第13讲 高级组件GUI设计

BACK

11.1 界面布局管理

11.1.1 FlowLayout

java.FlowLayout类是java.lang.Object类的直接子类。

FlowLayout的布局策略是将遵循这种布局策略的容器中的组件按照加入的先后顺序从左向右排列，当一行排满之后就转到下一行继续从左至右排列，每一行中的组件都居中排列。

FlowLayout是Applet缺省使用的布局编辑策略。

FlowLayout类有三个构造方法，分别是：

(1) FlowLayout()：用于创建一个版面设定为居中对齐、各组件的水平及垂直间隔为5个像素点的FlowLayout类的对象。

(2) FlowLayout(int align)：用于创建一个FlowLayout类的对象，版面按给出的align值对齐，各组件的水平及垂直间隔为5个像素。align的值可以是FlowLayout.LEFT(左对齐)、FlowLayout.RIGHT(右对齐)、FlowLayout.CENTER(居中对齐)方式。

(3) FlowLayout(int align，int hgap，int vgap)：用于创建一个既指定对齐方式，又指定组件间间隔的FlowLayout类的对象。参数align的作用及取值同上；参数hgap指定组件间的水平间隔；参数vgap指定各组件间的垂直间隔。间隔单位为像素点。

对于一个原本不使用FlowLayout布局编辑器的容器，若需要将其布局策略改为FlowLayout，可以使用setLayout(new FlowLayout( ))方法，该方法是所有容器的父类Container的方法，用于为容器设定布局编辑器。

11.1.2 BorderLayout

java.BorderLayout类是java.lang.Object类的直接子类。BorderLayout布局策略是把容器内的空间划分为东、西、南、北和中五个区域。这五个区域分别用字符串常量East、West、South、North和Center表示。向这个容器内每加入一个组件都应该指明把它放在容器的哪个区域中。分布在北部和南部区域的组件将横向扩展至占据整个容器的长度；分布在东部和西部的组件将伸展至占据容器剩余部分的全部宽度；最后剩余的部分将分配给位于中央的组件。如果某个区域没有分配组件，则其他组件可以占据它的空间。

BorderLayout类有两个构造方法，分别是无参数的BorderLayout( )和带参数的BorderLayout(int hgap, int vgap)。前者创建一个各组件间的水平间隔、垂直间隔都为0的BorderLayout类的对象；后者创建一个各组件间的水平间隔为hgap、垂直间隔为vgap的BorderLayout类的对象。

【示例程序c11_1.java】 使用BorderLayout布局策略在五个位置分别加入了四个按钮和一个标签，当点击按钮时，标签的文本就是按钮的标签的文本。

```
import javax.swing.*;

import java.awt.*;

import java.awt.event.*;

public class c11_1 extends JApplet implements
ActionListener

{   JButton bt1=new JButton("北部"),
```

```
            bt2=new JButton("西部"),
            bt3=new JButton("东部"),
            bt4=new JButton("南部");
JLabel lb1=new JLabel("中部");
Container cp=getContentPane( );
public void init( )
{ // 设置BorderLayout 布局，组件间隔为11
  cp.setLayout(new BorderLayout(11,11));
  cp.add("North",bt1);  //将bt1放置于北区
  bt1.addActionListener(this);
```

```
    cp.add("West",bt2); //将bt2放置于西区
    bt2.addActionListener(this);
    cp.add("East",bt3); //将bt3放置于东区
    bt3.addActionListener(this);
    cp.add("South",bt4); //将bt4放置于南区
    bt4.addActionListener(this);
    cp.add("Center",lb1); //将bt5放置于中区
}
```

```
public void actionPerformed(ActionEvent e)
  {
   if (e.getSource( )==bt1)  lb1.setText("按钮1");
   else if (e.getSource( )==bt2) lb1.setText("按钮2");
        else if (e.getSource( )==bt3)  lb1.setText("按钮3")
            else  lb1.setText("按钮4");
   }
}
```

图11.1 程序c11_1的运行结果

11.1.3 CardLayout

CardLayout的版面布局方式是将每个组件看成一张卡片，如同扑克牌一样将组件堆叠起来，而显示在屏幕上的每次只能是最上面的一个组件，这个被显示的组件将占据所有的容器空间。用户可通过表11.1所示的方法选择使用其中的卡片。

CardLayout类有两个构造方法，分别是CardLayout( )和CardLayout(int hgap,int vgap)。前者使用默认(间隔为0)方式创建一个CardLayout( )类对象；后者使用由hgap指定的水平间隔和由vgap指定的垂直间隔创建一个CardLayout( )类对象。

CardLayout类的常用成员方法列于表11.1中。

表11.1 CardLayout类的常用成员方法

成 员 方 法	说 明
first(Container container)	显示 container 中的第一个对象
last(Container container)	显示 container 中的最后一个对象
next(Container container)	显示下一个对象
previous(Container container)	显示上一个对象

【示例程序c11_2.java】 使用CardLayout的布局策略，在容器中放入三个按钮，显示第二个按钮。

```
import java.awt.*;
import javax.swing.*;
public class c11_2 extends JApplet
{   JButton bt1=new JButton("按钮A");
    JButton bt2=new JButton("按钮B");
    JButton bt3=new JButton("按钮C");
    Container cp=getContentPane( );
    //设置CardLayout布局
```

```
    CardLayout card=new CardLayout(20,20);
  public void init( )
  { cp.setLayout(card);
    cp.add("a",bt1);
    cp.add("b",bt2);    cp.add("c",bt3);
    card.next(cp); //显示按钮B
   }
}
```

图11.2 程序c11_2的运行结果

11.1.4 GridLayout

如果界面上需要放置的组件较多，且这些组件的大小又基本一致时，例如计算器、遥控器的面板，使用GridLayout布局策略是最佳的选择。GridLayout的布局策略是把容器的空间划分为若干行、若干列的网格区域，而每个组件按添加的顺序从左向右、从上向下地占据这些网格。

GridLayout类的三个构造方法如下:

(1) GridLayout( ): 按默认(1行1列)方式创建一个GridLayout布局。

(2) GridLayout(int rows,int cols): 创建一个具有rows行、cols列的GridLayout布局。

(3) GridLayout(int rows,int cols,int hgap,int vgap): 按指定的行数rows、列数cols、水平间隔hgap和垂直间隔vgap创建一个GridLayout布局。

【示例程序c11_3.java】 使用GridLayout的布局，在容器中放入4个按钮。

```
import java.awt.*;
import javax.swing.*;
public class c11_3 extends Japplet

{
  Container cp=getContentPane( );
  public void init( ) {
    //设置GridLayout布局
    GridLayout myLayout=new GridLayout(2,2,20,30);
```

```
        cp.setLayout(myLayout);
        cp.add(new JButton("按钮A"));
        cp.add(new JButton("按钮B"));
        cp.add(new JButton("按钮C"));
        cp.add(new JButton("按钮D"));
    }
}
```

图11.3 程序c11_3的运行结果

11.1.5 BoxLayout

BoxLayout是swing所提供的布局管理器，它的继承关系如下：

java.lang.Object

javax.swing.BoxLayout

BoxLayout只有两种排列方式，一种是水平，另一种是垂直。我们可以使用BoxLayout所提供的两个常量X_AXIS、Y_AXIS来指明组件在容器中是水平还是垂直排列的。BoxLayout类与Box类结合，可以提供多样化的布局。通常的做法是使用若干个Box容器，由于Box容器的默认布局是BoxLayout，而且只能使用这个布局，因此，每一个Box容器中组件的排列方式也只能按水平或垂直方向排列。

1．BoxLayout和Box的构造函数

创建BoxLayout类的对象的构造方法是：

```
BoxLayout(Container target,int axis)
```

其中，target是容器对象；axis指明target中组件的排列方式，其值可为表示水平排列的BoxLayout.X_AXIS，或为表示垂直排列的BoxLayout.Y_AXIS。

讲到BoxLayout，我们不能不提到Box这个容器。Box这个容器默认的布局是BoxLayout，而且只能使用这个布局，否则编译时就会产生错误。由于BoxLayout是以水平或垂直方式排列的，因此，当我们要创建一个Box容器时，就必须指定Box容器中组件的排列方式是水平还是垂直的。Box的构造函数为：

```
Box(int axis)
```

