第12讲 常用组件GUI设计

10.1 Swing 概 述

图10.1 图形用户界面中常用的组件

10.1.1 Swing中常用的包

由于Swing功能相当强大且复杂，考虑到功能分解的原则，Java系统将Swing按不同的功能分为表10.1所示的若干个包，它们分别针对于不同的组件及事件。

表10.1 Swing中常用的包

包　　名	说　　明
javax.swing	最常用的包，包含了各种 Swing 组件的类
javax.swing.border	包含与 Swing 组件外框有关的类
javax.swing.colorchooser	针对 Swing 调色板组件(JColorChooser)设计的类
javax.swing.event	处理由 Swing 组件产生的事件
javax.swing.filechooser	包含针对 Swing 文件选择对话框(JFileChooser)所设计的类
javax.swing.plaf	处理 Swing 组件外观的相关类
javax.swing.plaf.basic	
javax.swing.plaf.metal	
javax.swing.plaf.multi	
javax.swing.table	针对 Swing 组件表格(JTable)所设计的类
javax.swing.text	包含与 Swing 文字组件相关的类
javax.swing.text.html	
javax.swing.text.html.parser	
javax.swing.text.rtf	
javax.swing.tree	针对 Swing 树状元素(JTree)所设计的类
javax.swing.undo	提供 Swing 文字组件的 Redo 与 Undo 功能

10.1.2 常用组件的继承关系

本章所述组件的继承关系如下：

java.lang.Object

java.awt.Component

java.awt.Container

javax.swing.JComponent

javax.swing.JLabel

javax.swing.JTextField

javax.swing.JTextArea

javax.swing.JList

javax.swing.JComboBox

javax.swing.AbstractButton

javax.swing.JButton

javax.swing.JToggleButton

javax.swing.JCheckBox

javax.swing. JRadioButton

☞ 10.2 事件响应原理

10.2.1 委托事件模型

Java采用委托事件模型来处理事件。委托事件模型的特点是将事件的处理委托给独立的对象，而不是组件本身，从而将使用者界面与程序逻辑分开。整个“委托事件模型”由产生事件的对象(事件源)、事件对象及监听者对象之间的关系所组成。

在事件产生时，产生事件的对象将与该事件相关的信息封装在一个称为“事件对象”的对象中，并将该对象传递给监听者对象，监听者对象根据该事件对象内的信息决定适当的处理方式。监听者对象要收到事件发生的通知，必须在程序代码中向产生事件的对象注册，当事件产生时，产生事件的对象就会主动通知监听者对象，监听者对象就可以根据产生该事件的对象来决定处理事件的方法。

监听者对象(listener)就是用来处理事件的对象。监听者对象等候事件的发生，并在事件发生时收到通知。

10.2.2 Swing组件的事件及监听者

不同事件源上发生的事件种类不同，不同的事件由不同的监听者处理。表10.2列出了Swing中各种组件可激发的事件及事件监听者之间的对应关系。表10.3列出了Swing提供的各事件监听者与各事件类成员方法之间的关系。请读者务必注意：对应不同事件需要不同的事件监听者，而每个事件监听者都有相应的成员方法，我们处理事件的程序代码要写在对应的成员方法体中。

表10.2 Swing中组件、事件及事件监听者之间的对应关系

组　　件	可激发的事件 (Event)	事件监听者 (EventListener)
AbstractButton (JButton,JToggleButton, JCheckBox,JRadioButton)	ActionEvent ChangeEvent ItemEvent	ActionListener ChangeListener ItemListener
JFileChooser	ActionEvent	ActionListener
JTextField JPasswordField	ActionEvent CaretEvent DocumentEvent UndoableEvent	ActionListener CaretListener DocumentListener UndoableListener
JTextArea	CaretEvent DocumentEvent UndoableEvent	CaretListener DocumentListener UndoableListener
JTextPane JEditorPane	CaretEvent DocumentEvent UndoableEvent HyperlinkEvent	CaretListener DocumentListener UndoableListener HyperlinkListener
JComboBox	ActionEvent ItemEvent	ActionListener ItemListener

表10.2 Swing中组件、事件及事件监听者之间的对应关系

JList	ListSelectionEvent ListDataEvent	ListSelectionListener ListDataListener
JMenuItem	ActionEvent ChangeEvent ItemEvent MenuKeyEvent MenuDragMouseEvent	ActionListener ChangeListener ItemListener MenuKeyListener MenuDragMouseListener
JMenu	MenuEvent	MenuListener
JPopupMenu	PopupMenuEvent	PopupMenuListener
JProgressBar	ChangeEvent	ChangeListener
JSlider	ChangeEvent	ChangeListener
JScrollBar	AdjustmentEvent	AdjustmentListener
JTable	ListSelectionEvent TableModeEvent TableColumnModelEvent CellEditorEvent	ListSelectionListener TableModeListener TableColumnModelListener CellEditorListener
JTabbedPane	ChangeEvent	ChangeListener
JTree	TreeSelectionEvent TreeExpansionEvent TreeWillExpandEvent TreeModeEvent	TreeSelectionListener TreeExpansionListener TreeWillExpandListener TreeModeListener
JTimer	ActionEvent	ActionListener

表10.3 Swing提供的各监听者与各事件类成员方法之间的关系

事件监听者	成 员 方 法
CaretListener	caretUpdate(CaretEvent e)
CellEditorListener	editingCanceled(ChangeEvent e) editingStopped(ChangeEvent e)
ChangeListener	stateChanged(ChangeEvent e)
DocumentListener	changedUpdate(DocumentEvent e) insertUpdate(DocumentEvent e) removeUpdate(DocumentEvent e)
HyperlinkListener	hyperlinkUpdate(HyperlinkEvent e)
ListDataListener	contentsChanged(ListDataEvent e) intervalAdded(ListDataEvent e) intervalRemoved(ListDataEvent e)
ListSelectionListener	valueChanged(ListSelectionEvent e)
MenuDragMouseListener	menuDragMouseDragged(MenuDragMouseEvent e) menuDragMouseEntered(MenuDragMouseEvent e) menuDragMouseExited(MenuDragMouseEvent e) menuDragMouseReleased(MenuDragMouseEvent e)

