第5章 图形用户界面

本章导读

通过图形用户界面(Graphics User Interface,GUI),用户和程序之间可以方便地进行交互。采用 Java 进行图形用户界面设计,或者说进行桌面应用程序开发并非难事。图形用户界面程序主要由三个要素构成:组件、布局和事件。开发人员将提供不同交互功能的组件按某种布局方式进行排列,从而构建用户界面,与用户进行交互。交互则主要采用事件驱动的方式来完成,当用户的动作触发某种事件,例如,单击鼠标或移动鼠标可触发鼠标事件,按下键盘上的某个键则会触发键盘事件等,事件处理器监听到事件发生就会响应,也就是进行相应的事件处理,从而实现对应业务逻辑。Java 平台提供了几种可供选择的图形用户界面库,用于开发界面应用程序。本章将介绍 Java 图形用户界面设计方法。

本章要点

- Java 图形用户界面库。
- 底层容器窗口和对话框的使用方法。
- 菜单的制作方法。
- 常用布局的设计方法。
- 常用组件及面板的使用方法。
- 事件处理机制及方法。

5.1 Java 图形用户界面库概述

1. Java 图形用户界面库

(1) AWT。

AWT(Abstract Window Toolkit)又称为抽象窗口工具集,它是 Java 最早提供的图形用户界面库。AWT 尽管开发较早但功能完备,可以开发基本的跨平台桌面应用程序。采用 AWT 开发的桌面应用程序外观样式在不同的操作系统下是不同的,这主要是因为 AWT 中的组件要调用所在操作系统平台的相应(对等)组件完成桌面应用程序的外观渲染及功能。因此,采用 AWT 组件运行时开销比较大,通常 AWT 中的组件被称为重量级组件。

(2) Swing。

Swing 是 Java 的另一个图形用户界面库,JDK 1.2 后引入。Swing 继承于 AWT,但 Swing 的功能更强大、性能更优化,与 AWT 相比较,更能体现 Java 语言的跨平台性。采用 Swing 开发的桌面应用程序在不同的操作系统平台上外观是相同的,也就是说,Swing 的用

户界面组件是由 Java 平台自己绘制的,不再调用操作系统的对等控件。因此,与 AWT 相对应的 Swing 中的组件通常称为轻量级组件。Swing 不仅组件比 AWT 更加丰富,设计方面也更加成熟,层次更加清晰,因此建议使用 Swing 开发桌面应用程序。

(3) JavaFX。

AWT 和 Swing 推出以来一直是 Java 开发桌面应用程序的主要框架,但随着应用需求的提高,桌面应用程序中大量图片、音频、视频的使用,AWT 和 Swing 中的通用组件已逐渐无法满足应用的需求,因此 Java 推出了 JavaFX,以满足现代桌面应用的开发需求。2007 年,JavaFX 第一个版本发布,随着 JavaFX 的逐渐完善,它将逐渐成为 Java 平台上主要的图形用户界面库。

除上述三种图形用户界面库之外,也可以使用第三方提供的图形用户界面库,但需要额外安装才能使用。

2. Swing 继承架构

本书主要使用 Swing 进行桌面应用程序开发,在使用 Swing 之前,首先讲述一下 Swing 的继承架构,如图 5.1 所示。

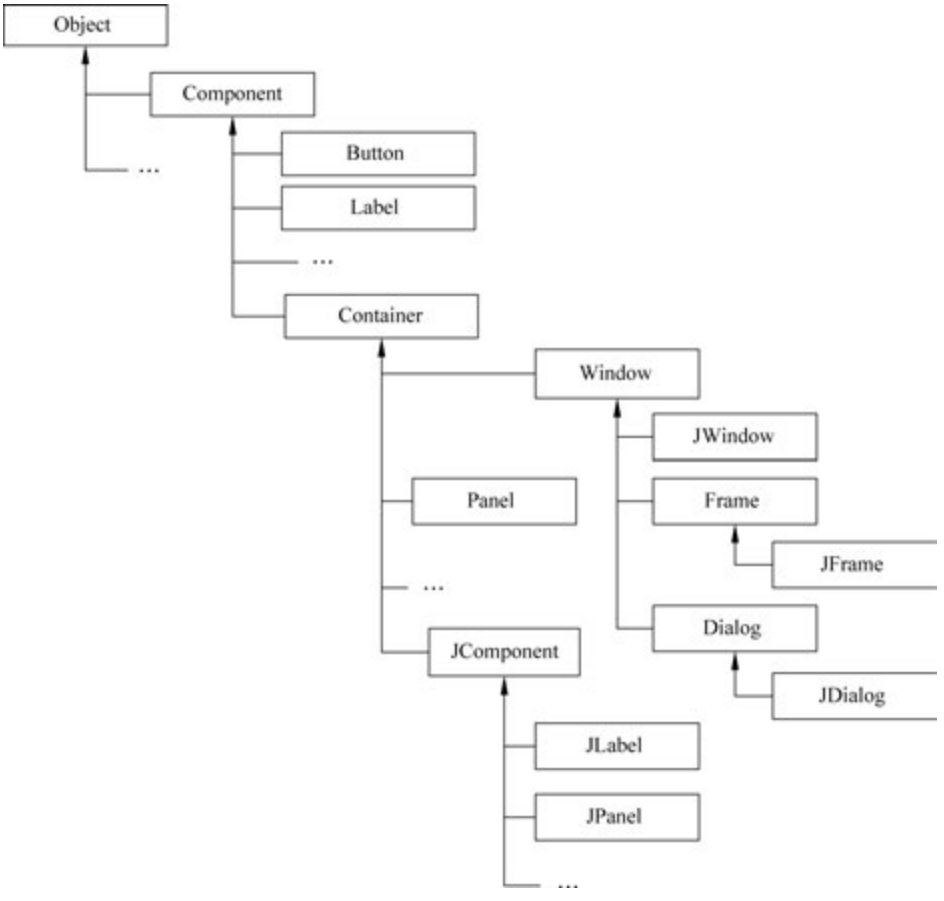

图 5.1 Swing 继承架构

Swing 继承于 AWT,图 5.1 中所有以"J"开头的组件为 Swing 中的组件,以区别于 AWT 组件。在 Swing 的继承架构中,除 4 个顶层窗口组件 JFrame、JDialog、JWindow 和

JApplet 直接继承于 AWT 中的相应组件，其余组件都继承于 JComponent。4 个顶层窗口组件通常也称为底层容器，支持与操作系统的交互；其余组件在设计过程中要添加到某种底层容器的实例中，按某种布局方式进行排列，然后绑定事件监视器进行事件处理，完成和用户交互。Swing 内容非常丰富，本章主要讲述常用组件、布局、事件以及对话框等基本使用方法。

5.2 窗　　口

在 Swing 应用程序开发中，通常使用 JFrame 类创建的实例作为底层容器，这样的容器可以采用 JFrame 类直接创建，也可以采用继承于 JFrame 的子类创建。采用 JFrame 创建的“计算器”窗口如图 5.2 所示。窗口可以设置标题，另外窗口自带“最小化”“最大化”和“关闭”按钮。JFrame 默认布局为 BorderLayout。

JFrame 类中提供了一系列方法完成窗口属性的设置。

图 5.2　“计算器”窗口

JFrame 常用方法如下。

- JFrame([String s])创建窗口的构造方法、参数用于设置窗口标题。
- setTitle(String title)用来设置窗口标题。
- setBounds(int x,int y,int width,int height)用来设置窗口显示的位置和大小。
- setVisible(boolean b)用来设置窗口是否可见，默认不可见。
- dispose()用来撤销当前窗口，释放资源。
- setDefaultCloseOperation(int operation)用来设置“关闭”按钮动作，operation 可选参数如表 5.1 所示。

表 5.1　窗口“关闭”按钮常量

JFrame 类中 static 常量	值	动　　作
DO_NOTHING_ON_CLOSE	0	不执行任何操作
HIDE_ON_CLOSE	1	隐藏窗口
DISPOSE_ON_CLOSE	2	撤销窗口，释放资源
EXIT_ON_CLOSE	3	结束应用程序

例 5-1　窗口。

源文件为 Sample5_1.java。代码如下。

```
import javax.swing.*;
public class Sample5_1{
    private JFrame f;
    public Sample5_1(){
        init();
    }
    private void init(){
        f = new JFrame("计算器");          //使用 JFrame 创建窗口，标题为"计算器"
```

```
        f.setSize(300,200);                    //设置窗口大小
        f.setResizable(false);                 //设置窗口不可以调整大小
        f.setVisible(true);                    //设置窗口可见
    }
    public static void main(String[] args){
        Sample5_1 t = new Sample5_1();
    }
}
```

