Java程序设计

计算机科学与技术

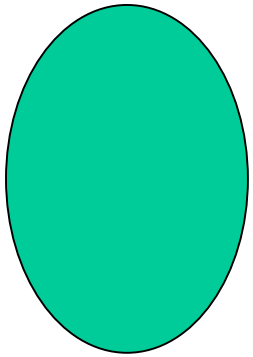
贡正仙

zhxgong@suda.edu.cn

第11章 Swing程序设计

ss

dfdsf
d



导读

- 重点

  布局管理器

  事件处理

11. Swing概述

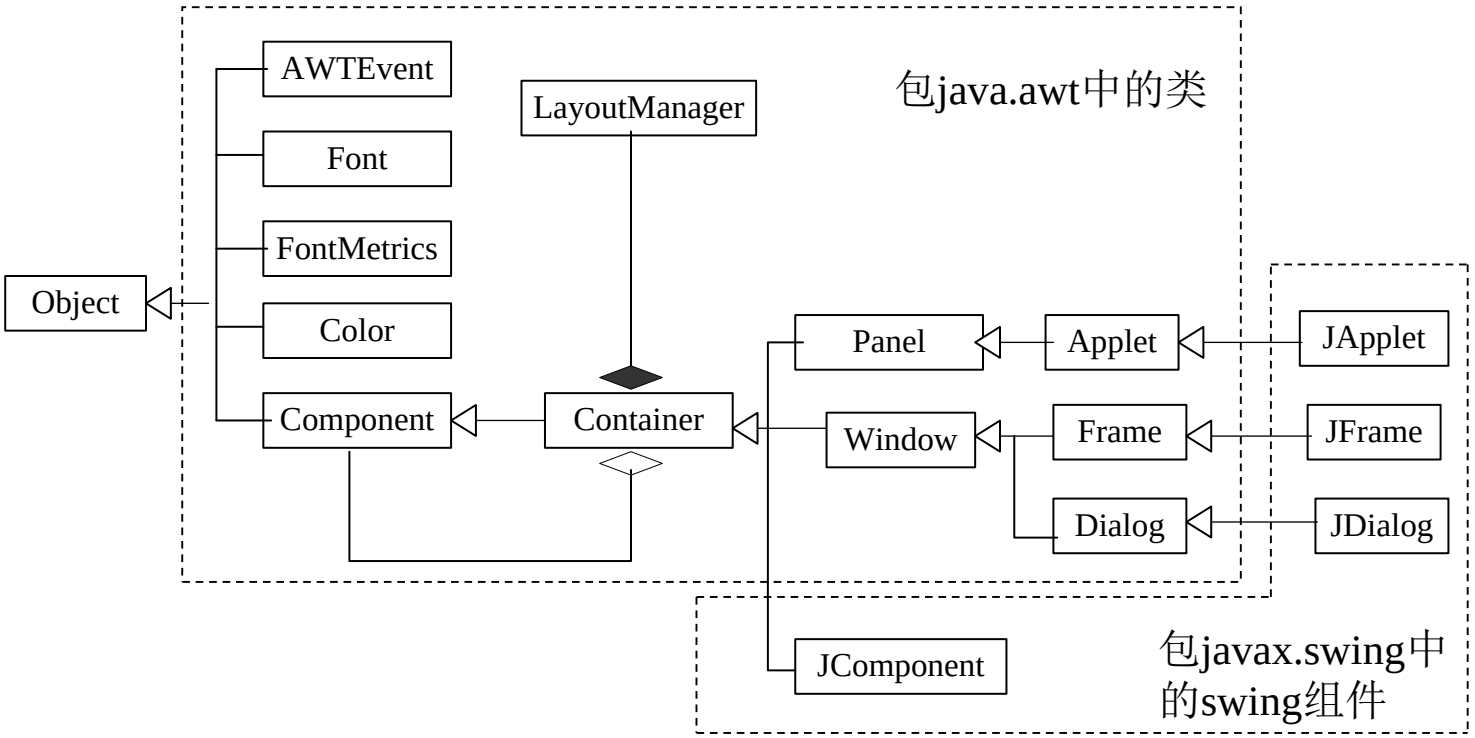
- GUI－Graphic User Interface
  - 构成：
    - 抽象窗口工具集AWT (Abstract Window Toolkit)
      - 对运行的各平台，组件通过各自的代理映射成平台特定组件
      - 适用于简单的GUI程序，对复杂的GUI项目不适用
      - 容易发生平台特定故障
      - 重型组件
    - Swing组件库
      - 大多数组件直接使用Java代码编写
      - 更少依赖目标机器上的平台、本地资源
      - 轻型组件
    - Applet——自学

11. Swing概述

- GUI－Graphic User Interface
  - 说明：
    - Swing组件不能取代AWT的全部类
      - 能替代AWT的用户界面组件（Button、TextField等）
      - 辅助类(Graphics、Color、Font、FontMetrics、LayoutManager)保持不变
      - 使用AWT的事件模型
    - 课程介绍方式：
      - AWT中仍被Swing使用的内容：
        - » 容器和LayoutManager
        - » 事件模型
        - » 辅助类的使用——绘图
      - Swing组件
        - » 基本用法
        - » 各组件展开——简介