表10.3 Swing提供的各监听者与各事件类成员方法之间的关系

MenuKeyListener	menuKeyPressed(MenuKeyEvent e) menuKeyReleased(MenuKeyEvent e) menuKeyTyped(MenuKeyEvent e)
MenuListener	menuCanceled(MenuEvent e) menuDeselected(MenuEvent e) menuSelected(MenuEvent e)
PopupMenuListener	popupMenuCanceled(PopupMenuEvent e) popupMenuWillBecomelavisible(PopupMenuEvent e) popupMenuWillB ecomeVisible(PopupMenuEvent e)
TableColumnModelListener	columnAdded(TableColumnModelEvent e) columnMarginChanged(ChangeEvent e) columnMoved(TableColumnModelEvent e) columnRemoved(TableColumnModelEvent e) columnSelectionChanged(ListSelectionEvent e)
TableModelListener	tableChanged(TableModelEvent e)

10.2.3 Java.awt事件类继承关系

Java语言提供了一组事件类来处理不同对象产生的事件。Java.awt事件类的继承关系如下：

java.awt.Object

java.util.Event

Java.util.Event.Object

. java.awt.AWTEvent

java.awt.event.ActionEvent.

java.awt.event.AdjustmentEvent

java.awt.event.InvocationEvent

java.awt.event.ItemEvent

java.awt.event.TextEvent

java.awt.event.ComponentEvent

java.awt.event.ContainerEvent

java.awt.event.FocusEvent

java.awt.event.PaintEvent

java.awt.event.WindowEvent

java.awt.event.InputEvent

java.awt.event.KeyEvent

java.awt.event.MouseEvent

10.2.4 AWT中的事件与事件监听者

Swing并不是用来取代原有的AWT的，事实上当我们使用Swing组件时常常还是需要使用AWT功能的。例如，鼠标和键盘操作、窗口的变化、组件的增加或删除等都是比较低层的事件，对于这些事件我们必须使用AWT包提供的处理方法来处理。

所有的Swing组件都是java.awt.Component的子类，它们具有如下继承关系：

java.lang.Object

java.awt.Component

java.awt.Container

javax.swing.JComponent

javax.swing的各种swing组件

所以，可以利用java.awt.Component类与java.awt.Container类提供的事件及事件监听者来处理诸如鼠标和键盘操作等低层事件。

表10.4列出了java.awt.Component与java.awt.Container类提供的事件与事件监听者之间的关系。表10.5列出了各事件监听者与各成员方法之间的对应关系。同理，对于AWT中的不同事件需要不同的事件监听者，而每个监听者都有相应的成员方法，我们处理事件的程序代码要写在对应的成员方法体中。

表10.4 Component类与Container类提供的事件与事件监听者之间的关系

事　件 (Event)	事件监听者(Listener)	说　　明
ComponentEvent	ComponentListener	主要处理组件大小的改变、位置的改变及可见与不可见状态等
ContainerEvent	ContainerListener	主要处理组件的加入或移出容器
FocusEvent	FocusListener	主要处理取得焦点或移开焦点等操作
KeyEvent	KeyListener	主要处理键盘的操作
MouseEvent	MouseListener	主要处理鼠标是否在某个组件上、是否按下鼠标等操作
MouseMotionEvent	MouseMotionListener	主要追踪鼠标的位置
WindowEvent	WindowListener	主要处理窗口问题，如打开、关闭、最大或最小化等

表10.5 AWT的各事件监听者与各成员方法之间的关系

AWT 事件监听者	成 员 方 法
ActionListener	actionPerformed(ActionEvent e)
WindowListener	windowActivated(WindowEvent e) windowClosed(WindowEvent e) windowClosing(WindowEvent e) windowDeactivated(WindowEvent e) windowDeiconified(WindowEvent e) windowIconified(WindowEvent e) windowOpened(WindowEvent e)
MouseListener	mouseClicked(MouseEvent e) mouseEntered(MouseEvent e) mouseExited(MouseEvent e) mousePressed(MouseEvent e) mouseReleased(MouseEvent e)

表10.5 AWT的各事件监听者与各成员方法之间的关系

MouseMotionListener	mouseDragged(MouseEvent e) mouseMoved(MouseEvent e)
ContainerListener	componentAdded(ContainerEvent e) componentRemoved(ContainerEvent e)
ComponentListener	componentHidden(ComponentEvent e) componentMoved(ComponentEvent e) componentResized(ComponentEvent e) componentShown(ComponentEvent e)
FocusListener	focusGained(FocusEvent e) focusLost(FocusEvent e)
ItemListener	itemStateChanged(ItemEvent e)
KeyListener	keyPressed(KeyEvent e) keyReleased(KeyEvent e) keyTyped(KeyEvent e)
AdjustmentListener	adjustmentValueChanged(AdjustmentEvent e)

10.3 JLabel 组 件

JLabel组件被称为标签，它是一个静态组件，也是标准组件中最简单的一个组件。每个标签用一个标签类的对象表示，可以显示一行静态文本。标签只起信息说明的作用，而不接受用户的输入，也无事件响应。

创建标签JLabel类对象的构造方法如表10.6所示。

表10.6 JLabel类对象的构造方法

构 造 方 法	功能及参数说明
JLabel( )	创建一个空标签
JLabel(Icon icon)	创建图标为 icon 的标签
JLabel(Icon icon,int halig)	创建图标为 icon 的标签，并指定水平排列方式(LEFT、CENTER、RIGHT、LEADING 和 TRAILING)
JLabel(String text)	创建一个含有文字的标签
JLabel(String text,int halig)	创建一个含有文字的标签，并指定水平排列方式
JLabel(String text,Icon icon,int halig)	创建一个含有文字及图标的标签，并指定水平排列方式