程序运行结果如图 5.2 所示。

5.3 菜　　单

在 Swing 中使用 JMenubar、JMenu 和 JMenuItem 创建菜单。菜单制作基本步骤如下。

(1) 使用 JMenubar 创建菜单条。

(2) 使用 JMenu 创建菜单。

(3) 使用 JMenuItem 创建菜单项。

(4) 将菜单项添加到菜单中,再将菜单添加到菜单条中,最后将菜单条添加到窗口中。

(5) 为菜单项添加事件监听器。

(6) 编写事件处理代码。

注意:菜单条、菜单及菜单项的创建及添加操作没有先后顺序要求。

例 5-2　菜单。

源文件为 Sample5_2.java。代码如下。

```
import javax.swing.*;
import java.awt.event.*;
public class Sample5_2{
    private JFrame f;
    public Sample5_2(){
        f = new JFrame("计算器");
        f.setSize(240,150);
        f.setResizable(false);

        JMenuBar menubar = new JMenuBar();
        f.setJMenuBar(menubar);
        JMenu menu1 = new JMenu("系统(V)");
        menu1.setMnemonic('V');
        JMenu menu2 = new JMenu("编辑(E)");
        menu2.setMnemonic('E');
        JMenu menu3 = new JMenu("帮助(H)");
        menu3.setMnemonic('H');
        menubar.add(menu1);
        menubar.add(menu2);
        menubar.add(menu3);

        JMenuItem item2_3 = new JMenuItem("编辑备忘录");
item2_3.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_E,ActionEvent.CTRL_MASK));
        JMenuItem item1_2 = new JMenuItem("退出");
```

```
item1_2.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_Q,ActionEvent.CTRL_MASK));
        JMenuItem item2_1 = new JMenuItem("复制",new ImageIcon("FIRST.GIF")); item2_1.
setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_C,ActionEvent.CTRL_MASK));
        JMenuItem item2_2 = new JMenuItem("粘贴",new ImageIcon("LAST.GIF"));        item2_2.
setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_V,ActionEvent.CTRL_MASK));
        JMenuItem item3_1 = new JMenuItem("关于计算器");

        JMenu subMenu = new JMenu("备忘录");
        JMenuItem subItem1 = new JMenuItem("浏览");
        JMenuItem subItem2 = new JMenuItem("复制");
        JMenuItem subItem3 = new JMenuItem("删除");
        JMenuItem subItem4 = new JMenuItem("创建");
        subMenu.add(subItem1);
        subMenu.add(subItem2);
        subMenu.add(subItem3);
        subMenu.add(subItem4);

        menu1.add(subMenu);
        menu1.addSeparator();
        menu1.add(item1_2);
        menu2.add(item2_1);
        menu2.add(item2_2);
        menu2.addSeparator();
        menu2.add(item2_3);
        menu3.add(item3_1);
        f.setVisible(true);
    }
        public static void main(String[] args){
            Sample5_2 t = new Sample5_2();
       }
}
```

运行结果如图 5.3 所示。

图 5.3　例 5-2 运行结果

例 5-3　弹出式菜单。

使用 JPopupMenu 创建弹出式菜单,基本步骤如下。

(1) 创建弹出式菜单。

(2) 创建菜单项,将菜单项添加到弹出式菜单中。

(3) 为显示弹出式菜单的组件添加事件监听器,并处理弹出式菜单事件。

源文件为 Sample5_3.java。代码如下。

```
import javax.swing.*;
import java.awt.event.*;
public class Sample5_3  extends JFrame{
    JPopupMenu popup;
    public Sample5_3(){
        super("弹出式菜单测试窗口");        //调用 JFrame 构造方法创建窗口并设置窗口标题
        setSize(300, 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);        //关闭窗口退出程序
        popup = new JPopupMenu();                              //创建弹出式菜单
        JMenuItem add = new JMenuItem("添加");;                //创建菜单项
        JMenuItem del = new JMenuItem("删除");
        JMenuItem exit = new JMenuItem("退出");
        popup.add(add);                                        //将菜单项添加到弹出式菜单中
        popup.add(del);
        popup.add(exit);
        addMouseListener(new MouseAdapter(){                   //为窗口添加鼠标事件监听器
            public void mouseReleased(MouseEvent event){
              if(event.isPopupTrigger())                       //判断是否为弹出菜单触发事件
                  popup.show(event.getComponent(),event.getX(),event.getY());
//在鼠标右键单击的位置显示弹出式菜单
            }
       }
    );
        setVisible(true);                                      //设置窗口可见
   }
public static void main(String args[]) {
  new Sample5_3();                                             //创建窗口
 }
}
```

运行结果如图 5.4 所示。

图 5.4　例 5-3 运行结果

5.4 组件及面板

除4个顶层窗口组件外，Swing中的其余组件全部直接或间接继承于JComponent。Swing中的组件很丰富，这里只介绍其中几个常用的组件。

1. 标签

使用JLabel创建标签组件。标签可用来显示文本信息，也可用于显示图像信息。

2. 按钮

使用JButton创建按钮组件。按钮有按下和释放两种状态，通过接收用户操作捕获按钮状态，从而与用户进行交互。按钮上可以显示文本，也可以显示图片，并且在不同状态下可以显示不同信息。

3. 文本框

使用JTextField创建文本框。文本框可用于输入或显示单行文本。

4. 文本区

使用JTextArea创建文本区。文本区中允许输入或显示多行文本，文本区本身没有滚动条，通常将文本区放入滚动面板，借助滚动面板的滚动条显示文本区内容。

5. 单选按钮

使用JRadioButton创建单选按钮，单选按钮有选中和未选中两种状态。通常单选按钮成组使用，也就是与BottonGroup类联合，构成按钮组。按钮组中的单选按钮一个时刻只能有一个按钮处于被选中状态。

6. 复选框

使用JCheckBox创建复选框，复选框有选定和未选定两种状态，可同时选定多个复选框。

7. 选择框

使用JComboBox创建选择框，选择框允许用户从其列表中选择所需值，也允许用户在选择框中输入值。选择框列表中的值可以在创建时设置，也可以在创建选择框后，调用选择框方法addItem()或者insertItem()等，管理选择框列表中的值。

8. 列表框

使用JList创建列表框。列表框也是通过列表的方式供用户选择，但外观样式与选择框略有不同，并且列表框可以多选，而选择框只能单选。

9. 密码框

使用JPasswordField创建密码框。用来输入密码信息，以回显字符显示用户输入的密码。回显字符默认为“*”，可以调用密码框的setEchoChar(char c)方法修改回显字符。

10. 树组件

使用JTree创建树组件，主要用于展示树结构的数据。例如，呈现文件系统目录结构、组织架构图、菜单选项等。借助JTree，用户能够方便地展开和折叠树节点，对节点进行选择操作。

11. 表格组件

使用JTable创建表格组件。它可以将数据以表格的形式呈现给用户，支持多种数据类

型的显示,允许用户对表格进行选择、排序、编辑等操作。常用于展示数据库查询结果、统计数据、配置信息等。

12. 面板

由于 JComponent 继承于 AWT 的 Container,因此 JComponent 的子类实例都可用于容器。但常用的用来承载其他组件的容器有 JPanel 和 JScrollPane 两种。通过将组件添加到面板中(面板中也可以添加其他面板),能够实现对组件的分层管理,使组件布局更合理,界面更美观。

(1) JPanel 面板。

JPanel 可以创建一个普通面板,可以向面板中添加组件,然后再将面板添加到顶层窗口中。JPanel 面板的默认布局是 FlowLayout。

(2) JScrollPane 面板。

JScrollPane 可以创建一个带有滚动条的滚动面板,主要为一些不带滚动条的组件添加滚动条,并可以通过滚动条使用组件,例如,JTextArea、JList。JScrollPane 的默认布局为 ScrollPaneLayout,该布局将组件填满整个滚动面板。