Box的构造函数中只有一个参数axis，用以指定Box中的组件是按水平还是按垂直方式排列的。axis的值可以用BoxLayout.X_AXIS或BoxLayout.Y_AXIS指定，也可使用Box类提供的两个方法creatHorizontalBox( )与creatVerticalBox( )来指定。

2．BoxLayout类和Box类的常用成员方法

表11.2 BoxLayout类常用的成员方法

成 员 方 法	说 明
getLayoutAlignmentX(Container con)	返回值表示 Container 中对象 X 轴方向的对齐方式
getLayoutAlignmentY(Container con)	返回值表示 Container 中对象 Y 轴方向的对齐方式
setLayoutAlignmentX(Container con)	设置 Container 中对象 X 轴方向的对齐方式
setLayoutAlignmentY(Container con)	设置 Container 中对象 Y 轴方向的对齐方式
LayoutContainer (Container target)	设置 target 窗口容器的布局方式为 BoxLayout

表11.3 Box类常用的成员方法

成 员 方 法	说 明
Box.createHorizontalBox( )	构造水平排列的 Box 组件
Box.createVerticalBox( )	构造垂直排列的 Box 组件
Component.createGlue( )	构造可向水平与垂直方向延伸的 Glue 组件
Component.createHorizontalGlue( )	构造水平的 Glue 组件
Component.createVerticalGlue( )	构造垂直的 Glue 组件
Component.createHorizontalStrut(int width)	构造水平的 Strut 组件
Component.createVerticalStrut(int height)	构造垂直的 Strut 组件
Component.createRigidArea(Dimension d)	构造 Rigid 组件
AecessibleContext.getAccessibleContext( )	取得与 JComponent 相关联的 AccessibleContext
setLayout(LayoutManager l)	抛出 AWTError，Box 只使用 BoxLayout 布局

为方便布局管理，Box类还提供了4种透明组件Glue、Strut、Rigid和Filler，可以将这些透明组件插入其他组件的中间，使这些组件产生分开的效果。这4种透明组件的作用是：

Glue：将Glue两边的组件挤到容器的两端。

Strut：将Strut两端的组件按水平或垂直方向指定的大小分开。

Rigid：可以设置二维的限制，将组件按水平或垂直方向指定的大小分开。

Filler：不仅可以设置二维的限制，将组件按水平或垂直方向指定的大小分开，而且还可以设置最大、较佳、最小的长宽大小。

它们的具体用法请参阅示例程序c11_4.java。

3．使用BoxLayout时需要注意的事项

（1）当组件按BoxLayout布局排列好后，不管窗口缩小或放大都不会变动。

（2）当使用水平排列方式时，若放进去的组件不等高，则系统将会使所有的组件与最高组件等高。

（3）当放在同一行的组件超出容器的宽度时，系统不会自动换行,需要用户自行处理。

【示例程序c11_4.java】 使用BoxLayout布局，将6个组件按结果要求排列。

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class c11_4  extends JApplet
{
   public void init( )
   {   Container cp=getContentPane( );
       Box bBox=Box.createHorizontalBox( );  //设置bBox中的组件按水平方向排列
```

```
cp.add(bBox);  //将bBox容器添加到cp对象中
 //bBox中放入vBox1容器
Box vBox1=Box.createVerticalBox( );  //设置bBox1中的组件按垂直方向排列
JLabel lb=new JLabel("这是标签");
vBox1.add(lb);     //添加标签到vBox1中
JButton bt1=new JButton("这是按钮1");
bt1.setMaximumSize(new Dimension(110,200));  //设置按钮的最大长度
vBox1.add(bt1);  //添加按钮到vBox1中
bBox.add(vBox1);  //添加vBox1到bBox中
 //bBox中放入vBox2容器
```

```
Box vBox2=Box.createVerticalBox( );
bBox.add(vBox2);
JTextField tf1=new JTextField("这是文本框",11);
//设置文本框在容器中沿X方向居中对齐
tf1.setAlignmentX(Component.CENTER_ALIGNMENT);
tf1.setMaximumSize(new Dimension(150,50));
vBox2.add(tf1);
 //vBox2容器中放入vBox2h容器
Box vBox2h=Box.createHorizontalBox( );  //vBox2h容器上的组件按水平方向排列
vBox2.add(vBox2h);
 //vBox2h容器中放入vBox2h1
```

```
Box vBox2h1=Box.createVerticalBox( );  //vBox2h1容器上的组件按垂直方向排列
    //加入垂直透明组件Strut，间隔为20像素
vBox2h1.add(Box.createVerticalStrut(20));
vBox2h1.add(new JTextArea("这是文本区域",15,11));
vBox2h1.add(Box.createVerticalStrut(20));
vBox2h.add(vBox2h1);
  // vBox2h容器中放入vBox2h2
Box vBox2h2=Box.createVerticalBox( );
vBox2h2.add(new JButton("这是按钮2"));
```

