第 8 章　加法计算器

单元概述

本单元以项目为导向，介绍了编写 Java 图形用户界面程序的基本过程；通过本单元的学习，读者能够了解图形用户界面的基础知识，掌握如何使用容器组件和控制组件构建简单的图形用户界面，理解布局管理器的作用和用法，理解委托事件处理模型的工作原理，掌握动作事件的处理方法。

教学重点及难点

教学重点：框架、布局管理器、控制组件、委托事件处理模型、动作事件

教学难点：布局管理器、委托事件处理模型

8.1　项目任务

设计实现一个小型加法器，具体的界面效果如图 8-1 所示。当用户输入两个整数，并点击等号按钮时，加法器能够计算并显示结果。

图 8-1　加法器的运行效果图

8.2　项目分析

1.项目完成思路

根据项目任务描述的项目功能需求，本项目需要先构造一个图形用户界面，然后再实现交互功能，具体可以按照以下过程实现：

(1)构造界面外观

先构造一个图形用户界面窗体，在窗体上顺序摆放三个文本框，其中两个用于输入计算的整数，最后一个显示计算结果。在第三个文本框之前再摆放一个显示“＝”的按钮。

(2)实现交互功能

为按钮注册动作事件监听器。在监听器中读取前两个文本框中的整数,然后将计算的结果显示在第三个文本框中。

2.需解决的问题

(1)如何构造界面外观?

具体需解决的问题包括:如何构造矩形的窗体、文本框和按钮等界面组件?如何在窗体中按顺序摆放这些界面组件?

(2)如何实现交互功能?

具体需解决的问题包括:如何实现能够处理按钮动作事件的监听器?以及如何完成监听器的注册?

解决以上问题涉及的技术将在下一节技术准备中详细阐述。

8.3 技术准备

8.3.1 构造简单的图形界面

在程序设计中一项重要的任务就是设计和构造用户界面。用户界面是人与计算机交互的接口,用户界面的好坏对软件的应用有着直接的影响。图形用户界面(Graphics User Interfaces,GUI)使用图形化的方式为人与程序进行交互提供了一种友好的机制。与字符界面相比,图形界面省去了记忆各种命令的麻烦,界面美观,操作简便。因此,目前绝大多数应用软件都是采用图形界面的。

首先,介绍几个基本的概念。Java 中使用 GUI 组件构成图形用户界面,GUI 组件按照作用的不同可以分成两类:容器和控制组件。其中,容器是用来组织其他组件的单元,在容器中可以容纳多个其他的容器和控制组件。而控制组件的作用是完成与用户的交互功能,简单地说就是受用户控制的组件,它是组成用户界面的最小单位,它和容器不同,不能容纳其他的组件。总的来说,要构造一个图形界面,首先应该创建一个合适的容器,然后再在容器中按照特定的规则摆放各种满足交互需求的控制组件,这个特定的规则是通过为容器设置布局管理器来完成的。

下面以图 8-2 所示的图形界面为例来阐述如何构造一个图形用户界面的外观,以及容器、控制组件和布局管理器之间的关系。

图 8-2 简单图形界面

图 8-2 所示的图形界面主要由两部分组成,一部分是带标题栏的窗口(Java 中称为框架),另一部分是带有“确定”和“取消”字样的两个按钮。要构造这样一个界面,首先应该创建框架,然后在其上安放两个按钮即可。这里框架就是一种容器,而按钮就是一种控制组件。下面来详

细阐述一下如何创建框架以及如何将按钮添加到框架上。

1.创建框架

(1)示例代码

【例 8-1】实现简单框架界面程序。

```
// TestFrame.java
//引入程序中需要使用的 JFrame 类
import javax.swing.JFrame;
public class TestFrame
{
    public static void main(String[] args)
    {
        //创建一个标题为"图形界面程序"的框架
        JFrame frame=new JFrame("图形界面程序");
        //设置框架初始显示的大小
        frame.setSize(200,100);
        //设置框架在关闭的同时退出应用程序
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        //显示框架
        frame.setVisible(true);
    }
}
```

程序运行后显示如图 8-3 所示的图形界面。

图 8-3　例 8-1 的运行结果

(2)代码分析

例 8-1 先引入了 javax.swing 包中的 JFrame 类,然后在 main 方法中创建了类 JFrame 的一个实例,从图 8-3 可以看到,传递给构造方法的字符串参数显示在框架的标题栏上。这说明,Java 中可以通过 JFrame 创建框架。另外,程序通过 frame.setSize(200,100)方法设置框架的大小为 200 像素宽,100 像素高。如果注释掉这条语句,可以看到程序运行后只显示一个标题栏。然后程序通过 frame.setVisible(true)方法将创建的框架对象显示出来,也就是说,如果注释掉这条语句,将什么都看不到。最后,frame.setDefaultCloseOpration (JFrame.EXIT_ON_CLOSE)方法的作用是当关闭框架的同时退出应用程序。也就是说,如果没有这条语句,当关闭框架时应用程序仍在继续运行。

(3)知识点

①框架类 JFrame

其实,Java 中用于定义框架的类有两个:Frame 和 JFrame,它们都可以创建框架,只不过,前者定义在 java.awt 包中,属于 AWT 组件,而后者定义在 javax.swing 包中,属于 Swing 组

件。其中有这样一个规律，凡是以“J”开头的就是 Swing 组件。按照本章开始时所说，尽量要使用这种以“J”开头的 Swing 组件，本书实例中介绍的也全部都是 Swing 组件，不过 Java 中 Swing 组件有 250 多个，而且还在继续扩充，因此，本章只能介绍其中一小部分，更多的 Swing 组件还需要大家通过查阅其他参考书或者 JDK 帮助文档来进一步学习。

在 Java 中，框架是最常用的容器之一，它是编写图形化应用程序所使用的最外层容器。也就是说，编写一个图形化应用程序，先要创建一个框架，然后通过这个框架来组织其他的 GUI 组件。与其相似的容器还有 JApplet，它是编写 Java 小应用程序所使用的最外层容器。关于 Java 小应用程序留到本章最后再来介绍。

②JFrame 的构造方法

- public JFrame()：创建不带标题的框架。
- public JFrame(String title)：创建指定标题的框架。标题通过参数 title 指定。

③JFrame 的常用方法

- public void setSize(int width , int height)：设置框架的大小，width 为框架的宽度，height 为框架的高度，它们以像素为单位。
- public void setVisible(boolean b)：设置框架的可视状态。参数为 true 框架显示，参数为 false 框架隐藏。
- public void setDefaultCloseOpration(intoperation)：设置框架缺省的关闭操作，operation 的值如果设置为 JFrame. EXIT_ON_CLOSE，则在关闭框架时退出应用程序。

(4)举一反三

①创建并显示一个标题为“MyFrame”、宽为 400、高为 300 的框架。

②如果创建按钮的 AWT 组件类是 Button，那么对应的 Swing 组件类是什么？

2.添加组件

通过 JFrame 可以创建一个框架，但是，要创建的图形界面除了要有框架外，上面还要有两个按钮，那么如何创建按钮以及如何将按钮添加到框架上呢？

(1)示例代码

【例 8-2】 添加按钮没有使用布局管理器程序。

```
// TestFrame.java
import javax.swing.*;
public class TestFrame
{
    public static void main(String[] args)
    {
        //创建一个标题为“图形界面程序”的框架
        JFrame frame=new JFrame("图形界面程序");
        //创建一个标签为“Ok”的按钮，并添加在框架的内容窗格上
        frame.getContentPane().add(new JButton("确定"));
        //创建一个标签为“Cancel”的按钮，并添加在框架的内容窗格上
        frame.getContentPane().add(new JButton("取消"));
        //设置框架初始显示的大小
        frame.setSize(200,100);
```

```
        //实现在关闭框架的时候退出应用程序的功能
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        //显示框架
        frame.setVisible(true);
    }
}
```