(3) JSplitPane 面板。

JSplitPane 可以创建分割面板,将窗口分割成几部分,分别用于显示不同信息。

前面简单介绍了几种 Swing 中的常用组件,这些组件都提供了若干属性及方法,常用属性和方法的使用见例 5-4。

例 5-4 使用 Java 组件及面板。

源文件为 Sample5_4.java。代码如下。

```
import java.awt.*;
import javax.swing.*;
public class Sample5_4 extends JFrame{
    public Sample5_4(){
        setTitle("常用组件");                                    //设置窗体的标题
        setBounds(100, 100,1000, 1000);                          //设置窗体的显示位置及大小
        setLayout(null);                                         //设置为不采用任何布局管理器
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JButton button1 = new JButton();                         //创建按钮对象
        button1.setBounds(280,145, 70, 23);                      //设置标签的显示位置及大小
        button1.setText("确定");
        add(button1);                                            //将按钮添加到窗体中
        JButton button2 = new JButton();                         //创建按钮对象
        button2.setBounds(280,175, 70, 23);                      //设置标签的显示位置及大小
        button2.setText("取消");
        add(button2);                                            //将按钮添加到窗体中
        JLabel label1 = new JLabel();                            //创建标签对象
        label1.setText("性别:");                                 //设置标签文本
        label1.setBounds(10, 36, 46, 15);                        //设置标签的显示位置及大小
        add(label1);                                             //将标签添加到窗体中
        ButtonGroup buttonGroup = new ButtonGroup();             //创建按钮组对象

        JRadioButton manRadioButton = new JRadioButton();        //创建单选按钮对象
        buttonGroup.add(manRadioButton);                         //将单选按钮添加到按钮组中
```

```
manRadioButton.setSelected(true);              //设置单选按钮默认为被选中
manRadioButton.setText("男");                  //设置单选按钮的文本
manRadioButton.setBounds(62, 36, 46, 23);      //设置单选按钮的显示位置及大小
add(manRadioButton);                           //将单选按钮添加到窗体中

JRadioButton womanRadioButton = new JRadioButton();
buttonGroup.add(womanRadioButton);
womanRadioButton.setText("女");
womanRadioButton.setBounds(114, 36, 46, 23);
add(womanRadioButton);

JLabel label2 = new JLabel();                  //创建标签对象
label2.setText("爱好: ");                      //设置标签文本
label2.setBounds(10, 70, 46, 15);              //设置标签的显示位置及大小
add(label2);                                   //将标签添加到窗体中

JCheckBox readingCheckBox = new JCheckBox();   //创建复选框对象
readingCheckBox.setText("读书");               //设置复选框的标签文本
readingCheckBox.setBounds(62, 67, 55, 23);     //设置复选框的显示位置及大小
add(readingCheckBox);                          //将复选框添加到窗体中

JCheckBox musicCheckBox = new JCheckBox();
musicCheckBox.setText("听音乐");
musicCheckBox.setBounds(123, 67, 68, 23);
add(musicCheckBox);

JCheckBox pingpongCheckBox = new JCheckBox();
pingpongCheckBox.setText("乒乓球");
pingpongCheckBox.setBounds(197, 67, 75, 23);
add(pingpongCheckBox);

JLabel label3 = new JLabel();                  //创建标签对象
label3.setText("学历: ");                      //设置标签文本
label3.setBounds(10, 110, 46, 15);             //设置标签的显示位置及大小
add(label3); //将标签添加到窗体中
String[] schoolAges = { "本科", "硕士", "博士" };     //创建选项数组
JComboBox<String> comboBox = new JComboBox<String>(schoolAges);
//创建选择框对象
comboBox.setEditable(true);                    //设置选择框为可编辑
comboBox.setMaximumRowCount(3);                //设置显示行数
comboBox.addItem("大专");                      //添加一个选项
comboBox.setSelectedItem("本科");              //设置索引为 0 的选项被选中
comboBox.setBounds(62, 110, 104, 21);          //设置选择框的显示位置及大小
add(comboBox);                                 //将选择框添加到窗体中

JLabel label4 = new JLabel();                  //创建标签对象
label4.setText("姓名: ");                      //设置标签文本
label4.setBounds(10, 10, 46, 15);              //设置标签的显示位置及大小
add(label4);                                   //将标签添加到窗体中

JTextField textField = new JTextField();       //创建文本框对象
```

```
        textField.setHorizontalAlignment(JTextField.CENTER);      //居中对齐
        textField.setFont(new Font("", Font.BOLD, 12)); //设置文本框内容的字体样式
        textField.setBounds(62, 7, 100, 21);          //设置文本框的显示位置及大小
        add(textField);                               //将文本框添加到窗体中

        JLabel label5 = new JLabel();                 //创建标签对象
        label5.setText("密码: ");                     //设置标签文本
        label5.setBounds(180, 10, 46, 15);            //设置标签的显示位置及大小
        add(label5);                                  //将标签添加到窗体中

        JPasswordField passwordField = new JPasswordField();     //创建密码框对象
        passwordField.setEchoChar('￥');              //设置回显字符为‘￥’
        passwordField.setBounds(230, 7, 120, 21); //设置密码框的显示位置及大小
        add(passwordField);                           //将密码框添加到窗体中

        JLabel label6 = new JLabel();
        label6.setText("备注: ");
        label6.setBounds(10, 150, 46, 15);
        add(label6);

        JTextArea textArea = new JTextArea();         //创建文本域对象
        textArea.setColumns(15);                      //设置文本域显示文字的列数
        textArea.setRows(3);                          //设置文本域显示文字的行数
        textArea.setLineWrap(true);                   //设置文本域自动换行

        JScrollPane scrollPane = new JScrollPane();   //创建滚动面板对象
        scrollPane.setViewportView(textArea);         //将文本域添加到滚动面板中
        Dimension dime = textArea.getPreferredSize();//获得文本域的首选大小
        scrollPane.setBounds(62, 145, dime.width, dime.height);
        //设置滚动面板的位置及大小
        add(scrollPane);                              //将滚动面板添加到窗体中
        setVisible(true);                             //设置窗体可见，默认为不可见
        }
    public static void main(String args [] ){
        Sample5_4 frame = new Sample5_4();
    }
}
```

运行结果如图 5.5 所示。

图 5.5　例 5-4 运行结果

5.5 常用布局

布局是指组件在容器中的排列方式。Java 中采用 AWT 包中的布局类实例，管理容器中组件的布局设计。容器使用布局对象管理组件布局的基本步骤是：首先，创建容器对象；然后，创建布局对象；接下来调用容器的 setLayout(布局对象)方法设置容器所使用的布局；最后，将组件添加到容器中，添加时需按指定的布局方式添加。

常用布局如下。

1. FlowLayout 布局

使用 FlowLayout 创建的对象通常称为流布局对象，它的默认布局方式为由上到下、由左到右、居中排列组件。组件之间水平和垂直间隔为 5px，组件的大小为默认的最佳大小。此布局下如果改变组件大小，需调用组件的 setPreferredSize(Dimension preferredSize)方法。

2. BorderLayout 布局

使用 BorderLayout 创建的对象也称为边界布局对象，它的默认布局方式是把整个容器分为 5 个区域，分别为东、西、南、北、中。这 5 个区域采用 BorderLayout 类的 5 个静态常量表示，分别为 EAST、WEST、SOUTH、NORTH、CEHTER。这 5 个区域中间的那个区域最大，且每个区域只能放一个组件。该组件占满相应整个区域。边界布局的组件默认水平和垂直间隔为 0px，可以使用 BorderLayout 类的 setHgap(int h)和 setVgap(int v)修改，将组件添加到采用 BorderLayout 布局的容器时必须指明添加的区域。

3. GridLayout 布局

使用 GridLayout 创建的对象也称为网格布局对象，它的默认布局方式是将容器平均分成若干行和列，从而构成若干大小均等的网格。每个网格可以添加一个组件，添加组件时默认由上向下、由左向右添加到每个网格中，并占满整个网格区域。网格布局中组件之间默认水平和垂直间隔为 0px，修改方法同 BorderLayout。

4. CardLayout 布局

使用 CardLayout 创建的对象也称为卡片布局对象，卡片布局默认的布局方式是将多个组件层叠地放在容器中，最先加入容器的组件放在最上面，以后添加的依次向下排列；一个时刻容器只显示这些层叠组件中的一个，并且该组件占满容器空间。添加在卡片布局容器中的组件不仅有先后顺序，而且每个组件还有一个特有的字符串标识符，可以依据组件的序号调用组件，也可以依据标识符调用组件。

5. null 布局

在设置为 null(空布局)的容器中，组件通常使用 setBounds(int x, int y, int width, int height)方法指定组件的大小以及在容器中显示的位置。

5.6 事件处理

图形用户界面是人机交互的接口，用户针对图形界面的任何操作都会产生事件。例如，在文本框中输入文本、单击按钮、改变窗口大小等，不同操作会产生不同事件。Java 通过事

件处理机制完成应用程序响应用户的不同操作。本节主要介绍 Java 事件处理机制及常用事件处理方法。

5.6.1 Java 事件处理概述

1. 事件处理相关概念

学习 Java 事件处理机制必须首先掌握事件源、监听器和事件接口概念。

(1) 事件源。

发生事件的对象称为事件源。例如,在文本框中输入文本操作,事件源为文本框;单击按钮操作,事件源为按钮;改变窗口大小,事件源为窗口等。

(2) 监听器。

用于监听事件源发生事件的对象。例如,用户单击按钮操作需要应用程序做出响应,此时可以给按钮添加监听器,以监听事件的发生并捕获事件。

(3) 事件接口。

事件接口中定义了处理事件的空方法,作用是指明事件发生时需要调用哪个方法处理事件以响应用户的操作。

2. 事件处理机制

当事件源发生某种事件时,如动作事件、鼠标事件、键盘事件、焦点事件等,Java 运行环境自动创建封装了该事件的对象,同时添加到事件源用于侦听事件发生的监听器,就会侦听到事件的发生,并捕获事件对象,然后监听器自动调用用于处理该事件的方法,处理侦听到的事件。不同事件有不同的处理方法,这些方法定义在相应事件处理的接口中,并在事件监听器对象所在类中具体实现。

