第17章
Introduction to Java Programming, 8E

创建图形用户界面

学习目标

- 使用不同的用户界面组件，例如，JButton、JCheckBox、JRadioButton、JLabel、JTextField、JTextArea、JComboBox、JList、JScrollBar和JSlider创建图形用户界面（17.2～17.11节）。
- 为不同类型的事件创建监听器（17.2～17.11节）。
- 探究JButton（17.2节）。
- 探究JCheckBox（17.3节）。
- 探究JRadioButton（17.4节）。
- 探究JLabel（17.5节）。
- 探究JTextField（17.6节）。
- 探究JTextArea（17.7节）。
- 探究JComboBox（17.8节）。
- 探究JList（17.9节）。
- 探究JScrollBar（17.10节）。
- 探究JSlider（17.11节）。
- 在一个应用程序中显示多个窗口（17.12节）。

17.1 引言

图形用户界面（GUI）可以使系统对用户更友好且更易于使用。创建一个GUI需要创造性以及有关GUI组件如何工作的知识。在Java中，GUI组件非常灵活且功能多样，因而可以创建各种各样有用的用户界面。

许多Java集成开发环境都提供了可视化设计和开发GUI接口的工具。这样，可以用最少的编码快速为Java应用程序或applet将用户界面（UI）元素集合在一起。然而，任何工具都不是万能的。有时需要修改这些工具生成的程序。因此，在开始使用可视化工具之前，必须理解Java GUI程序设计的一些基本概念。

前几章简要介绍了一些GUI组件。本章详细地介绍经常会用到的GUI组件（参见图17-1）。（因为本章没有介绍什么新概念，所以教师可以将本章布置给学生自学。）

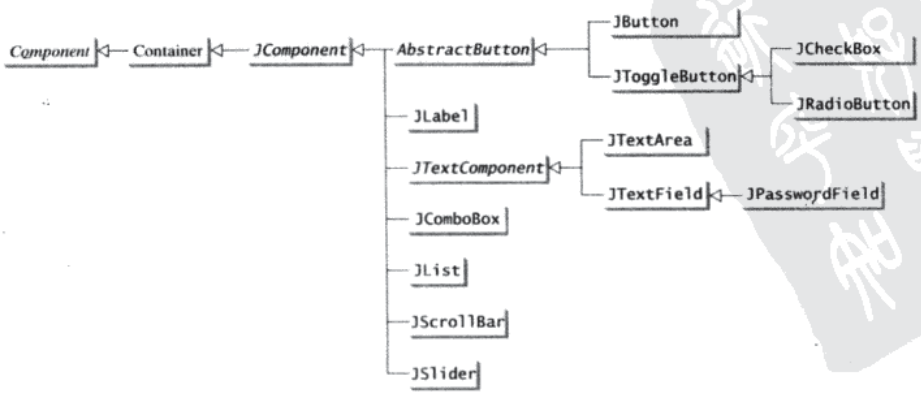


图17-1 这些Swing组件经常用来创建用户接口

注意 本书整篇都使用前缀jbt、jchk、jrb、jlbl、jtf、jpf、jta、jcbo、jlst、jscb和jsld来命名JButton、JCheckBox、JRadioButton、JLabel、JTextField、JPasswordField、JTextArea、JComboBox、JList、JScrollBar和JSlider的引用变量。

17.2 按钮

按钮（button）是点击时触发动作事件的组件。Swing提供了常规按钮、开关按钮、复选框按钮和单选按钮。这些按钮的公共特性在javax.swing.AbstractButton中定义，如图17-2所示。

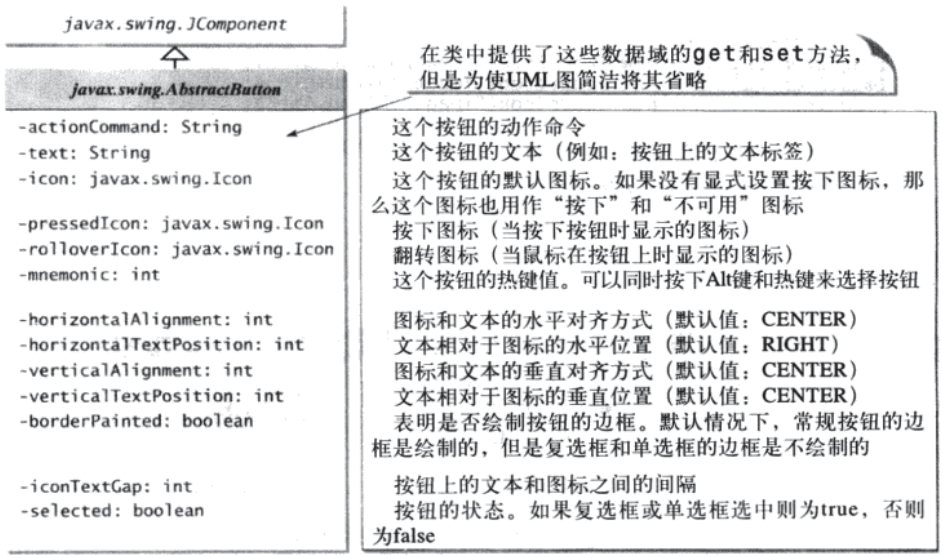


图17-2 AbstractButton定义了不同类型按钮的公共特性

本节介绍定义在JButton类中的常规按钮。JButton继承AbstractButton并且提供创建按钮的几个构造方法，如图17-3所示。

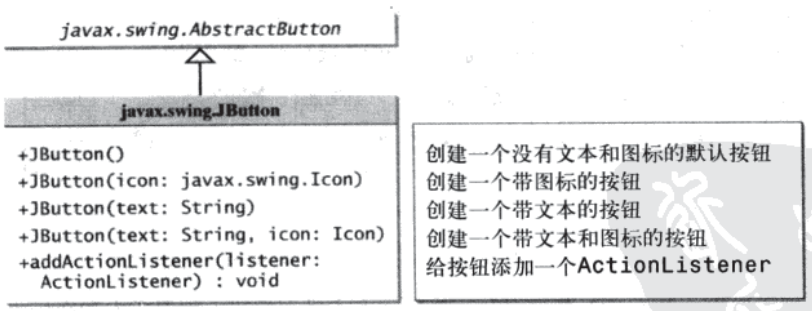


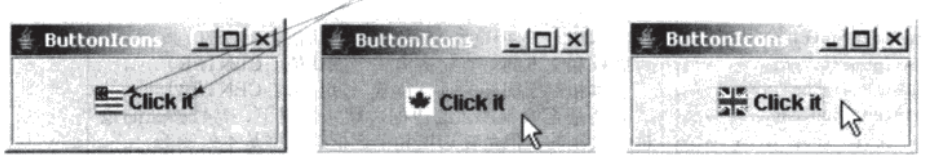
图17-3 JButton定义常规的下压按钮

17.2.1 图标、按下图标和翻转图标

每个常规按钮都有一个默认图标、一个按下图标和一个翻转图标。通常使用的是默认图标。其他的图标是为了显示特殊效果。当按下一个按钮的时候，显示按下图标，当鼠标移到按钮上而不按下时，显示翻转图标。例如，程序清单17-1将美国国旗作为常规图标显示，将加拿大国旗作为按下图标，将英国国旗作为翻转图标，如图17-4所示。

程序清单17-1 TestButtonIcons.java

```
1 import javax.swing.*;
2
3 public class TestButtonIcons extends JFrame  {
4   public static void main(String[] args) {
5     // Create a frame and set its properties
6     JFrame frame = new TestButtonIcons();
7     frame.setTitle("ButtonIcons");
8     frame.setSize(200, 100);
9     frame.setLocationRelativeTo(null); // Center the frame
10    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
11    frame.setVisible(true);
12  }
13
14  public TestButtonIcons() {
15    ImageIcon usIcon = new ImageIcon("image/usIcon.gif");
16    ImageIcon caIcon = new ImageIcon("image/caIcon.gif");
17    ImageIcon ukIcon = new ImageIcon("image/ukIcon.gif");
18
19    JButton jbt = new JButton("Click it", usIcon);
20    jbt.setPressedIcon(caIcon);
21    jbt.setRolloverIcon(ukIcon);
22
23    add(jbt);
24  }
25 }
```

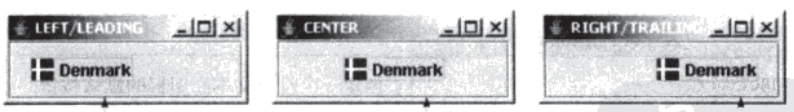


a）默认图标　　b）按下图标　　c）翻转图标

图17-4 一个按钮可以有多种类型的图标

17.2.2 对齐方式

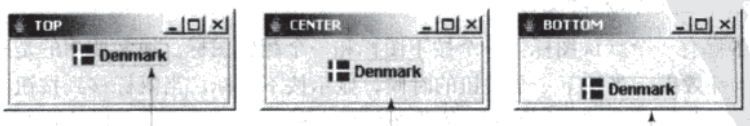
水平对齐（horizontal alignment）指定以什么样的水平方式在按钮上放置图标和文本。可以使用五个常量LEADING、LEFT、CENTER、RIGHT和TRAILING之一作为参数调用setHorizontalAlignment(int)方法来设置水平对齐方式，如图17-5所示。目前，LEADING和LEFT一样，TRAILING和RIGHT一样。今后可能会对它们加以区分。默认的水平对齐方式是SwingConstants.CENTER。



水平向左　　水平居中　　水平向右

图17-5 可以指定如何在按钮的水平方向上放置图标和文本

垂直对齐（vertical alignment）指定以什么样的垂直方式在按钮上放置文本和图标。可以使用三个常量TOP、CENTER和BOTTOM之一作为参数调用setVerticalAlignment(int)方法设置垂直对齐方式，如图17-6所示。默认的垂直对齐方式是SwingConstants.CENTER。



垂直向上　　垂直居中　　垂直向下

图17-6 可以指定如何在按钮的垂直方向上放置图标和文本

17.2.3 文本位置

水平文本位置（horizontal text position）指定文本相对于图标的水平位置。可以使用五个常量LEADING、LEFT、CENTER、RIGHT和TRAILING之一作为参数调用setHorizontalTextPosition(int)方法设置文本的水平位置，如图17-7所示。目前，LEADING和LEFT一样，TRAILING和RIGHT一样。Java今后可能会区分它们。默认的水平文本位置是SwingConstants.RIGHT。

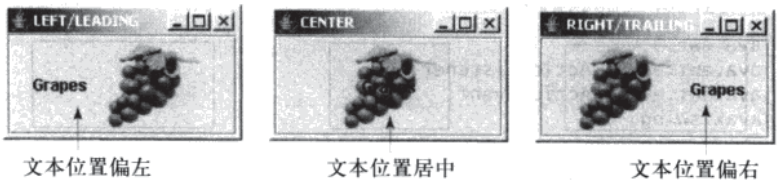


图17-7 可以指定文本相对于图标的水平位置

垂直文本位置（vertical text position）指定文本相对于图标的垂直位置。可以使用三个常量TOP、CENTER和BOTTOM之一作为参数调用setVerticalTextPosition(int)方法设置文本的垂直位置，如图17-8所示。默认的垂直文本位置是SwingConstants.CENTER。

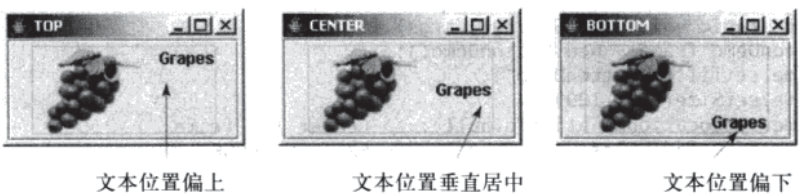


图17-8 可以指定文本相对于图标的垂直位置

注意 AbstractButton类中使用的常量LEFT、CENTER、RIGHT、LEADING、TRAILING、TOP和BOTTOM也可以被许多其他的Swing组件使用。这些常量集中定义在javax.swing.SwingConstants接口中。由于所有的Swing GUI组件都实现了SwingConstants，所以，可以通过SwingConstants或GUI组件引用这些常量。例如，SwingConstants.CENTER与JButton.CENTER是一样的。

JButton可以触发许多类型的事件，但是，通常需要添加监听器以响应ActionEvent事件。当按下一个按钮时，它就会触发一个ActionEvent事件。

17.2.4 使用按钮

本节给出一个程序，如程序清单17-2所示，它在面板上显示一条消息，然后使用两个按钮：<=和=>，在面板上向左或向右移动这个消息。用户界面的布局如图17-9所示。

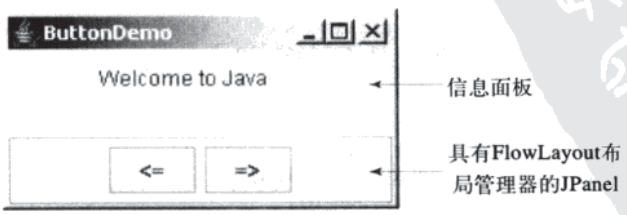


图17-9 点击按钮<=和=>使面板上的消息分别向左和向右移动

下面是程序的主要步骤：

1）创建用户界面。

创建一个显示消息的MessagePanel对象。MessagePanel类是在程序清单15-8中创建的。将MessagePanel对象放置在框架中央。在一个面板上创建两个按钮<=和=>。然后将这个面板放在框架的南边。