表10.7 JLabel类常用成员方法

成 员 方 法	功 能 说 明
Icon getIcon( )	获取此标签的图标
void setIcon(Icon icon)	设置标签的图标
String getText( )	获取此标签的文本
void setText(String lable)	设置标签的文本
void setHorizontalAlignment(int alig)	设置标签内组件的水平对齐方式
void setVerticalAlignment(int alig)	设置标签内组件的垂直对齐方式
void setHorizontalTextPosition(int tp)	设置标签内文字与图标的水平相对位置
void setVerticalTextPosition (int tp)	设置标签内文字与图标的垂直相对位置

10.4 JButton组件与JToggleButton组件

JButton组件与JToggleButton组件通常被称为按钮，它是一个具有按下、抬起两种状态的组件。用户可以指定按下按钮(单击事件)时所执行的操作(事件响应)。按钮上通常有一行文字(标签)或一个图标以表明它的功能。此外，Swing组件中的按钮还可以实现下述效果：

(1) 改变按钮的图标，即一个按钮可以有多个图标，可根据Swing按钮所处的状态而自动变换不同的图标。

(2) 为按钮加入提示，即当鼠标在按钮上稍做停留时，在按钮边可出现提示，当鼠标移出按钮时，提示自动消失。

(3) 在按钮上设置快捷键。

(4) 设置默认按钮，即通过回车键运行此按钮的功能。

10.4.1 AbstractButton类的常用成员方法

表10.8 AbstractButton类常用成员方法

成 员 方 法	功 能 说 明
Icon getIcon( )	获取默认图标
void setIcon(Icon icon)	设置此按钮的默认图标
String getLabel( )	获取标签文本
void setLabel(String lable)	设置标签的文本
void setHorizontalAlignment(int alig)	设置文本与图标的水平对齐方式(CENTER,LEFT, RIGHT,LEADING,TRAILING)
void setVerticalAlignment(int alig)	设置文本与图标的垂直对齐方式(CENTER,TOP, BOTTOM)
void setHorizontalTextPosition(int tp)	设置文本与图标的水平相对位置(CENTER,LEFT, RIGHT,LEADING,TRAILING)
void setVerticalTextPosition (int tp)	设置文本与图标的垂直相对位置(CENTER,TOP, BOTTOM)

表10.8 AbstractButton类常用成员方法

String getText( )	获取此按钮的文本
void addChangeListener(ChangeListener I)	给按钮添加指定的 ChangeListener
void addActionListener(ActionListener I)	给按钮添加指定的 ActionListener
void addItemListener(ItemListener I)	给按钮添加指定的 ItemListener
void removeActionListener(ActionListener I)	从按钮中删除指定的 ActionListener
void remove ChangeListener(ChangeListener I)	从按钮中删除指定的 ChangeListener
void remove ItemListener(ItemListener I)	从按钮中删除指定的 ItemListener
void setPressedIcon(Icon pricon)	设置按钮按下时的图标
void setRolloverIcon(Icon roicon)	设置鼠标经过时按钮的图标
void setRolloverEnabled(boolean b)	设置翻转效果是否有效
void setRolloverSelectedIcon(Icon seicon)	设置按钮的翻转并选择图标
void setEnabled(boolean b)	设定按钮是否禁用
void setSelected(boolean b)	设置按钮的状态
void setText(String text)	设置按钮的文本
boolean isSelected( )	获取按钮的状态
Icon getSelectedIcon( )	获取按钮的图标

10.4.2 JButton类的构造方法

按钮可分为有、无标签的和有、无图标的等几种情况，因此，系统提供了表10.10所示的JButton类的构造方法来创建这几种按钮对象。

表10.9 JButton类构造方法

构 造 方 法	功 能 说 明
JButton( )	创建一个无标签的按钮
JButton(String text)	创建一个有标签的按钮
JButton(Icon icon)	创建一个有图标的按钮
JButton(String text, Icon icon)	创建一个有标签和图标的按钮

10.4.3 JToggleButton类的构造方法

JToggleButton按钮与JButton按钮的区别仅在于：当按下JButton按钮并释放鼠标后，按钮会自动弹起；按下JToggleButton按钮并释放鼠标后，按钮不会自动弹起，除非再按一次。表10.10列出了JToggleButton类的构造方法。

表10.10 JToggleButton类构造方法

构 造 方 法	功 能 说 明
JToggleButton( )	创建一个无标签的按钮
JToggleButton(String text)	创建一个标签为 text 的按钮
JToggleButton(String text,boolean selected)	创建一个有标签的按钮，且初始状态为 false
JToggleButton(Icon icon)	创建一个图标为 icon 的按钮
JToggleButton(Icon icon,boolean selected)	创建一个有图标的按钮，且初始状态为 false
JToggleButton(String text, Icon icon)	创建一个既有标签又有图标的按钮
JToggleButton(String text, Icon icon,boolean selected)	创建一个有标签和图标的按钮，且初始状态为 false

10.4.4 ActionEvent事件及其响应

按照Java的委托事件模型，当我们在所设计的用户界面上按下一个按钮时会激发一个事件，这个事件称为动作事件。动作事件由AWT的ActionEvent类的方法来处理。

1．动作事件

ActionEvent类含有ACTION_PERFORMED事件，它是引发某个动作的执行事件。能触发这个事件的动作包括：单击按钮；双击一个列表中的选项；选择菜单项；在文本框中输入回车等。

2．ActionEvent类可使用的主要方法

getSource( )方法：用来获取引发事件的对象名。

getActionCommand( )方法：用来获取对象的标签或事先为这个对象设置的命令名。

getSource( )方法是EventObject类提供的，getActionCommand( )方法是ActionEvent类提供的。由于ActionEvent类继承了EventObject类，因此，ActionEvent类可以使用这两个方法。

3．事件响应

当用户点击对象时，就会引发ActionEvent类代表的动作事件。例如，下面的语句：

对象名.addActionListener(this);

注册事件源对象的监听者对象为this，而且要求this对象的类必须声明该类并实现ActionListener接口。当事件发生时，引发的事件将被此事件的监听者监听到，并引用ActionListener的actionPerformed(ActionEvent e)方法响应动作事件。在此方法体中可以引用ActionEvent事件的getSource( )方法来获取引发事件的对象，也可以引用getActionCommand( )方法来获取对象标签或事先为这个对象设置的命令名。用户可在actionPerformed( )方法体中写入处理此事件的程序代码。