3. 事件处理方法

(1) 为事件源添加监听器。

用户的不同操作会产生不同事件,并且同一操作可以同时产生多种事件。例如,单击按钮可以产生动作事件(Action Event),也可以产生鼠标事件(Mouse Event)。如果应用程序只需要为用户单击按钮操作做出响应,可以只为事件源(按钮)添加一个监听器即可;如果需要为用户单击按钮操作产生的不同事件分别做出响应,此时需要为事件源添加多个监听器,分别监听并捕获不同的事件。Java 中,事件源调用 addxxxListener(监听器对象)方法为事件源添加事件监听器。

例如,为按钮 button 添加动作事件监听器 actionListener 的代码如下。

```
button.addActionListener(actionListener);
```

为按钮 button 添加鼠标事件监听器 mouseListener 的代码如下。

```
button.addMouseListener(mouseListener);
```

(2) 实现事件接口。

不同事件调用不同的方法处理。当监听器侦听到事件发生时将自动调用对应的方法处理事件。这些方法必须在监听器对象所在类中具体实现。例如,用于侦听动作事件的监听器 actionListener,当侦听到 button 上的动作事件时就会自动调用 actionPerformed()方法,该方法在动作事件接口 ActionListener 中声明,在 actionListener 对象所在类中实现。代码

如下。

```
A actionListener = new A();
class A implements ActionListener{
    …
    public void actionPerformed(ActionEvent e){
        …
    }
}
```

5.6.2 Java 常用事件处理

1. 动作事件

(1) 动作事件概述。

当动作事件(ActionEvent)发生时,Java 运行环境将创建一个 ActionEvent 类的对象,用于封装刚刚发生的动作事件。ActionEvent 类中有以下两个常用方法。

public object getsource():获取事件源对象。

public string getActionCommand():获取与该事件相关的命令字符串。

与动作事件相关的事件接口为 ActionListener,该接口中仅声明了一个方法:

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

当动作事件发生时,监听器自动调用 actionPerformed()方法,并将 Java 运行环境创建的动作事件对象作为参数传递到该方法中。

(2) 动作事件举例。

Java 中许多组件可以产生动作事件,如按钮、文本框、菜单项、密码框和单选钮等,下面举例说明动作事件的处理过程。

例 5-5 动作事件。

源文件为 Sample5_5.java。代码如下。

```
import javax.swing.*;
import java.awt.event.*;                                    //引入事件处理所需类
public class Sample5_5{
     private JFrame f;
     public Sample5_5(){
         f = new JFrame("主窗口");
         f.setSize(300,200);
         JMenuBar menubar = new JMenuBar();
         f.setJMenuBar(menubar);
         JMenu menu1 = new JMenu("系统");
         menubar.add(menu1);
         JMenuItem item1_1 = new JMenuItem("退出");
         menu1.add(item1_1);
         CommandListener ol = new CommandListener();        //创建监听器对象
         item1_1.addActionListener(ol);                     //菜单项添加动作事件监听器
         f.setVisible(true);
      }
     public static void main(String[] args){
         Sample5_5 t = new Sample5_5();
```

```
        }
        class CommandListener implements ActionListener{     //实现 ActionListener 接口
            public void actionPerformed(ActionEvent e){      //重写 actionPerformed()方法
                System.exit(0);                              //退出应用程序
            }
        }
    }
```

运行结果如图 5.6 所示。单击“退出”菜单项将触发动作事件,然后自动调用 ActionListener 接口中由 CommandListener 类重写的 actionPerformed()方法,实现退出应用程序功能。

图 5.6　例 5-5 运行结果

2. 鼠标事件

(1) 鼠标事件概述。

鼠标事件类为 MouseEvent。该类的常用方法如下。

.getX():获取鼠标在事件源上的横坐标。

.getY():获取鼠标在事件源上的纵坐标。

.getSource():获取事件源对象。

.getClickcount():获取鼠标单击的次数。

. getModifiers():获取鼠标左键或右键。

与鼠标事件相关的接口包括 MouseListener 和 MouseMotionListener。MouseListener 接口中声明了 5 个方法,分别用于处理鼠标的 5 种操作。

.mousePressed(MouseEvent e):用于处理鼠标键按下事件。

.mouseReleased(MouseEvent e):用于处理鼠标键释放事件。

.mouseEntered(MouseEvent e):用于处理鼠标移入事件源对象事件。

.mouseExited(MouseEvent e):用于处理鼠标移出事件源对象事件。

.mouseClicked(MouseEvent e):用于处理单击鼠标键事件。

MouseMotionListener 接口中声明了两个方法,分别用于处理鼠标在事件源上的移动和拖动事件。

.mouseDragged(MouseEvent e):用于处理鼠标在事件源对象上拖动事件。

.mouseMoved(MouseEvent e):用于处理鼠标在事件源对象上移动事件。

(2) 鼠标事件举例。

Java 中所有组件都可以触发鼠标事件,下面通过实例演示鼠标事件的处理过程。

例 5-6　鼠标事件。

源文件为 Sample5_6.java。代码如下。

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
public class Sample5_6{
    Sample5_5 mainWindow;
    JFrame f;
    JButton button1, button2;
    JLabel label1,label2;
    JTextField textField;
    JPasswordField passwordField;
```

```
public Sample5_6(){
    f = new JFrame();
    f.setTitle("登录窗口");
    f.setBounds(100, 100,300, 200);
    f.setLayout(null);
    f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    button1 = new JButton();
    button1.setBounds(10,85, 70, 23);
    button1.setText("确定");
    f.add(button1);
    button2 = new JButton();
    button2.setBounds(100,85, 70, 23);
    button2.setText("退出");
    f.add(button2);

    label1 = new JLabel();
    label1.setText("用户名:");
    label1.setBounds(10, 10, 56, 15);
    f.add(label1);
    textField = new JTextField();
    textField.setHorizontalAlignment(JTextField.CENTER);
    textField.setFont(new Font("", Font.BOLD, 12));
    textField.setBounds(62, 7, 110, 21);
    f.add(textField);
    label2 = new JLabel();
    label2.setText("密码:");
    label2.setBounds(10, 40, 46, 15);
    f.add(label2);
    passwordField = new JPasswordField();
    passwordField.setEchoChar('¥');
    passwordField.setHorizontalAlignment( JPasswordField.CENTER);
    passwordField.setBounds(62, 37, 110, 21);
    f.add(passwordField);
    MListener ml = new MListener();                    //创建监听器对象
    button1.addMouseListener(ml);                      //按钮添加鼠标事件监听器
    button2.addMouseListener(ml);
    f.setVisible(true);
}
public static void main(String args[]){
    Sample5_6 frame = new Sample5_6();
}
class MListener implements MouseListener{             //实现鼠标事件接口
    public void mouseClicked(MouseEvent e) {           //重写单击鼠标事件方法
        JButton buttonName = (JButton)e.getSource(); //获取事件源
        if(buttonName.getText().equals("确定")){    //判断事件源名称是否为"确定"
            String username = textField.getText();  //获取用户名
            String password = new String(passwordField.getPassword());     //获取密码
            if(username.equals("admins") && password.equals("admins")) {
                mainWindow = new Sample5_5();
            }
            else {
```

