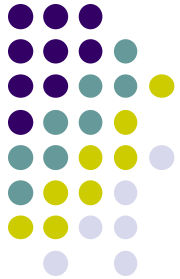
第九章 Swing图形用户界面



- 9.1 概述
- 9.2 容器组件
- 9.3 基本组件
- 9.4 布局管理器
- 9.5 事件处理模型
- 9.6 本章小结

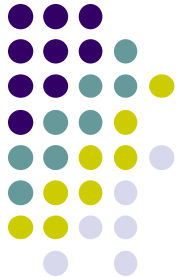
9.1 概述

- 图形用户界面(GUI)
  - Command Line → Graphics User Interface
  - 数据的图形显示形式
  - 友好的交互方式
  - 简化计算机软件的学习过程
  - GUI示例
    - jdk1.7\demo\jfc\SwingSet2
    - GUI组件: Labels, Text fields, Buttons等等

9.1 概述

- 在Java语言中，有两个包（java.awt和javax.swing）囊括了实现图形用户界面的所有基本元素，这些基本元素主要包括容器、组件、绘图工具和布局管理器等。
- java.awt是java1.1用来建立GUI的图形包， awt是抽象窗口工具包（Abstract Window Toolkit）的缩写，其中的组件常被称为AWT组件。
- javax.swing是Java2提出的AWT的改进包，主要改善了组件的显示外观，增强了组件的控制能力。

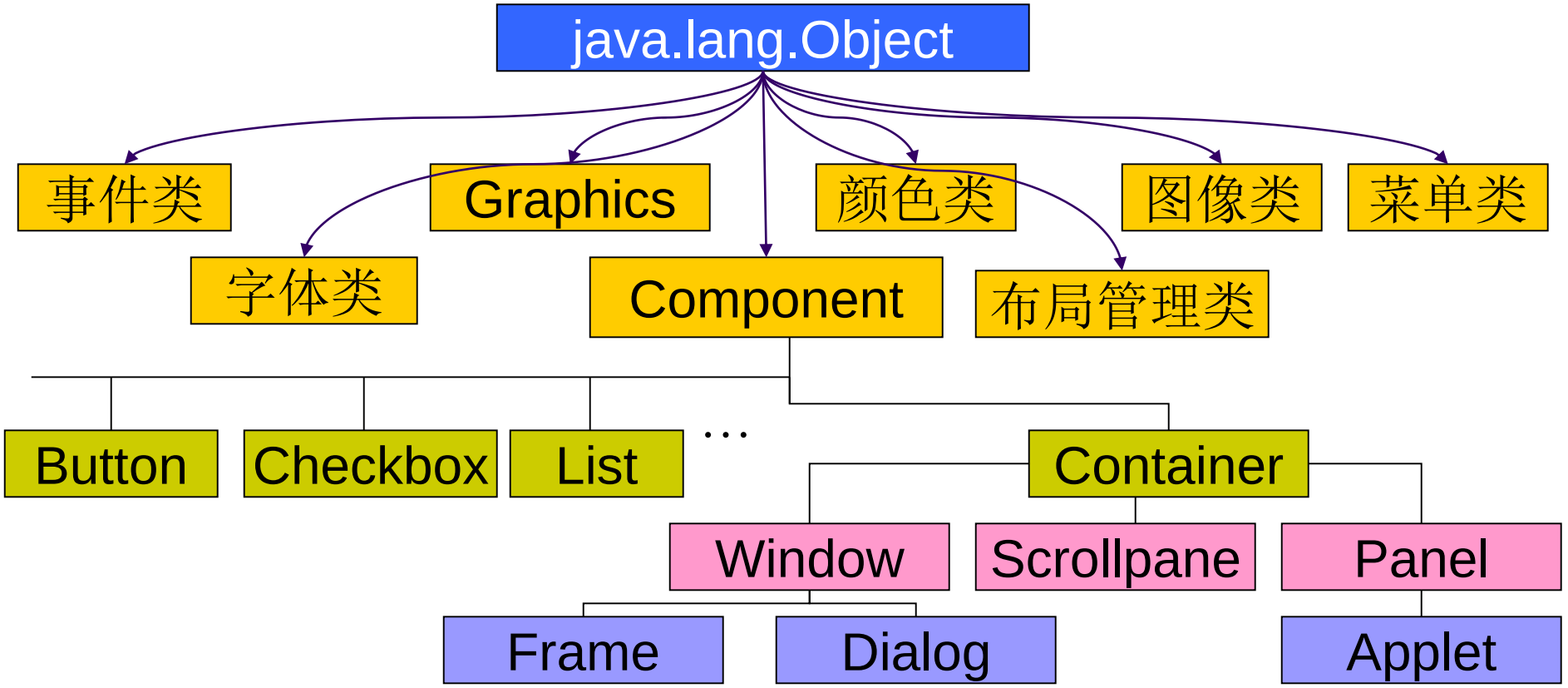
9.1 概述



- 早期版本的AWT组件
  - 在java.awt包里，包括Button、Checkbox、Scrollbar等，都是Component类的子类
  - 大部分含有native code，所以随操作系统平台的不同会显示出不同的样子，而不能进行更改，是重量级组件(heavyweight components)
  - 没有弹性、缺乏效率

9.1 概述

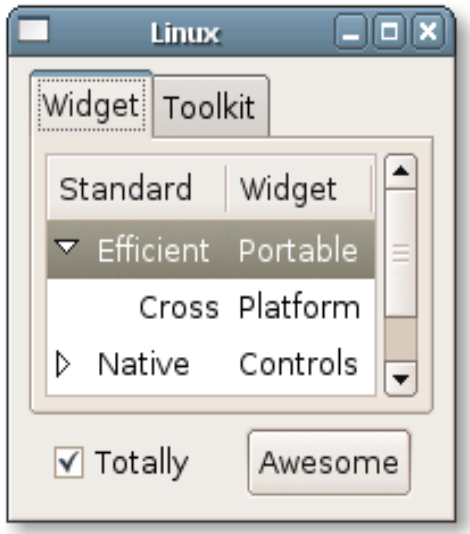
- **AWT与Swing**
  - **java.awt包中的类及相互关系可以用下图来描述：**



