Chapter 13

图形界面编程

本章之前的所有应用程序都是命令行界面，没有特定的用户输入界面。从本章开始，将要学习图形界面编程。对于一个软件来说，不但要有比较强大完善的功能，而且还要有一个简

洁美观的界面。本章主要学习如何进行图形界面编程。其中包括 AWT 和 Swing 两部分内容。

本章主要内容有:

◎ Swing 基础
◎ 事件
◎ Swing 组件
◎ 布局管理器

13.1 AWT 简介

AWT 的全称是抽象窗口工具集(Abstract Window Toolkit)。它是一个特殊的组件，其中包含有其他的组件。它的库类也非常丰富，包括了创建 Java 图形界面程序的所有工具。用户可以利用 AWT，在容器中创建标签、按钮、复选框、文本框等用户界面元素。

AWT 中包括了图形界面编程的基本类库。它是 Java 语言 GUI 程序设计的核心，它为用户提供基本的界面构件。这些构件是为了使用户和机器之间能够更好地进行交互，而用来建立图形用户界面的独立平台。其中主要由以下几部分组成，包括：组件类(Component)、容器类(Container)、图形类(Graphics)和布局管理器(LayoutManager)。

- Component(组件)——按钮、标签、菜单等组件的抽象基本类。
- Container(容器)——扩展组件的抽象基本类。如 Panel、Applet、Window、Dialog 和 Frame 等是由 Container 演变的类，容器中可以包括多个组件。
- LayoutManager(布局管理器)——定义容器中组件摆放位置和大小接口。Java 中定义了几种默认的布局管理器。
- Graphics(图形类)——组件内与图形处理相关的类，每个组件都包含一个图形类的对象。

在 AWT 中存在缺少剪贴板、缺少打印支持等缺陷，甚至没有弹出式菜单和滚动窗口等，

因此 Swing 的产生也就成为必然。Swing 是纯 Java 实现的轻量级(light-weight)组件，它不依赖系统的支持。本章主要讨论 Swing 组件基本的使用方法和使用 Swing 组件创建用户界面的初步方法。

13.2 Swing 基础

Swing 元素的屏幕显示性能要比 AWT 要好，而且 Swing 是使用纯 Java 来实现的。所以 Swing 也理所当然地具有 Java 的跨平台性。但 Swing 并不是真正使用原生平台提供设备，而是仅仅在模仿。因此，可以在任何平台上来使用 Swing 图形用户界面组件。它不必在它们自己本地窗口中绘制组件，而是在它们所在的重量级窗口中绘制，因为 Swing 绝大部分是轻量级的组件。

注意

AWT 组件具有平台相关性，它是系统对等类的实现；而 Swing 组件在不同平台具有一致性的表现，另外还可以提供本地系统不支持的一些特征。因此 Swing 比 AWT 的组件实用性更强。Swing 采用了 MVC(模型-视图-控制，Model-View-Controller)设计模式。

13.2.1 Swing 的类层次结构

Javax.swing 包中有顶层容器和轻量级两种类型的组件，Swing 轻量级的组件都是由 AWT 的 Container 类来直接或者是间接派生而来的。

```
java.awt.Component
  +-java.awt.Container
    +-java.awt.Window
      +-java.awt.Frame-javax.swing.JFrame
      +-javax.Dialog-javax.swing.JDialog
      +-javax.swing.JWindow
    +-java.awt.Applet-javax.swing.JApplet
    +-javax.swing.Box
    +-javax.swing.Jcomponet
```

Swing 包是 JFC(Java Foundation Classes)的一部分，它由许多包组成，如表 13-1 所示。

表 13-1 Swing 包组成内容

包	描 述
Com.sum.swing.plaf.motif	实现 Motif 界面样式代表类
Com.sum.java.swing.plaf.windows	实现 Windows 界面样式的代表类

javax.swing	Swing 组件和使用工具
javax.swing.border	Swing 轻量组件的边框
javax.swing.colorchooser	JcolorChooser 的支持类/接口
javax.swing.event	事件和侦听器类
javax.swing.filechooser	JFileChooser 的支持类/接口
javax.swing.pending	未完全实现的 Swing 组件
javax.swing.plaf	抽象类，定义 UI 代表的行为
javax.swing.plaf.basic	实现所有标准界面样式公共基类
javax.swing.plaf.metal	它们实现 Metal 界面样式代表类
javax.swing.table	Jtable 组件
javax.swing.text	支持文档的显示和编辑
javax.swing.text.html	支持显示和编辑 HTML 文档
javax.swing.text.html.parser	HTML 文档的分析器
javax.swing.text.rtf	支持显示和编辑 RTF 文件
javax.swing.tree	Jtree 组件的支持类
javax.swing.undo	支持取消操作

在表 13-1 所示的包中，javax.swing 是 Swing 所提供最大的包，其中包含有 100 个类和 25 个接口，并且绝大部分的组件都包含在 Swing 包中。

- javax.swing.event 包中定义了事件和事件处理类，这与 java.awt.event 包类似，主要包括事件类和监听器接口、事件适配器。
- javax.swing.pending 包主要是一些没有完全实现的组件。
- javax.swing.table 包中主要是 Jtable 类的支持类。
- javax.swing.tree 包同样也是 Jtree 类的支持类。
- javax.swing.text、javax.swing.text.html、javax.swing.text.html.parser 和 javax.swing.text.rtf 包都是与文档显示和编辑相关的包。

13.2.2 Swing 特点

- 组件的多样化：虽然 AWT 是 Swing 的基础，但是 Swing 中却提供了比 AWT 更多的图形界面组件。而且 Swing 中组件的类名都是由字母“J”开头，还增加了一些比较复杂的高级组件，如 JTable、JTree。
- MVC 模式：MVC 模型的普遍实用性是 Swing 比较突出的特点。其主要包括有模型、视图、控制器 3 部分结构。模型用于保存所用到的数据，视图则用于显示数据的内容，控制器用于处理用户和模块交互事件。MVC 设计思想是图形用户界面设计比较通用，这样可以使内容本身和显示方式分类，使数据显示更加灵活多变。
- 可存取性支持：Swing 的所有组件为了实现对可存取的支持而实现了 Accessible 接口。
- 支持键盘操作：在 Swing 中支持传统意义上的热键操作，这样就可以替代用户鼠标操作。

- 使用图标(Icon)：在 Swing 中可以通过添加图标来修饰自己。这样就更加丰富了 Swing 的功能，而且也会使得界面更加美观。

13.2.3 Swing 程序结构简介

使用 Swing 进行程序设计，首先要引入 Swing 的包，创建顶层的容器，在容器中创建按钮和标签等一系列的组件，并将组件添加到顶层容器中，然后在组件的周围添加边界，最后对组件的事件进行处理。

下面就通过一个 HelloWorld 的示例来演示第一个 Swing 程序，这个程序就是按照前面所介绍的流程进行编写的。

```
01  //首先导入 Swing 需要的包
02  import javax.swing.*;
03  import java.awt.*;
04  import java.awt.event.*;
05  //创建 HelloWorld 类
06  public class HelloWorld {
07      //创建主方法
08      public static void main(String[] args) {
09          try {                                   //try 语句块，监视该段程序
10              //设置窗口风格
11  UIManager.setLookAndFeel(UIManager.getCrossPlatformLookAndFeelClassName());
12          } catch (Exception e) {                 //捕获异常
13              e.printStackTrace();                //异常信息输出
14          }
15          JFrame frame = new JFrame("Swing 第一个示例");//创建顶层容器并初始化
16          Container c = frame.getContentPane();   //获取面板容器
17          JPanel pane = new JPanel();             //创建面板 panel 并初始化
18          c.add(pane);                            //将面板添加到窗口
19          pane.setLayout(new FlowLayout());       //设置布局管理器 FlowLayout
20          final JLabel label = new JLabel();      //创建标签 label 并初始化
21          JButton button = new JButton("按钮");   //创建 button 并初始化
22          pane.add(label);                        //向容器中添加组件 label
23          pane.add(button);                       //向容器中添加组件 button
24          //对按钮事件的处理方法
25      button.addActionListener(new ActionListener() {
26              public void actionPerformed(ActionEvent e) {
27                  label.setText("HelloWorld！");  //设置 label 显示的内容
28              }
29          });
30          //窗口设置结束，开始显示
31          frame.addWindowListener(new WindowAdapter() {
32              //匿名类用于注册监听器
```

```
33                          public void windowClosing(WindowEvent e) {
34                              System.exit(0);}                    //程序退出
35                      });
36                      frame.setSize(300,240);                     //设置窗口大小
37                      frame.setVisible(true);                     //显示窗口
38                  }
39      }
```