```
                    JOptionPane.showMessageDialog(null,"用户名或密码错误,输入无效!","消
息对话框",JOptionPane.WARNING_MESSAGE);
                }
            }
            else{
                System.exit(0);
            }
        }
        public void mouseEntered(MouseEvent e){}          //根据需要重写该接口中的其余方法
        public void mousePressed(MouseEvent e){}
        public void mouseReleased(MouseEvent e){}
        public void mouseExited(MouseEvent e){}
    }
}
```

运行结果如图 5.7 所示。

图 5.7　例 5-6 运行结果

说明：在“用户名”文本框中输入用户名，在“密码”框中输入密码；然后，单击“确定”按钮，判断用户名和密码是否都是“admins”，如果是则打开如图 5.6 所示主窗口；否则使用消息对话框提示用户名或密码错误。

3. 键盘事件

(1) 键盘事件概述。

用于封装键盘事件的类为 KeyEvent，该类常用方法如下。

.getsource()：获取事件源。

.getKeychar()：获取按键字符。

.getKeyCode()：获取按键 keyCode 值，即键码值。

.isActionKey()：判断按键是否为“动作”键。

.isControlDown()：判断按键是否为 Ctrl 键。

.isAltDown()：判断按键是否为 Alt 键。

.isShiftDown()：判断按键是否为 Shift 键。

与键盘事件相关的事件接口为 keyListener，该接口中定义了三个方法，分别用于处理按下键、释放键以及按键事件。

.keyPressed(keyEvent e)：用于处理按下键盘上某个键时触发的事件。

.keyReleased(keyEvent e)：用于处理释放键盘上某个键时触发的事件。

.keyTyped(keyEvent e)：用于处理键盘上某个键被按下又释放时触发的事件。

(2) 键盘事件举例。

在图形界面中,当某个组件获得焦点时,按下键盘上的键则会触发键盘事件,下面通过实例演示键盘事件的处理过程。

例 5-7 键盘事件。

源文件为 Sample5_7.java。代码如下。

```
import javax.swing.*;
import javax.swing.event.TreeSelectionEvent;
import javax.swing.event.TreeSelectionListener;
import javax.swing.table.DefaultTableModel;
import javax.swing.tree.DefaultMutableTreeNode;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;

public class Sample5_7 extends JFrame {
    private JTree tree;
    private JTable table;

    public Sample5_7() {
        //创建树结构
        DefaultMutableTreeNode root = new DefaultMutableTreeNode("Root");
        DefaultMutableTreeNode child1 = new DefaultMutableTreeNode("Child 1");
        DefaultMutableTreeNode child2 = new DefaultMutableTreeNode("Child 2");
        root.add(child1);
        root.add(child2);
        tree = new JTree(root);
        //创建表格
        String[] columnNames = {"Name", "Value"};
        Object[][] data = {
                {"Item 1", 10},
                {"Item 2", 20},
                {"Item 3", 30}};
        DefaultTableModel model = new DefaultTableModel(data, columnNames);
        table = new JTable(model);
        //为树添加键盘事件监听器,并使用键盘适配器类完成事件处理
        tree.addKeyListener(new KeyAdapter() {
            public void keyPressed(KeyEvent e) {
                if (e.getKeyCode() == KeyEvent.VK_ENTER) {          //如果是 Enter 键
                    DefaultMutableTreeNode selectedNode = (DefaultMutableTreeNode) tree.
getLastSelectedPathComponent();
                    if (selectedNode != null) {
                        JOptionPane.showMessageDialog(Sample5_7.this, "你按下了 Enter 键,
选中的节点是: " + selectedNode);
                    }
                }
            }
        });
        //为树添加选择事件监听器
        tree.addTreeSelectionListener(new TreeSelectionListener() {   //实现选择事件接口
            public void valueChanged(TreeSelectionEvent e) {
```

```
                DefaultMutableTreeNode selectedNode = (DefaultMutableTreeNode) tree.
getLastSelectedPathComponent();                                    //获取选中节点
                if (selectedNode != null) {
                    JOptionPane.showMessageDialog(Sample5_7.this, "你选择了节点: " +
selectedNode);
                }
            }
        });
        //创建分割面板,分别显示树组件和表格组件
        JSplitPane splitPane = new JSplitPane(JSplitPane.HORIZONTAL_SPLIT, new JScrollPane
(tree), new JScrollPane(table));
        splitPane.setDividerLocation(200);          //设置分割面板分隔条位置
        add(splitPane);                             //将分割面板添加到窗口
        setTitle("Key Event Example");
        setSize(400, 300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setVisible(true);
    }
    public static void main(String[] args) {
        new Sample5_7();
    }
}
```

运行结果如图 5.8 所示。

(a) 例5-7运行初始界面　(b) 选中节点Root并按下Enter键消息提示界面　(c) 单击Child1节点后消息提示界面

图 5.8　例 5-7 运行结果

说明：使用 DefaultMutableTreeNode 构建树的节点，并且用 JTree 呈现；利用 DefaultTableModel 构建表格数据模型，再用 JTable 显示；为 JTree 添加键盘事件监听器，对 VK_ENTER 按键事件进行监听，当按下 Enter 键时，弹出对话框显示选中的节点；为 JTree 添加选择事件监听器(TreeSelectionListener)，当选择节点时，弹出对话框显示选中的节点；使用 JSplitPane 把窗口分割成两部分，分别用于显示树和表格。

在 Java 中有很多事件接口，如 ActionListener、MouseListener、KeyListener 等，每个接口中可能包含多个方法，当应用程序不需要实现接口中的全部方法时，为了保证程序正确运行，除必须重写的方法外，其余方法也要进行空实现(见例 5-6)。为此，Java 提供了相应的事件适配器类，如 MouseAdapter、KeyAdapter 等。适配器类实现了相应事件接口中的方法，如 KeyAdapter 实现了 KeyListener 接口；然后，使用适配器的子类(通常使用匿名子类)重写所需要的方法即可，不必实现接口中的全部方法。事件适配器类是简化事件接口实现的一种方法，可以代替接口处理事件，本例中使用了 KeyAdapter 键盘事件适配器类。

除上述几个事件类及接口，Java API 中还提供了处理其他事件的事件类及相应接口。例如，FocusEvent 事件类及 FocusListener 事件接口，ItemEvent 事件类及 ItemListener 接

口，DocumentEvent 事件类及 DocumentListener 接口，TextEvent 事件类以及 TextListener 接口，WindowEvent 事件类以及 WindowListener 接口，等等。分别用于处理焦点事件、列表框以及复选框等选项事件、文本区文本编辑事件、文本框内容改变事件以及窗口事件，等等。需要处理哪种事件，就在事件源上添加对应事件的监听器，并在监听器对象所在类中实现相应事件接口即可。

5.7 对 话 框

对话框主要用于程序运行过程中简要的信息交互。例如，利用消息对话框进行重要信息提示或错误信息提示，利用确认对话框确认用户操作，利用文件对话框打开或保存文件等。常用对话框包括消息对话框、输入对话框、确认对话框、文件对话框等。另外，还可以使用 JDialog 类的子类创建自定义对话框。

对话框也是一种底层容器，但与窗口（JFrame 类的实例）略有不同，对话框必须依赖于某个窗口。对话框分为有模式和无模式两种，有模式对话框处于激活状态时，用户不可以再激活对话框所在程序中的其他窗口，直到该对话框事件处理结束；无模式对话框处于激活状态时，不堵塞其他线程的执行，可以再激活其他窗口。

1. 消息对话框、确认对话框、输入对话框

消息对话框主要用于重要信息提示或错误信息提示等；确认对话框主要用于确认用户操作；输入对话框主要用于用户输入信息。这三种对话框都是有模式对话框，采用 javax.swing 包中的 JOptionPane 类的静态方法获得。

（1）消息对话框采用 JOptionPane 类的静态方法 showMessagDialog()获得，具体语法格式如下。

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

（2）确认对话框采用 JOptionPane 类的静态方法 showConfirmDialog()获得，具体语法格式如下。

```
public static int showConfirmDialog(Component parentComponent, Object message, String title,
int optionType, int messageType)
```

（3）输入对话框采用 JOptionPane 类的静态方法 showInputDialog()获得，具体语法格式如下。

