第 5 章　GUI 程序设计

【内容简介】

GUI(Graphics User Interface,图形用户界面)是指以图形的方式实现用户与计算机之间交互操作的应用程序界面。GUI 以图形的方式,借助菜单、按钮等标准界面元素和鼠标操作,帮助用户方便地向计算机发出命令、启动程序,并将程序的运行结果同样以图形的形式显示给用户。本章围绕 Java 语言 GUI 程序设计,详细介绍了 Java 图形 API、GUI 界面设计基础、事件处理机制、菜单和工具栏,以及对话框与其他常用组件的使用等主要知识。

本章将通过绘图软件和学生管理系统案例帮助读者系统地掌握 GUI 程序的设计方法和相关知识。

通过本章的学习,读者将初步具有综合应用所学知识设计 GUI 程序的能力。

【教学目标】

- ❑ 掌握建立 GUI 应用程序的步骤;
- ❑ 掌握窗口、对话框的使用方法;
- ❑ 掌握常用控件的使用方法;
- ❑ 理解事件的处理机制,掌握基本的事件编程方法;
- ❑ 掌握菜单和工具栏的设计方法和技巧;
- ❑ 能够综合应用 GUI 的有关知识编写应用程序。

5.1　Java 图形 API

Java 图形 API(应用程序接口)主要包含界面组件类及界面绘图类。

5.1.1　界面组件类

Java 提供了丰富的界面组件类,这些组件类的层次结构关系如图 5-1 所示。

AWT(Abstract Windows Toolkit,抽象窗口工具包)是 Sun 公司提供的用于图形界面编程的基础类库。它支持图形用户界面编程的功能包括:用户界面组件、事件处理模型、图形和图像工具、布局管理等。AWT 功能有限,仅适用于简单的 GUI 程序,而且对底层平台依赖较大。

Swing 是 Java 1.2 引入的新的 GUI 组件库。Swing 包括 javax.swing 包及其子包。Swing 独立于 AWT,但它是在 AWT 基础上产生的,其功能比 AWT 组件的功能更加强大。Swing 组件的层次结构如图 5-2 所示。

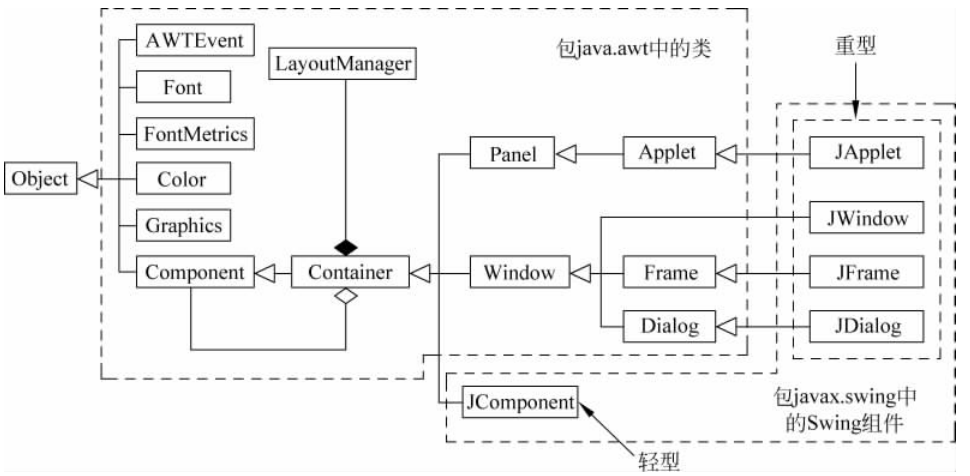

图 5-1　Java 图形 API 的层次结构

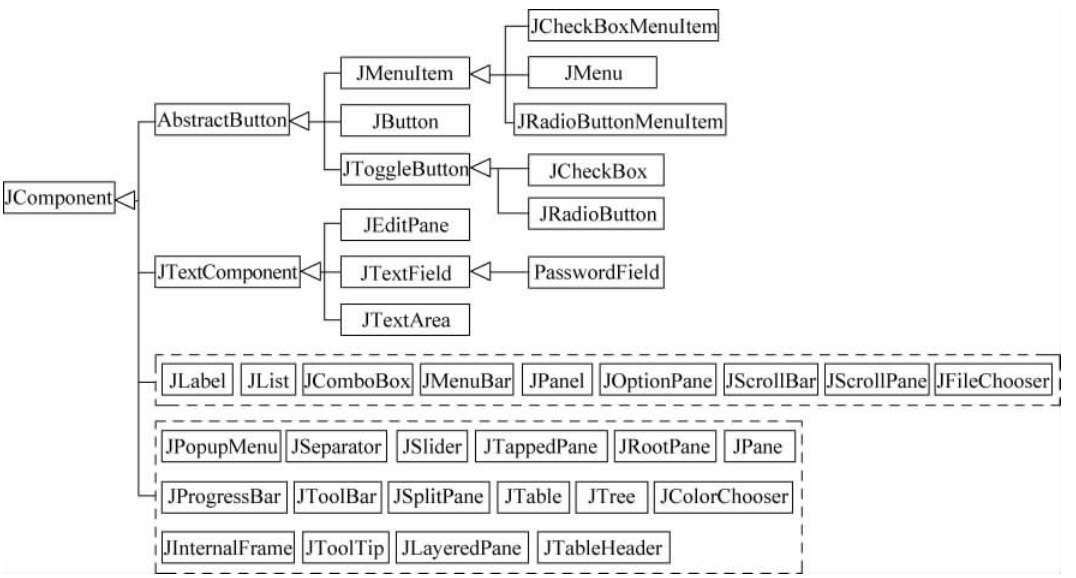

图 5-2　Swing 组件的层次结构

(1) Swing 是由纯 Java 实现的，没有本地代码，不依赖操作系统的支持，它比 AWT 组件具有更强的实用性。

(2) Swing 在不同的平台上表现一致，并且能提供本地窗口系统不支持的其他特性。

(3) Swing 采用了一种 MVC(Model-View-Controller，模型-视图-控制)设计模式，其中，模型用来保存内容，视图用来显示内容，控制器用来控制用户输入。

(4) Swing 采用可插入的外观，允许用户选择自己喜欢的界面风格。

(5) Swing 组件都以 J 开头，例如 JButton 和 JPanel 等，而相应的 AWT 是 Button 和 Panel。

(6) Swing 的包是 javax. swing，而 AWT 的包是 java. awt。

Swing 组件从功能上可分为：容器类和组件类。容器类都是由 Container 派生而来。一般 JFrame、JApplet、JDialog、JWindow 作为顶级容器，JPanel、JScrollPane、JSplitPane、JToolBar 作为中间容器，此外还有一些特殊的容器，如 JInternalFrame、JLayeredPane、JRootPane 等。组件类放置在容器中，构成 GUI 的基本要素，如 JButton、JCheckBox、JMenu、JRadioButton、JLabel、JList、JTextField、JTextArea、JScrollPane 等。Swing 的组件都是由 JComponent 派生而来。

Swing 组件不能取代 AWT 的全部类，只能替代 AWT 用户界面组件，辅助类仍保持不变。此外，Swing 组件还使用 AWT 的事件模型。

5.1.2 界面绘制类

Java 语言的类库中提供了丰富的绘图功能，其中大部分对图形、文本、图像的操作方法都定义在 Graphics 中。此外，Java2D API 增强了 AWT 的图形、文本和图像功能，使用 Java2D API 可以开发出更为强大的图形、图像程序。

1. Font 与 Color 类

java.awt 包中的 Font 类可以创建字体。构造方法是：

```
Font(String name, int style, int size)
```

其中，name 为字体的名称，如 Dialog、Frame、宋体、楷体等；style 为字体风格，有三个字体风格常量：Font.PLAIN(正常字体)、Font.BOLD(黑体)、Font.ITALIC(斜体)，可以组合使用，如 Font.BOLD+Font.ITALIC；size 为字体大小，以点来衡量，一个点(point)是 1/72 英寸。

java.awt 包中的 Color 类用于创建颜色。常用的构造方法如下。

(1) Color(int r, int g, int b)：用指定的红色、绿色和蓝色三个颜色分量值创建一个颜色，其中每个值在 0～255 范围内。

(2) Color 类中封装了一些颜色常量，如 Color.red，Color.blue 等，可以通过这些常量获得常用的颜色。

使用 Graphics 类的 setFont(Font font)方法可以设置字体；Graphics 类的 setColor(Color c)方法可以设置颜色。

2. Graphics 类

Graphics 是一个抽象类，用于在可视组件内绘图。该类有许多公有的方法，可以用作显示图像和文本、绘制和填充各种几何图形。

由于 Graphics 是一个抽象类，因此不能直接建立实例。可以从现有的图形对象或使用 Component 的 getGraphics()方法得到 Graphics 对象。通常在 paint()或 paintComponent()方法中进行绘图，可使用 Graphics 类型参数。组件在任何必需的时候将重新绘制自己，这时会调用这两个方法。AWT 程序使用 paint()，Swing 程序使用 paintComponent()。

Graphics 提供的基本绘图方法如下。

1）绘制直线

drawLine(int x1, int y1, int x2, int y2)：绘制一条直线。

2）绘制矩形

(1) drawRect(int x, int y, int width, int height)：绘制矩形轮廓。

(2) fillRect(int x, int y, int width, int height)：绘制填充矩形。

3）绘制椭圆

(1) drawOval(int x, int y, int width, int height)：绘制由矩形确定的椭圆轮廓。

(2) fillOval(int x, int y, int width, int height)：绘制由矩形确定的填充椭圆。

4）绘制文本

drawString(String str, int x, int y)：绘制由指定的字符串给出的文本。

5）绘制折线

(1) drawPolygon(int[] xPoints, int[] yPoints, int nPoints)：绘制由 x 和 y 坐标数组定义的闭合多边形的轮廓。

(2) drawPolygon(Polygon p)：绘制由指定的 Polygon 对象定义的多边形的轮廓。

(3) fillPolygon(int[] xPoints, int[] yPoints, int nPoints)：绘制由 x 和 y 坐标数组定义的填充闭合多边形。

(4) fillPolygon(Polygon p)：填充由指定的 Polygon 对象定义的填充多边形。

6）显示图像

(1) drawImage(Image img, int x, int y, ImageObserver observer)：observer 为渲染过程中的通知对象，一般为绘图容器；img 为要显示的图像；(x,y)为显示图像左上角的位置。