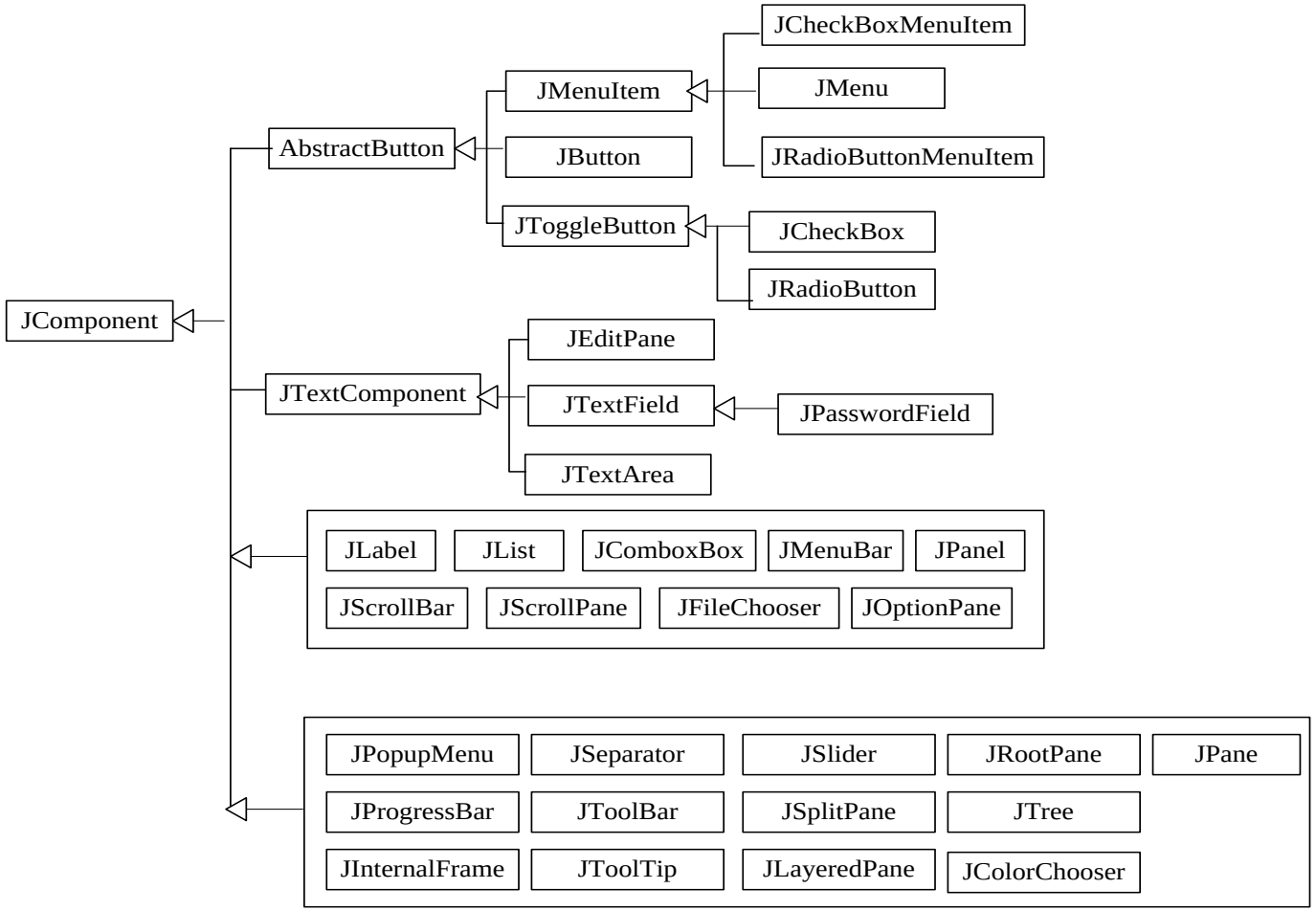
Java图形API

- Java图形程序设计所用类的层次结构



Java图形API

- Java图形程序设计所用类的层次结构



11.2 创 建 窗 体

在开发Java应用程序时，通常情况下利用JFrame类创建窗体。

利用JFrame类创建的窗体分别包含标题、最小化按钮、最大化按钮和关闭按钮。

JFrame类提供了一系列用来设置窗体的方法。

--》

JFrame常用方法

JFrame() 创建一个无标题的窗口。
JFrame(String s) 创建标题为s的窗口。
public void setTitle(String title)方法，可以设置窗体的标题
public void setBounds(int a,int b,int width,int height 设置窗口的初始位置是(a,b)，即距屏幕左面a个像素、距屏幕上方b个像素；窗口的宽是width，高是height。
public void setSize(int width,int height) 设置窗口的大小。
public void setLocation(int x,int y) 设置窗口的位置，默认位置是(0,0)。
public void setVisible(boolean b) 设置窗口是否可见，窗口默认是不可见的。
public void setResizable(boolean b) 设置窗口是否可调整大小，默认可调整大小。

例题1效果图

图 10.2 创建窗口

- 设置标题
- 设置大小
- 设置是否可以改变窗口大小
- 设置窗口位置-》窗体居中

11.2创建窗体

窗体居中问题（选学）

java.awt.Toolkit类可用于得到屏幕的宽和高

```
Dimension
   dm=Toolkit.getDefaultToolkit().getScreenSize();
int x=dm.width;
int y=dm.height;
x=(x-jf.getWidth())/2;
y=(y-jf.getHeight())/2;
f.setLocation(x,y);
```

关闭窗 体

在创建窗体时，通常情况下需要设置关闭按钮的动作。**public void setDefaultCloseOperation(int operation)** 该方法用来设置单击窗体右上角的关闭图标后，程序会做出怎样的处理。

该方法的入口参数可以从JFrame类提供的静态常量中选择，可选的静态常量如下表所示。

静 态 常 量	常 量 值	执 行 操 作
HIDE_ON_CLOSE	1	隐藏窗口，为默认操作
DO_NOTHING_ON_CLOSE	0	不执行任何操作
DISPOSE_ON_CLOSE	2	移除窗口
EXIT_ON_CLOSE	3	退出窗口

11.3 常用布局管理器

布局管理器负责管理组件在容器中的排列方式。Java是跨平台的开发语言，它能够实现“一次编写，到处运行”，为实现这个目标，Java所开发的应用程序，必须使用布局管理器管理每个容器中组件的布局，因为不同的平台（即操作系统或者手机等硬件平台）显示组件的策略和方式是不同的，无法确定不同平台的组件大小和样式。

11.3.1 不使用布局管理器

在布局管理器出现之前，所有的应用程序都使用直接定位的方式排列容器中的组件，例如VC、Delphi、VB等开发语言都使用这种布局方式。Java也提供了对绝对定位的组件排列方式的支持，但是这样布局的程序界面不能保证在其他操作平台中也能正常显示。如果需要开发的程序只在单一的系统中使用，可以考虑使用这种布局管理方式。

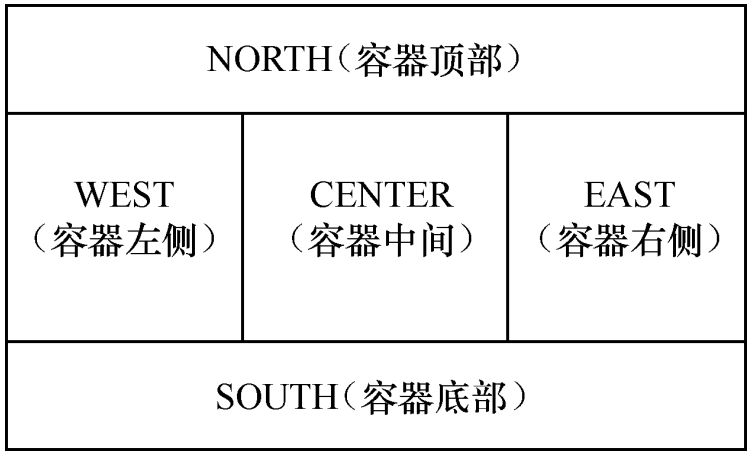
通过setLayout(LayoutManager mgr)方法设置组件容器采用的布局管理器，如果不采用任何布局管理器，则可以将其设置为null，例如：

```
getContentPane().setLayout(null);
```

【例2】

11.3.2 BorderLayout布局管理器

由BorderLayout类实现的布局管理器称为边界布局管理器，它的布局方式是将容器划分为5个部分。**边界布局管理器为JFrame窗体的默认布局管理器**，其布局格式如下图所示。



BorderLayout布局管理器

如果组件容器采用了边界布局管理器，将组件添加到容器时，需要设置组件的显示位置，通过方法add(Component comp, Object constraints)添加并设置，该方法的第一个参数为欲添加的组件对象，第二个参数为组件的显示位置，可以从BorderLayout类的静态常量中选择，如下表所示。

静 态 常 量	常 量 值	组件对齐方式
CENTER	"Center"	显示在容器中间
NORTH	"North"	显示在容器顶部
SOUTH	"South"	显示在容器底部
WEST	"West"	显示在容器左侧
EAST	"East"	显示在容器右侧

边界布局管理器默认组件水平间距和垂直间距均为0像素，可以通过BorderLayout类的方法setHgap(int hgap)和setVgap(int vgap)设置组件的水平间距和垂直间距。

【例4】

11.3.3 FlowLayout布局管理器

由FlowLayout类实现的布局管理器称为流布局管理器，它的布局方式是首先在一行上排列组件，当该行没有足够的空间时，则回行显示，当容器的大小发生改变时，将自动调整组件的排列方式。如下图所示：

【例3】

FlowLayout布局管理器

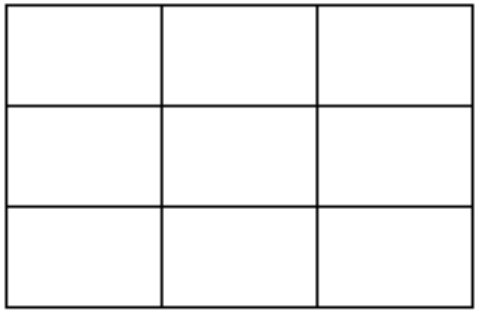
流布局管理器默认为居中显示组件，可以通过FlowLayout类的setAlignment(int align)方法设置组件的对齐方式，该方法的参数可以从FlowLayout类的静态常量中选择，如下表所示。

静 态 常 量	常 量 值	组件对齐方式
LEFT	0	靠左侧显示
CENTER	1	居中显示，为默认对齐方式
RIGHT	2	靠右侧显示

流布局管理器默认组件的水平间距和垂直间距均为5像素，可以通过FlowLayout类的方法setHgap(int hgap)和setVgap(int vgap)设置组件的水平间距和垂直间距。

11.3.4 GridLayout布局管理器

由GridLayout类实现的布局管理器称为网格布局管理器，它的布局方式是将容器按照用户的设置平均划分成若干网，如下图所示。



在通过构造方法GridLayout(int rows, int cols)创建网格布局管理器对象时，参数rows用来设置网格的行数，参数cols用来设置网格的列数，在设置时分为以下4种情况：