5

9.1 概述

- 较新的Swing组件
  - 其名称都是在原来AWT
    JCheckBox、JScrollbar
  - Java1.2推出，架构在A
    是取代，AWT仍然是Swi
  - 完全是由java语言编写的，
    由宿主平台的窗口系统所提供的代码，是
    （lightweight components）
  - 可提供更丰富的视觉感受，被越来越多地

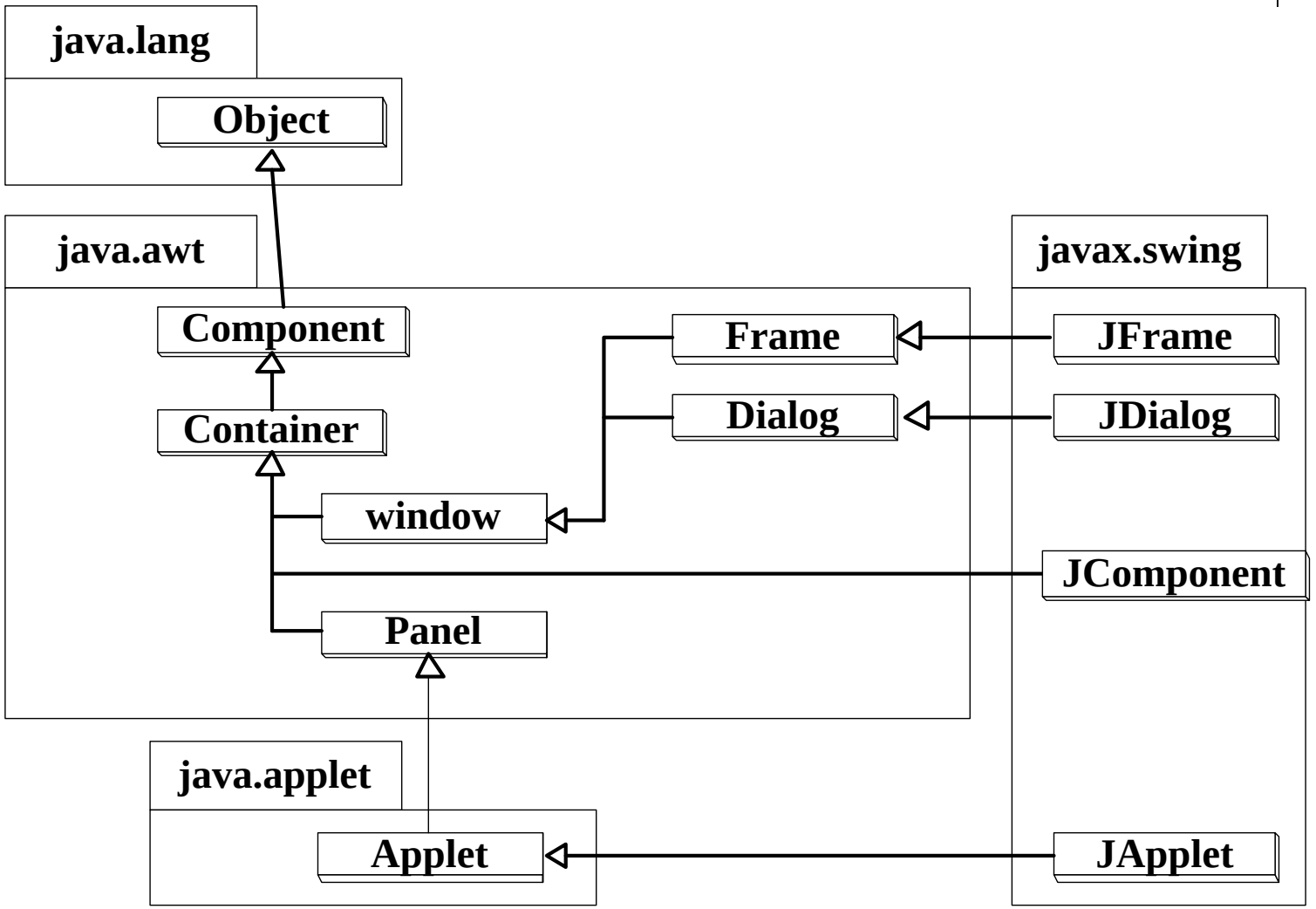


9.1 概述

- Swing组件
  - 在javax.swing包中，源于AWT（package java.awt）的组件
  - 定义了两种类型的组件：
    - 顶层容器（JFrame，JApplet，JDialog和JWindow）
    - 轻量级组件（JComponent）
  - Swing组件都是AWT的Container类的直接子类和间接子类。
  - Swing组件以"J"开头，除了有与AWT类似的按钮（JButton)、标签（JLabel)、复选框（JCheckBox)、菜单（JMenu）等基本组件外，还增加了一个丰富的高层组件集合，如表格（JTable）、树（JTree）。