(2) drawImage(Image img, int x, int y, int w, int h, ImageObserver observer)：w, h 表示显示的宽度和高度。其他参数同上。

(3) drawImage(Image img,int dx1, int dy1, int dx2, int dy2, int sx1, int sy1, int sx2, int sy2, ImageObserver observer)：(dx1,dy1)和(dx2,dy2)分别为目标区域的左上角坐标和右下角坐标，(sx1,sy1)和(sx2,sy2)分别为源图像被绘制区域的左上角坐标和右下角坐标。

3. Graphics2D

Graphics2D 扩展了 Graphics 类，它可以提供几何形状、坐标转换、颜色管理以及文本布局等更精确的控制。Graphics2D 定义了几种方法，用于添加或改变图形中的属性。其中的大多数都采用某一代表特定属性的对象，如 Paint 或 Stroke 对象。可以修改 Graphics2D 的状态属性，来改变画笔宽度和笔画的连接方式，设置剪切路径以限定绘制区域，在绘制时平移、旋转、缩放或修剪对象以及定义用来填充形状的颜色和图案等。

Grahpics2D 仍然保留 Graphics 绘图方法，同时增加了许多新的绘图方法，如 Ellipse2D、Rectangle2D、RoundRectangle2D、Arc2D、Line2D 等。它们既可以用单精度浮点数指定坐标尺寸(如 Ellipse2D. Float)，也可以用双精度浮点数指定坐标尺寸(如 Ellipse. Double)。具体使用时，要先利用这些方法构造形状，然后再调用 draw()方法来进行绘图。

5.2 GUI 界面设计基础

5.2.1 窗口

窗口是最主要的用户界面。Java 创建无边窗口使用 JWindow；创建有边窗口使用 JFrame(框架)。JFrame 类由 Frame 类派生而来,是 Container 家族的一个子类,它一般作为顶级容器,用于创建框架窗口。框架窗口是一种带有边框、标题及关闭和最小化图标的窗口。GUI 应用程序通常至少使用一个框架窗口。

(1) 直接用 JFrame 类创建窗口,例如:

```
JFrame frame = new JFrame();
frame.setTitle("我的窗口");                    //设置标题
frame.setSize(300,200);                        //设置大小
frame.setVisible(true);                        //设置可见
```

(2) 继承 JFrame 创建窗口,例如:

```
import javax.swing.JFrame;
public class MyWindow extends JFrame {
    public MyWindow() {
        setTitle("我的窗口");
        setSize(300, 200);
        setVisible(true);
    }
    public static void main(String[] args) {
        new MyWindow();
    }
}
```

1. 窗口的属性设置

1) 设置标题

可通过 super(String title)调用基类的构造方法,或通过 setTitle(String title)方法设置标题。

2) 设置初始位置

可通过 setLocation(int x,int y)方法设置初始位置。如果希望窗口居中,需要先取出屏幕的位置,再计算出窗口的位置并进行设置。例如:

```
setSize(400,300);                              //设置出窗口大小
Dimensionsize = Toolkit.getDefaultToolkit().getScreenSize();   //取屏幕大小
setLocation((size.width - getWidth())/2,(size.height - getHeight())/2);
```

3) 设置大小

可通过 setSize(int width,int height)方法设置窗口大小。

4）使窗口最大化

窗口显示出来以后，可使用 setExtendedState(JFrame.MAXIMIZED_BOTH)方法使窗口最大化。

5）设置图标

可通过 setIconImage(Icon icon)方法设置窗口的图标。例如：

```
java.net.URL url = getClass().getResource("web.gif");      //图像要与类放在一起
Image image = Toolkit.getDefaultToolkit().getImage(url);
setIconImage(image);
```

或者

```
setIconImage((new ImageIcon("icon.gif")).getImage());      //图像要与工程放在一起
```

6）设置关闭行为

通过 setDefaultCloseOperation(int operation)定义关闭行为。operation 取值如下。

(1) DO_NOTHING_ON_CLOSE：当窗口关闭时，不做任何处理。

(2) HIDE_ON_CLOSE：当窗口关闭时，隐藏这个窗口。

(3) DISPOSE_ON_CLOSE：当窗口关闭时，隐藏并处理这个窗口。

(4) EXIT_ON_CLOSE：当窗口关闭时，退出程序。

(5) 默认是 HIDE_ON_CLOSE。

7）设置外观

在建立窗口实例前调用 JFrame.setDefaultLookAndFeelDecorated(true)方法，可使窗口采用 Swing 界面风格。

2. 将组件添加到窗体

将组件添加到窗体有以下两种方式。

(1) 用 getContentPane()方法获得内容面板，再向内容面板中加入组件。例如：

```
JButton b1 = new JButton("确定");
Container con = getContentPane();
con.add(b1);
```

(2) 建立一个中间容器(如 JPanel 或 JDesktopPane)，把组件添加到容器中，再用 setContentPane()方法把该容器设置为内容面板。例如：

```
JButton b1 = new JButton("确定");
JPanel p1 = new JPanel();                    //建立一个面板
p1.add(b1);                                  //把其他组件添加到 p1 中
setContentPane(p1);                          //把 p1 对象设置成内容面板
```

5.2.2　常用组件

1. 面板

JPanel 组件是一个中间容器，用于将小型的轻量级组件组合在一起，可以调用其 add()

方法将组件添加到面板。JPanel 的默认布局为 FlowLayout。

JPanel 的常用构造方法如下。

(1) JPanel()。

(2) JPanel(LayoutManager layout)：layout 指明布局方式。

例如,下面的语句建立两个面板。

```
JPanel p1 = new JPanel();                          //使用默认布局建立面板
JPanel p2 = new JPanel(new FlowLayout(FlowLayout.LEFT));   //流式左对齐
```

2. 标签

标签既可以显示文本也可以显示图像。主要的构造方法如下。

(1) JLabel()。

(2) JLabel(Icon icon)。

(3) JLabel(String text, Icon icon, int align)。

其中,text 表示使用的字符串；icon 表示使用的图标；align 表示水平对齐方式,其值可以为：LEFT、RIGHT、CENTER。

标签的主要方法有以下几个。

(1) void setFont(Font f)：设置字体。

(2) String getText()：获取文本。

(3) void setText(String text)：设置文本。

(4) voidsetIcon(Icon icon)：设置图标。

3. 复选按钮和单选按钮

复选按钮(JCheckBox)可以选择多项,而单选按钮(JRadioButton)只能选择一项。

复选按钮的主要构造方法如下。

(1) JCheckBox()。

(2) JCheckBox(String text)。

(3) JCheckBox(String text, boolean selected)。

其中,text 为标题,selected 表示是否选择。

单选按钮的主要构造方法如下。

(1) JRadioButton()。

(2) JRadioButton(String text)。

(3) JRadioButton(String text, boolean selected) 。

其中,text 为标题,selected 表示是否选择。

单选按钮在使用时需要分组,建立组的方法如下。

```
JRadioButton rabSexM = new JRadionButton("男", true);
JRadioButton rabSexM = new JRadionButton("女", false);
ButtonGroup group = new ButtonGroup();             //建立组
group.add(rabSexM);                                //将单选按钮添到组中
group.add(rabSexM);                                //将单选按钮添到组中
```

复选按钮和单选按钮常用的方法有以下几个。

(1) String getActionCommand()：获得 actionCommand。

(2) void setActionCommand(String actionCommand)：设置 actionCommand。

(3) boolean isSelected()：判断是否处于选中状态。

(4) setSelected(boolean b)：设置选中状态。

4. 按钮

按钮(JButton)是 AbstractButton 的子类。主要的构造方法有以下几个。

(1) JButton()：创建一个无标题的按钮。

(2) JButton(Icon icon)：创建一个图标按钮。

(3) JButton(String text)：创建一个带有指定标题的按钮。

(4) JButton(String text, Icon icon)：创建一个既有标题又有图标的按钮。

主要的方法有以下几个。

(1) void addActionListener(ActionListener l)：注册行为事件。

(2) void setMnemonic(char mnemonic)：设置热键。

(3) void setToolTipText(String s)：设置提示文本。

(4) void setEnabled(boolean b)：设置是否响应事件。

(5) void setPressedIcon(Icon pressedIcob)：设置按下状态的图标。

(6) void setRolloverIcon(Icon rollerIcon)：设置转滚状态的图标。

(7) void setRolloverEnabled(boolean b)：用于设置是否可转滚。

5. 切换按钮

JToggleButton 是一个可切换的按钮，与 JButton 类似，主要的差别在于一般的按钮按下去会自动弹回来，而 JToggleButton 按钮按下去会处于按下状态，不会弹回来，除非再按一次。其构造方法与 JButton 的类似，只是可多一个参数，用于表明是否选中。例如：

JToggleButton(Icon icon,boolean selected)：建立一个有图像但没有文字的 JToggleButton，且设置其初始状态(有无选取)。

可以调用 setSelectedIcon()、setRolloverSelectedIcon()、setDisableSelectedIcon()方法设置不同状态的图标。

切换按钮也可以分组，属于同一组的切换按钮是互斥的。分组方法类似于单选按钮。

5.2.3　界面布局

组件是要放在容器里的。Java 为了实现跨平台的特性并且获得动态的布局效果，将容器内的所有组件安排给一个布局管理器负责管理。容器可以通过选择不同的布局管理器来决定布局。布局管理器主要包括：FlowLayout，BorderLayout，GridLayout，CardLayout。

1. 流式布局

通过此布局,组件从左上角开始按从左到右、从上到下的方式排列。默认的情况下,组件居中,间距为 5 个像素。它是面板的默认布局。

FlowLayout 的构造方法有以下几个。

(1) FowLayout():生成一个默认的流式布局。

(2) FlowLayout(int alignment):可以设定每一行组件的对齐方式。alignment 可取的值有:FlowLayout.LEFT,FlowLayout.RIGHT,FlowLayout.CENTER。

(3) FlowLayout(int alignment,int horz,int vert):可以设定对齐方式以及通过参数 horz 和 vert 分别设定组件的水平和垂直间距。