以上程序中完全按照前面所介绍到的流程，首先是将 Swing 所需要的包导入(第 1~4 行，然后在主方法中创建顶层容器 frame 并初始化(第 15 行)。然后获取面板容器并将面板添加到容器中(第 16~18 行)，然后创建一个标签组件并初始化，接下来创建按钮组件并为按钮设置事件监听，最后处理事件监听(第 20~27 行)。运行程序，会弹出一个窗口，单击按钮，就会在按钮旁边出现“HelloWorld！”字样。如图 13-1 所示。

图 13-1　HelloWorld 示例

13.3 Swing 组件

Swing 的组件与 AWT 组件相似，但又为每一个组件增添了新的方法，并提供了更多的高级组件。所以本节 Swing 的基本组件选取几个比较典型的组件进行详细讲解，本节没有讨论到的组件，读者在使用中遇到困难时，可参阅 API 文档。

13.3.1　按钮(Jbutton)

Swing 中的按钮是 Jbutton，它是 javax.swing.AbstracButton 类的子类，Swing 中的按钮可以显示图像，并且可以将按钮设置为窗口的默认图标，而且还可以将多个图像指定给一个按钮。在前面的 HelloWorld 例子中就是用到了一个按钮。Jbutton 类的继承关系如下：

```
java.lang.Object
    +--java.awt.Component
        +--java.awt.Container
            +--javax.swing.JComponent
                +--javax.swing.AbstractButton
                    +--javax.swing.JButton
```

在 Jbutton 中有如下几个比较常用的构造方。

- JButton(Icon icon) ：按钮上显示图标。
- JButton(String text)：按钮上显示字符。
- JButton(String text, Icon icon)：按钮上既显示图标又显示字符。

下面就是一个设置按钮的示例。

```
01  public class JButtontest {
02      public static void main(String[] args) {
03      JFrame f = new JFrame("这是一个按钮");
04      Container contentPane = f.getContentPane();
05      JButton b = new JButton("按钮");//创建一个带初始文本的按钮
06      //如果没有设置文字的位置，系统默认值会将文字置于图形的右边中间位置
07  //设置文本相对于图标的水平方向的位置
08  b.setHorizontalTextPosition(JButton.CENTER);
09  //设置文本相对于图标的垂直方向的位置
10  b.setVerticalTextPosition(JButton.BOTTOM);
11      contentPane.add(b);
12      f.pack();
13      f.show();
14      f.addWindowListener(new WindowAdapter() {
15          public void windowClosing(WindowEvent e) {
16              System.exit(0);
17          }
18      });
19  }
20  }
```

程序运行结果如图 13-2 所示。首先创建一个窗口，并设置窗口的显示文本(第 3 行)。然后在窗口中设置一个按钮并设置按钮的显示文字(第 5 行)。接下来对文本相对于图标的方向位置进行设置(第 8 行)。最后为按钮添加事件监听(第 14~18 行)。

图 13-2　按钮示例效果

13.3.2　复选框(JCheckBox)

使用复选框可以完成多项选择，Swing 中的复选框和 AWT 中复选框相比，优点就是 Swing 复选框中可以添加图片。该类是 javax.swing.JToggleButton 的子类，其继承关系如下：

```
java.lang.Object
  +--java.awt.Component
      +--java.awt.Container
          +--javax.swing.JComponent
              +--javax.swing.AbstractButton
                  +--javax.swing.JToggleButton
                      +--javax.swing.JcheckBox
```

复选框可以为每一次的单击操作添加一个事件。

复选框的构造方法如下。

- JCheckBox()：创建一个无文本、无图标、未被选定的复选框。
- JCheckBox(Action a)：创建一个复选框，属性由参数 Action 提供。

- JCheckBox(Icon icon)：创建一个有图标，但未被选定的复选框。
- JCheckBox(Icon icon, boolean selected)：创建一个有图标，并且指定是否被选定的复选框。
- JCheckBox(String text)：创建一个有文本，但未被选定的复选框。
- JCheckBox(String text, boolean selected)：创建一个有文本，并且指定是否被选定的复选框。
- JCheckBox(String text, Icon icon)：创建一个指定文本和图标，但未被选定的复选框。
- JCheckBox(String text, Icon icon, boolean selected)：创建一个指定文本和图标，并指定是否被选定的复选框。

下面的示例是一个使用复选框来选择喜欢的城市的例子，其代码如下。

```
01  public class JCheckBoxTest  {
02      JFrame f = null;
03      JCheckBoxTest() {
04          f = new JFrame("复选框示例");                    // 创建一个 JFrame 实例对象
05          Container contentPane = f.getContentPane();      // 定义一个容器
06          contentPane.setLayout(new GridLayout(2, 1));     // 设置窗口的布局
07          JPanel p1 = new JPanel();                        // 创建一个面板对象 p1
08          p1.setLayout(new GridLayout(1, 3));
09          p1.setBorder(BorderFactory.createTitledBorder("选择你喜欢的城市？"));
10          JCheckBox c1 = new JCheckBox("北京");
11          JCheckBox c2 = new JCheckBox("上海");
12          JCheckBox c3 = new JCheckBox("青岛");
13          p1.add(c1);
14          p1.add(c2);
15          p1.add(c3);
16
17          contentPane.add(p1);
18
19          f.pack();
20          f.show();
21          f.addWindowListener(new WindowAdapter() {          // 创建一个窗口监听器
22                  public void windowClosing(WindowEvent e) {
23                      System.exit(0);
24                  }
25              });
26      }
27      public static void main(String args[]) {
28          new JCheckBoxTest();
29      }
30  }
```

程序运行结果如图 13-3 所示。首先要创建 JFrame 对象、容器，并设置窗口的布局格式(第 4~7 行)，定义 3 个 JCheckBox 对象，并设置各

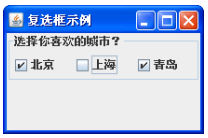

图 13-3　复选框示例效果

自的显示文本(第 10~12 行)。然后为窗口设置事件监听(第 21~25 行)。最后创建主方法，并在主方法中调用构造方法(第 27~29 行)。

13.3.3 单选框(JRadioButton)

单选框(JRadioButton)与 AWT 中的复选框组功能类似，通常 JRadioButton 和 ButtonGroup 配合在一起使用，作用是一次创建一组按钮，并且在这一组按钮中，每一次只能够选中一个按钮。需要使用到 add()方法将 JRadioButton 添加到 ButtonGroup 中。

注意

使用 ButtonGroup 对象进行分组是逻辑分组而不是物理分组。创建一组按钮通常需要创建一个 JPanel 或者类似容器，并将按钮添加到容器中。

JRadioButton 的常用构造方法如下。

- JRadioButton()：用于创建一个未指定图标和文本，并且未被选定的单选按钮。
- JRadioButton(Action a)：用于创建一个属性来自 Action 的单选按钮。
- JRadioButton(Icon icon)：用于创建一个指定图标未被选定的单选按钮。
- JRadioButton(Icon icon, boolean selected)：用于创建一个指定图像和状态的单选按钮。
- JRadioButton(String text)：用于创建一个指定文本未被选择的单选按钮。
- JRadioButton(String text, boolean selected)：用于创建一个执行文本和选择状态的单选按钮。
- JRadioButton(String text, Icon icon)：用于创建一个指定文本和图标未被选择的单选按钮。
- JRadioButton(String text, Icon icon, boolean selected)：用于创建一个具有指定的文本、图像和选择状态的单选按钮。