程序运行后显示如图 8-4 所示的图形界面。

图 8-4　例 8-2 的运行结果

(2)代码分析

例 8-2 与例 8-1 相比只添加了两条语句。其中，frame.getContentPane()可以获取框架的内容窗格，这里要说明一下：在 Java 中，一般不直接在 JFrame 中添加组件，而是将组件添加到 JFrame 上附着的一层称为内容窗格的容器中，这种容器缺省是透明的，看上去就和直接加到 JFrame 中一样。另外，还可以看到：使用 new JButton("确定")可以创建一个显示为"确定"的按钮，同样，使用 new JButton("取消")可以创建一个显示为"取消"的按钮，然后，通过内容窗格的 add 方法可以将这个按钮添加到内容窗格上。最后，在显示框架的时候，就可以看到带有按钮组件的框架了，显示效果如图 8-4 所示。

但是从图 8-4 中可以看到，显示的效果并不是同时显示两个按钮，而是"取消"按钮填满了整个内容窗格，而且，"确定"按钮没有显示出来。这说明按钮在内容窗格中的摆放规则不满足需要，那如何改变这个规则呢？这就需要通过为容器设置布局管理器来设定容器的布局规则，那么如何构造布局管理器以及如何将布局管理器设置给容器呢？下面再看一个例子。

【例 8-3】添加按钮并使用布局管理器程序。

```
//TestFrame.java
import java.awt.*;
import javax.swing.*;
public class TestFrame
{
    public static void main(String[] args)
    {
        JFrame frame=new JFrame("图形界面程序");
        //将框架的内容窗格的布局管理器设置为 FlowLayout 布局
        frame.getContentPane().setLayout(new FlowLayout());
        frame.getContentPane().add(new JButton("确定"));
        frame.getContentPane().add(new JButton("取消"));
        frame.setSize(200,100);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

程序运行后显示如图 8-5 所示的图形界面。

图 8-5　例 8-3 的运行结果

例 8-3 与例 8-2 相比添加了一条语句，从这条语句中，可以看到 new FlowLayout()创建了一种名为 FlowLayout 的布局管理器，通过内容窗格容器的 setLayout 方法便可以将这种布局管理器设置给内容窗格了。其中 FlowLayout 的布局规则是，将组件按照从左到右的顺序依次摆放到容器中，如果一行放不下，就换一行。关于布局管理器的详细内容，下一小节再详细介绍。

(3)知识点

①获取框架内容窗格的方法

• public Container　getContentPane()：其中 Container 类是所有容器类的父类。

②所有容器常用的方法

• add(Component comp)：该方法可以将组件添加到容器中，其中 Component 类是所有组件类的父类。

• setLayout(LayoutManager　mgr)：该方法可以为容器设置一种布局管理器，其中 LayoutManager 是所有布局管理器类必须实现的接口，相当于所有布局管理器类的父类。

③构造图形用户界面的基本思路

根据界面设计确定需要的容器，然后通过 setLayout 方法为容器设置合适的布局管理器，最后通过 add 方法将组件添加到容器中。这里需要注意的是容器中的组件包括控制组件，也包括一部分容器。

(4)举一反三

创建并显示一个标题为“MyFrame”、宽为 400、高为 300 的框架。并在框架上从左到右摆放三个按钮“Button1”、“Button2”和“Button3”。

8.3.2　布局管理器

Java 为了实现良好的平台无关性，它不像其他语言那样使用像素来排列 GUI 组件，而是使用一种抽象的布局管理器来安排，上节中提到的 FlowLayout 就是一种布局管理器。在 Java 中有很多种布局管理器，其中 java. awt 包中定义了五个基本的布局管理器：FlowLayout、BorderLayout、GridLayout、CardLayout 和 GridBagLayout。这些类实现了 LayoutManager 接口，也就是说凡是实现了 LayoutManager 接口的类都可以作为布局管理器使用。这些类的实例便是一种布局管理器，每一种布局管理器都有不同的布局规则。它们通过容器的 setLayout 方法设置给容器后，容器就按照这种布局管理器的布局规则进行布局了。下面简单介绍一下五种基本布局管理器中的前三种。

1. FlowLayout

FlowLayout 是最简单的布局管理器，它的布局规则是按照添加的顺序将组件由左到右排列在容器中，一行排满后再换一行。下面通过一个例子来看一下它如何使用。

(1)示例代码

【例 8-4】应用 FlowLayout 布局管理器的程序。

```
// TestFlowLayout.java
import java.awt.*;
import javax.swing.*;
//定义一个 JFrame 的子类 TestFlowLayout,去扩展 JFrame
public class TestFlowLayout extends JFrame
{
    //构造方法
    public TestFlowLayout(String title)
    {
        //调用父类的构造方法,完成标题的初始化。
        super(title);
        //获取框架的内容窗格,其中 Container 类是所有容器类的父类
        Container  cp=this.getContentPane();
        //创建左对齐,水平间距 10 像素,垂直间距 30 像素的 FlowLayout 布局
        //并将其设置为内容窗格的布局管理器。
        cp.setLayout(new FlowLayout(FlowLayout.LEFT,10,30));
        //在内容窗格上添加按钮
        cp.add(new JButton("Button1"));
        cp.add(new JButton("Button2"));
        cp.add(new JButton("Button3"));
        cp.add(new JButton("Button4"));
    }
    public static void main(String[] args)
    {
        //创建这个 JFrame 子类的对象,框架的标题为“TestFlowLayout”
        TestFlowLayout frame=new TestFlowLayout("TestFlowLayout");
        frame.setSize(300,200);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```


程序运行后显示如图 8-6 所示的图形界面。

图 8-6　例 8-4 的运行结果

(2)代码分析

本例定义了一个 TestFlowLayout 类,它是 JFrame 的子类。在它的构造方法中完成了为这个扩展框架设置布局和添加组件的工作,然后在 main 方法中创建了这个扩展的框架,并把它显示出来,这种方式和上一节中创建框架的方式得到的效果是一样的,但这种方式更符合面向对象的思想,因此,以后创建图形用户界面时都是使用这种扩展原有框架类的方式完成的。

在本例中创建了一个 FlowLayout 的对象,并将其通过 setLayout 方法设置给了内容窗格,然后在内容窗格中通过 add 方法将按钮添加到内容窗格中,得到如图 8-6 所示的布局效果。如果改变框架的大小,这时会发现按钮的位置发生变化。

本例中使用的 Container 类是所有容器类的父类,因此,用其定义的引用 cp 可以引用所有类型的容器对象。

(3)知识点

FlowLayout 的构造方法:

①public FlowLayout(int align , int hGap , int vGap):三个参数的构造方法。参数 align 能够指定组件在容器中排列的对齐方式,它的值使用 FlowLayout 类中的三个静态常量 FlowLayout. RIGHT、FlowLaout. CENTER 和 FlowLaout. LEFT。参数 hGap 指定水平排列的组件之间的间距,vGap 指定垂直排列的组件之间的间距,它们以像素为单位。

