第5章 图形用户界面设计

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。本章主要介绍如何设计友好的图形用户界面应用程序。

5.1 图形用户界面概述

1. awt 和 swing 图形用户界面包

图形用户界面的构件一般包括菜单、输入输出组件、按钮、画板、窗口、对话框等,这些组件构成 Java 的抽象窗口工具包(Abstract Window Toolkit,AWT)。

Java 语言提供了两种类型的图形用户界面包。在 J2SE 早期版本中,主要是 awt 图形用户界面包。它是一个强大的工具集,但隐藏着一个严重的问题:awt 组件的设计是把与显示相关的工作和处理组件事件的工作都交给本地对等组件完成,因此用 awt 包编写的程序会在不同的操作平台上显示不同的效果。Java 是遵循"一次编译,到处运行"理念的,而 awt 包中图形组件的绘制方法却不能做到这一点。

为了解决这个问题,Java 在 awt 抽象窗口工具包的基础上,开发出了 javax.swing 图形用户界面包。javax.swing 包内的组件称为 swing 组件。swing 组件是用 Java 实现的轻量级(light-weight)组件,没有本地代码,不依赖操作系统的支持,这是它与 awt 组件的最大区别。由于 awt 组件通过与具体平台相关的对等类(peer)实现,因此 swing 比 awt 组件具有更强的实用性。swing 在不同的平台上表现一致,并且有能力提供本地窗口系统不支持的其他特性。swing 组件没有本地代码,不依赖操作平台,而且有更好的性能。swing 组件的功能也有很大的增强,如增加了剪贴板、打印支持功能等。swing 组件除了保持 awt 组件原有组件之外,还增加了一个丰富的高层组件集合,如表格(JTable)、树(JTree)。

把 awt 图形用户界面包称为 awt 组件,也称为重量组件;把 swing 图形用户界面包称为 swing 组件,又称为轻量组件。

本章主要介绍 swing 组件的图形用户界面设计方法,但由于 swing 组件是 awt 的子类,不可避免地会涉及 awt 类包的有关内容。

2. swing 组件的层次结构

swing 组件的层次结构是为了编制基于事件的用户图形界面应用程序而在 awt 组件的基础上建立的。swing 组件包括标准的图形用户界面的要素,如窗口、对话框、构件、事件处理、版面设计管理及接口、例外处理等。

swing 组件的内容非常丰富，本章主要涉及 JApplet、JFrame 和 JConponent 三大类型。其中，JConponent 是其他常用轻量级组件的父类。swing 组件的层次结构如图 5-1 所示。

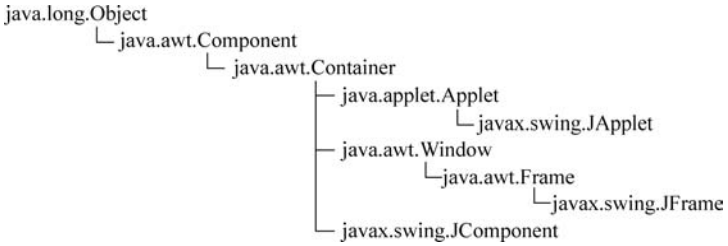

图 5-1　swing 组件的层次结构

5.2　窗体容器和组件

5.2.1　窗体容器 JFrame 类

JFrame 是带有标题、边框的顶层窗体。窗体是一个容器，在其内部可以添加其他组件。JFrame 包含标题栏、菜单、可放置其他组件的窗体内部区域和自带的按钮。JFrame 的外观如图 5-2 所示。

图 5-2　窗体的结构

1. 创建窗体

创建窗体有两种方法：一种方法是创建 JFrame 类的子类，并重写其构造方法；另一种方法是创建 JFrame 类的一个对象：

```
JFrame win = new JFrame("最简单窗体");
```

2. JFrame 类的方法

JFrame 类的常用方法如表 5-1 所示。

表 5-1　JFrame 类的常用方法

方　法　名	功　　能
JFrame();	创建无标题的窗体
JFrame(String s);	创建标题为 s 的窗体
setMenuBar(MenuBar mb);	设置菜单
dispose();	关闭窗体，释放占用资源
setVisible(bolean b);	设置窗体的可见性
setSize(int width,int height);	设置窗体的大小
Validate();	使窗体中的组件能显示出来

续表

方 法 名	功 能
setTitle(String title);	设置标题内容
getTitle();	获取标题内容
setDefaultCloseOperation(int operation)	设置窗体的关闭按钮可用,其中常数 operation 为 EXIT_ON_CLOSE

下面分别举例说明通过构造 JFrame 对象设计窗体及利用 JFrame 子类设计窗体的方法。

【例 5-1】 通过构造 JFrame 对象创建最简单窗体。

```
1  import javax.swing.*;
2  class Example5_1
3  {
4    public static void main(String[] args)
5    {
6      JFrame win = new JFrame("最简单窗体");                    //实例化 JFrame 窗体对象
7      win.setSize(300,200);                                    //设置窗体大小
8      win.setVisible(true);                                    //设置窗体可见
9      win.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);      //设置关闭窗体
10    }
11 }
```

图 5-3 最简单窗体

程序的运行结果如图 5-3 所示。

【程序说明】

程序的第 9 行 setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE)为设置窗体关闭按钮的关闭动作,如果没有该语句,当用户试图关闭窗口时,窗体只是隐藏,并没有真正从内存中退出。

【例 5-2】 通过构造 JFrame 子类创建最简单窗体。

```
1  import javax.swing.*;
2  class  Example5_2 extends JFrame
3  {
4    Example5_2()                                               //构造方法
5    {
6      setSize(300,200);                                        //设置窗体大小
7      setVisible(true);                                        //设置窗体可见
8      setTitle("最简单窗体");                                   //设置窗体标题栏内容
9      setDefaultCloseOperation(EXIT_ON_CLOSE);                 //设置关闭窗体,退出程序
10   }
11   public static void main(String[] args)
12   {
13     Example5_2  win = new Example5_2();
14   }
15 }
```

程序的运行结果与例 5-1 相同,见图 5-3。

5.2.2 按钮和事件处理

1. 按钮 JButton 类

当单击应用程序中的按钮时,应用程序能触发某个事件从而执行相应的操作。在 Java

中，javax.swing 包中的 JButton 类用于构建按钮对象。

(1) 按钮 JButton 类的常用方法。

JButton 的常用方法如表 5-2 所示。

表 5-2　JButton 的常用方法

方　法　名	功　　能
JButton()	创建不带标签文本或图标的按钮
JButton(Action a)	创建一个按钮，其属性从 Action 中获取
JButton(Icon icon)	创建一个带图标的按钮
JButton(String text)	创建一个带标签文本的按钮
JButton(String text, Icon icon)	创建一个带标签文本和图标的按钮
getLabel()	获取按钮上的标签文本内容
setLabel(String label)	设置按钮的标签文本内容
setText(String　s)	设置按钮上的标签文本内容
setIcon(Icon　icon)	设置按钮上的图标
setHorizontalTextPosition(int textPosition)	参数 textPosition 确定按钮上图标的位置，取值为 SwingConstants.RIGHT、SwingConstants.LEFT、SwingConstants.CENTER、SwingConstants.LEADING、SwingConstants.TRAILING
setVerticalTextPosition(int textPosition)	参数 textPosition 确定按钮上图标的位置，取值为 SwingConstants.CENTER、SwingConstants.TOP、SwingConstants.BOTTOM
setMnemonic(char　c)	设置按钮的快捷键为(Alt+C)
setEnabled(boolean　a)	设置按钮可否单击
addActionListener(ActionListener l)	设置事件源的监视器
removeActionListener(ActionListener l)	删除事件监视器

(2) 创建按钮对象。

创建按钮对象的方法为：

```
JButton btn = new JButton(String text);
```

由于按钮是一个普通组件，设计时必须放置到一个容器中。下面的示例就是将按钮放置到一个窗体容器内。

【例 5-3】 构造一个带按钮的窗体。

```
1  /* 构造带按钮的窗体程序 */
2  import javax.swing.*;
3  import java.awt.FlowLayout;                          //引用 awt 包的界面布局管理
4  class Btn extends JFrame
5  {
6    JButton btn = new JButton("确定");                 //创建按钮对象
7     Btn()
8     {
9       setSize(300,200);                               //设置窗体大小
10      setVisible(true);                               //设置窗体可见
11      setDefaultCloseOperation(EXIT_ON_CLOSE);        //设置关闭窗体，退出程序
12      setLayout(new FlowLayout());                    //设置窗体为浮动布局
13      add(btn);                                       //把按钮对象添加到窗体中
14      validate();                                     //使窗体中的组件为可见
```

```
15       }
16   }
17
18 public class  Example5_3
19 {
20     public static void main(String[ ] args)
21     {
22         Btn btn = new Btn();
23     }
24 }
```

【程序说明】

(1) 在本例设计 Btn 和 Ex5_3 两个类。按 Java 的命名规则,在一个文件中,只能有一个类可以用 public 修饰,且文件名必须和带 public 的类同名。因此,本程序的文件名必须命名为 Example5_3.java。

(2) 程序的第 3 行为引用界面布局管理,第 12 行为把窗体布局设置为浮动布局。在窗体中如果不进行界面布局管理,添加到窗体中的按钮组件将与窗体的内部空间一样大小。关于界面布局管理内容将在 5.3 节详细讲解。

图 5-4 按钮程序运行结果

(3) 第 6 行为实例化 JButton 按钮对象,第 13 行把实例化后的按钮对象添加到窗体中。

(4) 第 14 行 validate()是窗体 JFrame 的一个方法,其功能是使窗体中的组件为可见。

程序运行结果如图 5-4 所示。

2. 处理按钮事件

用鼠标单击例 5-3 中的按钮什么也不会发生,这是因为在例 5-3 中没有定义按钮的事件。要定义按钮的处理事件,需要用到 ActionListener 接口。ActionListener 是 java.awt.event 包中的一个接口,定义了事件的处理方法。java.awt.event 包对这个接口的定义是:

```
public interface ActionListener extends EventListener
{       //说明抽象方法
       public abstract void actionPerformed(ActionEvent e)
}
```

在设计按钮对象 btn 处理事件的类时,就要实现这个接口,其一般形式如下:

```
class  ClassName  implements ActionListener
  {
       ⋮
       btn.addActionListener(this);
       ⋮
       public void actionPerformed(ActionEvent e)
        {
            ⋮
         }
  }
```

其中，ClassName 为监听对象的类名，通过实现 ActionListener 接口，使得监视器能知道事件的发生。在 Java 中，要求产生事件的组件向它的监视器注册，这样事件源与监视器就建立了一个关联。建立关联的语句如下。

```
对象名.addActionListener(ClassName);
```

其中，对象是事件源，ClassName 是监视器。例如：

```
btn.addActionListener(this);
```

这条语句的意思是，按钮对象（事件源）btn 向它的监视器（当前类）注册，也就是产生事件的事件源对象向监视器注册。

当单击按钮对象 btn 时，按钮对象（事件源）会产生一个 ActionEvent 事件，事件监视器监听到这个事件，把它作为实现 ActionListener 接口的 actionPerformed 方法参数，在 actionPerformed 方法中处理动作事件。

【例 5-4】 设计一个按钮事件程序。

```
1  /* 按钮触发事件示例 */
2  import javax.swing.*;
3  import java.awt.FlowLayout;
4  import java.awt.event.*;
5  class BtnIcon extends JFrame implements ActionListener
6  {
7    ImageIcon icon = new ImageIcon("win.jpg");                    //创建图标对象
8    JButton jbtn = new JButton("打开新窗体",icon);
9    BtnIcon()
10    {
11      setSize(200,200);
12      setVisible(true);
13      setTitle("按钮功能演示");
14      setDefaultCloseOperation(EXIT_ON_CLOSE);
15      setLayout(new FlowLayout());
16      add(jbtn);
17      validate();
18      jbtn.addActionListener(this);
19    }
20   public void actionPerformed(ActionEvent e)
21    {
22        JFrame newf = new JFrame("新建窗体");
23        newf.setSize(150,150);
24        newf.setVisible(true);
25    }
26 }                                                                //BtnIcon 类结束
27 public class Example5_4
28 {
29    public static void main(String[] args)
30    {  new BtnIcon();    }  ← 由于该对象只使用一次，故创建匿名对象
31 }
```

【程序说明】

(1) 在程序的第 5 行的类声明中,通过 implements ActionListener 实现监听接口。由于接口 ActionListener 在 java.awt.event 包中,故在第 4 行引用该包。