10.4.5 应用举例

【示例程序c10_1.java】 编写一个单击按钮时改变标签文本与按钮文本的内容的程序。

```
import java.awt.*;

import javax.swing.*;

import java.awt.event.*;

//声明该类实现ActionListener 接口，监听者对象是c10_1类的对象

public class c10_1 extends JApplet implements ActionListener
//JApplet是Applet的子类
```

```
{  Container cp=getContentPane( );  //创建窗口容器对象
   Icon ro=new ImageIcon("g1.gif");  //创建图标对象
   Icon ge=new ImageIcon("g2.gif");
   Icon pr=new ImageIcon("g3.gif");
   JButton bt=new JButton( );  //创建按钮对象
   Icon icon=new ImageIcon("g4.jpg");
   JLabel lb=new JLabel("Java",icon,JLabel.CENTER);  //创建标签对象
  public void init( )
   {
```

```
bt.setRolloverEnabled(true); //将按钮图标变化功能打开
bt.setText("OK");  //添加按钮文本
bt.setHorizontalTextPosition(JLabel.CENTER); //将按钮文字放在图标中间
bt.setVerticalTextPosition(JLabel.BOTTOM);  //设置按钮文字在图标下方
cp.add(lb,BorderLayout.NORTH); //添加标签在JApple界面的北部位置上
cp.add(bt,BorderLayout.SOUTH);
bt.setIcon(ge); //设置鼠标离开按钮的图标
bt.setRolloverIcon(ro); //设置鼠标在按钮上的图标
bt.setPressedIcon(pr);  //设置鼠标按下按钮的图标
bt.addActionListener(this); //注册bt的监听者对象this
}
```

```
// actionPerformed( )方法在单击bt时被系统自动调用
public void actionPerformed(ActionEvent e)
{ if(e.getSource( )==bt) //判断动作事件是否由bt1引发的
    {
     if(lb.getText( )=="Hello")  //修改标签文本
        lb.setText(“你好!”);
      else
        lb.setText("Hello");
      if(bt.getText( )=="OK")    //修改按钮文本
        bt.setText("确定");
      else
        bt.setText("OK");
    }
 }
}
```

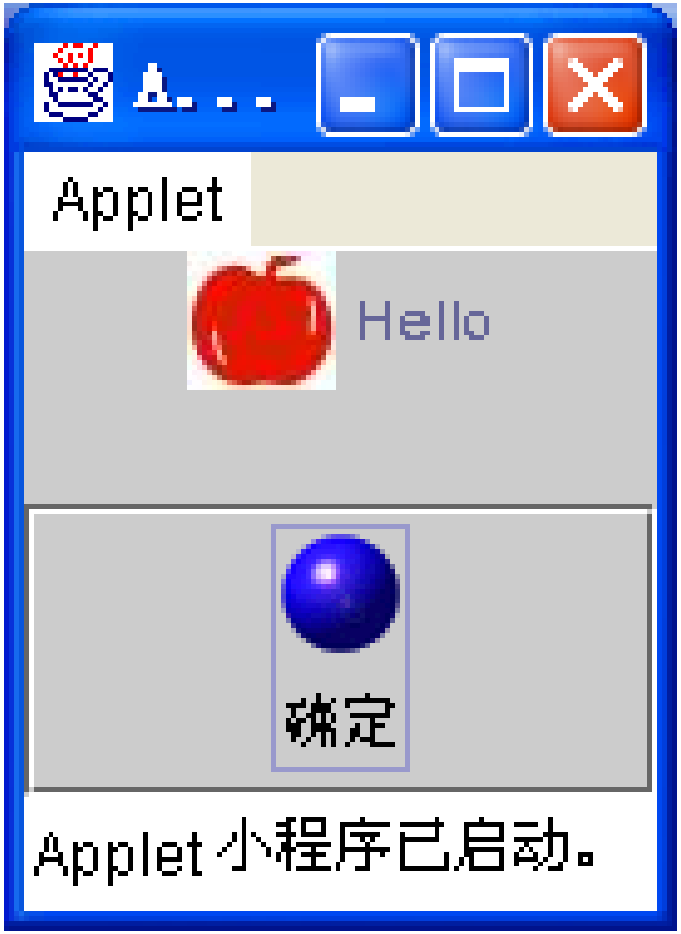

图10.2 程序c10_1的运行结果

程序说明：

(1) javax.swing.JApplet是java.awt.Applet的子类，当向JApplet中添加Swing组件时要使用JApplet的getContentPane( )方法获得一个Container对象，再引用这个Container对象的add( )方法将JComponent及其子类对象添加到JApplet中。

(2) 程序中创建的按钮有一个图标，根据鼠标的移动来改变图标。

(3) 当单击一个按钮时，按钮为监听者对象发送一个称为“动作事件”的事件对象。产生事件的按钮是事件源，通常说这个按钮激活一个事件。在这个程序中，bt是事件源，为了能够让按钮知道为哪个监听对象发送动作事件对象，必须注册按钮的监听者对象。我们通过调用按钮对象的addActionListener( )方法来实现，即bt.addActionListener(this)，其中this为监听者对象，即c10_1类的对象(通常把包含事件源的对象作为监听对象)。为了把c10_1类指定为一个动作监听对象，必须声明该类实现ActionListener接口，即

public class c10_1 extends JApplet implements ActionListener

其中，ActionListener 接口仅仅包含了一个抽象方法actionPerformed( )。ActionEvent的对象e代表一个动作事件，并且支持那些用来发现所产生事件的特征的方法。

当按钮激活一个动作事件时，则此按钮引发的事件将被此事件的监听者监听到并调用actionPerformed(ActionEvent e)方法对该事件进行处理。

在方法体中调用e.getSource( )方法来获取引发事件的按钮对象。

(4) 把组件放到用户希望的位置，在Java语言中是通过布局来实现的。JApplet界面上的默认布局是BordLayout布局，它将界面划分为东、南、西、北、中五个区域，我们把组件添加到JApplet界面时，可以将组件放在这五个区域的任一位置上。更复杂的内容我们将在后面的章节介绍。