2）处理事件。

创建并注册监听器，根据是否点击向左按钮或向右按钮处理动作事件，以便向左或向右移动消息。

程序清单17-2 ButtonDemo.java

```
1 import java.awt.*;
2 import java.awt.event.ActionListener;
3 import java.awt.event.ActionEvent;
4 import javax.swing.*;
5
6 public class ButtonDemo extends JFrame {
7   // Create a panel for displaying message
8   protected MessagePanel messagePanel
9     = new MessagePanel("Welcome to Java");
10
11  // Declare two buttons to move the message left and right
12  private JButton jbtLeft = new JButton("<=");
13  private JButton jbtRight = new JButton("=>");
14
15  public static void main(String[] args) {
16    ButtonDemo frame = new ButtonDemo();
17    frame.setTitle("ButtonDemo");
18    frame.setSize(250, 100);
19    frame.setLocationRelativeTo(null); // Center the frame
20    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
21    frame.setVisible(true);
22  }
23
24  public ButtonDemo() {
25    // Set the background color of messagePanel
26    messagePanel.setBackground(Color.white);
27
28    // Create Panel jpButtons to hold two Buttons "<=" and "right =>"
29    JPanel jpButtons = new JPanel();
30    jpButtons.add(jbtLeft);
31    jpButtons.add(jbtRight);
32
33    // Set keyboard mnemonics
34    jbtLeft.setMnemonic('L');
35    jbtRight.setMnemonic('R');
36
37    // Set icons and remove text
38 //   jbtLeft.setIcon(new ImageIcon("image/left.gif"));
39 //   jbtRight.setIcon(new ImageIcon("image/right.gif"));
40 //   jbtLeft.setText(null);
41 //   jbtRight.setText(null);
42
43    // Set tool tip text on the buttons
44    jbtLeft.setToolTipText("Move message to left");
45    jbtRight.setToolTipText("Move message to right");
46
47    // Place panels in the frame
48    setLayout(new BorderLayout());
49    add(messagePanel, BorderLayout.CENTER);
50    add(jpButtons, BorderLayout.SOUTH);
51
52    // Register listeners with the buttons
53    jbtLeft.addActionListener(new ActionListener() {
54      public void actionPerformed(ActionEvent e) {
55        messagePanel.moveLeft();
```

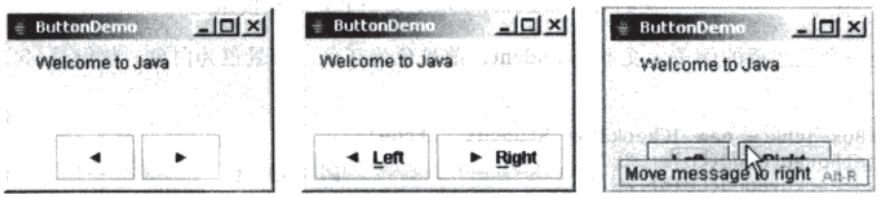
```
56        }
57      });
58      jbtRight.addActionListener(new ActionListener() {
59        public void actionPerformed(ActionEvent e) {
60          messagePanel.moveRight();
61        }
62      });
63    }
64 }
```

messagePanel（第8行）被刻意声明为protected，这样，它就可以被后面例子中的子类引用。

可以使用setIcon方法在按钮上设置一个图标图像。如果把第38～41行的以下代码中的注释去掉：

```
// jbtLeft.setIcon(new ImageIcon("image/left.gif"));
// jbtRight.setIcon(new ImageIcon("image/right.gif"));
// jbtLeft.setText(null);
// jbtRight.setText(null);
```

按钮上的文本就被图标替换，如图17-10a所示。"image/left.gif"放置在"c:\book\image\left.gif"中。注意，反斜杠是Windows文件路径的符号。而Java中应该使用斜杠。



a) b) c)

图17-10 可以在JButton上设置一个图标并且使用按钮的热键访问按钮

如果愿意，就可以在按钮上同时设置本文和图标，如图17-10b所示。默认情况下，文本和图标在水平方向和垂直方向都是居中的。

按钮也可以通过键盘上的热键来访问。由于把左边按钮的热键属性设置为'L'（第34行），所以，按下Alt+L键相当于点击<=按钮。如果将左边按钮的文本改为"Left"，而将右边按钮的文本改为"Right"，那么这两个按钮上文字中的L和R都会有下划线，如图17-10b所示。

每个按钮都有一个工具提示文本（第44～45行），它在鼠标放在按钮上但不被按下时出现，如图17-10c所示。

注意 由于MessagePanel没有在Java API中，所以应该将MessagePanel.java放在和ButtonDemo.java相同的目录中。

17.3 复选框

一个开关按钮（toggle button）就像是电灯开关一样会有两种状态。JToggleButton类继承AbstractButton并实现了一个开关按钮。通常，使用JToggleButton的子类JCheckBox和JRadioButton让用户在开或关这两种状态之间进行选择。本节介绍复选框按钮类JCheckBox，单选按钮类JRadioButton将在17.4节介绍。

JCheckBox继承AbstractButton类的所有属性，例如，text、icon、mnemonic、verticalAlignment、horizontalAlignment、horizontalTextPosition、verticalTextPosition和selected，而且还提供了创建复选框的几种构造方法，如图17-11所示。

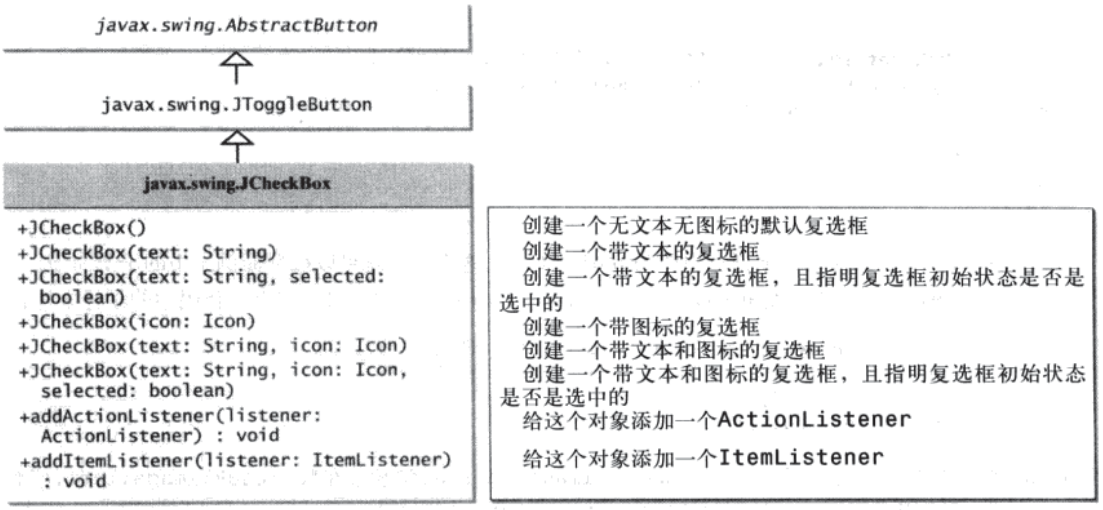


图17-11 JCheckBox定义一个复选框按钮

下面是一个复选框的例子，文本是Student，前景色为红色，背景色为白色，热键是'S'，而初始状态为选中的。

```
JCheckBox jchk = new JCheckBox("Student", true);
jchk.setForeground(Color.RED);
jchk.setBackground(Color.WHITE);
jchk.setMnemonic('S');
```

☑ Student

当点击（选中或取消）一个复选框时，它会触发一个ItemEvent事件，然后触发一个ActionEvent事件。要了解复选框是否被选中，使用isSelected()方法。

程序清单17-3给出的程序将三个名为Centered、Bold和Italic的复选框添加到前面的例子中。用户使用它们指定信息是否居中、是否为粗体或斜体，如图17-12所示。

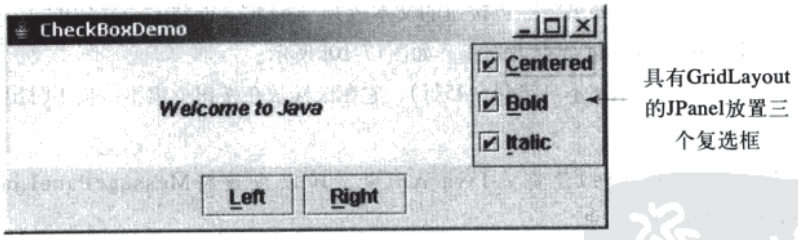


图17-12 添加三个复选框以指定如何显示消息

编写这个程序至少有两种方法。第一种是修改上面的ButtonDemo类，插入代码来添加复选框并处理它们的事件。第二种是创建一个扩展自ButtonDemo类的子类。第一种方法的实现留作练习题。程序清单17-3给出实现第二种方法的代码。

CheckBoxDemo类扩展ButtonDemo类，并且增加三个复选框来控制如何显示消息。当构造一个CheckBoxDemo（第12行）时，调用其父类的无参数构造方法，因此，不必改写已经在ButtonDemo构造方法中的代码。

程序清单17-3 CheckBoxDemo.java

```
1 import java.awt.*;
2 import java.awt.event.*;
3 import javax.swing.*;
4
5 public class CheckBoxDemo extends ButtonDemo {
6   // Create three check boxes to control the display of message
7   private JCheckBox jchkCentered = new JCheckBox("Centered");
8   private JCheckBox jchkBold = new JCheckBox("Bold");
9   private JCheckBox jchkItalic = new JCheckBox("Italic");
10
11  public static void main(String[] args) {
12    CheckBoxDemo frame = new CheckBoxDemo();
13    frame.setTitle("CheckBoxDemo");
14    frame.setSize(500, 200);
15    frame.setLocationRelativeTo(null); // Center the frame
16    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
17    frame.setVisible(true);
18  }
19
20  public CheckBoxDemo() {
21    // Set mnemonic keys
22    jchkCentered.setMnemonic('C');
23    jchkBold.setMnemonic('B');
24    jchkItalic.setMnemonic('I');
25
26    // Create a new panel to hold check boxes
27    JPanel jpCheckBoxes = new JPanel();
28    jpCheckBoxes.setLayout(new GridLayout(3, 1));
29    jpCheckBoxes.add(jchkCentered);
30    jpCheckBoxes.add(jchkBold);
31    jpCheckBoxes.add(jchkItalic);
32    add(jpCheckBoxes, BorderLayout.EAST);
33
34    // Register listeners with the check boxes
35    jchkCentered.addActionListener(new ActionListener() {
36      public void actionPerformed(ActionEvent e) {
37        messagePanel.setCentered(jchkCentered.isSelected());
38      }
39    });
40    jchkBold.addActionListener(new ActionListener() {
41      public void actionPerformed(ActionEvent e) {
42        setNewFont();
43      }
44    });
45    jchkItalic.addActionListener(new ActionListener() {
46      public void actionPerformed(ActionEvent e) {
47        setNewFont();
48      }
49    });
50  }
51
52  private void setNewFont() {
53    // Determine a font style
54    int fontStyle = Font.PLAIN;
55    fontStyle += (jchkBold.isSelected() ? Font.BOLD : Font.PLAIN);
56    fontStyle += (jchkItalic.isSelected() ? Font.ITALIC : Font.PLAIN);
57
58    // Set font for the message
59    Font font = messagePanel.getFont();
60    messagePanel.setFont(
61      new Font(font.getName(), fontStyle, font.getSize()));
62  }
63 }
```

当选中或取消选中复选框时，调用监听器的actionPerformed方法处理这个事件。选中或取消选中Centered复选框时，MessagePanel类的centered属性被设置为true或false。

使用getName()和getSize()方法，从messagePanel.getFont()中获取MessagePanel当前使用的字体名和字体大小。字体风格（Font.BOLD和Font.ITALIC）在复选框中指定。如果没有选择字体风格，那么字体风格就是Font.PLAIN。可以通过将选择的表示字体的整数加在一起组合成字体风格。

分别将C、B、I设置为复选框Centered、Bold和Italic的热键（第22～24行）。可以使用鼠标或快捷键来选择一个复选框。

MessagePanel类中继承了定义在Component类中的setFont方法（第60行）。这个方法自动调用repaint方法。调用MessagePanel中的setFont方法可以自动重新绘制这条消息。

点击复选框时，会触发ActionEvent事件和ItemEvent事件。处理ActionEvent事件或ItemEvent事件都可以重新显示这个消息。这个例子处理ActionEvent事件。如果希望处理ItemEvent事件，可以为ItemEvent创建一个监听器并且将其注册到一个复选框。这个监听器必须实现itemStateChanged处理器来处理一个ItemEvent。例如，下面的代码向jchkCentered注册一个ItemListener：

```
// To listen for ItemEvent
jchkCentered.addItemListener(new ItemListener() {
  /** Handle ItemEvent */
  public void itemStateChanged(ItemEvent e) {
    messagePanel.setCentered(jchkCentered.isSelected());
  }
});
```

17.4 单选按钮

单选按钮（radio button）也称为*选项按钮*（option button），它可以让用户从一组选项中选择一个单一的条目。从外观上看，单选按钮类似于复选框。不过不管选中与否，复选框都是方形的，而单选按钮都是圆的，不论它是填充的（选中时）还是空白的（未选中时）。

JRadioButton类继承AbstractButton类，同时提供一些创建单选按钮的构造方法，如图17-13所示。这些构造方法类似于JCheckBox的构造方法。

下面是一个单选按钮的例子，文本为Student，前景色为红色，背景色为白色，热键为S，而初始状态为选中。

```
JRadioButton jrb = new JRadioButton("Student", true);
jrb.setForeground(Color.RED);
jrb.setBackground(Color.WHITE);
jrb.setMnemonic('S');
```

为了将单选按钮放在一组，需要创建java.swing.ButtonGroup的一个实例，并且使用add方法把这些单选按钮添加到这个实例中，如下所示：

```
ButtonGroup group = new ButtonGroup();
group.add(jrb1);
group.add(jrb2);
```

这个代码给单选按钮jrb1和jrb2创建了一个单选按钮组，这样，单选按钮jrb1和jrb2在选择时就是互斥的。如果没有创建这个组，jrb1与jrb2就是相互独立的。