(2) 在程序的第 18 行设置监听对象(按钮 Jbtn)向监听器(当前类)注册。一旦单击按钮,立刻被监听器接收,从而触发第 20 行 actionPerformed()方法中的 ActionEvent 事件。

(3) 为了创建一个图标,可以用 ImageIcon 类创建图标对象:

```
ImageIcon icon = new ImageIcon(String filename);
```

程序的第 7 行创建了一个图标对象,以便在按钮中使用。其中,filename 为图标的路径及文件名,若缺省路径,则要将图标文件存放到程序的同一目录中。

程序运行结果如图 5-5 所示。

图 5-5 按钮事件

5.3 面板容器和界面布局管理

5.3.1 面板 JPanel 类

面板 JPanel 是一个可放置其他组件的容器。作为普通容器,必须将它放置到一个顶层容器(窗体)内。可以在 JPanel 中使用 add 方法放置其他组件。面板主要用于合理安排界面布局。

创建面板的一般步骤如下。

(1) 创建面板对象。

```
JPanel myPanel = new Panel();
```

(2) 将面板添加到窗体容器中。

```
add(myPanel);
```

(3) 把组件放置到面板上。

```
myPanel.add(其他组件);
```

【例 5-5】 面板使用示例。

```
1   /* 面板 JPanel 简单示例 */
2   import java.awt.*;
3   import javax.swing.*;
4   class PanelTest extends JFrame
5   {
6     JPanel panel1 = new JPanel();
7     JPanel panel2 = new JPanel();
```

```
8         JButton button1 = new JButton("Button1");
9         JButton button2 = new JButton("Button2");
10        JButton button3 = new JButton("Button3");
11        JButton button4 = new JButton("Button4");
12   PanelTest()
13    {
14        setSize(200,150);
15        setVisible(true);
16        setTitle("面板容器示例");
17        setDefaultCloseOperation(EXIT_ON_CLOSE);
18        setLayout(new FlowLayout());
19        //将面板容器加入窗体中
20        add(panel1);
21        add(panel2);
22        //将其他组件加入面板容器中
23       panel1.add(button1);
24       panel1.add(button2);
25       panel2.add(button3);
26       panel2.add(button4);
27       panel1.setBackground(Color.red);
28       panel2.setBackground(Color.cyan);
29       validate();
30     }
31  }
32
33   public class Example5_5
34   {
35     public static void main(String[] args)
36       {      new PanelTest();      }
37   }
```

设置背景色,使其可见

使用匿名对象

图 5-6　面板容器

程序运行结果如图 5-6 所示。

5.3.2　界面布局策略

当运行前面的例 5-8 程序时,如果把窗体拉大一些,窗体内部的组件也会跟着发生变化。那么,通过学习界面布局管理的知识以控制组件在容器的位置。

Java 在 java.awt 包中定义了 5 种界面布局策略,分别是 FlowLayout、BorderLayout、CardLayout、GridLayout 和 GridBagLayout。

设置布局的格式为:

```
容器对象.setLayout(布局策略);
```

1. 浮动布局 FlowLayout

浮动布局是按照组件的顺序,用 add 方法将组件从左至右在一行排列,一行放不下时就自动换行,每行组件均按居中方式进行排列。这个布局方式在前面的示例中多次使用。

其设置的方法为:

```
setLayout(new FlowLayout());
```

2. 边界布局 BorderLayout

BorderLayout 类把容器划分成 5 个区域,分别标记为 North、South、West、East 和 Center。每

个组件用 add 方法放置到区域中,中间区域的空间最大。

其设置的方法为:

```
setLayout(new BorderLayout());
```

【例 5-6】 边界布局示例。

```
1  /* 边界布局示例 */
2  import java.awt.*;
3  import javax.swing.*;
4  class BordTest extends JFrame
5   {
6     BordTest()
7     {
8        setSize(300,200);
9        setVisible(true);
10       setTitle("边界布局示例");
11       setDefaultCloseOperation(EXIT_ON_CLOSE);
12       setLayout(new BorderLayout());
13       //将其他组件加入到窗体
14       add("East",   new JButton("东"));
15       add("South",  new JButton("南"));
16       add("West",   new JButton("西"));
17       add("North",  new JButton("北"));
18       add("Center", new JButton("中"));
19       validate();
20     }
21   }
22
23  public class Example5_6
24   {
25      public static void main(String[] args)
26       {    new BordTest();      }
27   }
```

图 5-7 边界布局排列组件

程序运行结果如图 5-7 所示。

3. 网格布局 GridLayout

GridLayout 类以矩形网格形式对容器中的组件进行布局。容器被分成大小相等的单元格,单元格的大小由最大的构件决定,用 add 方法将组件一行一行地从左至右放置到布局的每个单元格中。

其设置的方法为:

```
setLayout(new GridLayout(int row, int cols));
```

其中,row 是网格的行数;cols 是网格的列数。

【例 5-7】 网格布局示例。

```
1  /* 网格布局示例 */
2  import java.awt.*;
3  import javax.swing.*;
4  class GridTest extends JFrame
5  {
```

```
6     GridTest()
7     {
8        setSize(300,200);
9        setVisible(true);
10       setTitle("网格布局示例");
11       setDefaultCloseOperation(EXIT_ON_CLOSE);
12       setLayout(new GridLayout(3,2));                    //设置网格布局
13       //下面通过循环构造一组按钮并将其加入窗体中
14       for (int i = 1;i < = 6 ;i++)
15         {
16            add(new JButton("按钮" + i));
17          }
18        validate();
19    }
20 }
21
22 public class Example5_7
23 {
24   public static void main(String[ ] args)
25    {
26       new GridTest();
27    }
28 }
```

程序运行结果如图 5-8 所示。

4. 卡片布局 CardLayout

这种布局包含几个卡片，在某一时刻只有一个卡片是可见的，而且第一个卡片显示的内容可用自己的布局来管理。

CardLayout 的布局可以包含多个组件，但是实际上某一时刻容器只能从这些组件中选出一个来显示，就像一叠“扑克牌”每次只能显示最上面的一张，如图 5-9 所示。

图 5-8　网格布局排列组件

图 5-9　一叠“扑克牌”

卡片布局设置的方法为：

```
setLayout(new CardLayout());
```

卡片的顺序由组件对象本身在容器内部的顺序决定。CardLayout 定义了一组方法，这些方法允许应用程序按顺序地浏览这些卡片，或显示指定的卡片。

CardLayout 的主要方法如下。

- first(Container parent)：显示容器 parent 中的第一张卡片。

- next(Container parent)：显示容器 parent 中的下一张卡片。
- previous(Container parent)：显示容器 parent 中的前一张卡片。
- last(Container parent)：显示容器 parent 中的最后一张卡片。

【例 5-8】 卡片布局示例。

```
/* 卡片布局示例 */
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
class CardTest extends JFrame implements ActionListener
{
   JButton btn[] = new JButton[5];
    CardLayout card = new CardLayout();
    Panel p = new Panel();
    CardTest()
     {
       setSize(300,200);
       setVisible(true);
       setTitle("卡片布局示例");
       setDefaultCloseOperation(EXIT_ON_CLOSE);
       add(p);
       p.setLayout(card);
       for (int i = 1;i <= 4;i++)
        {
          btn[i] = new JButton("卡片" + i);
          p.add(btn[i],"卡片标识" + i);
          btn[i].addActionListener(this);
        }
       validate();
     }

  public void actionPerformed(ActionEvent e)
   {
       card.next(p);
   }
}

public class Example5_8
{
  public static void main(String[] args)
    {
       new CardTest();
    }
}
```

程序运行结果如图 5-10 所示。

图 5-10 卡片布局显示最上面一个组件

5.4 JComponent 类组件的使用

5.4.1 JComponent 类组件

JComponent 类是除顶层容器外的所有 swing 组件的父类，其常用子类如表 5-3 所示。

表 5-3　JComponent 类的一些常用子类

类　名	功　能	类　名	功　能
JButton	创建按钮对象	JProgressBar	创建进度指示条
JComboBox	创建下拉列表对象	JRadioButton	创建单选按钮
JCheckBox	创建复选框对象	JScrollBar	创建滚动条
JFileChooser	创建文件选择器	JScrollPane	创建滚动窗体
JLabel	创建文本标签	JSlider	创建滑杆
JMenu	创建菜单对象	JSplitPane	创建拆分窗体
JMenuBar	创建菜单条对象	JTable	创建表格
JMenuItem	创建菜单项对象	JTestArea	创建文本区
JPanel	创建面板对象	JToolBar	创建工具条
JPasswordField	创建口令文本框对象	JToolTip	创建工具提示对象
JPopupMenu	创建弹出式菜单	JTree	创建树对象

JComponent 类的常用方法如表 5-4 所示，它们是表 5-3 中所列的子类都继承了这些方法。

表 5-4　JComponent 类的常用方法

类别	方　法　名	功　能
设置组件的颜色	void　setBackground(Color c)	设置组件的背景色
	void　setForeground(Color c)	设置组件的前景色
设置组件的字体	void setFont(Font f)	设置组件上的字体
	Font(String name, int style, int size)	字体的构造方法：name 为字体名称，style 为字体式样 size 为字体大小
设置组件的大小与位置	void setSize(int width, int height)	设置组件的宽度和高度
	void setLocation(int x, int y)	设置组件在容器中的位置，左上角坐标为(0,0)
	void setBounds(int x, int y, int width, int height)	设置组件在容器中的位置及组件大小
设置组件的激活与可见性	void setEnabled(boolean b)	设置组件是否被激活
	boolean isEnabled()	判断组件是否为激活状态
	void setVisible(boolean b)	设置组件是否可见

在上述设置组件颜色的方法中，Color 类是 Java.awt 包中的类。用 Color 类的构造方法 Color(int red,int green,int blue)可以创建一个颜色对象，三个颜色值取值都为 0～255。Color 类还有 red、blue、green、orange、cyan、yellow、pink 等静态常量。

5.4.2　文本组件和标签

1. JTextComponent 类

JTextComponent 是一个允许设置、检索和修改文本的类。它是 swing 文本组件的基类，通过它定义了 JTextField、JTextArea 和 JEditorPane 3 个子类。它们的继承关系如图 5-11 所示。

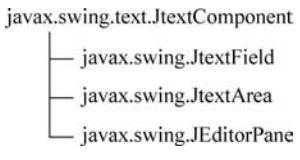

图 5-11　JTextComponent 类组件的继承关系

JTextComponent 类的常用方法如表 5-5 所示。

表 5-5 JTextComponent 类常用方法

方 法 名	功 能
void setText(String t)	设置文本内容
string getText()	获取文本内容
boolean isEdit()	检测文本的可编辑性
void setEditable(boolean b)	设置文本的可编辑性
string getSelectedText()	获取选取文本内容
void select(int selStart,int selEnd)	选取文本内容
void copy()	将选定的内容传输到系统剪贴板
void cut()	将选定的内容传输到系统剪贴板,并把文本组件中的内容删除

由于 JTextComponent 是文本框 JTextField 类、文本区 JTextArea 类及文本组件 JEditorPane 共同的父类。因此,下面要介绍的文本框 JTextField 类、文本区 JTextArea 类及文本组件 JEditorPane 都继承了 JTextComponent 的方法属性。

2. 文本框 JTextField

文本框 JTextField 是对单行文本进行编辑的组件,用来接受用户的输入码或显示可编辑的文本。

(1) 文本框 JTextField 类的定义。

swing 对文本框 JTextField 类的定义如下。

```
1  public class JTextField extends JTextComponent
2  {
3    public JTextField();
4    publicJTextField(String text);
5    public JTextField(int columns);
6    publicJTextField(String text,int columns);
7    public void addActionListener(ActionListener l);
8    public void remove ActionListener(ActionListener l);
9  }
```

其中,第 3～6 行是文本框的构造方法。

第 7 行 addActionListener(ActionListener l)为指定事件监听者。

第 8 行 remove ActionListener(ActionListener l)为删除事件监听者。

由于 JTextField 是 JTextComponent 的子类,因此还具有 JTextComponent 所有的方法和属性。

(2) 创建文本框。

创建文本框时,一般要以初始的文本字符串或能容纳的字符数为参数:

```
JTextField  text = new  JTextField(String str);
```