(5) 程序中使用了AbstractButton类的成员方法，详细描述见表10.8。

10.5 JCheckBox和JRadioButton组件

JCheckBox组件被称为复选框(也称检测框)，它提供“选中/ON”和“未选中/OFF”两种状态。用户点击某复选框就会改变该复选框原有的状态。

JRadioButton组件被称为选项按钮，在Java中JRadioButton组件与JCheckBox组件功能完全一样，只是图形不同，复选框为方形图标，选项按钮为圆形图标。由于目前所使用软件的RadioButton多为单选按钮，即在同类的一组组件中，用户只能选择其中之一为ON，其余为OFF。Java为了与其他系统一致，专门提供了javax.swing.ButtonGroup类，这个类的功能就是实现诸如JRadioButton、JRadioButtonMenuItem与JToggleButton等组件的多选一功能。ButtonGroup类可被AbstractButton类的子类所使用。

10.5.1 JCheckBox类的构造方法

表10.11 JCheckBox类构造方法

构 造 方 法	功 能 说 明
JCheckBox( )	创建一个无标签的复选框对象
JCheckBox(String text)	创建一个有标签的复选框对象
JCheckBox(String text,boolean selected)	创建一个有标签的复选框对象，且初始状态为 false
JCheckBox(Icon icon)	创建一个有图标的复选框对象
JCheckBox(Icon icon,boolean selected)	创建一个有图标的复选框对象，且初始状态为 false
JCheckBox(String text, Icon icon)	创建一个有标签和图标的复选框对象
JCheckBox(String text, Icon icon,boolean selected)	创建一个有标签和图标的复选框对象，且初始状态为 false

10.5.2 JRadioButton的构造方法

表10.12 JRadioButton类构造方法

构 造 方 法	功 能 说 明
JRadioButton( )	创建一个无标签的 JRadioButton 对象
JRadioButton(String text)	创建一个有标签的 JRadioButton 对象
JRadioButton(String text,boolean selected)	创建一个有标签的 JRadioButton 对象，且初始状态为 false
JRadioButton(Icon icon)	创建一个有图标的 JRadioButton 对象
JRadioButton(Icon icon,boolean selected)	创建一个有图标的 JRadioButton 对象，且初始状态为 false
JRadioButton(String text, Icon icon)	创建一个有标签和图标的 JRadioButton 对象
JRadioButton(String text, Icon icon,boolean selected)	创建一个有标签和图标的 JRadioButton 对象，且初始状态为 false

10.5.3 ItemEvent事件

1. 选择事件

ItemEvent类只包含一个事件ITEM_STATE_CHANGED，引发这类事件的动作包括：

(1) 改变复选框JCheckbox对象的选中或不选中状态；

(2) 改变单选按钮JRadioButton对象的选中或不选中状态；

(3) 改变下拉列表框JComboBox对象中选项的选中或不选中状态；

(4) 改变菜单项JMenuItem对象中选项的选中或不选中状态；

(5) 改变JCheckboxMenuItem对象中选项的选中或不选中状态。

2．ItemEvent类的主要方法

(1) ItemSelectable getItemSelectable( )。getItemSelectable( )方法返回引发选中状态变化的事件源，例如JCheckbox对象。能引发选中状态变化的事件都必须是实现了ItemSelectable接口类的对象，该方法的返回值就是这些类的对象的引用。此外，ItemEvent类的事件也可以使用其父类EventObject类提供的getSource( )方法返回引发选中状态变化的事件源。

(2) Object getItem( )。getItem( )方法返回引发选中状态变化事件的具体选择项，例如JComboBox中的具体item。通过调用这个方法可以知道用户选中了哪个选项。

(3) int getStateChange( )。getStateChange( )方法返回此组件到底有没有被选中。它的返回值是一个整型值，通常用ItemEvent类的静态常量SELECTED(代表选项被选中)和DESELECTED(代表选项被放弃或不选)来表达。

3．事件响应

当用户点击对象使其选中状态发生变化时，就会引发ItemEvent类代表的选择事件。例如下面的语句：

对象名.addItemListener(this);

注册事件源对象的监听者对象为this，而且要求this对象的类必须声明该类实现ItemListener接口。当事件发生时，引发的事件将被此事件的监听者监听到，并引用ItemListener中的itemStateChanged(ItemEvent e)方法响应对象的状态改变。在此方法体中，引用ItemEvent事件的e.getItemSelectable( )方法以获得引发选择事件的事件源对象，再引用getStateChange( )方法获取选择事件之后的状态。用户在itemStateChanged( )方法体中编写处理事件的程序代码。

10.5.4 应用举例

【示例程序c10_2.java】 根据复选框及选择按钮来改变标签组件的文本大小及颜色。

```
import javax.swing.*;

import java.awt.*;

import java.awt.event.*;

public class c10_2 extends JApplet implements
ItemListener,ActionListener

{  int i1=0,i2=0,i3=0;

   int fonti=10;
```

```
Font font;
Container ctp=getContentPane( );
JLabel lb=new JLabel("请选择");
JCheckBox cb1,cb2,cb3;  //声明复选框对象
JRadioButton r1,r2,r3;   //声明按钮对象
ButtonGroup bg=new ButtonGroup( ); //创建按钮组对象，实现JRadioButton多选一功能
public void init( )
{ ctp.setLayout(new FlowLayout( )); //设置布局方式为流式布局
  cb1=new JCheckBox("红色",false); //创建复选框
  cb1.addItemListener(this);       //注册cb给监听者this
```

```
ctp.add(cb1);  //添加复选框在界面上
cb2=new JCheckBox("绿色",false);
cb2.addItemListener(this);
ctp.add(cb2);
cb3=new JCheckBox("蓝色",false);
cb3.addItemListener(this);
ctp.add(cb3);
r1=new JRadioButton("10");
r1.addActionListener(this);
ctp.add(r1);  //加载按钮到界面上
```

```
        r2=new JRadioButton("16");
        r2.addActionListener(this);
        ctp.add(r2);
        r3=new JRadioButton("24");
        r3.addActionListener(this);
        ctp.add(r3);
        bg.add(r1);  //加载按钮到按钮组
        bg.add(r2);
        bg.add(r3);
        ctp.add(lb);  //加载标签到界面上
    }
```

```
public void itemStateChanged(ItemEvent e)
{  JCheckBox cbx=(JCheckBox)e.getItem( );
   if (cbx.getText( )=="红色")
    {  if(e.getStateChange( )==e.SELECTED)i1=255;  //判断组件是否被选
       else i1=0;}
   if (cbx.getText( )=="绿色")
    {if(e.getStateChange( )==e.SELECTED)i2=255;
       else i2=0;  }
   if (cbx.getText( )=="蓝色")
    {if(cbx.isSelected( ))i3=255;  //判断组件是否被选
       else i3=0;  }
   font=new Font("宋体",Font.BOLD,fonti);
   lb.setFont(font);
   lb.setForeground(new Color(i1,i2,i3));
}
```

```
public void actionPerformed(ActionEvent e)
  {   String rbt=e.getActionCommand( );
if (rbt=="10")    fonti=10;
      else if (rbt=="16")    fonti=16;
          else    fonti=24;
      font=new Font("宋体",Font.BOLD,fonti);
      lb.setFont(font);
      lb.setForeground(new Color(i1,i2,i3));
   }
}
```