②public FlowLayout(int align):一个参数的构造方法。参数 align 如同上一个构造方法中的 align,默认的水平间距和垂直间距是 5 个像素。

③public FlowLayout():无参的构造方法。默认的对齐方式是居中对齐,默认的水平间距和垂直间距是 5 个像素。

(4)举一反三

请编写一个 Application,其功能为:在其图形框架上按右对齐方式摆放三个按钮,三个按钮的标签分别显示为:“Button 1”,“Button 2”,“Button 3”。

2. BorderLayout

BorderLayout 布局管理器的布局规则是将容器分成五个部分:东区、南区、西区、北区和中央。组件添加到这五个区中。下面通过一个例子来看一下它如何使用。

(1)示例代码

【例 8-5】 应用 BorderLayout 布局管理器的程序。

```
// TestBorderLayout.java
import java.awt.*;
import javax.swing.*;
public class TestBorderLayout extends JFrame
{
    public TestBorderLayout(String title)
    {
        //调用父类的构造方法,完成标题的初始化。
        super(title);
        //获取框架的内容窗格,其中 Container 类是所有容器类的父类
```

```
        Container  cp=this.getContentPane();
        //设置内容窗格的布局为 BorderLayout
        cp.setLayout(new BorderLayout());
        //在内容窗格的不同区中添加按钮组件。
        cp.add(new JButton("East"),BorderLayout.EAST);
        cp.add(new JButton("South"),BorderLayout.SOUTH);
        cp.add(new JButton("West"),BorderLayout.WEST);
        cp.add(new JButton("North"),BorderLayout.NORTH);
        cp.add(new JButton("Center"),BorderLayout.CENTER);
    }
    public static void main(String[] args)
    {
        TestBorderLayout frame=new TestBorderLayout("TestBorderLayout");
        frame.setSize(300,200);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

程序运行后显示如图 8-7 所示的图形界面。

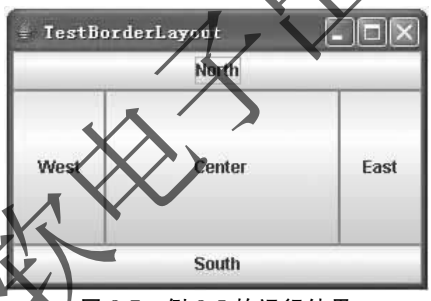

图 8-7　例 8-5 的运行结果

(2)代码分析

在本例中使用 BorderLayout 的对象作为内容窗格的布局管理器，这里可以看到将组件添加到容器的 add 方法与前面有所不同，在添加时可以指定组件放置的位置。在本例中，五个区域全部添加了组件，如果某个区域没有添加组件，则这个区域会被相邻区域占据。

【注意】如果在使用 add 方法时没有指明将组件添加到哪个区域，则缺省为中央区域，也就是说 add(component)与 add(component , BorderLayout.CENTER)是等价的。

(3)知识点

①BorderLayout 的构造方法

• public BorderLayout(int hGap , int vGap)：具有两个参数的构造方法。参数 hGap 指定水平排列的组件之间的间距，vGap 指定垂直排列的组件之间的间距，它们以像素为单位。

• public BorderLayout()：无参的构造方法。默认没有水平间距和垂直间距。

②添加组件的方法

• add(Component comp , int index)：其中，参数 comp 指定需要向容器中添加的组件，参

数 index 指定将组件添加到哪个区域中，它的取值可以是 BorderLayout. EAST、BorderLayout. SOUTH、BorderLayout. WEST、BorderLayout. NORTH 和 BorderLayout. CENTER，分别代表五个区域。

(4)举一反三

请编写一个 Application，其功能为：在其框架的内容窗格上安排两个按钮，分别命名为 Button1，Button2，内容窗格的布局为 BorderLayout 布局，并将两个按钮放置在内容窗格的东部区域和西部区域。

3. GridLayout

GridLayout 布局管理器的布局规则是将容器划分成若干行和若干列的网格，然后在这些大小相同的网格中按照添加的顺序从左到右排列组件，如果一行排满，就从下一行接着排。下面通过一个例子来看一下如何使用 GridLayout 布局管理器。

(1)示例代码

【例 8-6】 应用 GridLayout 布局管理器的程序。

```
// TestGridLayout.java
import java.awt.*;
import javax.swing.*;
//扩展 JFrame，定义 JFrame 类的子类
public class TestGridLayout extends JFrame
{
    //能够设置标题的构造方法，完成框架的初始化
    public TestGridLayout(String title)
    {
        //调用父类的构造方法，完成标题的初始化。
        super(title);
        //获取框架的内容窗格，其中 Container 类是所有容器类的父类
        Container cp=this.getContentPane();
        //设置内容窗格的布局为 GridLayout
        cp.setLayout(new GridLayout(2,3));
        //按顺序在内容窗格中添加不同的按钮组件
        cp.add(new JButton("Button1"));
        cp.add(new JButton("Button2"));
        cp.add(new JButton("Button3"));
        cp.add(new JButton("Button4"));
        cp.add(new JButton("Button5"));
    }
    public static void main(String[] args)
    {
        TestGridLayout frame=new TestGridLayout("TestGridLayout");
        frame.setSize(300,200);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

程序运行后显示如图 8-8 所示的图形界面。

图 8-8　例 8-6 的运行结果

(2)代码分析

在本例中使用 GridLayout 的对象作为内容窗格的布局管理器，通过 GridLayout 布局，将内容窗格划分成两行、三列的六个区域，在其中添加了 5 个按钮，与 FlowLayout 相比，相同点是，容器添加组件的方法一样，而且添加的顺序决定了组件在容器中的位置，不同点是组件的位置不会随着框架大小的变化而改变，而组件的大小会随着框架大小的变化而发生变化。

(3)知识点

GridLayout 的构造方法：

①public GridLayout(int rows , int columns , int hGap , int vGap)：该构造方法通过前两个参数指定将容器划分的行数和列数，通过后两个参数指定组件之间的水平和垂直间距，以像素为单位。

②public GridLayout(int rows , int columns)：该构造方法通过两个参数指定将容器划分的行数和列数，默认组件之间的水平和垂直间距为 0 像素。

③public GridLayout()：无参构造方法，构造的布局为在一行上添加若干个组件，默认组件之间的水平和垂直间距为 0 像素。

【注意】虽然 GridLayout 的构造方法可以指定容器划分网格的行数和列数，但是最终显示的列数却不一定是指定的列数，这与实际添加的组件也有关系。比如，构造方法中指定容器网格是 2 行 3 列，如果添加的组件是 4 个，那么显示的是两行两列，如果添加的组件是 7 个，那么显示的是两行 4 列。

(4)举一反三

请编写一个 Application，其功能为：在其框架的内容窗格上按照 3 行 4 列摆放 10 个相同大小的按钮，按钮的标题为“component1”“component2”等等，按钮之间的间隔为 5 个像素。

8.3.3　交互与事件处理

前面阐述了如何绘制简单的图形用户界面外观，但是这些程序对用户的操作没有反应，也就是说都没有交互能力。本节所阐述的就是如何为前面绘制的图形界面添加交互能力，使得程序用户在点击按钮的时候或者进行其他操作的时候程序能够做出反应。