(3) 主要方法。

它主要继承其父类 JTextComponent 的方法。从上述文本框的定义可知,它可以实现 ActionListener 监听接口的方法。

【例 5-9】 文本框应用示例。

```
1  /* JTextField 类示例 */
```

```
2  import javax.swing.*;
3  import java.awt.FlowLayout;
4  class TxtfdTest extends JFrame
5  {
6    JTextField txt;                                    //声明文本框对象
7    TxtfdTest()
8    {
9      setSize(300,200);
10     setVisible(true);
11     setTitle("创建文本框示例");
12     setDefaultCloseOperation(EXIT_ON_CLOSE);
13     setLayout(new FlowLayout());                     //设置窗体为浮动布局
14     txt = new JTextField(20);                        //对象实例化
15     add(txt);                                        //将文本框添加到窗体中
16     validate();
17     txt.setText("重新设置了文本");                    //设置文本内容
18   }
19 }
20
21 public class Example5_9
22 {
23   public static void main(String[] args)
24   {
25     new TxtfdTest();
26   }
27 }
```

【程序说明】

程序的第 2 行引用了 swing 类包，因此可以在第 4 行将类声明为 JFrame 的子类，并且可以使用文本组件 JTextField。第 6 行声明 TextField 对象，第 14 行实例化对象。第 15 行把文本框添加到窗体中，第 16 行的 validate 方法使文本框对象显示出来。第 17 行重新设置文本框的内容。

程序运行结果如图 5-12 所示。

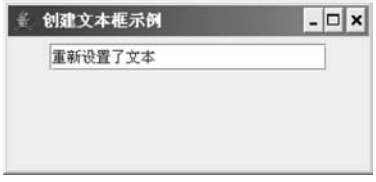

图 5-12　文本框组件程序运行结果

3. 密码框 JPasswordField

密码框 JPasswordField 是 JtextField 的子类，允许编辑单行文本，可以设置为不显示输入的原始字符，而是显示指定的回显字符。

JPasswordField 类的主要方法为 setEchoChar(char c)，其中的字符 c 为回显字符。

【例 5-10】 设计一个密码验证程序。

```
1  /* 密码框示例   */
2  import javax.swing.*;
3  import java.awt.*;
4  import java.awt.event.*;
5  class Passwd extends JFrame implements ActionListener
6   {
7     JLabel lb = new JLabel("请输入密码:");            //创建标签对象
8     JPasswordField txt1 = new  JPasswordField(25);    //创建密码框对象
9     JButton bn = new JButton("确定");
10     JTextField txt2 = new JTextField(25);
```

```
11      Passwd()
12      {
13        setSize(300,200);
14        setVisible(true);
15        setTitle("密码验证");
16        setDefaultCloseOperation(EXIT_ON_CLOSE);
17        setLayout(new FlowLayout());
18        add(lb);
19        add(txt1);
20        txt1.setEchoChar('*');                          //设置回显的字符为*号
21        add(bn);
22        add(txt2);
23        validate();
24        bn.addActionListener(this);
25      }
26
27     public void actionPerformed(ActionEvent e)
28      {
29         if (txt1.getText().equals("abc"))              //比较字符串相等
30            txt2.setText("密码正确!!");
31         else
32            txt2.setText("密码错误!!");
33       }
34   }
35
36  public class  Example5_10
37   {
38     public static void main(String[] args)
39      {
40         new Passwd();
41      }
42   }
```

图 5-13　密码框组件程序运行结果

【程序说明】

程序的第 29 行,比较两个字符串的内容是否相等,要用 equals 方法。若比较两个数值是否相等,则用==符号。

程序运行结果如图 5-13 所示。

4. 文本区 JTextArea

文本区 JTextArea 是对多行文本进行编辑的组件,用控制符来控制文本的格式。例如,\n 为换行,\t 为插入一个 Tab 字符。

(1) 文本区 JTextArea 类的定义。

TextArea 类是 JTextComponent 的子类,继承了其父类的方法和属性。swing 对这个类的定义如下。

```
1  public class JTextArea extends JTextComponet
2  {
3    public JTextArea()
4    public JTextArea(String text)
5    public JTextArea(int rows,int columns)
6    public JTextArea(String text,int rows,int cols)
7    public void append(String str)
```

```
8     public void insert(String str,int pos)
9     public void replaceRange(String str,int start,int end)
10 }
```

其中,第 3～6 行是文本区的构造方法。

第 7 行 append(String str)方法向文本区追加文本内容。

第 8 行 insert(String str,int pos)方法在指定的位置插入文本内容。

第 9 行 replaceRange(String str,int start,int end)方法替换指定的开始与结束位置间的文本内容,str 为替换的文本,start 为开始位置,end 为结束位置。

(2) 创建文本区。

通常创建文本区时,要说明这个文本区的行数、列数或文本内容:

```
JTextArea txt1 = new JTextArea(7,35);
```

(3) 主要方法。

它主要继承 JTextComponent 类的方法,从文本区的定义可知,它还有 append(String str)等方法。

通过下面的例子,可以了解文本组件的基本功能及使用方法。

【例 5-11】 JTextArea 类示例。

```
1  /* JTextArea 类示例 */
2  import javax.swing.*;
3  import java.awt.*;
4  import java.awt.event.*;
5  class AreaTest extends JFrame implements ActionListener
6  {
7    JTextArea txt1 = new JTextArea(7,35);              //创建文本区对象
8    JTextField txt2 = new JTextField(35);              //创建文本框对象
9    String str = "窗外飘起蒙蒙小雨,\n平添一夜寒意," +
10                "\n多少的思绪藏在心底,";
11     AreaTest()
12     {
13        setSize(400,300);
14        setVisible(true);
15        setTitle("文本组件示例");
16        setDefaultCloseOperation(EXIT_ON_CLOSE);
17        setLayout(new FlowLayout());                   //设置浮动布局
18        txt1.setText(str);                             //设置文本区的文本内容
19        add(txt1);                                     //将文本区添加到窗体中
20        add(txt2);                                     //将文本框添加到窗体中
21        validate();
22        txt2.addActionListener(this);                  //把文本框注册为监听对象
23     }
24
25     public void actionPerformed(ActionEvent e)
26     {
27       String s = txt2.getText();                      //从文本框中获取文字内容
28       txt1.append("\n" + s);                          //将文本框的字符串追加到文本区中
29      }
30  }
31
32 public class  Example5_11
```

```
33 {
34    public static void main(String[ ] args)
35      {  new AreaTest();    }
36 }
```

图 5-14　文本区组件程序运行结果

程序运行结果如图 5-14 所示，在文本框中输入文字内容，按 Enter 键，则将文本框中的文字追加到文本区中。

5. 标签 JLabel 类

标签是用户只能查看其内容但不能修改的文本组件，一般用作说明。在例 5-10 密码验证程序中已经使用了标签。标签 JLabel 上可以添加图像，当鼠标指针悬停在标签上时，可以显示一段提示文字。标签 JLabel 的常用方法如表 5-6 所示。

表 5-6　标签 JLabel 类常用方法

方 法 名	功 能
JLabel()	创建空标签的构造方法
JLabel(String text)	创建具有文字 text 的构造方法
JLabel(Icon icon)	创建具有图标 icon 的构造方法
JLabel(String s, Icon icon, int textPosition)	创建具有文字、图标和排列方式的构造方法 参数 textPosition 确定标签上图标的位置，取值为 SwingConstants. RIGHT、SwingConstants. LEFT、SwingConstants. CENTER、SwingConstants. LEADING、SwingConstants. TRAILING
setText(String s)	设置标签内容
setIcon(Icon icon)	设置标签的图标
setToolTipText(String text)	设置当鼠标指针悬停在标签上时显示文字 text 内容

【例 5-12】 创建一个包含带有图标的按钮和标签的窗体。

```
1  /* 带有图标的按钮和标签 */
2  import javax.swing.*;
3  import java.awt.*;
4  import java.awt.event.*;
5  class LbTest extends JFrame implements ActionListener
6  {
7    LbTest(String s)
8     {
9      setSize(300,200);
10     setVisible(true);
11     setTitle(s);
12     setLayout(new FlowLayout());
13     ImageIcon icon1 = new ImageIcon("s1.jpg");
14     ImageIcon icon2 = new ImageIcon("s2.jpg");
15     ImageIcon icon3 = new ImageIcon("s3.jpg");
16     JButton jbtn = new JButton("我是按钮",icon1);
17     jbtn.setRolloverIcon(icon2);                    //当鼠标指针悬停在按钮上时变换图标
18     JLabel jlb = new JLabel("我是标签",icon3,SwingConstants.CENTER);
19     jlb.setToolTipText("QQ 头像");                  //当鼠标指针悬停在标签上时显示提示文本
```

```
20      add(jbtn);
21      add(jlb);
22      jbtn.addActionListener(this);
23      setDefaultCloseOperation(EXIT_ON_CLOSE);
24      validate();
25      }
26    public void actionPerformed(ActionEvent e)
27    {
28        JInternalFrame in_window;                          //声明内部窗体对象
29        in_window = new JInternalFrame("内部窗体",true,true,true,true);
30        in_window.setSize(250,200);
31        in_window.setVisible(true);
32        add(in_window);
33        JTextArea text = new JTextArea(5,15);              //创建文本区对象
34        in_window.add(text,BorderLayout.CENTER);           //按边界布局添加到窗体中
35     }
36  }
37
38  public class Example5_12
39  {
40       public static void main(String args[])
41        {
42            LbTest win = new LbTest("有图标的按钮和标签");
43        }
44  }
```

【程序说明】

(1) 程序的第 16 行和第 18 行创建了一个带图标的按钮和一个带图标的标签。

(2) 第 17 行设置按钮对象的翻滚图标,即当鼠标指针悬停在按钮上时,按钮上的图标由 icon1 变换为 icon2。

(3) 第 19 行设置标签的提示文本,当鼠标指针悬停在标签上时将显示提示文本内容。

(4) 在第 28 行和第 29 行创建了一个内部窗体对象。JinternalFrame 的构造方法为:

```
JInternalFrame(String title,可否改变大小,可否关闭,可否最大化,可否最小化)
```

当参数为 true 时,可改变;否则,不可改变。

程序运行结果如图 5-15 所示。

(a) 鼠标指针悬停在标签上显示提示信息　　(b) 单击按钮打开内部窗体

图 5-15　带有图标的按钮和标签

5.4.3　单选按钮、复选框和下拉列表

单选按钮(JRadioButton)、复选框(JCheckBox)和下拉列表(JComboBox)是一组表示多

种“选择”的 swing 组件。

1. 单选按钮(JRadioButton)和复选框(JCheckBox)

单选按钮 JradioButton 和复选框 JCheckBox 都只有两种状态：选中或未选中。它们的构造方法和其他常用方法类似，故把它们放在一起介绍。

复选框 JCheckBox 的常用方法如表 5-7 所示。

表 5-7 复选框 JCheckBox 及单选按钮 JradioButton 类常用方法

方 法 名	功 能
JCheckBox()或 JRadioButton()	没有标签的构造方法
JCheckBox(String s)或 JRadioButton(String s)	具有标签 s 的构造方法
JCheckBox(Icon icon)或 JRadioButton(Icon icon)	具有图标 icon 的构造方法
JCheckBox(String s, Icon icon)或 JRadioButton(String s, Icon icon)	具有标签和图标的构造方法
JCheckBox(String s, Icon icon,boolean t)或 JRadioButton(String s, Icon icon,boolean t)	具有标签和图标且初始状态为 t 的构造方法
getItem()	获取产生事件的对象
getStateChange()	返回该选择项是否被选中