图10.3 程序c10_2的运行结果

10.6 JComboBox组件

10.6.1 JComboBox类的构造方法及成员方法

表10.13 JComboBox类的构造方法和常用成员方法

方法		说明
构造方法	JComboBox(Vector items)	使用向量表 items 构造一个 JComboBox 对象
	JComboBox( )	构造一个空的 JComboBox 对象。必要时可使用 addItem 方法添加选项
	JComboBox(ComboBoxModela Model)	从已有的 Model 获取选项，构造 JComboBox 对象
	JComboBox(Object[ ] items)	使用数组构造一个 JComboBox 对象
成员方法	void addActionListener(ActionListener e)	添加指定的 ActionListener
	void addItemListener(ItemListener aListener)	添加指定的 ItemListener
	void addItem(Object anObject)	给选项表添加选项
	String getActionCommand( )	获取动作命令
	Object getItemAt(int index)	获取指定下标的列表项
	int getItemCount( )	获取列表中的选项数
	int getSelectedIndex( )	获取当前选择的下标
	int getSelectedItem( )	获取当前选择的项

10.6.2 事件响应

JComboBox组件能够响应的事件分为选择事件与动作事件。若用户选取下拉列表中的选择项时，则激发选择事件，使用ItemListener事件监听者进行处理；若用户在JComboBox上直接输入选择项并回车时，则激发动作事件，使用ActionListener事件监听者进行处理。

下面通过一个具体的程序来说明按钮的事件响应。

【示例程序c10_3.java】 在JComboBox组件中添加4个学生的名字选项，当点击下拉列表选择项时得到学生的名字，将他的成绩用标签文本显示。

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class c10_3 extends JApplet implements ItemListener
{
   Container ctp=getContentPane( );
   JLabel lb1=new JLabel("姓名:"),
          lb2=new JLabel("英语:"),
```

```
        lb3=new JLabel("     ");
        String name[ ]={"李林","赵欣","张扬","童梅"},
        score[ ]={"80","104","75","87"};
JComboBox cbx=new JComboBox( );  //创建下拉式列表框对象
public void init( )
{
   ctp.setLayout(new FlowLayout( ));  //设置流式布局
   for (int j=0;j<name.length;j++)    //添加选项到下拉式列表框对象中
      cbx.addItem(name[j]);
   ctp.add(lb1);
   ctp.add(cbx);  //添加下拉式列表框对象到容器上
```

```
        cbx.addItemListener(this);  //注册cbx给监听对象
    ctp.add(lb2);
    ctp.add(lb3);
  }
public void itemStateChanged(ItemEvent e)
{
    int c=0;
    String str=(String)e.getItem( );  //获取所选项给str
    for(int i=0;i<name.length;i++)
      if(str==name[i])   //判断str是否是name数组中某个元素的内容
         c=cbx.getSelectedIndex( );  //将该选项的下标给c
     lb3.setText(score[c]);  //获取该学生的成绩
  }
}
```

程序说明：

下拉式列表框产生ItemEvent代表的选择事件。该程序中的语句cbx.addItemListener(this);表示注册JComboBox类的对象cbx给监听者对象。当用户单击下拉列表的某个选项时，系统自动产生一个包含这个事件有关信息的ItemEvent类的对象e，并把该对象作为实际参数传递给被自动调用的监听者的选择事件响应方法：itemStateChanged(ItemEvent e)。在这个方法中通过调用ItemEvent 事件的方法e.getItem( )获得引发当前选择事件的下拉列表事件源(被选中的项),再调用getSelectedIndex( )获取该选项的下标值，从而得到name数组的下标值，最终将这个元素的内容作为新的标签文本输出。

图10.4 c10_3运行结果

10.7 JList 组 件

JList称为列表组件，它将所有选项放入列表框中。如果将JList放入滚动面板(JScrollPane)中，则会出现滚动菜单效果。利用JList提供的成员方法，用户可以指定显示在列表框中的选项个数，而多余的选项则可通过列表的上下滚动来显现。

JList组件与JComboBox组件的最大区别是：JComboBox组件一次只能选择一项，而JList组件一次可以选择一项或多项。选择多项时可以是连续区间选择(按住Shift键进行选择)，也可以是不连续的选择(按住Ctrl键进行选择)。

10.7.1 JList类的构造方法及成员方法

表10.14 JList类的构造方法和成员方法

方法		说明
构造方法	JList(Vectorl istData)	使用包含元素的向量构造 JList 对象
	JList( )	使用空的模式构造 JList 对象
	JList(ListModel dataModel)	使用 dataModel 模式构造 JList 对象
	JList(Object[ ] listData)	使用指定的数组构造 JList 对象
成员方法	void addListSelectionListener(ListSelectionListener e)	添加指定的 ListSelectionListener
	int getSelectedIndex( )	获取所选项的第一个下标
	int getSelectedIndices( )	获取所有选项的下标
	void setSelection Background(Color c)	设置单元格的背景颜色
	void setSelection Foreground(Color c)	设置单元格的前景颜色
	int getVisibleRowCount( )	得到可见的列表选项值
	void setVisibleRowCount(int num)	设置可见的列表选项

10.7.2 ListSelectionEvent事件

JList组件的事件处理一般可分为两种：一种是当用户单击列表框中的某一个选项并选中它时，将产生ListSelectionEvent类的选择事件，此事件是Swing的事件；另一种是当用户双击列表框中的某个选项时，则产生MouseEvent类的动作事件。JList类通过locatToindex( )方法来得知是单击还是双击。

若希望实现JList的ListSelectionEvent事件，首先必须声明实现监听者对象的类接口ListSelectionListener，并通过JList类的addListSelectionListener( )方法注册文本框的监听者对象，再在ListSelectionListener接口的valueChanged (ListSelectionEvent e)方法体中写入有关代码，就可以响应ListSelectionEvent事件了。下面通过示例程序来加以说明。

【示例程序c10_4.java】 设置一个JLabel组件和JList组件，点击列表框中的选项，将所选项的值作为JLabel组件的文本输出。

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
public class c10_4 extends JApplet implements
ListSelectionListener
{   JList lis=null;
    JLabel lb=null;
```

```
String[ ] s={"小学","初中","高中","大学","研究生"};
public void init( )
 {  Container cp=getContentPane( );
    cp.setLayout(new BorderLayout( ));
    lb=new JLabel( );
    lis=new JList(s);
    lis.setVisibleRowCount(3);  //设置列表框的可见选项行数，选项超过则出现滚动条
    lis.setBorder(BorderFactory.createTitledBorder("请选择"));//设置列表框的边框文本
```

```
    lis.addListSelectionListener(this);  //注册lis给监听者对象
    cp.add(lb,BorderLayout.NORTH);
     //将lis对象放入滚动容器，再将此容器加载到界面上。
    cp.add(new JScrollPane(lis),BorderLayout.CENTER);
  }
public void valueChanged(ListSelectionEvent e)
{  int m=0;
    String str="选取的是： ";
    //取得所有选项的下标值给index数组
```

```
        int[ ] index = lis.getSelectedIndices( );
        for(int i=0;i<index.length;i++)
        { //根据取得的下标值，找到相应的数组元素
            m=index[i];
            str=str+s[m]+"  ";
          }
       lb.setText(str);//输出选中项的值
    }
}
```