例如,下面的示例演示了流式布局,运行效果如图 5-3 所示。

```
public class LayoutDemo extends JFrame {
    JButton b1, b2, b3, b4, b5;
    public LayoutDemo() {
        Container con = this.getContentPane();
        setLayout(new GridLayout(3, 3, 5, 5));  //3 行 3 列,间隔是 5 个像素
        b1 = new JButton("one");  b2 = new JButton("two");
        b3 = new JButton("three"); b4 = new JButton("four");
        b5 = new JButton("five");
        con.add(b1);con.add(b2);con.add(b3);con.add(b4);con.add(b5);
        setTitle("网格布局演示");              //设置窗口标题
        setSize(500, 400);
        setVisible(true);
    }
    public static void main(String[] args) {
        new LayoutDemo();
    }
}
```

图 5-3 流式布局演示

2. 边界布局

通过此布局,组件可以被置于容器的东、南、西、北、中位置。它是窗口(JWinodow),框架(JFrame)和对话框(JDialog)等的默认布局。

BorderLayout 的构造方法如下。

(1) BorderLayout( )：生成默认的边界布局。默认无间距。

(2) BorderLayout(int horz,int vert)：可以设定组件间的水平和垂直间距。

在设置成边界布局的容器中添加组件，需要指定组件的位置。如：

```
container.add(p1, BorderLayout.SOUTH);
```

或：

```
container.add(p1,"South");
```

图 5-4　边界布局演示

例如，下面的示例演示了边界布局，运行效果如图 5-4 所示。

```
public class LayoutDemo extends JFrame {
    JButton b1, b2, b3, b4, b5;
    public LayoutDemo() {
        Container con = this.getContentPane();
        b1 = new JButton("one"); b2 = new JButton("two");
        b3 = new JButton("three"); b4 = new JButton("four");
        b5 = new JButton("five");
        con.add(b1);con.add(b2);con.add(b3);con.add(b4);con.add(b5);
        setTitle("边界布局演示");                    //设置窗口标题
        setSize(500, 400);
        setVisible(true);
    }
    public static void main(String[] args) {
        new LayoutDemo();
    }
}
```

3. 网格布局

可将容器区域划分为一个矩形网格。通过此布局，组件按行和列排列，大小相同。

GridLayout 的构造方法如下。

(1) GridLayout()：生成一个单列的网格布局。默认无间距。

(2) GridLayout(int row,int col)：生成一个指定行数和列数的网格布局。

图 5-5　网格布局演示

(3) GridLayout(int row, int col, int horz, int vert)：可以设置组件之间的水平和垂直间距。

网格布局是以行为基准的，在组件数目多时自动扩展列，在组件数目少时自动收缩列，行数始终不变，组件按行优先顺序排列。

例如，下面的示例演示了网格布局，运行效果如图 5-5 所示。

```
public class LayoutDemo extends JFrame {
    JButton b1, b2, b3, b4, b5;
    public LayoutDemo() {
        Container con = this.getContentPane();
        setLayout(new GridLayout(3, 3, 5, 5));  //3 行 3 列,间隔是 5 个像素
        b1 = new JButton("one"); b2 = new JButton("two");
        b3 = new JButton("three"); b4 = new JButton("four");
        b5 = new JButton("five");
        con.add(b1);con.add(b2);con.add(b3);con.add(b4);con.add(b5);
        setTitle("网格布局演示");              //设置窗口标题
        setSize(500, 400);
        setVisible(true);
    }
    public static void main(String[] args) {
        new LayoutDemo();
    }
}
```

4. 卡片布局

CardLayout 提供了像管理一系列卡片一样管理组件的功能。采用这种布局,多个组件拥有同一个显示空间,不过一个时刻只能显示一个组件,可以进行翻页。

CardLayout 的构造方法如下。

(1) CardLayout():创建一个无间距的卡片布局。

(2) CardLayout(int hgap, int vgap):创建一个具有指定水平和垂直间距的卡片布局。

卡片布局的几个重要方法如下。

(1) void next(Container parent):显示下一页。

(2) void previous(Container parent):显示前一页。

(3) void first(Container parent):显示第一页。

(4) void last(Container parent):显示最后一页。

(5) void show(Container parent,String name):显示指定页。

在设置成卡片布局的容器中添加组件,需要指定组件的标题。例如:

```
container.add("card1",card1);                    //添加一个标题为 card1 的组件
```

5. 无布局

当容器的布局管理对象为空时,容器采用的是无布局方式。在无布局时,需要设置组件的位置和大小。例如:

```
Container.setLayout(null);                       //设置为无布局
JButton btn = new JButton("确定");               //建立按钮
btn.setBounds(20,30,20,100);                     //设置按钮位置和大小
container.add(btn);                              //添加 btn 的组件
```

除上面介绍的几种布局管理器外,Java 还提供了 BoxLayout、GridBagLayout 等多种布局管理器,实际应用中往往采用多种布局管理来完成复杂的图形用户界面。

5.2.4　案例 5-1　设计绘图软件界面

第 3 章中设计了一组图形类，但是还没有实现绘图软件的功能，这个案例先来设计界面。界面效果如图 5-6 所示。窗口分为上、左、中三个区。上区是绘图工具栏，提供图形类型选择、填充方式选择以及线条粗细选择工具；左区是颜色工具栏，用于选择颜色，最上面的色块表示当前颜色；中区为绘图区。

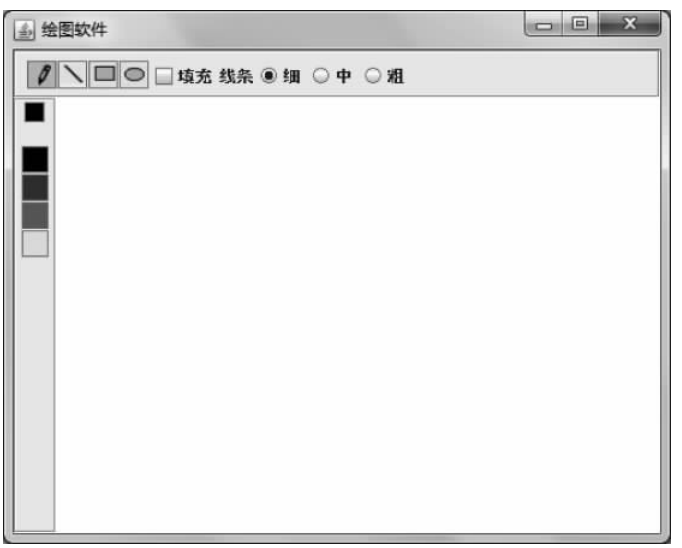

图 5-6　绘图软件界面

【技术要点】

(1) 整体界面布局符合边界布局的特点，因此，窗体的内容面板采用默认的布局方式(边界布局)。绘图工具栏(drawTool)放在 North，颜色工具栏(colorTool)放在 West，绘图区放在 Center。

(2) drawTool 和 colorTool 是两个面板，为了使面板有一定的宽度或高度，使用 setPreferredSize()预设大小。drawTool 和 colorTool 均采用流式布局，并设置了间隙。drawTool 的设置了左对齐，colorTool 的设置了居中。

(3) 绘图工具栏上的图形类型选择按钮利用 JToggleButton 实现，填充方式选择用 JCheckBox 实现，线条选择用 JRadioButton 实现，颜色选择用 JButton 实现，同时使用标签显示提示文本。

【设计步骤】

(1) 为项目准备图像素材，如下所示。

(2) 在 NetBeans 中新建一个 Java 应用程序项目，项目命名为 Exam5_2_4，将图像素材复制到项目文件夹根目录下。

(3) 在项目中建立包 exam5_2_4,并将案例 3-4 所建的类复制到该包下。

(4) 打开 DrawWindow,添加组件定义。

```
public class DrawWindow extends JFrame {
    JToggleButton b1, b2, b3, b4;                    //用于图形类型选择
    JCheckBox ch;                                    //用于填充类型选择
    JButton c0, c1, c2, c3, c4;                      //用于颜色选择
    JRadioButton j1, j2, j3;                         //用于线条选择
    DrawBoard drawBoard;                             //绘图面板
    public DrawWindow() {
      …
    }
}
```

(5) 在构造方法中建立组件,并将组件添加到窗体,代码如下所示。

```
public DrawWindow() {
    Container con = this.getContentPane();                    //取出内容面板
    //以下是建立绘图工具栏
    JPanel drawTool = new JPanel(new FlowLayout(FlowLayout.LEFT, 1, 5));
    drawTool.setPreferredSize(new Dimension(1, 36));          //预设大小
    drawTool.setBorder(BorderFactory.createEtchedBorder());   //设置边
    b1 = new JToggleButton(new ImageIcon("b0.jpg"), true);
    b1.setSelectedIcon(new ImageIcon("a0.jpg"));
    b1.setPreferredSize(new Dimension(22, 21));
    b2 = new JToggleButton(new ImageIcon("b1.jpg"), true);
    b2.setSelectedIcon(new ImageIcon("a1.jpg"));
    b2.setPreferredSize(new Dimension(22, 21));
    b3 = new JToggleButton(new ImageIcon("b2.jpg"), true);
    b3.setSelectedIcon(new ImageIcon("a2.jpg"));
    b3.setPreferredSize(new Dimension(22, 21));
    b4 = new JToggleButton(new ImageIcon("b3.jpg"), true);
    b4.setSelectedIcon(new ImageIcon("a3.jpg"));
    b4.setPreferredSize(new Dimension(22, 21));
    drawTool.add(new JLabel("  "));             //添加占位,使得后面的按钮与左侧有一定距离
    ButtonGroup bg1 = new ButtonGroup();                      //建立组
    bg1.add(b1); bg1.add(b2); bg1.add(b3); bg1.add(b4);       //将切换按钮添加到组
    drawTool.add(b1); drawTool.add(b2);
    drawTool.add(b3);  drawTool.add(b4);
    ch = new JCheckBox("填充", false);
    drawTool.add(ch);
    j1 = new JRadioButton("细", true);
    j2 = new JRadioButton("中", false);
    j3 = new JRadioButton("粗", false);
    ButtonGroup group = new ButtonGroup();                    //建立组
    group.add(j1);  group.add(j2);  group.add(j3);
    drawTool.add(new JLabel("线条"));
    drawTool.add(j1); drawTool.add(j2); drawTool.add(j3);
    //以下是建立颜色工具栏
    JPanel colorTool = new JPanel(new FlowLayout(FlowLayout.CENTER, 0, 1));
    colorTool.setPreferredSize(new Dimension(32, 1));
    colorTool.setBorder(BorderFactory.createEtchedBorder());
    c0 = new JButton(); c0.setBackground(Color.BLACK);
    c0.setPreferredSize(new Dimension(15, 15));
    c1 = new JButton(); c1.setBackground(Color.BLACK);
```

```
        c1.setPreferredSize(new Dimension(20, 20));
        c2 = new JButton(); c2.setBackground(Color.BLUE);
        c2.setPreferredSize(new Dimension(20, 20));
        c3 = new JButton(); c3.setBackground(Color.RED);
        c3.setPreferredSize(new Dimension(20, 20));
        c4 = new JButton(); c4.setBackground(Color.YELLOW);
        c4.setPreferredSize(new Dimension(20, 20));
        colorTool.add(c0);
        colorTool.add(new JLabel("      "));                    //占位
        colorTool.add(c1); colorTool.add(c2);
        colorTool.add(c3); colorTool.add(c4);
        drawBoard = new DrawBoard();
        con.add(drawTool, "North");
        con.add(colorTool, "West");
        con.add(drawBoard, "Center");
        setTitle("绘图程序");                                    //设置窗口标题
        setSize(500, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setVisible(true);
    }
```