注意 ButtonGroup不是java.awt.Component的子类，所以，ButtonGroup对象不能添加到容器中。

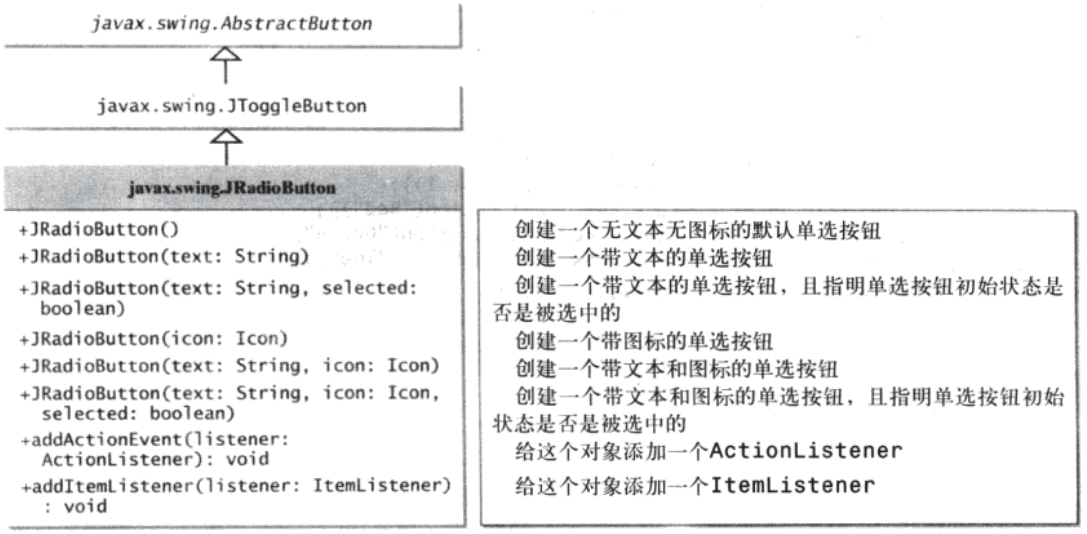


图17-13 JRadioButton定义一个单选按钮

当改变一个单选按钮（选中或者取消选中）时，它会触发一个ItemEvent事件以及一个ActionEvent事件。要判断一个单选按钮是否被选中，使用isSelected()方法。

程序清单17-4给出的程序是给前面的例子中添加三个名为Red、Green和Blue的单选按钮，让用户选择消息的颜色，如图17-14所示。

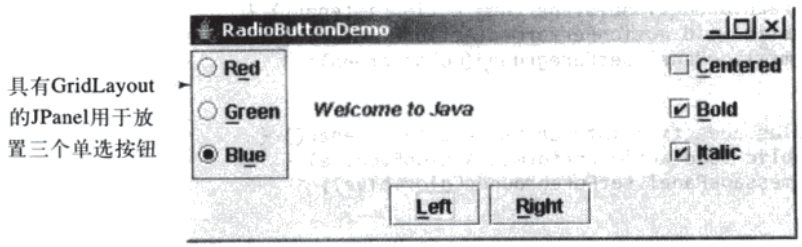


图17-14 添加三个单选按钮以指定消息的颜色

至少有两种方法编写这个程序。第一种是修改前面的CheckBoxDemo类，插入添加单选按钮以及处理它们的事件的代码。第二种是创建一个扩展CheckBoxDemo的子类。程序清单17-4给出实现第二种方法的代码。

程序清单17-4 RadioButtonDemo.java

```
1 import java.awt.*;
2 import java.awt.event.*;
3 import javax.swing.*;
4
5 public class RadioButtonDemo extends CheckBoxDemo {
6   // Declare radio buttons
7   private JRadioButton jrbRed, jrbGreen, jrbBlue;
8
9   public static void main(String[] args) {
10    RadioButtonDemo frame = new RadioButtonDemo();
11    frame.setSize(500, 200);
12    frame.setLocationRelativeTo(null); // Center the frame
13    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
14      frame.setTitle("RadioButtonDemo");
15      frame.setVisible(true);
16    }
17
18    public RadioButtonDemo() {
19      // Create a new panel to hold check boxes
20      JPanel jpRadioButtons = new JPanel();
21      jpRadioButtons.setLayout(new GridLayout(3, 1));
22      jpRadioButtons.add(jrbRed = new JRadioButton("Red"));
23      jpRadioButtons.add(jrbGreen = new JRadioButton("Green"));
24      jpRadioButtons.add(jrbBlue = new JRadioButton("Blue"));
25      add(jpRadioButtons, BorderLayout.WEST);
26
27      // Create a radio-button group to group three buttons
28      ButtonGroup group = new ButtonGroup();
29      group.add(jrbRed);
30      group.add(jrbGreen);
31      group.add(jrbBlue);
32
33      // Set keyboard mnemonics
34      jrbRed.setMnemonic('E');
35      jrbGreen.setMnemonic('G');
36      jrbBlue.setMnemonic('U');
37
38      // Register listeners for radio buttons
39      jrbRed.addActionListener(new ActionListener() {
40        public void actionPerformed(ActionEvent e) {
41          messagePanel.setForeground(Color.red);
42        }
43      });
44      jrbGreen.addActionListener(new ActionListener() {
45        public void actionPerformed(ActionEvent e) {
46          messagePanel.setForeground(Color.green);
47        }
48      });
49      jrbBlue.addActionListener(new ActionListener() {
50        public void actionPerformed(ActionEvent e) {
51          messagePanel.setForeground(Color.blue);
52        }
53      });
54
55      // Set initial message color to blue
56      jrbBlue.setSelected(true);
57      messagePanel.setForeground(Color.blue);
58    }
59  }
```

RadioButtonDemo类扩展CheckBoxDemo类，并且增加三个指定消息颜色的单选按钮。当点击一个单选按钮时，它的动作事件监听器设置messagePanel中对应的前景色。

R和B已经被设置为Right按钮和Bold复选框的热键。为了避免冲突，将E、G和U分别设置为单选按钮Red、Green和Blue的热键（第34～36行）。

程序创建一个ButtonGroup，并且将三个JRadioButton的实例（jrbRed、jrbGreen和jrbBlue）放到这个组中（第28～31行）。

当选中或取消选中一个单选按钮时，它会触发ActionEvent事件和ItemEvent事件。可以处理ActionEvent或ItemEvent来选择一种颜色。这个例子处理ActionEvent事件。请使用ItemEvent事件改写这些代码作为练习题。

17.5 标签

标签（label）是一个显示小段文字、一幅图像或同时显示两者的区域。它经常用来给其他组件（通

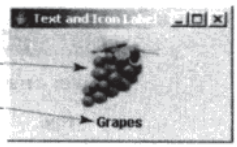
常为文本域）做标记。图17-15列出了JLabel中的构造方法和方法。

JLabel继承了JComponent类的所有属性，并具有与JButton类相似的许多属性，例如，text、icon、horizontalAlignment、verticalAlignment、horizontalTextPosition、VerticalTextPosition和iconTextGap。例如，下述代码显示了一个带文本和图标的标签：

```
// Create an image icon from an image file
ImageIcon icon = new ImageIcon("image/grapes.gif");

// Create a label with a text, an icon,
// with centered horizontal alignment
JLabel jlbl = new JLabel("Grapes", icon, SwingConstants.CENTER);

//Set label's text alignment and gap between text and icon
jlbl.setHorizontalTextPosition(SwingConstants.CENTER);
jlbl.setVerticalTextPosition(SwingConstants.BOTTOM);
jlbl.setIconTextGap(5);
```



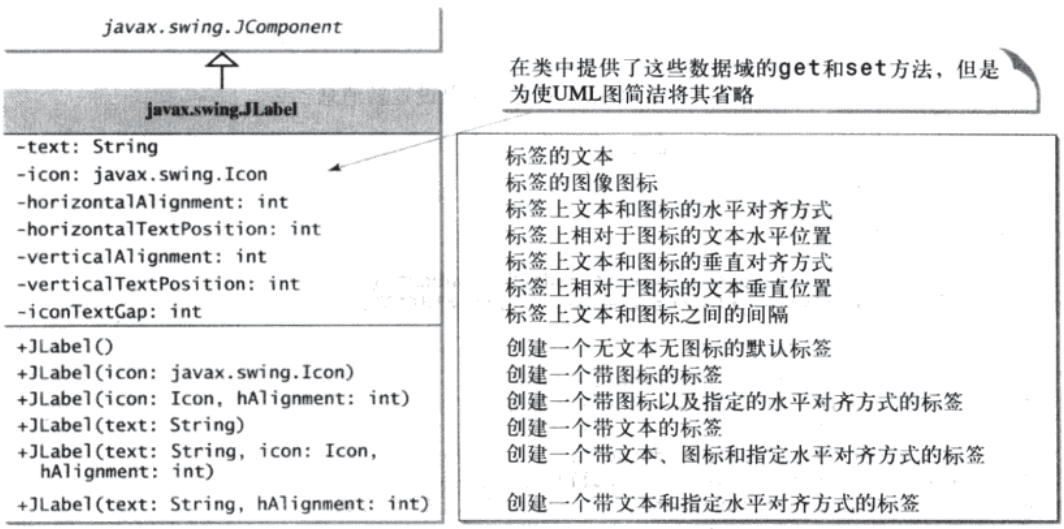


图17-15 JLabel显示文本或图标或两者都显示

17.6 文本域

文本域（text field）可用于输入或显示一个字符串。JTextField类是JTextComponent类的一个子类。图17-16列出JTextField的构造方法和方法。

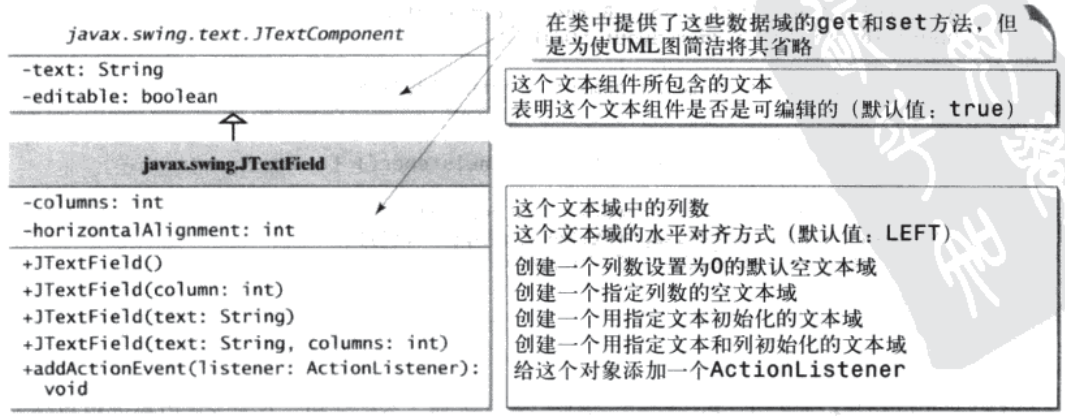


图17-16 JTextField允许输入或显示一个字符串

JTextField类继承JTextComponent类，而JTextComponent类继承JComponent类。下面是创建一个文本域的例子，它的前景色为红色，对齐方式为水平向右对齐方式：

```
JTextField jtfMessage = new JTextField("T-Strom");
jtfMessage.setForeground(Color.RED);
jtfMessage.setHorizontalAlignment(SwingConstants.RIGHT);
```

把光标移动到文本域，然后按下回车键时，会触发一个ActionEvent事件。

程序清单17-5给前一个例子添加一个文本域，允许用户设置一条新消息，如图17-17所示。

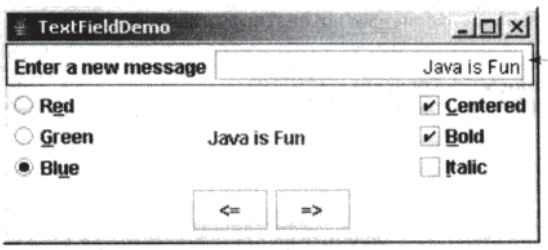


图17-17 添加标签和文本域以设置新消息

程序清单17-5 TextFieldDemo.java

```
1 import java.awt.*;
2 import java.awt.event.*;
3 import javax.swing.*;
4
5 public class TextFieldDemo extends RadioButtonDemo {
6   private JTextField jtfMessage = new JTextField(10);
7
8   /** Main method */
9   public static void main(String[] args) {
10    TextFieldDemo frame = new TextFieldDemo();
11    frame.pack();
12    frame.setTitle("TextFieldDemo");
13    frame.setLocationRelativeTo(null); // Center the frame
14    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
15    frame.setVisible(true);
16  }
17
18  public TextFieldDemo() {
19    // Create a new panel to hold label and text field
20    JPanel jpTextField = new JPanel();
21    jpTextField.setLayout(new BorderLayout(5, 0));
22    jpTextField.add(
23      new JLabel("Enter a new message"), BorderLayout.WEST);
24    jpTextField.add(jtfMessage, BorderLayout.CENTER);
25    add(jpTextField, BorderLayout.NORTH);
26
27    jtfMessage.setHorizontalAlignment(JTextField.RIGHT);
28
29    // Register listener
30    jtfMessage.addActionListener(new ActionListener() {
31      /** Handle ActionEvent */
32      public void actionPerformed(ActionEvent e) {
33        messagePanel.setMessage(jtfMessage.getText());
34        jtfMessage.requestFocusInWindow();
35      }
36    });
37  }
38 }
```

TextFieldDemo类扩展RadioButtonDemo类，并且添加一个标签和一个本文域以允许用户输入新

消息。在文本域内设置一条新消息，然后按下回车键，就会显示一条新消息。在文本域上按下回车键就会触发一个动作事件。监听器在messagePanel中设置一条新消息（第33行）。

方法pack()可以按照放在框架里的组件的大小自动调整框架的尺寸。

Component类中定义的requestFocusInWindow()方法（第34行）请求组件接收输入焦点。这样，jtfMessage.requestFocusInWindow()方法（第34行）请求将输入聚焦于jtfMessage上。所以，在调用actionPerformed方法后，将会看到光标在jtfMessage上。

注意 如果使用一个文本域输入密码，那么可以使用JPasswordField替换JTextField。JPasswordField扩展JTextField并且用回显字符（例如，******）隐藏输入的文本。默认情况下，回显字符是*。可以使用setEchoChar(char)方法指定一个新的回显字符。

17.7 文本区域

如果想让用户输入多行文本，必须创建JTextField的几个实例。一个更好的可选择的办法是使用JTextArea，它允许用户输入多行文本。图17-18列出JTextArea的构造方法和方法。