```
        vBox2h2.add(Box.createVerticalGlue( ));  //加入垂直透明组件Glue，组件挤到两边
        vBox2h2.add(new JButton("这是按钮4"));
        vBox2h.add(vBox2h2);
    }
}
```

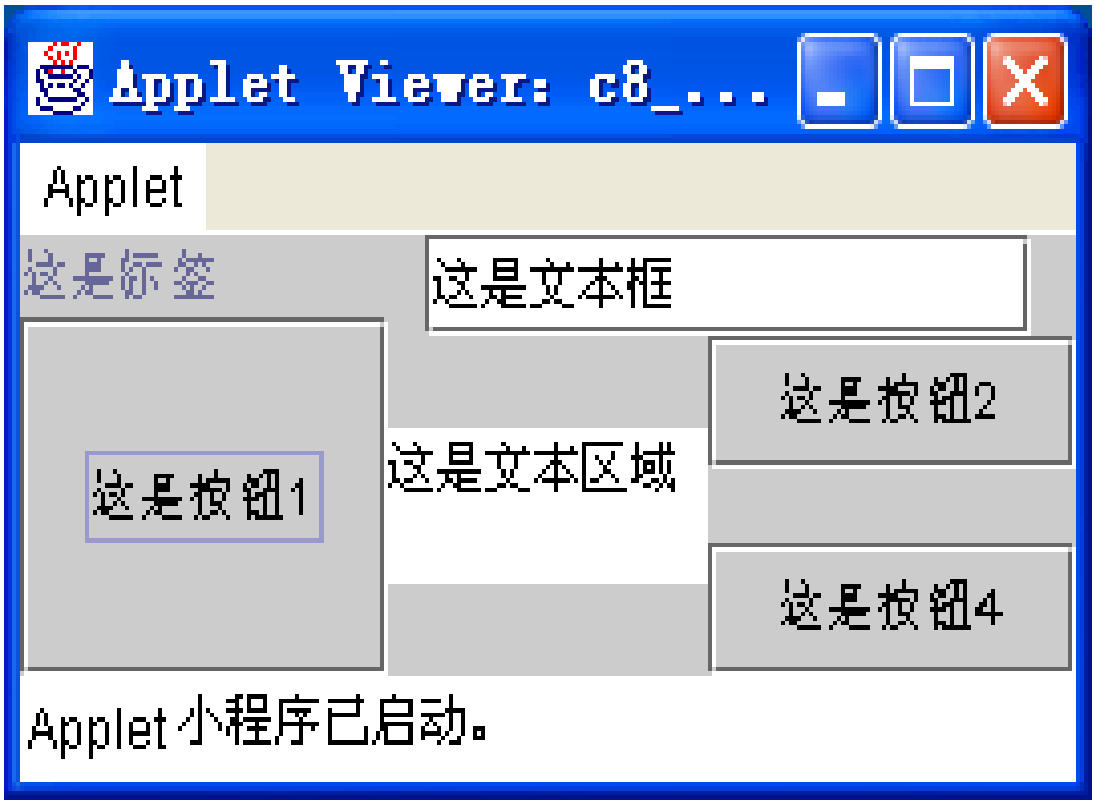

图11.4 程序c11_4的运行结果

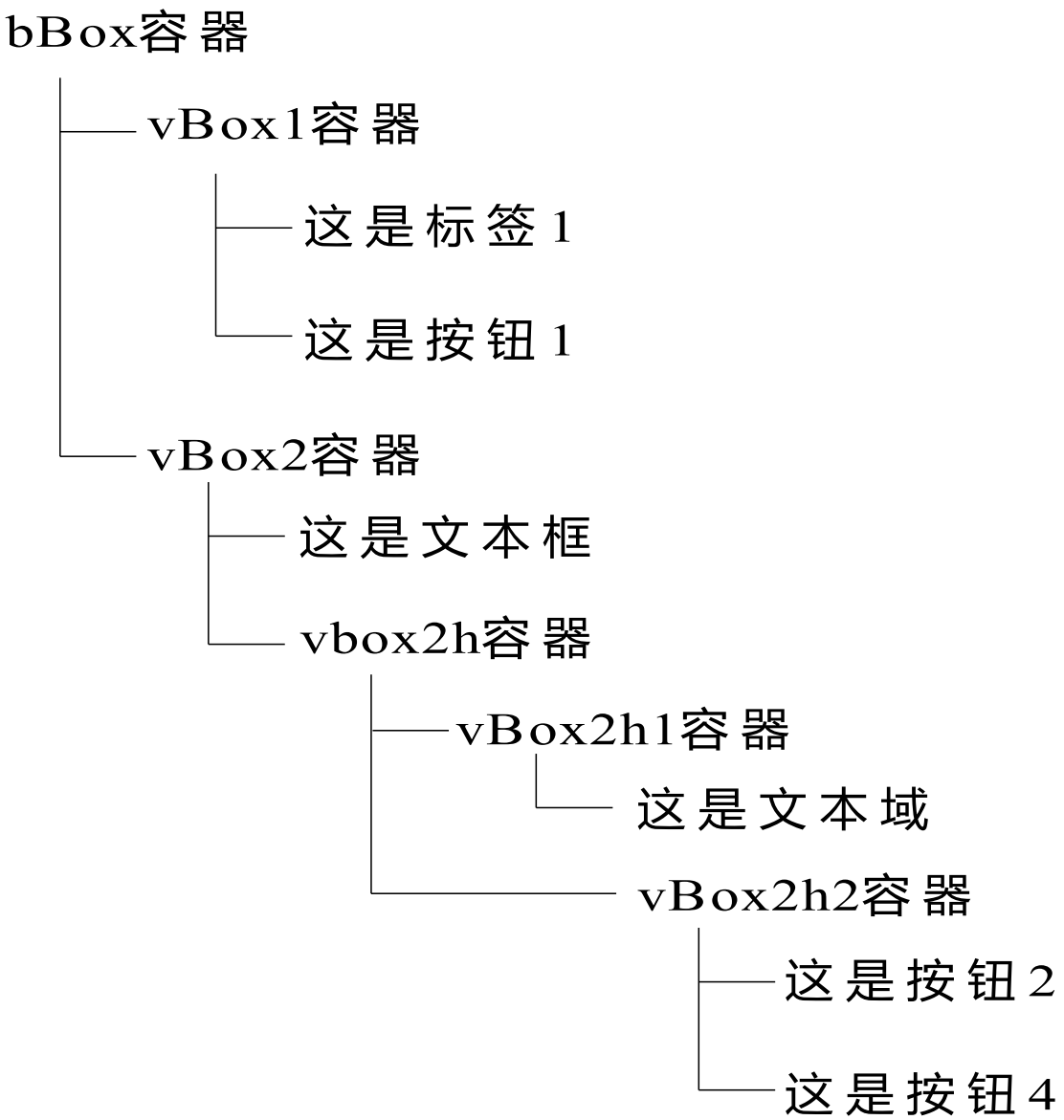
bBox容器
vBox1容器
这是标签1
这是按钮1
vBox2容器
这是文本框
vbox2h容器
vBox2h1容器
这是文本域
vBox2h2容器
这是按钮2
这是按钮4

图11·5 图形层次结构

从图11.5中可以看出：bBox容器是最底层容器，它上面放入两个容器vBox1和vBox2。vBox1容器中放入一个标签1组件和一个按钮1组件。vBox2容器中放入一个文本框组件和一个vbox2h容器。vbox2h容器中放入vBox2h1容器和vBox2h2容器。vBox2h1容器中放入一个文本区域组件。vBox2h2容器中放入一个按钮2组件和一个按钮4组件。

11.2 键盘事件(KeyEvent)

当用户使用键盘进行操作时则会产生KeyEvent事件。处理KeyEvent事件的监听者对象是可以实现KeyListener接口的类，或者是继承KeyAdapter的子类。在KeyListener这个接口中有如下三个事件：

keyPressed(KeyEvent e); 代表键盘按键被按下的事件。

keyReleased(KeyEvent e); 代表键盘按键被放开的事件。

keyTyped(KeyEvent e); 代表按键被敲击的事件。

KeyEvent类中的常用方法有：

(1) public char getKeyChar( )方法，它返回引发键盘事件的按键对应的Unicode字符。如果这个按键没有Unicode字符与之对应，则返回KeyEvent类的一个静态常量KeyEvent.CHAR-UNDEFINED。

(2) public String getKeyText( )方法，它返回引发键盘事件的按键的文本内容。

【示例程序c11_5.java】 键盘事件程序。

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
public class c11_5 extends JApplet
{
    String s,s1;
    JLabel lb1=new JLabel("请按键盘");
    JLabel lb2=new JLabel("复制结果");
    JTextField tf1=new JTextField(11);  //用来输入文字
```

```
JTextArea tf2=new JTextArea(5,11);  //用来显示文字内容
Container cp=getContentPane( );
FlowLayout flow=new FlowLayout(FlowLayout.CENTER,5,5);
public void init( )
{  cp.setLayout(flow);
   cp.add(lb1);
   cp.add(tf1);
   cp.add(lb2);
   cp.add(tf2);
   tf1.addKeyListener(new koLis( ));
}
```

```
    class koLis extends KeyAdapter
    { public void keyTyped(KeyEvent e)
      {  s=tf1.getText( )+e.getKeyChar( );  //获取文本框的内容及键入的字符
         if(e.getKeyChar( )=='\n')
            //若按回车键,则将文本框的内容送入文本域中
           {   s1=tf2.getText( )+s;
               tf1.setText("");
               tf2.setText(s1);
           }
      }
    }
}
```

图11.6 c11_5运行结果

11.3 鼠标事件(MouseEvent)

在图形用户界面中，鼠标主要用来进行选择、切换或绘画。当用户用鼠标进行交互操作时，会产生鼠标事件MouseEvent。处理MouseEvent事件的监听者对象是可以实现MouseListener接口和MouseMotionListener接口的类，或者是继承MouseAdapter的子类。与Mouse有关的事件可分为两类：一类是MouseListener接口，共提供5种方法，主要针对鼠标的按键与位置作检测；另一类是MouseMotionListener接口，共提供2种方法，主要针对鼠标的坐标与拖动操作做处理。这些方法列于表11.4中。

表11.4 MouseEvent事件及其监听者

事件监听者	成 员 方 法	说 明
MouseListener	moveClicked(MouseEvent e)	代表鼠标点击事件
	moveEntered(MouseEvent e)	代表鼠标进入事件
	moveExited(MouseEvent e)	代表鼠标离开事件
	movePressed(MouseEvent e)	代表鼠标按下事件
	moveReleased(MouseEvent e)	代表鼠标释放事件
MouseMotionListener	moveDragged(MouseEvent e)	代表鼠标拖动事件
	moveMoved(MouseEvent e)	代表鼠标移动事件

表11.5 MouseEvent类的常用成员方法

成 员 方 法	功 能 说 明
public int getX( )	返回发生鼠标事件的 X 坐标
public int getY( )	返回发生鼠标事件的 Y 坐标
public Point getPoint( )	返回 Point 对象，包含鼠标事件发生的坐标点
public int getClickCount( )	返回鼠标点击事件的点击次数

<【示例程序c11_6.java】 鼠标事件程序。

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
public class c11_6 extends JApplet
{
   int x,y;
   JLabel lb1=new JLabel("X:"), lb2=new JLabel("Y:"), lb3=new JLabel("")
   JTextField tf1=new JTextField(5), tf2=new JTextField(5);
   Container cp=getContentPane( );
   FlowLayout flow=new FlowLayout(FlowLayout.CENTER,5,5);
   public void init( )
   {
```

```
        cp.setLayout(flow);
        cp.add(lb1);
        cp.add(tf1);
        cp.add(lb2);
        cp.add(tf2);
        cp.add(lb3);
        addMouseListener(new mouseListener( ));
        addMouseMotionListener(new koLis( ));
      }
 class mouseListener implements MouseListener
{
  public void mouseClicked(MouseEvent e)
     { lb3.setText("点击鼠标");  }
```