(6) 保存并运行程序。

5.3　事件处理机制

在 GUI 应用程序中,用户操作(如单击鼠标或按某个键)、程序代码或系统内部都可以产生事件。在事件驱动机制中,应用程序代码可以响应事件来执行一系列的操作,称为事件处理。

5.3.1　事件处理模型

事件(Event)可以定义为程序发生了某些事情的信号。外部用户行为,如移动鼠标,单击鼠标和按键等,可以引发事件;系统内部,如时钟等,也可引发事件。

发生事件的对象称为事件的源对象,简称为事件源。例如,按钮是单击按钮事件的事件源。在 Java 中用事件类来描述事件,所有事件类的根类是 java.uti.EventObject。不同的事件对应不同的事件类。如单击按钮事件对应 ActionEvent 类,键盘事件对应 KeyEvent 类。事件对象包含与事件有关的信息。

Java 使用授权处理模型来处理事件:在事件源上引发事件,由监听者来处理事件。同一个事件源上可能发生多种事件,事件源可以把在其自身所有可能发生的事件分别授权给不同的事件处理者来处理,这个过程称为注册事件监听者。监听者是能够对事件进行监听,并能对发生的事件进行处理的对象。事件监听者类必须实现相应的监听者接口。监听者接口是定义在事件源和监听者之间的接口规范,它定义了事件处理方法的格式。事件处理模型如图 5-7 所示。

Java 为每个事件类型定义了一个接口,如表 5-1 所示。

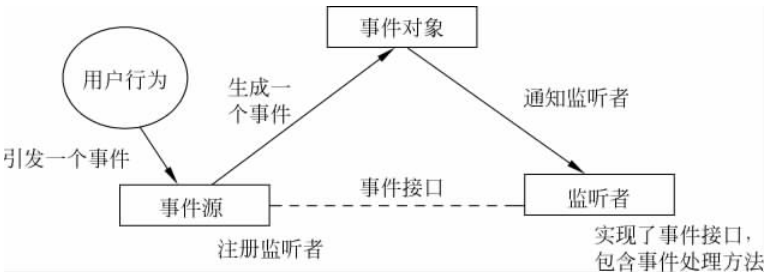

图 5-7 事件处理模型

表 5-1 事件类、监听者接口和事件处理方法

事 件 类	监听器接口	事件处理方法
ActionEvent	ActionListener	actionPerformed(ActionEvent e)
KeyEvent	KeyListener	keyPressed(KeyEvent e) keyReleased(KeyEvent e) keyTyped(KeyEvent e)
FocusEvent	FocusListener	focusGained(FocusEvent e) focusLost(FocusEvent e)
MouseEvent	MouseListener	mouseClicked(MouseEvent e) mousePressed(MouseEvent e) mouseReleased(MouseEvent e) mouseEntered(MouseEvent e) mouseExited(MouseEvent e)
	MouseMotionListener	mouseDragged(MouseEvent e) mouseMoved(MouseEvent e)
ItemEvent	ItemListener	itemStateChanged(ItemEvent e)
WindowEvent	WindowListener	windowActivated(WindowEvent e) windowClosed(WindowEvent e) windowClosing(WindowEvent e) windowDeactivated(WindowEvent e) windowDeiconified(WindowEvent e) windowIconified(WindowEvent e) windowOpened(WindowEvent e)

5.3.2 事件处理

1. 事件处理的基本步骤

1) 声明实现监听者接口

监听者类必须实现相应的监听者接口。例如,下面程序中的斜体部分声明实现 ActionListener 接口。

```
public class ActionEventDemo extends JFrame implements ActionListener {
```

```
    …
}
```

2）实现监听者接口方法

```
public class ActionEventDemo extends JFrame implements ActionListener{
    …
    public void actionPerformed(ActionEvent e) {
        if(e.getSource() == b1){
            JoptionPane.showMessage(null, "单击了按钮b1");
        }else {
            JoptionPane.showMessage(null, "单击了按钮b2");
        }
    }
}
```

3）注册监听者

为事件源注册监听者类对象。例如，在 ActionEventDemo 构造方法中增加以下两条语句。

```
public class ActionEventDemo extends JFrame implements ActionListener{
     JButton b1 = new JButton("b1");
     JButton b2 = new JButton("b2");
     public ActionEventDemo() {
         …
         b1.addActionListener(this);
         b2.addActionListener(this);
         …
    }
    …
}
```

2. 事件处理的程序结构

事件处理程序大致有以下三种结构。

（1）事件源所在的类实现监听者接口。前面所述的步骤就是这种方式。

（2）非事件源所在的类实现监听者接口。

```
public class ActionEventDemo extends JFrame{
    JButton b1 = new JButton("b1");
    JButton b2 = new JButton("b2");
    public ActionEventDemo(){
        …
        MyActionListener listener = new MyActionListener(this);
        b1.addActionListener(listener);
        b2.addActionListener(listener);
        …
    }
}
class  MyActionListener implements ActionListener{
```

```
    ActionEventDemo fp;
    public MyActionListener(ActionEventDemo fp){
        this.fp = fp;
    }
    public void actionPerformed(ActionEvent e) {
        if(e.getSource() == fp.b1) {
            JOptionPane.showMessage(null, "单击了按钮 b1");
        } else {
            JOptionPane.showMessage(null, "单击了按钮 b2");
        }
    }
}
```

(3) 匿名类实现监听者接口。如下代码,通过匿名类处理窗口关闭事件。

```
public class WindowEventDemo extends JFrame {
     public WindowEventDemo() {
          this.addWindowListener(new WindowAdapter() {
              @Override
              public void windowClosing(WindowEvent e) {
                  dispose();
                  System.exit(0);
              }
          });
      }
      public static void main(String args[]) {
          new WindowEventDemo();
      }
}
```

3. 事件适配器

事件适配器是实现了监听者接口的类,名称格式为 XXXAdapter。例如,窗口事件的适配器是 WindowAdatper,键盘事件的适配器是 KeyAdapter。具有一个方法以上的监听者接口都有对应的适配器。适配器实现了对应监听者接口的所有方法,通过继承适配器来定义监听者类,就不用实现接口的所有的方法,而只需重写所需要的方法。

5.3.3 常用事件

1. 行为事件

单击按钮、选择菜单项、在文本框中按回车键等都产生行为事件(ActionEvent)。处理这种事件需要实现 ActionListener 接口。前面知识中已接触过。

2. 鼠标事件

鼠标事件(MouseEvent)的事件源一般为容器。当鼠标键按下、释放、单击、进来、离开、移动、拖动时会引发鼠标事件。可以通过实现 java.awt.event 包中的两个接口:MouseListener

接口和 MouseMotionListener 接口处理鼠标事件。

(1) MouseMotionListener 包含的方法如下。

① mouseDragged(MouseEvent e)：鼠标拖动。

② mouseMoved(MouseEvent e)：鼠标移动。

(2) MouseListener 包含的方法如下。

① mousePressed(MouseEvent e)：鼠标按下。

② mouseReleased(MouseEvent e)：鼠标释放。

③ mouseEntered(MouseEvent e)：鼠标进来。

④ mouseExited(MouseEvent e)：鼠标离开。

⑤ mouseClicked(MouseEvent e)：鼠标单击。

(3) MouseEvent 类的一些方法。

① int getX()：返回鼠标事件发生时坐标点的 x 值。

② int getY()：返回鼠标事件发生时坐标点的 y 值。

③ Point getPoint()：返回 Point 对象，包含鼠标事件发生点的坐标点，使用 Point 类的方法 getX()和 getY()可得到坐标点的 x、y 值。

④ int getClickCount()：得到鼠标单击的次数。鼠标单击，返回整数值 1；鼠标双击，返回整数值 2。

3. 键盘事件

键盘事件(KeyEvent)的监听者接口是 KeyListener。该接口包含以下三个方法。

(1) keyPressed(KeyEvent e)：按下某个键时调用此方法。

(2) keyReleased(KeyEvent e)：释放某个键时调用此方法。

(3) keyTyped(KeyEvent e)：按下又释放某个键时调用此方法。

KeyEvent 有以下两个重要的方法。

(1) char getKeyChar()：返回与此事件中的键关联的字符。

(2) int getKeyCode()：返回与此事件中的键关联的键位码 keyCode。

对于 Ascii 键使用第一个方法来判断，对于功能键和光标键等使用第二个方法来判断。第二个方法用在 keyPressed()或 keyReleased()事件中才有效。

4. 窗口事件

当打开、关闭、激活、停用、图标化或取消图标化 Window 对象时，或者焦点转移到 Window 内或移出 Window 时，由 Window 对象生成窗口事件(WindowEvent)。WindowListener 接口的方法如下。