下面是一个单选框的示例，代码如下。

```
01  public class JRadioButtonTest  {
02      JFrame f = null;
03      JRadioButton r4 = null;
04      JRadioButton r5 = null;
05      JRadioButtonTest() {
06          f = new JFrame("单选框示例");                    // 创建一个 JFrame 的对象
07          Container contentPane = f.getContentPane();      // 创建一个容器
08          contentPane.setLayout(new GridLayout(2, 1));     // 设置这个窗口的布局
09          JPanel p1 = new JPanel();                        // 创建一个面板对象 p1
10          p1.setLayout(new GridLayout(1, 3));              // 设置布局管理器的格式
11          p1.setBorder(BorderFactory.createTitledBorder("选择你喜欢的城市"));
12          // 定义 3 个 JRadioButton 单选按钮
13          JRadioButton r1 = new JRadioButton("北京");
14          JRadioButton r2 = new JRadioButton("上海");
15          JRadioButton r3 = new JRadioButton("青岛");
```

```
16                  p1.add(r1);
17                  p1.add(r2);
18                  p1.add(r3);
19                  JPanel p2 = new JPanel();
20                  p2.setLayout(new GridLayout(2, 1));
21                  contentPane.add(p1);
22                  contentPane.add(p2);
23                  f.pack();
24                  f.show();
25                  f.addWindowListener(new WindowAdapter() {// 添加一个窗口监听器
26                                  public void windowClosing(WindowEvent e) {
27                                          System.exit(0);
28                                  }
29                          });
30          }
31          public static void main(String args[]) {
32                  new JRadioButtonTest();
33          }
34  }
```

程序运行结果如图 13-4 所示。首先要创建 JFrame 对象、容器，并设置窗口的布局格式(第 6~9 行)。定义 3 个 JRadioButton 对象，并设置各自的显示文本和布局格式(第 12~20 行)。然后为窗口设置事件监听(第 25~29 行)。最后创建主方法，并在主方法中调用构造方法(第 31~33 行)。

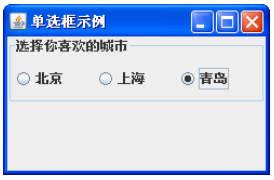

图 13-4　单选框示例效果

13.3.4　组合框(JComboBox)

组合框，顾名思义，就是将一些组件(如按钮及下拉菜单)组合的组件。用户可以使用下拉菜单选择不同的选项，还可以在组合框处于编辑状态时，在组合框中键入值。

组合框有以下比较常用的构造方法。

- JComboBox()：创建一个没有数据选项的组合框。
- JComboBox(ComboBoxModel aModel)：创建一个数据来源于 ComboBoxModel 的组合框。
- JComboBox(Object[] items)：创建一个指定数组元素作为选项的组合框。
- JComboBox(Vector<?> items)：创建一个指定 Vector 中元素的组合框。

下面是一个使用组合框的示例，用来选择喜欢的旅游景点和喜欢的城市。

```
01  public class JComboBoxTest {
02  public static void main(String[] args) {
03          JFrame jf = new JFrame("JComboBoxDemo1");      // 创建一个 JFrame 对象
04          Container contentPane = jf.getContentPane();      // 定义一个容器
```

```
05          contentPane.setLayout(new GridLayout(1, 2));
06          // 定义一个字符串数组，并将其初始化
07          String[] str = { "故宫", "泰山", "张家界", "颐和园", "孔府"};
08          Vector vector = new Vector();                    //创建一个 Vector 对象
09          //向 Vector 对象中添加数据
10          vector.addElement("北京");
11          vector.addElement("上海");
12          vector.addElement("青岛");
13          vector.addElement("广州");
14          vector.addElement("成都");
15          vector.addElement("西安");
16          JComboBox combo1 = new JComboBox(str); // 定义一个 JComboBox 对象
17          // 利用 JComboBox 类所提供的 addItem()方法，加入一个项目到此 JComboBox 中。
18          combo1.addItem("泰山");
19          //创建一个带有指定标题的标题框
20          combo1.setBorder(BorderFactory.createTitledBorder("你想去哪个景点游 23      玩？"));
21                JComboBox combo2 = new JComboBox(vector);
22                combo2.setBorder(BorderFactory.createTitledBorder("你喜欢的城市"));
23                contentPane.add(combo1);
24                contentPane.add(combo2);
25                jf.pack();
26                jf.show();
27                jf.addWindowListener(new WindowAdapter() {              // 添加窗口监听器
28                            public void windowClosing(WindowEvent e) {
29                                  System.exit(0);
30                            }
31                      });
32          }
33    }
```

以上程序运行结果如图 13-5 所示。首先在主方法中定义一个字符串数组并初始化(第 7 行)。然后创建一个 Vector 对象，并为其添加数据(第 8~15 行)。接下来定义 JComboBox 对象并使用 addItem 方法添加默认选项(第 16~18 行)。再创建一个 JComboBox，并添加数据(第 21 行)。最后为窗口添加事件监听(第 27~29 行)。

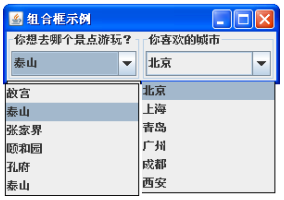

图 13-5　组合框效果

13.3.5　进程条(JprogressBar)

进程条就是用图形的方式来模拟描述任务进度的组件。任务的过程中，进程条会以百分比的形式来显示。在程序结束后，进程条显示百分之百。进程条的继承关系如下：

```
java.lang.Object
   +--java.awt.Component
         +--java.awt.Container
```

```
        +--javax.swing.JComponent
            +--javax.swing.JProgressBar
```

进程条的常用构造方法如下。

- JProgressBar()：创建一个显示矩形的水平进程条。
- JProgressBar(BoundedRangeModel newModel)：创建一个指定进程条数据模型的水平进程条。
- JProgressBar(int orient)：创建一个指定方向(VERTICAL 或 HORIZONTAL)的进程条。
- JProgressBar(int min, int max)：创建一个指定最小值和最大值的水平进程条。
- JProgressBar(int orient, int min, int max)：创建一个指定方向、最小值和最大值的进程条。

13.3.6 表格(JTable)

表格(JTable)是 Swing 新增加的组件，主要是为了将数据以表格的形式显示。通常用数据模型类的对象来保存数据，数据模型类派生于 AbstractTableModel 类，并且必须重写抽象模型类的几个方法，例如 getColumnCount，getRowCount，getColumnName，getValueAt。因为表格会从这个数据模型的对象中自动获取数据，数据模型类的对象负责表格大小(行/列)、数据填写、表格单元更新等与表格有关的属性和操作。表格类的继承关系如下：

```
java.lang.Object
+--java.awt.Component
+--java.awt.Container
+--javax.swing.JComponent
+--javax.swing.JTabel
```

表格类的构造方法如下。

- JTable()：使用系统默认的模型创建一个 JTable 实例。
- JTable(int numRows,int numColumns)：创建一个使用 DefaultTableModel 指定行、列的空表格。
- JTable(Object[][] rowData,Object[][] columnNames)：创建一个显示二维数据的表格。
- JTable(TableModel dm)：创建一个指定数据模式和默认字段模式的 JTable 实例。
- JTable(TableModel dm,TableColumnModel cm)：创建一个指定数据模式和字段模式的 JTable 实例。
- JTable(TableModel dm,TableColumnModel cm,ListSelectionModel sm)：创建一个指定数据模式、字段模式与选择模式的 JTable 实例。
- JTable(Vector rowData,Vector columnNames)：创建一个以 Vector 为数据源，并显示行名称的 JTable 实例。

13.3.7 树(JTree)

树(JTree)中特定的节点可以由 TreePath 标识或由其显示行标识。当展开某一个节点的所有祖先时，将显示该节点，折叠节点是隐藏位于折叠祖先下面的节点。可查看节点的父类节点都是可以展开的，但是它们可以显示也可以不显示。显示节点必须是可查看的并且位于显示区域。

与显示相关的方法如下。

- isRootVisible()：返回树的根节点是否显示。
- setRootVisible()：设置是否显示树的根节点。
- scrollPathToVisible()：确保展开所有的路径。
- scrollRowToVisible()：按行滚动标识的条目，直到显示出来。
- getVisibleRowCount()：返回显示区域的显示行的数目。
- setVisibleRowCount()：设置显示区域中显示行的数目。