【例 5-13】 创建包含单选按钮和复选框的窗体。

```
1  /* 单选按钮和复选框示例 */
2  import java.awt.*;
3  import java.awt.event.*;
4  import javax.swing.*;
5  class BRTest extends JFrame implements ItemListener,ActionListener
6  {
7     JTextField text = new JTextField(15); ;
8     BRTest(String s)
9     {
10     setSize(200,200);
11     setVisible(true);
12     setTitle(s);
13     setLayout(new FlowLayout());
14     //添加 3 个复选框
15     JCheckBox cb1 = new JCheckBox("C 语言");
16     cb1.addItemListener(this);
17     add(cb1);
18     JCheckBox cb2 = new JCheckBox("C++语言");
19     cb2.addItemListener(this);
20     add(cb2);
21     JCheckBox cb3 = new JCheckBox("Java 语言");
22     cb3.addItemListener(this);
23     add(cb3);
24     //添加 3 个单选按钮
25     JRadioButton b1 = new JRadioButton("鲜花");
26     b1.addActionListener(this);
27     add(b1);
28     JRadioButton b2 = new JRadioButton("鼓掌");
29     b2.addActionListener(this);
30     add(b2);
```

```
31      JRadioButton b3 = new JRadioButton("鸡蛋");
32      b3.addActionListener(this);
33      add(b3);
34      //定义按钮组,单选按钮只有放到按钮组中才能实现单选功能
35      ButtonGroup bg = new ButtonGroup();
36      bg.add(b1);
37      bg.add(b2);
38      bg.add(b3);
39      //添加文本框
40      add(text);
41      validate();
42      setDefaultCloseOperation(EXIT_ON_CLOSE);
43     }
44    public void itemStateChanged(ItemEvent ie)
45     {
46       JCheckBox cb = (JCheckBox)ie.getItem();
47       text.setText(cb.getText());
48     }
49    public void actionPerformed(ActionEvent ae)
50     {
51       text.setText(ae.getActionCommand());
52     }
53  }
54  //主类
55  public class Example5_13
56  {
57     public static void main(String args[])
58     {
59          new BRTest("单选按钮和复选框示例");
60     }
61  }
```

【程序说明】

(1) 复选框 JCheckBox 实现 ItemListener 接口,通过 itemStateChanged 方法,触发 ItemEvent 事件,如本程序的第 44～48 行。

(2) 单选按钮 JRadioButton 实现 ActionListener 接口,通过 actionPerformed 方法,触发 ActionEvent 事件,如本程序的第 49～52 行。

(3) 程序的第 35～38 行将单选按钮 JRadioButton 对象放到按钮组 ButtonGroup 中,只有放到按钮组中才能实现一次只能选中一个按钮的单选功能。

程序运行结果如图 5-16 所示。

图 5-16　单选按钮和复选框示例

2. 下拉列表(JComboBox)

下拉列表(JComboBox)通常显示一个可选条目,允许用户在一个下拉列表中选择不同条目,用户也可以在文本区内输入选择项。JComboBox 的构造函数如下:

```
JComboBox()
JComboBox(Vector v)
```

其中,v 是初始选项。

要增加选项,则使用方法:

```
void addItem(Object obj)
```

其中,obj 是加入下拉列表的对象。

【例 5-14】 创建包括一个下拉列表和一个标签的窗体。标签显示一个图标。下拉列表的可选条目是“中国”“俄罗斯”“韩国”“联合国”。当选择一个图标,标签就更新为这个国家的国旗。

```
/* 下拉列表示例 */
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
class CobTest extends JFrame implements ItemListener
{
    JLabel jlb;
    ImageIcon france, germany, italy, japan;
    CobTest(String s)
     {
        setSize(300,200);
        setVisible(true);
        setTitle(s);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLayout(new FlowLayout());
        JComboBox jc = new JComboBox();
        jc.addItem("中国");
        jc.addItem("俄罗斯");
        jc.addItem("韩国");
        jc.addItem("联合国");
        jc.addItemListener(this);
        add(jc);
        jlb = new JLabel(new ImageIcon("中国.jpg"));
        add(jlb);
        validate();
      }
   public void itemStateChanged(ItemEvent ie)
   {
      String s = (String)ie.getItem().toString();
      jlb.setIcon(new ImageIcon(s + ".jpg"));
   }
}
//主类
public class Example5_14
 {
    public static void main(String args[])
       {  new CobTest("下拉列表示例");   }
 }
```

图 5-17 下拉列表程序运行结果

程序运行结果如图 5-17 所示。

5.4.4 卡片选项页面(JTabbedPane)

在 5.3.2 节中介绍过卡片布局 CardLayout,该布局不太直观,swing 用 JTabbedPane 类对它进行了修补,由 JTabbedPane 处理这些卡片面板。在设计用户操作界面时,使用卡片选

项页面，可以扩大安排功能组件的范围，用户操作起来更加方便。

【例 5-15】 卡片选项页面示例。

```
1  /* 卡片选项页面 */
2  import javax.swing.*;
3  import java.awt.*;
4  import java.awt.event.*;
5  class TtpDemo extends JFrame
6   {
7      TtpDemo()
8      {
9        super("卡片选项页面示例");
10       setSize(300,200);setVisible(true);
11       JTabbedPane jtp = new JTabbedPane();
12       ImageIcon icon1 = new ImageIcon("c1.gif");
13       ImageIcon icon2 = new ImageIcon("c2.gif");
14       ImageIcon icon3 = new ImageIcon("c3.gif");
15       jtp.addTab("城市",icon1, new CitiesPanel(),"城市名称");
16       jtp.addTab("文学", icon2, new BookPanel(),"文学书目");
17       jtp.addTab("网站", icon3, new NetPanel(),"精选网址");
18       getContentPane().add(jtp);
19       validate();
20       setDefaultCloseOperation(EXIT_ON_CLOSE);
21      }
22   }
23  //定义面板 CitiesPanel
24 class CitiesPanel extends JPanel
25  {
26    CitiesPanel()
27    {
28        JButton b1 = new JButton("北京");
29        JButton b2 = new JButton("上海");
30        JButton b3 = new JButton("深圳");
31        JButton b4 = new JButton("厦门");
32        add(b1); add(b2); add(b3); add(b4);
33      }
34  }
35  //定义面板 BookPanel
36 class BookPanel extends JPanel
37  {
38     BookPanel()
39     {
40         JCheckBox cb1 = new JCheckBox("西游记");
41         JCheckBox cb2 = new JCheckBox("三国演义");
42         JCheckBox cb3 = new JCheckBox("红楼梦");
43         add(cb1); add(cb2); add(cb3);
44       }
45  }
46  //定义面板 NetPanel
47 class  NetPanel extends JPanel
48  {
49    NetPanel()
50    {
51       JComboBox jcb = new JComboBox();
52       jcb.addItem("思维论坛");
```

```
53        jcb.addItem("百度搜索");
54        jcb.addItem("Java 爱好者");
55        add(jcb);
56        }
57    }
58  //主类
59 public class Example5_15
60 {   public static void main(String args[])
61       {   new TtpDemo();    }
62 }
```

【程序说明】

(1) 在程序的第 23～57 行,设计了三个面板类(CitiesPanel、BookPanel 和 NetPanel)。

(2) 在程序的第 11 行建立 JTabbedPane 的实例对象 JTp。

(3) 通过实例对象 JTp 调用方法 addTab()将面板添加到 JTabbedPane 中。

addTab 方法有 3 种结构方式:

- addTab(String title, Component component);
- addTab(String title, Icon icon, Component component);
- addTab(String title, Icon icon, Component component, String tip);

其中,title 为卡片标题; icon 为卡片图标; component 为放到选项页面中的面板; tip 为当鼠标停留在该页面标题时显示的提示文字。

本程序的第 15～17 行使用的是 addTab 方法的第三种结构方式。

程序运行结果如图 5-18 所示。

图 5-18　卡片选项页面

5.4.5　滑杆(JSlider)和进度指示条(JProgressBar)

滑杆 JSlider 能通过一个滑块的来回移动输入数据,在很多情况下显得直观(如声音控制)。进程条 JProgressBar 从“空”到“满”显示相关数据的状态,因此用户得到了一个状态的透视。下面的例子将滑动块同进程条挂接起来,当移动滑动块时,进程条也相应地改变。

【例 5-16】 滑杆和进度指示条配合使用示例。

```
1  /* 滑杆和进度指示条 */
2  Import java.awt.*;
3  Import java.awt.event.*;
4  Import javax.swing.*;
5  Import javax.swing.event.*;
6  Import javax.swing.border.*;
7  class P extends JPanel {
8     JProgressBar pb = new JProgressBar();
9     JSlider sb = new JSlider(JSlider.HORIZONTAL, 0, 100, 60);
10    public P() {
11       setLayout(new GridLayout(2,1));
12       add(pb);
13       sb.setValue(0);
14       sb.setPaintTicks(true);
```

```
15        sb.setMajorTickSpacing(20);
16        sb.setMinorTickSpacing(5);
17        sb.setBorder(new TitledBorder("移动滑杆"));
18        pb.setModel(sb.getModel());
19        pb.setStringPainted(true);
20        add(sb);
21     }
22  }
23  public class Example5_16{
24     public static void main(String args[]) {
25        JFrame f = new JFrame("滑杆和进度指示条");
26        f.setSize(300,150);
27        f.add(new P());
28        f.show();
29        f. setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
30     }
31  }
```

【程序说明】

(1) 进程指示条 JProgressBar 是用颜色动态填充一个长条矩形的组件，通常用于显示某任务完成的进度情况。它有以下 3 种常用的构造方法。

- JProgressBar(),
- JProgressBar(int min, int max),
- JProgressBar(int orient, int min, int max)。

其中，参数 orient 决定进度指示条是水平放置还是垂直放置，其取值为 JProgressBar.HORIZONTAL 或 JProgressBar.VERTICAL；min 和 max 表示进度指示条的最大值和最小值。

在程序的第 8 行，应用了第一种构造方法建立 JProgressBar 对象。它的最大值和最小值是系统默认的，分别取值 100 和 0，即把进度指示条 100 等分。

本程序用到 JProgressBar 的方法如下。

- setModel(BoundedRangeModel newModel)：进度条与任务源 newModel 挂接。
- setStringPainted(boolean b)：是否允许在进度条中显示完成进度的百分比。

程序的第 18 行，设置进度条的任务源为 JSlider；程序的第 19 行，设置允许在进度条中显示完成的百分比数。

(2) JSlider 的构造方法与作用基本与 JProgressBar 类似，用于显示某任务完成的进度情况，不同之处是它可以用鼠标拖动。

程序运行结果如图 5-19 所示。

图 5-19　滑杆和进度指示条

5.4.6　表格(JTable)

表格(JTable)是在设计用户界面(user interface)时非常有用的一个组件；尤其在需要将统计数据非常清楚且有条理地呈现在用户面前时，表格设计更能显出它的重要。JTable 组件的主要功能是把数据以二维表格的形式显示出来。

表格 JTable 的常用方法如表 5-8 所示。

表 5-8 表格 JTable 常用方法

方 法 名	功 能
JTable()	创建一个新的 JTable,使用系统默认的 Model
JTable(int row, int col)	创建具有 row 行、col 列的空表格
JTable(object[][]rowData,object[]columnNames)	创建显示二维数组数据表格,且可以显示列的名称。第一个参数是数据,第二个参数是在表格第一行中显示列的名称
JTable(TableModel dm)	创建表格并设置数据模式
JTable(Vector[][]rowData,Vector[]columnNames);	创建以 Vector 为输入来源的数据表格。第一个参数是数据,第二个参数是在表格第一行中显示列的名称
getModel()	获取表格的数据来源对象

【例 5-17】 利用 JTable 编制员工档案表。

```
1  /* 简单 JTable 表格 */
2  import javax.swing.*;
3  import java.awt.*;
4  import java.awt.event.*;
5  class TableDemo extends JFrame
6  {
7    public TableDemo()
8     {
9       super("员工档案表");
10      String[] columnNames = {"姓名","职务","电话","月薪","婚否"};
11      Object[][] data = {
12       {"李  强","经理","059568790231",new Integer(5000),new Boolean(true)},
13       {"吴  虹","秘书","059569785321",new Integer(3500),new Boolean(true)},
14       {"陈卫东","主管","059565498732",new Integer(4500),new Boolean(false)},
15       {"欧阳建","保安","059562796879",new Integer(2000),new Boolean(true)},
16       {"施乐乐","销售","059563541298",new Integer(4000),new Boolean(false)}
17      };
18      JTable table = new JTable(data,columnNames);
19      table.setPreferredScrollableViewportSize(new Dimension(500,70));
20      JScrollPane scrollPane = new JScrollPane(table);
21      getContentPane().add(scrollPane, BorderLayout.CENTER);
22      setDefaultCloseOperation(EXIT_ON_CLOSE);
23      pack();
24      setVisible(true);
25    }
26 }
27  //主类
28 public class Example5_17
29 {
30   public static void main(String[] args)
31    {
32         TableDemo frame = new TableDemo();
33    }
34 }
```