(1) windowActivated(WindowEvent e)：将 Window 设置为活动 Window 时调用。

(2) windowClosed(WindowEvent e)：因对窗口调用 dispose 而将其关闭时调用。

(3) windowClosing(WindowEvent e)：用户试图从窗口的系统菜单中关闭窗口时调用。

(4) windowDeactivated(WindowEvent e)：当 Window 不再是活动 Window 时调用。

(5) windowDeiconified(WindowEvent e)：窗口从最小化状态变为正常状态时调用。

(6) windowIconified(WindowEvent e)：窗口从正常状态变为最小化状态时调用。

(7) windowOpened(WindowEvent e)：窗口首次变为可见时调用。

5.3.4 案例 5-2 实现绘图软件

在案例 5-1 的基础上增加事件处理程序,实现绘图软件的功能。运行效果如图 5-8 所示。

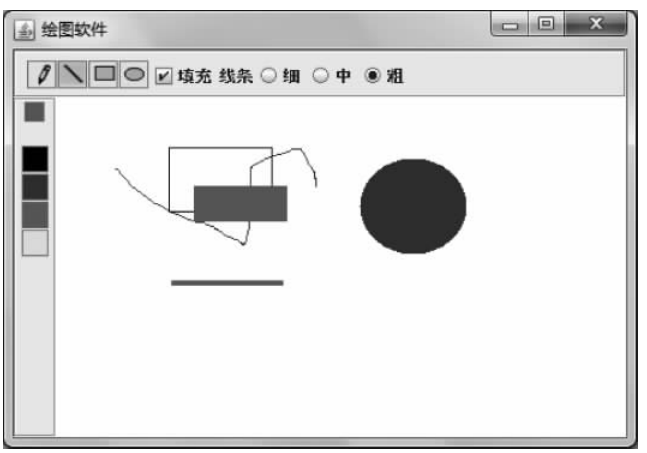

图 5-8 绘图软件运行效果

【技术要点】

(1) 为工具栏的组件添加行为事件处理程序,实现相应的功能。事件处理程序放在主窗口中实现。

(2) 为绘图区域添加鼠标事件处理程序实现绘图。鼠标按下时根据当前绘图类型,建立相应的图形对象,并添加到一个集合中。鼠标拖动时,对于任意画,先把新的鼠标位置作为上次短线的终端,然后用新位置建立新的线对象,并将其添加到一个集合中；对于绘制线、矩形和椭圆,只需用新的鼠标位置设置当前对象的第二个坐标(x2,y2)。事件处理放在绘图板中实现,并采用匿名类的方式。

(3) 为绘图区域添加键盘事件处理程序,判断 Shift 键是否按下,按下时将 shift 变量置 true,否则置 false,以便决定是否绘制正圆或正方形。

【设计步骤】

(1) 在 NetBeans 中新建一个 Java 应用程序项目,项目命名为 Exam5_3_4,将案例 5-1 的图像素材复制到当前项目根目录下。

(2) 在项目中建立包 exam5_3_4,并将案例 5-1 所设计的类复制到该包下。

(3) 打开 DrawWindow,添加实现 ActionListener 事件接口的代码,斜体部分为新加的。

```
public class DrawWindow extends JFrame implements ActionListener {
    …
    public DrawWindow() {
```

```
        …
        b1.addActionListener(this);
        b2.addActionListener(this);
        b3.addActionListener(this);
        b4.addActionListener(this);
        c0.addMouseListener(null);
        c1.addActionListener(this);
        c2.addActionListener(this);
        c3.addActionListener(this);
        c4.addActionListener(this);
        ch.addActionListener(this);
        j1.addActionListener(this);
        j2.addActionListener(this);
        j3.addActionListener(this);
        setTitle("绘图程序");                                    //设置窗口标题
        setSize(500, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setVisible(true);
        drawBoard.requestFocus();
    }
    @Override
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == b1) {
            drawBoard.type = ShapeType.DRAW;
        }else if (e.getSource() == b2) {
            drawBoard.type = ShapeType.LINE;
        }else if (e.getSource() == b3) {
            drawBoard.type = ShapeType.RECT;
        }else if (e.getSource() == b4) {
            drawBoard.type = ShapeType.OVAL;
        }else if (e.getSource() == c1) {
            drawBoard.color = Color.BLACK;
            c0.setBackground(Color.BLACK);
        }else if (e.getSource() == c2) {
            drawBoard.color = Color.BLUE;
            c0.setBackground(Color.BLUE);
        }else if (e.getSource() == c3) {
            drawBoard.color = Color.RED;
            c0.setBackground(Color.RED);
        }else if (e.getSource() == c4) {
            drawBoard.color = Color.YELLOW;
            c0.setBackground(Color.YELLOW);
        }else if (e.getSource() == ch) {
            drawBoard.fillType = ch.isSelected() ? FillType.FILL : FillType.NO_FILL;
        }else if (e.getSource() == j1) {
            drawBoard.thick = 1f;
        }else if (e.getSource() == j2) {
            drawBoard.thick = 2f;
        }else if (e.getSource() == j3) {
            drawBoard.thick = 4f;
        }
```

```
            drawBoard.requestFocus();
        }
        …
    }
```

(4) 打开 DrawBoard 类,添加变量定义,并在构造方法中通过匿名类的方式添加鼠标按下、鼠标拖动、键盘按下和键盘抬起事件处理代码,在 paintComponent 中删除测试代码,增加新的绘图代码。斜体部分为新加的。

```
public class DrawBoard extends JPanel {
    IShape.ShapeType type = IShape.ShapeType.DRAW;        //画图类型
    Color backColor = Color.WHITE;                        //默认背景颜色
    Color color = Color.BLACK;                            //默认颜色
    float thick = 1f;                                     //线条粗细
    IShape.FillType fillType = IShape.FillType.NO_FILL;   //填充方式
    Shape shape;                                          //存放正在画的图像对象
    List<Shape> list = new ArrayList<Shape>();            //存放图形的集合
    boolean shift = false;
    public DrawBoard() {
        this.setOpaque(true);
        this.setBackground(Color.WHITE);
        this.addMouseListener(new MouseAdapter() {
            @Override
            public void mousePressed(MouseEvent e) {
                switch (type) {
                    case DRAW:
                    case LINE:
                        shape = new Line(e.getX(), e.getY(), e.getX(), e.getY(), color,
                    thick);
                        break;
                    case RECT:
                        shape = new Rect(e.getX(), e.getY(), e.getX(), e.getY(), color,
                    thick, fillType);
                        break;
                    case OVAL:
                        shape = new Oval(e.getX(), e.getY(), e.getX(), e.getY(), color,
                    thick, fillType);
                        break;
                }
                list.add(shape);
            }
        });
        addMouseMotionListener(new MouseMotionAdapter() {
            @Override
            public void mouseDragged(MouseEvent e) {
                switch (type) {
                    case DRAW:
                        shape.setX2(e.getX());
                        shape.setY2(e.getY());
                        shape = new Line(e.getX(), e.getY(), e.getX(), e.getY(), color,
                    thick);
```

```
                    list.add(shape);
                    break;
                case LINE:
                case RECT:
                case OVAL:
                    shape.setX2(e.getX());
                    shape.setY2(e.getY());
                    if (shift) {
                        shape.setY1(shape.getY() + shape.getHeight() - shape.
                    getWidth());
                    }
                    break;
                }
                repaint();
            }
        });
        addKeyListener(new KeyAdapter() {
            @Override
            public void keyReleased(KeyEvent e) {
                if (e.getKeyCode() == KeyEvent.VK_SHIFT) {
                    shift = false;
                }
            }
            @Override
            public void keyPressed(KeyEvent e) {
                if (e.getKeyCode() == KeyEvent.VK_SHIFT) {
                    shift = true;
                }
            }
        });
    }
    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        for (Shape s : list) {
            s.draw((Graphics2D) g);
        }
    }
}
```

（5）保存并运行程序。

5.4　菜单和工具栏

菜单和工具栏可为用户选择功能提供直观快捷的方式。

5.4.1 菜单

1. 菜单的组件

1) 菜单条

菜单条(JMenuBar)是菜单的容器。菜单条的构造方法如下。

JMenuBar(): 建立一个新的 JMenuBar。

2) 菜单

菜单(JMenu)是用来存放和整合 JMenuItem 的组件。JMenu 可以是单层次结构,也可以是分层结构。菜单的构造方法有以下几个。

(1) JMenu(): 建立一个新的 JMenu。

(2) JMenu(Action a): 建立一个支持 Action 的新的 JMenu。

(3) JMenu(String s): 以指定的字符串名称建立一个新的 JMenu。

(4) JMenu(String,Boolean b): 以指定的字符串名称建立一个新的 JMenu,并决定这个菜单是否具有下拉式的属性。

3) 菜单项

菜单项(JMenuItem)继承 AbstractButton 类,因此 JMenuItem 具有许多 AbstractButton 的特性,也可以说 JMenuItem 是一种特殊的 JButton。菜单项的构造方法有以下几个。

(1) JMenuItem(): 建立一个新的 JMenuItem。

(2) JMenuItem(Action a): 建立一个支持 Action 的新的 JMenuItem。

(3) JMenuItem(Icon icon): 建立一个有图标的 JMenuItem。

(4) JMenuItem(String text): 建立一个有文字的 JMenuItem。

(5) JMenuItem(String text,Icon icon): 建立一个有图标和文字的 JMenuItem。

(6) JMenuItem(String text,int mnemonic): 建立一个有文字和快捷键的 JMenuItem。

2. 如何建立菜单

建立菜单首先要通过 JMenuBar 建立一个菜单条,然后使用 JMenu 建立菜单,再通过 JMenuItem 为每个菜单建立菜单项。例如,如下代码创建的菜单如图 5-9 所示。

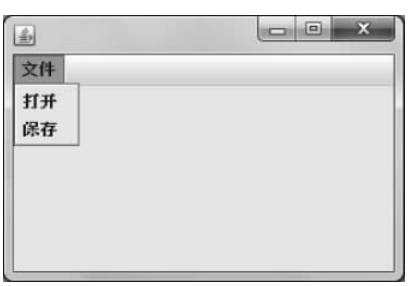

图 5-9 带菜单的窗口

```
package exam5_4_1;
import javax.swing.*;
public class Exam5_4_1 extends JFrame {
    public Exam5_4_1() {
        JMenuBar menubar = new JMenuBar();                    //建立菜单条
        JMenu fileMenu = new JMenu("文件");                   //建立菜单
        JMenuItem item1 = new JMenuItem("打开");              //建立菜单项
        JMenuItem item2 = new JMenuItem("保存");
        fileMenu.add(item1);                                  //将菜单项添加到菜单
```

```
        fileMenu.add(item2);
        menubar.add(fileMenu);                          //将菜单添加到菜单条
        setJMenuBar(menubar);                           //将菜单设置到窗口
        setSize(300,200);
        setVisible(true);
    }
    public static void main(String args[]) {
        new  Exam5_4_1();
    }
}
```

3. 建立菜单的高级技巧

1）建立图标菜单

```
JMenuItem item1 = new  JMenuItem("打开", new ImageIcon("m11.gif"));
```