与可查看相关的方法如下。

- isVisible()：返回当前路径查看标识。
- makeVisible()：设置当前路径的查看标识。

树的构造方法如下。

- JTree()：创建一个空节点的树。
- JTree(Hashtable<?,?> value)：创建一个由 Hashtable 元素构成节点的 JTree，它不显示根。
- JTree(Object[] value)：创建一个指定数组的元素作为不给显示的新的根节点的子节点的树实例。
- JTree(TreeModel newModel)：创建一个使用指定数据模型，并显示根结点的树实例。
- JTree(TreeNode root)：创建一个指定 TreeNode 作为其根，并显示根节点的树实例。
- JTree(TreeNode root, boolean asksAllowsChildren)：创建一个指定 TreeNode 作为其根，并指定根节点的显示方式的树实例。

下面是一个演示树的示例代码如下。

```
01  public class JTreeTest {
02      public static void main(String args[]) {
03          JFrame f = new JFrame("树示例");                    //创建窗体
04          //设置用户在此窗体上发起 "close" 时默认执行的操作
05          f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
06          f.getContentPane();
07          Box box = Box.createHorizontalBox();                //创建 Box 类对象
08          JTree tree1 = new JTree();                          //创建树
09          tree1.putClientProperty("JTree.lineStyle", "Angled");      //向此组件添加任意的键/值
10          JScrollPane scrollPane1 = new JScrollPane(tree1);   //创建一个滚动面板
11          tree1.setAutoscrolls(true);
12          JTree tree2 = new JTree();
13          JScrollPane scrollPane2 = new JScrollPane(tree2);
14          //向 Box 容器添加滚动面板
15          box.add(scrollPane1, BorderLayout.WEST);
```

```
16                box.add(scrollPane2, BorderLayout.EAST);
17                f.getContentPane().add(box, BorderLayout.CENTER);
18                f.setSize(300, 240);
19                f.setVisible(true);
20            }
21    }
```

以上程序中，首先创建窗体并设置(第 3 行)。创建树并添加值(第 8~9 行)。创建滚动面板并添加树(第 10 行)。向 Box 容器中添加滚动面板(第 15~16 行)。程序运行结果如图 13-6 所示。

图 13-6　树示例效果图

13.3.8　文本框(JTextField)与文本区(JTextArea)

文本框具有文本输入和编辑的功能，文本框组件用于获取到用户所输入的文本。除此之外还有文本区，文本区和文本框的区别是，前者可以输入多行文本，而文本框只接受单行文本的输入。

实现文本框功能的类是 JTextField，其中提供了多个方法，可以设置输入的文本字符长度限制。密码框和文本框的外观一样，并且也继承自 JTextField 类，密码框只提供专门的密码输入，输入内容不能直接显示，在密码框中以星号或其他形式的符号显示在上面。下面的示例是一个登录框，其中包括用户名和密码，代码如下。

```
01    public class JTextFieldDemo {
02        public static void main(String args[]) {
03            JFrame jframe = new JFrame("文本框和密码框示例");
04            Container contentPane = jframe.getContentPane();   // 返回窗体容器
05            contentPane.setLayout(new BorderLayout());         // 设置窗体容器布局
06            JPanel jpanel = new JPanel();                      // 声明面板容器
07            jpanel.setLayout(new GridLayout(3, 2));            // 设置面板布局
08            jpanel.setBorder(BorderFactory.createTitledBorder("请输入你的登陆信息"));
09            // 设置面板边界
10            JLabel lable1 = new JLabel("用户名：", JLabel.CENTER);      // 创建标签
11            JLabel lable2 = new JLabel("密码：", JLabel.CENTER);
12            JTextField t1 = new JTextField(10);              // 创建单行文本框，其长度为 10
13            JPasswordField t2 = new JPasswordField(10);
14            jpanel.add(lable1);              // 将标签及单行文本框依次添加到面板中
15            jpanel.add(t1);
16            jpanel.add(lable2);
```

```
17                jpanel.add(t2);
18                contentPane.add(jpanel);
19                jframe.setSize(500, 200);
20                jframe.setVisible(true);
21                jframe.addWindowListener(new WindowAdapter()  // 关闭事件
22                        {
23                            public void windowClosing(WindowEvent e) {
24                                System.exit(0);
25                            }
26                        });
27        }
28    }
```

以上程序中，首先创建窗体容器、面板容器并进行初始化设置(第 3~7 行)。创建标签和文本框，并将其添加到面板中(第 10~17 行)。程序运行结果如图 13-7 所示。

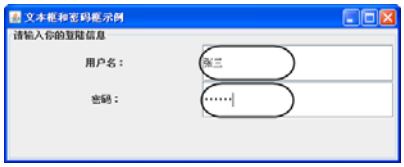

图 13-7　文本框示例效果

文本框中只能够输入单行的文字，文本区就解决了这个问题，在文本区中可以输入多行文字，该组件是由类 JTextArea 负责。下面是一个文本区的示例，代码如下。

```
01    public class TextAreaTest {
02        public static void main(String args[]) {
03            JFrame jframe = new JFrame("文本区示例");
04            Container contentPane = jframe.getContentPane();   // 返回窗体容器
05            contentPane.setLayout(new BorderLayout());         // 设置窗体容器布局
06            JPanel jpanel = new JPanel();                      // 声明面板容器
07            JTextArea jta = new JTextArea(10,20);              // 创建单行文本框，其长度为 10
08            jpanel.add(jta);
09            contentPane.add(jpanel);
10            jframe.setSize(500, 200);
11            jframe.setVisible(true);
12            jframe.addWindowListener(new WindowAdapter()  // 关闭事件
13                    {
14                        public void windowClosing(WindowEvent e) {
15                            System.exit(0);
16                        }
17                    });
18        }
19    }
```

以上程序首先创建容器并设置(第 3~6 行)。然后创建文本区，并设置单行长度(第 7 行)。最后设置窗口的大小以及窗口可见(第 10~11 行)。程序运行效果如图 13-8 所示。

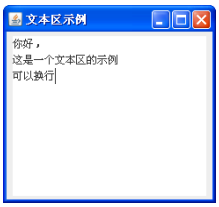

图 13-8　文本区示例效果

13.4 布局管理器

布局管理器可以对窗口中的组件做有效的布局。Java 中的容器和布局管理器相分离，也就是容器只是把组件放置进来。至于如何放置就要使用到布局管理器了。布局管理器主要包括有 BroderLayout、FlowLayout、GridLayout 等。本节就主要介绍一下常用的布局管理器。

13.4.1 FlowLayout 布局管理器

FlowLayout 布局管理器也叫做流式布局管理器，使用它可以将组件从左到右、从上到下进行排放。并且在默认的情况下尽可能地选用居中放置，总会根据自身的大小来进行自动排列，而不需要用户进行任何明确的操作。

流式布局管理器的特点是在一行上水平地进行排列，直到该行没有空间为止，然后就会重新换行进行排列。当用户缩小容器时，如果长度小于当前摆放的组件长度，则将多余的组件切换到下一行中进行排列。如果此时将容器放大，则会将第二排的组件重新放置到第一排中多出的空间中。

下面是一个 FlowLayout 布局管理器的示例。

```
01  public class FlowLayoutTest {
02      public FlowLayoutTest() {
03          JFrame f = new JFrame();
04          Container contentPane = f.getContentPane();
05          contentPane.setLayout(new FlowLayout());//设置容器的布局方式为 FlowLayout
06          contentPane.add(new JButton("第一个"));      //向容器中添加第一个按钮
07          contentPane.add(new JButton("第二个"));      //向容器中添加第二个按钮
08          contentPane.add(new JButton("第三个"));      //向容器中添加第三个按钮
09          contentPane.add(new JButton("第四个"));      //向容器中添加第四个按钮
10          contentPane.add(new JButton("第五个"));      //向容器中添加第五个按钮
11          contentPane.add(new JButton("第六个"));      //向容器中添加第六个按钮
12          f.setTitle("FlowLayout 示例");
13          f.setSize(400, 220);                         //设置框架的大小
14          f.setVisible(true);                  //将框架设置为可见状态
15          f.addWindowListener(new WindowAdapter() {//创建一个监听
16              public void windowClosing(WindowEvent e) {
```

```
17                          System.exit(0);
18                      }
19                  });
20          }
21          public static void main(String[] args) {
22                  new FlowLayoutTest();
23          }
24  }
```