```
public void mousePressed(MouseEvent e)
    { lb3.setText("鼠标按钮按下"); }
public void mouseEntered(MouseEvent e)
    { lb3.setText("鼠标进入窗口"); }
public void mouseExited(MouseEvent e)
    { lb3.setText("鼠标不在窗口"); }
public void mouseReleased(MouseEvent e)
    { lb3.setText("鼠标按钮松开"); }
}
class koLis implements MouseMotionListener
{
    public void mouseMoved(MouseEvent e)
    {
        x=e.getX( );
        y=e.getY( );
```

```
            tf1.setText(String.valueOf(x));
            tf2.setText(String.valueOf(y));
        }
    public void mouseDragged(MouseEvent e)
        { lb3.setText("拖动鼠标"); }
    }
}
```

图11.7 程序c11_6的运行结果

11.4 窗 口 与 面 板

11.4.1 JFrame容器

JFrame是Java Application程序的图形用户界面容器，是一个有边框的容器。JFrame类包含支持任何通用窗口特性的基本功能，如最小化窗口、移动窗口、重新设定窗口大小等。JFrame容器作为最底层容器，不能被其他容器所包含，但可以被其他容器创建并弹出成为独立的容器。JFrame类的继承关系如下：

java.lang.Object

java.awt.Component

java.awt.Container

java.awt.Window

java.awt.Frame

javax.swing.JFrame

使用JFrame容器需要注意以下几点：

(1) 可以使用JFrame( )方法创建一个无标签的JFrame对象，也可以使用JFrame(String title)方法创建一个标签为title的JFrame对象，还可以使用专门的方法getTitle( )和setTitle(String)来获取或指定Frame的标题。

(2) 使用setSize(int x,int y)方法设置JFrame容器大小。

(3) 创建的JFrame是不可见的，若要使其可见，则需要使用show( )方法或给出实际参数为true的setVisible(boolean)方法。

(4) 向JFrame中添加组件时，必须先取得ContentPane，然后再使用add( )方法把组件加入到此ContentPane中，而不能像AWT中的Frame那样直接调用add( )方法。

(5) JFrame类可以引发WindowEvent类代表的所有七种窗口事件(见表11.8)。

(6) 每个JFrame在其右上角都有三个控制图标(如图11.8所示)，分别代表最小化、最大化和关闭窗口的操作。其中，JFrame可自动完成窗口的最小化和最大化操作，而关闭窗口的操作不能通过点击关闭图标来实现，但可以使用下述三个办法之一来关闭窗口:

① 设置一个按钮，当用户点击此按钮时关闭窗口；

② 用WINDOWS_CLOSING事件做出响应，关闭窗口；

③ 使用菜单命令。

无论使用哪一种办法，都需要用到关闭JFrame的System.exit(0)方法。

【示例程序c11_7.java】 创建两个窗口对象，点击第一个窗口中的按钮时打开第二个窗口。

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
public class c11_7 implements ActionListener
{
    public c11_7( )
    {
```

```
        JFrame f=new JFrame("这是一个JFrame");  //创建JFrame对象
        Container cp=f.getContentPane( );  //创建JFrame的容器对象
        JButton bt1=new JButton("请点击");
        bt1.addActionListener(this);
        cp.add(bt1);
        f.pack( );          //调整窗口
        f.show( );        //显示窗口
        f.addWindowListener(new WinLis( ));
    }

public void actionPerformed(ActionEvent e)
{
```

```
        JLabel  lb=new JLabel("这是第二个窗口");
        JFrame nf=new JFrame( );
        Container cp1=nf.getContentPane( );
        nf.setTitle("这是一个新JFrame");
        nf.setSize(180,110);  //设置窗口大小
        nf.show( );
        cp1.add(lb);
        nf.addWindowListener(new WinLis( ));
    }
    public static void main(String[ ] arg)
    {
```

```
        new c11_7( );
    }
    class WinLis extends WindowAdapter
    {
        public void windowsClosing(WindowEvent e)
          {  System.exit(0);}
     }
}
```

图11.8 程序c11_7的运行结果1

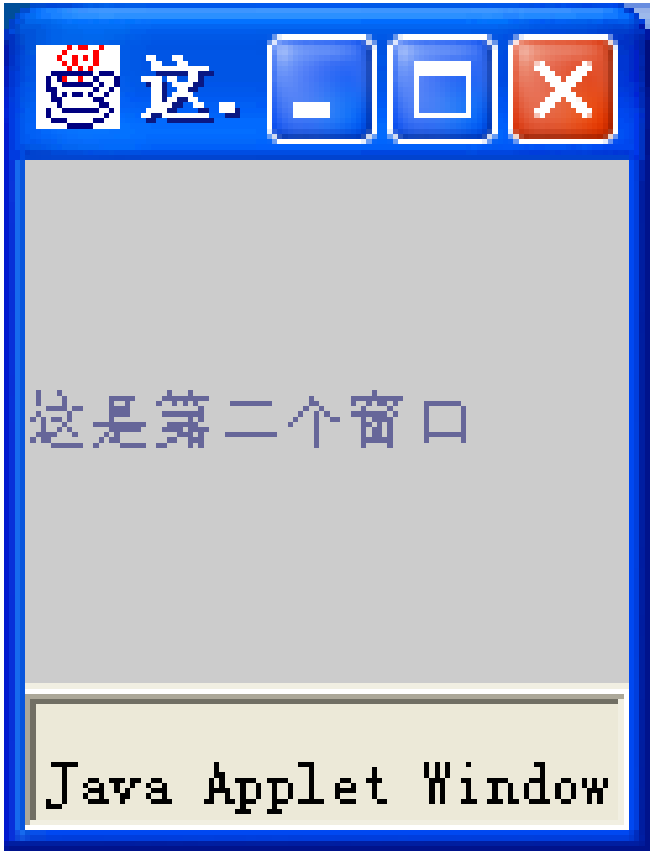

图11.9 程序c11_7的运行结果2

11.4.2 窗口事件(WindowEvent)

从c11_7.java程序的运行结果可以看出，若删除窗口事件，则窗口无法关闭。我们可以利用Java语言提供的窗口事件WindowEvent来对窗口进行操作。WindowEvent类包含表11.6所列的几个具体窗口事件。

表11.6 WindowEvent类包含的窗口事件

窗 口 事 件	说 明
WINDOW_ACTIVATED	代表窗口被激活(在屏幕的最前方待命)
WINDOW_DEACTIVATED	代表窗口失活(其他窗口被激活后原活动窗口失活)
WINDOW_OPENED	代表窗口被打开
WINDOW_CLOSED	代表窗口已被关闭(指已关闭后)
WINDOW_CLOSING	代表窗口正在被关闭(指关闭前，如点击窗口的关闭按钮)
WINDOW_ICONIFIED	代表使窗口最小化成图标
WINDOW_DEICONIFIED	代表使窗口从图标恢复

WindowEvent类的主要方法有getWindow( )和getSource( )。这两个方法的区别是：getWindow( )方法返回引发当前WindowEvent事件的具体窗口，返回值是具体的Window对象；getSource( )方法返回的是相同的事件引用，其返回值的类型为Object。

【示例程序c11_8.java】 制作两个JFrame窗口，实现两个窗口的切换、关闭、最小化等操作。

```
import java.awt.*;
import javax.swing.*;
import javax.swing.JFrame;
import java.awt.event.*;
public class c11_8
  {  JLabel  lb1=new JLabel("这是第一个窗口");
     JLabel  lb2=new JLabel("这是第二个窗口");
     public static void main(String[ ] arg)
      { new c11_8( );  }
```

```
public c11_8( )
{
   JFrame f1=new JFrame( ); //创建JFrame对象
   JFrame f2=new JFrame( );
   Container cp=f1.getContentPane( ); //创建JFrame的容器对象，获得ContentPane
   Container cp1=f2.getContentPane( );
   f1.setTitle("JFrame1");
   f2.setTitle("JFrame2");
   f1.setSize(150,110); //设置窗口大小
   f2.setSize(150,110);
   cp.add(lb1);
```

```
        f1.show( );       //设置窗口为可见
        cp1.add(lb2);
        f2.show( );
        f1.addWindowListener(new WinLis( ));
        f2.addWindowListener(new WinLis( ));
    }
class WinLis extends WindowAdapter
{ public void windowOpened(WindowEvent e)
    {  }  //打开窗口
```

```
        public void windowActivated(WindowEvent e)
          {  }  //将窗口设置成活动窗口
        public void windowDeactivated(WindowEvent e)
          { }  //将窗口设置成非活动窗口
        public void windowClosing(WindowEvent e)
          {    System.exit(0);  }    //窗口关闭
        public void windowIconified(WindowEvent e)
          { }  //最小化窗口
      }
}
```