2）设置热键和快捷键

```
JMenu fileMenu = new JMenu("文件(F)");                  //建立菜单
fileMenu.setMnemonic('F');                              //为菜单设置热键
JMenuItem item1 = new JMenuItem("打开(O)");             //建立菜单项
item1.setMnemonic('O');                                 //为菜单项设置热键
item1.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_O,InputEvent.CTRL_MASK));
                                                        //建立快捷键 Ctrl + U
```

3）弹出式菜单

JPopupMenu 是一种特别形式的 JMenu，其性质与 JMenu 几乎完全相同，但它并不固定在窗口的某个位置，而是由程序决定其出现的位置。例如：

```
public void mouseReleased(MouseEvent e){               //鼠标抬起事件
    if(e.isPopupTrigger()) {
        popup.show(e.getComponent(),e.getX(),e.getY()); //显示弹出菜单
    }
}
```

5.4.2　工具栏

工具栏是一个显示一组动作、命令或功能的组件。一般来说，工具栏中的组件都是带图标的按钮，可以使用户更加方便地选择所需的功能。

1. 建立工具栏

JToolBar 构造方法有以下几个。

(1) JToolBar()：建立一个新的 JToolBar，位置为默认的水平方向。

(2) JToolBar(int orientation)：建立一个指定位置的 JToolBar。

(3) JToolBar(String name)：建立一个指定名称的 JToolBar。

(4) JToolBar(String name,int orientation)：建立一个指定名称和位置的 JToolBar。

例如,下面的代码建立一个工具栏,该工具栏上包含一个按钮。

```
JToolBar toolBar = new JToolBar();
JButton b1 = new JButton();
b1.setToolTipText("打开文件");
b1.addActionListener(this);
b1.setIcon(new ImageIcon("b1.gif"));
toolBar.add(b1);
add(toolBar, BorderLayout.PAGE_START);                    //将工具栏设置到窗体
```

2. 建立工具栏的高级技巧

1) 设置浮动

void setFloatable(boolean b):设置工具栏是否可以浮动。

void setRollover(boolean rollover):设置工具栏是否可转滚。

2) 设置方向

void setOrientation(int o):设置工具栏方向。o 可取如下常量:

(1) JToolBar. HORIZONTAL

(2) JToolBar. VERTICAL

3) 添加分隔条

void addSeparator():添加分隔条。

5.4.3 案例 5-3 设计学生管理系统主界面

设计学生管理系统主界面,要求运行时最大化,并有背景图、菜单及工具栏,菜单结构如图 5-10 所示。运行效果如图 5-11 所示。

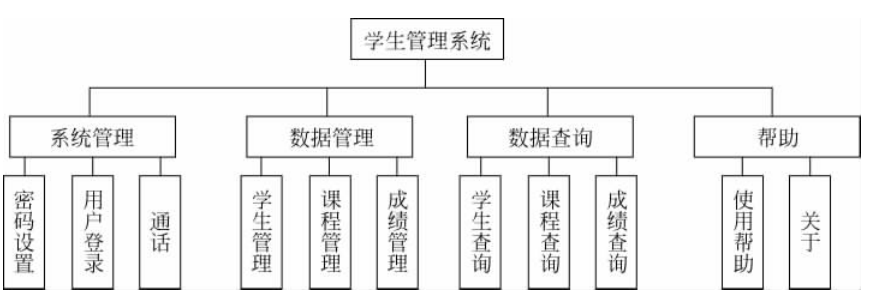

图 5-10 学生管理系统菜单结构

【技术要点】

(1) 单独设计一个方法 createMenu()用于建立菜单。在该方法中通过 JMenuBar 建立一个菜单条,然后使用 JMenu 建立菜单,再使用 JMenuItem 为每个菜单建立菜单项。

(2) 单独设计一个方法 createToolBar()用于创建工具栏。在该方法中创建 JToolBar 对象,使用 add()方法将带图标的按钮添加到工具栏中。

(3) 新建一个面板 PicPanel,继承 JPanel,利用该面板显示背景图。在其构造方法中,

通过 Toolkit.getDefaultToolkit().getImage()方法加载图像，在 paintComponent()中显示图像。

图 5-11　带有菜单工具栏的主界面

【设计步骤】

(1) 为项目准备图像素材，如下所示。

(2) 在 NetBeans 中新建一个 Java 应用程序项目，项目命名为 Exam5_4_3，将图像素材复制到项目文件夹根目录下。

(3) 在项目中建立包 exam5_4_3，在该包下新添加一个类，命名为 PicPanel，该类继承 JPanel，用于显示图像，代码如下所示。

```
public class PicPanel extends JPanel {
    Image image;
    public PicPanel(){
    }
    public PicPanel(String filename) {
        image = Toolkit.getDefaultToolkit().getImage(filename);
}
public void show(Image image) {
     this.image = image;
     repaint();
}
    public void show(String filename){
        image = Toolkit.getDefaultToolkit().getImage(filename);
        repaint();
    }
    @Override
    protected void paintComponent(Graphics g) {
```

```
        super.paintComponent(g);
        g.drawImage(image, 0, 0, this.getWidth(), this.getHeight(), this);
    }
}
```

(4) 在 exam5_4_3 包下新添加一个类,命名为 MainWindow,代码如下所示。

```
public class MainWindow extends JFrame implements ActionListener {
    JMenuBar menuBar;                                    //菜单条
    JMenu menu1, menu2, menu3, menu4;                    //菜单
    JMenuItem m11, m12, m13;                             //菜单项
    JMenuItem m21, m22, m23;
    JMenuItem m31, m32, m33;
    JMenuItem m41, m42;
    JToolBar toolBar;                                    //定义工具栏
    JButton b1, b2, b3;
    public MainWindow() {
    add(new PicPanel("back.jpg"),BorderLayout.CENTER);
        createMenu();
        createToolBar();
        setTitle("学生管理系统");                          //设置窗口标题
        setIconImage((new ImageIcon("icon.gif")).getImage());   //设置图标
        setSize(600, 400);                               //设置窗口大小
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  //设置关闭行为
        setVisible(true);                                //使窗口可见
        this.setExtendedState(JFrame.MAXIMIZED_BOTH);
    }
    private void createMenu() {
        menuBar = new JMenuBar();                        //建菜单条
        menu1 = new JMenu("系统管理(S)");
        menu1.setMnemonic('S');                          //设置热键
        m11 = new JMenuItem("密码设置");
        m12 = new JMenuItem("用户登录");
        m13 = new JMenuItem("退出(X)");
        m13.setAccelerator(KeyStroke.getKeyStroke('X', 2));
        menu1.add(m11);
        menu1.add(m12);
        menu1.addSeparator();                            //分割线的意思
        menu1.add(m13);
        menu2 = new JMenu("数据管理(D)");
        menu2.setMnemonic('D');                          //设置热键
        m21 = new JMenuItem("学生管理");
        m22 = new JMenuItem("课程管理");
        m23 = new JMenuItem("成绩管理");
        menu2.add(m21);
        menu2.add(m22);
        menu2.add(m23);
        menu3 = new JMenu("数据查询(F)");
        menu3.setMnemonic('F');                          //设置热键
        m31 = new JMenuItem("学生查询");
        m32 = new JMenuItem("课程查询");
```

```
        m33 = new JMenuItem("成绩查询");
        menu3.add(m31);
        menu3.add(m32);
        menu3.add(m33);
        menu4 = new JMenu("帮助(H)");
        menu4.setMnemonic('H');                                    //设置热键
        m41 = new JMenuItem("使用帮助");
        m42 = new JMenuItem("关于");
        menu4.add(m41);
        menu4.addSeparator();
        menu4.add(m42);
        m11.addActionListener(this);
        m12.addActionListener(this);
        m13.addActionListener(this);
        m21.addActionListener(this);
        m22.addActionListener(this);
        m23.addActionListener(this);
        m31.addActionListener(this);
        m32.addActionListener(this);
        m41.addActionListener(this);
        m42.addActionListener(this);
        menuBar.add(menu1);
        menuBar.add(menu2);
        menuBar.add(menu3);
        menuBar.add(menu4);
        this.setJMenuBar(menuBar);
    }
    private void createToolBar() {
        toolBar = new JToolBar();
        b1 = new JButton();
        b1.setToolTipText("学生管理");
        b1.addActionListener(this);
        b1.setIcon(new ImageIcon("b1.gif"));
        b2 = new JButton();
        b2.setToolTipText("课程管理");
        b2.addActionListener(this);
        b2.setIcon(new ImageIcon("b2.gif"));
        b3 = new JButton();
        b3.setToolTipText("成绩管理");
        b3.addActionListener(this);
        b3.setIcon(new ImageIcon("b3.gif"));
        b1.addActionListener(this);
        b2.addActionListener(this);
        b3.addActionListener(this);
        toolBar.add(b1);
        toolBar.add(b2);
        toolBar.add(b3);
        toolBar.setRollover(true);                                 //设置转滚效果,鼠标移上时出现边框
        add(toolBar, BorderLayout.PAGE_START);
    }
    public void actionPerformed(ActionEvent e) {
```

```
    }
    public static void main(String args[]) {
        new MainWindow();                                    //建立窗口
    }
}
```

(5) 保存并运行程序。

5.5 对话框与其他常用组件

对话框是一种类似于窗口的容器。Java 提供了丰富的组件,除了前面介绍的组件外,常用的还有文本框、密码域、文本区和组合框。

5.5.1 对话框

对话框类似于窗口,与一般窗口的区别在于它可依赖于其他窗口:当它所依赖的窗口消失或最小化时,对话框也消失;窗口还原时,对话框又会自动恢复。此外,对话框还具有模态特性。

1. JDialog 对话框

JDialog 与 JFrame 类似,是有边框、有标题、可独立存在的顶级容器。对话框分为无模态对话框和模态对话框。模态对话框只让程序响应对话框内部的事件,对于对话框以外的事件程序不响应;而无模态对话框可以让程序响应对话框以外的事件。

JDialog 的主要构造方法如下。

(1) JDialog()

(2) JDialog(Frame owner)

(3) JDialog(Frame owner, boolean modal)

(4) JDialog(Frame owner, String title)

(5) JDialog(Frame owner, String title, boolean modal)