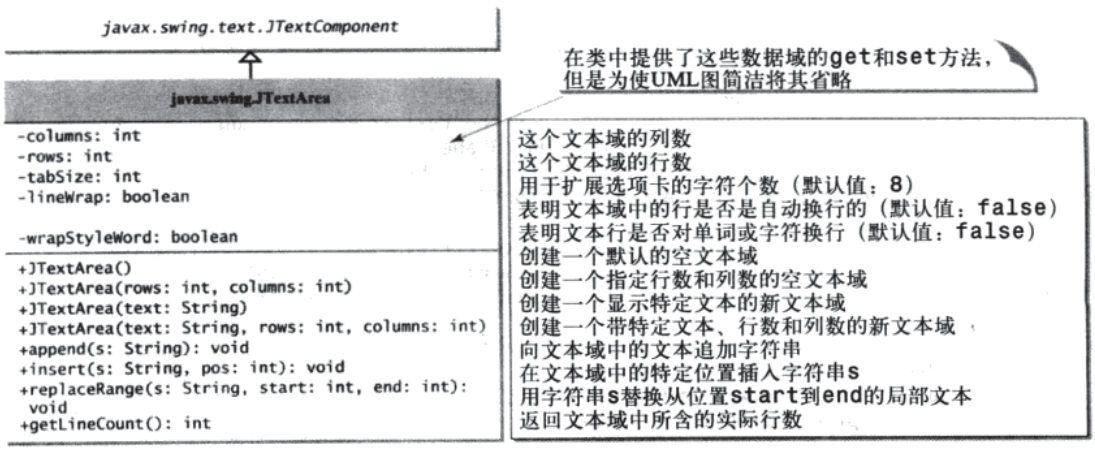


图17-18 JTextArea允许输入或显示多行字符

同JTextField类一样，JTextArea类也继承自JTextComponent类，并且包含方法getText、setText、isEditable以及setEditable。这里是一个创建5行20列文本域的例子，按单词自动换行，前景色为红色，字体为Courier，字型为粗体，字号大小为20像素。

```
JTextArea jtaNote = new JTextArea("This is a text area", 5, 20);
jtaNote.setLineWrap(true);
jtaNote.setWrapStyleWord(true);
jtaNote.setForeground(Color.red);
jtaNote.setFont(new Font("Courier", Font.BOLD, 20));
```

JTextArea不处理滚动，但是，可以创建一个JScrollPane的对象来保存JTextArea的一个实例，让JScrollPane处理JTextArea的滚动，如下所示：

```
// Create a scroll pane to hold text area
JScrollPane scrollPane = new JScrollPane(jta = new JTextArea());
add(scrollPane, BorderLayout.CENTER);
```

程序清单17-7给出的程序是在一个标签上显示图像和文本，在一个文本区域中显示文本，如图17-19所示。

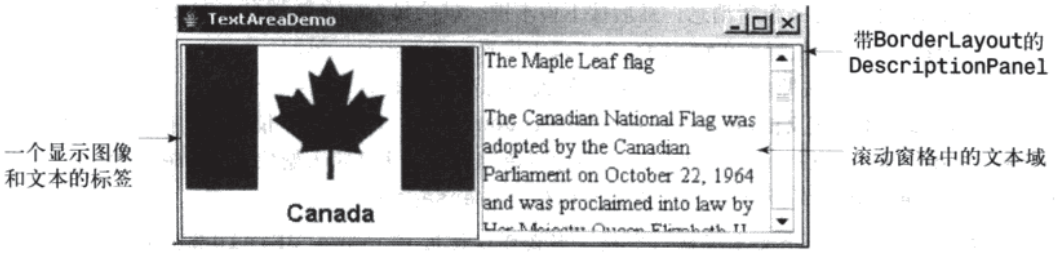


图17-19 程序在标签中显示图像和标题，且在文本区域中显示文本

程序清单17-6 DescriptionPanel.java

```
1 import javax.swing.*;
2 import java.awt.*;
3
4 public class DescriptionPanel extends JPanel {
5   /** Label for displaying an image icon and a text */
6   private JLabel jlblImageTitle = new JLabel();
7
8   /** Text area for displaying text */
9   private JTextArea jtaDescription = new JTextArea();
10
11  public DescriptionPanel() {
12    // Center the icon and text and place the text under the icon
13    jlblImageTitle.setHorizontalAlignment(JLabel.CENTER);
14    jlblImageTitle.setHorizontalTextPosition(JLabel.CENTER);
15    jlblImageTitle.setVerticalTextPosition(JLabel.BOTTOM);
16
17    // Set the font in the label and the text field
18    jlblImageTitle.setFont(new Font("SansSerif", Font.BOLD, 16));
19    jtaDescription.setFont(new Font("Serif", Font.PLAIN, 14));
20
21    // Set lineWrap and wrapStyleWord true for the text area
22    jtaDescription.setLineWrap(true);
23    jtaDescription.setWrapStyleWord(true);
24    jtaDescription.setEditable(false);
25
26    // Create a scroll pane to hold the text area
27    JScrollPane scrollPane = new JScrollPane(jtaDescription);
28
29    // Set BorderLayout for the panel, add label and scrollpane
30    setLayout(new BorderLayout(5, 5));
31    add(scrollPane, BorderLayout.CENTER);
32    add(jlblImageTitle, BorderLayout.WEST);
33  }
34
35  /** Set the title */
36  public void setTitle(String title) {
37    jlblImageTitle.setText(title);
38  }
39
40  /** Set the image icon */
41  public void setImageIcon(ImageIcon icon) {
42    jlblImageTitle.setIcon(icon);
43  }
44
45  /** Set the text description */
46  public void setDescription(String text) {
47    jtaDescription.setText(text);
48  }
49 }
```

这个程序有以下几个主要的步骤：

1）扩展JPanel，创建一个名为DescriptionPanel的类，如程序清单17-6所示。这个类中包含一个在滚动窗格中的文本域，还包含一个显示图像图标和标题的标签。现在的例子使用这个类，在后面的例子中还会复用这个类。

2）扩展JFrame，创建一个名为TextAreaDemo的类，如程序清单17-7所示。创建DescriptionPanel的一个实例，并且将它添加到框架的中心位置。DescriptionPanel和TextAreaDemo之间的关系如图17-20所示。

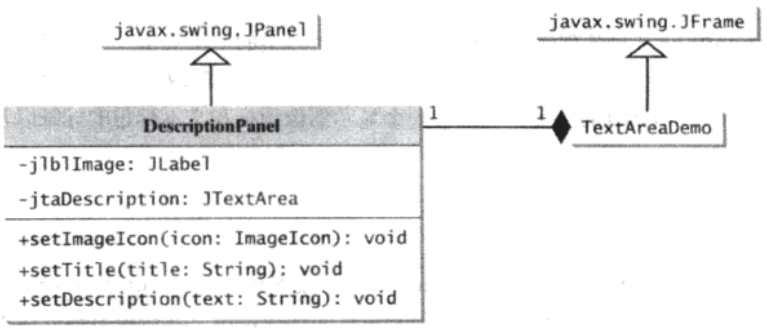


图17-20 TextAreaDemo类用DescriptionPanel类显示国旗的图像、标题及其文本描述

文本域在JScrollPane（第27行）内，它为这个文本域提供滚动功能。如果文本的尺寸大于文本域的物理尺寸，就会自动出现滚动条；如果文本被删除之后剩余的文本尺寸小于文本域尺寸，那么滚动条就会消失。

属性lineWrap设置为true（第22行），这样，当文本超出一行时就会自动换行。属性wrapStyleWord设置为true（第23行），这样，就按单词而不是按字符换行。文本域设置为不可编辑的（第24行），所以不能在文本域中编辑描述文字。

在这个例子中，没必要为DescriptionPanel创建一个独立的类。而且，这个创建出来的类可以在17.8节中复用，17.8节将使用它为多种图像显示描述面板。

程序清单17-7 TextAreaDemo.java

```
 1 import java.awt.*;
 2 import javax.swing.*;
 3
 4 public class TextAreaDemo extends JFrame {
 5   // Declare and create a description panel
 6   private DescriptionPanel descriptionPanel = new DescriptionPanel();
 7
 8   public static void main(String[] args) {
 9     TextAreaDemo frame = new TextAreaDemo();
10     frame.pack();
11     frame.setLocationRelativeTo(null); // Center the frame
12     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
13     frame.setTitle("TextAreaDemo");
14     frame.setVisible(true);
15   }
16
17   public TextAreaDemo() {
18     // Set title, text and image in the description panel
19     descriptionPanel.setTitle("Canada");
20     String description = "The Maple Leaf flag \n\n" +
21       "The Canadian National Flag was adopted by the Canadian " +
22       "Parliament on October 22, 1964 and was proclaimed into law " +
23       "by Her Majesty Queen Elizabeth II (the Queen of Canada) on " +
24       "February 15, 1965. The Canadian Flag (colloquially known " +
25       "as The Maple Leaf Flag) is a red flag of the proportions " +
```

```
26        "two by length and one by width, containing in its center a " +
27        "white square, with a single red stylized eleven-point " +
28        "maple leaf centered in the white square.";
29      descriptionPanel.setDescription(description);
30      descriptionPanel.setImageIcon(new ImageIcon("image/ca.gif"));
31
32      // Add the description panel to the frame
33      setLayout(new BorderLayout());
34      add(descriptionPanel, BorderLayout.CENTER);
35    }
36 }
```

程序TextAreaDemo类只是创建了DescriptionPanel类的一个实例（第6行），然后在描述面板内设置描述面板的标题（第19行）、图像（第30行）以及文本（第29行）。DescriptionPanel是JPanel.DescriptionPanel的子类，它包含一个显示图像图标和文本标题的标签以及显示关于图像的描述的一个文本区域。

17.8 组合框

组合框（combo box）也称为选择列表（choice list）或下拉式列表（drop-down list），它包含一个条目列表，用户能够从中进行选择。使用它可以限制用户的选择范围，并避免对输入数据有效性进行繁琐的检查。图17-21列出在JComboBox类中一些常用的构造方法和方法。

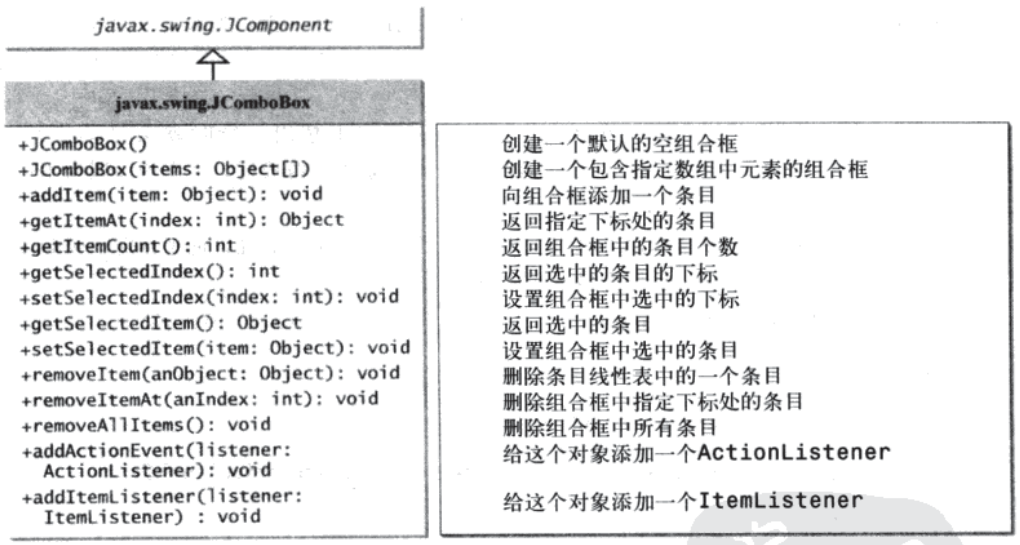


图17-21 JComboBox从条目集合中选择一个条目

下面的语句创建一个有四个条目的组合框，前景色为红色，背景色为白色，然后选中第二个条目：

```
JComboBox jcb = new JComboBox(new Object[]
  {"Item 1", "Item 2", "Item 3", "Item 4"});
jcb.setForeground(Color.red);
jcb.setBackground(Color.white);
jcb.setSelectedItem("Item 2");
```

JComboBox可以在很多其他事件中激发ActionEvent和ItemEvent事件。每当选中一个条目时，就会触发一个ActionEvent。每当选中一个新条目时，JComboBox就会激发ItemEvent事件两次，一次是取消前一个选择的条目，另一次是选中当前选择的条目。注意，如果当前条目被重新选择，那么就没有ItemEvent触发。要响应ItemEvent事件，需要实现itemStateChanged(ItemEvent e)处理器来

处理选择。要从一个JComboBox菜单中获取数据，可以使用getSelectedItem()方法返回当前已选定的条目，或者使用e.getItem()方法从itemStateChanged(ItemEvent e)处理器中获取条目。