```
public static String showInputDialog(Component parentComponent, Object message, String title,
int messageType)
```

（4）参数说明。

① parentComponent：该参数用于指定对话框可见时的位置。如果该参数不为空，对话框在 parentComponent 组件的正前方居中显示；如果为 null，则在屏幕正前方显示。

② message：指定对话框上的显示信息。

③ title：指定对话框标题。

④ messageType：指定消息风格，messageType 的取值不同，对话框的外观不同，并且每一种风格都提供了一个默认的图标。messageType 取值如下。

- ERROR_MESSAGE
- INFORMATION_MESSAGE
- WARNING_MESSAGE
- QUESTION_MESSAGE
- PLAIN_MESSAGE

⑤ optionType：指定出现在对话框底部的按钮。optionType 取值如下。

- DEFAULT_OPTION
- YES_NO_OPTION
- YES_NO_CANCEL_OPTION
- OK_CANCEL_OPTION

实际应用中不局限于上述按钮，可以自己指定按钮集。

⑥ showConfirmDialog()方法返回如下整数之一。

- YES_OPTION
- NO_OPTION
- CANCEL_OPTION
- OK_OPTION
- CLOSED_OPTION

例 5-8 使用消息对话框、确认对话框、输入对话框。

源文件为 Sample5_8.java。代码如下。

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class Sample5_8{
     private JFrame f;
     private JButton button1;
     private JButton button2;
     private JTextArea textArea;
     private JPanel p;
     private JScrollPane jsp;
     public Sample5_8(){
          init();
     }
     private void init(){
          f = new JFrame("消息对话框、确认对话框、输入对话框举例");
          f.setSize(300,200);
          f.setResizable(false);
          f.addWindowListener(new WindowAdapter(){
                        public void windowClosing(WindowEvent evt){
                                  System.exit(0);
                        }
          });

         textArea = new JTextArea();
         textArea.setLineWrap(true);
```

```
        button1 =  new JButton("输入");
        button2 =  new JButton("退出");
        p = new JPanel();
        p.add(button1);
        p.add(button2);
        jsp = new JScrollPane();
        jsp.setViewportView(textArea);
        f.add(p,BorderLayout.SOUTH);
        f.add(jsp,BorderLayout.CENTER);
        CommandListener ol = new CommandListener();
         button1.addActionListener(ol);
         button2.addActionListener(ol);
         f.setVisible(true);
    }
     public static void main(String[] args){
         Sample5_8 t = new Sample5_8();
    }
     class CommandListener implements ActionListener{
         public void actionPerformed(ActionEvent e){
              String name = e.getActionCommand();
              if(name.equals("输入")){
                 String str = JOptionPane.showInputDialog(textArea,"输入长度小于 10 的字
符串!","输入对话框",JOptionPane.PLAIN_MESSAGE);
                  if(str!= null){
                       if(str.length()> 10){
   JOptionPane.showMessageDialog(null,"输入字符串长度大于 10,输入无效!","消息对话框",
JOptionPane.WARNING_MESSAGE);
                       }
                       else{
                           textArea.append(str + "\n");
                       }
                  }
              }
              else{
                  int i = JOptionPane.showConfirmDialog(null,"是否真的需要退出系统","退
出确认对话框", JOptionPane.YES_NO_CANCEL_OPTION);
                  if(i == 0){
                     System.exit(0);
                  }
              }

          }
      }
  }
```

运行结果如图 5.9 所示。

说明：单击图 5.9(a)中的“输入”按钮，在文本区组件正前方显示输入对话框，然后在输入对话框中输入信息(图 5.9(b))，输入结束后，单击输入对话框中的“确定”按钮，因为输入字符串长度大于 10，因此，在屏幕正前方显示消息对话框，提示“输入字符串长度大于 10，输入无效!”(图 5.9(c))；否则在文本区中显示在输入对话框中输入的字符串(图 5.9(d))；

(a) 例5-8初始运行界面　(b) 弹出输入对话框输入信息

(c) 使用消息对话框进行错误提示　(d) 在文本区中显示输入对话框中的信息

(e) 退出确认对话框

图 5.9　例 5-8 运行结果

单击窗口中的“退出”按钮，将在屏幕正前方显示确认对话框，要求用户确认是否退出系统(图 5.9(e))，如果单击“是”按钮，则退出系统。

2. 文件对话框

文件对话框主要用于打开和保存文件，采用 javax.swing 包中的 JFileChooser 类创建。采用 JFileChooser()构造方法创建的文件对话框初始不可见。可以使用 JFileChooser 类的 showSaveDialog()方法调用保存文件对话框；使用 showOpenDialog()方法调用打开文件对话框，使得文件对话框可见。具体语法格式如下。

```
JfileChooser chooser = new JfileChooser();              //创建有模式、初始不可见的文件对话框
public int showSaveDialog(Component parent)             //调用保存文件对话框
public int showOpenDialog(Component parent)             //调用打开文件对话框
```

其中，参数 parent 用于指定文件对话框显示位置。showSaveDialog()和 showOpenDialog()方法返回如下整数值之一。

- APPROVE_OPTION//确认
- CANCEL_OPTION//取消

例 5-9　使用文件对话框。

源文件为 Sample5_9.java。代码如下。

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

```
public class Sample5_9{
      JFrame f;
      JFileChooser chooser;
      JButton button1;
      JButton button2;
      JTextArea textArea;
      JPanel p;
      JScrollPane jsp;
      public Sample5_9(){
         init();
      }
      private void init(){
         f = new JFrame("文件对话框举例");
         f.setSize(300,200);
         f.setResizable(false);
         f.addWindowListener(new WindowAdapter(){
            public void windowClosing(WindowEvent evt){
                System.exit(0);
            }
         });
         chooser = new JFileChooser();
         textArea = new JTextArea();
         textArea.setLineWrap(true);
         button1 = new JButton("打开文件");
         button2 = new JButton("保存文件");
         p = new JPanel();
         p.add(button1);
         p.add(button2);
         jsp = new JScrollPane();
         jsp.setViewportView(textArea);
         f.add(p,BorderLayout.SOUTH);
         f.add(jsp,BorderLayout.CENTER);
         CommandListener ol = new CommandListener();
         button1.addActionListener(ol);
         button2.addActionListener(ol);
         f.setVisible(true);
      }
      public static void main(String[] args){
         Sample5_9 t = new Sample5_9();
     }

      class CommandListener implements ActionListener{
         public void actionPerformed(ActionEvent e){
            String name = e.getActionCommand();
            if(name.equals("打开文件")){
                int returnVal = chooser.showOpenDialog(textArea);
                if(returnVal == JFileChooser.APPROVE_OPTION) {
                    textArea.append("You chose to open this file: " + chooser.getSelectedFile().
getName() + "\n");
                }
             }
```

```
                else{
                    int returnVal = chooser.showSaveDialog(null);
                    if(returnVal == JFileChooser.APPROVE_OPTION) {
                        textArea.append("You chose to save this file: " + chooser.getSelectedFile().
getName() + "\n");
                    }
                }
            }
        }
    }
```

运行结果如图 5.10 所示。

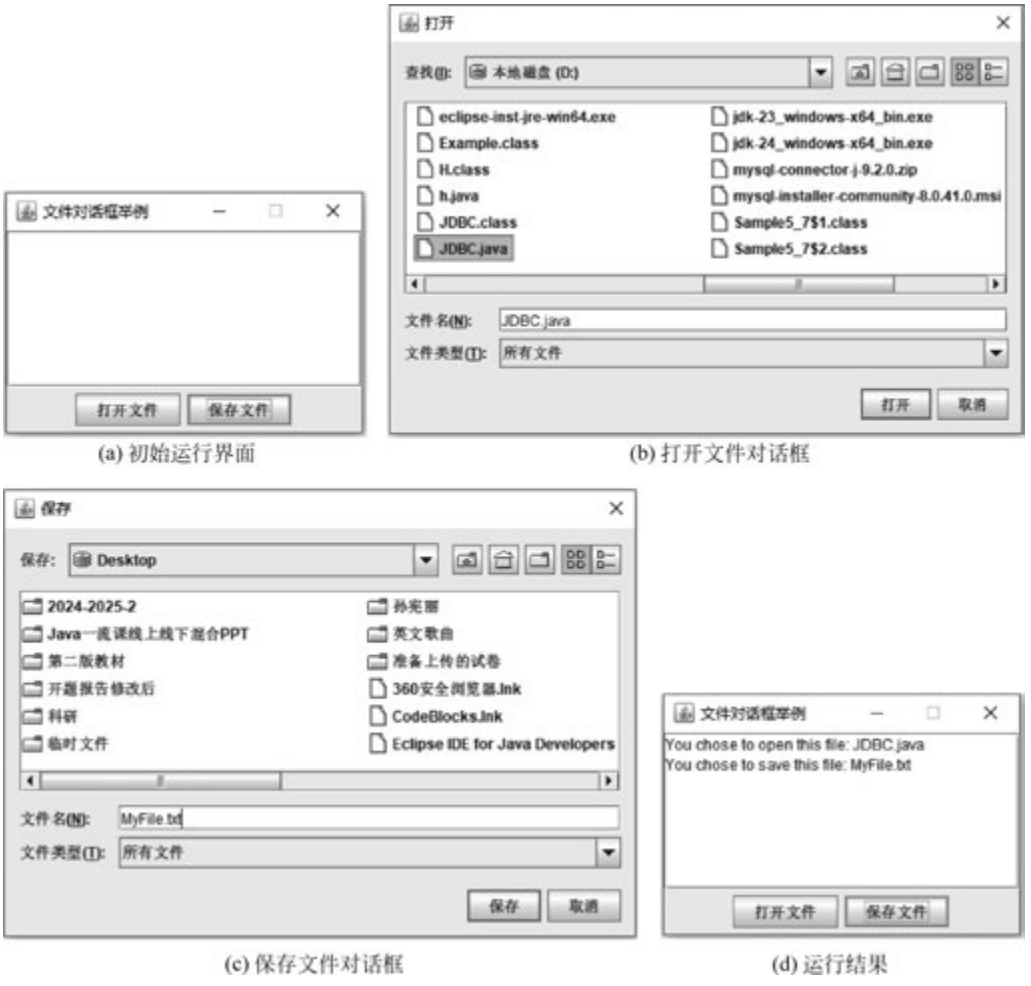

图 5.10　例 5-9 运行结果

说明：在图 5.10(a)中单击“打开文件”按钮，打开文件对话框将显示在“文本区”组件正前方(如图 5.10(b)所示)，在打开文件对话框中选择需要的文件，如果单击打开文件对话框中的“打开”按钮，则在文本区中显示所选文件名称；在图 5.10(a)中单击“保存文件”按钮，保存文件对话框将显示在屏幕正前方(如图 5.10(c)所示)，在保存文件对话框中选择路径，并输入文件名称，如果单击保存文件对话框中的“保存”按钮，则在文本区中显示保存文件名