1.事件处理模型

在 Java 中使用事件处理的方式来实现图形用户界面的交互，采用的事件处理机制称为委托事件处理模型。该模型规定的事件处理流程是这样的：当用户操作图形界面组件时，该组件会自动产生某种代表这种操作发生的信号，这个信号称为事件(event)，产生事件的组件称为事件源。然后一个称为监听器的对象会接收到这个事件，并对其进行处理。这样，当用户操作某个 GUI 组件时，只要有监听器监听这个组件产生的事件，那么程序就可以实现对用户行为的响应，也就是所谓的交互了。

要搞清楚这个模型具体如何使用，应该先弄清楚两个问题：

一是事件源与事件之间的具体关系，也就是有哪些事件源。它们在什么情况下会产生哪些类型的事件？知道了这些才能根据程序功能的需要来决定哪些事件需要处理，哪些可以忽略。

二是如何实现监听器？这是关键，因为对事件的处理是由监听器完成的。

首先，来解决一下第一个问题。

前面说过，当用户行为作用于 GUI 组件时，这些组件会产生某种事件(event)。产生事件的 GUI 组件称为事件源对象，所有的 GUI 组件都可以是事件源。事件是事件类的实例，它由事件源在用户行为的作用下自动产生，Java 中所有的事件类都是 Java.util.EventObject 的子类，它们的层次关系如图 8-9 所示。

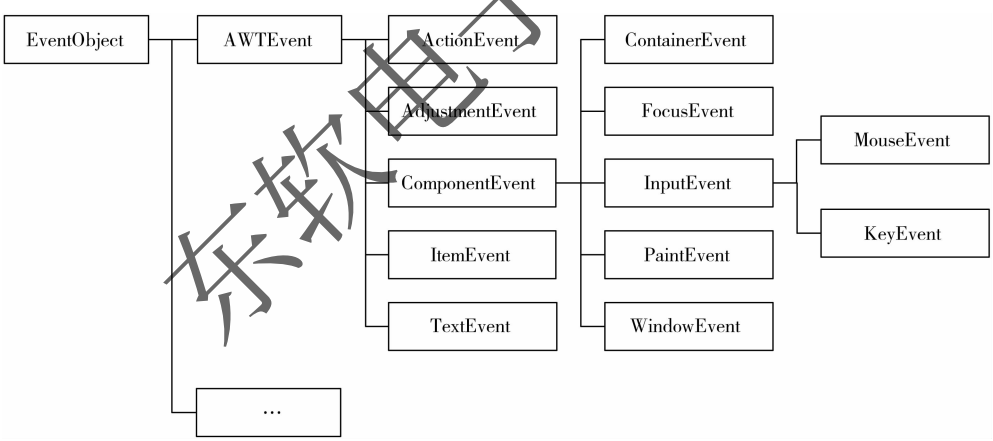

图 8-9　事件类的层次关系

EventObject 类的子类描述特定类型的事件，比如：ActionEvent 描述的是动作事件，WindowEvent 描述的是窗口事件，MouseEvent 描述的是鼠标事件等等。图中只列举了基本的 AWT 事件类，它们定义在 java.awt.event 包中，AWT 组件和 Swing 组件都可以产生这些类型的事件，省略号部分是扩展的事件类，这一部分数目比较多，详细情况可以参照 JDK 文档，它们主要定义在 javax.swing.event 包中，由 Swing 组件产生。表 8-1 列举了一些基本的用户行为、事件源和事件类型之间的关系。

表 8-1　　用户行为、事件源和事件类之间的关系

用户行为	事件源	事件类型
点击按钮	JButton(按钮)对象	ActionEvent
在文本域按下回车	JTextField(文本域)对象	ActionEvent
改变文本	JTextField(文本域)对象	TextEvent
改变文本	JTextArea(文本区)对象	TextEvent
窗口打开、关闭、最小化、还原或正在关闭	Window 及其子类对象	WindowEvent
在容器中添加或者删除组件	Container 的所有子类对象	ContainerEvent
组件移动、改变大小、隐藏或显示	Component 的所有子类对象	ComponentEvent
组件获取或者失去焦点	Component 的所有子类对象	FocuseEvent
按下或者释放键	Component 的所有子类对象	KeyEvent
鼠标按下、释放、点击、进入或离开组件、移动或者拖动	Component 的所有子类对象	MouseEvent

从表中可以看到不同的事件源在不同情况下产生事件的类型也是不同的，其中，所有的容器都能产生 ContainerEvent 事件，由于 Component 是所有 GUI 组件的父类，因此，所有的 GUI 组件都能产生 ComponentEvent、FocuseEvent、KeyEvent 和 MouseEvent。表中只列举了一小部分事件源和事件类，更多的内容在后面介绍各个 GUI 组件时再详细说明。

在了解了事件与事件源之间的关系后，再来解决一下第二个问题：如何实现监听器？监听器又是如何进行事件处理的？

前面说过，Java 使用委托事件处理模型来进行事件处理，就是事件源产生的事件委托给监听器进行处理。具体来说就是先要创建一个能够处理这种事件的监听器(监听器中实现了事件处理方法)，然后将这个监听器注册给产生这种事件的事件源(事件源中包含对应的注册方法)。这样，当用户行为作用于事件源(GUI 组件)时，事件源就会产生特定的事件，那么事先注册给它的特定监听器就能够接收到这个特定的事件，然后监听器调用其中实现的事件处理方法进行处理。具体的模型如图 8-10 所示。

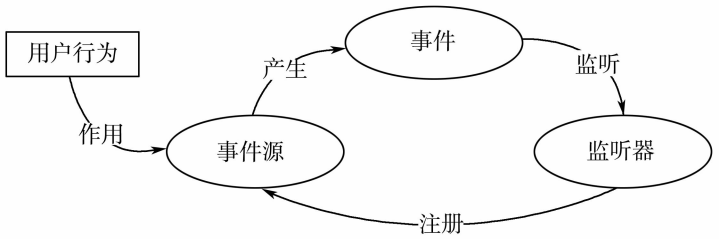

图 8-10　委托事件处理模型

在这个处理模型中，事件源就是 GUI 组件，在构造界面的时候已经创建完成。事件由事件源自动产生，剩下的问题就是如何来实现能够监听并处理事件的监听器了。那么具体如何实现一个监听器呢？

Java 中每一种事件类都有其对应的监听接口，要创建一个能够监听特定事件的监听器，就必须先定义一个实现监听接口的类，那么这个类的对象就是一个能够监听对应事件的监听器了。比如：ActionEvent 对应的监听接口是 ActionListener，要创建一个能够处理 AcitonEvent 事件的监听器就要实现 ActionListener 接口。在每一个监听接口中都定义了若干个事件处理方法，实现监听器实际上就是实现这些事件处理方法。当事件发生并且监听器接收到这个事件时，监听器就会调用事件处理方法进行处理。比如：ActionListener 接口中定义了 actionPerformed 方法，当 ActionListener 接收到 ActionEvent 发生时，就会调用这个方法，写在这个方法体中的事件处理语句就会被执行。表 8-2 中列举了基本事件和对应的监听接口，以及接口中定义的事件处理方法。