程序清单17-8给出的程序使用户通过选择组合框中的国家来查看某个国家国旗的图像及其描述，如图17-22所示。

程序清单17-8 ComboBoxDemo.java

```
1  import java.awt.*;
2  import java.awt.event.*;
3  import javax.swing.*;
4
5  public class ComboBoxDemo extends JFrame {
6    // Declare an array of Strings for flag titles
7    private String[] flagTitles = {"Canada", "China", "Denmark",
8      "France", "Germany", "India", "Norway", "United Kingdom",
9      "United States of America"};
10
11   // Declare an ImageIcon array for the national flags of 9 countries
12   private ImageIcon[] flagImage = {
13     new ImageIcon("image/ca.gif"),
14     new ImageIcon("image/china.gif"),
15     new ImageIcon("image/denmark.gif"),
16     new ImageIcon("image/fr.gif"),
17     new ImageIcon("image/germany.gif"),
18     new ImageIcon("image/india.gif"),
19     new ImageIcon("image/norway.gif"),
20     new ImageIcon("image/uk.gif"),
21     new ImageIcon("image/us.gif")
22   };
23
24   // Declare an array of strings for flag descriptions
25   private String[] flagDescription = new String[9];
26
27   // Declare and create a description panel
28   private DescriptionPanel descriptionPanel = new DescriptionPanel();
29
30   // Create a combo box for selecting countries
31   private JComboBox jcbo = new JComboBox(flagTitles);
32
33   public static void main(String[] args) {
34     ComboBoxDemo frame = new ComboBoxDemo();
35     frame.pack();
36     frame.setTitle("ComboBoxDemo");
37     frame.setLocationRelativeTo(null); // Center the frame
38     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
39     frame.setVisible(true);
40   }
41
42   public ComboBoxDemo() {
43     // Set text description
44     flagDescription[0] = "The Maple Leaf flag \n\n" +
45       "The Canadian National Flag was adopted by the Canadian " +
46       "Parliament on October 22, 1964 and was proclaimed into law " +
47       "by Her Majesty Queen Elizabeth II (the Queen of Canada) on " +
48       "February 15, 1965. The Canadian Flag (colloquially known " +
49       "as The Maple Leaf Flag) is a red flag of the proportions " +
50       "two by length and one by width, containing in its center a " +
51       "white square, with a single red stylized eleven-point " +
52       "maple leaf centered in the white square.";
53     flagDescription[1] = "Description for China ... ";
54     flagDescription[2] = "Description for Denmark ... ";
```

```
55      flagDescription[3] = "Description for France ... ";
56      flagDescription[4] = "Description for Germany ... ";
57      flagDescription[5] = "Description for India ... ";
58      flagDescription[6] = "Description for Norway ... ";
59      flagDescription[7] = "Description for UK ... ";
60      flagDescription[8] = "Description for US ... ";
61
62      // Set the first country (Canada) for display
63      setDisplay(0);
64
65      // Add combo box and description panel to the list
66      add(jcbo, BorderLayout.NORTH);
67      add(descriptionPanel, BorderLayout.CENTER);
68
69      // Register listener
70      jcbo.addItemListener(new ItemListener() {
71        /** Handle item selection */
72        public void itemStateChanged(ItemEvent e) {
73          setDisplay(jcbo.getSelectedIndex());
74        }
75      });
76    }
77
78    /** Set display information on the description panel */
79    public void setDisplay(int index) {
80      descriptionPanel.setTitle(flagTitles[index]);
81      descriptionPanel.setImageIcon(flagImage[index]);
82      descriptionPanel.setDescription(flagDescription[index]);
83    }
84  }
```

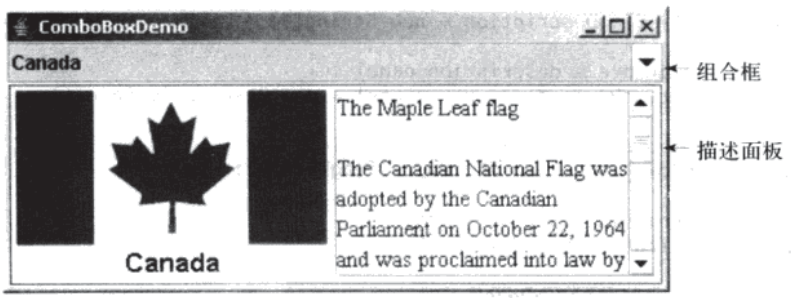


图17-22 选中组合框中的一个国家后，就显示该国的信息，包括国旗的图像及其描述

下面是这个程序的主要步骤：

1）创建用户界面。

创建一个组合框，将国家名作为选择值。创建一个`DescriptionPanel`对象。`DescriptionPanel`类在前一个例子中介绍过。将组合框放置在框架的北区，而将描述面板放置在框架的中央。

2）处理事件。

创建一个监听器来实现`itemStateChanged`处理器，在描述面板中为所选择的国家名设置国旗名、国旗图像及其描述。

监听器监听组合框触发的`ItemEvent`事件，并实现`ItemListener`（第70～75行）。可以不用`ItemEvent`事件，而是使用`ActionEvent`事件来改写这个程序来处理组合框条目的选择。

程序将国旗信息存储在三个数组：`flagTitles`、`flagImage`和`flagDescription`（第7～25行）中。数组`flagTitles`存放九个国家的名称，数组`flagImage`存放九个国家国旗的图像，数组`flagDescription`存放对这些国旗的描述。

程序创建DescriptionPanel类的一个实例（第28行），该类在程序清单17-6中。程序以数组flagTitles中的值为初值创建一个组合框（第31行）。当用户选择组合框中的一项后，执行itemStateChanged处理器，确定选项的序号并在面板上设置相应的国旗名、国旗图像及国旗描述。

17.9 列表框

列表框（list）是一个组件，它完成的功能与组合框基本相同，但它允许用户选择一个或多个值。Swing组件JList的功能非常丰富。图17-23列出了JList中一些常用的构造方法和方法。

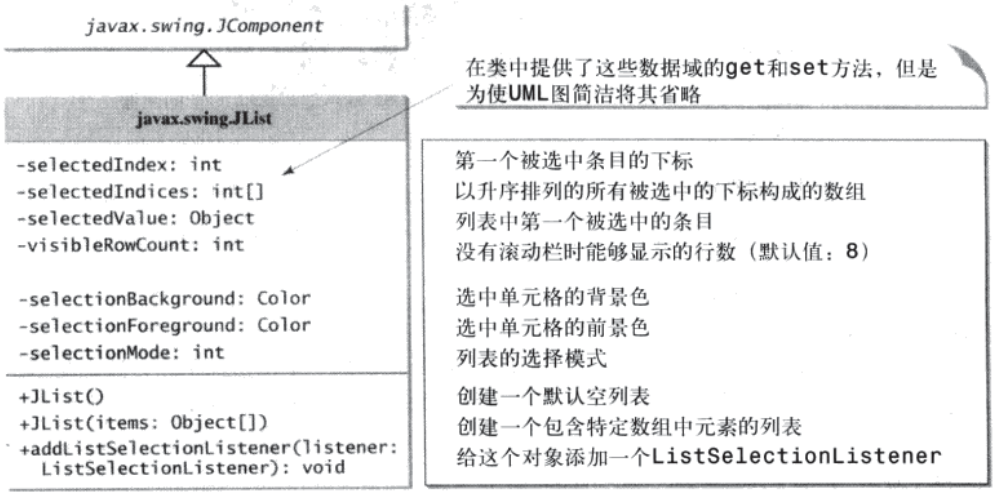


图17-23 JList允许从一个条目集合中选择多个条目

选择模式属性selectionMode是三个在javax.swing.ListSelectionModel类中定义的数值之一（SINGLE_SELECTION、SINGLE_INTERVAL_SELECTION、MULTIPLE_INTERVAL_SELECTION），它们分别表示选择的是单项选择、单区间选择还是多区间选择。单项选择只允许选择一项。单区间选择允许选择多项，但是选定的项必须是连续的。多区间选择允许选择多种连续项，没有限制，如图17-24所示。此属性的默认值为MULTIPLE_INTERVAL_SELECTION。

a) 单项选择 b) 单区间选择 c) 多区间选择

图17-24 JList有三种选择模式：单项选择、单区间选择和多区间选择

下面的语句创建有6个条目的列表框，前景色为红色，背景色为白色，选定单元格的前景色为粉色，选定单元格的背景色为黑色，可见行数为4行。

```
JList jlst = new JList(new Object[]
  {"Item 1", "Item 2", "Item 3", "Item 4", "Item 5", "Item 6"});
jlst.setForeground(Color.RED);
jlst.setBackground(Color.WHITE);
jlst.setSelectionForeground(Color.PINK);
jlst.setSelectionBackground(Color.BLACK);
jlst.setVisibleRowCount(4);
```

列表不能自动滚动。为了使一个列表能够滚动，创建一个滚动窗格并将该列表框添加到这个窗格中。

JList触发javax.swing.event.ListSelectionEvent事件，通知用于处理选择的监听器。监听器必须实现javax.swing.event.ListSelectionListener接口中的valueChanged处理器来处理这个事件。

程序清单17-9给出的程序让用户在列表中选择国家，并且在标签中显示选中国家的国旗。图17-25显示该程序的一个运行示例。

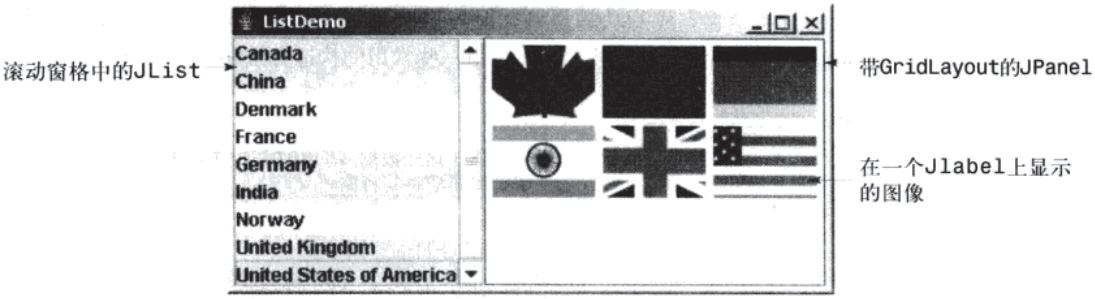


图17-25 选中列表框中的国家后，在标签中显示对应国旗的图像

下面是这个程序的主要步骤：

1）创建用户界面。

创建有九个国家名的列表作为选择值，然后将这个列表框放到一个滚动窗格中。将滚动窗格放到框架的西边。创建九个标签用来显示这九个国家的国旗图像。将标签放在面板中，再将面板放在框架的中央。

2）处理事件。

创建一个监听器，实现ListSelectionListener接口中的valueChanged方法，在标签中放置选定国家的国旗图像。

程序清单17-9 ListDemo.java

```
1 import java.awt.*;
2 import javax.swing.*;
3 import javax.swing.event.*;
4
5 public class ListDemo extends JFrame {
6   final int NUMBER_OF_FLAGS = 9;
7
8   // Declare an array of Strings for flag titles
9   private String[] flagTitles = {"Canada", "China", "Denmark",
10    "France", "Germany", "India", "Norway", "United Kingdom",
11    "United States of America"};
12
13  // The list for selecting countries
14  private JList jlst = new JList(flagTitles);
15
16  // Declare an ImageIcon array for the national flags of 9 countries
17  private ImageIcon[] imageIcons = {
18    new ImageIcon("image/ca.gif"),
19    new ImageIcon("image/china.gif"),
20    new ImageIcon("image/denmark.gif"),
21    new ImageIcon("image/fr.gif"),
22    new ImageIcon("image/germany.gif"),
23    new ImageIcon("image/india.gif"),
24    new ImageIcon("image/norway.gif"),
25    new ImageIcon("image/uk.gif"),
26    new ImageIcon("image/us.gif")
27  };
28
```

```
29  // Arrays of labels for displaying images
30  private JLabel[] jlblImageViewer = new JLabel[NUMBER_OF_FLAGS];
31
32  public static void main(String[] args) {
33    ListDemo frame = new ListDemo();
34    frame.setSize(650, 500);
35    frame.setTitle("ListDemo");
36    frame.setLocationRelativeTo(null); // Center the frame
37    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
38    frame.setVisible(true);
39  }
40
41  public ListDemo() {
42    // Create a panel to hold nine labels
43    JPanel p = new JPanel(new GridLayout(3, 3, 5, 5));
44
45    for (int i = 0; i < NUMBER_OF_FLAGS; i++) {
46      p.add(jlblImageViewer[i] = new JLabel());
47      jlblImageViewer[i].setHorizontalAlignment
48        (SwingConstants.CENTER);
49    }
50
51    // Add p and the list to the frame
52    add(p, BorderLayout.CENTER);
53    add(new JScrollPane(jlst), BorderLayout.WEST);
54
55    // Register listeners
56    jlst.addListSelectionListener(new ListSelectionListener() {
57      /** Handle list selection */
58      public void valueChanged(ListSelectionEvent e) {
59        // Get selected indices
60        int[] indices = jlst.getSelectedIndices();
61
62        int i;
63        // Set icons in the labels
64        for (i = 0; i < indices.length; i++) {
65          jlblImageViewer[i].setIcon(imageIcons[indices[i]]);
66        }
67
68        // Remove icons from the rest of the labels
69        for (; i < NUMBER_OF_FLAGS; i++) {
70          jlblImageViewer[i].setIcon(null);
71        }
72      }
73    });
74  }
75 }
```

为处理列表框中选定的国家名，匿名内部类监听器监听ListSelectionEvent（第56～73行）。ListSelectionEvent和ListSelectionListener都定义在javax.swing.event包中，因此，程序需要导入这个包（第3行）。

这个程序创建包含九个标签的数组，用来显示九个国家的国旗图像。程序将九个国家国旗的图像装入一个图像数组中（第17～27行），并创建一个与图像数组顺序相同的九国国名列表框（第9～11行）。这样，图像数组的下标0就对应于列表框的第一个国家。

列表框放置在一个滚动窗格中（第53行），这样，当列表框中的项数超出视图域时，列表框是可以滚动的。

默认情况下，列表框的选择模式是多区间选择，它允许用户在列表框中从不同块中选择多个选项。当用户在列表框中选择国家时，执行valueChanged处理器（第58～73行），它获取选定项的下标并给标签设置相应的图像图标以显示国旗。

17.10 滚动条

滚动条（JScrollBar）是一个允许用户从一个值的范围中进行选择的组件，如图17-26所示。

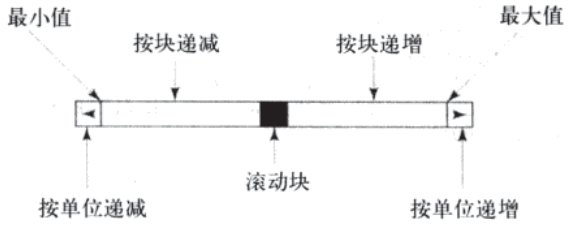


图17-26 滚动条可以用图形表示一个值的范围

通常，用户通过鼠标操作改变滚动条的值。例如，用户可以上下拖动滚动块，或者点击滚动条的单位递增区或块递增区。键盘操作也可改变滚动条的值。习惯上，Page Up键和Page Down键相当于点击块增加区和块减少区。