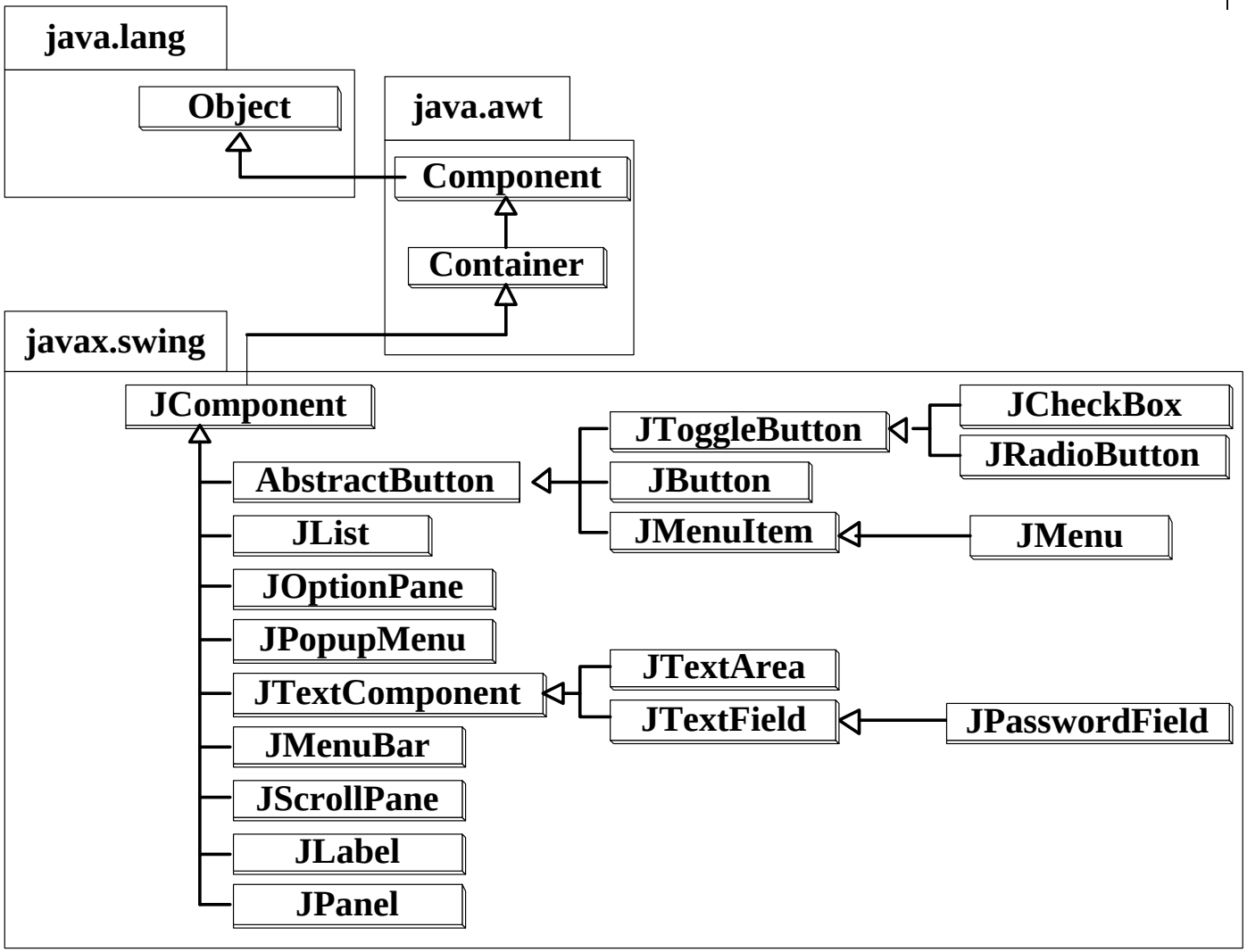
9.1 概述

- AWT与Swing顶层窗口类之间的关系



9.1 概述

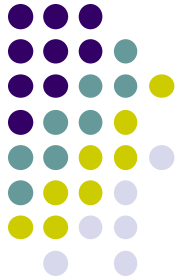
- Swing组件



9.1 概述

- **Swing的组件和容器层次**
  - **JComponent类是除了顶层容器以外所有Swing组件的基类，根据继承关系，我们可以在每个基类中找到大多数GUI组件的常见操作**
  - **Component 类**
    - 包含paint、repaint方法，可以在屏幕上绘制组件
    - 大多数GUI组件直接或间接扩展Component
  - **Container 类**
    - 容纳相关组件
    - 包括add方法，用来添加组件
    - 包括setLayout方法，这个方法可用来设置布局，以帮助Container对象对其中的组件进行定位和设置组件大小
  - **JComponent 类——多数Swing组件的超类**
    - 可抽换的外观和感觉，即可根据需求定制外观和感觉。
    - 快捷键 (通过键盘直接访问GUI组件)
    - 一般的事件处理功能

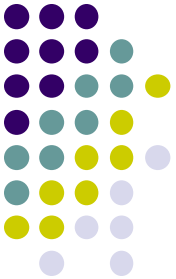
9.1 概述 -- Swing的组件和容器层次



- 通常将**javax.swing**包里的**Swing**组件归为三个层次
  - **顶层容器**
  - **中间层容器**

  容器本身也是一种组件
  - **基本组件**
- 容器层次结构
  - **是一个以顶层容器为根的树状组件集合**
  - **为了显示在屏幕上，每个组件必须是一套容器层次结构的一部分**
- 每个组件只能放置在某个容器内一次
  - **如果某个组件已经在一个容器中，又将它加到另外一个容器中，这个组件就会从第一个容器中清除**

9.1 概述 -- Swing的组件和容器层次

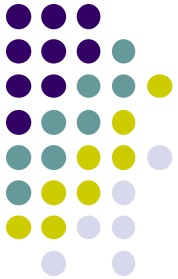


- 顶层容器
  - Swing提供三个顶层容器的类
    - JFrame 实现单个主窗口
    - JDialog 实现一个二级窗口(对话框)
    - JApplet 在浏览器窗口中实现一个applet显示区域
  - 必须和操作系统打交道，所以都是重量级组件
  - 从继承结构上来看，它们分别是从原来AWT组件的Frame、Dialog和Applet类继承而来。
  - 每个使用Swing组件的Java程序都必须至少有一个顶层容器，别的组件都必须放在这个顶层容器上才能显现出来。

9.1 概述-- Swing的组件和容器层次

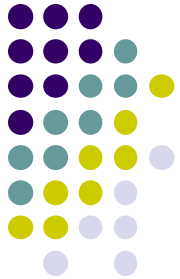
- 中间层容器
  - **其存在的目的仅仅是为了容纳别的组件，使界面有条理，美观，易于控制。**
  - **分为两类**
    - **一般用途的**
      - JPanel
      - JScrollPane
      - JSplitPane
      - JTabbedPane
      - JToolBar
    - **特殊用途的**
      - JInternalFrame
      - JRootPane
  - **可以直接从顶层容器中获得一个JRootPane对象来直接使用，而别的中间容器使用的时候需要新建一个对象。**