表 8-2　　事件类及其对应的监听接口和事件处理方法

事件类	监听接口	监听器中的事件处理方法
ActionEvent	ActionListener	actionPerformed(ActionEvent e)
AdjustmentEvent	AdjustmentListener	adjustmentValueChanged(AdjustmentEvent e)
ItemEvent	ItemListener	itemStateChanged(ItemEvent e)
TextEvent	TextListener	textValueChanged(TextEvent e)
WindowEvent	WindowListener	windowOpened(WindowEvent e) windowActivated(WindowEvent e) windowDeactivated(WindowEvent e) windowIconified(WindowEvent e) windowDeiconified(WindowEvent e) windowClosing(WindowEvent e) windowClosed(WindowEvent e)
ContainerEvent	ContainerListener	componentAdded(ContainerEvent e) componentRemoved(ContainerEvent e)
ComponentEvent	ComponentListener	componentMoved (ComponentEvent e) componentHidden (ComponentEvent e) componentResized (ComponentEvent e) componentShown (ComponentEvent e)
FocuseEvent	FocusListener	focusGained(FocusEvent e) focusLost(FocusEvent e)
KeyEvent	KeyListener	keyPressed(KeyEvent e) keyReleased(KeyEvent e) keyTyped(KeyEvent e)

（续表）

事件类	监听接口	监听器中的事件处理方法
MouseEvent	MouseListener	mousePressed(MouseEvent e) mouseReleased(MouseEvent e) mouseEntered(MouseEvent e) mouseExited(MouseEvent e) mouseClicked(MouseEvent e)
	MouseMotionListener	mouseDragged(MouseEvent e) mouseMoved(MouseEvent e)

从表中可以看到一些规律，就是 XXXEvent 对应的监听接口为 XXXListener，这里只有一个例外，就是 MouseEvent 对应两个监听接口 MouseListener 和 MouseMotionListener。

实现好监听器后还要将其注册给事件源，这样事件源产生事件时，事件才能被事先注册好的监听器监听到，并且进行相应的处理，从而实现程序的交互。注册是通过调用事件源的注册方法来实现的，在每一个事件源（GUI 组件）中都定义了若干个注册方法，这些注册方法与事件源能够产生的事件类型有关，与监听器类型对应，XXXListener 对应的注册方法为 addXXXListener。比如：按钮（JButton）可以产生 ActionEvent 事件，对应的监听器接口为 ActionListener，那么为按钮注册动作事件监听器的注册方法名为 addActionListener。

下面通过两个实例来进一步阐述一下具体如何进行事件处理，其中一个是动作事件处理，另外一个是窗口事件处理。

2.动作事件处理

（1）示例代码

【例 8-7】 简单动作事件处理程序。

```
//TestActionEvent.java
import javax.swing.*;
import java.awt.*;
//引入事件处理需要的事件类和监听接口
import java.awt.event.*;
//定义一个框架类，同时实现了动作事件的监听接口
public class TestActionEvent extends JFrame implements ActionListener
{
    //创建两个按钮
    private JButton jbtOk=new JButton("确定");
    private JButton jbtCancel=new JButton("取消");
    //构造方法
    public TestActionEvent(String title)
    {
        //初始化框架标题
        super(title);
        //设置内容窗格的布局为 FlowLayout
        getContentPane().setLayout(new FlowLayout());
```

```
        //在内容窗格上添加两个按钮
        getContentPane().add(jbtOk);
        getContentPane().add(jbtCancel);
        //为两个按钮注册监听器
        jbtOk.addActionListener(this);
        jbtCancel.addActionListener(this);
    }
    // main 方法
    public static void main(String[] args)
    {
        TestActionEvent frame=new TestActionEvent("动作事件");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(100, 80);
        frame.setVisible(true);
    }
    //事件处理方法,在事件发生时被调用
    public void actionPerformed(ActionEvent e)
    {
        //判断监听到的事件是否是按钮 jbtOk 产生的
        if (e.getSource() ==jbtOk)
        {
            System.out.println("确定按钮被点击");
        }
        else if (e.getSource() ==jbtCancel)
        {
            System.out.println("取消按钮被点击");
        }
    }
}
```

程序运行效果如图 8-11 所示。

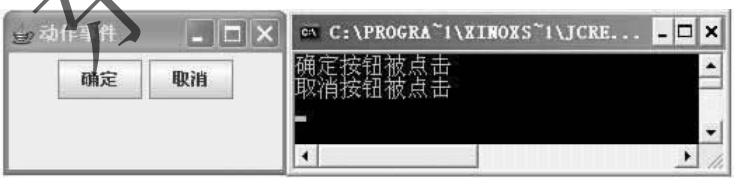

图 8-11　例 8-7 的运行效果

(2)代码分析

本例在框架上添加了两个按钮“确定”和“取消”,当用户点击两个按钮时,程序在控制台上打印哪个按钮被点击了。按照上一节介绍的事件处理模型,按钮 jbtOk 和 jbtCancel 是事件源,它们产生的事件类型是 ActionEvent 动作事件,TestActionEvent 类定义了一个框架,同时它也实现了 ActionListener,因此,它也是动作事件的监听器。在 TestActionEvent 类的构造方法中,通过调用两个按钮的 addActionListener 注册方法,将监听器注册给了这两个事件源。这样,当用户点击按钮时,按钮就会产生 ActionEvent 事件,事先注册给这个按钮的监听器(当前

的框架)就会监听到这个事件,然后调用事件处理方法 actionPerformed 方法进行处理。由于监听器负责监听两个事件源,所以在这个事件处理方法中,通过接收到的事件对象调用 getSource()方法来区分用户点击了哪个按钮。

(3)知识点

①动作事件类

类名:ActionEvent。

常用方法:

• public Object getSource():得到事件源引用,通常用于区分事件源。

• public String getActionCommand():得到动作命令,每个能够产生动作事件的事件源在产生动作事件时都会为动作事件赋予一个字符串类型的动作命令。对按钮来说,缺省就是按钮上显示的标签,比如:本例中的 Ok 按钮,它的动作命令就是"Ok",不过这个动作命令可以通过事件源的 setActionCommand 方法进行修改。通常这个方法用于区分动作事件的事件源。

②动作事件监听接口

监听接口名:ActionListener。

事件处理方法:

• public void actionPerformed(ActionEvent e);

③事件源的注册方法

• public void addActionListener(ActionListener listener):将动作事件监听器注册给事件源。

(4)举一反三

实现一个图形用户界面程序:在框架上顺序摆放三个按钮:Button1、Button2 和 Button3,在点击按钮时在控制台上显示哪个按钮被点击。

8.3.4　项目中用到的其他 GUI 组件

1.标签

标签是一种用于显示文本或者图片提示信息的控制组件。先看一个示例。

(1)示例代码

【例 8-8】使用标签程序。

```
// TestLabel.java
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class TestLabel extends JFrame{
    //定义了两个标签引用作为框架的属性
    private JLabel  jLabel1,jLabel2;
    public static void main(String[] args)  {
        //创建这个扩展框架对象
        TestLabel frame=new TestLabel();
        //pack 方法可以将容器的大小设置为刚好能盛放下其中所容纳的组件。
```

```
            frame.pack();
            frame.setVisible(true);
        }
        //构造方法
        public TestLabel(){
            setTitle("TestLabel");
            jLabel1=new JLabel("这是一个显示文本的标签");
            jLabel2=new JLabel(new ImageIcon("info.jpg"));
            getContentPane().setLayout(new FlowLayout());
            getContentPane().add(jLabel1);
            getContentPane().add(jLabel2);
        }
    }
```

本例中创建了两个标签,jLabel1 显示文本,jLabel2 显示图标。显示的效果如图 8-12 所示。

图 8-12　例 8-8 显示的界面

(2)标签的构造方法

①public JLabel():创建一个空标签。

②public JLabel(String text,int horizontalAlignment):创建一个指定内容字符串和水平对齐方式的标签。其中水平对齐方式可取值 SwingConstants.LEFT, SwingConstants.CENTER, SwingConstants.RIGHT。