注意 滚动条轨迹的宽度为maximum+visibleAmount。当滚动条设置为它的最大值时，滚动块的左侧就在maximum处，而右侧在maximum+visibleAmount处。

JScrollBar有以下属性，如图17-27所示。

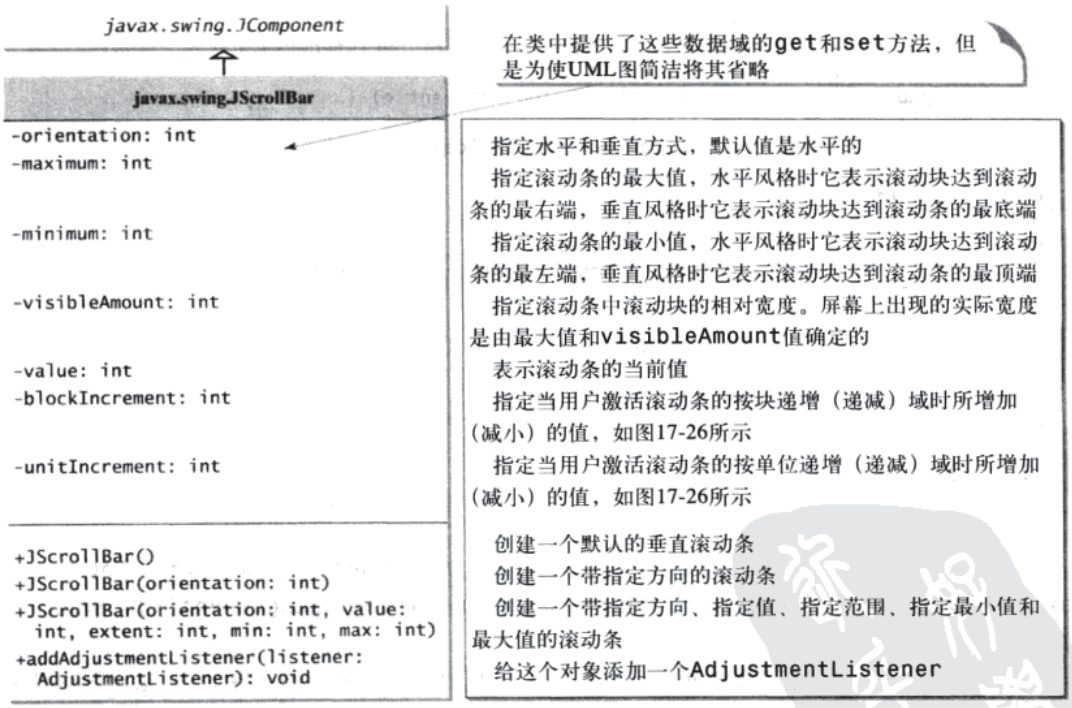


图17-27 JScrollBar允许在一个值的范围中进行选择

当用户改变滚动条的值时，滚动条会触发一个AdjustmentEvent的实例，传递给每一个已经注册的监听器。如果一个对象希望滚动条的值发生变化时能够通知它，就必须实现java.awt.event.AdjustmentListener监听器接口中的adjustmentValueChanged方法。

程序清单17-10给出使用水平滚动条和垂直滚动条来控制面板上显示一条消息的程序。水平滚动条用以左右移动消息，而垂直滚动条用以上下移动消息。程序的运行示例如图17-28所示。

图17-28 滚动条在面板上水平和垂直移动消息

下面是程序的主要步骤：

1）创建用户界面。

创建一个MessagePanel对象，将它放置于框架的中央。创建一个垂直滚动条，将它放到框架的东边。创建一个水平滚动条，将它放到框架的南边。

2）处理事件。

创建一个监听器实现adjustmentValueChanged处理器，根据滚动块在滚动条中的移动来移动消息。

程序清单17-10 ScrollBarDemo.java

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

public class ScrollBarDemo extends JFrame {
  // Create horizontal and vertical scroll bars
  private JScrollBar jscbHort =
    new JScrollBar(JScrollBar.HORIZONTAL);
  private JScrollBar jscbVert =
    new JScrollBar(JScrollBar.VERTICAL);

  // Create a MessagePanel
  private MessagePanel messagePanel =
    new MessagePanel("Welcome to Java");

  public static void main(String[] args) {
    ScrollBarDemo frame = new ScrollBarDemo();
    frame.setTitle("ScrollBarDemo");
    frame.setLocationRelativeTo(null); // Center the frame
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.pack();
    frame.setVisible(true);
  }

  public ScrollBarDemo() {
    // Add scroll bars and message panel to the frame
    setLayout(new BorderLayout());
    add(messagePanel, BorderLayout.CENTER);
    add(jscbVert, BorderLayout.EAST);
    add(jscbHort, BorderLayout.SOUTH);

    // Register listener for the scroll bars
    jscbHort.addAdjustmentListener(new AdjustmentListener() {
      public void adjustmentValueChanged(AdjustmentEvent e) {
        // getValue() and getMaximumValue() return int, but for better
        // precision, use double
        double value = jscbHort.getValue();
        double maximumValue = jscbHort.getMaximum();
        double newX = (value * messagePanel.getWidth() /
          maximumValue);
        messagePanel.setXCoordinate((int)newX);
      }
    });
    jscbVert.addAdjustmentListener(new AdjustmentListener() {
```

```
45        public void adjustmentValueChanged(AdjustmentEvent e) {
46          // getValue() and getMaximumValue() return int, but for better
47          // precision, use double
48          double value = jscbVert.getValue();
49          double maximumValue = jscbVert.getMaximum();
50          double newY = (value * messagePanel.getHeight() /
51            maximumValue);
52          messagePanel.setYCoordinate((int)newY);
53        }
54      });
55    }
56 }
```

程序创建两个滚动条（jscbVert和jscbHort）（第7～10行）和一个MessagePanel实例（messagePanel）（第13～14行）。将messagePanel放置在框架的中央，将jscbVert和jscbHort分别放置在框架的东区和南区（第29～30行）。

可以在构造方法中指定滚动条的方向，也可以使用setOrientation方法来指定。默认情况下，属性maximum的值为100，minimum的值为0，blockIncrement的值为10，而visibleAmount的值为10。

当用户拖动滚动块或点击单位增加区、单位减少区时，滚动条的值将改变。AdjustmentEvent事件的一个实例被触发，并通过调用adjustmentValueChanged处理器传递给监听器。垂直滚动条的监听器向上、向下移动消息（第33～43行），水平滚动条的监听器向左、向右移动消息（第44～54行）。

垂直滚动条的最大值对应为面板的高度，而水平滚动条的最大值对应为面板的宽度。水平滚动条的当前值与最大值之间的比率等于x的值与消息面板的宽度的比值。类似地，垂直滚动条的当前值与最大值之间的比率等于y的值与消息面板的高度的比值。随着滚动条的调整，相应地设置消息面板的x坐标和y坐标（第39和第50行）。

17.11 滑块

JSlider与JScrollBar类似，但是JSlider具有更多的属性，并且可以以多种形式显示。图17-29显示两个滑块。

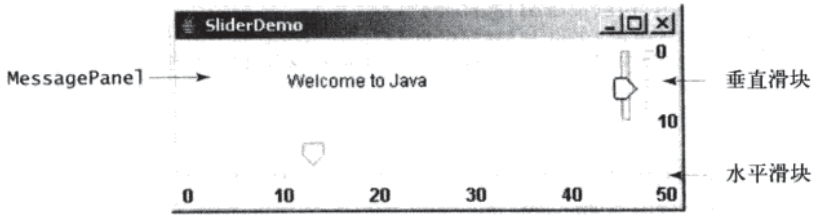


图17-29 滑块在面板上沿水平方向和垂直方向移动消息

JSlider允许用户以图形方式通过滑动按钮，在一个有界的区间中选择一个数值。滑块可以在主要刻度以及次要刻度之间滑动。刻度之间的像素值是由majorTickSpacing和minorTickSpacing属性控制的。滑块可以水平显示也可以垂直显示，可以带也可以不带刻度，可以有标签也可以没有。JSlider中经常使用的构造方法和属性如图17-30所示。

注意 垂直滚动条的值从上向下是增加的，但是默认情况下，垂直滚动条的值从上向下是减少的。

注意 图17-30中列出的所有属性都有相关的get和set方法，为了简洁可以忽略它们。习惯上，布尔属性的get方法命名为is<PropertyName>()。但是在JSlider类中，paintLabels、paintTicks、paintTrack和inverted的get方法分别命名为getPaintLabels()、getPaintTicks()、getPaintTrack()和getInverted()，这违反了Java的命名习惯。

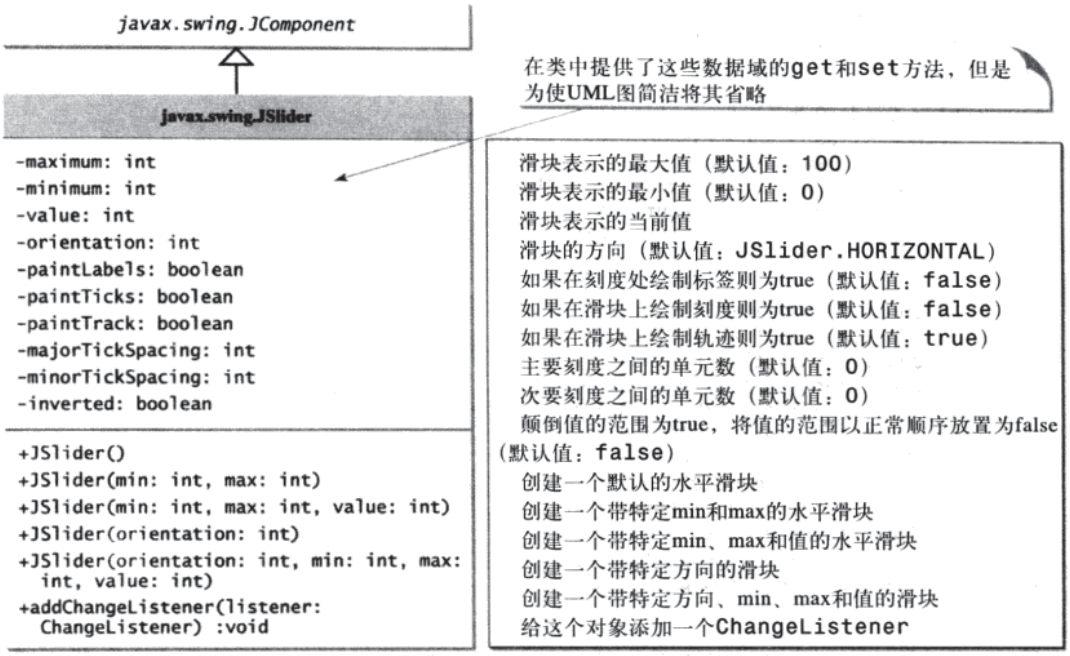


图17-30 JSlider允许从某个范围的值中进行选择

当用户改变滑块的值时，滑块就会触发javax.swing.event.ChangeEvent的一个实例，可以把它传递给任何已经注册的监听器。如果一个对象希望滑块的值发生变化时能够通知它，那就必须实现定义在包java.swing.event中的ChangeListener接口中的stateChanged方法。

程序清单17-11编写程序，使用滑块来控制在面板上显示一条消息，如图17-29所示。下面是程序的主要步骤：

1）创建用户界面。

创建一个MessagePanel对象，然后将它放置在框架的中央。创建一个垂直滑块，将它放置在框架的东边。创建一个水平滑块，将它放置在框架的南边。

2）处理事件。

创建一个监听器来实现ChangeListener接口中的stateChanged处理器，根据滑块在滑动区中的移动来移动这条消息。

程序清单17-11 SliderDemo.java