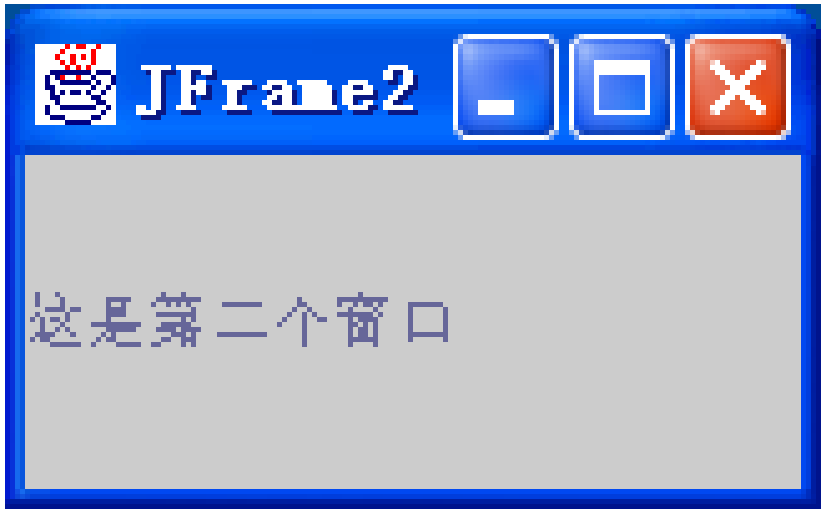

图11.10 程序c11_8的运行结果

11.4.3 JPanel容器

在设计用户界面时，为了更合理的安排各组件在窗口的位置，可以考虑将所需组件先排列在一个容器中，然后将其作为一个整体嵌入窗口。Panel和JPanel就是这样一类被称为面板的容器。它们是一类无边框的、不能被移动、放大、缩小或关闭的容器。AWT的Panel与Swing的JPanel的区别是：JPanel支持双缓冲(Double Buffering)功能，在处理动画上较少发生画面闪烁的情况。JPanel类的继承关系如下：

java.lang.Object

java.awt.Component

java.awt.Container

javax.swing.JComponent

javax.swing.JPanel

不能把JPanel作为一个图形界面程序最底层的容器，也不能指明JPanel的大小。JPanel总是作为一个容器组件被加入到JFrame、JApplet等其他容器中，当然，JPanel也可以加入到JPanel容器中。JPanel的大小由包含在它里边的组件，包容它的那个容器的布局策略，以及该容器中的其他组件所决定。

表11.7 JPanel类构造方法

构 造 方 法	功 能 说 明
JPanel( )	使用默认的 FlowLayout 方式创建 JPanel 对象
JPanel(boolean isDoubleBuffered)	使用默认的 FlowLayout 方式创建 JPanel 对象，当参数为 true 时，将建立具有双缓冲功能的 JPanel
JPanel(LayoutManager layout)	使用指定的 layout 布局方式创建 JPanel 对象
JPanel(LayoutManager layout, boolean isDoubleBuffered)	使用指定的 layout 布局方式创建 JPanel 对象，且参数为 true 时，具有双缓冲功能

【示例程序c11_9.java】 设置两个面板容器p1和p2，点击p1容器中的按钮使p2容器的标签产生结果。

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class c11_9
{
    public c11_9( )
    {
```

```
JFrame f=new JFrame("JPanelDemo");
Container cp=f.getContentPane( );
cp.setLayout(new GridLayout(2,1));
JLabel[ ] lb=new JLabel[3];  //创建标签数组
for(int i=0; i<lb.length ; i++)  //设置每个标签的属性
 {
   lb[i]=new JLabel("标签 "+(i+1),JLabel.CENTER);
   lb[i].setBackground(Color.yellow);  //设置标签颜色
   lb[i].setBorder(BorderFactory.createEtchedBorder( ));  //设置标签边框
   lb[i].setOpaque(true);  //让组件变为不透明，使标签颜色显示出来
 }
```

```
JPanel pal1=new JPanel(new GridLayout(1,1));  //创建面板对象
pal1.add(lb[0]);  //加载第0个标签在面板上
JPanel pal2=new JPanel(new GridLayout(1,2));
JPanel pal3=new JPanel(new BorderLayout( ));
pal3.add(lb[1],BorderLayout.NORTH);
pal3.add(lb[2],BorderLayout.SOUTH);
pal3.add(new JButton("中部"),BorderLayout.CENTER);
JPanel pal4=new JPanel(new FlowLayout( ));
pal4.add(new JTextArea("JTextArea",5,11));
pal2.add(pal3);
pal2.add(pal4);
cp.add(pal1);
cp.add(pal2);
f.pack( );
f.show( );
f.addWindowListener(new WinLis( ));
```

```
    }
    class WinLis extends WindowAdapter
    {
       public void windowClosing(WindowEvent e)
        { System.exit(0); }
     }
    public static void main(String[ ] arg)
     {
        new c11_9( );
     }
}
```

图11.11 程序c11_9的运行结果

程序说明：

该程序的最底层容器是cp容器，它上面放入了pal1容器和pal2容器。pal1容器中放入了一个标签1组件。pal2容器中放入pal3容器和pal4容器。pal3容器中放入一个标签2组件、一个按钮组件和一个标签3组件。pal4容器中放入一个文本域组件。

11.4.4 JScrollPane容器

当窗口里的内容大于窗口时，我们可以在窗口的右边和下边设置滚动条，借助于滚动条我们就可以看到整个窗口的内容。JScrollPane就是具有这种功能的组件，我们将它称为滚动面板，用于滚动窗口。JScrollPane类的继承关系如下：