③public JLabel(String text):创建一个指定文字的标签。

④public JLabel(Icon icon):创建一个指定图标的标签。图标可以利用 ImageIcon 类从图片文件中获取,比如:Icon icon=new ImageIcon(“images/Info.jpg”);

目前 Java 支持 GIF 和 JPEG 两种图片格式。

⑤public JLabel(Icon icon,int horizontalAlignment):创建一个指定图标和水平对齐方式的标签。

⑥public JLabel(String text,Icon icon,int horizontalAlignment):创建一个指定文本、图标和水平对齐方式的标签。

(3)常用的方法

①publicvoid setText(String text):设置标签上的文本。

②publicString getText():获取标签上的文本。

③publicvoid setIcon(Icon icon):设置标签上的图标。

④publicIcon getIcon():获取标签上的图标。

(4)对标签的事件处理

标签能产生鼠标、焦点、键盘和组件等事件。但是,由于标签用于显示提示信息,所以一般不对其进行事件处理。

2.按钮

按钮(JButton)是一种点击时能够产生动作事件的控制组件,一般按钮用于执行某种操作。

(1)示例代码

【例 8-9】使用按钮程序。

```
// TestButton.java
import java.awt.*;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.*;
public class TestButton extends JFrame  implements ActionListener
{
    private JButton jButton1;
    private JLabel  jLabel1;
    private int count=0;
    public static void main(String[] args)
    {
        TestButton frame=new TestButton();
        frame.pack();
        frame.setVisible(true);
    }
    public TestButton()
    {
        setTitle("TestButton");
        //创建一个带有标签文本和图标的按钮
        jButton1=new JButton("Press me",new ImageIcon("button.jpg"));
        //设置按钮的热键为“ALT+P”
        jButton1.setMnemonic('P');
        //设置按钮的提示信息为“press me”
        jButton1.setToolTipText("Press me");
        jLabel1=new JLabel("按钮点击了 0 次");
        getContentPane().setLayout(new FlowLayout());
        getContentPane().add(jButton1);
        getContentPane().add(jLabel1);
        //为按钮注册动作事件的事件监听器
        jButton1.addActionListener(this);
    }
    //事件处理方法
    public void actionPerformed(ActionEvent e)
    {
        count++;
        jLabel1.setText("按钮点击了 "+count+" 次");
    }
}
```

本例中在框架 TestButton 上添加了一个按钮 jButton1 和一个标签 jLabel1，按钮上同时显示了图标和文本。当点击按钮时，在标签上显示按钮被点击的次数。在程序中为按钮设置了热键 P，通过按键【ALT＋P】可以得到与点击相同的效果，另外，还为按钮设置了提示信息“Press me”，当鼠标移动到按钮上时就会显示这个提示信息。本例的运行效果如图 8-13 所示。

图 8-13　例 8-9 显示的界面

(2)按钮的构造方法

①public JButton()：创建一个空按钮。

②public JButton(String text)：创建一个标有指定文字的按钮。

③public JButton(Icon icon)：创建一个标有指定图标的按钮。

④public JButton(String text，Icon icon)：创建一个标有指定文字和图标的按钮。

(3)常用的方法

①publicvoid setMnemonic(int mnemonic)：设置按钮的热键，比如：setMnemonic('P')，设置按钮的热键为【ALT＋P】。

②publicvoid setToolTipText (String text) ：设置按钮的提示信息。

③publicvoid setText(String text)：设置按钮上的文本。

④publicString getText() ：获取按钮上的文本。

⑤publicvoid setIcon(Icon icon)：设置按钮上的图标。

⑥publicIcon getIcon()：获取按钮上的图标。

(4)对按钮的事件处理

按钮可以产生多种事件，不过由于按钮一般用于触发某种操作的执行，因此，一般情况下只处理按钮的动作事件 ActionEvent，要处理动作事件需要实现 ActionListener 中的 ActionPerformed 方法。

3.文本框

文本框(JTextField)是一种用于显示、输入或者编辑单行文本的控制组件。

(1)示例代码

【例 8-10】 使用文本框程序。

```
// TestTextField.java
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
class TestTextField extends JFrame implements ActionListener{
    JLabel p;
    JTextField in;
    JLabel out;
```

```
    String s="";
public MyFrame(){
    p=new JLabel ("请输入口令:");
    in=new JTextField(18);
    out=new JLabel();
    this.getContentPane().setLayout(new FlowLayout());
    this.getContentPane().add(p);
    this.getContentPane().add(in);
    this.getContentPane().add(out);
        in.addActionListener(this);
    }
    public void actionPerformed(ActionEvent evt){
        s=in.getText();
        if(s.equals("MyKey"))
        out.setText("通过!");
        else
            out.setText("口令错!");
        }
    public static void main(String[] args){
        TestTextField textField=new TestTextField();
        textField.setTitle("Show");
        textField.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        textField.setSize(250,250);
        textField.setVisible(true);
    }
}
```

本例中在框架 TestTextField 上添加了一个文本框，定义两个标签，其中第一个标签上面的文本信息为“请输入口令：”；第二个标签用于显示口令是否正确的提示信息；再定义一个文本框，用于口令输入，假设口令为字符串”MyKey”正确，则第二个标签设置为“通过!”，否则设置为“口令错!”。本例中实现动作事件的监听器，在 actionPerformed 方法中，通过文本框的 getText()方法获得文本框中用户输入的字符串，通过 String 类的 equals 方法判断输入的字符串是否等于“MyKey”，然后通过 JLabel 的 setText 方法修改第二个 JLabel 标签的内容。本例的运行效果如图 8-14 所示。

图 8-14　例 8-10 显示的界面

(2)文本框的构造方法

①public JTextField():创建一个空文本框。

②public JTextField(int columns):创建一个指定列数的空文本框。

③public JTextField(String text):用指定初始文字创建一个文本框。

④public JTextField(String text,int columns):创建一个文本框,并用指定文字和列数初始化。

(3)常用的方法

①public StringgetText():获取文本框中的文本。

②public voidsetText(String text):将给定字符串写入文本框当中

③public voidsetEditable(boolean editable):设置文本框的可编辑属性,true 为可编辑,false 为不可编辑,默认为 true。

④public voidsetColumns(int col):设置文本框的列数,文本框的长度可变。

(4)对文本框的事件处理

JTextField 能产生 ActionEvent 事件及其他组件事件。在文本域按回车键引发 ActionEvent 事件。不过,由于文本框一般用于文本的输入和编辑,因此,不对其进行事件处理。

8.4 项目学做

(1)引入项目所需要的包。

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
```

(2)构造加法器的图形界面,首先继承 JFrame,声明界面组成部分的组件,关于界面的构造在 MyComputer 的构造方法中完成。

```
classMyComputer extends JFrame implements ActionListener{
    JTextField JTest1,JTest2,JTest3;
    JLabel Jla1;
    JButton Jbu;
    test(){
        setTitle("cumputer");
        Container cp=this.getContentPane();
        cp.setLayout(new FlowLayout());
        cp.add(JTest1=new JTextField(5));
        JTest1.setText("0");
        cp.add(Jla1=new JLabel("+"));
        cp.add(JTest2=new JTextField(5));
        cp.add(Jbu=new JButton("="));
        cp.add(JTest3=new JTextField(5));
        JTest3.setEditable(false);
        Jbu.addActionListener(this);
    }
}
```

(3)为加法器添加动作事件,在类 MyComputer 后面实现动作事件对应的接口 implements ActionListener,在 MyComputer 中实现动作事件对应的方法 actionPerformed(ActionEvent e),具体的实现如下。