9.1 概述-- Swing的组件和容器层次



- 基本组件
  - **通常是在图形用户界面中和用户进行交互的组件**
  - **基本功能是和用户交互信息，而不像前两种组件那样是用来容纳别的组件的**
  - **根据功能的不同，可被分为三类**
    - **显示不可编辑信息的**JLabel、JProgressBar、JToolTip
    - **有控制功能、可以用来输入信息的**JButton、JCheckBox、JRadioButton、JComboBox、JList、JMenu、JSlider、JSpinner、JTexComponent等
    - **能提供格式化的信息并允许用户选择的**JColorChooser、JFileChooser、JTable、JTree

9.1 概述-- Swing的组件和容器层次

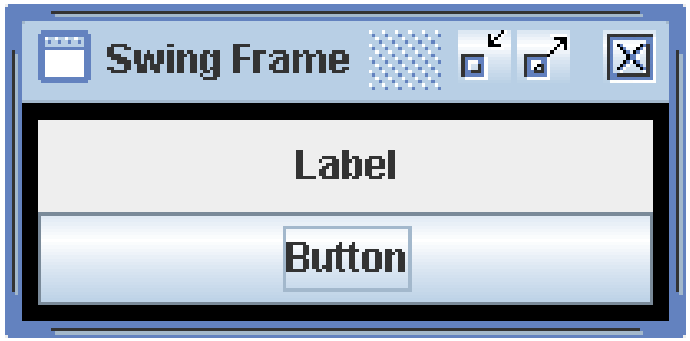


- 三层容器结构示例

```
import javax.swing.*;
import java.awt.*;
public class Ex10_1{
  public static void main(String[] args){
    JFrame.setDefaultLookAndFeelDecorated(true);
    JFrame frame=new JFrame("Swing Frame");
    Container contentPane=frame.getContentPane();
    JPanel panel=new JPanel();
    panel.setBorder(BorderFactory.createLineBorder(Color.black,5));
    panel.setLayout(new GridLayout(2,1));
    JLabel label=new JLabel("Label",SwingConstants.CENTER);
    JButton button=new JButton("Button");
    panel.add(label);
    panel.add(button);
    contentPane.add(panel);
    frame.pack();//对组件进行排列
    //frame.show();//显示
    frame.setVisible(true);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  }
}
```

9.1 概述-- Swing的组件和容器层次

- 运行结果

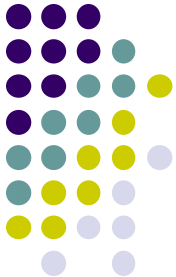


- 程序说明
  - 首先创建一个**JFrame类**顶级容器**frame**
  - 然后获得顶级容器的内容面板**contentPane**，只有通过它才能加入其他组件。
  - 然后创建一个**JPanel类**的中间容器**panel**，并设置边框以及布局
  - 然后创建基本控制组件**Label**，**button**，并将它们添加到中间容器上
  - 然后将中间容器通过内容面板添加到顶层容器上，并对组件进行排列

基于Swing的Java GUI设计思路

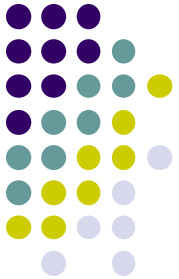
- 基本的java程序的GUI设计工具。主要包括下述几个概念：
  - 组件—Component
  - 容器—Container
  - 布局管理器—LayoutManager
  - 事件处理
- 在Java中，开发一个GUI程序，通常需要以下几步：
  - 构建一个顶层容器；通常是JFrame或JApplet
  - 构建若干个组件，组件可以是其它容器；
  - 设定容器的布局管理器；用容器的add方法将这些组件加入到这个容器中；
  - 设置组件事件；并将组件事件与代码关联。

第九章 Swing图形用户界面



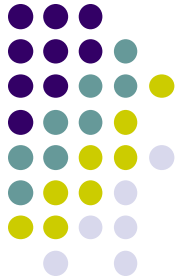
- 9.1 概述
- 9.2 容器组件
- 9.3 基本组件
- 9.4 布局管理器
- 9.5 事件处理模型
- 9.6 本章小结

9.2 容器组件



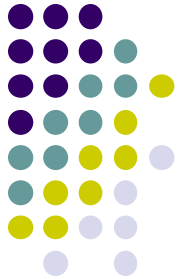
- JFrame
- JDialog
- JApplet
- JPanel
- JScrollPane
- JSplitPane
- JTabbedPane
- JToolBar

9.2 容器组件



- Swing提供了3个顶层容器类：
  - JFrame、JApplet、 JDialog
  - 都是重量级组件，分别继承了AWT组件Frame、Applet和Dialog
  - 每个顶层容器都有一个内容面板，通常直接或间接的容纳别的可视组件
  - 可以有选择地为顶层容器添加菜单，菜单被放置在顶层容器上

9.2 容器组件 —— JFrame



- **框体(JFrame)**
  - JFrame是一种具有边框的容器，它是Java Application程序的图形用户界面的最外层容器。
  - 特点：

(1) 有边框

(2) 能被移动、缩放和关闭

(3) 作为最外层容器，不能被其它容器所包含

(4) 默认布局：边界布局（BorderLayout）

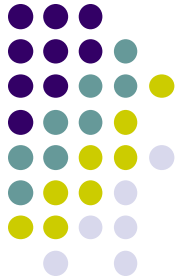
9.2 容器组件 —— JFrame

- 类 JFrame 是java.awt.Frame的子类
- 在Swing的组件中, JFrame 并不全是由Java编写的
  - 是一种与平台关系比较密切的组件(Heavyweight component)

```
java.lang.Object
   |
   +--java.awt.Component
          |
          +--java.awt.Container
                 |
                 +--java.awt.Window
                        |
                        +--java.awt.Frame
                               |
                               +--javax.swing.Jframe
```

