小
    this.setIcon(icon);
  //单元图片的图标
    this.place=place;
//单元图片的位置
  }
  public void move(Direction dir)
{                              //移动单元图片的方法
    Rectangle rec=
this.getBounds();                          //获取图片的
Rectangle对象
    switch (dir)
{                                          //判断方向
```

```
        case UP:                                        //向
    上移动
          this.setLocation(rec.x, rec.y - IMAGEWIDTH);
          break;
        case DOWN:                                      //向
    下移动
          this.setLocation(rec.x, rec.y + IMAGEWIDTH);
          break;
        case LEFT:                                      //向
    左移动
          this.setLocation(rec.x - IMAGEWIDTH, rec.y);
          break;
        case RIGHT:                                      //向
    右移动
          this.setLocation(rec.x + IMAGEWIDTH, rec.y);
          break;
      }
    }
    public int getX() {
      return
this.getBounds().x;                                    //获取单
元图片的x坐标
    }
    public int getY() {
      return
this.getBounds().y;                                    //获取单
元图片的y坐标
```

```
    }
    public int getPlace() {
      return
  place;                                          //获取单元
  图片的位置
    }
  }
```

（4）创建名称为GamePanel的类，此类继承Jpanel类、实现MouseListener接口，用于创建游戏面板对象。在GamePanel类中定义长度为9单元的图片数组对象，并通过init()方法对所有单元图片对象进行实例化，其关键代码如下：

```
  public class GamePanel extends JPanel implements
MouseListener {
    private static final long serialVersionUID
  =-6538319477834401 22L;
    private Cell[] cells=new
  Cell[9];                                //创建单元图片数组
    private Cell
  cellBlank=null;                                 //空白
    public GamePanel() {
      super();
      setLayout(null);                                    /
  /设置空布局
      init();                                            //
  初始化
    }
    //初始化游戏
```

```
  public void init() {
    int num=0;                                          //
图片序号
    Icon icon=null;                                      //
图标对象
    Cell
cell=null;                                          //单元图
片对象
    for (int i=0; i<3; i++)
{                                  //循环行
      for (int j=0; j<3; j++)
  {                                //循环列
        num= i * 3+ j;                                  //计
    算图片序号
        icon=SwingResourceManager.getIcon(GamePanel.class
    , "/pic/"+ (num+1)+
    ".jpg");                          //获取图片
        cell=new Cell(icon,
    num);                             //实例化单元图片对象
        cell.setLocation(j *Cell.IMAGEWIDTH,
    i*Cell.IMAGEWIDTH);    //设置单元图片的坐标
        cells[num]=
    cell;                                          //将单元图片存
    储到单元图片数组中
      }
    }
    for (int i = 0; i < cells.length; i++) {
```

```
        this.add(cells[i]);
          //向面板中添加所有单元图片
      }
    }
    /**
    *对图片进行随机排序
    */
    public void random() {
      Random rand=new
Random();                                    //实例化Random
      int m, n, x, y;
      if (cellBlank==null)
{                                          //判断空白的图片位置
是否为空
        cellBlank= cells[cells.length -
    1];                                 //取出空白的图片
        for (int i=0; i< cells.length; i++)
    {                                 //遍历所有单元图片
          if (i != cells.length - 1) {
            cells[i].addMouseListener(this);
              //对非空白图片注册鼠标监听
          }
        }
      }
      for (int i=0; i< cells.length; i++)
{                                    //遍历所有单元图片
```

```
        m=
    rand.nextInt(cells.length);
    //产生随机数
        n=
    rand.nextInt(cells.length);
    //产生随机数
        x=
    cells[m].getX();                                    //
    获取x坐标
        y=
    cells[m].getY();                                    //
    获取y坐标
        //对单元图片调换
        cells[m].setLocation(cells[n].getX(),
    cells[n].getY());
        cells[n].setLocation(x, y);
      }
    }
    @Override
    public void mousePressed(MouseEvent e) {
      Cell cell= (Cell)
e.getSource();                              //获取触发时
间的对象
      int x=
cellBlank.getX();                                   //获
取x坐标
```

```
    int y=
cellBlank.getY();                                    //获
取y坐标
    if ((x - cell.getX()) == Cell.IMAGEWIDTH &&
cell.getY() == y) {
      cell.move(Direction.RIGHT);
      //向右移动
      cellBlank.move(Direction.LEFT);
    } else if ((x - cell.getX()) ==-Cell.IMAGEWIDTH &&
cell.getY() == y) {
      cell.move(Direction.LEFT);
        //向左移动
      cellBlank.move(Direction.RIGHT);
    } else if (cell.getX() == x && (cell.getY() - y) ==
Cell.IMAGEWIDTH) {
      cell.move(Direction.UP);
        //向上移动
      cellBlank.move(Direction.DOWN);
    } else if (cell.getX() == x && (cell.getY() - y) ==-
Cell.IMAGEWIDTH) {
      cell.move(Direction.DOWN);
      //向下移动
      cellBlank.move(Direction.UP);
    }
    if (isSuccess())
{                                                    //判断是否拼图
成功
```

```
    int i= JOptionPane.showConfirmDialog(this, "成功,
再来一局？", "拼图成
功",JOptionPane.YES_NO_OPTION);                    //提示
成功
    if (i == JOptionPane.YES_OPTION) {
      random();
  //开始新一局
    }
  }
}
/**
*判断是否拼图成功
*@return 布尔值
*/
public boolean isSuccess() {
  for (int i=0; i< cells.length; i++)
{                              //遍历所有单元图片
    int x=
cells[i].getX();                                    //获
取x坐标
    int y=
cells[i].getY();                                    //获
取y坐标
    if (i != 0) {
      if (y /Cell.IMAGEWIDTH* 3+x /Cell.IMAGEWIDTH !=
  cells[i].getPlace())
```

```
        {                                    //判断单元图片位置是否
        正确
            return false;                    //只要有一
          个单元图片的位置不正确，就返回false
            }
          }
        }
        return true;                         //所有
    单元图片的位置都正确，就返回true
      }
    }
```

举一反三

根据本实例，读者可以开发以下程序。

游戏走出迷宫。

实例463 掷骰子

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\18\Ex18_463

实例说明

本实例开发了一个单机掷骰子游戏，玩家可以模拟下注10元、20元、50元或100元，而且单击完钱数按钮后，可以通过单击“加倍”按钮进行加倍下注，然后还需要单击“确认下注”按钮，才算完成了玩家的下注操作，同时在庄家处显示“跟了”的信息，这时玩家就可以选择压大还是压小，然后骰子开始动画变换，动画结束后将显示消息框进行提示。运行程序，效果如图18.5所示。

图18.5 掷骰子游戏

技术要点

使用线程控制骰子的切换，并将6个骰子图片文件的主文件名用阿拉伯数字命名（如1.png、2.png、3.png、4.png、5.png和6.png），这样命名之后，可以在程序中方便地通过数字来改变图片文件，以实现切换骰子的操作，在切换骰子时使用随机数产生骰子的主文件名编号，改变骰子的图片可以通过如下代码实现：