```
public void actionPerformed(ActionEvent e){
    double n1=0,n2=0;
    n1=Double.parseDouble(JTest1.getText());
    n2=Double.parseDouble(JTest2.getText());
    JTest3.setText((n1+n2+" "));
}
```

(4)在 main 方法中创建 MyComputer,使其可见,设置关闭方法等。

```
public static void main(String[] args){
        MyComputer t=new MyComputer ();
        t.pack();
        t.setVisible(true);
}
```

8.5　知识拓展

8.5.1　图形用户界面简介

在 Java 中,GUI 由 GUI 组件构成,目前常用的 GUI 组件类多数定义在 javax.swing 包中,称为 Swing 组件。这和早期的 Java 有些不同,早期的 GUI 组件定义在称为抽象窗口工具包(Abstract Window Toolkit,AWT)的 java.awt 包中。由于 AWT 组件直接绑定在本地图形用户界面功能上,对底层平台的依赖性较强,因此可移植性较差。到了 Java 2 时,出现了 Swing 组件,Swing 组件直接使用 Java 语言编写,对本地底层平台的依赖性较少,更灵活也更稳定。Swing 组件称为轻型组件,而 AWT 组件称为重型组件。因此,在编程中建议使用 Swing 组件,以提高程序的可移植性。但需要注意的是 Swing 不能完全取代 AWT 的全部,AWT 中的一些图形辅助类目前还在使用。

为了方便实现各种图形用户界面程序,Java 提供了大量的图形用户界面类和接口,比如组件类有:JFrame、JPanel、JDialog、JButton、JLabel、JTextField、JMenu 和 JMenuBar 等等,事件类有:ActionEvent、MouseEvent 和 KeyEvent 等等;监听接口有:ActionListener、MouseListener 和 MouseMotionListener 等等,布局管理器类有:BorderLayout、FlowLayout 和 GridLayout 等等,绘图相关的类有:Griphics、Color、Font 等等。

从列举的这些类和接口中可以感觉到,图形化程序设计比之前的命令行方式要复杂。特别是对于刚刚接触 Java 的人来讲,同时接触这么多的类和接口可能感觉无从下手。为了便于理解,可以简单地将图形用户界面程序设计的主要工作分成两部分。

(1)计并创建界面外观:主要是创建组成图形界面的各个组件,并按照设计组合排列,构成

完整的图形用户界面。

(2)现界面的交互功能:主要是为界面外观添加事件处理,实现能够处理各种界面事件的处理方法,以完成程序与用户之间进行交互的任务。

8.5.2 窗口事件

我们已经学习了事件处理模型,知道实现某一事件类型,先要实现其对应的监听接口,然后实现监听接口里面对应的方法,现在我们要处理窗口事件,大家按照我们之前讲的知识来实现,具体的代码如例 8-11。

【例 8-11】处理窗口事件程序。

```
// TestWindowEvent.java
import java.awt.*;
import java.awt.event.*;
import javax.swing.JFrame;
//实现一个框架,同时实现一个窗口事件监听器
public class TestWindowEvent extends JFrame  implements WindowListener
{
    //构造方法
    public TestWindowEvent(String title)
    {
        super(title);
        //注册窗口事件监听器
        addWindowListener(this);
    }
    //当窗口打开时被调用
    public void windowOpened(WindowEvent event)
    {
        System.out.println("窗口被打开");
    }
    //当窗口还原时被调用
    public void windowDeiconified(WindowEvent event)
    {
        System.out.println("窗口被还原");
    }
    //当窗口最小化时被调用
    public void windowIconified(WindowEvent event)
    {
        System.out.println("窗口被最小化");
    }
    //当窗口激活时被调用
    public void windowActivated(WindowEvent event)
    {
        System.out.println("窗口被激活");
```

```
        }
        //当窗口失效时被调用
        public void windowDeactivated(WindowEvent event)
        {
            System.out.println("窗口失效");
        }
        //当窗口正在关闭时被调用
        public void windowClosing(WindowEvent event)
        {
            System.out.println("窗口正在关闭");
        }
        //当窗口已经关闭时被调用
        public void windowClosed(WindowEvent event)
        {
            System.out.println("窗口被关闭");
        }
        //main 方法
        public static void main(String[] args)
        {
            TestWindowEvent frame=new TestWindowEvent("TestWindowEvent");
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            frame.setSize(100, 80);
            frame.setVisible(true);
        }
    }
```

程序运行效果如图 8-15 所示。

图 8-15　例 8-11 的运行效果

本例演示了如何处理窗口事件。由于 Window 类的任何子类都能发生窗口事件，所以本例以 JFrame 为例演示了窗口事件的处理过程。程序中 TestWindowEvent 是一个框架，因此它可以作为 Window 事件的事件源，同时它又实现了 WindowListener，所以，它也是监听器，在构造方法中，将它自己作为事件监听器，通过方法 addWindowListener 注册给了自己。从实例中可以看到 window 事件的事件处理方法有 7 个之多，分别是 windowOpened、windowClosing、windowClosed、windowActivated、windowDeactivated、windowIconfied 和 windowDeiconfied。

当窗口打开、正在关闭、关闭、激活、失效、最小化和还原时都会产生 WindowEvent，7 个事件处理方法分别用于处理这 7 种不同的情况，也就是说，windowOpened 方法在窗口打开时被执行，windowClosed 方法在窗口关闭时被执行，依此类推。

(1)窗口事件类

类名：WindowEvent。

可产生窗口事件的事件源：所有 Window 的子类。

(2)窗口事件监听接口

监听接口名：WindowListener。

事件处理方法：

①public void windowOpened (WindowEvent e)：用于处理窗口打开时情况。

②public void windowClosing (WindowEvent e)：用于处理窗口正在关闭时情况。

③public void windowClosed (WindowEvent e)：用于处理窗口关闭时情况。

④public void windowActivated (WindowEvent e)：用于处理窗口激活时情况。

⑤public void windowDeactivated (WindowEvent e)：用于处理窗口失效时情况。

⑥public void windowIconfied (WindowEvent e)：用于处理窗口最小化时情况。

⑦public void windowDeiconfied (WindowEvent e)：用于处理窗口还原时情况。

【注意】因为所有这些方法在接口中都是抽象的，所以即使只想处理其中某种情况的 Window 事件，所有这些方法也必须在监听器类中全部实现。

(3)事件源的注册方法

public void addWindowListener(WindowListener listener)：将 window 事件监听器注册给事件源。

8.5.3 事件裁剪类

在上一节中学习了 Window 事件的处理，其中监听接口中定义了 7 个事件处理方法，以对应产生窗口事件的 7 种情况，而实际编程中往往只要处理其中某一两种情况，但是限于接口实现的要求，却必须要实现全部 7 个方法，即使是用空的方法体实现。而这种情况在许多事件的监听接口中普遍存在。针对这种情况，Java 为那些具有多个事件处理方法的监听接口提供了对应的事件裁剪类，这个类通常命名为 XXXAdapter，有时也称为事件适配器类，在这种类中，以空的方法体实现了相应接口中的所有事件处理方法，在实际实现监听器类时就可以继承这个裁剪，这样在监听器类中就可以根据需要只实现监听接口中某些抽象方法，而不必全部实现接口中的抽象方法了。下面以窗口事件为例，介绍一下事件裁剪类的用法。

【例 8-12】通过窗口事件裁剪类实现窗口关闭事件处理程序。