```
1 import java.awt.*;
2 import javax.swing.*;
3 import javax.swing.event.*;
4
5 public class SliderDemo extends JFrame {
6   // Create horizontal and vertical sliders
7   private JSlider jsldHort = new JSlider(JSlider.HORIZONTAL);
8   private JSlider jsldVert = new JSlider(JSlider.VERTICAL);
9
10  // Create a MessagePanel
11  private MessagePanel messagePanel =
12    new MessagePanel("Welcome to Java");
13
14  public static void main(String[] args) {
15    SliderDemo frame = new SliderDemo();
16    frame.setTitle("SliderDemo");
17    frame.setLocationRelativeTo(null); // Center the frame
18    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
19      frame.pack();
20      frame.setVisible(true);
21    }
22
23    public SliderDemo() {
24      // Add sliders and message panel to the frame
25      setLayout(new BorderLayout(5, 5));
26      add(messagePanel, BorderLayout.CENTER);
27      add(jsldVert, BorderLayout.EAST);
28      add(jsldHort, BorderLayout.SOUTH);
29
30      // Set properties for sliders
31      jsldHort.setMaximum(50);
32      jsldHort.setPaintLabels(true);
33      jsldHort.setPaintTicks(true);
34      jsldHort.setMajorTickSpacing(10);
35      jsldHort.setMinorTickSpacing(1);
36      jsldHort.setPaintTrack(false);
37      jsldVert.setInverted(true);
38      jsldVert.setMaximum(10);
39      jsldVert.setPaintLabels(true);
40      jsldVert.setPaintTicks(true);
41      jsldVert.setMajorTickSpacing(10);
42      jsldVert.setMinorTickSpacing(1);
43
44      // Register listener for the sliders
45      jsldHort.addChangeListener(new ChangeListener() {
46        /** Handle scroll-bar adjustment actions */
47        public void stateChanged(ChangeEvent e) {
48          // getValue() and getMaximumValue() return int, but for better
49          // precision, use double
50          double value = jsldHort.getValue();
51          double maximumValue = jsldHort.getMaximum();
52          double newX = (value * messagePanel.getWidth() /
53            maximumValue);
54          messagePanel.setXCoordinate((int)newX);
55        }
56      });
57      jsldVert.addChangeListener(new ChangeListener() {
58        /** Handle scroll-bar adjustment actions */
59        public void stateChanged(ChangeEvent e) {
60          // getValue() and getMaximumValue() return int, but for better
61          // precision, use double
62          double value = jsldVert.getValue();
63          double maximumValue = jsldVert.getMaximum();
64          double newY = (value * messagePanel.getHeight() /
65            maximumValue);
66          messagePanel.setYCoordinate((int) newY);
67        }
68      });
69    }
70 }
```

JSlider与JScrollBar类似，但是JSlider具有更多的特性。如本例所示，可以在JSlider上指定最大值、标签、主要标记和次要标记（第31～35行）。也可以选择隐藏轨迹（第36行）。由于垂直滑块的值从上向下是递减的，所以，可以使用setInverted将其顺序颠倒过来（第37行）。

改变滑块的值时，JSlider会触发ChangeEvent事件。监听器需要实现ChangeListener接口中的stateChanged处理器（第45～68行）。注意，当调整滚动条时，JScrollBar会触发AdjustmentEvent事件。

17.12 创建多个窗口

有时侯，可能希望在一个应用程序中创建多个窗口。这个程序会新开一个窗口执行某个指定的任务。新开的窗口称为子窗口（subwindow），而主框架称作主窗口（main window）。

为了从应用程序中创建一个子窗口，需要定义一个指明任务的JFrame的子类，并且告诉新窗口做什么。然后，可以在应用程序中创建该子类的一个实例，通过把它设为可见的框架实例即可出现一个新窗口。

程序清单17-12给出的程序创建一个主窗口，它有一个在滚动窗格中的文本域和一个名为Show Histogram（显示直方图）的按钮。当用户点击这个按钮时，就会出现一个显示一个直方图的新窗口，这个直方图显示文本域中字母出现的次数。图17-31中包含这个程序的一个运行示例。

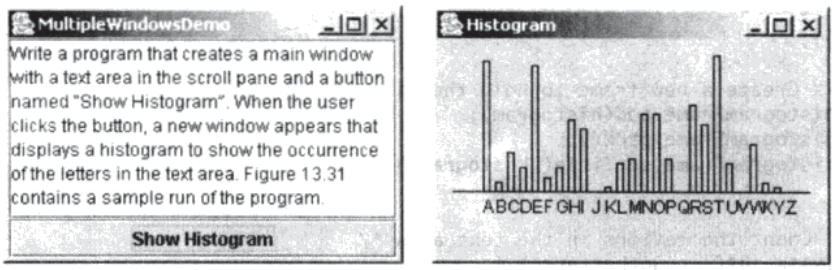


图17-31 在一个独立的框架中显示直方图

下面是这个程序的主要步骤：

1）在程序清单17-12中，为一个名为MultipleWindowsDemo的框架创建一个主类。在滚动窗格中添加一个文本域，然后将这个滚动窗格放置在框架的中央。创建一个按钮Show Histogram，然后将它放到框架的南边。

2）在程序清单17-13中创建一个名为Histogram的JPanel类的子类。这个类包含一个名为count的int[]类型的数据域，它是用来统计26个字母的出现次数的。在直方图中显示count的值。

3）实现MultipleWindowsDemo中的actionPerformed处理器，如下所示：

① 创建一个Histogram的实例，统计文本域中字母的出现次数，然后将统计结果传递给Histogram对象。

② 创建一个新框架，然后将Histogram对象放到框架的中央。显示这个框架。

程序清单17-12 MultipleWindowsDemo.java

```
1  import java.awt.*;
2  import java.awt.event.*;
3  import javax.swing.*;
4
5  public class MultipleWindowsDemo extends JFrame {
6    private JTextArea jta;
7    private JButton jbtShowHistogram = new JButton("Show Histogram");
8    private Histogram histogram = new Histogram();
9
10   // Create a new frame to hold the histogram panel
11   private JFrame histogramFrame = new JFrame();
12
13   public MultipleWindowsDemo() {
14     // Store text area in a scroll pane
15     JScrollPane scrollPane = new JScrollPane(jta = new JTextArea());
16     scrollPane.setPreferredSize(new Dimension(300, 200));
17     jta.setWrapStyleWord(true);
18     jta.setLineWrap(true);
19
20     // Place scroll pane and button in the frame
```

```
21      add(scrollPane, BorderLayout.CENTER);
22      add(jbtShowHistogram, BorderLayout.SOUTH);
23
24      // Register listener
25      jbtShowHistogram.addActionListener(new ActionListener() {
26        /** Handle the button action */
27        public void actionPerformed(ActionEvent e) {
28          // Count the letters in the text area
29          int[] count = countLetters();
30
31          // Set the letter count to histogram for display
32          histogram.showHistogram(count);
33
34          // Show the frame
35          histogramFrame.setVisible(true);
36        }
37      });
38
39      // Create a new frame to hold the histogram panel
40      histogramFrame.add(histogram);
41      histogramFrame.pack();
42      histogramFrame.setTitle("Histogram");
43    }
44
45    /** Count the letters in the text area */
46    private int[] countLetters() {
47      // Count for 26 letters
48      int[] count = new int[26];
49
50      // Get contents from the text area
51      String text = jta.getText();
52
53      // Count occurrences of each letter (case insensitive)
54      for (int i = 0; i < text.length(); i++) {
55        char character = text.charAt(i);
56
57        if ((character >= 'A') && (character <= 'Z')) {
58          count[character - 'A']++;
59        }
60        else if ((character >= 'a') && (character <= 'z')) {
61          count[character - 'a']++;
62        }
63      }
64
65      return count; // Return the count array
66    }
67
68    public static void main(String[] args) {
69      MultipleWindowsDemo frame = new MultipleWindowsDemo();
70      frame.setLocationRelativeTo(null); // Center the frame
71      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
72      frame.setTitle("MultipleWindowsDemo");
73      frame.pack();
74      frame.setVisible(true);
75    }
76  }
```

程序清单17-13 **Histogram.java**

```
1  import javax.swing.*;
2  import java.awt.*;
3
4  public class Histogram extends JPanel {
5    // Count the occurrences of 26 letters
6    private int[] count;
7
```