其中,参数 owner 指明对话框所依赖的窗口,title 指明对话框的标题,modal 指明对话框是否为模态。

2. JOptionPane 对话框

简单的对话框可以使用 JOptionPane 类的静态方法建立。

(1) showConfirmDialog():确认对话框,询问问题,带有 Yes,No 和 Cancel 按钮。

① 参数 1:包含该对话框的容器,该信息可以用来决定对话框窗口应该显示在屏幕的什么位置,若该参数是 null,或该参数不是一个 Frame 对象,则对话框会被显示在屏幕中央。

② 参数 2:可以是一个字符串、一个组件或一个图标,它被显示在对话框里。

③ 参数 3:一个字符串,指明对话框标题。

④ 参数 4：整数，指明哪个选项按钮被显示出来。可以有两个值：YES_NO_OPTION，YES_NO_CANCEL_OPTION。

⑤ 参数 5：整数，表示对话框的类型。可取的值有：ERROR _ MESSAGE，INFORMATION_MESSAGE，PLAIN_MESSAGE，QUESTION_MESSAGE，WARNING _MESSAGE。

⑥ 返回值：为一整数值，依用户单击什么按钮而定：YES_OPTION，NO_OPTION，CANCEL_OPTION，OK_OPTION，CLOSED_OPTION（当用户什么都不选直接关掉对话框时）。

例如，如下语句显示的对话框如图 5-12 所示。

```
int n = JOptionPane.showConfirmDialog(null,                    // 所属窗体
        "你喜欢 Java 吗?",                                      // 输出信息
        "问题对话框",                                           // 标题
        JOptionPane.YES_NO_OPTION );                           // 按钮类型
```

(2) showInputDialog()：输入对话框，用来接收文本输入并用字符串存储。

参数 1，2，3，4 相当于 Confirm 对话框的参数 1，2，3，5。

例如，如下语句显示的对话框如图 5-13 所示。

```
ImageIcon icon = new ImageIcon("icon.gif");
Object[] possibilities = { "C++", "Java", "VB" };
String s = (String) JOptionPane.showInputDialog(null,          // 所属窗体
     "请选择项目：\n 喜欢哪种语言?",                            // 输出信息
     "客户选择",                                               // 标题
     JOptionPane.PLAIN_MESSAGE,                                // 对话框类型
     icon,                                                     // 显示图标
     possibilities,                                            // 选项内容
     "Java" );                                                 // 默认选项
```

图 5-12　确认对话框

图 5-13　输入对话框

(3) showMessageDialog()：消息对话框，用于显示消息。

参数 1，2，3，4 与 Input 对话框一致。

例如，如下语句显示的对话框如图 5-14 所示。

```
JOptionPane.showMessageDialog(null, "Java 世界丰富多彩!");
```

(4) showOptionDialog()：包含上面所有的三种对话框类型。

① 前 5 个参数与 Confirm 对话框一样。

② 参数 6：要显示的一个 Icon 对象。

③ 参数 7：一个对象数组，它存放了在对话框中用于做出选择的组件和其他对象。

④ 参数 8：代表默认选项的对象。

例如，如下语句显示的对话框如图 5-15 所示。

```
Object options[] = { "是的", "不喜欢" };
int n = JOptionPane.showOptionDialog(null,
        "你喜欢 Java 吗?",
        "问题对话框",
        JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE,  //图标类型
        null,                                                    // 图标
        options, // 选项内容
        options[0] );                                            // 默认选项
```

图 5-14　消息对话框

图 5-15　选项对话框

5.5.2　其他组件介绍

1. 文本框

文本框(JTextField)允许输入或编辑单行文本。主要的构造方法有以下几个。

(1) JTextField()

(2) JTextField(int columns)

(3) JTextField(String text)

(4) JTextField(String text, int columns)

其中，columns 为初始字段长度，text 为初始文本。

主要的方法如下。

(1) void setFont(Font f)：设置字体。

(2) String getText()：获取文本。

(3) void setText()：设置文本。

(4) void setEditable(boolean b)：设置是否可编辑。

(5) void requestFocus()：设置焦点。

(6) void setHorizontalAlignment(int alignment)：设置文本对齐方式。可用的对齐方式有：JTextField. LEFT、JTextField. CENTER、JTextField. RIGHT。

2. 密码域

JPasswordField 也是一个单行的输入组件，与 JTextField 基本类似，不同的是

JPasswordField 增加了屏蔽输入的功能。密码域取值的方法如下。

```
String password = new String(txtPassword.getPassword());
```

3. 文本域

用于创建多行文本域。主要的构造方法如下。

(1) JTextArea()

(2) JTextArea(int rows, int cols)

(3) JTextArea(String text)

(4) JTextArea(String text, int rows, int cols)

其中,rows、cols 为行数和列数,text 为初始文本内容。

文本域常用的方法有以下几个。

(1) void setText(Stringtext):设置文本。

(2) void insert(String text, int pos):插入文本。

(3) void append(String text):添加文本。

(4) void replace(String text, int start, int end):替换文本。

文本区不自带滚动条,要加滚动条,需要使用滚动面板 JScrollPane,例如:

```
JTextArea tt = new JTextArea(10,50);
JScrollPane jsp = new JScrollPane(tt);
container.add(jsp);
```

4. 组合框

组合框(JComboBox)是文本编辑区和列表的组合。可以在文本编辑区中输入选项,也可以单击下拉按钮从显示的列表中进行选择。默认组合框是不能编辑的,需要通过 setEditable (true)设为可编辑。

组合框的构造方法如下。

(1) JComboBox():建立一个无选项的 JComboBox 组件。

(2) JComboBox(ComboBoxModel aModel):用数据模型建立一个 JComboBox 组件。

(3) JComboBox(Object[] items):利用数组对象建立一个 JComboBox 组件。

(4) JComboBox(Vector items):利用向量对象建立一个 JComboBox 组件。

例如,如下代码使用数组创建组合框。

```
String[] s = {"西瓜","苹果","草莓","香蕉","葡萄"};
JComboBox combo = new JComboBox(s);
```

常用的方法有以下几个。

(1) void addItem(Object object):通过字符串类或其他类加入选项。

(2) int getItemCount():获取条目的总数。

(3) void removeItem(Object object):通过字符串类或其他类删除选项。

(4) void removeItemAt(int index):通过索引删除选项。

(5) void insertItemAt(Object object,int index):在特定的位置插入元素。

(6) int getSelectedIndex()：获得所选项的索引值(索引值从 0 开始)。

(7) ObjectgetSelectedItem()：获得所选项的内容。

5.5.3 案例 5-4 用户登录与添加学生界面设计

设计学生管理系统的用户登录窗口和添加学生窗口。程序运行后,首先打开主界窗口,在主界面的上面弹出登录窗口,如果用户不登录,将无法切换到后面的主窗口;若用户登录正确,登录窗口将关闭,主窗口可以使用。运行界面如图 5-16 所示。添加学生数据窗口通过菜单或工具栏打开。运行界面如图 5-17 所示。

图 5-16 登录窗口

【技术要点】

(1) 登录窗口和添加学生数据窗口均继承 JDialog,并设置成模态。

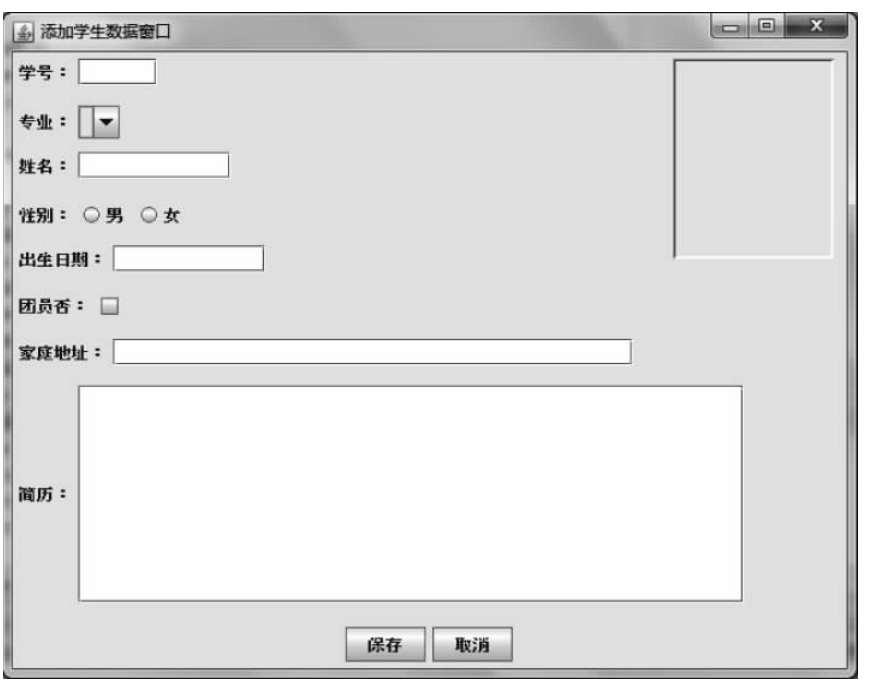

图 5-17 添加学生数据窗口

(2) 登录窗口整体布局为网格布局,3 行 1 列,每一行放一个面板。使用 JTextField 输入用户名,使用 JPasswordField 输入密码。在按钮的行为事件中对用户名和密码进行检验。

(3) 添加学生数据窗口的整体布局也是边界布局,分为上、中、下三个区。上又分为上左和上右,上左是网格布局,7 行 1 列,每一行放一个面板,均为左对齐流式布局,分别呈现的是学号、专业、姓名、性别、出生日期和地址；上右呈现照片。中区呈现简历,下区呈现【保存】和【取消】按钮。最下面一个面板放置两个按钮,采用默认布局。