【程序说明】

(1) 程序第 10 行为表格的表头,即各列标题,存放到字符串数组 columnNames 中。

(2) 程序第 11～17 行定义二维数组 data 存放表格数据。

(3) 程序第 18 行创建一个 JTable 类对象,构成一个以数组 columnNames 为列标题、以数组 data 为内容的表格。

(4) 程序第 19 行定义表格的显示尺寸。

(5) 程序第 20 行创建一个带滚动条的面板,把表格放到面板中。

(6) 程序第 21 行将带滚动条的面板添加到窗体中。

程序运行结果如图 5-20 所示。

图 5-20　简单表格

通过上面的例子可知,利用 JTable 类创建一个表格很简单,只要用数组作为表格的数据输入,将数组的元素填入 JTable 中,一个基本的表格就产生了。不过,这种方法虽然简便,但创建的表格还有不少缺陷。例如,上例表格中的每个单元格都只能接受同一种类型的数据,数据类型皆显示为 String 类型,原来声明为 Boolean 的数据都以 String 类型的形式出现。为了对解决这种复杂的情况,swing 提供了多种 Model 来解决这个问题,如 TableModel 类、AbstractTableModel 类等。这些类均放在类库 javax. swing. table package 中,可以在 Java API 中找到这个 package,下面通过改造上例来说明其用法。

设计表格程序时,依据 MVC(Model-View-Controller)的设计思想,先创建一个 AbstractTableModel 类型的对象来存放和处理数据,由于这个类是从 AbstractTableModel 类中继承的,其中有几个方法需要覆盖重写,如 getColumnCount、getRowCount、getColumnName 和 getValueAt。因为 JTable 会从这个对象中自动获取表格显示需要的数据,AbstractTableModel 类的对象负责表格大小的确定(行、列)、内容的填写、赋值、表格单元更新检测等一切与表格内容有关的属性及其操作。JTable 类生成的表格对象以该 TableModel 为参数,并负责将 TableModel 对象中的数据以表格的形式显示出来。

【例 5-18】 修改例 5-17,编制一个加强的员工档案表格。

```
1   /*JTable 表格应用*/
2   import javax.swing.JTable;
3   import javax.swing.table.AbstractTableModel;
4   import javax.swing.JScrollPane;
5   import javax.swing.JFrame;
6   import javax.swing.SwingUtilities;
7   import javax.swing.JOptionPane;
8   import java.awt.*;
9   import java.awt.event.*;
10  class TableDemo extends JFrame
11     {
12       public TableDemo()
13       { //首先调用父类 JFrame 的构造方法生成一个窗口
14        super("员工档案表");
15        //myModel 存放表格的数据
16        MyTableModel myModel = new MyTableModel();
17        //表格对象 table 的数据来源是 myModel 对象
```

```
        JTable table = new JTable(myModel);
        //表格的显示尺寸
        table.setPreferredScrollableViewportSize(new Dimension(500,70));
        //产生一个带滚动条的面板
        JScrollPane scrollPane = new JScrollPane(table);
        //将带滚动条的面板添加到窗口中
        getContentPane().add(scrollPane, BorderLayout.CENTER);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }
}
    //把要显示在表格中的数据存入字符串数组和 Object 数组中
class MyTableModel extends AbstractTableModel
 {
    private boolean DEBUG = true;
     //表格中第一行所要显示的内容存放在字符串数组 columnNames 中
    final String[] columnNames = {"姓名","职务","电话","月薪","婚否"};
    //表格中各行的内容保存在二维数组 data 中
    final Object[][] data = {
     {"李强", "经理","059568790231",new Integer(5000), new Boolean(false)},
     {"吴虹", "秘书","059569785321",new Integer(3500), new Boolean(true)},
     {"陈卫东", "主管","059565498732",new Integer(4500), new Boolean(false)},
     {"欧阳建", "保安","059562796879",new Integer(2000), new Boolean(true)},
     {"施乐乐", "销售","059563541298",new Integer(4000), new Boolean(false)}
    };
    /* 下面是重写 AbstractTableModel 中的方法,其主要用途是被 JTable 对象调用,
     * 以便在表格中正确地显示出来。
     * 要注意根据采用的数据类型加以恰当实现
     */
    //获得列的数目
      public int getColumnCount()
       {return columnNames.length;}
       //获得行的数目
      public int getRowCount()
      {return data.length;}
     //获得某列的名字,而目前各列的名字保存在字符串数组 columnNames 中
     public String getColumnName(int col)
      {return columnNames[col];}
     //获得某行某列的数据,而数据保存在对象数组 data 中
     public Object getValueAt(int row, int col)
      {return data[row][col];}
     //判断每个单元格的类型
     public Class getColumnClass(int c)
     {return getValueAt(0, c).getClass();}
     //将表格声明为可编辑的
     public boolean isCellEditable(int row, int col)
      {
       if (col < 2) {return false;}
       else {return true;}
      }
       //改变某个数据的值
     public void setValueAt(Object value, int row, int col)
      {
        if (DEBUG)
         {
```

```
72              System.out.println("Setting value at " + row + "," + col
73                 + " to " + value + " (an instance of " + value.getClass() + ")");
74            }
75          if (data[0][col] instanceof Integer && !(value instanceof Integer))
76           {
77            try
78             {
79                data[row][col] = new Integer(value.toString());
80                fireTableCellUpdated(row, col);
81             }
82           catch (NumberFormatException e)            //捕获异常,当程序发生异常时触发
83            {
84                TableDemo table = new TableDemo();
85                JOptionPane.showMessageDialog(table,
86                    "The \"" + getColumnName(col)
87                     + "\" column accepts only integer values.");
88             }
89           } else {
90                       data[row][col] = value;
91                       fireTableCellUpdated(row, col);
92                     }
93          if (DEBUG)
94          {
95             System.out.println("New value of data:");
96             printDebugData();
97          }
98        }
99        private void printDebugData()               //采用双重循环结构,输出二维数组元素
100        {
101          int numRows = getRowCount();
102          int numCols = getColumnCount();
103          for (int i = 0; i < numRows; i++)          //外循环控制行
104          {
105            System.out.print(" row " + i + ":");
106            for (int j = 0; j < numCols; j++)        //内循环控制列
107            {System.out.print(" " + data[i][j]);}
108             System.out.println();
109          }
110          System.out.println("----------------------------");
111        }
112      }
113     //主类
114    public class Example5_18
115    {
116      public static void main(String args[])
117       {
118            TableDemo frame = new TableDemo();
119            frame.pack();
120            frame.setVisible(true);
121       }
122    }
```

程序运行结果如图 5-21 所示。

图 5-21 加强的表格

5.5 菜单与对话框

5.5.1 菜单

菜单是图形用户界面程序设计经常使用的组件。菜单又分为下拉式菜单(JMenu)和弹出式菜单(JPopupMenu)。

一个菜单由多个菜单项组成,选择一个菜单项就可以触发一个动作事件。多个菜单又可以组合成一个新的菜单增加在最顶层框架(JFrame)上,一般的窗口类都要创建菜单栏、多个菜单和一个菜单项。下拉式菜单如图 5-22 所示。

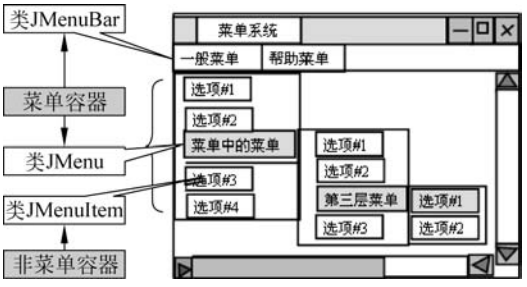

图 5-22 下拉式菜单的组成

(1) 一个菜单栏(JMenuBar)包含多个菜单,通过 JFrame 的 setMenuBar 方法加入一个 JFrame 中。一个菜单栏可以包含任意多个菜单对象,通过 Add 方法来增加菜单对象。

(2) 一个菜单(JMenu)是菜单项的集合,并且有一个标题,这个标题出现在菜单上,当单击这个标题时,这些菜单项立即弹出。使用它自身的 add 方法,可以增加菜单项(JmenuItem)或菜单(JMenu)对象。

(3) 菜单项在菜单中表示一个选项,并且可以注册一个动作监听器(ActionListener),以产生动作事件。

表 5-9 所示为 JMenuBar、JMenu、JMenuItem 的构造函数和常用方法。

建立菜单的步骤如下。

(1) 创建菜单栏对象,并将菜单条对象添加到窗体中。

```
JMenuBar mbar = new JMenuBar();
setJMenuBar(mbar);                                    //窗体类 Frame 的方法
```

(2) 创建菜单对象,并将菜单对象添加到菜单栏中。

```
menu1 = newJMenu("File");
```

```
menu2 = newJMenu("Edit")
mbar.add(menu1);
mbar.add(menu2);
```

表 5-9　与 JMenu 相关的构造函数和常用方法

方　法　名	功　　能
JMenuBar	创建菜单栏
add(JMenu menu)	在菜单栏中添加菜单
JMenu()	创建菜单
JMenu(String label)	创建具有指定标题内容的菜单
add(JMenuItem mi)	在菜单中添加菜单项
addSeparator()	在菜单中添加一条分隔线
insert(JMenuItem mi, int index)	在菜单的指定位置插入菜单项
JMenuItem()	创建菜单项
JMenuItem(String label)	创建具有指定标题内容的菜单项
getLabel()	获取菜单项的标题内容
setLabel(String label)	设置菜单项的标题内容
setEnabled(boolean b)	设置菜单项是否可以选择
addActionListener(ActionListener l)	添加监视器,设置菜单项接收操作事件

(3) 创建菜单项对象,并将菜单项对象添加到相应的菜单中。

```
mi1 = new JMenuItem("New");
mi2 = new JMenuItem("Open");
mi3 = new JMenuItem("Save");
mi4 = new JMenuItem("Close");
menu1.add(mi1);
menu1.add(mi2);
menu2.add(mi3);
menu2.add(mi4);
```

【例 5-19】 设计一个菜单程序。

这个程序包含“文件”和“编辑”菜单。菜单下又包含菜单项。“文件”菜单包含的菜单项为“新建文件”“打开文件”“退出”,“编辑”菜单包含的菜单项为“剪切”“复制”“粘贴”。除了“文件”和“退出”菜单项外,其他的所有的菜单项功能都暂时被关闭。

```
1  /* 演示菜单程序 */
2  import javax.swing.*;
3  import java.awt.event.*;
4  public class  Example5_19 extends JFrame implements ActionListener
5  {  JMenuItem fileNew = new JMenuItem("新建文件");
6     JMenuItem fileOpen = new JMenuItem("打开文件");
7     JMenuItem fileExit = new JMenuItem("退出");
8     JMenuItem editCut = new JMenuItem("剪切");
9     JMenuItem editCopy = new JMenuItem("复制");
10    JMenuItem editPaste = new JMenuItem("粘贴");
11    public  Example5_19()
12   {   super("菜单演示程序");
13       JMenu file = new JMenu("文件");
```

定义菜单项

```
14        file.add(fileNew); fileNew.setEnabled(false);
15        file.add(fileOpen); fileOpen.setEnabled(false);
16        file.addSeparator();
17        file.add(fileExit);  fileExit.setEnabled(true);
18        JMenu edit = new JMenu("编辑");
19        edit.add(editCut);    editCut.setEnabled(false);
20        edit.add(editCopy);   editCopy.setEnabled(false);
21        edit.add(editPaste); editPaste.setEnabled(false);
22        JMenuBar bar = new JMenuBar();
23        setJMenuBar(bar);
24        bar.add(file);
25        bar.add(edit);
26        fileExit.addActionListener(this);
27        setSize(250, 200);
28        setVisible(true);
29        setDefaultCloseOperation(EXIT_ON_CLOSE);
30     }
31     public void actionPerformed(ActionEvent e)
32     {
33        if(e.getSource() == fileExit)
34             System.exit(0);
35     }
36    public static void main(String args[])
37     {   Example5_19 f = new  Example5_19(); }
38  }
```

添加一条分隔线

在“文件”菜单中添加菜单项，并设置“退出”项可选，其余菜单项设为不可选

在“编辑”菜单中添加菜单项，并设置菜单项为不可选

创建菜单栏并设置到该窗体中。将“文件”和“编辑”菜单添加到菜单栏中

向监视器注册“退出”菜单项，若其他菜单项要触发事件，均需要向监视器注册

图 5-23　显示下拉式菜单

程序运行结果如图 5-23 所示。

菜单与按钮类似，两者都要产生动作事件。这两个组件几乎是所有 GUI 程序的标准组件。

5.5.2　弹出式菜单

在窗体中，右击，弹出的菜单称为弹出式菜单，也称快捷菜单。弹出的菜单类 JPopupMenu 的构造方法和常用方法如下。

```
public JpopupMenu()                              //创建弹出式菜单对象
public JpopupMenu(String label)                  //创建带标识的弹出式菜单对象
void add(JMenuItem menuItem)                     //将指定菜单项添加到菜单
void addSeparator()                              //将分隔符添加到菜单
void show(Component invoker, int x, int y)       //在组件 invoker 的坐标 x、y 处显示弹出式菜单
```