称(如图 5.10(d)所示)。

5.8 绘制图形图像

在 Java Swing 包中,提供了一些可用于绘制图形、图像的类,常用类及方法如下。

1. JPanel 类

JPanel 是一个容器,常被用作绘图的区域。可以通过继承 JPanel 并重写其 paintComponent(Graphics g) 方法进行图形、图像的绘制。

2. Graphics 类

Graphics 类提供了一系列用于绘制图形、图像的方法,如绘制直线、矩形、椭圆等。

常用方法如下。

drawLine(int x1, int y1, int x2, int y2):绘制一条从点 (x1, y1) 到点 (x2, y2) 的直线。

drawRect(int x, int y, int width, int height):绘制一个矩形。

drawOval(int x, int y, int width, int height):绘制一个椭圆。

drawImage(Image img, int x, int y, int width, int height, ImageObserver observer):绘制图像。其中,img 是待绘制的图像;observer 用于接收图像加载状态的通知,如果不需要接收通知,可以设置为 null。

setColor(Color c):设置绘图的颜色。

fillRect(int x, int y, int width, int height):填充一个矩形。

fillOval(int x, int y, int width, int height):填充一个椭圆。

3. Graphics2D 类

Graphics2D 是 Graphics 类的子类,提供了更强大的绘图功能,如抗锯齿、渐变填充等。在 paintComponent()方法中可以将 Graphics 对象强制转换为 Graphics2D 对象来使用这些高级功能。

常用方法如下。

setRenderingHint(RenderingHints.Key hintKey, Object hintValue):设置渲染样式,如抗锯齿。

setStroke(Stroke s):设置线条的样式。

setPaint(Paint p):设置填充的样式,如渐变填充。

例 5-10 绘制基本图形。

源文件为 Sample5_10.java。代码如下。

```
import javax.swing.*;
import java.awt.*;
//自定义 JPanel 类用于绘图
class DrawingPanel extends JPanel {
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        Graphics2D g2d = (Graphics2D) g;
        g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);
                                                    //设置抗锯齿
        g2d.setColor(Color.RED);                    //设置绘图颜色
```

```
        g2d.drawLine(50, 50, 200, 50);                    //绘制直线
        g2d.setColor(Color.BLUE);
        g2d.drawRect(50, 100, 150, 100);                  //绘制矩形
        g2d.setColor(Color.GREEN);
        g2d.fillRect(50, 250, 150, 100);                  //填充矩形

        g2d.setColor(Color.ORANGE);
        g2d.drawOval(250, 50, 150, 100);                  //绘制椭圆
        g2d.setColor(Color.MAGENTA);
        g2d.fillOval(250, 250, 150, 100);                 //填充椭圆
    }
}
public class Sample5_10 {
    public static void main(String[] args) {
        JFrame frame = new JFrame("简单图形绘制示例");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(450, 400);
        DrawingPanel drawingPanel = new DrawingPanel();     //创建自定义 JPanel
        frame.add(drawingPanel);                            //将 JPanel 添加到 JFrame 中
        frame.setVisible(true);
    }
}
```

运行结果如图 5.11 所示。

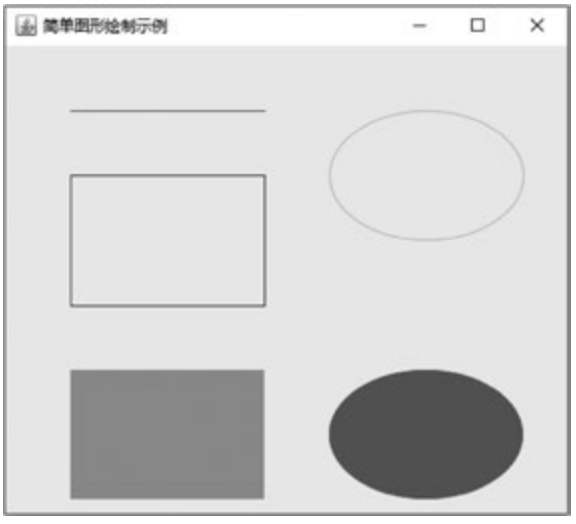

图 5.11　例 5-10 运行结果

例 5-11　绘制背景图片。

源文件为 Sample5_11.java。代码如下。

```
import javax.swing.*;
import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import javax.imageio.ImageIO;
```

```
class Sample5_11 extends JPanel {
      BufferedImage backgroundImage;
      public void paintComponent(Graphics g) {
          super.paintComponent(g);
          try {
              backgroundImage = ImageIO.read(new File("back.jpg"));   //读取图片文件
          } catch (IOException e) {
              e.printStackTrace();
          }
          if (backgroundImage != null) {
              g.drawImage(backgroundImage,0,0, getWidth(), getHeight(), this);    //绘制图片
          }
      }
      public static void main(String[] args) {
          JFrame frame = new JFrame("背景图片示例");
          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
          frame.setSize(450, 400);
          Sample5_11 drawingPanel = new Sample5_11();     //创建自定义 JPanel
          frame.add(drawingPanel);                        //将 JPanel 添加到 JFrame 中
          frame.setVisible(true);
        }
}
```

运行结果如图 5.12 所示。

图 5.12　例 5-11 运行结果

说明：back.jpg 文件需要存储在程序运行的目录下，或者根据实际情况修改文件路径。

5.9　播 放 音 频

在 Java 中，可以使用 javax.sound.sampled 包、javax.sound.midi 包或者第三方音频库中的类和接口来处理和播放.au、.aiff、.wav、.midi、.mp3 等音频文件。

其中，javax.sound.sampled 包常用类、接口和方法如下。

1. AudioSystem 类

AudioSystem 类用于处理音频系统资源，例如，获取音频输入流、获取音频格式等。

常用方法如下。

getAudioInputStream(File file)：从指定的文件中获取音频输入流。

getLine(Line.Info info)：获取指定类型的音频行。

2. Clip 接口

Clip 代表一个音频剪辑，可以将音频数据加载到其中并进行播放、暂停、停止等操作。常用方法如下。

open(AudioInputStream stream)：打开一个音频输入流，将音频数据加载到 Clip 中。

start()：开始播放音频。

stop()：停止播放音频。

close()：关闭 Clip，释放相关资源。

setFramePosition(int frames)：设置音频播放的起始帧位置。

例 5-12 简单的音乐播放器。

源文件为 Sample5_12.java。代码如下。

```
import javax.sound.sampled.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import java.io.IOException;
public class Sample5_12 extends JFrame {
    Clip clip;
    JButton playButton, stopButton, loopButton;
    boolean isLooping = false;
    public Sample5_12() {
        setTitle("简单音乐播放器");
        setSize(300, 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        playButton = new JButton("播放");
        stopButton = new JButton("停止");
        loopButton = new JButton("循环播放");
        stopButton.setEnabled(false);
        loopButton.setEnabled(false);
        playButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                playMusic();
            }
        });
        stopButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                stopMusic();
            }
        });
        loopButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
```

```
                loopMusic();
            }
        });
        JPanel panel = new JPanel();
        panel.add(playButton);
        panel.add(stopButton);
        panel.add(loopButton);
        add(panel);
    }
    private void playMusic() {
        try {
            File audioFile = new File("MyMusic.wav");  //指定播放文件
            AudioInputStream audioStream = AudioSystem.getAudioInputStream(audioFile);
                                                                    //获取音频输入流
            clip = AudioSystem.getClip();                  //创建 Clip 对象
            clip.open(audioStream);              //打开音频输入流,将音频数据加载到 Clip 中
            clip.start();                                  //播放音频文件
            playButton.setEnabled(false);
            stopButton.setEnabled(true);
            loopButton.setEnabled(true);
        } catch (UnsupportedAudioFileException | IOException | LineUnavailableException ex) {
            ex.printStackTrace();
        }
    }
    private void stopMusic() {
        if (clip != null) {
            clip.stop();                                   //停止播放
            clip.close();                                  //关闭 Clip,释放资源
            playButton.setEnabled(true);
            stopButton.setEnabled(false);
            loopButton.setEnabled(false);
            isLooping = false;
        }
    }
    private void loopMusic() {
        if (clip != null) {
            if (!isLooping) {                              //判断是否处于循环播放状态
                clip.loop(Clip.LOOP_CONTINUOUSLY);         //设置为循环播放
                loopButton.setText("停止循环");
                isLooping = true;
            } else {
                clip.loop(0);                              //停止循环
                loopButton.setText("循环播放");
                isLooping = false;
            }
        }
    }
    public static void main(String[] args) {
                Sample5_12 player = new Sample5_12();
                player.setVisible(true);
    }
}
```