```
v1=(int)(Math.random()*6+1);
v2=(int)(Math.random()*6+1);
v3=(int)(Math.random()*6+1);
lb_dice_1.setIcon(SwingResourceManager.getIcon(DiceGameFrame.class, "/icon/"+v1+".png"));
lb_dice_2.setIcon(SwingResourceManager.getIcon(DiceGameFrame.class, "/icon/"+v2+".png"));
lb_dice_3.setIcon(SwingResourceManager.getIcon(DiceGameFrame.class, "/icon/"+v3+".png"));
```

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的DiceGameFrame窗体类，在窗体DiceGameFrame中创建内部线程类DiceThread，用于判断骰子的点数并确定玩家赢还是庄家赢，该内部线程类的代码如下：

```
private class DiceThread implements Runnable {
  public void run() {
    while (true) {
      stopIndex++;
      v1 = (int) (Math.random() * 6 + 1); //随机产生第一个骰子的点数
      v2 = (int) (Math.random() * 6 + 1); //随机产生第二个骰子的点数
      v3 = (int) (Math.random() * 6 + 1); //随机产生第三个骰子的点数
      //显示骰子的图片
      lb_dice_1.setIcon(SwingResourceManager.getIcon(DiceGameFrame.class, "/icon/"+ v1 +".png"));
      lb_dice_2.setIcon(SwingResourceManager.getIcon(DiceGameFrame.class, "/icon/"+ v2 +".png"));
      lb_dice_3.setIcon(SwingResourceManager.getIcon(DiceGameFrame.class, "/icon/"+ v3 +".png"));
      int totalValues=v1+v2+v3;          //骰子的点数总和
      if (stopIndex==50) {               //当stopIndex为50时，显示消息框，并提示最终的点数
        if (flag == true) {
```

```
    if (totalValues>9) {          //骰子的点数为大时
显示的提示信息
      JOptionPane.showMessageDialog(null, "点数
  是："+ totalValues +"，大。\n玩家赢！！！\n总钱
  数：人民币"+ totalMoney +"元");
    } else {
      JOptionPane.showMessageDialog(null, "点数
  是："+ totalValues +"，小。\n庄家赢！！！\n总钱
  数：人民币"+ totalMoney +"元");
    }
  } else {
    if (totalValues<=9) {       //骰子的点数为小时显
示的提示信息
      JOptionPane.showMessageDialog(null, "点数
  是："+ totalValues +"，小。\n玩家赢！！！\n总钱
  数：人民币"+ totalMoney +"元");
    } else {
      JOptionPane.showMessageDialog(null, "点数
  是："+ totalValues +"，大。\n庄家赢！！！\n总钱
  数：人民币"+ totalMoney +"元");
    }
  }
  //这里省略了部分代码
  break;
}
try {
  Thread.sleep(20);
```

```
        } catch (Exception ex) {
          System.out.println(flag);
        }
      }
    }
}
```

举一反三

根据本实例，读者可以开发以下程序。

游戏找不同。

18.2 休闲小游戏

实例464 打字母游戏

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\18\Ex18_464

实例说明

本实例开发了一个练习指法的打字母游戏。程序运行后，不断从上向下“掉”苹果，苹果上标有字母，按键盘上对应的字母键，若正确，则该苹果消失；若错误，可以在苹果没有落地之前继续按键，如果正确将得分，如果苹果已经落地，则该字母没有打中，不得分，苹果落地后还会出现新的苹果。运行程序，效果如图18.6所示。

图18.6 打字母游戏

技术要点

随机获得大写字母的ASCII值，然后把ASCII值转换为大写字母字符，再将大写字母字符转换为字符串，这些操作可以通过如下代码来实现：

```
int letter=(int)(Math.random()*26)+65;          //产生65~90之间的随机整数，即大写字母A~Z的ASCII值
char c=(char)letter;                            //将字符的ASCII值转换为字符
String s=String.valueOf(c);                     //将字符转换为字符串
```

实现过程

（1）新建一个项目。

（2）在项目中创建一个类，名称为RandomBuildLetter，调用该类的方法可以随机产生65~90之间的随机整数，用于以后转换成大写字母，该类的代码如下：

```
/**
*@author 张振坤
*产生65~90之间随机整数的类
*/
public class RandomBuildLetter {
  public RandomBuildLetter() {
    super();
  }
  public int[] getLetter(int letterCounts) {
    int[] letter=new int[letterCounts];      //根据参数创建整型数组
```

```
    for (int i = 0; i < letterCounts; i++) {
      int a= (int) getRandomLetter();      //调用方法产生
    65~90之间的随机整数，即大写字母A-Z的ASCII值
      letter[i]= a;                        //将产生的数赋值给
    数组
    }
    return letter;                          //返回数组对象
  }
  public int getRandomLetter() {
    int letter = (int) (Math.random() * 26) + 65; //产生
  65～90之间的随机整数，即大写字母的ASCII值
    return letter;
  }
}
```

（3）在项目中创建一个继承 JFrame 类的主窗体类TypeLetterFrame，并在成员声明区定义标签数组，用于显示随机产生的带字母的苹果，定义RandomBuildLetter类的实例用于获得随机产生的大写字母的ASCII值，定义一个向量对象用于存储准备击打的字母，关键代码如下：

```
private RandomBuildLetter buildLetter = new
RandomBuildLetter();  //创建随机产生字母的类的对象
private JLabel[]
labels=null;                                  //创建标签数组
private Vector<String>vector=new Vector<String>
();              //创建存储准备击打字母的向量
```

（4）在窗体TypeLetterFrame中，创建addLetter()方法，用于实现在窗体上显示字母、计算出现的字母总数等操作，该方法的代码如

下：