GridLayout布局管理器

（1）只设置了网格的行数，即rows大于0，cols等于0：在这种情况下，容器将先按行排列组件，当组件个数大于rows时，则再增加一列，依次类推；

（2）只设置了网格的列数，即rows等于0，cols大于0：在这种情况下，容器将先按列排列组件，当组件个数大于cols时，则再增加一行，依次类推；

（3）同时设置了网格的行数和列数，即rows大于0，cols大于0：在这种情况下，容器将先按行排列组件，当组件个数大于rows时，则再增加一列，依次类推；

GridLayout布局管理器

（4）同时设置了网格的行数和列数，但是容器中的组件个数大于网格数（rows∧cols）；在这种情况下，将再增加一列，依次类推。

网格布局管理器默认组件的水平间距和垂直间距均为0像素，可以通过GridLayout类的方法setHgap(int hgap)和setVgap(int vgap)设置组件的水平间距和垂直间距。

【例5】

11.4 常用容器

JComponent是Container的子类，因此JComponent子类创建的组件也都是容器。

容器经常用来添加组件。 JFrame是**底层容器**，

但如果将所有的组件都添加到由JFrame窗体提供的默认组件容器中，将存在如下两个问题：

（1）一个界面中的所有组件只能采用一种布局方式，这样很难得到一个美观的界面；
（2）有些布局方式只能管理有限个组件，例如JFrame窗体默认的BorderLayout布局管理器，最多只能管理5个组件。

11.4.1 中间容器—面板

通过使用面板，可以实现对所有组件进行分层管理，即对不同关系的组件采用不同的布局管理方式，使组件的布局更合理，使软件界面更美观。这些面板被习惯地称做**中间容器**；

中间容器必须被添加到底层容器中才能发挥作用。

常用中间面板

1. JPanel 面板（掌握）

使用JPanel创建面板，再向这个面板添加组件，然后把这个面板添加到其它容器中.

JPanel面板的默认布局是FlowLayout布局。

2. 滚动窗格JScrollPane:

可以将文本区放到一个滚动窗格中。

3. 拆分窗格JSplitPane:窗格有两种类型水平拆分和垂直拆分

4. JLayeredPane分层窗格:分层窗格使用

JScrollPane面板

JScrollPane类实现了一个带有滚动条的面板，用来为某些组件添加滚动条，例如在学习JList和JTextArea组件时均用到了该组件，JScrollPane类提供的常用方法如下表所示。

方　法	功　能
setViewportView(Component view)	设置在滚动面板中显示的组件对象
setHorizontalScrollBarPolicy(int policy)	设置水平滚动条的显示策略
setVerticalScrollBarPolicy(int policy)	设置垂直滚动条的显示策略
setWheelScrollingEnabled(false)	设置滚动面板的滚动条是否支持鼠标的滚动轮

JScrollPane面板

在调用上表中设置滚动条显示策略方法设置滚动条的显示策略时，方法的参数可以选择JScrollPane类中设置滚动条显示策略的静态常量，如下表所示。

静 态 常 量	常 量 值	滚动条的显示策略
HORIZONTAL_SCROLLBAR_AS_NEEDED	30	设置水平滚动条为只在需要时显示，默认策略
HORIZONTAL_SCROLLBAR_NEVER	31	设置水平滚动条为永远不显示
HORIZONTAL_SCROLLBAR_ALWAYS	32	设置水平滚动条为一直显示
VERTICAL_SCROLLBAR_AS_NEEDED	20	设置垂直滚动条为只在需要时显示，默认策略
VERTICAL_SCROLLBAR_NEVER	21	设置垂直滚动条为永远不显示
VERTICAL_SCROLLBAR_ALWAYS	22	设置垂直滚动条为一直显示

11.5 常 用 组 件

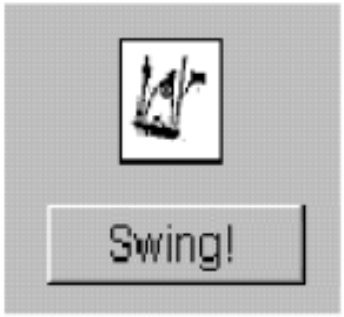
软件界面是软件和用户之间的交流平台，而组件则是绘制软件界面的基本元素，是软件和用户之间的交流要素。

例如用文本框来显示相关信息，用单选按钮、复选按钮、文本框等接受用户的输入信息，用按钮来提交用户的输入信息。

本节将对用来绘制软件界面的常用组件做详细的介绍，并针对每个组件给出一个典型例子，以方便读者学习和参考。

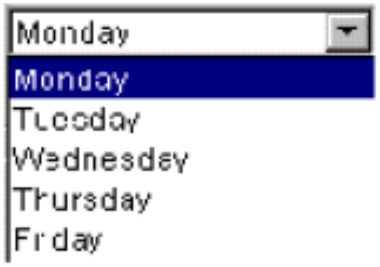
Swing用户组件

- Swing组件主要包括：
  - 文本处理、按钮、标签、列表、pane、组合框、滚动条、滚动pane、菜单、表格、树等
  - 其中一些组件的图示：

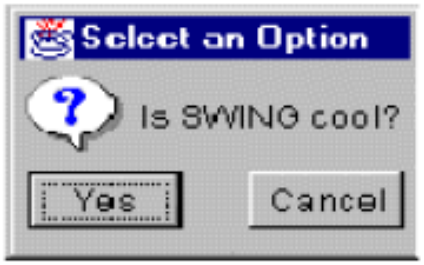


JApplet

JButton



JComboBox

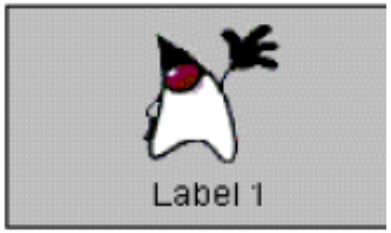


JOptionPane

Swing用户组件

- Swing组件的图示(续)：

JList



JLabel

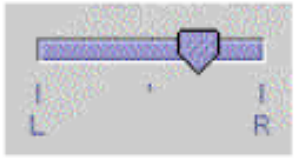
JScrollPane

JTable

JTooltip

JTree

JScrollBar



JSlider

常用组件

标签：由JComponent的子类JLabel类用来创建标签。

文本组件：

文本框：由JComponent的子类JTextField创建文本框。

文本区：由JComponent的子类JTexArea创建文本区。

密码框：由JComponent的子类JPasswordField创建密码框。

按钮类组件：

按钮：由JComponent的子类JButton类用来创建按钮。

复选按钮：由JComponent的子类JCheckBox类用来创建复选按钮。

单选按钮：由JComponent的子类JRadioButton类用来创建单选按钮

选择类组件：

选择框：由JComponent的子类JCheckBox类用来创建选择框。

下拉列表：由JComponent的子类JComboBox类用来创建下拉列表。

11.5.1 JLabel（标签）组件

- 标签组件：JLabel组件用来显示文本和图像，可以只显示其中的一者，也可以二者同时显示

显示静态文本，用于显示提示信息

- setText(String str)：设置文本内容
- getText()：获取文本内容
- setFont(Font font)方法设置标签文本的字体及大小如果需要在标签中显示图片

java.lang.Object
└ java.awt.Component
└ java.awt.Container
└ javax.swing.JComponent
└ **javax.swing.JLabel**

JLabel（标签）组件

JLabel类，通过setHorizontalAlignment(int alignment)方法设置文本的显示位置，该方法的参数可以从JLabel类提供的静态常量中选择，可选的静态常量如下表所示。