```
java.lang.Object
    java.awt.Component
        java.awt.Container
            javax.swing.JComponent
            javax.swing.  JScrollPane
```

创建JScrollPane类的对象的构造方法列于表11.8中。

表11.8 JScrollPane类的构造方法

构 造 方 法	功 能 说 明
JScrollPane( )	创建空的 JScrollPane 对象
JScrollPane(Component view)	创建 JScrollPane 对象，并加入一个 view 组件，当组件内容大于显示区域时会自动产生滚动条
JScrollPane(int v,int h)	创建有水平滚动条与垂直滚动条的 JScrollPane 对象
JScrollPane(Component view,int v,int h)	创建有水平滚动条与垂直滚动条的 JScrollPane 对象，并加入一个 view 组件

还可以利用下面的参数来设置JScrollPane中滚动条的出现时机：

HORIZONTAl_SCROLLBAR_ALWAYS：显示水平滚动条。

VERTICAl_SCROLLBAR_ALWAYS：显示垂直滚动条。

HORIZONTAl_SCROLLBAR_NEVER：不显示水平滚动条。

VERTICAL_SCROLLBAR_NEVER：不显示垂直滚动条。

HORIZONTAl_SCROLLBAR_AS_NEEDED：需要时显示水平滚动条，即当组件的内容在水平方向上大于显示区域时出现水平滚动条。

VERTICAl_SCROLLBAR_AS_NEEDED：需要时显示垂直滚动条，即当组件的内容在垂直方向上大于显示区域时出现垂直滚动条。

这些参数是在ScrollPaneConstants接口中定义的，而JScrollPane类实现了此接口，因此也就能使用这些参数。

11.4.5 JScrollbar组件

事实上，JScrollPane是由JViewPort和JScrollBar组件组成的。JViewPort组件主要负责显示内容的区域大小；JScrollBar组件则产生窗口滚动条，让用户看到整个内容。用户使用JScrollPane组件时不会直接与JViewPort和JScrollBar组件打交道，使用比较方便。但是，当我们想对滚动条做更细的设置时，例如在拖动时一次滚动多少区域等，就必须了解JScrollBar所提供的功能。

JScrollbar被称为滚动条，是一种比较特殊的GUI组件，它可以是水平方向的，也可以是垂直方向的。创建JScrollbar类的对象将创建一个如图11.12所示的含有增加箭头、减少箭头、滚动槽和滚动块的滚动条。

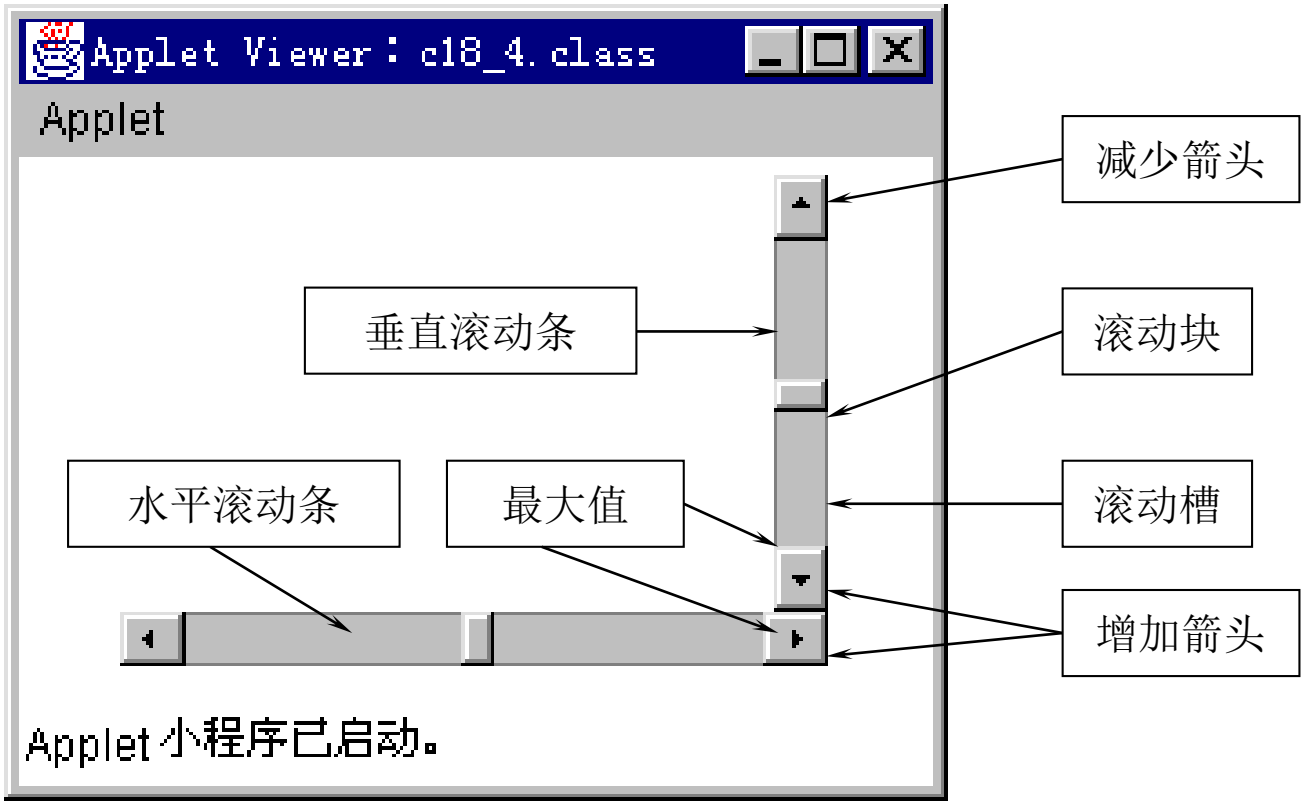

图11.12 滚动条各部分的名称

1. JScrollbar组件的构造方法

表11.9 JScrollbar类构造方法

构 造 方 法	功 能 说 明
JScrollbar( )	按默认方式创建一个滚动条对象(垂直滚动条)
JScrollbar(int orientation)	按指定方向创建一个滚动条，包括 Scrollbar.VERTICAL(垂直)和 Scrollbar.HORIZONTAL(水平)
Scrollbar(int orientation,int value, int extent,int minimum, int maximum)	使用指定的方向、初始值、滚动块的大小、最小值和最大值创建一个滚动条

在JScrollbar的构造方法中，参数orientation的值可为JScrollbar.VERTICAL或JScrollbar.HORIZONTAL，用于指定滚动条的方向；value值用于指定滚动块的初始值，默认为0，表示开始时滚动块就设置在这个位置上；extent值用于指定滚动块的位移区域，当用户在滚动块与滚动箭头之间点击时，将由这个值来确定滚动块应移动的位置；minimum值用于指定滚动槽的最小值(默认值为0)；maximum值用于指定滚动槽的最大值(默认值为100)。在实际使用中，如果value值小于minimum值，那么value值被设置成等于minimum值。如果value值大于maximum值，那么value值就设置成等于maximum值。若设extent=20，minimum=0，maximum=100，则滚动块的位移范围为0～80。

2. JScrollbar类的常用成员方法

表11.10 JScrollbar类的成员方法

成 员 方 法	功 能 说 明
getOrientation( )	获取滚动条的方向
setOrientation (int orientation)	设置滚动条的方向
getValue( )	获取滚动条的当前值
setValue(int value)	设置滚动条的值
getMinimum( )	返回滚动条的最小值
setMinimum(int newMinimum)	设置滚动条的最小值
getMaximum( )	获取滚动条的最大值
setMaximum(int newMaximum)	设置滚动条的最大值
setVisibleAmount(int extent)	设置滚动条的 extent 值
setUnitIncrement(int v)	设置滚动条的单位增量
setBlockIncrement(int v)	设置滚动条的块增量
getBlockIncrement( )	获取滚动条的块增量
setValues(int value,int extent,int minimum,int maximum)	设置滚动条的各项值
addAdjustmentListener(AdjustmentListener l)	添加指定的监听者对象
removeAdjustmentListener(AdjustmentListener l)	删除指定的监听者对象

3. 调整事件(AdjustmentEvent)

JScrollbar能够接受调整事件(AdjustmentEvent)。AdjustmentEvent类只包含一个事件，即代表GUI组件状态发生连续变化的事件ADJUSTMENT_VALUE_CHANGED。当用户通过各种方式改变滚动块位置从而改变其代表的数值时，都会引发调整事件。

为了对JScrollbar上发生的调整事件进行处理，在程序中必须注册调整事件的监听者对象，并通过相应的方法来实现其功能。例如，在c11_11.java中注册了滚动条的AdjustmentListener接口的调整事件的监听者对象sbr.addAdjustmentListener(this)，并在实现了监听者的类中定义了响应事件的adjustmentValueChanged(AdjustmentEvent e)方法，当改变滚动块位置时，系统就会自动调用该方法进行处理。

【示例程序c11_10.java】 设计一个标签、三个滚动条，三个滚动条分别代表红、绿、蓝三种颜色，改变滑块的值从而改变标签背景的颜色。

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class c11_10 implements AdjustmentListener
{
    int r=0,g=0,b=0;
    String s,s1=" ",s2=" ",s3=" ";
    JScrollBar sbr,sbg,sbb;  //声明建立滚动条对象
    JPanel pa1,pa2,pa3;
```

```
        JLabel lb1= new JLabel("刻度："),
        lb2=new JLabel(" 标签 "),
        lb3=new JLabel("    调色版  "),
        lbr=new JLabel("红色"),
        lbg=new JLabel("绿色"),
        lbb=new JLabel("蓝色");
public c11_11( )
  { JFrame f = new JFrame("JScrollBar");
    Container cp=f.getContentPane( );
    Box baseBox=Box.createVerticalBox( );
    cp.add(baseBox);
    Box box1=Box.createHorizontalBox( );
```

```
box1.add(lb1);
box1.add(lb2);
baseBox.add(box1);
Box box3=Box.createVerticalBox( );
baseBox.add(box3);
lb3.setBackground(new Color(0,0,0));  //设置标签颜色
lb3.setBorder(BorderFactory.createEtchedBorder( ));//设置标签边框
lb3.setOpaque(true);   //让组件变为不透明，使标签颜色显示出来
lb3.setMaximumSize(new Dimension(450,200));
box3.add(lb3);
sbr=new     JScrollBar(JScrollBar.HORIZONTAL,11,11,0,260);  //创建水平方向滚动条对象
```

```
sbr.setUnitIncrement(5);  //设置此滚动条拖动滚动块时的单位增量
sbr.setBlockIncrement(11);   //设置鼠标在滚动条上点击时滚动块的块增量
sbr.addAdjustmentListener(this);   //注册sbr给监听对象
box3.add(lbr); box3.add(sbr);
sbg=new JScrollBar(JScrollBar.HORIZONTAL,11,11,0,260);
sbg.setUnitIncrement(5);
sbg.setBlockIncrement(11);
sbg.addAdjustmentListener(this);
box3.add(lbg); box3.add(sbg);
sbb=new JScrollBar(JScrollBar.HORIZONTAL,11,11,0,260);
sbb.setUnitIncrement(5);
sbb.setBlockIncrement(11);
```

```
        sbb.addAdjustmentListener(this);
        box3.add(lbb);box3.add(sbb);
        f.pack( );
        f.show( );
        f.addWindowListener(new WinLis( ));
    }
class WinLis extends WindowAdapter
    { public void windowClosing(WindowEvent e)
        { System.exit(0); }
    }
public void adjustmentValueChanged(AdjustmentEvent e)
{   if ((JScrollBar)e.getSource( )==sbr)
        { r=e.getValue( ); s1="red: ";}
```

```
        if ((JScrollBar)e.getSource( )==sbg)
            { g=e.getValue( ); s2="green: ";}
        if ((JScrollBar)e.getSource( )==sbb)
            { b=e.getValue( ); s3="blue: ";}
        s=s1+r+" "+s2+g+" "+s3+b;
        lb2.setText(s);
        lb3.setBackground(new Color(r,g,b));
    }
    public static void main(String[ ] arg)
    {  new c11_11( );    }
}
```