```
  private void addLetter() {
    int seed=10;                                          //设
置标签之间偏移量的变量
    //调用RandomBuildLetter类的方法随机产生8个整数并赋值给
数组，即8个A～Z之间字母的ASCII值
    int[] letters = buildLetter.getLetter(8);
    labels=new
JLabel[letters.length];                          //创建显示带字
母苹果的标签数组
    //实例化标签数组的每个对象
    for (int i = 0; i < letters.length; i++) {
      int value= letters[i];                               //
获得数组letters中的值
      char c= (char) value;                              //将
数组letters中的值转换为字符
      String s=String.valueOf(c);                        //将
字符转换为字符串
      labels[i]=new JLabel();                            //实
例化标签对象
      labels[i].setToolTipText(s);                         /
/设置标签的提示文本
      labels[i].setIcon(SwingResourceManager.getIcon(TypeLe
tterFrame.class, "/icon/"+ s+ ".png"));  //设置标签显示的
图片，即带字母的苹果
      int x= (int) (Math.random() * 60)+
seed;             //随机产生标签显示位置的横坐标
```

```
        int y= (int) (Math.random() *
80);                    //随机产生标签显示位置的纵坐标
        labels[i].setBounds(x,y,100,30);                    //
设置标签的显示位置和大小
        backgroundPanel.add(labels[i]);                     //
将标签添加到背景面板中
        seed+=60;                                          //调整标
签之间的偏移量
        vector.add(s);                                      //将
字符串字母添到到向量对象中
        totalLetters++;                                    //计算
出现字母的总个数
    }
}
```

举一反三

根据本实例，读者可以开发以下程序。

游戏快快吃豆。

实例465 画梅花

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\18\Ex18_465

实例说明

本实例开发了一个画梅花的小游戏，运行程序，效果如图18.7所示，在树枝上只有很少的几朵梅花，然后在窗体的左侧选择花的形状，在右侧的树枝上单击即可画梅花，画梅花后的效果如图18.8所示。单击鼠标右键，即可删除绘制的梅花。

技术要点

在实现画梅花程序时，主要应用了JPanel面板的鼠标事件。在画梅花时，捕获到鼠标的左键被按下的事件，并且获取到鼠标指针的位置，在该位置画梅花，即添加一个JLabel控件，并将该组件的 ico 属性设置为所选择的梅花样式；在删除梅花时，捕获到鼠标的右键按下事件，并且获取到鼠标指针所在位置的JLabel控件（即梅花），并将该组件从面板中移除即可。

图18.7 画梅花之前的效果

图18.8 画梅花之后的效果

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的DrawPlumBlossomFrame窗体类。

（3）在DrawPlumBlossomFrame窗体的左侧添加一个Jpanel控件，将其opaque属性设置为false，用于将该JPanel的背景设置为透明，并将窗体的布局设置为null（绝对布局），然后添加4个JLabel控件，用于显示梅花形状。

（4）分别将4个Jlabel控件的variabel属性设置为flower1、flower2、flower3和flower4，并将它们的 ico 属性设置为 images 文件夹中的 flower1.png、flower1.png、flower1.png 和 flower4.png。

（5）在DrawPlumBlossomFrame窗体的右侧添加一个JPanel控件，将其variable属性设置为canvasPane；将其opaque属性设置为false，并将窗体的布局设置为null（绝对布局），以方便在鼠标单击位置处画梅花，然后为该JPanel控件添加鼠标按下事件，并判断如果按下的是鼠标左键，则根据选择的花朵形状，在窗体上画梅花；如果按下的是鼠标右键，则获取鼠标所在位置的梅花，并将其删除，具体代码如下：

```
canvasPane.addMouseListener(new MouseAdapter() {
  public void mousePressed(final MouseEvent e) {
    if (e.getModifiers()== InputEvent.BUTTON1_MASK)
{            //按下鼠标左键
      final JLabel flower=new
  JLabel();                          //显示梅花的标签
      flower.setIcon(SwingResourceManager.getIcon(DrawPlu
  mBlossomFrame.class, "/images/"+ flowerType+".png"));
      if ("flower1".equals(flowerType))
  {                        //设置第一种类型梅花的大小及位
  置
        flower.setBounds(e.getX() - 6, e.getY() - 12, 30,
    36);
      } else if ("flower2".equals(flowerType))
  {                    //设置第二种类型梅花的大小及位置
        flower.setBounds(e.getX() - 28, e.getY() - 30,
    51, 43);
      } else if ("flower3".equals(flowerType))
  {                    //设置第三种类型梅花的大小及位置
```