静 态 常 量	常 量 值	标签内容显示位置
LEFT	2	靠左侧显示
CENTER	0	居中显示
RIGHT	4	靠右侧显示

JLabel（标签）组件

JLabel通过setIcon(Icon icon)方法显示图片。

如果想在标签中既显示文本，又显示图片：

setHorizontalTextPosition(int textPosition)方法设置文字相对图片在水平方向的显示位置。

setVerticalTextPosition(int textPosition)方法设置文字相对图片在垂直方向的显示位置

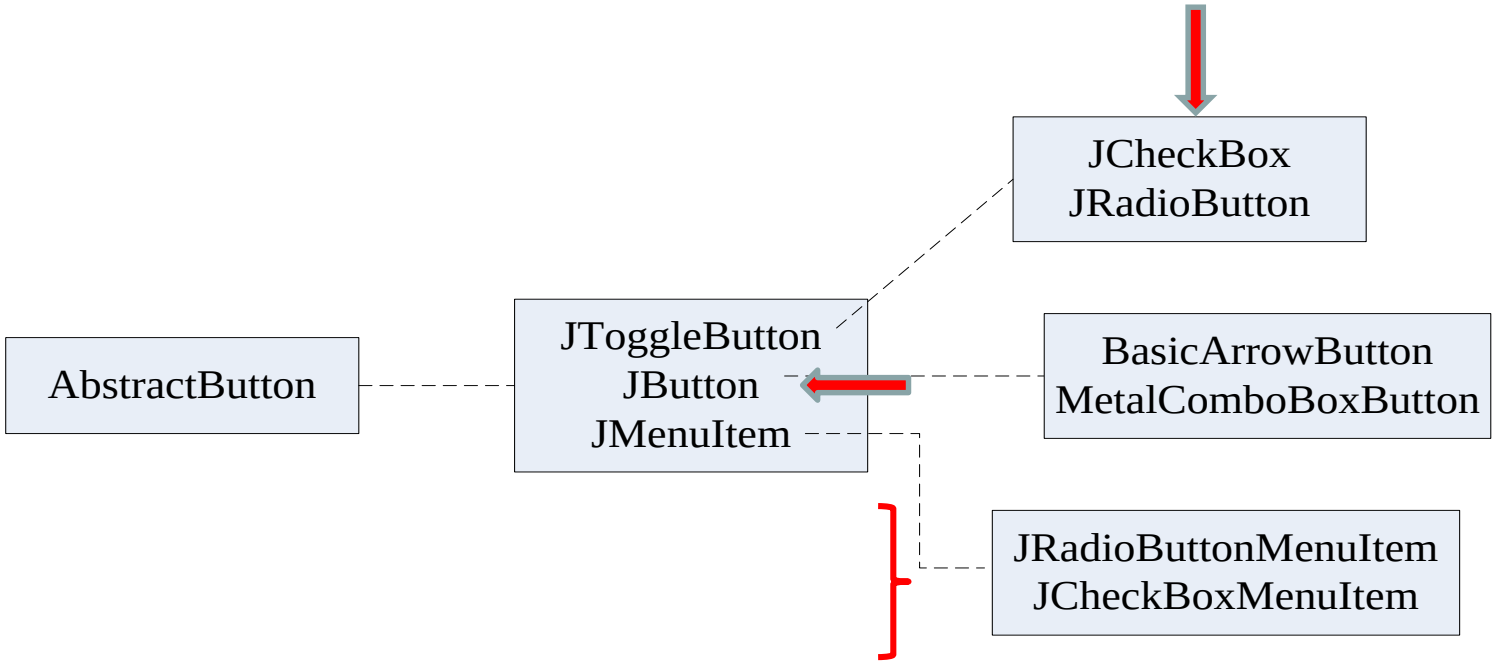
入口参数可以从JLabel类提供的静态常量中选择，可选的静态常量如下表所示。

静 态 常 量	常 量 值	标签内容显示位置
TOP	1	文字显示在图片的上方
CENTER	0	文字与图片在垂直方向重叠显示
BOTTOM	3	文字显示在图片的下方

【例8】

按钮组件

所有的按钮类都继承自AbstractButton类



常用事件ActionEvent

JButton（按钮）组件

JButton组件是最简单的按钮组件，只是在按下和释放两个状态之间进行切换，可以通过捕获按下并释放的动作执行一些操作，从而完成和用户的交互。

JButton类提供了一系列用来设置按钮的方法，例如通过setText(String text)方法设置按钮的标签文本，通过下面的代码就可以创建一个最简单按钮：

```
final JButton button = new JButton();
button.setBounds(10, 10, 70, 23);
button.setText("确 定");
getContentPane().add(button);
```

JButton（按钮）组件

更多的是为按钮设置图片：

setIcon(Icon defaultIcon)用来设置按钮在默认状态下显示的图片；

setRolloverIcon(Icon rolloverIcon)用来设置当光标移动到按钮上方时显示的图片；

setPressedIcon(Icon pressedIcon)用来设置当按钮被按下时显示的图片。

JRadioButton（单选按钮）组件

JRadioButton组件实现一个单选按钮，用户可以很方便地查看单选按钮的状态。JRadioButton类可以单独使用，也可以与ButtonGroup类联合使用，当单独使用时，该单选按钮可以被选定和取消选定，当与ButtonGroup类联合使用时，则组成了一个单选按钮组，此时用户只能选定按钮组中的一个单选按钮，取消选定的操作将由ButtonGroup类自动完成。

ButtonGroup类用来创建一个按钮组，按钮组的作用是负责维护该组按钮的“开启”状态，在按钮组中只能有一个按钮处于“开启”状态。假设在按钮组中有且仅有A按钮处于开启状态，在“开启”其他按钮时，按钮组将自动关闭A按钮的“开启”状态。

JRadioButton（单选按钮）组件

按钮组经常用来维护由JRadio Button、JRadioButtonMenuItem或JToggleButton类型的按钮组成的按钮组。ButtonGroup类提供的常用方法如下表所示：

方　法	功　能
add(AbstractButton b)	添加按钮到按钮组中
remove(AbstractButton b)	从按钮组中移除按钮
getButtonCount()	返回按钮组中包含按钮的个数，返回值为int型
getElements()	返回一个Enumeration类型的对象，通过该对象可以遍历按钮组中包含的所有按钮对象

JRadioButto（单选按钮）组件

JRadioButton类提供了一系列用来设置单选按钮的方法，例如通过setText(String text)方法设置单选按钮的标签文本，通过setSelected(boolean b)方法设置单选按钮的状态，默认情况下未被选中，当设为true时表示单选按钮被选中。

【例12】

JCheckBox（复选框）组件

JCheckBox组件实现一个复选框，该复选框可以被选定和取消选定，并且可以同时选定多个。用户可以很方便地查看复选框的状态。

JCheckBox类提供了一系列用来设置复选框的方法：

- setText(String text)方法设置复选框的标签文本
- setSelected(boolean b)方法设置复选框的状态，默认情况下未被选中，当设为true时表示复选框被选中。

【例13】

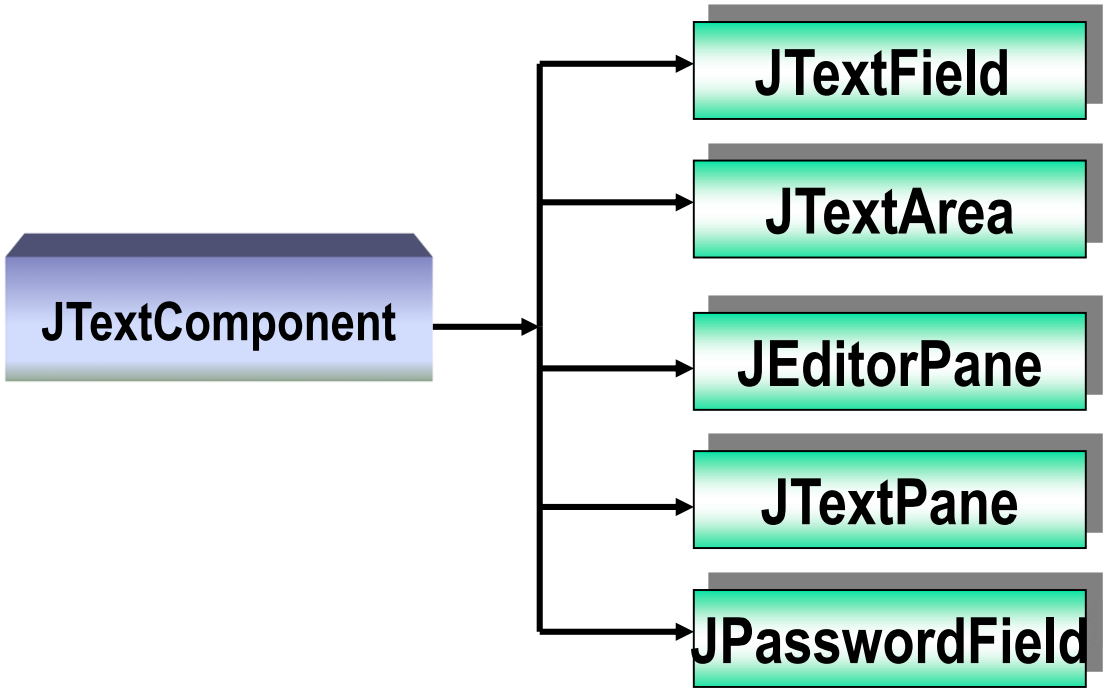
JCheckBox和JRadioButton示例

```
import java.awt.*;
import javax.swing.*;
class Hobby extends JPanel {
  JCheckBox c1 = new JCheckBox("阅读",false);
  JCheckBox c2 = new JCheckBox("音乐",false);
  JCheckBox c3 = new JCheckBox("绘画",false);
  JRadioButton rad1 = new JRadioButton("大专");
  JRadioButton rad2 = new JRadioButton("本科");
  JRadioButton rad
  JLabel j1 = new
  JLabel j2 = new
  JButton exitbtn
  public Hobby( )
     setLayout(new
     add(j1); add(
     add(j2); add(
   }
}
```

```
public class Hobbytest extends JFrame {
   Hobbytest() {
      super();
      getContentPane().add(new Hobby());
      setSize(300,200);
   }
  public static void m
    new Hobbytest();
}
```