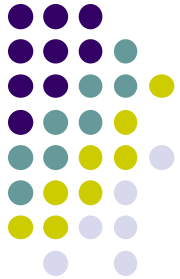
  - 每个包含Swing 组件的主窗口都应用JFrame 来实现

9.2 容器组件 —— JFrame



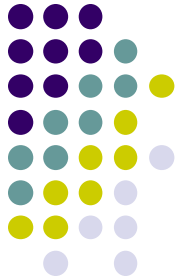
- 构造函数
  - JFrame(); 建立一个无标题的Frame
  - JFrame(String title); 建立一个标题为title的Frame。
- 常用方法
  - String getTitle(); 获取窗口标题
  - void setTitle(String s); 设置窗口标题
  - void setVisible(boolean b); 设置窗口可见性
  - void setBounds(int a, int b, int width, int height); 设置窗口位置和大小
  - void setBackground(Color c); 设置窗口背景颜色
  - void pack(); 用紧凑方式显示窗口
  - void setSize(int width , int height); 设置窗口大小
  - void setLocation(int x, int y); 设置初始位置。其中x, y是窗口左上角在屏幕上的坐标值。

9.2 容器组件 —— JFrame



- **常用方法**
  - **通过setDefaultCloseOperation(int operation)方法设置关闭行为。**
  - **其中operation的取值可以是以下几种：**
    - **DO_NOTHING_ON_CLOSE：当窗口关闭时，不做任何处理；**
    - **HIDE_ON_CLOSE：当窗口关闭时，隐藏这个窗口；**
    - **DISPOSE_ON_CLOSE：当窗口关闭时，隐藏并处理这个窗口；**
    - **EXIT_ON_CLOSE：当窗口关闭时，退出程序。**
    - **默认是HIDE_ON_CLOSE。**

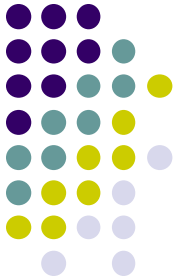
9.2 容器组件 —— JFrame



- 创建窗体有两种方式：
  - **直接创建JFrame的对象，适合于简单框体；**
  - **继承JFrame类，在继承类中编写代码对框体进行详细的刻画，适合框体比较复杂的情况。**
  - **大多数情况下开发人员都用第二种方法。**

- 注意：框体创建以后是不可见的，必须调用Window类的show( )方法或Component类的setVisible( )方法显示该框体
- 见以下两个例子：

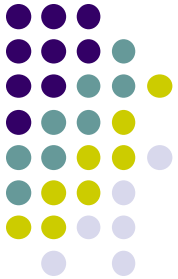
9.2 容器组件 —— JFrame



- 直接创建JFrame的对象

```
import javax.swing.*;
public class Sample10_2 {

    public static void main(String[] args){
        // 创建一个框体对象
        JFrame myWindow = new JFrame();
        // 向框体中添加一个标签
        myWindow.add(new JLabel( “这是一个框体，演示了JFrame类的基本功能"));
        // 设置框体的标题
        myWindow.setTitle( “自定义的框体");
        // 设置框体的大小
        myWindow.setBounds(80,80,480,180);
        // 根据接收的boolean设置框体是否可以调整大小
        myWindow.setResizable(false);
        // 设置框体的可见性
        myWindow.setVisible(true);
    }
}
```

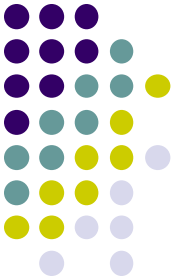
9.2 容器组件 —— JFrame



- 继承JFrame类

```
import javax.swing.*;
class MyWindow extends JFrame{
    // 定义无参构造函数
    public MyWindow(){}
    public MyWindow(boolean b){
        // 向框体中添加一个标签
        this.add(new JLabel("这是一个框体，演示了JFrame类的基本功能"));
        // 设置框体的标题
        this.setTitle("自定义的框体");
        // 设置框体的大小
        this.setBounds(80,80,480,180);
        // 根据接收的boolean设置框体是否可以调整大小
        this.setResizable(b);
        // 设置框体的可见性
        this.setVisible(true);
    }
}
public class Sample10_2{
        public static void main(String[] args){
        // 创建MyWindow类的对象，并传递False值使得框体不能调整大小
        new MyWindow(false);
    }
}
```

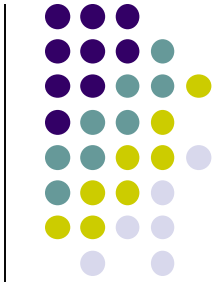
9.2 容器组件 —— JFrame



- 继承JFrame类

```
import javax.swing.*;
public class MyWindow extends JFrame{
    // 定义无参构造函数
    public MyWindow(){}
    public MyWindow(boolean b){
        // 向窗体中添加一个标签
        this.add(new JLabel("这是一个窗体，演示了JFrame类的基本功能"));
        // 设置窗体的标题
        this.setTitle("自定义的窗体");
        // 设置窗体的大小
        this.setBounds(80,80,480,180);
        // 根据接收的boolean设置窗体是否可以调整大小
        this.setResizable(b);
        // 设置窗体的可见性
        this.setVisible(true);
    }
    public static void main(String[] args){
        new MyWindow(false);
    }
}
```

9.2 容器组件 —— JFrame



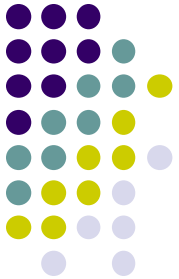
- 在JFrame中加入组件的方法
  - 用getContentPane( )方法获得JFrame的内容面板，再对其加入组件：

```
Container c=frame.getContentPane()
c.add(childComponent)
```

  - 建立一个JPanel或 JDesktopPane之类的中间容器，把组件添加到容器中，用setContentPane()方法把该容器置为JFrame的内容面板：

```
JPanel contentPane=new JPanel( );
……//把其它组件添加到Jpanel中;
frame.setContentPane(contentPane);
//把contentPane对象设置成为frame的内容面板
```