上述程序中的语句：

lis.addListSelectionListener(this);

表示把lis注册给ListSelectionEvent的监听者ListSelectionListener。当用户单击某个选项时，系统会自动引用ListSelectionListener的valueChanged( )方法来处理选项的改变。

程序c10_4的运行结果见图10.5。

图10.5 程序c10_4的运行结果

10.8 JTextField与JTextArea组件

10.8.1 JTextField组件的构造方法及成员方法

JtextField被称为文本框。它定义了一个单行条形文本区，可以输出任何基于文本的信息，也可以接受用户的输入。表10.15列出了JTextField类的构造方法和成员方法。

表10.15 JTextField类构造方法和成员方法

方法		功能说明
构造方法	JTextField( )	创建一个 JTextField 对象
	JTextField(int n)	创建一个列宽为 n 的空 JTextField 对象
	JTextField(String s)	创建一个 JTextField 对象，并显示字符串 s
	JTextField(String s,int n)	创建一个 JTextField 对象，并以指定的字宽 n 显示字符串 s
	JTextField(Document doc,String s,int n)	使用指定的文件存储模式创建一个 JTextField 对象，并以指定的字宽 n 显示字符串 s
成员方法	int getColumns( )	获取此对象的列数
	void setColumns(int Columns)	设置此对象的列数
	void addActionListener(ActionListener e)	添加指定的动作事件监听程序
	void setFont(Font f)	设置字体
	void setHorizontalAlignment(int alig)	设置文本的水平对齐方式(LEFT、CENTER、RIGHT)
	void setActionCommand(String cmd)	设置动作事件使用的命令字符串

10.8.2 JTextArea组件的构造方法及成员方法

JTextArea被称为文本域。它与文本框的主要区别是：文本框只能输入/输出一行文本，而文本域可以输入/输出多行文本。表10.16列出了JTextArea类的构造方法和成员方法。

表10.16 JTextArea类构造方法和成员方法

方法		功能说明
构造方法	JTextArea( )	创建一个 JTextArea 对象
	JTextArea (int n,int m)	创建一个具有 n 行 m 列的空 JTextArea 对象
	JTextArea(String s)	创建一个 JTextArea 对象，并显示字符串 s
	JTextArea(String s,int n,int m)	创建一个 JTextArea 对象，并以指定的行数 n 和列数 m 显示字符串 s
	JTextArea(String s,int n,int m,int k)	创建一个 JTextArea 对象，并以指定的行数 n、列数 m 和滚动条的方向显示字符串 s
	JTextArea(Document doc)	使用指定的文件存储模式创建一个 JTextArea 对象
	JTextArea(Document doc,String s,int n)	使用指定的文件存储模式创建一个 JTextArea 对象，并以指定的字宽 n 显示字符串 s
成员方法	void setFont(Font f)	设置字体
	void insert(String str,int pos)	在指定的位置插入指定的文本
	void append(String str)	将指定的文本添加到末尾
	void replaceRange(String str,int start,int end)	将指定范围的文本用指定的新文本替换
	public int getRows( )	获取此对象的行数
	public void setRows(int rows)	设置此对象的行数
	public int getColumns( )	获取此对象的列数
	public void setColumns(int Columns)	设置此对象的列数

10.8.3 事件处理

JTextField类只引发ActionEvent事件，当用户在文本框中按回车键时引发。JTextArea的事件响应由JTextComponent类决定。JTextComponent类可以引发两种事件：DocumentEvent事件与UndoableEditEvent事件。当用户修改了文本区域中的文本，如做文本的增、删、改等操作时，TextComponent类将引发DocumentEvent事件；当用户在文本区域上撤消所做的增、删、改时，TextComponent类将引发UndoableEditEvent事件。