```
// TestWindowAdapter.java
import java.awt.*;
import java.awt.event.*;
import javax.swing.JFrame;
//定义一个框架
public class TestWindowAdapter extends JFrame
```

```
{
    //构造方法
    public TestWindowAdapter(String title)
    {
        super(title);
        //注册窗口事件监听器
        addWindowListener(new MyListener());
    }
    //main 方法
    public static void main(String[] args)
    {
        TestWindowAdapter frame=new TestWindowAdapter("TestWindowAdapter");
        frame.setSize(200, 100);
        frame.setVisible(true);
    }
    //通过继承 WindowAdapter 实现一个监听器类
    class  MyListener extends WindowAdapter
    {
        //当窗口正在关闭时被调用
        public void windowClosing(WindowEvent event)
        {
            System.out.println("Window closing");
            System.exit(0);
        }
    }
}
```

当关闭窗口时，控制台的输出信息如图 8-16 所示。

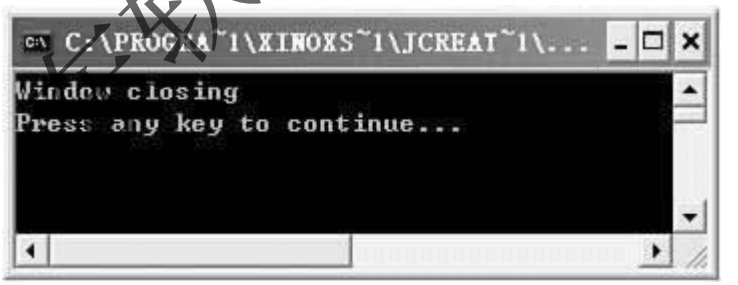

图 8-16　例 8-12 的输出信息

本例通过继承 WindowAdapter 来实现了一个监听器类，通过监听器实现了在关闭窗口时打印提示信息，并退出应用程序的功能。通过这个例子与例 8-11 的比较，可以看到借助于裁剪类，事件监听器的实现被简化了。但是，由于 Java 是单继承，所以，不能像例 8-11 那样一个类既是框架类，又是监听器类。而且，一个监听器类也不能同时继承两个裁剪类，也就说一个监听器类不能监听一种以上的事件，而通过实现监听接口却可以做到。

8.6 强化训练

请编写一个 Application 实现如下功能：定义一个用于给出提示信息的标签、两个文本框和一个显示“计算”的按钮，其中，一个文本框用于获取用户给出的一个整数，点击“计算”按钮后，求出该数的平方后将计算结果在另一个文本框中输出。

（知识点考察：定义标签和文本框，数值型数据与字符串类型相互转换；动作事件的处理）

8.7 课后习题

一、选择题

1. 下面属于容器类的是（　　）。

（A）JFrame　　（B）JTextField

（C）Color　　（D）JMenu

2. FlowLayout 的布局策略是（　　）。

（A）按添加的顺序由左至右将组件排列在容器中。

（B）按设定的行数和列数以网格的形式排列组件。

（C）将窗口划分成五部分，在这五个区域中添加组件。

（D）组件相互叠加排列在容器中。

3. BorderLayout 的布局策略是（　　）。

（A）按添加的顺序由左至右将组件排列在容器中。

（B）按设定的行数和列数以网格的形式排列组件。

（C）将窗口划分成五部分，在这五个区域中添加组件。

（D）组件相互叠加排列在容器中。

4. GridLayout 的布局策略是（　　）。

（A）按添加的顺序由左至右将组件排列在容器中。

（B）按设定的行数和列数以网格的形式排列组件。

（C）将窗口划分成五部分，在这五个区域中添加组件。

（D）组件相互叠加排列在容器中。

5. JFrame 中内容窗格缺省的布局管理器是（　　）。

（A）FlowLayout　　（B）BorderLayout

（C）GridLayout　　（D）CardLayout

二、填空题

1. Java 的 Swing 包中定义框架的类是_____。

2. Java 的 Swing 包中定义按钮的类是_____。

3. Java 的 Swing 包中定义文本域的类是_____。

4. Java 的 Swing 包中定义标签的类是_____。

5. ActionEvent 类定义在_____包中。

6. ActionEvent 事件的监听接口是_____，注册方法名是_____，事件处理方法名是_____。

7. WindowEvent 事件的监听接口是_____，注册方法名是_____。

8. 设置容器布局管理器的方法是_____。

9. 显示 JFrame 框架的方法名是_____。

10. 设置 JFrame 框架标题的方法名是_____。

11. 设置 JFrame 框架大小的方法名是_____。

12. 设置按钮上文本的方法名是_____，获取按钮上文本的方法名是_____。

13. 设置文本域上文本的方法名是_____，获取文本域上文本的方法名是_____，设置文本域可编辑属性的方法名是_____。

三、编程题

1. 请编写一个 Application，其功能为：在其图形窗口按右对齐方式摆放三个按钮，三个按钮的标题分别显示为："Button 1"，"Button 2"，"Button 3"。（知识点考察：FlowLayout 布局管理器的使用）

2. 请编写一个 Application，其功能为：在其框架的内容网格上安排两个按钮，分别命名为 East，West，内容网格的布局为 BorderLayout 布局，并将两个按钮放置在内容网格的东部区域和西部区域。（知识点考察：BorderLayout 布局管理器的使用方法）

3. 设计一个程序在窗口的东南西北中各放置一按钮，水平和垂直的间距均为 6，具体实现效果如图 8-17 所示。

图 8-17　题 3 的效果图

4. 将六个按钮顺序摆放在窗口中，且中央对齐，每个组件之间水平间距 10，垂直间距 10，具体实现效果如图 8-18 所示。

图 8-18　题 4 的效果图

5. 设计一个计算器的面板，只要求布置九个数字按钮，具体实现效果如图 8-19 所示。

图 8-19　题 5 的效果图

6. 请编写一个 Application，其功能为：在窗口上摆放两个标签。构造第一个标签时，令其上面的文本信息为“我将参加 Java 程序设计考试。”，将第二个标签构造为空标签。程序将第一个标签的信息复制到第二个标签上，并增加信息“希望自己考取好成绩。”。要求第一个标签以红色为背景，绿色为前景；第二个标签以绿色为背景，蓝色为前景。（知识点考察：定义标签，设置标签文本值和背景颜色）

7. 请编写一个 Application 实现如下功能：定义三个文本框，其中，第一个文本框上面的文本信息为“请输入口令：”；第二个文本框为口令输入域；第三个文本框上的信息由程序设置：若口令（假设口令为字符串“MyKey”）正确，则设置为“通过！”，否则设置为“口令错！”；。（知识点考察：定义文本框，设置和获取文本框的文本值）

8. 编写 Application，其中包含两个按钮 b1、b2，初始时 b1 的前景为蓝色，b2 的前景为红色，它们的标签分别为“蓝按钮”“红按钮”。无论哪个按钮被点击，都将该按钮上的标记改为“已按过”，并使该按钮变灰。（知识点考察：定义并设置按钮的前景色和背景色，点击按钮触发事件处理过程）