9.2 容器组件



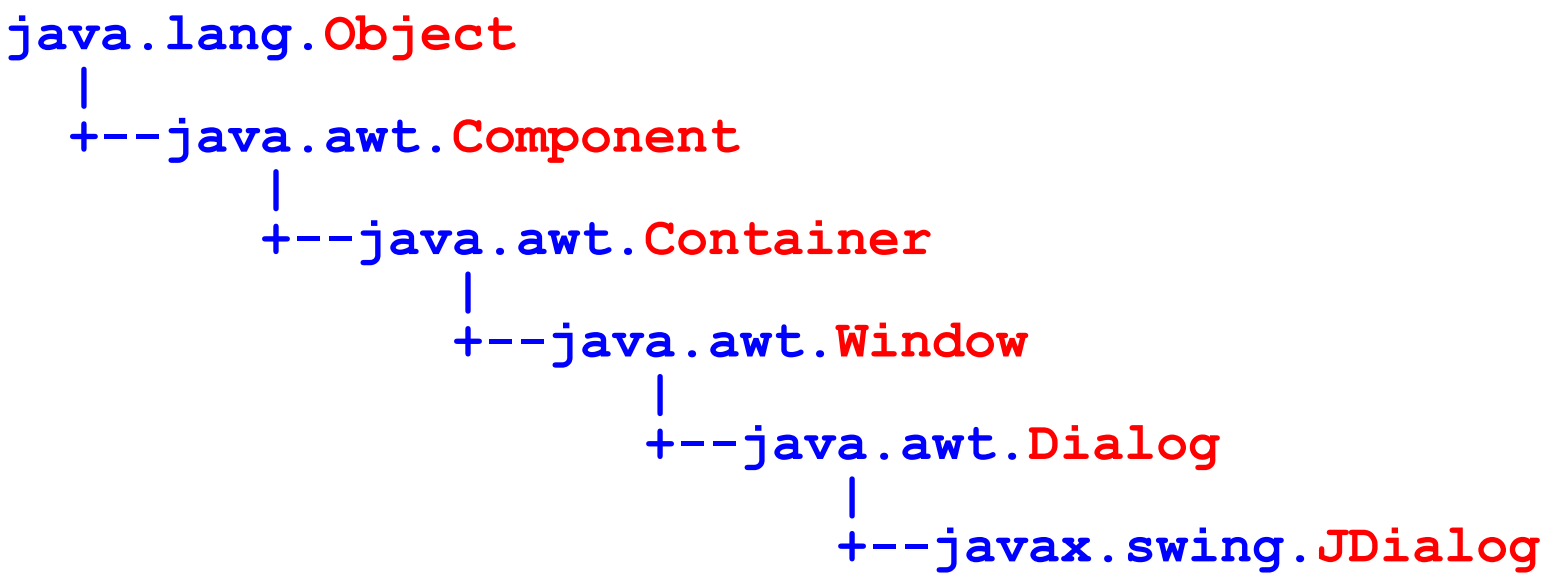
- JFrame
- JDialog
- JApplet
- JPanel
- JScrollPane
- JSplitPane
- JTabbedPane
- JToolBar

9.2 容器组件 —— JDialog

- 对话框（JDialog）
  - 对话框（JDialog）与JFrame一样，都是有边框和标题的独立使用的容器，不能被其它容器所包容。
  - 与JFrame的不同之处：不能作为程序的最外层容器，它必须属于某个JFrame，由该JFrame弹出。
  - 对话框可以接受用户的输入，实现与用户的交互作用。

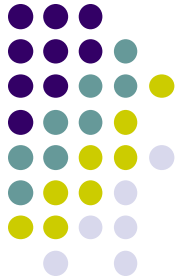
9.2 容器组件 —— JDialog

- JDialog的继承结构



  - **要实现对话框，需要从JDialog派生一个类**

9.2 容器组件 —— JDialog



- 构造函数
  - JDialog(Frame owner)

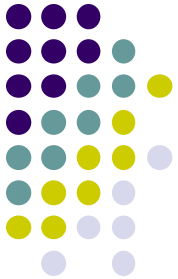
    功能：建立无标题的通用对话框，它属于窗口owner，可以响应其它窗口。
  - JDialog(Frame owner,b oolean d)

    功能：建立无标题的通用对话框，它属于窗口owner，若d为true则不能响应其它窗口。
  - JDialog(Frame owner, String txt)

    功能：建立标题为txt的通用对话框，它属于窗口owner，可以响应其它窗口。
  - JDialog(Frame owner, String txt, boolean d)

    功能：建立标题为txt的通用对话框，它属于窗口owner，若d为true则不能响应其它窗口。

9.2 容器组件 —— JDialog

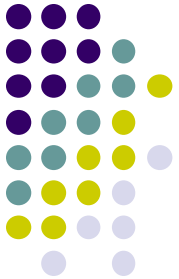


- 常用方法

表列出了通用对话框JDialog类的几个常用的方法。

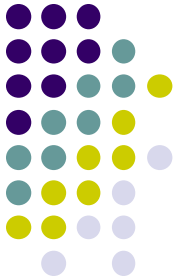
方 法	说 明
Container getContentPane()	返回该通用对话框的contentPanel
void setLayout(LayoutManager manager)	设置布局管理器
Component add(Component comp)	在对话框中添加组件
void setBackground(Color c)	设置对话框的背景色
void setLocation(int x, int y)	设置对话框的显示位置
void setSize(int width, int height)	设置对话框的大小
void setVisible(boolean b)	设置对话框是否可见

9.2 容器组件



- JFrame
- JDialog
- JApplet
- JPanel
- JScrollPane
- JSplitPane
- JTabbedPane
- JToolBar

9.2 容器组件 —— JApplet

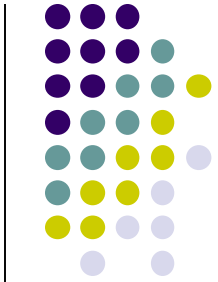


- JApplet的继承结构

```
java.lang.Object
   |
   +--java.awt.Component
          |
          +--java.awt.Container
                 |
                 +--java.awt.Panel
                        |
                        +--java.awt.Applet
                               |
                               +--javax.swing.JApplet
```

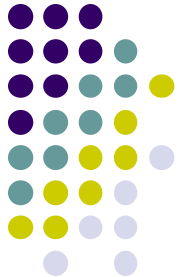
- 每个包含Swing 组件的Applet 都应作为JApplet 的子类来实现
- JApplet类的顶层容器由浏览器提供，通常不需要自己产生一个JApplet类的对象。
- JFrame和JDialog通过构造方法进行创建。