```
            flower.setBounds(e.getX() - 5, e.getY() - 15, 30,
        23);
            } else
        {                                                    //设置第四
        种类型梅花的大小及位置
            flower.setBounds(e.getX() - 29, e.getY() - 25,
        58, 51);
            }
            canvasPane.add(flower);
            //添加梅花
            canvasPane.repaint();
            //重绘面板
        } else if (e.getModifiers()==
    InputEvent.BUTTON3_MASK) {          //按下右键
            Component at=
        canvasPane.getComponentAt(e.getPoint());       //获取鼠
        标位置的组件
            if (at instanceof JLabel)
        {                                    //判断是否为JLabel
            canvasPane.remove(at);
            //移除组件
            canvasPane.repaint();
            //重绘面板
            }
        }
    }
});
```

举一反三

根据本实例，读者可以开发以下程序。

游戏打地鼠。

实例466 打造自己的开心农场

本实例是一个提高效率、人性化的程序

实例位置：光盘\mingrisoft\18\Ex18_466

实例说明

开心农场是目前比较流行的一款网络游戏。在开心农场里，用户可以种植自己的蔬菜或水果，当一个快乐的农民。本实例开发了一个开心农场游戏，运行程序，效果如图18.9所示，单击“播种”按钮，可以播种种子；单击“生长”按钮，可以让作物处于生长阶段；单击“开花”按钮，可以让作物处于开花阶段；单击“结果”按钮，可以让作物结果；单击“收获”按钮，可以收获果实到仓库中，收获两个果实并且第3棵作物已经结果的效果如图18.10所示。

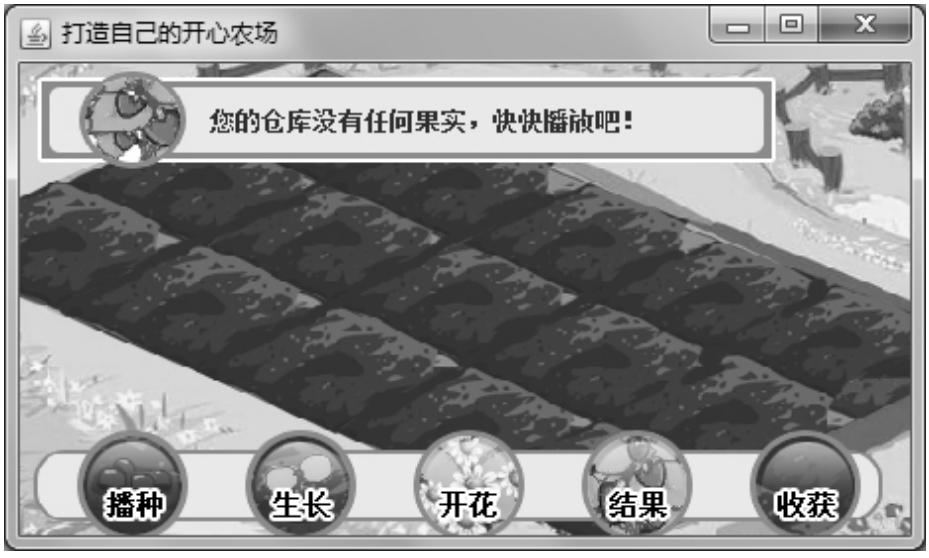


图18.9 游戏刚刚运行时的效果

图18.10 收获2 个果实且第3 棵作物已结果的效果

技术要点

本实例主要应用了Java中的类、对象、成员变量、成员方法和局部变量等技术。另外，在实现作物各种状态的改变时，可以应用已有类Crop。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的MainFrame窗体类，用于完成播种、生长、开花、结果和收获等操作。

（3）编写一个农场类，名称为Farm，在该类中编写seed()方法，用于实现播种操作。在该方法中，如果作物的状态为未播种，则进行播种，将作物显示为播种状态，并修改成员变量state的值为1（表示已播种），否则，设置提示信息为不能播种。

（4）在农场类 Farm 中编写 grow()方法，用于实现生长操作。在该方法中，如果作物的状态为已播种，则让其处于生长阶段，并修改成员变量 state 的值为 2（表示已生长），否则，设置提示信息为不能生长。

（5）在农场类Farm中编写fruit()方法，用于实现结果操作。在该方法中，如果作物的状态为已开花，则让其处于结果阶段，并修改成员变量 state 的值为 4（表示已结果），否则，设置提示信息为不能结果。

（6）在农场类Farm中编写harvest()方法，用于实现收获操作。在该方法中，如果作物的状态为已结果，则收获果实，让作物处于未播种状态，并修改成员变量state的值为0（表示未播种），否则，设置提示信息为不能收获。

（7）在农场类Farm中编写getMessage()方法，用于根据作物的状态属性，确定对应的提示信息，具体代码如下：

```
public String getMessage() {
  String message ="";
  switch (state) {
    case 0:
      message ="作物还没有播种";
      break;
    case 1:
      message ="作物刚刚播种";
      break;
    case 2:
      message ="作物正在生长";
      break;
    case 3:
      message ="作物正处于开花期";
      break;
    case 4:
      message ="作物已经结果";
```

```
        break;
    }
    return message;
}
```

举一反三

根据本实例，读者可以开发以下程序。

游戏打造自己的开心牧场。

18.3 其他

实例467 小猪走迷宫

本实例是一个人性化的实例

实例位置：光盘\mingrisoft\18\Ex18_467

实例说明

本实例实现了小猪走迷宫的游戏开发。运行程序，开始游戏，效果如图 18.11 所示，用户可以通过键盘上的方向键控制小猪的行走，如果在行走过程中撞到墙壁，会弹出消息框进行提示；如果顺利走出迷宫，则效果如图18.12所示。

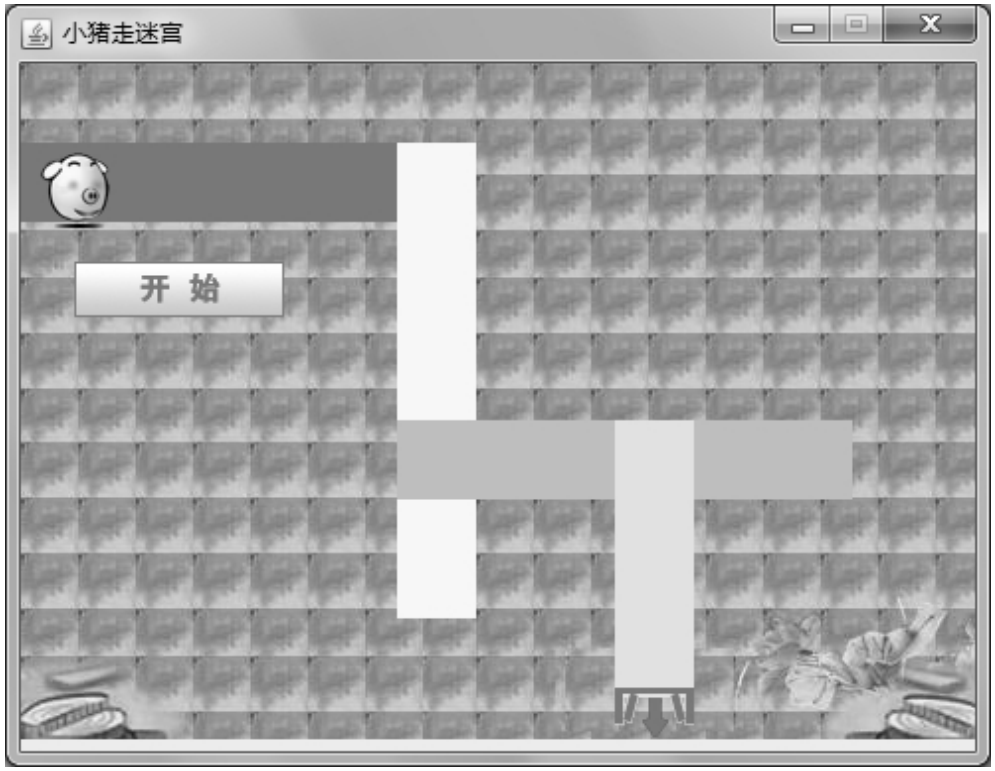


图18.11 游戏开始时的效果

图18.12 走出迷宫后的效果

技术要点

本实例通过键盘事件类KeyEvent的getKeyCode()方法，获得按键的整数代码，来控制小猪的移动，并使用Rectangle类的intersects()方法判断两个Rectangle对象是否相交，实现对小猪是否撞墙的判断。

（1）KeyEvent类的getKeyCode()方法，用于获得键盘上实际按键的整数代码，该方法的定义如下：

public int getKeyCode()

参数说明

返回值：键盘上实际按键的整数代码。

（2）使用Rectangle类的intersects()方法判断两个Rectangle对象是否相交，从而实现判断小猪是否撞墙的操作，Rectangle类的intersects()方法定义如下：

public boolean intersects(Rectangle r)

参数说明

● r：指定的Rectangle 对象。

● 返回值：如果指定的 Rectangle 对象与当前 Rectangle 对象相交，则返回 true；否则返回false。

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承JFrame类的PigWalkMazeFrame窗体类，并实现KeyListener和Runnable接口，目的就是通过键盘控制小猪的移动，并在小猪走出迷宫时，通过新的线程改变小猪走出迷宫的图片。

（3）在窗体类 PigWalkMazeFrame 中通过键盘的按键事件控制小猪的移动，键盘按键事件的代码如下：