文本组件

- JTextComponent 为所有 Swing 文本组件的根类

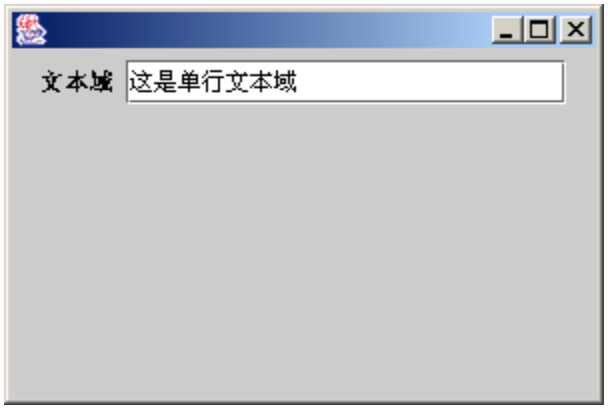


常用事件
KeyEvent
FocusEvent
ActionEvent

JTextField（文本框）组件

JTextField组件实现一个文本框，用来接受用户输入的单行文本信息。

```
…
Container con = getContentPane();
con.setLayout(new FlowLayout());
JLabel jl = new JLabel(“文本域”);
con.add(jl);
JTextField tf = new JTextField(20);
con.add(tf);
…
```



JTextField（文本框）组件

JTextField

- 构造函数包括：
  - **JTextField()**
  - **JTextField(Document doc, String text, int columns)**
  - **JTextField(int columns)**
  - **JTextField(String text)**
  - **JTextField(String text, int columns)**

```
JTextField textField= new JTextField ("请输入姓名");
textField.setText("请输入姓名");
```

JTextField（文本框）组件

JTextField类提供的常用方法如下表所示。

方　　法	功　　能
getPreferredSize()	获得文本框的首选大小，返回值为Dimensions类型的对象
scrollRectToVisible(Rectangle r)	向左或向右滚动文本框中的内容
setColumns(int columns)	设置文本框最多可显示内容的列数
setFont(Font f)	设置文本框的字体
setScrollOffset(int scrollOffset)	设置文本框的滚动偏移量（以像素为单位）
setHorizontalAlignment(int alignment)	设置文本框内容的水平对齐方式

【例17】

JPasswordField（密码框）组件

JPasswordField组件实现一个密码框，用来接受用户输入的单行文本信息，在密码框中并不显示用户输入的真实信息，而是通过显示一个指定的回显字符作为占位符。

新创建密码框的默认回显字符为“*”，可以通过setEchoChar(char c)方法修改回显字符，例如将回显字符修改为“#”。这两中效果如下图所示：

JPasswordField（密码框）组件

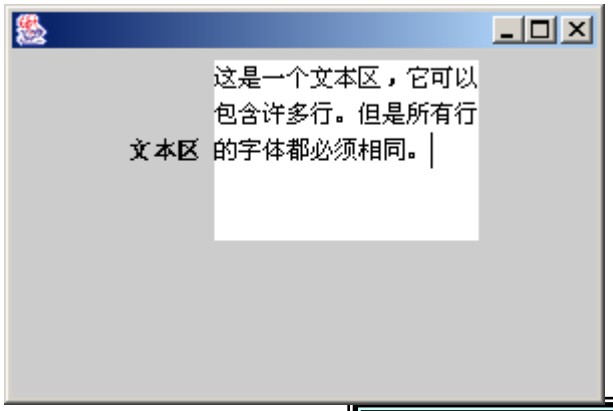
JPasswordField类提供的常用方法如下表所示。

方　　法	功　　能
setEchoChar(char c)	设置回显字符为指定字符
getEchoChar()	获得回显字符，返回值为char型
echoCharIsSet()	查看是否已经设置了回显字符，如果设置了则返回true，否则返回false
getPassword()	获得用户输入的文本信息，返回值为char型数组

【例19】

JTextArea 组件

JTextArea用于接受来自用户的多行文本，可用于实现可滚动界面




```
…
JLabel jl = new JLabel(“文本区”);
con.add(jl);
JTextArea ta = new JTextArea(5,10);
con.add(ta);
…
```

JTextArea 组件

- 构造函数：
  - **JTextArea()**
  - **JTextArea(int rows, int cols)**
  - **JTextArea(String text)**
  - **JTextArea(String text, int rows, int cols)**
  - **JTextArea(Document doc)**
  - **JTextArea(Document doc, String text, int rows, int cols)**

- JList 不支持滚动。要启用滚动，可使用下列代码:

```
JScrollPane myScrollPane=new JScrollPane();
myScrollPane.getViewport().setView(dataList);
```

JTextArea（文本域）组件

在创建文本域时，可以通过setLineWrap(boolean wrap)方法设置文本是否自动换行，默认为false，即不自动换行，如果改为自动换行，需要设置wrap参数为true。

JTextArea类的常用方法如下表所示

方　法	功　能
append(String str)	将指定文本追加到文档结尾
insert(String str, int pos)	将指定文本插入指定位置
replaceRange(String str, int start, int end)	用指定新文本替换从指示的起始位置到结尾位置的文本
getColumnWidth()	获取列的宽度
getColumns()	返回文本域中的列数
getLineCount()	确定文本区中所包含的行数
getPreferredSize()	返回文本域的首选大小
getRows()	返回文本域中的行数
setLineWrap(boolean wrap)	设置文本区是否自动换行，默认为false，即不自动换行

【例7】

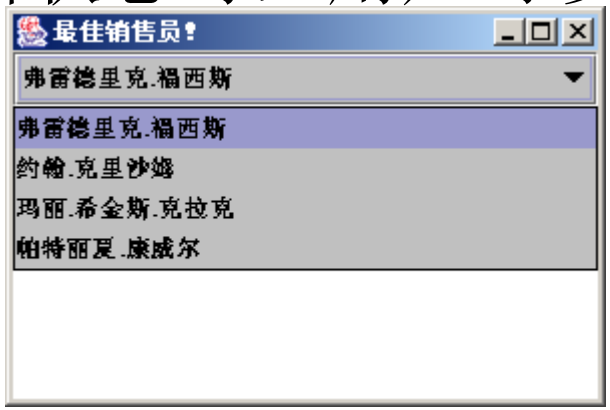
选择性输入组件

- 为了简化表单填写过程，通常为用户提供多种可供选择的选项，而无需用户写出他们的响应。
- 常用于选择性输入的组件有：
  - **复选框JCheckBox（在Button已讲）**
  - **单选框JRadioButton（在Button已讲）**
  - **列表框JList**
  - **组合框JComboBox**

常用事件ItemEvent

JComboBox（选择框）组件

- JComboBox组件实现一个选择框，用户可以从下拉列表中选择相应的值，该选择框还可以设置为可编辑的，当设置为可编辑状态时，用户可以在选择框中输入相应的值。




```
…
String names[] = {"弗雷德里克.福西斯", "约翰.克里沙姆
", "玛丽.希金斯.克拉克","帕特丽夏.康威尔"};
JComboBox authors = new JComboBox(names);
…
```