9.2 容器组件



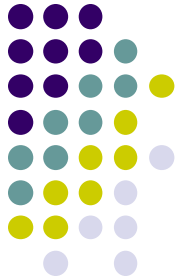
- JFrame
- JDialog
- JApplet
- JPanel
- JScrollPane
- JSplitPane
- JTabbedPane
- JToolBar

9.2 容器组件 —— JPanel



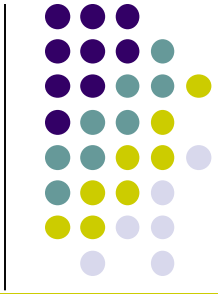
- 面板JPanel
  - Swing的JPanel，它替代了AWT的画布（Canvas）和面板（Panel），兼有二者的功能。
  - 是一种经常使用的轻量级中间容器
  - 在默认状态下，除了背景色外它并不绘制任何东西
  - 很容易地设置边框和绘制特性，可以把它设置为顶层容器contentPane。有效地利用JPanel可以使版面管理更为容易
  - JPanel不能象JFrame那样能够在桌面上浮动。

9.2 容器组件 —— JPanel



- 可以通过setLayout方法来改变其布局
- 也可以在创建一个JPanel对象时就为它确定某种布局方式。
- 在默认状态下panel使用FlowLayout布局，将各组件布局在一行
- 对于复杂GUI，可以添加各种组件(包括面板组件)
  - 面板(JPanel)的大小由它所包含的组件决定
  - 当组件个数增加，面板(JPanel)也会随之而增大

9.2 容器组件 —— JPanel



- 常用方法

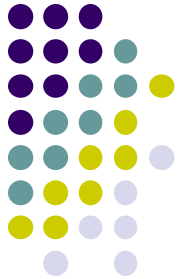
名称	说明
JPanel()	创建一个JPanel，默认布局是FlowLayout
JPanel(LayoutManager layout)	创建一个指定布局的JPanel
void add(Component comp)	添加组件
void add(Component comp, int index)	把组件添加到特定位置上
int getComponentCount()	获得这个panel里所有组件的数量
Component getComponent(int index)	获得指定序号的某个组件
Component getComponentAt(int x, int y)	获得指定位置的某个组件
void remove(Component)	移除某个组件
void removeAll()	移除所有组件
void setLayout(LayoutManager layout)	设置布局
LayoutManager getLayout()	得到现有布局
void setBorder(Border border)	设置边框

9.2 容器组件 —— JPanel

- 利用**JPanel**创建界面

```
import java.awt.*;
import javax.swing.*;
public class JPanelDemo extends JFrame{
        public JPanel getGUI(){
              JPanel p=new JPanel();
               p.add(new JButton("Press me"));
               return p;
        }
        public static void main(String args[]){
                JPanelDemo jp=new JPanelDemo();
                jp.setTitle("JPanel Demo");
                jp.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                jp.setContentPane(jp.getGUI());
                jp.setSize(200,200);
                jp.setVisible(true);
          }
}
```

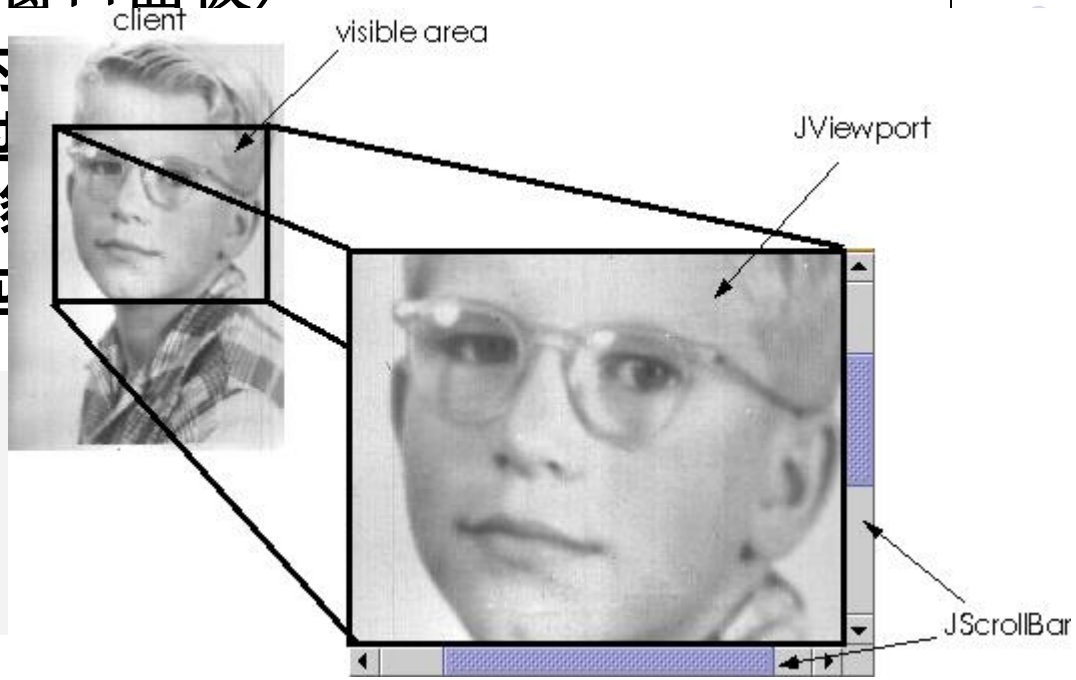
9.2 容器组件



- JFrame
- JDialog
- JApplet
- JPanel
- JScrollPane
- JSplitPane
- JTabbedPane
- JToolBar

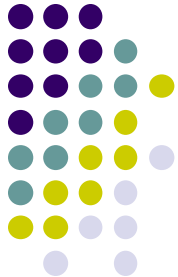
9.2 容器组件 —— JScrollPane

- JScrollPane（滚动窗口面板）
  - 当容器内要容纳的内
    有一个滚动条，通过
    JScrollPane就是能够
  - 由九个部分组成，包