```
public void keyPressed(KeyEvent e) {
  if ((gobuttonY==286))
{                                          //如果小猪的纵坐标等
于286
    Thread thread = new Thread(this);
    thread.start();
      //启动线程
  }
  if (e.getKeyCode()==KeyEvent.VK_UP)
{                              //如果用户按下了向上键
    Rectangle rectAngle=new
Rectangle(gobuttonX,gobuttonY,20,20);  //创建Rectangle对
象
    if (rectAngle.intersects(rect1)||
rectAngle.intersects(rect2)||
rectAngle.intersects(rect3)||
```

```
  rectAngle.intersects(rect4)) {                          //
  判断小猪是否走出了迷宫
        gobuttonY=gobuttonY -
    2;                                  //设置变量坐标
        goButton.setLocation(gobuttonX,gobuttonY);
            //设置按钮坐标
      //如果小猪走出了迷宫} else {
        JOptionPane.showMessageDialog(this, "撞墙了吧！重新
    开始吧！", "撞墙啦！", JOptionPane.DEFAULT_OPTION);
      }
    } else if (e.getKeyCode() == KeyEvent.VK_DOWN) { //判断
用户是否按向下键
      Rectangle rectAngle = new Rectangle(gobuttonX,
  gobuttonY, 20, 20);
      if (rectAngle.intersects(rect1)||
  rectAngle.intersects(rect2)||
  rectAngle.intersects(rect3)||
  rectAngle.intersects(rect4)) {
        gobuttonY = gobuttonY + 2;
        goButton.setLocation(gobuttonX, gobuttonY);
      } else {
        JOptionPane.showMessageDialog(this, "撞墙了吧！重新
    开始吧！", "撞墙啦！",JOptionPane.DEFAULT_OPTION);
      }
    } else if (e.getKeyCode()==KeyEvent.VK_LEFT)
{                          //如果用户按向左键
```

```
        Rectangle rectAngle = new Rectangle(gobuttonX,
    gobuttonY, 20, 20);
        if (rectAngle.intersects(rect1)||
    rectAngle.intersects(rect2)||
    rectAngle.intersects(rect3)||
    rectAngle.intersects(rect4)) {
          gobuttonX = gobuttonX - 2;
          goButton.setLocation(gobuttonX, gobuttonY);
        } else {
          JOptionPane.showMessageDialog(this, "撞墙了吧！重新
      开始吧！", "撞墙啦！",JOptionPane.DEFAULT_OPTION);
        }
      } else if (e.getKeyCode()==KeyEvent.VK_RIGHT)
{                          //如果用户按向右键
        Rectangle rectAngle = new Rectangle(gobuttonX,
    gobuttonY, 20, 20);
        if (rectAngle.intersects(rect1)||
    rectAngle.intersects(rect2)||
    rectAngle.intersects(rect3)||
    rectAngle.intersects(rect4)) {
          gobuttonX = gobuttonX + 2;
          goButton.setLocation(gobuttonX, gobuttonY);
        } else {
          JOptionPane.showMessageDialog(this, "撞墙了吧！重新
      开始吧！", "撞墙啦！",JOptionPane.DEFAULT_OPTION);
        }
      }
```

```
    }
```

（4）当小猪走出迷宫后，可以通过“开始”按钮重新开始游戏。“开始”按钮的事件代码如下：

```
    public void buttonAction(ActionEvent e) {
      goButton.setIcon(imageIcon);
            //重新设置按钮的显示图片
      goButton.addKeyListener(this);
            //为按钮添加键盘事件
      goButton.setBounds(0,40, imageIcon.getIconWidth(),
  imageIcon.getIconHeight());
      //设置小猪位置
      gobuttonX=goButton.getBounds().x;
          //获得小猪当前位置的x坐标
      gobuttonY=goButton.getBounds().y;
          //获得小猪当前位置的y坐标
      goButton.requestFocus();
          //设置按钮获取焦点
    }
```

（5）当小猪走出迷宫后，通过线程来改变小猪走出迷宫的图标。run()方法实现该功能的代码如下：

```
    public void run() {
      URL
  out=getClass().getResource("/images/pigOut.png");
      //获取图片URL
      ImageIcon imageout=new
  ImageIcon(out);                              //创建图片对象
```

```
        goButton.setIcon(imageout);
            //设置小猪按钮显示图片
        goButton.setBounds(gobuttonX,gobuttonY -
    imageout.getIconHeight()+50, imageout.getIconWidth(),
    imageout.getIconHeight());          //重新设置按钮位置
        goButton.removeKeyListener(this);
              //按钮移除键盘事件
    }
```

举一反三

根据本实例，读者可以开发以下程序。

游戏俄罗斯方块。

实例468 海滩捉螃蟹

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\18\Ex18_468

实例说明

本实例使用线程和鼠标事件监听器开发一个捉螃蟹的程序。程序启动后，会有一个线程控制螃蟹随机出现在沙滩的某个小洞里，用鼠标单击某个螃蟹，表示它被抓，螃蟹会“流泪”，而鼠标也会变成拾取物品的动作。运行程序，效果如图18.13所示。

图18.13 海滩捉螃蟹

技术要点

本实例使用随机数随机确定螃蟹出现的顺序，并使用线程进行循环控制，使螃蟹不停地出现。

（1）开发中唯一的一个难点就是如何控制螃蟹的显示。考虑到游戏对用户的吸引程度，螃蟹的显示应该是随机确定位置，而不能以固定的顺序出现，这就需要使用随机数。本程序采用的是Math类的random()方法，关键代码如下：