JComboBox（选择框）组件

JComboBox的构造函数：

- **public JComboBox()**：使用缺省数据模型创建对象
- **public JComboBox(ComboBoxModel asModel)**：使用现有 **ComboBoxModel** 中的项目的组合框
- **public JComboBox(Object [] items)**：包含指定数组元素的组合框

JComboBox（选择框）组件

例如创建一个包含选项“身份证”、“士兵证”和“驾驶证”的选择框，具体代码如下：

```
String[] idCards = { "身份证", "士兵证", "驾驶证" };
JComboBox idCardComboBox = new JComboBox(idCards);
```

可以通过方法addItem(Object item)和insertItemAt(Object item, int index)向选择框中添加选项，例如：

```
JComboBox idCardComboBox = new JComboBox();
comboBox.addItem("士兵证");
comboBox.addItem("驾驶证");
comboBox.insertItemAt("身份证", 0);
```

也可以通过setModel(ComboBoxModel aModel)方法初始化该选择框包含的选项，例如：

```
String[] idCards = { "身份证", "士兵证", "驾驶证" };
JComboBox idCardComboBox = new JComboBox();
comboBox.setModel(new DefaultComboBoxModel(idCards));
```

JComboBox（选择框）组件

- 选择框
  - 属性：
    - selectedIndex：int值，表示选定项的序号
    - selectedItem：Object类型，表示选定项

JComboBox（选择框）组件

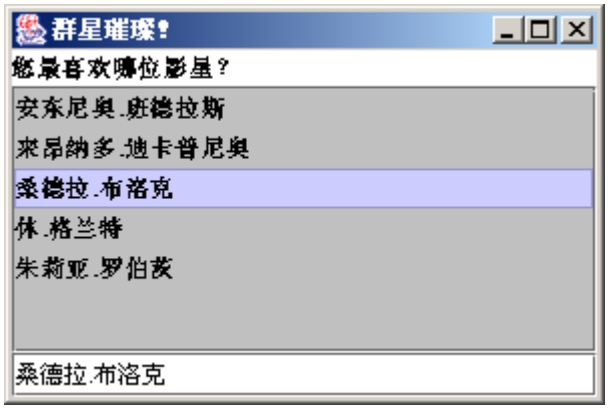
JComboBox类提供了一系列用来设置选择框的方法，例如通过方法setSelectedItem()或setSelectedIndex()设置选择框的默认选项；通过方法setEditable()设置选择框是否可编辑。JComboBox类提供的常用方法如下表所示。

方　　法	功　　能
addItem(Object item)	添加选项到选项列表的尾部
insertItemAt(Object item, int index)	添加选项到选项列表的指定索引位置，索引从0开始
removeItem(Object item)	从选项列表中移除指定的选项
removeItemAt(int index)	从选项列表中移除指定索引位置的选项
removeAllItems()	移除选项列表中的所有选项
setSelectedItem(Object item)	设置指定选项为选择框的默认选项
setSelectedIndex(int index)	设置指定索引位置的选项为选择框的默认选项
setMaximumRowCount(int count)	设置选择框弹出时显示选项的最多行数，默认为8行
setEditable(boolean isEdit)	设置选择框是否可编辑，当设置为true时表示可编辑，默认为不可编辑（false）

【例15】

列表框

JList组件实现一个列表框，列表框与选择框的主要区别是列表框可以多选，而选择框只能单选。




```
...
String stars[] = {"安东尼奥.班德拉斯","来昂纳多.迪卡普尼奥", "桑德拉.布洛克","休.格兰特","朱莉亚.罗伯茨"};
JList moviestars = new JList(stars);
...
```

列表框

JList构造函数：

- public JList()：
- public JList(ListModel dataModel) :构造一个列表，用它显示指定模型中的元素
- public JList (Object [] listData) :构造一个列表以显示指定数组listData的元素

JList 不支持滚动。要启用滚动，可使用下列代码:

```
JScrollPane myScrollPane=new JScrollPane();
        myScrollPane.getViewport().setView(dataList);
```

JList（列表框）组件

- 列表框常用属性：
  - selectedIndex
  - selectedIndices：int数组，表示选定的多项的序号
  - selectedValue：选定的第一个选定值
  - selectedValues
  - visibleRowCount：列表不用滚动可看到的行数

JList（列表框）组件

JList类提供了一系列用来设置列表框的方法：

方　法	功　能
setSelectedIndex(int index)	选中指定索引的一个选项
setSelectedIndices(int[] indices)	选中指定索引的一组选项
setSelectionBackground(Color selectionBackground)	设置被选项的背景颜色
setSelectionForeground(Color selectionForeground)	设置被选项的字体颜色
getSelectedIndices()	以int[]形式获得被选中的所有选项的索引值
getSelectedValues()	以Object[]形式获得被选中的所有选项的内容
clearSelection()	取消所有被选中的项
isSelectionEmpty()	查看是否有被选中的项，如果有则返回true
isSelectedIndex(int index)	查看指定项是否已经被选中
ensureIndexIsVisible(int index)	使指定项在选择窗口中可见
setFixedCellHeight(int height)	设置选择窗口中每个选项的高度
setVisibleRowCount(int visibleRowCount)	设置在选择窗口中最多可见选项的个数
getPreferredScrollableViewportSize()	获得使指定个数的选项可见需要的窗口高度
setSelectionMode(int selectionMode)	设置列表框的选择模式，即单选还是多选

【例16】

JList（列表框）组件

由JList组件实现的列表框有三种选取模式，可以通过JList类的setSelectionMode(int selectionMode)方法设置具体的选取模式，该方法的参数可以从ListSelectionModel类中的静态常量中选择。这三种选取模式包括一种单选模式和两种多选模式，具体信息如下表所示。

静 态 常 量	常 量 值	标签内容显示位置
SINGLE_SELECTION	0	只允许选取一个
SINGLE_INTERVAL_SELECTION	1	只允许连续选取多个
MULTIPLE_INTERVAL_SELECTION	2	既允许连续选取，又允许间隔选取

11.6 处理事件

学习组件除了要熟悉组件的属性和功能外，一个更重要的方面是学习怎样处理组件上发生的界面事件。当用户在文本框中键入文本后按回车键、单击按钮、在一个下拉式列表中选择一个条目等操作时，都发生界面事件。

程序有时需对发生的事件作出反应，来实现特定的任务，例如，用户单击一个名字叫“确定”或名字叫“取消”的按钮，程序可能需要作出不同的处理。

11.6.1 事件处理模式

1．**事件源** ：能够产生事件的对象都可以成为事件源 ．

2．**监视器** ：事件源通过调用相应的方法将**某个对象**注册为自己的监视器。对于文本框，这个方法是：

addActionListener(监视器);

事件源注册监视器之后，相应的操作就会导致相应的事件的发生，并通知监视器，监视器就会作出相应的处理。

11.6.1 事件处理模式

3. 处理事件的接口:

监视器负责处理事件源发生的事件。监视器是一个对象，为了处理事件源发生的事件，监视器这个对象会自动调用一个方法来处理事件。

Java规定：为了让监视器这个对象能对事件源发生的事件进行处理，创建该监视器对象的类必须声明实现相应的接口，那么当事件源发生事件时，监视器就自动调用被类重写的某个接口方法(如图10.7)。