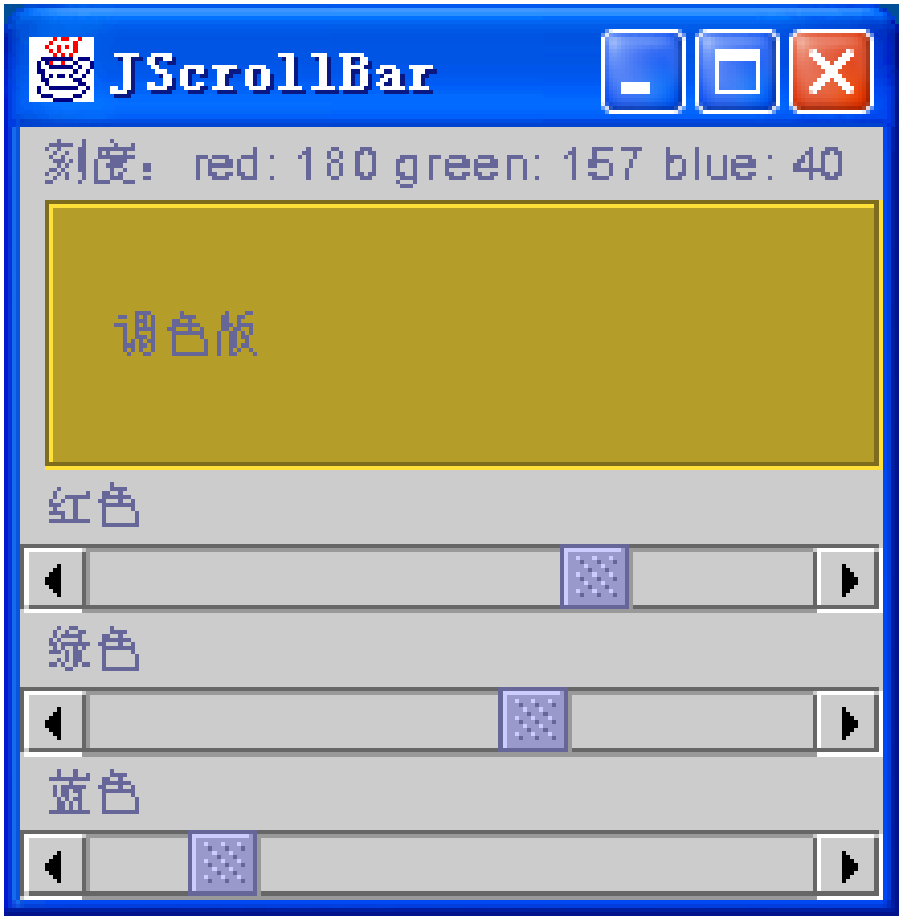

图11.13 程序c11_10的运行结果

11.4.6 JTabbedPane容器

当界面上需要放置的组件很多时，可以使用的另一种容器是JTabbedPane。JTabbedPane容器与我们日常使用的卡片盒类似，它由多个称为标签框架的卡片和表明该框架的标签组成。每个标签框架和标签都自成一个系统(也可称为一张卡片)，我们可以在标签框架中加入各式各样的组件及功能。由于这些卡片被叠放在一起，为了使用方便，卡片上的标签在顶行或底部排成一行(也可以在左边或右边排成一列)，当用鼠标点击某一个标签时，这个标签所在的卡片(标签框架窗口)就会被翻到最上面，显示出此框架的内容。

JTabbedPane类的继承关系如下：

java.lang.Object

java.awt.Component

java.awt.Container

javax.swing.JComponent

javax.swing.JTabbedPane

可以使用下述两个构造方法创建JTabbedPane类的对象：

JTabbedPane( )：创建一个空的JTabbedPane对象。

JTabbedPane(int tapposition)：创建一个空的JTabbedPane对象，并指定标签的位置，如TOP、BOTTOM、LEFT或RIGHT。

【示例程序c11_11.java】 建立带多个卡片的窗口。

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class c11_11 extends JFrame
 { // 创建 JTabbedPane 对象，并指定标签显示在上方
   JTabbedPane jtab=new JTabbedPane(JTabbedPane.TOP);
   JScrollPane sp; //声明JScrollPane对象
   public static void main(String args[ ])
    { c11_11 f=new c11_11( );
      f.setTitle("JTabbedPane对象的应用");
      f.setSize(300,300);
      f.setVisible(true);
    }
```

```
public c11_11( )
 {
  JLabel lb[ ]=new JLabel[6];  //声明JLabel 数组
  Icon pic;  //创建图片对象
  String title;  //创建标签名称对象
  for(int i=1;i<=5;i++)
  {  pic=new ImageIcon("00"+i+".jpg");  //获得图片
     lb[i]=new JLabel( );  //创建 JLabel 对象
     lb[i].setIcon(pic);  //设定 JLabel 图标
     title = "第 "+ String.valueOf(i) +" 页";  //设定标签名称
     jtab.add(title,lb[i]);  //将 JLabel 对象加入 jtab
  }
```