【例 5-20】 在文本框中显示弹出式菜单项。

```
1  /* 右键弹出式菜单 */
2  import java.awt.*;
3  import java.awt.event.*;
4  import javax.swing.*;
5  public class Example5_20 extends JFrame implements ActionListener
6  {
7    JPopupMenu popup = new JPopupMenu();
8    JTextField txt = new JTextField(10);
9    public Example5_21()
10   {
11     super("右键弹出式菜单");
12     setSize(300,250);
13     setVisible(true);
```

实例化弹出菜单

```
14      setLayout(new FlowLayout());
15      add(txt);
16      setDefaultCloseOperation(EXIT_ON_CLOSE);
17      JMenuItem m1 = new JMenuItem("菜单项 1");
18      JMenuItem m2 = new JMenuItem("菜单项 2");
19      JMenuItem m3 = new JMenuItem("菜单项 3");
20      JMenuItem m4 = new JMenuItem("菜单项 4");
21      popup.add(m1);
22      popup.add(m2);
23      popup.add(m3);
24      popup.addSeparator();
25      popup.add(m4);
26      m1.addActionListener(this);
27      m2.addActionListener(this);
28      m3.addActionListener(this);
29      m4.addActionListener(this);
30      addMouseListener(new MouseAdapter()
31      {
32        public void mouseClicked(MouseEvent e)
33        {
34          if (e.getButton() == MouseEvent.BUTTON3)
35          {
36              popup.show(e.getComponent(),
37                         e.getX(),
38                         e.getY());
39          }
40        }
41      });
42      validate();
43    }
44    public void actionPerformed(ActionEvent e)
45    {
46      txt.setText(
47        ((JMenuItem)e.getSource()).getText());
48    }
49    public static void main(String args[])
50    {   new Example5_20();     }
51  }
```

实例化菜单项

添加菜单项到菜单上

定义处理鼠标事件的匿名类

在坐标处显示弹出右键菜单

获取鼠标选中的菜单项文本

程序运行结果如图 5-24 所示。

图 5-24　在文本框中显示弹出式菜单项

5.5.3　对话框

对话框(JDialog)是一个有边框、有标题且独立存在的容器，是一个从某个窗口弹出的特殊窗口。对话框与 JFrame 一样，不能被其他容器包容，不能作为程序的最顶层容器，也不能包含菜单。JDialog 必须隶属于一个 JFrame 窗口，并由这个 JFrame 窗口负责弹出。如果它的父窗口 JFrame 消失，它也随之消失。

1. 对话框的构造

一般来说，对话框有两种类型，如下所述。

(1) “有模式”对话框(Medel Dialog)：当这个对话框处于激活状态时，只让程序响应对话框

内部的事件,阻塞隶属父窗口对象的输入,而且它将阻塞其他线程的执行,直到该对话框被关闭。

(2)“无模式”对话框(Non-modal Dialog):这种对话框并不阻塞隶属父窗口对象的输入,可以与父窗口对象并存,除非特别声明,一般的对话框是“无模式”的。

一个对话框类使用如表 5-10 所示的 4 种构造方法进行初始化。

表 5-10 JDialog 类的构造方法及其含义

构 造 函 数	含 义
JDialog(Type parent)	创建以 parent 为父类的“无模式”对话框,parent 可以为 JFrame 或 JDialog
JDialog(Type parent,Boolean modal)	创建以 parent 为父对象对话框,parent 可以为 JFrame 或 JDialog
JDialog(Type parent,String title)	创建以 parent 为父类、title 为标题的“无模式”对话框,parent 可以为 JFrame 或 JDialog
JDialog(Type parent,String title, Boolean modal)	创建以 parent 为父类、title 为标题的对话框,parent 可以为 JFrame 或 JDialog

【例 5-21】 设计一个本对话框与窗口传递数据的程序。

```
1  /* 本示例说明对话框与窗口传递数据 */
2  import java.awt.*;
3  import java.awt.event.*;
4  import javax.swing.*;
5  class Win extends JFrame  implements ActionListener
6  { JButton btn1 = new JButton("打开对话框");
7    JTextArea txt = new JTextArea(5, 8);
8    Win()
9    {  super("对话框与窗体传递消息");
10      setBounds(50, 50, 200, 200);
11      setVisible(true);
12      addWindowListener(new WindowAdapter(){
13        public void windowClosing (WindowEvent e){
14          System.exit(0);
15        }
16      } );
17   setLayout(new BorderLayout());
18   add(btn1, "North"); add(txt, "Center");
19   btn1.addActionListener(this);
20   validate();
21  }
22    public void actionPerformed(ActionEvent e){
23    Dia dia = new Dia(this, "传递消息对话框", true);
24    dia.setVisible(true);
25    txt.append(dia.getMessage());
26  }
27 }
28  //构造对话框类
29  class Dia extends JDialog implements ActionListener
30  {
31     JTextField txt = new JTextField(10);
32     Dia(JFrame f, String s, boolean b){
33       super(f, s, b);
34       setSize(300, 100);
35       setLayout(new FlowLayout());
```

构造窗体类

关闭窗体的另一种方法,构造窗体适配器匿名类

由于有模式的对话框,这时将产生阻塞,直到对话框关闭。窗体获取对话框传递的消息,放到文本区中

构造对话框,所依赖的窗体为f,对话框的标题为s,是否为有模模式由参数b决定

```
36        add(txt);
37        txt.addActionListener(this);
38        validate();
39     }
40    public void actionPerformed(ActionEvent e){
41        setVisible(false);
42     }
43      //把对话框的消息传递出去
44   public String getMessage(){
45        return txt.getText();
46     }
47   }
48   //主类
49    public class Example5_21{
50      public static void main(String args[]) {
51        new Win();
52     }
53   }
```

程序运行结果如图 5-25 所示。

(a) 主窗体　(b) 对话框　(c) 从对话框传递来的消息

图 5-25　对话框传递消息给窗体

2. 消息对话框

Java 有一种与用户交互操作的特殊消息对话框 JOptionPane 类，其基本外形通常如图 5-26 所示。

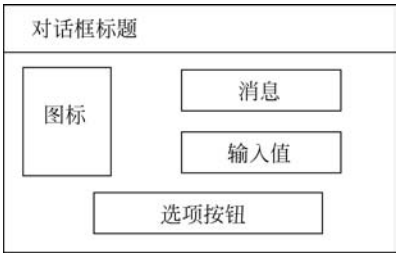

图 5-26　消息对话框基本外形

1）消息对话框的构造方法

消息对话框的构造方法因其参数不同，所表现的形式有所不同，其常用构造方法如表 5-11 所示。

表 5-11　消息对话框的常用构造方法及其含义

构 造 方 法	说　明
JOptionPane()	创建一个带有测试消息的 JOptionPane 对话框
JOptionPane(Object message)	创建一个显示消息的 JOptionPane 对话框，提供默认选项
JOptionPane(Object message, int messageType)	创建一个显示消息的 JOptionPane 对话框，使其具有指定的消息类型和默认选项
JOptionPane(Object message, int messageType, int optionType)	创建一个显示消息的 JOptionPane 对话框，使其具有指定的消息类型和选项
JOptionPane(Object message, int messageType, int optionType, Icon icon)	创建一个显示消息的 JOptionPane 对话框，使其具有指定的消息类型、选项和图标

其中：

messageType 定义消息对话框的样式。对话框的外观布置因此值而异，并提供默认图标。messageType 取值为 ERROR_MESSAGE、INFORMATION_MESSAGE、WARNING_MESSAGE、QUESTION_MESSAGE、PLAIN_MESSAGE。

optionType 定义在对话框的底部显示的选项按钮的集合为 DEFAULT_OPTION、YES_NO_OPTION、YES_NO_CANCEL_OPTION、OK_CANCEL_OPTION。

2) 消息对话框的静态方法

JOptionPane 类通常调用静态方法 showXxxxDialog()来确定对话框的显示类型，其方法如表 5-12 所示。

表 5-12 showXxxxDialog()显示对话框的类型

方 法 名	说 明
showConfirmDialog()	确认对话框：询问一个确认问题，如 yes/no/cancel
showInputDialog()	输入对话框：包括一个文本字段和两个按钮，确定和取消
showMessageDialog()	消息对话框：告知用户某事已发生
showOptionDialog()	自定义格式对话框

例如，显示消息对话框类型，如图 5-27 所示。

```
JOptionPane.showMessageDialog(null, "提示的内容", "提示对话框",JOptionPane.ERROR_MESSAGE);
```

显示确认对话框类型，如图 5-28 所示。

```
JOptionPane.showConfirmDialog(null, "选择的内容", "选择对话框",JOptionPane.YES_NO_CANCEL_
OPTION);
```

图 5-27 消息对话框

图 5-28 确认对话框

【例 5-22】 消息对话框示例。

```
1  import javax.swing.*;
2  public class Example5_22
3  {
4    public static void main(String args[])
5    {
6      JOptionPane  d_input = new JOptionPane();
7      String str = d_input.showInputDialog(null, "1 + 2 = ?");
8      if (str.equals("3"))
9          d_input.showMessageDialog(null, "回答正确。");
10     else
11         d_input.showMessageDialog(null, "回答错误!");
12     d_input.showConfirmDialog(null, "测试完毕!");
13     System.exit(0);                          //退出程序;
14   }
15 }
```

程序运行第 7 行输入对话框语句时，显示的结果如图 5-29 所示。

3. 文件选择对话框

实例化 JFileChooser 类后，调用 showOpenDialog 方法，能够打开一个文件选择对话框，可用于打开或保存文件时的文件选择，如图 5-30 所示。

```
JFileChooser file = new JFileChooser();
file.showOpenDialog(null);
```

图 5-29　“输入”对话框

图 5-30　文件选择对话框

【例 5-23】 文件选择对话框示例。

```
1  import javax.swing.*;
2  import java.awt.*;
3
4  public class Example5_23
5   {
6    public static void main(String args[])
7     {
8        JOptionPane  d_input = new JOptionPane();
9        JFileChooser file = new JFileChooser();
10       file.showOpenDialog(null);
11       String str = file.getSelectedFile().toString();
12       d_input.showMessageDialog(null, str);
13    }
14 }
```

获取选择的文件名

运行程序后，显示图 5-30 中文件选择对话框，选择要打开的文件，返回所选择的文件路径及文件名，如图 5-31 所示。

图 5-31　显示返回所选择的文件路径及文件名

4. 颜色选择对话框

实例化 JColorChooser 类后，调用 showOpenDialog 方法，能够打开一个颜色选择对话框，如图 5-32 所示。

```
JColorChooser  color = new JColorChooser();
color.showDialog(null,"",null);
```

【例 5-24】 颜色选择对话框示例。

```
1  import javax.swing. * ;
2  import java.awt. * ;
3
4  public class Example5_24
5  {
6    public static void main(String args[])
7     {
8        JOptionPane  d_input = new JOptionPane();
9        JColorChooser  color = new JColorChooser() ;
10       Color c = color.showDialog(null,"",null);      ← 返回所选择颜色的值
11       d_input.showMessageDialog(null, c);
12    }
13 }
```

运行程序后,显示图 5-32 中的颜色选择对话框,选择颜色后,返回所选择颜色的值,如图 5-33 所示。

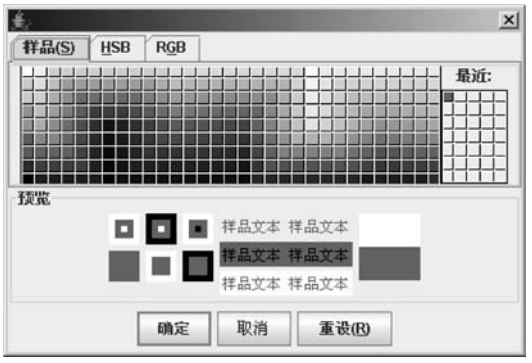

图 5-32 颜色选择对话框

图 5-33 显示返回所选颜色的值

5.6 树

5.6.1 树的概念

在 Microsoft Windows 文件管理器中,目录及文件都是以树状的层次结构显示出来的,就如同阶梯一般,一层包着一层,这样的做法不仅可以让用户清楚地了解各节点之间的关系,也使得用户在找寻相关的文件时更为方便。