以上程序中，首先创建容器并设置容器布局格式为流式布局(第 3~5 行)。在容器中创建 6 个按钮(第 6~11 行)。最后对窗口进行设置(第 13~15 行)。程序运行效果对比如图 13-9 和图 13-10 所示。

图 13-9　FlowLayout 布局效果

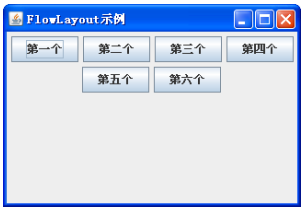

图 13-10　FlowLayout 效果对比

13.4.2　BorderLayout 布局管理器

BorderLayout 布局管理器又叫边界布局管理器，在 Java 中是最基础的布局管理器之一，而且也是比较常用的管理器，也是面板默认的布局管理器。

边界布局管理器的特点就是将整个面板分为 5 个部分，分别对应东、西、南、北、中。如果一个面板被设置成边界布局后，所有填入某一区域的组件都会按照该区域的空间进行调整，直到完全充满该区域。如果此时将面板的大小进行调整，则四周区域的大小不会发生改变，只有中间区域被放大或缩小。

下面是一个使用边界布局管理器的示例，代码如下。

```
01  public class BorderLayoutTest {
02          public BorderLayoutTest() {
03                  JFrame jf = new JFrame();
04                  Container contentPane = jf.getContentPane();
05                  //设置容器的布局方式为 BorderLayout
06                  contentPane.setLayout(new BorderLayout());
07                  contentPane.add(new JButton("东"), BorderLayout.EAST);      // 将按钮放到东侧
08                  contentPane.add(new JButton("西"), BorderLayout.WEST);      // 将按钮放到西侧
09                  contentPane.add(new JButton("南"), BorderLayout.SOUTH);     // 将按钮放到南侧
10                  contentPane.add(new JButton("北"), BorderLayout.NORTH);     // 将按钮放到北侧
11                  contentPane.add(new JLabel("中", JLabel.CENTER),
12                                  BorderLayout.CENTER);                       // 将标签放到中间
13                  jf.setTitle("BorderLayout 布局管理器示例");                 // 设置标题
14                  jf.pack();
```

```
15                  jf.setVisible(true);
16                  //对一个窗口进行关闭操作的事件
17                  jf.addWindowListener(new WindowAdapter() {
18                      public void windowClosing(WindowEvent e) {
19                          System.exit(0);
20                      }
21                  });
22              }
23              public static void main(String[] args) {
24                  new BorderLayoutTest();
25              }
26      }
```

以上程序中，首先创建容器并设置布局格式为边界布局(第 3~6 行)。在容器中添加 5 个按钮，分别设置为东、西、南、北、中(第 7~12 行)。最后对窗口属性进行设置(第 13~15 行)。程序运行效果对比如图 13-11 和图 13-12 所示。

图 13-11　边界布局管理器效果

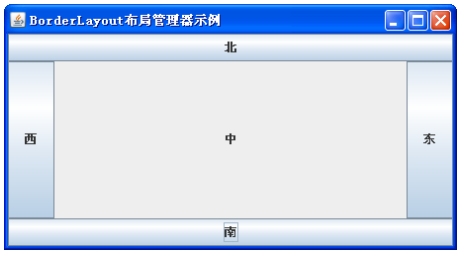

图 13-12　边界布局管理器效果对比

13.4.3　BoxLayout 布局管理器

BoxLayout 布局管理器又叫做箱式布局管理器，顾名思义，就相当于将一组组件放置到一个箱子中然后将箱子排成一列。用户还可以通过传入构造方法的参数来决定如何排列，分为横向和纵向两种。

创建箱式布局管理器时需要设置参数，用于选择横向布局或是纵向布局，选择横向布局时，组件的排列顺序是从左到右。选择纵向布局时，组件的排列顺序是从上到下。关于组件的大小设置，主要有以下 3 个方法。

- setPreperredSize(Dimension dim)：该方法用于设置首选大小。
- setMaximumSize(Dimension dim)：该方法用于设置最大值。
- setMinimumSize(Dimension dim)：该方法用于设置最小值。

使用横向布局，如果设置组件的大小，首先需要计算组件的最大高度，然后将所有的组件设置成整个高度，接下来要获取到所有组件的宽度，并进行宽度值的累加。

下面是一个横式布局管理器的示例，代码如下。

```
01      public class BoxLayoutTest {
02          public BoxLayoutTest() {
```

```
03              JFrame f = new JFrame();
04              Container contentPane = f.getContentPane();
05              Box baseBox = Box.createHorizontalBox();          //创建一个 Box 类对象
06              contentPane.add(baseBox);
07              baseBox.add(new JButton("按钮 1"));               //定义第一个按钮
08              baseBox.add(new JButton("按钮 2"));               //定义第二个按钮
09              f.setTitle("BoxLayout 示例");
10              f.setSize(new Dimension(200, 50));                //设置窗口的大小
11              f.setVisible(true);
12              f.addWindowListener(new WindowAdapter() {         //对窗口进行监听
13                  public void windowClosing(WindowEvent e) {
14                      System.exit(0);
15                  }
16              });
17          }
18          public static void main(String[] args) {
19              new BoxLayoutTest();
20          }
21      }
```

以上程序中，首先创建容器(第 3~4 行)。创建 Box 对象并初始化，添加到容器中(第 5~6 行)。在 Box 中添加两个按钮(第 7~8 行)。最后设置窗口属性(第 9~14 行)。程序运行结果如图 13-13 所示。

下面是纵向布局示例，代码如下。

```
01  public class BoxLayoutVerticalTest {
02      public BoxLayoutVerticalTest() {
03          JFrame f = new JFrame();
04          Container contentPane = f.getContentPane();
05          Box baseBox = Box.createVerticalBox();            //创建一个 Box 类对象
06          contentPane.add(baseBox);
07          baseBox.add(new JButton("按钮 1"));               //定义第一个按钮
08          baseBox.add(new JButton("按钮 2"));               //定义第二个按钮
09          f.setTitle("BoxLayout 示例");
10          f.setSize(new Dimension(200, 50));                //设置窗口的大小
11          f.setVisible(true);
12          f.addWindowListener(new WindowAdapter() {         //对窗口进行监听
13              public void windowClosing(WindowEvent e) {
14                  System.exit(0);
15              }
16          });
17      }
18      public static void main(String[] args) {
19          new BoxLayoutVerticalTest();
20      }
```

```
21    }
```

以上程序中，首先创建容器(第 3~4 行)。创建 Box 对象并初始化，添加到容器中(第 5~6 行)。在 Box 中添加两个按钮(第 7~8 行)。最后设置窗口属性(第 9~14 行)。程序运行结果如图 13-14 所示。

图 13-13　横向布局效果

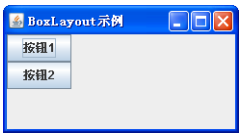

图 13-14　纵向布局效果

13.4.4　GridLayout 布局管理器

GridLayout 是网格布局管理器，该管理器负责将一个容器按照规则的形状分割成多个区域。对 GridLayout 的设置，可以根据横向或者纵向的放法，两者指定的方法分别是 setHgap 和 setVgap。而且可以直接在构造方法中指定，横向组件间隔宽度和纵向组件间隔宽度对应属性是 hgap 和 vgap。