运行结果如图 5.13 所示。

(a) 初始运行界面　(b) 播放音频后的界面　(c) 循环播放界面

图 5.13　例 5-12 运行结果

说明：运行程序首先出现如图 5.13(a)所示界面；单击“播放”按钮，播放指定音频文件，并显示如图 5.13(b)所示界面；单击“循环播放”按钮，设置循环播放，并显示如图 5.13(c)所示界面；单击“停止循环”按钮，停止循环；单击“停止”按钮，停止音乐播放。

小　　结

本章主要讲述了图形界面设计的基础知识，包括图形界面库的基本功能及特点，Swing 继承架构，图形界面设计思想及过程，窗口、菜单、常用组件以及对话框的使用方法；另外，还叙述了事件处理机制、过程及方法；最后介绍了图形、图像的绘制方法以及音频文件的处理方法。

思考练习 5

一、思考题

1. 简述 AWT 和 Swing 图形用户界面库的主要区别。
2. 简述 Swing 的继承架构。
3. 简述 Java 底层容器的种类及默认布局方式。
4. 简述窗口的作用和使用方法。
5. 简述菜单的制作方法。
6. 列出常用组件功能及创建方法。
7. 什么是布局？Java 中常用的布局有哪些？请简要描述其特点。
8. 什么是事件？写出你知道的 Java 事件。
9. 什么是事件源、监听器、事件接口？
10. 简述 Java 事件处理过程。
11. 什么是焦点？组件如何获得焦点？
12. 在 Java 事件处理中，事件适配器类的作用是什么？请举例说明其使用方法。
13. 当一个按钮被单击时，可能产生哪些事件？分别需要实现哪些事件接口来处理这些事件？
14. 简述对话框和窗口的不同之处。
15. 简述有模式对话框和无模式对话框的区别。
16. 简述创建消息对话框的方法。
17. 简述 showMessageDialog(Component parentComponent, Object message, String title, int messageType)方法中，messageType 的不同取值对应的默认图标。

二、判断题

1. Swing 继承于 AWT，其功能更强大、性能更优化，在不同操作系统上外观相同。（　　）

2. JFrame 默认布局为 FlowLayout。（　　）

3. JTextField 用于输入或显示单行文本，JTextArea 用于输入或显示多行文本。（　　）

4. 单选按钮与 ButtonGroup 类联合使用时，同一个时刻按钮组中只能有一个单选按钮处于被选中状态。（　　）

5. BorderLayout 布局中每个区域只能放一个组件。（　　）

6. 在 Java 中，事件源是发生事件的对象，监听器是监听事件源发生事件的对象。（　　）

7. 有模式对话框处于激活状态下不堵塞其他线程执行。（　　）

8. AudioSystem 类的 getAudioInputStream (File file) 方法用于获取音频输入流，Clip 接口用于将音频数据加载到其中并进行播放等操作。（　　）

三、选择题

1. 使用 JFrame 创建窗口时，设置窗口标题的方法是（　　）。

A. JFrame ([String s])

B. setTitle (String title)

C. setBounds (int x,int y,int width,int height)

D. setVisible (boolean b)

2. 以下组件中，可用于显示文本信息或图像信息的是（　　）。

A. JButton　　B. JTextField　　C. JLabel　　D. JTextArea

3. FlowLayout 布局的默认排列方式是（　　）。

A. 由上到下、由左到右、居中排列组件

B. 把容器分为东、西、南、北、中 5 个区域排列组件

C. 将容器平均分成若干行和列排列组件

D. 将多个组件层叠排列

4. 处理动作事件需要实现的接口是（　　）。

A. MouseListener　　B. KeyListener

C. ActionListener　　D. WindowListener

5. MouseEvent 类中，用于获取鼠标单击次数的方法是（　　）。

A. getX()　　B. getY()　　C. getClickcount()　　D. getSource()

6. 使用 JOptionPane 类创建消息对话框的方法是（　　）。

A. showConfirmDialog()　　B. showInputDialog()

C. showMessageDialog()　　D. 以上都不是

7. 文件对话框采用 javax.swing 包中的（　　）类创建。

A. JOptionPane　　B. JFileChooser　　C. JDialog　　D. JFrame

8. Graphics 类中，用于绘制矩形的方法是（　　）。

A. drawLine(int x1, int y1, int x2, int y2)

B. drawRect(int x, int y, int width, int height)

C. drawOval(int x, int y, int width, int height)

D. fillRect(int x, int y, int width, int height)

四、程序填空题

在登录窗口中,单击“确定”按钮时显示“Hello!”,单击“取消”按钮时清空文本框和密码框信息。

```
import Java.awt. * ;
import Javax.swing. * ;

_____
public class Test _____ implements ActionListener {
    JLabel label1,label2;
    JTextField textField;
    JButton button1,button2;
    JPasswordField passwordField;
    public Test() {
        super();
        setTitle("登录窗口");
        setBounds(100, 100,200, 200);
        setLayout(null);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        label1 = new JLabel();
        label1.setText("用户名");
        label1.setBounds(10, 10, 46, 15);
        add(label1);
        textField = new JTextField();
        textField.setBounds(62, 10, 100, 21);
        add(textField);
        label2 = new JLabel();
        label2.setText("密码");
        label2.setBounds(10, 50, 46, 15);
        add(label2);
        passwordField = new JPasswordField();
        passwordField.setBounds(62, 50, 100, 21);
        add(passwordField);
         button1 = new JButton();
        button1.setBounds(10,90, 70, 23);
         button1.setText("确定");
        button1.addActionListener(this);
        add(button1);
        button2 = new JButton();
        button2.setBounds(100,90, 70, 23);
        button2.setText("取消");
        _____;
         add(button2);
     }
public void actionPerformed(ActionEvent e) {
    JButton obj = (JButton)e.getSource();
    if(obj == button1){
         JOptionPane.showMessageDialog(null,"Hello!");
     }
    if(obj == button2){
       textField.setText("");
```

```
            passwordField.setText("");
             }
        }
         public static void main(String args[]) {
             _____;
             frame.setVisible(true);
         }
}
```

五、编程题

1. 编写应用程序实现以下功能：创建一个窗体，标题为“考试”，窗体中间放置一个密码框；窗体下方放置两个按钮，分别是“确认”和“退出”。要求：

（1）当单击“确认”按钮时，判断密码框中输入的密码是否为“user01”，如果密码正确，弹出消息对话框，提示“密码正确!”；否则在消息对话框中提示“密码不正确!”。

（2）当单击“退出”按钮时，弹出确认对话框，如果确认，则退出应用程序。

2. 编写应用程序实现以下功能：创建一个窗体，标题为“考试”，窗体中间放置一个文本区；另外，窗体中还包括一个菜单“操作”，该菜单包括两个菜单项，分别是“输入”和“退出”。要求：

（1）当单击“输入”菜单项时，弹出输入对话框，并将在输入对话框中输入的信息显示在文本区中。

（2）当单击“退出”菜单项时，退出应用程序。

3. 编写应用程序实现如图 5.14 所示界面，在窗体的中间存在一个文本区，下方存在一个文本框和一个发送按钮。要求：

（1）文本区要存在滚动条。

（2）单击“发送”按钮时，会将文本框的内容显示在文本区中。

（3）实现窗口关闭功能。

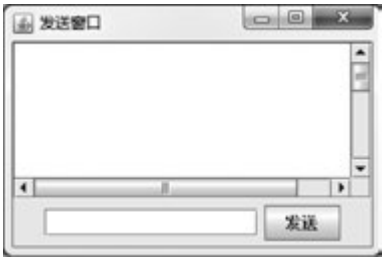

图 5.14　编程题 3 界面

4. 创建一个窗口，窗口中包括两个文本框、一个 Test 按钮和一个文本区；要求单击按钮时在文本区中显示两个文本框中的数字之和。

5. 创建一个窗口，窗口中包括一个“确定”按钮和一个文本框；要求在单击按钮时，判断在文本框中输入的内容是否为“exit”，如果是，则退出应用程序。