```
 8   /** Set the count and display histogram */
 9   public void showHistogram(int[] count) {
10     this.count = count;
11     repaint();
12   }
13
14   /** Paint the histogram */
15   protected void paintComponent(Graphics g) {
16     if (count == null) return; // No display if count is null
17
18     super.paintComponent(g);
19
20     // Find the panel size and bar width and interval dynamically
21     int width = getWidth();
22     int height = getHeight();
23     int interval = (width - 40) / count.length;
24     int individualWidth = (int)(((width - 40) / 24) * 0.60);
25
26     // Find the maximum count. The maximum count has the highest bar
27     int maxCount = 0;
28     for (int i = 0; i < count.length; i++) {
29       if (maxCount < count[i])
30         maxCount = count[i];
31     }
32
33     // x is the start position for the first bar in the histogram
34     int x = 30;
35
36     // Draw a horizontal base line
37     g.drawLine(10, height - 45, width - 10, height - 45);
38     for (int i = 0; i < count.length; i++) {
39       // Find the bar height
40       int barHeight =
41         (int)(((double)count[i] / (double)maxCount) * (height - 55));
42
43       // Display a bar (i.e. rectangle)
44       g.drawRect(x, height - 45 - barHeight, individualWidth,
45         barHeight);
46
47       // Display a letter under the base line
48       g.drawString((char)(65 + i) + "", x, height - 30);
49
50       // Move x for displaying the next character
51       x += interval;
52     }
53   }
54
55   /** Override getPreferredSize */
56   public Dimension getPreferredSize() {
57     return new Dimension(300, 300);
58   }
59 }
```

程序包含两个类MultipleWindowsDemo和Histogram，它们的关系如图17-32所示。

MultipleWindowsDemo是一个框架，它包括一个在滚动窗格中的文本域以及一个按钮。Histogram是JPanel的子类，用于显示文本域中字母出现次数的直方图。

当用户点击按钮Show Histogram后，处理器会统计文本域中每个字母出现的次数。统计字母时不区分大小写。不统计非字母的字符。统计结果存储在一个26个元素组成的int型数组中。数组中的第一个元素存放的是字母'a'或'A'的个数，最后一个元素存放的是'z'或'Z'的个数。数组count的值传递给显示字母统计情况的直方图。

MultipleWindowsDemo类包含一个main方法。这个main方法创建MultipleWindowsDemo的一个

实例并显示框架。MultipleWindowsDemo类还包含一个名为histogramFrame的JFrame实例，而histogramFrame又包含了一个Histogram的实例。当用户点击按钮Show Histogram之后，histogramFrame被设置为可见的，用以显示直方图。

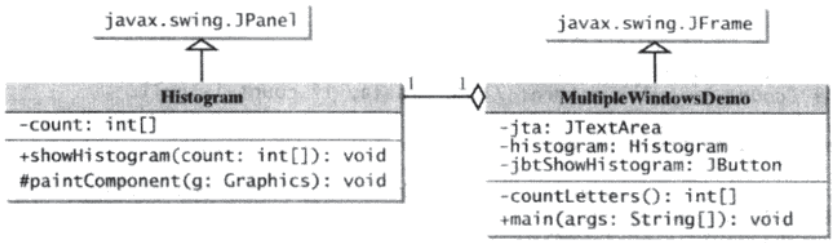


图17-32 MultipleWindowsDemo使用Histogram在框架中显示一个直方图，该图表示文本域中字母出现的次数

直方图中块的高度和宽度是根据直方图窗口的大小动态决定的。

不能把JFrame的实例添加到容器中。例如，将histogramFrame添加到主框架会导致一个运行时异常。不过，创建一个框架实例并将它设置为可见，就可以出现一个新窗口。

本章小结

- 学习如何使用像JButton、JCheckBox、JRadioButton、JLabel、JTextField、JTextArea、JComboBox、JList、JScrollBar和JSlider这样的Swing GUI组件来创建图形用户界面。学习如何处理这些组件上的事件。
- 可以在按钮（JButton、JCheckBox和JRadioButton）和标签（JLabel）上显示文本和图标。

复习题

17.2～17.4节

17.1 如何创建一个标有OK的按钮？如何改变按钮上的文本？如何在按钮上设置一个图标、按下图标和翻转图标？

17.2 假定一个JButton对象jbtOK，编写语句设置按钮的前景色为红色、背景色为黄色、热键为'K'、工具提示文本为"Click OK to proceed"、水平对齐方式为RIGHT、垂直对齐方式为BOTTOM、水平文本位置为LEFT、垂直文本位置为TOP，而图标文本间隔为5。

17.3 如何创建一个复选框？如何创建一个初始状态是选中状态的复选框？如何确定复选框是否选中？

17.4 如何创建一个单选按钮？如何创建一个初始状态是选中状态的单选按钮？如何把单选按钮组织在一起？如何确定单选按钮是否选中？

17.5 列出定义在AbstractButton类中定义的至少五个属性。

17.5～17.9节

17.6 如何创建一个命名为"Address"的标签？如何改变标签上的名字？如何在标签上设置一个图标？

17.7 假设有一个JLabel对象jlblMap，编写语句设置标签的前景色为红色、背景色为黄色、热键为'K'、工具提示文本为"Map image"、水平对齐方式为RIGHT、垂直对齐方式为BOTTOM、水平文本位置为LEFT、垂直文本位置为TOP，而图标文本间隔为5。

17.8 如何创建一个10列的文本域，并且将默认文本设置为"Welcome to Java"？如何在文本域中添加代码以检验文本域是否为空？

17.9 如何创建一个10行20列的文本域？如何在文本域中插入三行代码？如何创建一个可滚动的文本域？

17.10 如何创建一个组合框，向它添加三个条目，并再取一个选定项？

17.11 如何用一个字符串数组创建列表框？

17.10～17.12节

17.12 如何创建一个水平的滚动条？滚动条可以触发什么事件？

17.13 如何创建一个垂直的滑块？滑块可以触发什么事件？

17.14 解释如何在一个应用程序中创建和显示多个框架。

编程练习题

教学注意 教师可以将第18章的练习题作为本章的练习题布置给学生，要求学生编写Java应用程序，而不是编写Java applet。

17.2～17.5节

*17.1 （修改程序清单17-2）改写程序清单17-2，添加一组选择背景颜色的单选按钮。供选择的颜色有红色、黄色、白色、灰色和绿色（参见图17-33）。

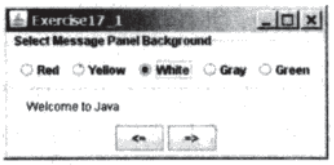


图17-33 <=和>=按钮在面板上移动消息，也可以为消息设置背景色

*17.2 （选择几何图形）编写一个绘制各种几何图形的程序，如图17-34所示。用户从单选按钮中选择一个几何图形，并且在复选框中确认是否被填充（提示，使用程序清单15-3中介绍的FigurePanel类来显示一个图形）。

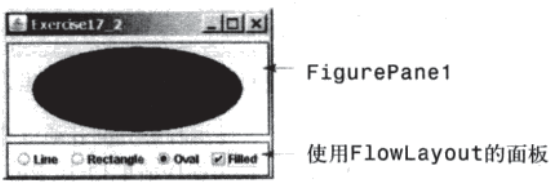


图17-34 选择一个图形时，程序就会显示直线、矩形和椭圆

**17.3 （交通信号灯）编写一个程序，模拟交通信号灯。程序让用户从红、黄、绿三色灯中选择一种。当选择一个单选按钮后，相应的灯被打开，并且一次只能亮一种灯（如图17-35所示）。程序开始时所有的灯都不亮。

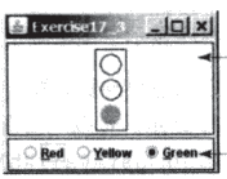
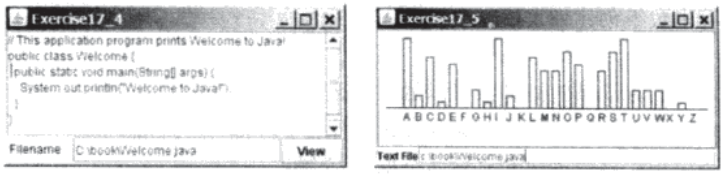


图17-35 单选框放在一组中，在组中只能选择一种颜色来控制交通信号灯

17.6～17.10节

**17.4 （文本浏览器）编写一个程序在文本域中显示一个文本文件，如图17-36所示。用户在文本域中输入一个文件名，然后点击View按钮；随后在文本域中会显示这个文件。



a) b)

图17-36 a）程序将文件中的文本显示到文本域；
b）程序显示一个直方图来表示文件中每个字母出现的次数

**17.5 （创建表示字母出现次数的直方图）在程序清单17-12中，开发的程序显示一个直方图，它显示的是文本区域中每个字母出现的次数。复用程序清单17-12中的`Histogram`类来编写一个在面板上显示直方图的程序。直方图应该显示文本文件中每个字母出现的次数，如图17-36b所示。假设字母是不区分大小写的。

（1）将显示直方图的面板放置在框架的中央位置。

（2）在面板中放置一个标签和文本域，并且将面板放在框架的南边。文本文件将从这个文本域输入。

（3）在文本域中点击回车键后，程序就会计算每个字母出现的次数，并且用直方图显示。

*17.6 （创建一个英里/公里的转换器）编写一个程序来转换英里和公里，如图17-37所示。如果在英里文本域Mile中输入一个值之后按下回车键，就会在公里文本域Kilometer中显示对应的公里值。同样的，在公里文本域Kilometer中输入一个值之后按下回车键，就会在英里文本域Mile中显示对应的英里值。

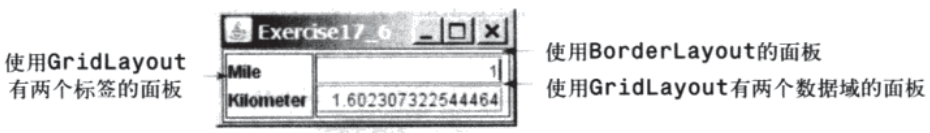


图17-37 程序可以将英里转换成公里，反之亦然

*17.7 （设置时钟的时间）编写一个程序，显示时钟时间并通过在三个文本域中输入小时、分钟和秒来设置时钟的时间，如图17-38所示。使用程序清单15-10中的`StillClock`类。

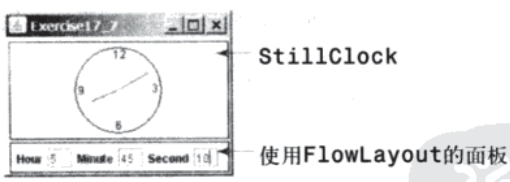


图17-38 程序显示字体域确定的时间

**17.8 （选择一种字体）编写一个程序，可以动态地改变面板上显示的消息的字体。这个消息可以同时以粗体和斜体显示，也可以在面板上居中显示。可以从组合框中选择字体名和字体大小，如图17-39所示。使用`GraphicsEnvironment`类中的`getAvailableFontFamilyNames()`方法可以得到可用的字体名（12.8节）。字体大小的组合框初始化为从1到100之间的数字。

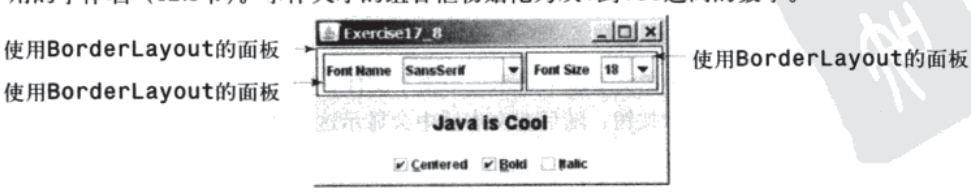


图17-39 可以动态设置消息的字体

**17.9 （演示JLabel的属性）编写一个程序，允许用户动态地设置属性：水平对齐方式horizontalAlignment、垂直对齐方式verticalAlignment、水平文本对齐方式horizontalTextAlignment和垂直文本对齐方式verticalTextAlignment，如图17-40所示。

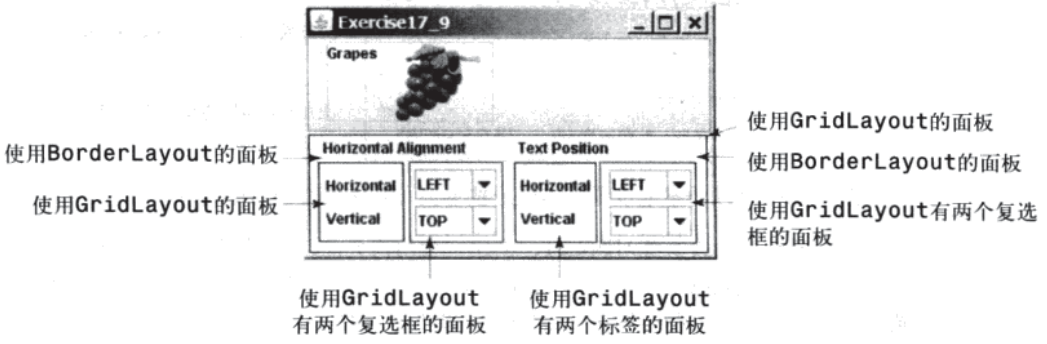


图17-40 可以动态地设置按钮的对齐方式以及按钮文本的位置属性

*17.10 （逐步给程序清单17-2增加新特性）如下所示，逐步改进程序清单17-2（如图17-41）：

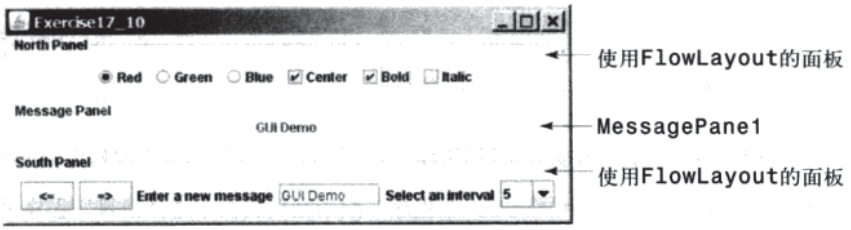


图17-41 程序使用按钮、标签、文本域、组合框、单选按钮、复选框和边框

（1）在放置按钮的同一个面板上添加标签为"Enter a new message"的文本域。在该文本域中输入新消息之后按下回车键，新消息就会显示在消息面板中。

（2）在放置按钮的同一个面板上添加标签为"Select an interval"的组合框。这个组合框允许用户选择移动消息的新间隔。该值的选择范围从5到100，间隔为5。用户也可以在组合框中输入新的间隔值。

（3）添加三个单选按钮，用户可以使用它们选择消息的前景色为红色、绿色和蓝色。这些单选按钮组成一组放在一个面板中，这个面板放置在框架的内容窗格的北边。

（4）添加三个复选框，用户可以使用它们来设置消息居中，并且用粗体或斜体显示消息。将这些复选框和单选按钮放在同一个面板中。

（5）给消息面板添加标题为"Message Panel"的边框，给按钮的面板添加标题为"South Panel"的边框，给单选按钮和复选框所在的面板添加标题为"North Panel"的边框。

*17.11 （演示JTextField的属性）编写一个程序，动态地设置文本域的水平对齐属性和列宽属性，如图17-42所示。

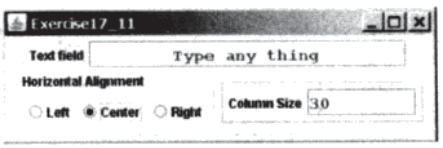


图17-42 可以动态地设置文本域的水平对齐属性和列宽属性

*17.12 （演示JTextArea的属性）编写一个程序，演示文本域的换行方式。程序用复选框选择文本域是否自动换行。在文本区域可以换行的情况下，需要指定按单词还是按字符换行，如图17-43所示。

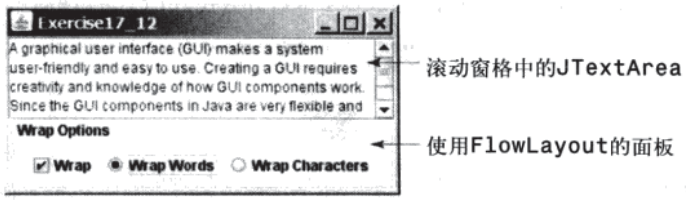


图17-43 可以动态地设置文本域是按单词还是按字符换行的选项

*17.13 （比较不同利率的贷款）改写练习题4.21，创建一个用户界面，如图17-44所示。程序应该允许用户从文本域输入贷款额以及以年为单位的贷款年限，然后，在文本域中显示针对每种利率的月偿还额和总偿还额，利率从5%到8%，按1/8（0.125%）递增。

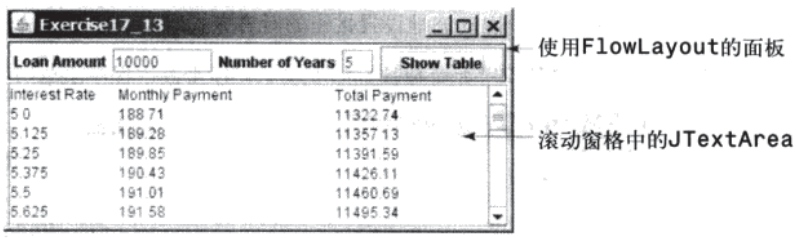


图17-44 程序显示按不同利率给定贷款的月偿还额和总偿还额构成的表格

*17.14 （使用JComboBox和JList）编写一个程序，演示在列表框中选择条目。程序用组合框指定选择方式，如图17-45所示。当选择条目后，列表框下方的标签中就会显示选定项。

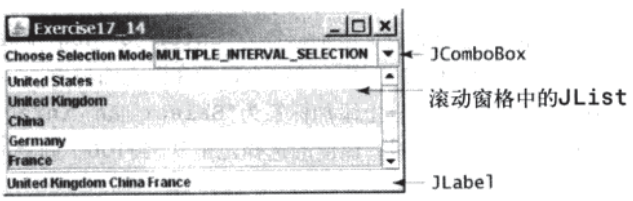


图17-45 可以在列表中选择单项选择、单区间选择或者多区间选择

17.11～17.13节

**17.15 （使用JScrollBar）编写一个程序，使用滚动条选择标签的前景色，如图17-46所示。使用三个水平滚动条选择一种颜色的红色、绿色和蓝色的成分。给放置滚动条的面板上加一个带标题的边框。

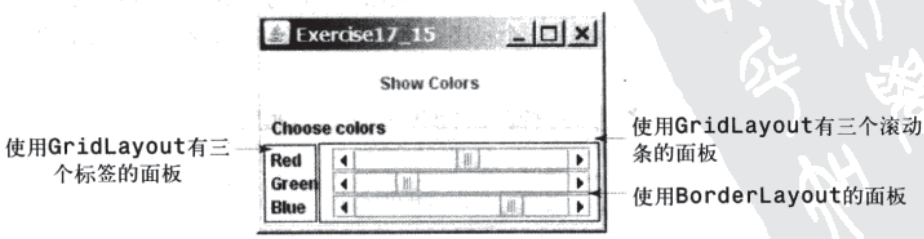


图17-46 标签中的前景色随着滚动条的调整而改变

**17.16 （使用JSlider）使用滑块改写前一个练习题。

***17.17 （显示一个日历）编写一个程序，显示当前月的日历，如图17-47所示。使用标签并且在标签上设置文本以显示日历。使用14.3节中的GregorianCalendar类获取月份、年份、该月第一天和该月的天数等信息。

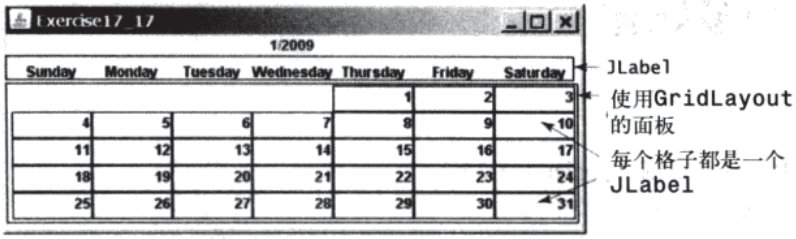


图17-47 程序显示当月的日历

*17.18 （修改程序清单17-12）不使用程序清单17-12中的Histogram组件显示字母出现的次数，而是采用条形图显示，这样，显示结果如图17-48所示。

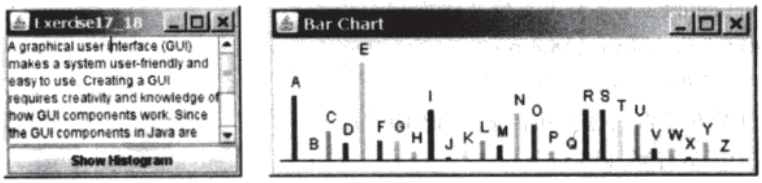


图17-48 使用条形图显示每个字母出现的次数

**17.19 （显示国旗和国旗的描述）程序清单17-8给出的程序允许用户通过在组合框中选择国家来查看该国的国旗图像及对它的描述。对它的描述在程序中是用字符串表示的。改写这个程序从一个文件中读取它的文本描述。假设相关的描述存储在text目录下的description0.txt, ..., description8.txt文件中，这九个文件依次对应于九个国家：加拿大、中国、丹麦、法国、德国、印度、挪威、英国和美国。

17.20 （放幻灯片）练习题16.13使用图像开发了一个幻灯片展示。使用文本文件改写练习题16.13来开发一个幻灯片展示。假设十个名为slide0.txt, slide1.txt, ..., slide9.txt的文本文件都存储在text目录下。每张幻灯片显示一个文件的文本，每张幻灯片持续显示一秒。幻灯片依次显示。当显示完最后一张幻灯片，重新显示第一张，依此类推。使用一个文本域显示幻灯片。