11.6.1 事件处理模式效果图

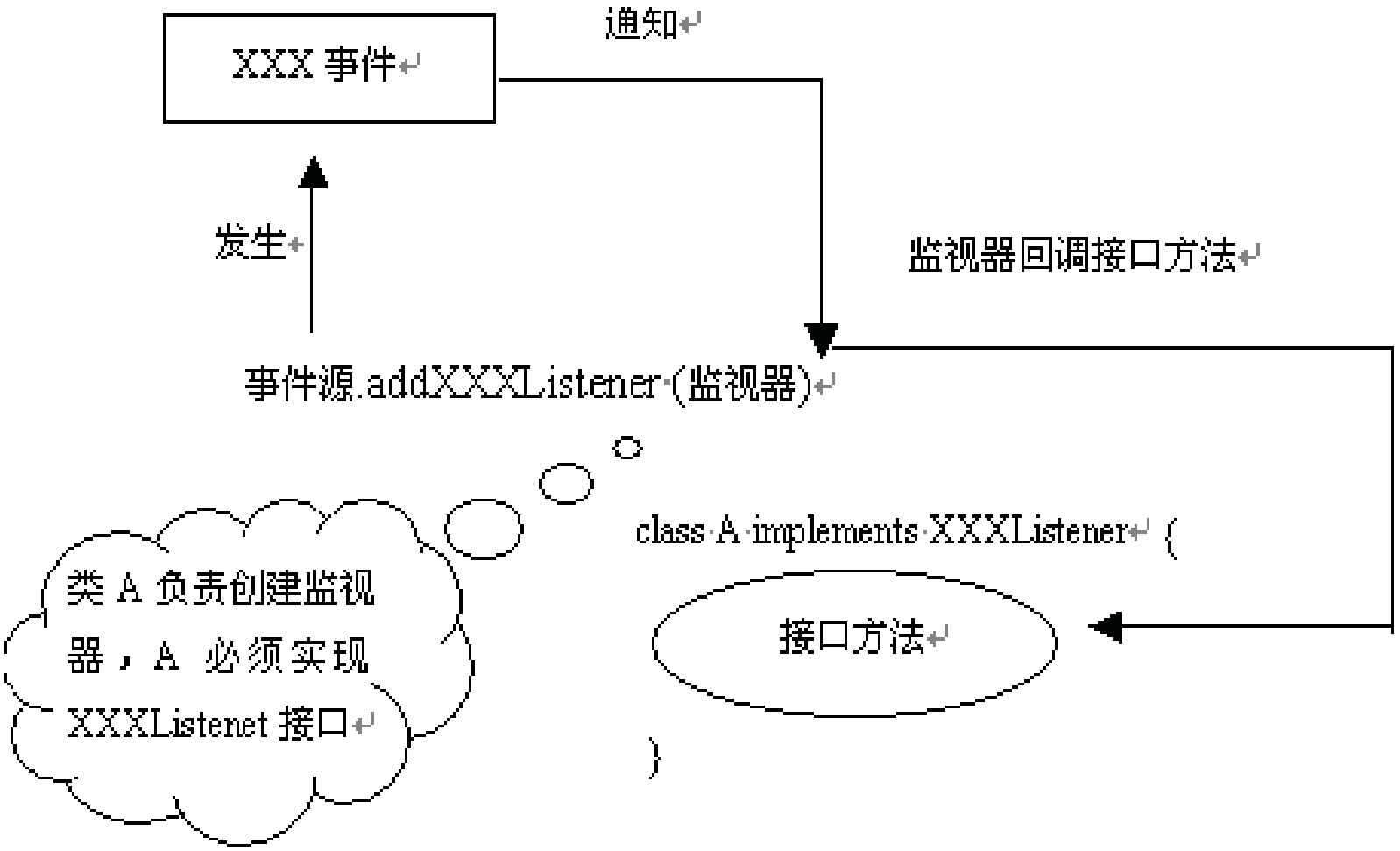


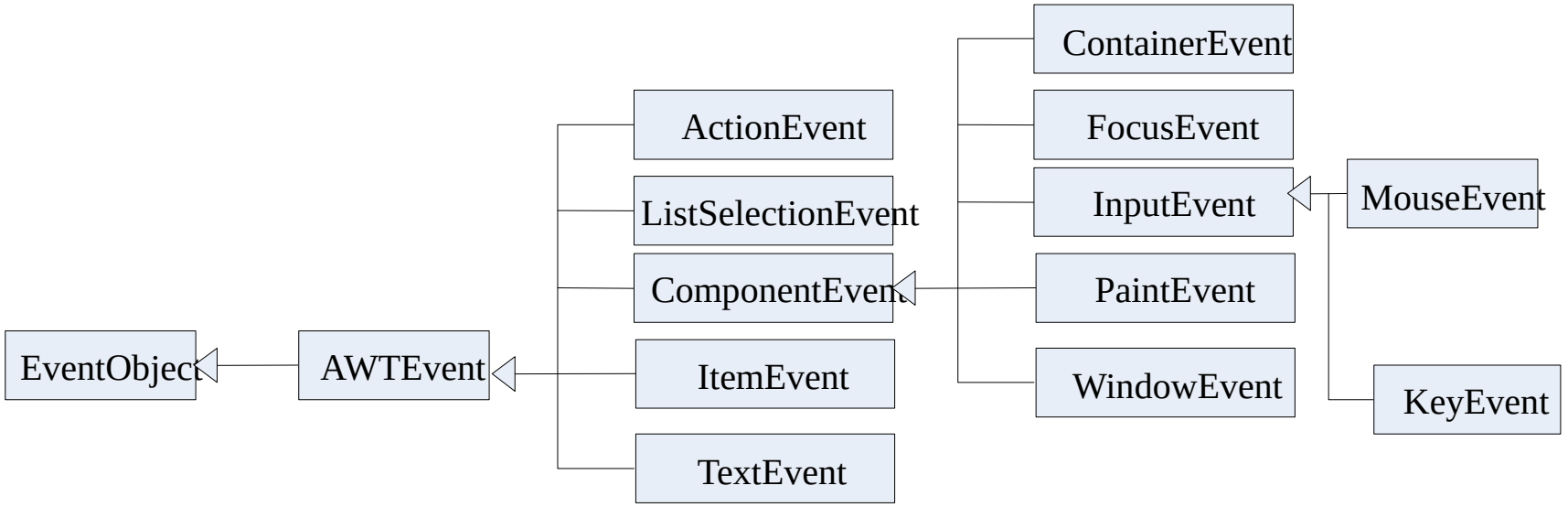
图 10.7 处理事件示意图

11.6.1事件处理

- 事件驱动程序设计：
  - 事件驱动：
    - 事件注册、监听和处理
      - 一个事件源可以触发多种事件，如果它注册了某种事件的监听器，那么这种事件就会被接收和处理
      - 事件源本身并不处理事件，而是委托给相应的事件监听器来处理，因此这种模式称为***委托模式***
      - 事件、事件监听器和监听器方法

11.6.1事件处理

- 事件驱动程序设计：
  - 事件驱动：
    - 一个事件是事件类的实例，事件类的根类是java.util.EventObject，常用事件层次关系：



11.6.1事件处理

- 事件驱动程序设计：
  - 事件驱动：
    - 用户行为、源对象和事件类型

用户行为	源对象	事件类型
点击按钮	Jbutton	ActionEvent
改变文本	JTextComponent	TextEvent
在文本域按下回车	JTextField	ActionEvent
选定一个新项	JComboBox	ItemEvent,ActionEvent
选定多项	Jlist	ListSelectionEvent
点击复选按钮	JCheckBox	ItemEvent,ActionEvent
点击单选按钮	JRadioButton	ItemEvent,ActionEvent
选定菜单项	JMenuItem	ActionEvent
移动滚动条	JScrollBar	AdjustmentEvent
窗口打开、关闭、图标化、还原、关闭	Window	WindowEvent
在容器中添加、删除组件	Container	ContainerEvent
组件移动、改变大小、隐藏显式	Component	ComponentEvent
组件获得或者失去焦点	Component	FocusEvent
释放或按下键	Component	KeyEvent
移动鼠标	Component	MouseEvent

事件类型	监听器接口	监听器方法（处理器）
ActionEvent	ActionListener	actionPerformed(ActionEvent e)
ItemEvent	ItemListener	itemStateChanged(ItemEvent e)
WindowEvent	WindowListener	windowClosing(WindowEvent e)
		windowOpened(WindowEvent e)
		windowIconified(WindowEvent e)
		windowDeiconified(WindowEvent e)
		windowClosed(WindowEvent e)
		windowActivated(WindowEvent e)
		windowDeactivated(WindowEvent e)
ContainerEvent	ContainerListener	componentAdded(ContainerEvent e)
		componentRemoved(ContainerEvent e)
ComponentEvent	ComponentListener	componentMoved(ComponentEvent e)
		componentHidden(ComponentEvent e)
		componentResized(ComponentEvent e)
		componentShown(ComponentEvent e)
FocusEvent	FocusListener	focusGained(FocusEvent e)
		focusLost(FocusEvent e)
TextEvent	TextListener	textValueChanged(TextEvent e)
KeyEvent	KeyListener	keyPressed(KeyEvent e)
		keyReleased(KeyEvent e)
		keyTyped(KeyEvent e)
MouseEvent	MouseListener	mousePressed(MouseEvent e)
		mouseReleased(MouseEvent e)
		mouseEntered(MouseEvent e)
		mouseExited(MouseEvent e)
		mouseClicked(MouseEvent e)
	MouseMotionListener	mouseDragged(MouseEvent e)
		mouseMoved(MouseEvent e)
AdjustmentEvent	AdjustmentListener	adjustmentValueChanged(AdjustmentEvent e)