```
int index= (int) (Math.random() * 6);                    //生成随机的螃蟹索引
```

（2）螃蟹的显示需要不停地循环，在循环中检测螃蟹是否显示，并为空位置添加螃蟹图片， run()方法的代码如下：

```
public void run() {
  while (true) {                                    //使用无限循环
    try{
      Thread.sleep(1000);                           //使线程休眠1秒
      int index= (int) (Math.random() * 6);         //生成随机的螃蟹索引
      if (carb[index].getIcon()==null) {            //如果螃蟹标签没有设置图片
        carb[index].setIcon(imgCarb);               //为该标签添加螃蟹图片
      }
    } catch (InterruptedException e) {
      e.printStackTrace();
    }
  }
}
```

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame类的 CaptureCarbFrame窗体类，同时该窗体类实现了Runnable接口，因此可以通过该类创建线程对象，其中run()方法的代码见关键技术部分。

（3）编写内部类MouseCrab，该类实现了MouseListener接口，即窗体控件的鼠标事件监听器，用于在鼠标按键的按下和释放以及鼠标

离开控件区域动作发生时替换鼠标的光标图片，这样会使程序界面更加形象，因为替换的两个图片能够形成抓取物品的连贯动作，分别作为窗体和标签控件的事件监听器，MouseCrab类的代码如下：

```
    private final class MouseCrab implements MouseListener {
      private final Cursor cursor1;                    //鼠标图标1
      private final Cursor cursor2;                    //鼠标图标2
      private MouseCrab(Cursor cursor1, Cursor cursor2) {
        this.cursor1 = cursor1;
        this.cursor2 = cursor2;
      }
      @Override
      public void mouseReleased(MouseEvent e) {
        setCursor(cursor1);                            //鼠标按键释放时设置光标为cursor1
      }
      @Override
      public void mousePressed(MouseEvent e) {
        setCursor(cursor2);                            //鼠标按键按下时设置光标为cursor2
      }
      @Override
      public void mouseExited(MouseEvent e) {
        setCursor(cursor1);                            //鼠标离开控件区域时设置光标为cursor1
      }
```

```
    @Override
    public void mouseEntered(MouseEvent e) {
    }
    @Override
    public void mouseClicked(MouseEvent e) {
    }
}
```

（4）本实例还创建了一个鼠标事件监听器类Catcher，它对鼠标按键进行判断之后才设置控件图片，但是设置的不是鼠标光标图片，而是显示螃蟹的标签控件的图片。在鼠标单击时，换成螃蟹流泪的图片，而鼠标释放和初始状态都是一个微笑的螃蟹图片。该事件监听器主要作为显示螃蟹的标签控件的事件监听器，代码如下：

```
private final class Catcher extends MouseAdapter {
    @Override
    public void mousePressed(MouseEvent e) {
      if (e.getButton() != MouseEvent.BUTTON1)
        return;
      Object source= e.getSource();                    //获取事件源，即螃蟹标签
      if (source instanceof JLabel) {                  //如果事件源是标签组件
        JLabel carb= (JLabel) source;                  //强制转换为JLabel标签
        if (carb.getIcon() != null)
          carb.setIcon(imgCarb2);                      //为该标签添加螃蟹图片
      }
```

```
    }
    @Override
    public void mouseReleased(MouseEvent e) {
      if (e.getButton() != MouseEvent.BUTTON1)
        return;
      Object source= e.getSource();                    //获取事
件源，即螃蟹标签
      if (source instanceof JLabel) {                  //如果
事件源是标签组件
        JLabel carb= (JLabel) source;                  //强制转换
    为JLabel标签
        carb.setIcon(null);                            //清除标签
    中的螃蟹图片
      }
    }
}
```

举一反三

根据本实例，读者可以开发以下程序。

游戏小猫钓鱼。

实例469 荒山打猎游戏

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\18\Ex18_469

实例说明

本实例程序界面上，底部会有野猪随机出现并以不固定的速度移动，上方有小鸟以反方向飞过，当用鼠标在它们身上进行单击操作

时，会打中该动物，动物消失，在界面左上角得到相应分数，但是如果动物跑出界面，游戏就会扣除一定的分数。另外，界面右上角会显示当前剩余子弹数量，如无子弹，需要等待系统装载子弹。运行程序，效果如图18.14所示。

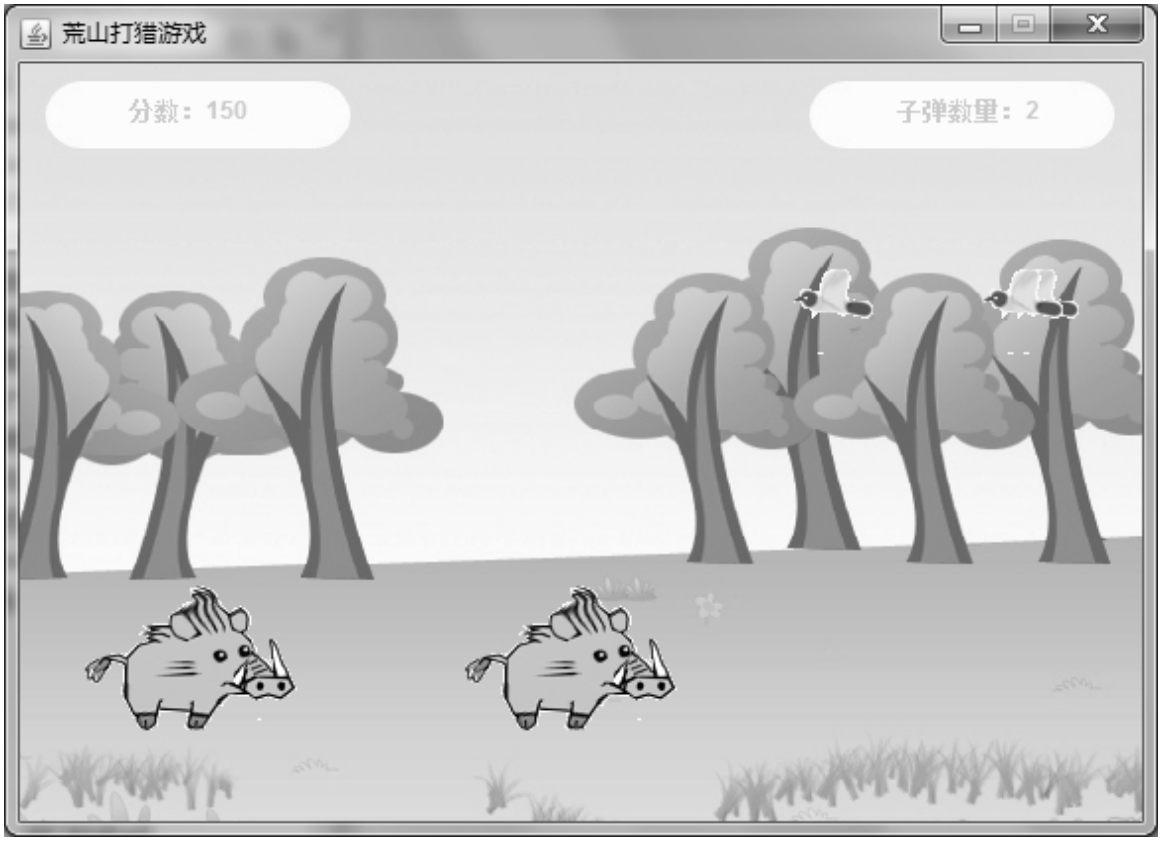


图18.14 荒山打猎游戏

技术要点

本程序的难点在于控制动物控件的移动速度。如果每个动物移动的速度相同，就会使程序运行效果枯燥乏味，没有游戏难度也就没有进行下去的意义，所以要在线程中控制每个控件的移动速度。在线程循环中，可以通过随机数来确定新创建的动物控件移动线程的休眠时间，这样就可以为每个动物控件设置不同的移动速度，如在MainFrame窗体类的内部线程类PigThread中，调用了Math类的random()方法随机确定线程休眠的时间。

窗体类MainFrame的内部线程类PigThread的代码如下：