（4）单击照片，使用文件对话框 JFileChooser 选择文件。

【设计步骤】

（1）在 NetBeans 下新建一个 Java 应用程序项目，项目命名为 Exam5_5_3。将案例 5-3 设计图像素材复制到当前项目文件夹的根目录下。

（2）在项目中新建一个包 exam5_5_3，将案例 5-3 设计的类复制到该包下。

（3）在 exam5_5_3 包下新添加一个类，命名为 LoginWindow，代码如下所示。

```
public class LoginWindow extends JDialog implements ActionListener {
    JTextField txtUsername = new JTextField(10);              //用户名文本框
    JPasswordField txtPassword = new JPasswordField(10);      //密码域
    JButton btnOK = new JButton("确定");
    JButton btnCancel = new JButton("取消");
    public LoginWindow() {
        Container contentPane = this.getContentPane();        //取出内容面板
        contentPane.setLayout(new GridLayout(3, 1, 5, 5));    //设置布局为 5 行 1 列
        JPanel p1 = new JPanel();
        JPanel p2 = new JPanel();
        JPanel p3 = new JPanel();
        p1.add(new JLabel("用户名："));p1.add(txtUsername);
        p2.add(new JLabel("密　码："));p2.add(txtPassword);
        p3.add(btnOK); p3.add(btnCancel);
        contentPane.add(p1);                                  //将面板添加到内容面板
        contentPane.add(p2);
        contentPane.add(p3);
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);    //设置自动关闭窗口
        btnOK.addActionListener(this);                        //注册事件接听者
        btnCancel.addActionListener(this);
        txtUsername.addActionListener(this);
        txtPassword.addActionListener(this);
        setSize(250, 150);                                    //设置窗口的大小
        Dimension size = Toolkit.getDefaultToolkit().getScreenSize();
        setLocation((size.width - 300) / 2, (size.height - 220) / 2);
        setTitle("登录窗口");
        this.addWindowListener(new WindowAdapter() {
            @Override
            public void windowClosing(WindowEvent e) {
                System.exit(0);
            }
        });
        setModal(true);                                       //设置模态
        setResizable(false);                                  //不让用户改变窗口的大小
        setVisible(true);
    }
    public void actionPerformed(ActionEvent e) {              //事件处理方法
        if (e.getSource() == btnOK || e.getSource() == txtPassword) {
            if (txtUsername.getText().trim().equals("yang")
                    && new String(txtPassword.getPassword()).equals("1234")) {
                dispose();                                    //关闭登录窗口
            } else {
```

```
                JOptionPane.showMessageDialog(null, "用户名或密码错误!");
                txtUsername.requestFocus();                    //设置焦点
            }
        } else if (e.getSource() == btnCancel) {               //单击【取消】按钮
            dispose();                                         //关闭窗口
            System.exit(0);                                    //退出程序
        } else if (e.getSource() == txtUsername) {             //在用户名文本框中按回车键
            txtPassword.requestFocus();                        //设置焦点
        }
    }
    public static void main(String args[]) {
        new LoginWindow();
    }
}
```

(4) 打开 MainWindow,在构造方法的最后加入如下代码。

```
new LoginWindow();
```

(5) 运行 LoginWindow,测试登录窗口。

(6) 在 exam5_5_3 包下新添加一个类,命名为 AddStudent,代码如下所示。

```
public class AddStudent extends JFrame implements ActionListener {
    String title[] = {"学号", "专业", "姓名", "性别", "出生日期", "团员否", "家庭地址"};
    JTextField txtNo = new JTextField(5);
    JComboBox comMajor = new JComboBox();
    JTextField txtName = new JTextField(10);
    JRadioButton radSexM = new JRadioButton("男");
    JRadioButton radSexF = new JRadioButton("女");
    JTextField txtBirthDate = new JTextField(10);
    JCheckBox chIsMember = new JCheckBox("");
    JTextField txtAddress = new JTextField(35);
    JTextArea txtResume = new JTextArea(10, 45);
    PicPanel panelPic = new PicPanel();
    JButton btnOK = new JButton("保存");
    JButton btnCancel = new JButton("取消");
    String filename;
    public AddStudent() {
        Container con = getContentPane();
        ButtonGroup group = new ButtonGroup();
        group.add(radSexM);
        group.add(radSexF);
        panelPic.setBorder(BorderFactory.createLoweredBevelBorder());
        panelPic.setPreferredSize(new Dimension(120, 150));
        JPanel top = new JPanel(new BorderLayout());
        JPanel topLeft = new JPanel(new GridLayout(7, 1));
        JPanel topRight = new JPanel();
        topRight.setPreferredSize(new Dimension(140, 1));
        topRight.add(panelPic);
        JPanel p[] = new JPanel[7];
        for (int i = 0; i < 7; i++) {
```

```
            p[i] = new JPanel(new FlowLayout(FlowLayout.LEFT));
            p[i].add(new JLabel(title[i] + ": "));
            topLeft.add(p[i]);
        }
        p[0].add(txtNo);
        p[1].add(comMajor);
        p[2].add(txtName);
        p[3].add(radSexM);
        p[3].add(radSexF);
        p[4].add(txtBirthDate);
        p[5].add(chIsMember);
        p[6].add(txtAddress);
        top.add(topLeft, "Center");
        top.add(topRight, "East");
        JPanel center = new JPanel(new FlowLayout(FlowLayout.LEFT));
        center.add(new JLabel("简历: "), "West");
        center.add(new JScrollPane(txtResume), "Center");
        JPanel bottom = new JPanel();
        bottom.add(btnOK);
        bottom.add(btnCancel);
        con.add(top, "North");
        con.add(center, "Center");
        con.add(bottom, "South");
        panelPic.addMouseListener(new MouseAdapter() {
            @Override
            public void mousePressed(MouseEvent e) {
                JFileChooser chooser = new JFileChooser();
                chooser.addChoosableFileFilter(new FileNameExtensionFilter("JPEG 图片文
            件", "jpg"));
                int returnVal = chooser.showOpenDialog(null);
                if (returnVal == JFileChooser.APPROVE_OPTION) {
                    File file = chooser.getSelectedFile();
                    filename = file.getAbsolutePath();
                    panelPic.show(filename);
                }
            }
        });
        btnOK.addActionListener(this);
        btnCancel.addActionListener(this);
        setTitle("添加学生数据窗口");
        setSize(640, 500);
        setVisible(true);
    }
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == btnOK) {
            ;
        } else {
            dispose();
        }
    }
    public static void main(String args[]) {
```

```
            new AddStudent();
        }
    }
```

(7) 保存项目,单独运行 AddStudent 进行测试。

小结

GUI 是指以图形的方式实现用户与计算机之间交互操作的应用程序界面。早期的 GUI 程序设计主要使用 AWT。Java 1.2 开始引入称为 Swing 的新的 GUI 组件库,其功能比 AWT 组件的功能更加强大。

Java 语言的类库中提供了丰富的绘图功能,其中大部分对图形、文本、图像的操作方法都定义在 Graphics 中。Java2D API 增强了 AWT 的图形、文本和图像功能,使用 Java2D API 可以开发出更为强大的图形、图像程序。

Swing 组件从功能上可分为:容器类和组件类。窗口和对话框是最主要的容器。Java 创建窗口使用 JFrame(框架)。对话框(JDialog)是一种类似于窗口的容器。与一般窗口的区别在于它可依赖于其他窗口,并具有模态特性。简单的对话框可使用 JOptionPane 类的静态方法建立。

容器可以通过选择不同的布局管理器来决定布局。布局管理器主要包括:FlowLayout,BorderLayout、GridLayout、CardLayout。

在 GUI 应用程序中,用户操作(如单击鼠标或按某个键)、程序代码或系统内部都可以产生“事件”。在事件驱动机制中,应用程序代码可以响应事件来执行一系列的操作,称为“事件处理”。Java 使用授权处理模型来处理事件。常用的事件有行为事件(ActionEvent),鼠标事件(MouseEvent),键盘事件(KeyEvent),窗口事件(WindowEvent)等。

菜单和工具栏为用户选择功能提供了直观快捷的方式。窗口上的菜单使用 JMenuBar 创建,工具栏使用 ToolBar 控件创建。

习题

一、思考题

5-1 Swing 与 AWT 有什么联系和区别?

5-2 绘图程序怎样获得 Graphics?

5-3 JFrame 和 JDialog 有什么区别?

5-4 Java 有哪些常用布局?各有什么特点?JPanel 和 JFrame 的默认布局是什么?

5-5 简述在 Java 的委托事件处理模型中,事件源、事件对象、事件监听者、监听者接口的概念及它们在处理事件时的相互关系。

5-6 窗口有哪些常用的方法和事件?什么是事件适配器?窗口事件的事件适配器是什么?

5-7　键盘事件对象的两个方法 getKeyChar()和 getKeyCode()有什么区别?

5-8　什么是菜单? 怎样为窗口创建菜单? JMenu 中能否添加 JMenu? 在文本框中,单击鼠标右键欲弹出一个菜单,该怎样设计?

二、编程题

5-9　设计一个 Windows 应用程序,能够求二次方程的根。要求利用窗口界面输入二次方程的系数,用 JOptionPane 的静态方法打开对话框,并显示方程的根。

5-10　设计产品管理系统主界面,主界面包含菜单、工具栏和背景图。

5-11　设计产品管理系统的登录界面和产品添加界面。

实验

题目: GUI 程序设计

一、实验目的

(1) 掌握图形界面程序的设计步骤;
(2) 掌握 Java 程序菜单、工具栏等设计方法;
(3) 掌握常用控件的使用;
(4) 进一步理解面向对象的程序设计方法;
(5) 培养学生应用程序设计能力。

二、实验要求

设计图书借阅管理系统界面,要求:

(1) 设计图书借阅管理系统的主界面,完成菜单和工具栏的设计。菜单参考图 5-18。

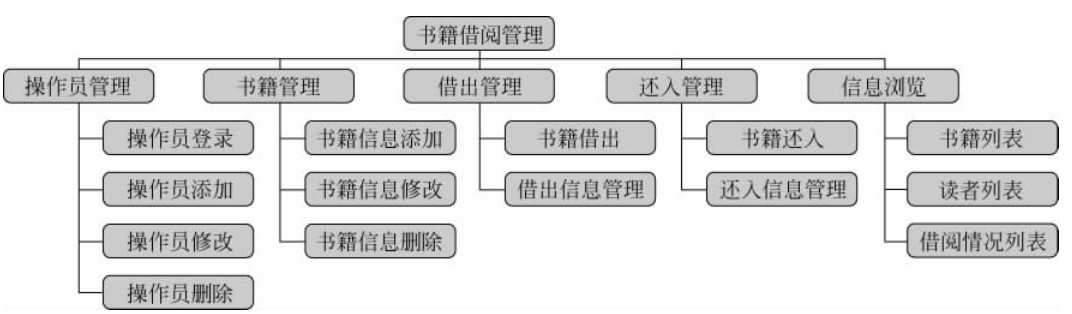

图 5-18　实验程序菜单

(2) 完成书籍信息添加界面的设计,包括书名、类型、作者、单价、出版社、数量、简介等录入项。

(3) 实现主窗口对子窗口的调用。