对于树的结构,最上层的点称为根节点(root node),在根节点下面的节点称为子节点(child node),子节点下面还可以有子节点,因此会形成所谓父节点与子节点的关系。每个节点可以有零个到多个的子节点。当一个节点没有任何的子节点时,就称这个节点为树叶节点(leaf node),反之称为树枝节点(internal node),其关系如图 5-34 所示。

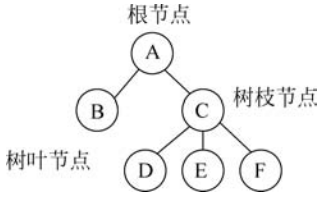

图 5-34 树的结构

节点 A 为根节点,节点 B、C、D、E、F 为 A 的子节点。节点 C 是节点 D、E、F 的父节点,节点 D、E、F 是节点 C 的子节点。

节点 B、D、E、F 为树叶节点，节点 A、C 为树枝节点。

在 Java 中，JTree 是建立树结构的类。初始状态的树状视图，在默认情形下，只显示根节点和它的直接子节点。用户可以双击分节点的图标或单击图标前的“开关”使该节点展开或收缩。

5.6.2　树的构造方法

1. 树的创建

可以使用 JTree 类的构造方法创建树。JTree 的常用构造方法如下。

- JTree()：建立一个系统默认的树。
- JTree(Hashtable value)：应用 Hashtable 表建立树，不显示根节点。
- JTree(Object[] value)：应用数组 Object[]建立树，不显示根节点。
- JTree(TreeNode root)：应用节点 TreeNode 建立树。
- JTree(TreeNode root, boolean askAllowsChildren)：应用节点 TreeNode 建立树，并确定是否允许有子节点。

【例 5-25】 建立一个系统默认的简单树结构。

```
1  /* 最简单的 JTree 示例 */
2  import java.awt.*;
3  import javax.swing.*;
4  class TreeDemo extends JFrame
5  {
6    public TreeDemo()
7    {
8     setSize(400,300);
9     setTitle("演示怎样使用 JTree");
10    show();
11    JScrollPane jPanel = new JScrollPane();
12    getContentPane().add(jPanel);
13    //创建系统默认的树状对象
14    JTree jtree = new JTree();
15    //在面板上添加树状结构
16    jPanel.getViewport().add(jtree, null);
17    validate();
18    //设置"关闭窗口"按钮
19    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
20    }
21 }
22  //主类
23 public class Example5_25
24 {
25   public static void main(String args[])
26    {
27         TreeDemo frame = new TreeDemo();
28    }
29 }
```

【程序说明】

本程序非常简单,利用 JTree()构造了一个系统默认的树对象,并将些对象放在带滚动条的面板 JScrollPane 中,当树结构大于窗体空间时,可以利用滚动条来滚动面板。

程序运行结果如图 5-35 所示。

图 5-35 系统默认的简单树结构

2. 树节点的创建

例 5-25 虽然简单,但并没有多少实质意义,因为各个节点的数据都是 Java 的默认值,而非用户定义的数据。树结构的类是 JTree,要构造一个由用户定义枝节点的树,JTree 必须同树枝节点类 TreePath 和 TreeNode 共同来完成。

树节点由 javax.swing.tree 包中的接口 TreeNode 定义,该接口是 MutableTreeNode 类的子类,而 MutableTreeNode 又由 DefaultMutableTreeNode 类实现,因此,在创建树时,要使用 DefaultMutableTreeNode 类为该树创建节点。

DefaultMutableTreeNode 类的常用构造方法是:

```
DefaultMutableTreeNode(Object userObject);
DefaultMutableTreeNode(Object userObject, boolean allowChildren);
```

第一个构造方法创建的节点默认可以有子节点,即它可以用 add 方法添加其他节点作为它的子节点。

对于一个节点,可以使用方法 setAllowsChildren(boolean b)来设置是否允许有子节点。

应用节点 TreeNode 构造树的步骤如下:

(1) 定义节点。

```
DefaultMutableTreeNode n1 = new DefaultMutableTreeNode("节点 1");
DefaultMutableTreeNode n2 = new DefaultMutableTreeNode("节点 2");
DefaultMutableTreeNode n3 = new DefaultMutableTreeNode("节点 3");
…
```

(2) 定义树,同时确定 n1 为根节点。

```
JTree tree = new JTree(n1);
```

(3) 添加子节点。

```
n1.dd(n2);
n1.add(n3);
```

【例 5-26】 建立一个应用 TreeNode 构造如图 5-36 所示结构的树。

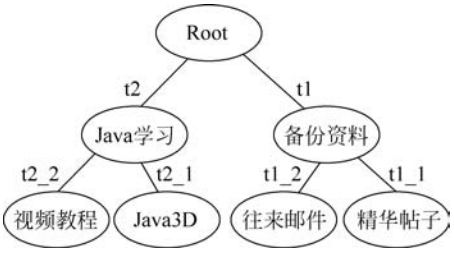

图 5-36 TreeNode 树结构

```
/* "利用 TreeNode 构造树" */
import javax.swing.*;
import javax.swing.tree.*;
import java.awt.*;
import java.awt.event.*;
class Mytree extends JFrame
{
 Mytree(String s)
  {
    super(s);
    Container con = getContentPane();        //定义 JFrame 窗体容器
    //定义节点
  DefaultMutableTreeNode root = new DefaultMutableTreeNode("c:\\");
  DefaultMutableTreeNode t1 = new DefaultMutableTreeNode("备份资料");
  DefaultMutableTreeNode t2 = new DefaultMutableTreeNode("Java 学习");
  DefaultMutableTreeNode t1_1 = new DefaultMutableTreeNode(
                                          "思维论坛精华帖子");
  DefaultMutableTreeNode t1_2 = new DefaultMutableTreeNode("来往邮件");
  DefaultMutableTreeNode t2_1 = new DefaultMutableTreeNode("视频教程");
  DefaultMutableTreeNode t2_2 = new DefaultMutableTreeNode("Java3D");
  //创建根节点为 root 的树
  JTree tree = new JTree(root);
  //定义 t1、t2 为 root 的子节点
  root.add(t1);
  root.add(t2);
  //定义 t1_1,t1_2 为 t1 的子节点
  t1.add(t1_1);   t1.add(t1_2);
  //定义 t2_1,t2_2 为 t2 的子节点
  t2.add(t2_1);
  t2.add(t2_2);
  JScrollPane scrollpane = new JScrollPane(tree);      //带滚动条的面板(树放置其中)
  con.add(scrollpane);
  setSize(300,200);
  setVisible(true);
  validate();
  setDefaultCloseOperation(EXIT_ON_CLOSE);
   }
 }
//主类
public class Example5_26
{
  public static void main(String[] args)
   {
       new Mytree("利用 TreeNode 构造树");
   }
 }
```

程序运行结果如图 5-37 所示。

图 5-37　利用 TreeNode 构造树

3. 利用哈希表构造树

在下面的例子中，介绍利用哈希表(Hash table)的数据来建立树结构。

【例 5-27】 建立一个利用哈希表数据构造树的结构。

```
1  /* 利用哈希表定义树结构 */
2   import java.awt.*;
3   import java.awt.event.*;
4   import javax.swing.*;
5   import java.util.*;
6  //定义树结构类
7  class TreesDemo //extends JPanel
8   {
9    public TreesDemo()
10    {
11     JFrame f = new JFrame("哈希表定义树结构演示");
12      Hashtable hashtable1 = new Hashtable();
13      Hashtable hashtable2 = new Hashtable();
14      String[] s1 = {"思维论坛","Java 爱好者","网上书店"};
15      String[] s2 = {"公司文件","私人文件","往来信件"};
16      String[] s3 = {"本机磁盘(C:)","本机磁盘(D:)","本机磁盘(E:)"};
17      hashtable1.put("桌面",hashtable2);
18      hashtable2.put("收藏夹",s1);
19      hashtable2.put("我的公文包",s2);
20      hashtable2.put("我的电脑",s3);
21      //树进行初始化,其数据来源是 root 对象
22      JTree tree = new JTree(hashtable1);
23      JScrollPane scroll = new JScrollPane();
24      scroll.setViewportView(tree);
25      Container con = f.getContentPane();
26      con.add(scroll);
27      f.setSize(200,300);
28      f.setVisible(true);
29      f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
30     }
31 }
32 //主类
33 public class Example5_27 {
34   public static void main(String args[])
35     { TreesDemo jf = new TreesDemo(); }
36 }
```

图 5-38 利用哈希表数据建立树结构

【程序说明】

哈希表是常用的数据结构,它是利用 key-value 对的方式来存储数据,也就是说当要在哈希表中找寻数据时,必须先提供关键字 key 的值,哈希表会根据 key 值找到所关联的数据项值 value。哈希表是利用 put 方法将 key-value 对放入哈希表中。程序第 17 行中,“收藏夹”就是一个 key 值,而字符串数组 s1 的数据“思维论坛”“Java 爱好者”“网上书店”是“收藏夹”对应的数据项 value 值。

程序运行结果如图 5-38 所示。

4. 处理节点事件

树中的节点可以发生选择事件,即单击节点时产生事

件。一个树对象处理事件的接口是 TreeSelectionListener，可以使用：

```
树对象.addTreeSelectionListener(this);
```

方法获得一个监视器。

树对象通过使用方法 getLastSelectedPathComponent 获取选中的节点，使用方法 getUserObject 得到与节点相关的信息。

【例 5-28】 处理节点事件。

```
/* 处理节点事件 */
 import java.awt.*;
 import java.awt.event.*;
 import javax.swing.*;
 import javax.swing.tree.*;
 import javax.swing.event.*;
 //定义树结构类
class TreesDemo extends JFrame implements TreeSelectionListener
{
 JTree tree = null;
 JTextArea text = new JTextArea(20,20);
 public TreesDemo()
  {
   super("处理节点事件");
   Container con = getContentPane();
  String[][] data = {
     {"我的电脑","本机磁盘(C:)","本机磁盘(D:)","本机磁盘(E:)"},
     {"收藏夹","思维论坛","Java 爱好者","网上书店"},
     {"我的公文包","公司文件","私人文件","往来信件"},
   };
    DefaultMutableTreeNode  root;                    //定义根节点
    DefaultMutableTreeNode  treeNode[][];            //定义节点数组
    //建立根节点对象
    root = new DefaultMutableTreeNode("桌面");
     //声明节点数组容量
    treeNode = new DefaultMutableTreeNode[4][4];
    //外循环建立父节点,内循环建立子节点
   for (int i = 0;i < data.length;i++)
    {
      //建立父节点 treeNode
      treeNode[i][0] = new DefaultMutableTreeNode(data[i][0]);
      root.add(treeNode[i][0]);
      for(int j = 1;j < 4;j++)
      { //给节点 treeNode 添加多个子节点
        treeNode[i][0].add(new DefaultMutableTreeNode(data[i][j]));
      }
   }
    //创建根为 root 的树对象 tree
  tree = new JTree(root);
  JScrollPane scrollpane = new JScrollPane(text);    //创建带滚动条的面板
  JSplitPane splitpane = new JSplitPane(JSplitPane.HORIZONTAL_SPLIT,
                  true,tree,scrollpane);             //创建可拆分的窗体
  con.add(splitpane);
  tree.addTreeSelectionListener(this);
  setSize(500,200);
  setVisible(true);
```

```
47   validate();
48   setDefaultCloseOperation(EXIT_ON_CLOSE);
49  }
50  //定义节点事件
51  public void valueChanged(TreeSelectionEvent e)
52  {
53   if(e.getSource() == tree)
54     {
55     //定义被选中的节点
56      DefaultMutableTreeNode node =
57      (DefaultMutableTreeNode)tree.getLastSelectedPathComponent();
58      if(node.isLeaf()){
59        //获取叶节点所定义的文本信息
60        String str = node.toString();
61        if(str.equals("本机磁盘(C:)"))
62             {  text.setText(str + ":\n 这里显示'C:盘文件'");}
63        else if(str.equals("本机磁盘(D:)"))
64             {  text.setText(str + ":\n 这里显示'D:盘文件'");}
65         else if(str.equals("思维论坛"))
66            {  text.setText(str + ":\n 这里显示'www.zsm8.com 的精华帖子'");}
67         else if(str.equals("Java 爱好者"))
68            {  text.setText(str + ":\n 这里显示'Java 爱好者网址'");}
69         else if(str.equals("网上书店"))
70            {  text.setText(str + ":\n 这里显示'网上书店的购物信息'");}
71         else if(str.equals("公司文件"))
72            {  text.setText(str + ":\n 这里显示'公司内部文件'"); }
73       }
74      else {
75          text.setText(node.getUserObject().toString());
76         }
77      }
78    }   //valueChanged()_end
79  }
80   //主类
81   public class Example5_28
82   {
83     public static void main(String args[])
84     {
85        new TreesDemo();
86     }
87  }
```