```
class PigThread extends Thread {
  @Override
  public void run() {
    while (true) {
      PigLabel pig=new PigLabel();                    //创
  建代表野猪的标签控件
      pig.setSize(120,80);                           //设置
  控件初始大小
      backgroundPanel.add(pig);                       //添加
  控件到背景面板
      try{
        sleep((long) (random() * 3000)+500);          //线
    程随机休眠一段时间
      } catch (InterruptedException e) {
        e.printStackTrace();
      }
    }
}
}
```

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承 JLabel 类的 BirdLabel 标签类，用于表示小鸟，并且实现Runnable接口。通过线程控制小鸟的移动效果，以及实现扣分功能。BirdLabel标签类中run()方法的代码如下：

```
public void run() {
  parent = null;
  int width = 0;
```

```
    try{
      while(width<=0||parent==null){
        if (parent == null){
          parent=getParent();                          //获取父
      容器
        } else {
          width=parent.getWidth();                     //获取父
      容器的宽度
        }
        Thread.sleep(10);
      }
      for (int i = width; i > 0 && parent != null; i -= 8)
    {
        setLocation(i,y);                              //从右
      向左移动本组件位置
        Thread.sleep(sleepTime);                       //休
      眠片刻
      }
    } catch (InterruptedException e) {
      e.printStackTrace();
    }
    if (parent != null) {
      MainFrame.appScore(-score* 10);                  //自
    然销毁将扣分
    }
    destory();                                         //移
动完毕，销毁本组件
```

```
}
```

（3）在项目中创建一个继承JLabel类的PigLabel标签类，用于表示野猪，并且实现Runnable接口。通过线程控制野猪的移动效果，以及实现扣分功能。PigLabel标签类中run()方法的代码如下：

```
public void run() {
  parent = null;
  int width = 0;
  while (width<=0 ||parent==null) {                    //
获取父容器的宽度
    if (parent == null)
      parent=getParent();                              //获取父
    容器
    else
      width=parent.getWidth();                          //获取
    父容器的宽度
    }
    for (int i = 0; i < width && parent != null; i += 8) {
      setLocation(i,y);                                  //从
    左向右移动本组件
      try{
        Thread.sleep(sleepTime);                         //休
      眠片刻
      } catch (InterruptedException e) {
        e.printStackTrace();
      }
    }
    if (parent != null) {
```

```
        MainFrame.appScore(-score* 10);                    //自
    然销毁将扣分
        }
        destory();                                          //移
    动完毕，销毁本组件
    }}
```

（4）在项目中创建一个继承JFrame类的主窗体类MainFrame，在该类中分别创建生成小鸟和小猪角色的内部线程类，其中生成小鸟角色的类代码如下：

```
    class BirdThread extends Thread {
        @Override
        public void run() {
            while (true) {
                BirdLabel bird=new BirdLabel();             //创建
        代表小鸟的标签控件
                bird.setSize(50,50);                        //设
        置控件初始大小
                backgroundPanel.add(bird);                  //添
        加控件到背景面板
                try {
                    sleep((long) (Math.random() * 3000)+500);    //线
            程随机休眠一段时间
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }
```

}

举一反三

根据本实例，读者可以开发以下程序。

游戏坦克大战。

实例470 警察抓小偷

本实例可以方便操作、提高效率

实例位置：光盘\mingrisoft\18\Ex18_470

实例说明

本实例开发了一个警察抓小偷游戏。程序运行后，小偷在窗体中左侧位置的一定范围内随机跑动，警察在窗体的中上部位置，当用鼠标单击击中小偷时，则表示小偷被打中，游戏过关，单击“再来一次”按钮，又可以开始新的游戏。运行程序，效果如图18.15所示。

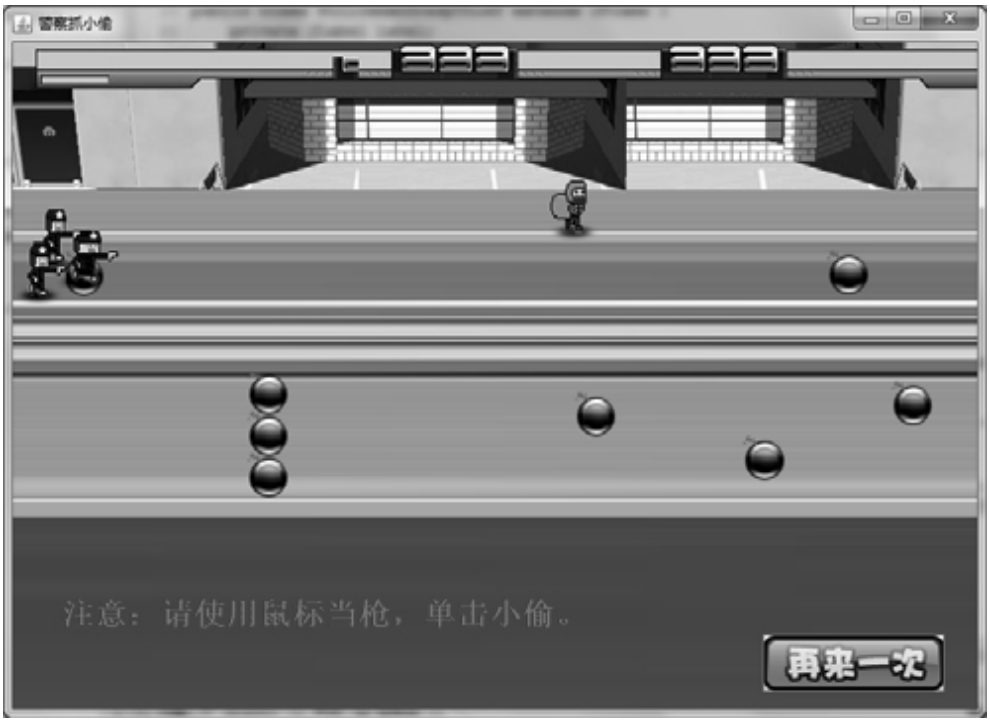


图18.15 警察抓小偷

技术要点

在使用线程控制小偷标签的移动位置时，通过线程类的成员变量x来控制小偷左右运动；通过随机数控制小偷上下运动。改变小偷标签移动位置可以通过如下代码实现：