11.6.2 动作事件处理

动作事件由ActionEvent类定义，最常用的是当单击按钮后将产生动作事件，可以通过实现ActionListener接口处理相应的动作事件。

ActionListener接口只有一个抽象方法，将在动作发生后被触发，例如单击按钮之后，ActionListener接口的具体定义如下：

```
public interface ActionListener extends EventListener {
    public void actionPerformed(ActionEvent e);
}
```

动作事件处理

ActionEvent类中有两个比较常用的方法：

（1）getSource()：用来获得触发此次事件的组件对象，返回值类型为Object；

（2）getActionCommand()：用来获得与当前动作相关的命令字符串，返回值类型为String。

三种常用的事件侦听：

(1)匿名类方式

(2)内部类方式

(3)窗口类方式

11.6.3 焦点事件处理

焦点事件由FocusEvent类捕获，所有的组件都能产生焦点事件，可以通过实现FocusListener接口处理相应的动作事件。

FocusListener接口有两个抽象方法，分别在组件获得或失去焦点时被触发，FocusListener接口的具体定义如下：

```
public interface FocusListener extends EventListener {
    public void focusGained(FocusEvent e);          // 当组件获得焦点时将触发该方法
    public void focusLost(FocusEvent e);            // 当组件失去焦点时将触发该方法
}
```

FocusEvent类中比较常用的方法是getSource()，用来获得触发此次事件的组件对象，返回值类型为Object。

【例24】

11.6.5 键盘事件处理

键盘事件由KeyEvent类捕获，最常用的是当向文本框输入内容时将发生键盘事件，可以通过实现KeyListener接口处理相应的键盘事件。

KeyListener接口有3个抽象方法，分别在发生击键事件、键被按下和释放时被触发，Focus Listener接口的具体定义如下：

```
public interface KeyListener extends EventListener {
    public void keyTyped(KeyEvent e);
    public void keyPressed(KeyEvent e);
    public void keyReleased(KeyEvent e);
}
```

键盘事件处理

KeyEvent类中比较常用的方法如下表所示：

方　法	功　能
getSource()	用来获得触发此次事件的组件对象，返回值为Object类型
getKeyChar()	用来获得与此事件中的键相关联的字符
getKeyCode()	用来获得与此事件中的键相关联的整数keyCode
getKeyText(int keyCode)	用来获得描述keyCode的标签，例如“A”、“F1”、“HOME”等
isActionKey()	用来查看此事件中的键是否为“动作”键
isControlDown()	用来查看“Ctrl”键在此次事件中是否被按下
isAltDown()	用来查看“Alt”键在此次事件中是否被按下
isShiftDown()	用来查看“Shift”键在此次事件中是否被按下

在KeyEvent类中以“VK_”开头的静态常量代表各个按键的keyCode，可以通过这些静态常量判断事件中的按键，以及获得按键的标签。【例26】

11.6.4 鼠标事件处理

鼠标事件由MouseEvent类捕获，所有的组件都能产生鼠标事件，可以通过实现MouseListener接口处理相应的鼠标事件。

MouseListener接口有5个抽象方法，分别在光标移入（出）组件时、鼠标按键被按下（释放）时和发生单击事件时被触发。

所谓单击事件，就是按键被按下并释放。

需要注意的是，如果按键是在移出组件之后才被释放，则不会触发单击事件。

鼠标事件处理

MouseListener接口的具体定义如下：

```
public interface MouseListener extends EventListener {
  // 光标移入组件时被触发
   public void mouseEntered(MouseEvent e);
  // 鼠标按键被按下时触发
   public void mousePressed(MouseEvent e);
  // 鼠标按键被释放时触发
   public void mouseReleased(MouseEvent e);
  // 发生单击事件时被触发
   public void mouseClicked(MouseEvent e);
  // 光标移出组件时被触发
   public void mouseExited(MouseEvent e);}
```

鼠标事件处理

MouseEvent类中比较常用的方法如下表所示。

方　法	功　能
getSource()	用来获得触发此次事件的组件对象，返回值为Object类型
getButton()	用来获得代表触发此次按下、释放或单击事件的按键的int型值
getClickCount()	用来获得单击按键的次数

可以通过下表中的静态常量，判断通过getButton()方法得到的值代表哪个键。

静 态 常 量	常 量 值	代 表 的 键
BUTTON1	1	代表鼠标左键
BUTTON2	2	代表鼠标滚轮
BUTTON3	3	代表鼠标右键

【例25】

窗口事件

JFrame及子类创建的窗口可以调用

setDefaultCloseOperation(int operation);

方法设置窗口的关闭方式,Operation**取**JFrame**的**static**常量：**

DO_NOTHING_ON_CLOSE（什么也不做）

HIDE_ON_CLOSE（隐藏当前窗口）

DISPOSE_ON_CLOSE（隐藏当前窗口，并释放窗体占有的其它资源）

EXIT_ON_CLOSE（结束窗口所在的应用程序）

但是setDefaultCloseOperation方式可能不能满足程序的需要，比如，用户单击窗口上的关闭图标时，可能程序需要提示用户是否需要保存窗口中的有关数据到磁盘等 .

这些要通过处理事件来满足程序的要求。需要注意的是，如果准备处理窗口事件，必须事先保证窗口的默认关闭方式为DO_NOTHING_ON_CLOSE（什么也不做）。

1.WindowEvent事件源

JFrame是Window的子类,凡是Window子类创建的对象都可以发生WindowEvent事件，即窗口事件。

2.WindowListener接口

(1) public void windowActivated(WindowEvent e) 当窗口从非激活状态到激活时，窗口的监视器调用该方法。

（2） public void windowDeactivated(WindowEvent e) 当窗口激活状态到非激活状态时，窗口的监视器调用该方法。

（3） public void windowClosing(WindowEvent e) 当窗口正在被关闭时，窗口的监视器调用该方法。

（4） public void windowClosed(WindowEvent e) 当窗口关闭后，窗口的监视器调用该方法。

（5） public void windowIconified(WindowEvent e) 当窗口图标化时，窗口的监视器调用该方法。

（6） public void windowDeiconified(WindowEvent e) 当窗口撤消图标化时，窗口的监视器调用该方法。

（7） public void windowOpened(WindowEvent e) 当窗口打开时，窗口的监视器调用该方法。

3. WindowAdapter适配器

适配器可以代替接口来处理事件，当Java提供处理事件的接口中多于一个方法时，Java相应地就提供一个适配器类，比如WindowAdapter类。**适配器已经实现了相应的接口**，例如WindowAdapter类实现了WindowListener接口。因此，可以使用WindowAdapte的子类创建的对象做监视器，在子类中重写所需要的接口方法即可。

使用适配器做监视器，只处理窗口关闭事件，因此只需重写windowColsing方法即可。

事件总结

1．授权模式

Java的事件处理是基于授权模式，即事件源调用调用方法将某个对象注册为自己的监视器。

2．接口回调

addXXXListener(XXXListener listener)方法中的参数是一个接口，listener可以引用任何实现了该接口的类所创建的对象，当事件源发生事件时，接口listener立刻回调被类实现的接口中的某个方法。

3.方法绑定

当事件源触发事件发生后，监视器准确知道去调用哪个方法。

4.保持松耦合

当事件源触发事件发生后，系统知道某个方法会被执行，但无须关心到底是哪个对象去调用了这个方法，因为任何实现接口的类的实例(做为监视器)都可以调用这个方法来处理事件。