【程序说明】

(1) 程序的第 21 行和第 22 行定义根节点和节点数组；第 24 行创建根节点对象；第 28 行～第 37 行,用双重循环建立节点(treeNode[i][0]为父节点,treeNode[i][j]为子节点),其中外循环建立父节点,内循环建立子节点,并把子节点添加到对应的父节点中。

(2) 程序的第 39 行创建根为 root 的树对象 tree；第 40 行建立一个带滚动条的面板(文本区放置到其上)；在第 41 行,创建一个水平分割的可拆分窗体,tree 在左边,带滚动条的面板在右边。

(3) 第 51 行定义节点事件,当单击树的节点时将触发这些事件。

(4) 第 56 行定义被选中的节点；第 58～73 行为获取叶节点所定义的文本信息；第 75 行为获取节点标题内容。

程序运行结果如图 5-39 所示。

图 5-39　处理节点事件

实验 5

【实验目的】

(1) 掌握窗体的创建以及几种常用方法。

(2) 掌握按钮的创建以及动作监听。

(3) 掌握面板、文本组件、选择框的应用。

(4) 熟练掌握布局的应用。

(5) 熟悉掌握应用类 Canvas 创建画布对象。

(6) 掌握菜单、菜单项的创建以及使用菜单的技巧。

(7) 掌握树结构的应用。

【实验内容】

(1) 运行下列程序,并写出其输出结果。

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class Ex5_1 extends JFrame implements ActionListener
{  JTextField text;
   JButton buttonEnter,buttonQuit;
   JLabel str;
   Ex5_1()
   {
      setBounds(100,100,300,200);
      setVisible(true);
      setLayout(new FlowLayout());
      setDefaultCloseOperation(EXIT_ON_CLOSE);
      str = new JLabel("在文本框输入数字字符按 Enter 键或单击按钮");
      add(str);
      text = new JTextField("0",10);
      add(text);
      buttonEnter = new JButton("确定");
      buttonQuit  = new JButton("清除");
      add(buttonEnter); add(buttonQuit);
      validate();
      buttonEnter.addActionListener(this);
      buttonQuit.addActionListener(this);
      text.addActionListener(this);
   }
```

```
    public void actionPerformed(ActionEvent e)
    {   if(e.getSource() == buttonEnter||e.getSource() == text)
          {   double number = 0;
              try {   number = Double.valueOf(text.getText()).doubleValue();
                      text.setText("" + Math.sqrt(number));
                  }
              catch(NumberFormatException event)
                  {   text.setText("请输入数字字符");
                  }
          }
        else if(e.getSource() == buttonQuit)
         {   text.setText("0");
         }
    }
    public static void main(String args[])
     {     new Ex5_1();}
}
```

(2) 运行下列程序,分析其输出结果。

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class Ex5_2 extends JFrame implements ActionListener
{   JTextField text1,text2;
    JPasswordField passtext;
    Ex5_2()
    {
        setSize(300,200);
        setVisible(true);
        setLayout(new FlowLayout());
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        text1 = new JTextField("输入密码:",10);
        text1.setEditable(false);
        passtext = new JPasswordField(10);
        passtext.setEchoChar('*');
        text2 = new JTextField("我是一个文本框",20);
        add(text1);add(passtext);add(text2);

        validate();
        passtext.addActionListener(this);
    }
    public void actionPerformed(ActionEvent e)
    {
        text2.setText("设置的密码为:   " + passtext.getText());
    }
    public static void main(String args[])
     {
         new Ex5_2();
     }
}
```

(3) 运行下列程序,画出其输出界面。

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

```
import java.awt.event.*;
class MyFrame extends JFrame implements ItemListener,ActionListener
{  JCheckBox box; JTextArea text; JButton button;
   MyFrame(String s)
   {  super(s);
      box = new JCheckBox("设置窗口是否可调整大小");
      text = new JTextArea(12,12);
      button = new JButton("关闭窗口");
      button.addActionListener(this);
      box.addItemListener(this);
      setSize(300,200);
      setVisible(true);
      add(text,BorderLayout.CENTER);
      add(box,BorderLayout.SOUTH);
      add(button,BorderLayout.NORTH);
      setResizable(false);
      validate();
   }
   public void itemStateChanged(ItemEvent e)
   {
     if(box.isSelected() == true) setResizable(true);
     else setResizable(false);
   }
   public void actionPerformed(ActionEvent e)
   {  dispose();   }
}
class Ex5_3
{  public static void main(String args[])
   {  new MyFrame("窗口");   }
}
```

(4) 运行下列程序,并写出其输出结果。

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
class Mypanel extends JPanel implements ActionListener
{
   JButton button1,button2,button3;
   Color backColor;
   Mypanel()                          //构造方法。当创建面板对象时,面板被初始化为有 3 个按钮
   {  button1 = new JButton("确定");
      button2 = new JButton("取消");
      button3 = new JButton("保存");
      add(button1);add(button2);add(button3);
      setBackground(Color.pink);                    //设置面板的底色
      backColor = getBackground();                  //获取底色
      button1.addActionListener(this);button2.addActionListener(this);
   }
   public void actionPerformed(ActionEvent e)
   {  if(e.getSource() == button1)
          setBackground(Color.cyan);
      else if(e.getSource() == button2)
          setBackground(backColor);
   }
}
```

```
public class Ex5_4 extends JFrame
{  Mypanel panel1,panel2,panel3;
   JButton button;
   Ex5_4()
   {
      setSize(300,200);
      setVisible(true);
      setDefaultCloseOperation(EXIT_ON_CLOSE);
      setLayout(new FlowLayout());
      panel1 = new Mypanel();panel2 = new Mypanel();panel3 = new Mypanel();
      button = new JButton("我不在那些面板里");
      add(panel1);add(panel2);add(panel3);
      add(button);
      validate();
   }
    public static void main(String args[])
    {
       Ex5_4 win = new Ex5_4();
    }
}
```

(5) 运行下列程序,并写出其输出结果。

```
import javax.swing.*;
import java.awt.*;
class Mycanvas extends Canvas
{
  String s;
   Mycanvas(String s)
   {  this.s = s;
      setSize(90,80);
      setBackground(Color.cyan);
   }
   public void paint(Graphics g)
   {  if(s.equals("circle"))
         g.drawOval(20,25,30,30);
      else if(s.equals("rect"))
         g.drawRect(30,35,20,20);
   }
}

public class Ex5_5 extends JFrame
{
   Mycanvas canvas1,canvas2;
   Ex5_5()
   {
      setSize(300,200);
      setVisible(true);
      setDefaultCloseOperation(EXIT_ON_CLOSE);
      setLayout(new FlowLayout());
      canvas1 = new Mycanvas("circle");
      canvas2 = new Mycanvas("rect");
      add(canvas1);
      JPanel p = new JPanel();
      p.setBackground(Color.pink);
```

```
        p.add(canvas2);
        add(p);
        validate();
    }
     public static void main(String args[])
     {
       new Ex5_5();
      }
}
```

(6) 运行下列的程序,并画出其输出的结果。

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class Ex5_6 extends JFrame
{
  Ex5_6()
  {
       setSize(500,300);
       setVisible(true);
       setDefaultCloseOperation(EXIT_ON_CLOSE);
       setLayout(new GridLayout(12,12));
       JButton button[][] = new JButton[12][12];
       for(int i = 0;i < 12;i++)
         {  for(int j = 0;j < 12;j++)
              {  button[i][j] = new JButton();
                 if((i + j) % 2 == 0)
                   button[i][j].setBackground(Color.black);
                 else
                   button[i][j].setBackground(Color.white);
                 add(button[i][j]);
              }
          }
      validate();
  }
  public static void main(String args[])
   {
    new Ex5_6();
   }
}
```

(7) 运行下列程序,并画出其输出界面。

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
class Herwindow extends JFrame  implements ActionListener
{  JMenuBar menubar;JMenu menu;JMenuItem item;
   Herwindow(String s)
   {  super(s);
      setSize(160,170);
      setVisible(true);
      menubar = new JMenuBar();
      menu = new JMenu("文件");
      item = new JMenuItem("退出");
```

```
        item.addActionListener(this);
        menu.add(item);
        menubar.add(menu);menubar.add(menu);
        setJMenuBar(menubar);
        validate();
    }
    public void actionPerformed(ActionEvent e)
    {   if(e.getSource() == item)
         {   System.exit(0);
         }
    }
}

public class Ex5_7
{   public static void main(String args[])
    {   Herwindow window = new Herwindow("法制之窗");
    }
}
```

(8) 运行下列程序,并写出其输出结果。

```
import javax.swing.*;
import javax.swing.tree.*;
import java.awt.*;
public class Ex5_8 extends JFrame
{
  Ex5_8()
  {
    setSize(300,200);
    setVisible(true);
    setDefaultCloseOperation(EXIT_ON_CLOSE);
    Container con = getContentPane();
    DefaultMutableTreeNode root = new DefaultMutableTreeNode("c:\\");   //树的根节点
    DefaultMutableTreeNode t1 = new DefaultMutableTreeNode("dos");      //节点
    DefaultMutableTreeNode t2 = new DefaultMutableTreeNode("java");     //节点
    DefaultMutableTreeNode t1_1 = new DefaultMutableTreeNode("wps");
    DefaultMutableTreeNode t1_2 = new DefaultMutableTreeNode("epg");
    DefaultMutableTreeNode t2_1 = new DefaultMutableTreeNode("applet");
    DefaultMutableTreeNode t2_2 = new DefaultMutableTreeNode("jre");
    root.add(t1);root.add(t2);
    t1.add(t1_1);t1.add(t1_2);
    t2.add(t2_1);t2.add(t2_2);
    JTree tree  = new JTree(root);                                      //创建根为 root 的树
    DefaultTreeCellRenderer render = new  DefaultTreeCellRenderer();
    render.setLeafIcon(new ImageIcon("leaf.gif"));
    render.setBackground(Color.yellow);
    render.setClosedIcon(new ImageIcon("close.gif"));
    render.setOpenIcon(new ImageIcon("open.gif"));
    render.setTextSelectionColor(Color.red);
    render.setTextNonSelectionColor(Color.green);
    render.setFont(new Font("TimeRoman",Font.BOLD,16));
    tree.setCellRenderer(render);
    JScrollPane scrollpane = new JScrollPane(tree);
    con.add(scrollpane);
    validate();
  }
```

```
    public static void main(String args[])
     {
       new Ex5_8();
     }
}
```

习题 5

1. 什么是图形用户界面？列举所用的应用程序中使用的组件。

2. 创建一个窗体，窗体中有一个按钮，当单击按钮后，就会弹出一个新窗体。

3. 什么是容器的布局？试列举并简述 Java 中常用的几种布局策略。

4. 设计一个如图 5-40 所示的加法计算器，在文本框中输入两个整数，单击“＝”按钮时，在第三个文本框中显示这两个数的和。

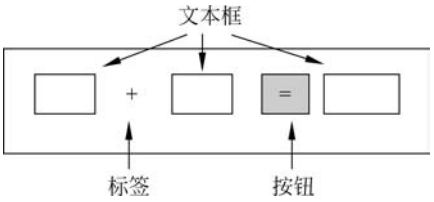

图 5-40　加法计算器

5. 说明文本框与标签之间的区别。

6. 编写一个程序，其中包含一个标签、一个文本框和一个按钮。当用户单击按钮时，程序把文本框中的内容复制到标签中。

7. 设计一个类似 Windows 系统的计算器，其中要使用按钮、文本框、布局管理、标签等构件，能实现多位数的加、减、乘、除运算功能。

8. 编写图形界面的应用程序，该程序包含一个菜单，选择这个菜单的“退出”选项可以关闭窗口并结束程序。

9. 设计一个模拟的文字编辑器，并用菜单实现退出的功能。

10. 将通讯录内容显示到一个表格中。

11. 改进例 5-27，编写一个能动态改变树节点的程序。