下面是一个网格布局管理器的示例，代码如下。

```
01    public class GridLayoutTest    {
02        JPanel p1;
03        int i = 1;
04        JFrame f;
05        public GridLayoutTest() {
06            f = new JFrame();                                    // 创建一个 JFrame 的对象
07            Container contentPane = f.getContentPane();
08            contentPane.setLayout(new GridLayout(2, 1));//设置容器的布局方式为 GridLayout
09            p1 = new JPanel();
10            p1.setLayout(new GridLayout(2,3));
11            p1.add(new JButton("按钮 1"));                  // 将第一个按钮添加到对象 p1 中
12            p1.add(new JButton("按钮 2"));                  // 将第二个按钮添加到对象 p1 中
13            p1.add(new JButton("按钮 3"));                  // 将第三个按钮添加到对象 p1 中
14            p1.add(new JButton("按钮 4"));                  // 将第四个按钮添加到对象 p1 中
15            p1.add(new JButton("按钮 5"));                  // 将第五个按钮添加到对象 p1 中
16            p1.add(new JButton("按钮 6"));                  // 将第六个按钮添加到对象 p1 中
17            contentPane.add(p1);
18            f.setTitle("CardLayout 示例");
19            f.pack();
20            f.setVisible(true);                                  // 将框架设置为可见状态
21            f.addWindowListener(new WindowAdapter() {       //添加窗口事件监听
22                public void windowClosing(WindowEvent e) {
23                    System.exit(0);
24                }
25            });
26        }
```

```
27          public static void main(String[] args) {
28              new GridLayoutTest();
29          }
30      }
```

以上程序中，首先声明容器以及整型 i(第 2~4 行)。在构造方法中对容器进行初始化并设置布局(第 6~9 行)。在面板中添加 6 个按钮(第 11~16 行)。最后设置窗口的属性(第 18~23 行)。运行结果如图 13-15 所示。

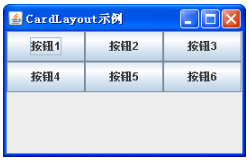

图 13-15　网格布局管理

13.5 事件处理

事件处理是图形界面和用户进行交互的重要组成部分，Java 中的事件处理机制主要包括有事件源、事件和事件处理器 3 个部分。首先要做的是为事件注册相对应的事件处理器，并制定事件，然后由事件处理器获取后进行相应的事件处理。

13.5.1　事件监听器

事件监听器是监听所触发事件的对象，其中包含有对事件发生后的事件处理操作。对于不同的事件，Java 中也定义了所相应的事件监听器接口。如下是几个比较常用的事件监听器接口。

- ActionListener：接收操作事件的监听器接口。
- AdjustmentListener：接收调整事件的监听器接口。
- FocusListener：接收组件上的键盘焦点事件的监听器接口。
- InputMethodListener：接收输入方法事件的监听接口。
- KeyListener：用于接收键盘事件的监听接口。
- MouseListener：接收组件上的鼠标事件(包括按下、单击、进入或者离开)的监听器接口。
- MouseMotionListener：接收组件上的鼠标移动事件的监听接口。
- MouseWheelListener：接收组件上的鼠标滚轮事件的监听接口。
- TextListener：接收文本事件的监听器接口。
- WindowListener：接收窗口事件的监听接口。

使用监听器，首先要定义监听器类，并实现相应的监听器接口。然后要在组件上使用 addXxxxListener 的方式为组件添加事件监听，然后设置相应的事件处理方法。当组件中的事件触发后，就会根据所添加的事件处理方法进行事件处理。

13.5.2　事件适配器

前面介绍了事件监听器，都是以实现事件监听器的接口方式进行定义的。而且第 10 章中也

介绍过，实现接口，就要实现接口中的所有方法。Java 中针对每个事件接听器接口，系统定义了相应的实现类，并称为事件适配器。只需要继承事件适配器并覆盖几个必要的方法就可以了。这样一来也使得代码变得更加简洁。比较常用的事件监听器有如下几个。

- ComponentAdapter：接收组件事件的抽象适配器。
- ContainerAdapter：接收容器事件的抽象适配器。
- FocusAdapter：接收键盘焦点事件的抽象适配器。
- KeyAdapter：接收键盘事件的抽象适配器。
- MouseAdapter：接收鼠标事件的抽象适配器类。
- MouseMotinAdapter：接收鼠标移动事件的抽象适配器类。
- WindowAdapter：接收窗口事件的抽象适配器类。

下面是一个关闭窗口的示例，单击窗口中的时，就会关闭窗口。代码如下：

```
01  public class WindowAdapterTest {
02
03      public static void main(String[] args) {
04
05          Frame fra=new Frame("窗口关闭示例");
06          fra.setSize(200,200);
07          fra.setVisible(true);
08          fra.addWindowListener(new WindowClose());
09      }
10  }
11  class WindowClose extends WindowAdapter{
12  public void windowClosing(WindowEvent e)
13      {
14      System.exit(1);
15  }
16  }
```

以上程序中，创建一个窗口并设置属性(第 5~7 行)。为窗口添加事件监听(第 8 行)。设置监听实现方法(第 11~14 行)。程序运行结果如图 13-16 所示。

图 13-16　窗口关闭示例

13.5.3　事件

事件就是触发一个组件所产生的动作，在 Swing 中，有很多的事件，例如鼠标事件、焦点

事件等，每一个事件类都会与一个事件类接口相对应，并且由事件所引起的动作都会存放在接口需要实现的方法中。

1. 鼠标事件

鼠标事件是由 MouseEvent 负责，而鼠标监听器则有两种，分别是 MouseListener 和 MouseMotionListener。其中 MouseListener 负责鼠标的按下、抬起、进入某一区域。当组件注册了鼠标监听器后，如果组件发生以上的动作事件后，就会激活相应的事件处理方法。

鼠标事件可以获取到鼠标的光标进入按钮、离开按钮、单击按钮、按下按钮等多种事件。下面的示例就是一个获取到鼠标光标对按钮的一些操作事件。

```
01  public class MouseTest extends WindowAdapter
02                      implements MouseListener {
03      JFrame jfarme = null;
04      JButton button1 = null;
05      JLabel label = null;
06      public MouseTest() {
07          jfarme = new JFrame("鼠标事件示例");              //创建一个框架
08          Container contentPane = jfarme.getContentPane();  //获得一个窗口容器
09          contentPane.setLayout(new GridLayout(2, 1));//设置布局方式
10          button1 = new JButton("按钮");                    //创建一个按钮
11  label = new JLabel("起始状态，还没有鼠标事件", JLabel.CENTER);//创建一个文本标签
12          button1.addMouseListener(this);                   //为按钮添加事件监听
13          contentPane.add(label);
14          contentPane.add(button1);
15          jfarme.pack();                                    //将窗口调节到适当的大小
16          jfarme.show();                                    //显示窗口
17          jfarme.addWindowListener(this);                   //添加窗口事件监听
18      }
19      public void mousePressed(MouseEvent e) {              //激发按下鼠标事件
20          label.setText("你已经按下鼠标按钮");
21      }
22      public void mouseReleased(MouseEvent e) {             //激发释放鼠标按键
23          label.setText("你已经抬起鼠标按钮");
24      }
25      public void mouseEntered(MouseEvent e) {              //鼠标进入组件时调用
26          label.setText("光标已经进入按钮");
27      }
28      public void mouseExited(MouseEvent e) {                   //鼠标离开组件时调用
29          label.setText("光标已经离开按钮");
30      }
31      public void mouseClicked(MouseEvent e) {              //鼠标在组件上单击时调用
32          label.setText("你单击按钮");
```

```
33          }
34          public void windowClosing(WindowEvent e) {
35              System.exit(0);
36          }
37          public static void main(String[] args) {
38              new MouseTest();
39          }
40      }
```

以上程序中，首先创建窗口，并在窗口中创建标签和按钮(第 3~5 行)。为按钮添加事件监听(第 12 行)。设置触发按钮事件所产生的方法(第 19~34 行)。程序运行效果对比如图 13-17~图 13-19 所示。

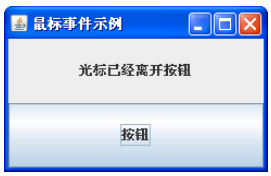

图 13-17　光标离开按钮

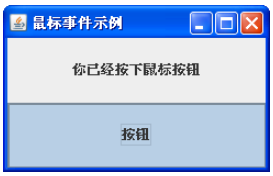

图 13-18　按下按钮

图 13-19　单击按钮

MouseMotionListener 主要负责鼠标的拖动事件处理。当用户在注册了 MouseMotionListener 的组件上拖动鼠标时，就激活 mouseDragged(MouseEvent e)方法，如果用户此时没有按下鼠标而是移动鼠标时，就会激活 mouseMoved(MouseEvent e)方法。