```
    if (flag== false) {                                    //
flag为false
      x+=20;                                          // x的值增加
  表示向右运动
      if (x==640) {                                    //当小偷标
  签左侧边界的横坐标是640时
        flag= true;                                   //将flag赋值
    为true
      }
    } else {                                           // flag为
true
      x -=20;                                         // x的值减少
  表示向左运动
      if (x==100) {                                    //当小偷标
  签左侧边界的横坐标是100时
        flag= false;                                  //将flag赋
    值为false
      }
    }
    //生成100~200之间的随机整数，用于设置小偷标签上边界的纵坐
标
    int y = (int) (Math.random() * 100) + 100;
    lb_thief.setLocation(x,y);                              //设
置小偷标签的显示位置
```

实现过程

（1）新建一个项目。

（2）在项目中创建一个继承 JFrame 类的 PolicemanGraspThief 窗体类，在窗体PolicemanGraspThief 中创建内部线程类 GraspThiefThread，用于实现动画效果，该内部线程类的代码如下：

```
private class GraspThiefThread implements Runnable {
  boolean flag= false;                                    //标识小偷向左运动还是向右运动的变量
  int x=400;                                              //小偷标签左侧边界的横坐标
  public void run() {
    while (true) {
      if (stop) {                                         // stop为true时，显示提示文本为“打中了”标签
        int x= lb_thief.getX();                           //获得小偷标签的横坐标
        int y= lb_thief.getY();                           //获得小偷标签的纵坐标
        lb_tip.setBounds(x,y+60,50,50);                   //设置提示文本为“打中了”标签的显示位置和大小
        lb_tip.setVisible(true);                          //显示提示文本为“打中了”的标签
        thread=null;                                      //释放线程资源
        break;                                            //退出循环，结束线程的执行
      }
```

```
    if (flag== false) {                    // flag为
false，向右运动
      x+=20;                              // x的值增加表示
  向右运动
      if (x==640) {                       //当小偷标签左侧
  边界的横坐标是640时
        flag= true;                       //将flag赋值为
    true
      }
    } else {                              // flag为
true，向左运动
      x -=20;                             // x的值减少表示
  向左运动
      if (x==100) {                       //当小偷标签左侧
  边界的横坐标是100时
        flag= false;                      //将flag赋值为
    false
      }
    }
    //生成100～200之间的随机整数，用于设置小偷标签上边
界的纵坐标
    int y = (int) (Math.random() * 100) + 100;
    lb_thief.setLocation(x,y);                //设置小
偷标签的显示位置
    try {
      Thread.sleep(200);                  //休眠200毫秒
    } catch (InterruptedException e) {
```

```
            e.printStackTrace();
        }
    }
  }
}
```

举一反三

根据本实例，读者可以开发以下程序。

游戏对对碰。

图书在版编目（CIP）数据

Java程序开发范例宝典/赛奎春，郭鑫，宋禹蒙编著.--北京：人民邮电出版社，2015.1

（软件工程师：典藏版）

ISBN 978-7-115-37016-7

Ⅰ.①J… Ⅱ.①赛…②郭…③宋… Ⅲ.①JAVA语言—程序设计 Ⅳ.①TP312

中国版本图书馆CIP数据核字（2014）第256709号

内容提要

本书紧密围绕编程者在编程中遇到的实际问题和开发中应该掌握的技术，全面介绍了应用Java进行程序开发的各方面技术和技巧。全书分为 18 章，内容包括窗体与界面设计、控件应用、Commons 组件应用、数据库技术、SQL 查询相关技术、数据库高级应用、图形图像技术、动画、文件操作技术、办公文档、JFreeChart图表、报表打印、操作PDF、解析XML文件、网络技术、邮件收发技术、Java安全、游戏开发。全书共提供了 470 个实例，每个实例都突出实用性，其中大部分是程序开发者梦寐以求的有关问题的解决方案。

本书附有配套光盘。光盘提供了书中所有实例的源代码，源代码全部经过精心调试，在 Windows XP/Windows Server 2003/Windows 7 下测试通过，保证能够正常运行。

本书适用于广大计算机爱好者和编程人员，也可供大中专院校师生阅读。

♦编著 赛奎春 郭鑫 宋禹蒙

责任编辑 李永涛

责任印制 杨林杰

♦人民邮电出版社出版发行 北京市丰台区成寿寺路11号

邮编 100164 电子邮件 315@ptpress.com.cn

网址　http://www.ptpress.com.cn
三河市海波印务有限公司印刷
♦开本：787×1092　1/16
印张：44.25
字数：1197千字　　2015年1月第1版
印数：1－3000册　　2015年1月河北第1次印刷
定价：99.00元（附光盘）
读者服务热线：(010)81055410　印装质量热线：
(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京崇工商广字第0021号

链接：http://pan.baidu.com/s/1sj0p3y9 密码：zo9j