```
    getContentPane( ).add(jtab);  //将 jtab 加入到窗口中
    int v=ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED;
    int h=ScrollPaneConstants.HORIZONTAL_SCROLLBAR_AS_NEEDED;
    sp=new JScrollPane(jtab,v,h);  //创建JScrollPane对象,并加载jtab对象
    getContentPane( ).add(sp);  //将 sp 加入窗口中
    addWindowListener(new WinLis( ));
  }
 class WinLis extends WindowAdapter
  { public void windowClosing(WindowEvent e)
     { System.exit(0); }
  }
}
```

图11.14 程序c11_11的运行结果

11.5 菜 单 设 计

菜单和工具栏可提供简单明了的指示说明，让用户顺利地完成软件的操作。菜单是非常重要的GUI组件，是软件中必备的组件之一，利用菜单可以将程序功能模块化。

Java语言中提供了多种样式的菜单，如一般式、复选框式、快捷键式及弹出式菜单等。这里我们仅介绍一般式菜单。在Java中，一个一般式菜单由如图11.15所示的菜单栏(JMenuBar)、菜单(JMenu)和菜单项(JMenuItem)三类对象组成。

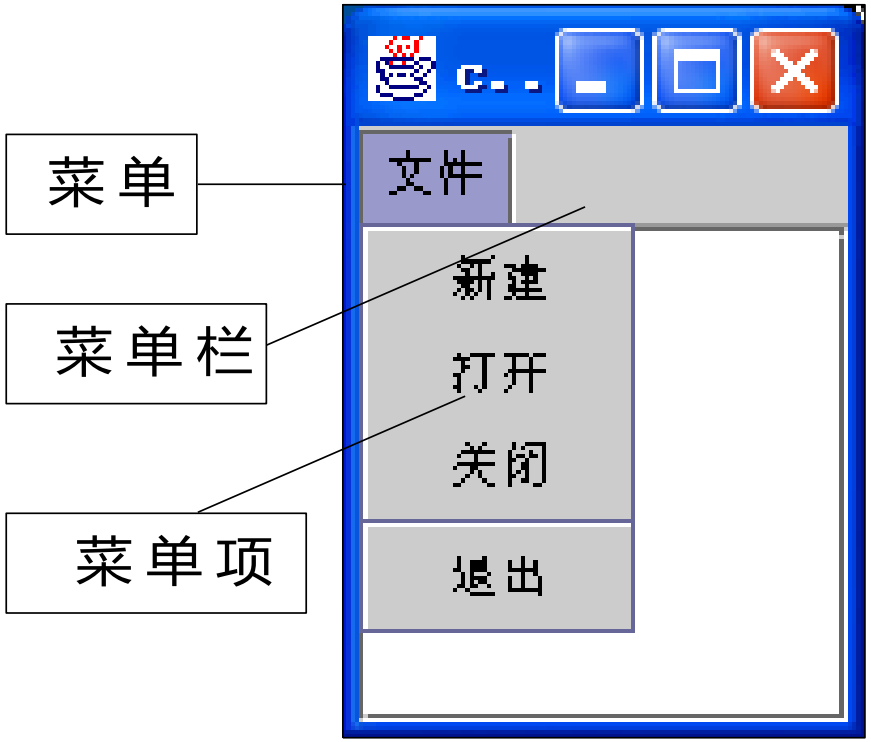

图11.15 菜单的组成

java.lang.Object

java.awt.Component

java.awt.Container

javax.swing.JComponent

javax.swing.JMenuBar

javax.swing.AbstractButton

javax.swing.JMenuItem

javax.swing.JMenu

1．菜单栏(JMenuBar)

菜单栏用来封装与菜单栏相关的各项操作，它只用来管理菜单，不参与交互式操作。Java应用程序中的菜单都包含在一个菜单栏对象之中。创建JMenuBar类菜单栏对象的构造方法只有一个，那就是：

JMenuBar( )

JMenuBar需要结合至少一个以上的JMenu组件才会在画面上显现出视觉效果。

2．菜单(JMenu)

菜单是用来存放和整合菜单项(JMenuItem)的组件，它是构成一个菜单不可或缺的组件之一。菜单可以是单一层次的结构，也可以是一个多层次的结构，具体使用何种形式的结构则取决于界面设计上的需要。创建菜单对象使用JMenu类的下述构造方法：

JMenu( )：创建一个空标签的JMenu对象。

JMenu(String text)：使用指定的标签创建一个JMenu对象。

JMenu(String text，Boolean b)：使用指定的标签创建一个JMenu对象，并给出此菜单是否具有下拉式的属性。

JMenu(Action a)：创建一个支持Action的JMenu对象。

3．菜单项(JMenuItem)

菜单项用来封装与菜单项相关的操作,它是菜单系统中最基本的组件。从前面列出的继承关系可以看出，菜单项JMenuItem继承AbstractButton类，因此JMenuItem具有许多AbstractButton的特性，也可以说JMenuItem是一种特殊的按钮。所以JMenuItem支持了许多在按钮上好用的功能，例如加入图标(Icon)，以及当我们在菜单中选择到某一项JMenuItem时就如同按下按钮的操作一样会触发ActionEvent事件。我们可以通过ActionEvent的机制来针对不同的JMenuItem编写其对应的程序代码。

创建菜单项对象使用javax.swing.JMenuItem类的构造方法列于表11.11中。此外，还可以通过调用成员方法开放或禁止菜单项的使用，关于这些成员方法请读者查阅Java手册或系统帮助。

表11.11 JMenuItem类的构造方法

构 造 方 法	功 能 说 明
JMenuItem( )	使用空标签构造一个菜单项对象
JMenuItem(Action a)	创建一个支持 Action 的菜单项对象
JMenuItem(String text)	使用指定的标签创建一个菜单项对象
JMenuItem(Icon icon)	创建一个指定图标的菜单项对象
JMenuItem(String text，Icon icon)	创建一个指定标签和图标的菜单项对象
JMenuItem(String text，int mnemonic)	创建一个指定标签和键盘快捷设置的菜单项对象

4. 制作菜单的一般步骤

制作一个可用的菜单系统，一般需要经过下面的几个步骤：

(1) 创建一个JMenuBar对象并将其放置在一个JFrame中；

(2) 创建JMenu对象；

(3) 创建JMenuItem对象并将其添加到JMenu对象中；

(4) 把JMenu对象添加到JMenuBar中。

当然，上面的这几步主要是创建菜单的结构，如果要使用菜单所指出的功能，则必须要为菜单项注册监听者，并在监听者提供的事件处理程序中写入相应的代码。

【示例程序c11_12.java】　创建菜单。

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

public class c11_12 extends JFrame implements ActionListener
  {
      JTextArea tf=new JTextArea( );
      JMenuBar bar=new JMenuBar( );  //创建JMenuBar对象
      JMenu menu=new JMenu("文件");  //创建JMenu对象
      JMenuItem newf=new JMenuItem("新建");  //创建JMenuItem对象
      JMenuItem open=new JMenuItem("打开");
      JMenuItem close=new JMenuItem("关闭");
      JMenuItem quit=new JMenuItem("退出");
      public c11_12( )
```

```
{   super("c11_12");          //设定JFrame的标签
    getContentPane( ).add(new JScrollPane(tf));  //创建JFrame的容器对象
    tf.setEditable(false);  //设置文本区域不可编辑
    bar.setOpaque(true); //设置bar为不透明，若设置bar为透明，则在选择菜单时
                          //会有残影存留在JMenuBar上
    setJMenuBar(bar);          //加入bar到Jframe中
    menu.add(newf);          //加入JMenuItem对象到menu中
    menu.add(open);
    menu.add(close);
    menu.addSeparator( );  //在JMenu中加入一分隔线
    menu.add(quit);
    bar.add(menu);          //将menu加载到bar上
```

```
    newf.addActionListener(this); //注册JMenuItem对象给监听者对象
    open.addActionListener(this);
    close.addActionListener(this);
    quit.addActionListener(this);
    addWindowListener(new WinLis( ));
}
public void actionPerformed(ActionEvent e)
{   if(e.getSource( )==newf)tf.setText("新建");
    if(e.getSource( )==open)tf.setText("打开");
    if(e.getSource( )==close)tf.setText("关闭");
    if(e.getSource( )==quit)System.exit(0);
 }
```

```
        class WinLis extends  WindowAdapter

        {  public void windowClosing(WindowEvent e)
              { System.exit(0); }
        }
        public static void main(String[ ] args)
        {  JFrame f = new c11_12( );
           f.setSize(400,200);
           f.setVisible(true);
        }
}
```

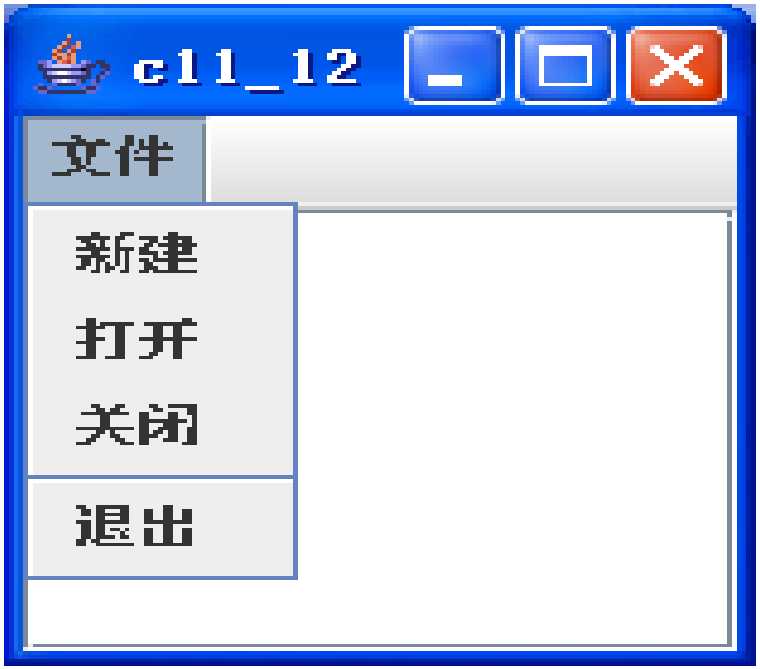

图11.16 程序c11_12的运行结果

练习：编程实现调色板功能

程序打包，命名规则：姓名-学号-班级-第13讲作业.rar

提交教学平台