2. 键盘事件

键盘事件是用于处理在键盘上所输入的信息，表示键盘事件的类是 KeyEvent，该类可以获取到产生键盘事件的事件源，而且还可以获取到键盘输入按键的信息。

```
01  public class KeyTest extends KeyAdapter
02              implements ActionListener {
03      JFrame jframe = null;          //声明一个 JFrame 类对象
04      JLabel label = null;           //声明一个 JLabel 类对象
05      JTextField jtext = null;       //声明一个 JTextField 类对象
06      String key = "";
07      public KeyTest() {
08          jframe = new JFrame("键盘监听示例"); // 为这个 JFrame 设置一个标题
09          Container contentPane = jframe.getContentPane();
10          contentPane.setLayout(new GridLayout(3, 1));
11          label = new JLabel();              // 创建一个标签对象
12          jtext = new JTextField();
13          jtext.requestFocus();
14          jtext.addKeyListener(this);
15          JButton b = new JButton("清除"); // 创建一个按钮
16          b.addActionListener(this);
17          contentPane.add(label);
```

```
18                  contentPane.add(jtext);
19                  contentPane.add(b);
20                  jframe.pack();
21                  jframe.show();
22                  jframe.addWindowListener(new WindowAdapter() {        //添加窗口事件监听
23                      public void windowClosing(WindowEvent e) {
24                          System.exit(0);
25                      }
26                  });
27              }
28              public void actionPerformed(ActionEvent e) {
29                  key = "";
30                  label.setText("");
31                  jtext.setText("");
32                  jtext.requestFocus();
33              }
34              public void keyTyped(KeyEvent e) {
35                  char c = e.getKeyChar();       //获取事件源上的字符
36                  if (c == 'o') {                //如果输入字母 o，就会产生新窗口
37                      JFrame newF = new JFrame("这是输入 o 后弹出的新窗口");
38                      newF.setSize(200, 200);
39                      newF.show();
40                  }
41                  key = key + c;
42                  label.setText(key);
43              }
44              public static void main(String[] args) {
45                  new KeyTest();
46              }
47      }
```

以上程序中，首先创建窗口、标签、文本框和按钮对象(第 3~5 行)。将容器、标签、文本框和按钮进行初始化，并添加相应事件监听(第 8~22 行)。对添加的事件监听进行处理(第 22~42 行)。程序运行结果如图 13-20 所示。

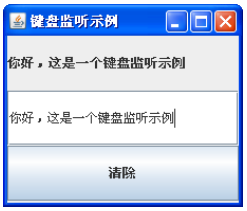

图 13-20　键盘事件效果

3. 焦点事件

鼠标单击到某一个按钮上的图片时，这个按钮就获得了焦点。例如，鼠标单击窗口的标题栏，就说明这个标题栏获得了焦点。单击这个窗口中的任意一个组件时，这个组件就获得了焦点。并且不同的组件获得焦点后的显示是不同的，例如在文本框获取到焦点时，鼠标的光标就会不停的闪动。

一个窗口中的组件，在同一时间只能够有一个组件获得焦点。如果单击了另一个组件，此时该组件就会失去焦点。在 Java 中与焦点相关的方法如下。

- FocusOwner：焦点拥有者，也就是持有焦点的组件。
- FocusWindow：焦点窗口，含有焦点拥有者的窗口。
- ActiveWindow：活动窗口，包含有焦点的框架或者对话框。

除去弹出对话框外，大部分情况下焦点窗口和活动窗口是一个窗口，因为弹出对话框不是放置在一个窗体中的组件获取焦点，所以它的焦点窗口和活动窗口不是同一个。

13.6 举例

在本章前面介绍了图形用户界面的基本操作，本小节就以一个示例，将前面所介绍到的内容贯穿起来，该示例是一个科学计算器，其中可以计算正弦余弦，开 2 次方根等操作。其操作界面如图 13-21 所示。

图 13-21　计算器示例

13.6.1　界面设计

一个完整的计算器是从界面开始设计，首先需要确定界面的大小以及显示的位置，然后就要在界面中设置一些按钮，并依次设置按钮的布局格式。在此选用的是 FlowLayout 布局格式，并将界面分为 3 部分。第一部分只放置一个文本框，用于显示用户所选择的数字和运算得到的结果。第二部分则是要放置 0~9 的显示数字，用户可以单击数字进行操作。第三部分则是放置计算器的各种运算符，用户可以在此区域中选择所要进行的运算。在设计完各个按钮后，需要将所有的运算符按钮添加事件监听，界面的实现方法代码如下。

```
01    public void disp() {
```

```
02              list = new JFrame("我的计算器");
03              list.setSize(360, 230);
04              list.setLocation(400, 300);
05              list.setBackground(Color.LIGHT_GRAY);
06              list.setLayout(new FlowLayout(FlowLayout.CENTER));
07              list.setResizable(false);
08              show = new JTextField(31);
09              show.setText("0");
10              show.setHorizontalAlignment(SwingConstants.RIGHT);
11              show.setEditable(false);
12              list.add(show);
13
14              Panel dispMain = new Panel();
15              list.add(dispMain);
16              dispMain.setLayout(new GridLayout(1, 2, 10, 10));
17              Panel dispLeft = new Panel();
18              dispMain.add(dispLeft);
19              dispLeft.setLayout(new GridLayout(4, 3, 3, 3));
20              Panel dispRight = new Panel();
21              dispMain.add(dispRight);
22              dispRight.setLayout(new GridLayout(4, 3, 3, 3));
23              for (l = 9; l >= 0; l--) {
24                  b[l] = new JButton(String.valueOf(l));
25                  b[l].setForeground(Color.BLUE);
26                  dispLeft.add(b[l]);
27                  b[l].addActionListener(this);
28              }
29              jia = new JButton("+");
30              jian = new JButton("-");
31              cheng = new JButton("*");
32              chu = new JButton("/");
33              xy = new JButton("x^y");
34              cos = new JButton("cos");
35              sin = new JButton("sin");
36              ln = new JButton("ln");
37              ce = new JButton("CE");
38              equ = new JButton("=");
39              bfh = new JButton("%");
40              point = new JButton(".");
41              zf = new JButton(" +/- ");
42              sqrt = new JButton("sqrt");
43              dispRight.add(chu);
44              dispRight.add(sqrt);
```

```
45              dispRight.add(ln);
46              dispRight.add(cheng);
47              dispRight.add(sin);
48              dispRight.add(bfh);
49              dispRight.add(jian);
50              dispRight.add(cos);
51              dispRight.add(ce);
52              dispRight.add(jia);
53              dispRight.add(xy);
54              dispRight.add(equ);
55
56              dispLeft.add(zf);
57              dispLeft.add(point);
58              ce.addActionListener(this);
59              jia.addActionListener(this);
60              jian.addActionListener(this);
61              cheng.addActionListener(this);
62              chu.addActionListener(this);
63              equ.addActionListener(this);
64              point.addActionListener(this);
65              zf.addActionListener(this);
66              sqrt.addActionListener(this);
67              bfh.addActionListener(this);
68              sin.addActionListener(this);
69              cos.addActionListener(this);
70              xy.addActionListener(this);
71              list.addWindowListener(this);
72              ln.addActionListener(this);
73              list.setVisible(true);
74          }
```

以上程序中，首先创建所使用到的组件并进行初始化(第 2~42 行)，然后为所使用到的按钮添加时间监听(第 58~72 行)。

13.6.2 运算设计步骤

运算设计包括计算器中所出现的所有的运算，如加、减、乘、除、对数运算、正弦运算和余弦运算等。

1. 加减乘除运算

计算器中，加减乘除运算可以说是最基本的运算，所有的计算器都要实现这个运算功能。此处实现的加减乘除运算，以加法为例，首先是要获取到用户所输入的第一个数字，并进行保存，接下来判断用户所选择的运算符，并获取用户所输入的第二个数据，并依次进行累加，当