(1) 实现文本框的ActionEvent事件。当监听者对象的类声明实现了ActionListener接口，并且通过textField.addActionListener(this)语句注册文本框的监听者对象后，监听程序内部动作事件的actionPerformed(ActionEvent e) 方法就可以响应动作事件了。

(2) 实现文本区域的DocumentEvent事件。当监听者对象的类声明实现了DocumentEditListener接口，并且通过textArea.addDocumentListener(this)语句注册文本区域的监听者对象后，监听程序内部的DocumentListener接口的下列三个方法之一就可以响应DocumentEvent事件了：

changedUpdate(DocumentEvent e); //修改文本操作

inset Update(DocumentEvent e); //增加文本操作

remove Update(DocumentEvent e); //删除文本操作

(3) 实现文本域的UndoableEditEvent事件。当监听者对象的类声明实现了UndoableEditListener接口，并且通过textArea.addUndoableEditListener(this)语句注册文本域的监听者对象后，监听程序内部的UndoableEditListener接口的UndoableEditHappened(UndoableEditEvent e)方法就可以响应UndoableEditEvent事件了。

10.8.4 应用举例

【示例程序c10_5.java】 文本框和文本域组件的使用。

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class c10_5 extends JApplet implements ActionListener
{
   Container ctp=getContentPane( );
   ImageIcon icon1=new ImageIcon("g1.gif"),
                  icon2=new ImageIcon("g2.gif");
```

```
JLabel lb1=new JLabel("输入文字后按回车:",icon1,JLabel.CENTER),
     lb2=new JLabel("输出结果:",icon2,JLabel.CENTER);
JTextField tf1=new JTextField(10);  //创建文本框对象
JTextArea tf2=new JTextArea(5,10);  //创建文本区域对象
public void init( )
{
  ctp.setLayout(new FlowLayout( ));
  ctp.add(lb1);
  ctp.add(tf1);
  ctp.add(lb2);
  ctp.add(tf2);
  tf1.addActionListener(this);
}
```

```
    public void actionPerformed(ActionEvent e)
    {
      String str;
      str=tf1.getText( );    //获得文本框的文本给str(此方法是
JTextComponent类的方法)
      //   tf2.setText(str);
      tf2.append(str+"\n");  //将str添加到文本区域中(append方法是
JTextArea类的方法)
    }
}
```

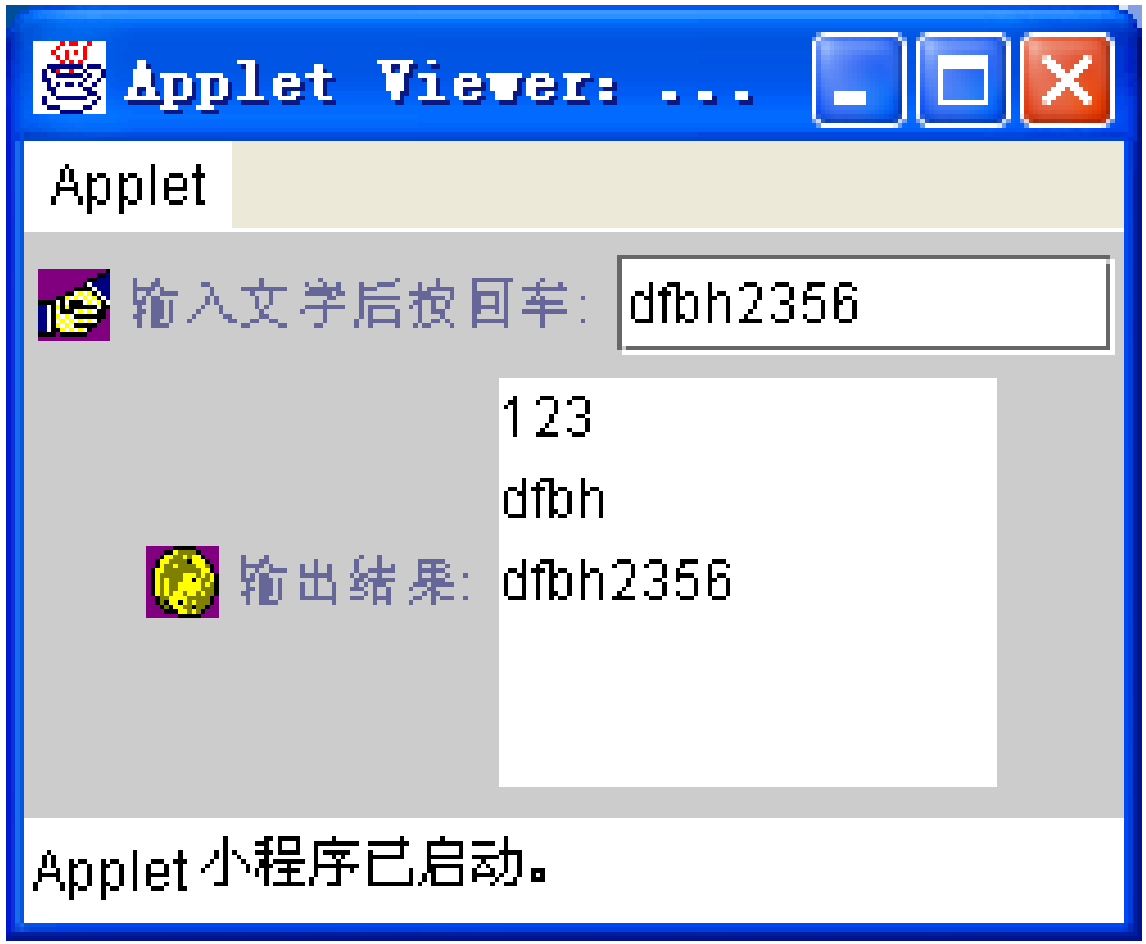

图10.6 程序c10_5的运行结果

练习：按钮、文本框和文本域组件的使用。

文本框输入内容

点击按钮

文本框输入的内容在文本域组件中显示

程序打包，命名规则：姓名-学号-班级-第12讲作业.rar

提交教学平台