用户单击“=”时，将所输入的数据运算结果显示到文本框中。其中主要方法代码如下。

```
01  if (e.getSource() == jia ) {                    // 加运算，可连加
02                  if (j == 0) {
03                      sum = getValue;
04                  } else if (action == 1) {
05                      sum += getValue;
06                  }
07                  setSum();
08                  j++;
09                  p = 0;
10                  i = 0;
11                  action = 1;
12              } else if (e.getSource() == jian) {          // 减运算，可连减
13                  if (j == 0) {
14                      sum = getValue;
15                  } else if (action == 2) {
16                      sum -= getValue;
17                  }
18                  setSum();
19                  j++;
20                  p = 0;
21                  i = 0;
22                  action = 2;
23              } else if (e.getSource() == cheng) {             // 乘运算，可连乘
24                  if (j == 0) {
25                      sum = getValue;
26                  } else if (action == 3) {
27                      sum *= getValue;
28                  }
29                  setSum();
30                  j++;
31                  p = 0;
32                  i = 0;
33                  action = 3;
34              } else if (e.getSource() == chu) {          // 除运算，可连除
35                  if (j == 0)
36                      sum = getValue;
37                  else if (action == 4) {
38                      sum /= getValue;
39                  }
40                  setSum();
41                  j++;
42                  p = 0;
```

```
43                  i = 0;
44                  action = 4;
45              } else if (e.getSource() == equ) {    // 等号，运算最后一个操作数
46                  switch (action) {
47                  case 1:
48                      show.setText(String.valueOf(sum += getValue));
49                      break;
50                  case 2:
51                      show.setText(String.valueOf(sum -= getValue));
52                      break;
53                  case 3:
54                      show.setText(String.valueOf(sum *= getValue));
55                      break;
56                  case 4:
57                      show.setText(String.valueOf(sum /= getValue));
58                      break;
59                  case 5:
60                      show.setText(String.valueOf(sum=((int)sum)^((int)getValue)));
61                      break;
62              }
```

在以上程序中，分别实现的是基本的加减乘除方法。以加法为例，首先判断是否单击的是该按钮(第 1 行)。如果是第一次按下，则将值赋值给 sum，否则将 sum 和当前按下值的和重新赋值给 sum(第 2 行~6 行)。调用 setSum 方法，在计算器文本框中显示据结果(第 7 行)。最后将 j 的值累加，i 和 p 的值设置为 0，action 的值设置为 1(第 8~11 行)。

2. 正弦运算和余弦运算

正弦运算和余弦运算首先需要获取到用户所输入的数据，并将其保存，然后获取到用户单击“sin”或“cos”时，调用所对应的方法，调用 Math 类的 sin()或者 cos()方法，最后调用 setSum()方法将结果输出到文本框中。正弦运算和余弦运算对应的方法代码如下。

```
01  if(e.getSource()==cos){                    //余弦运算
02                  sum=Math.cos(getValue);
03                  setSum();
04                  i = 0;
05              }
06              if(e.getSource()==sin){          //正弦运算
07                  sum=Math.sin(getValue);
08                  setSum();
09                  i = 0;
10              }
```

以上程序中，获取到用户所按下的数字，并调用 Math 类中的 cos 或 sin 方法进行计算(第 2

行和第 7 行)。然后调用 setSum 方法显示到文本框中。

用于显示结果的方法 setSum()代码如下。

```
01    public void setSum() {                    // 用于将结果显示
02            show.setText(String.valueOf(sum));
03            String s = show.getText();
04            char a = s.charAt((s.length() - 1));
05            char b = s.charAt((s.length() - 2));
06            if (a == '0' && b == '.') { // 判断是否是整数，如果是整数，则去掉后面的小数点和 0
07                  show.setText(String.valueOf(Math.round(sum)));
08            }
09        }
```

以上程序中，将文本框内容设置为 sum(第 2 行)。获取到文本框中的数字(第 3 行)。将获取到的内容长度减 1 赋值给 a，该值减 2 赋值给 b，并判断是否为整数，如果是整数，则去掉后面的小数点，否则保留输出(第 4~7 行)。

3. 对数运算(ln)

对数运算首先需要获取到用户所输入的数据，并保存。对数 lnN 指的是 loge(N)，在 Math 类中有方法 log 所对应的就是 ln，可以在获取到用户数据后，调用该方法，并将数据传入，然后调用 setSum 方法输出显示(第 2 行)。该运算所对应的方法代码如下。

```
01    if(e.getSource()==ln){              // 对数运算
02            sum=Math.log(getValue);
03            setSum();
04            i = 0;
05        }
```

4. 计算器中的其他运算

除了上面所提到的几种运算，在计算器中还有几个比较不常用到的运算，如^、开 2 次方根、%等。其设计步骤和前面类似，也是首先获取到用户所输入的数据，然后根据用户所单击的运算符进行运算，最调用 setSum 方法来输出显示到文本框中。其方法代码如下。

```
01    if(e.getSource()==xy){              //^运算
02
03            if(j==0)
04                  sum=getValue;
05            else if(action==5)
06                  sum=((int)sum)^((int)getValue);
07            setSum();
08            j++;
09            i = 0;
```

```
10                  action=5;
11              }else if (e.getSource() == sqrt) {    // 开 2 次方根
12                  sum = Math.sqrt(getValue);
13                  setSum();
14                  i = 0;
15              }else if (e.getSource() == point) {          // 小数点，只能按一个小数点
16                  if (p == 0)
17                      show.setText(show.getText() + e.getActionCommand());
18                  p = 1;
19              } else if (e.getSource() == c || e.getSource() == ce) { // 清空与复位
20                  i = 0;
21                  j = 0;
22                  p = 0;
23                  sum = 0;
24                  action = 0;
25                  show.setText("0");
26              } else if (e.getSource() == bfh) {           // 百分号
27                  sum = getValue / 100;
28                  setSum();
29                  i = 0;
30              } else if (e.getSource() == zf) {            // 正负号切换，正号不显示
31                  String s = show.getText();
32                  char a = s.charAt(0);
33                  if (a == '-') {
34                      show.setText("");
35                      for (l = 1; l < s.length(); l++) { // 去掉负号
36                          show.setText(show.getText() + s.charAt(l));
37                      }
38                  } else if (getValue != 0) {                  // 加上负号
39                      show.setText("-" + s);
40                  }
41              }
42              for (l = 0; l < 10; l++) {                       // 0~9 数字键触发
43                  if (e.getSource() == b[l]) {
44                      if (i == 0)
45                          show.setText("");
46                      String s = show.getText();
47                      if (s.length() < slength)
48                          show.setText(show.getText() + e.getActionCommand());
49                      if (e.getSource() == b[0] && getValue == 0 && p == 0)
50                          show.setText("0");
51                      if (e.getSource() != b[0] && getValue == 0 && p == 0)
52                          show.setText(e.getActionCommand());
```

```
53                          i++;                    // i 用来标记数字键触发的状态
54                      }
55          }
```

以上程序中，包括有除第 1~3 小节介绍的基本运算之外的算法。开 2 次方根是调用相应的 sqrt 方法(第 12 行)。如果用户单击“CE”按钮，则表示清空，此时将所有的数值全部赋值为 0(第 19~25 行)。正负号判断，首先获取到字符串，然后判断第 0 个是否为“-”(第 33~39 行)。

13.7 本章习题

一、常规题

(1) Swing 的事件处理机制包括(　　)、事件和事件监听者。

(2) Java 事件处理包括建立事件源、(　　)和将事件源注册到监听器。

(3) 在 Swing 中，可以根据不同用户的习惯，设置不同的界面显示风格，Swing 提供了 3 种显示风格，分别是(　　)风格、(　　)风格和(　　)风格。

(4) Swing 的顶层容器有(　　)、JApplet、JWindow 和 JDialog。

(5) (　　)由一个玻璃面板、一个内容面板和一个可选择的菜单条组成。

二、问答题

(1) Swing 有哪些特点？

(2) Swing 中组件布局有哪几种方式？

(3) AWT 和 Swing 有什么区别？它们都怎么使用？

三、上机练习

(1) 使用 Swing 组件编写用户的登录窗口。

(2) 使用 Swing 组件显示一个整数的各位数字。

(3) 实现 Java 程序，要求单击鼠标画线，双击鼠标擦除所有画的线。