- JScrollPane主要是通过移动JViewport(视角)来实现的。
- JViewport是一种特殊的对象，用于查看基层组件，滚动条实际就是沿着组件移动视口，同时描绘出它在下面"看到"的内容。

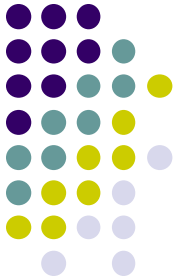
9.2 容器组件 —— JScrollPane



- JScrollPane常用API

名称	说明
static int HORIZONTAL_SCROLLBAR_ALWAYS	水平滚动条策略常数：总是显示
static int HORIZONTAL_SCROLLBAR_AS_NEEDED	水平滚动条策略常数：当显示内容水平区域大于显示区域时才出现
static int HORIZONTAL_SCROLLBAR_NEVER	水平滚动条策略常数：总是不显示
static int VERTICAL_SCROLLBAR_ALWAYS	垂直滚动条策略常数：总是显示
static int VERTICAL_SCROLLBAR_AS_NEEDED	垂直滚动条策略常数：当显示内容垂直区域大于显示区域时才出现
static int VERTICAL_SCROLLBAR_NEVER	垂直滚动条策略常数：总是不显示
JScrollPane()	建立一个空的JScrollPane对象
JScrollPane(Component view）	建立一个显示组件的JScrollPane对象，当组件内容大于显示区域时，自动产生滚动条

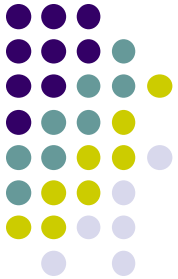
9.2 容器组件 —— JScrollPane



- JScrollPane常用API

名称	说明
void setViewportView(Component)	设置JscroollPane要中心地带要显示的组件
void setVerticalScrollBarPolicy(int) int getVerticalScrollBarPolicy()	设置或者读取垂直滚动条策略常数
void setHorizontalScrollBarPolicy(int) int getHorizontalScrollBarPolicy()	设置或者读取水平滚动条策略常数
void setViewportBorder(Border) Border getViewportBorder()	设置或者读取中心显示地带的边框
void setWheelScrollingEnabled(Boolean) Boolean isWheelScrollingEnabled()	设置或者读取是否随着鼠标滚轮滚动出现或隐藏滚动条，默认状态下为真
void setColumnHeaderView(Component) void setRowHeaderView(Component)	设置显示在上面的边上的组件 设置显示在左面的边上的组件
void setCorner(String key, Component corner)	设置要显示在指定角上的组件
Component getCorner(String key)	获得指定角上的组件

9.2 容器组件



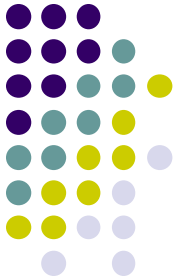
- JFrame
- JDialog
- JApplet
- JPanel
- JScrollPane
- JSplitPane
- JTabbedPane
- JToolBar

9.2 容器组件 —— JSplitPane

- JSplitPane
  - 可以把两个组件显示在两个显示区域内，且随着区域间分隔线的拖动，区域内组件的大小也随之发生变动

件是否
动）
e放到
滚动

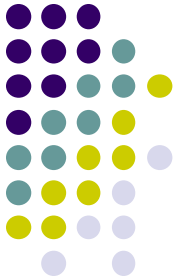
9.2 容器组件 —— JSplitPane



- **JSplitPane常用API**

名称	说明
static int HORIZONTAL_SPLIT	水平分割常数
static int VERTICAL_SPLIT	垂直分割常数
JSplitPane()	创建一个JSplitPane，以水平方向排列，两边各是一个button，没有动态拖曳功能
JSplitPane(int newOrientation)	建立一个指定分割方向的JSplitPane，没有动态拖曳功能，参数为两个分割常数之一
JSplitPane(int newOrientation, Boolean newContinuousLayout)	指定分割方向，并可指定是否有动态拖曳功能
JSplitPane(int newOrientation, Boolean newContinuousLayout, Component newLeftComponent, Component newRightComponent)	指定分割方向、是否具有动态拖曳功能，和两侧组件

9.2 容器组件 —— JSplitPane



- **JSplitPane常用API**

名称	说明
JSplitPane(int newOrientation, Component newLeftComponent, Component newRightComponent)	指定分割方向和两侧组件，无自动拖曳功能
void setOrientation(int newOrientation) int getOrientation()	设置或获得分割方向
void setDividerSize(int) int getDividerSize()	设置或读取分隔线条的粗细
void setContinuousLayout(boolean nCL) boolean isContinuousLayout()	设置或读取是否使用动态拖曳功能
void setOneTouchExpandable(Boolean oTE) boolean is OneTouchExpandable()	设置或读取是否在分隔线上显示一个控键来完全扩展和完全压缩单侧内容。
void remove(Component comp) void add(Component comp)	删除或添加组件。只可以添加两个组件

9.2 容器组件 —— JSplitPane

- JSplitPaneDemo.java

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

public class SplitPaneDemo implements ListSelectionListener {
    private Vector imageNames;
    private JLabel picture;
    private JList list;
    private JSplitPane splitPane;
    public SplitPaneDemo() {
        ResourceBundle imageResource;
        try {  //Read image names from a properties file
            imageResource = ResourceBundle.getBundle("imagenames");
            String imageNamesString = imageResource.getString("images");
            imageNames = parseList(imageNamesString);
        }
        catch (MissingResourceException e) {
             System.exit(1);
        }
```

9.2 容器组件 —— JSplitPaneDemo.java

```
    list = new JList(imageNames);
    list.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
    list.setSelectedIndex(0);
    list.addListSelectionListener(this);
    JScrollPane listScrollPane = new JScrollPane(list);
    ImageIcon firstImage = new
          ImageIcon("./build/classes/"+(String)imageNames.firstElement());
    picture = new JLabel(firstImage);
    picture.setPreferredSize(new Dimension(firstImage.getIconWidth(),
                                           firstImage.getIconHeight()));
    JScrollPane pictureScrollPane = new JScrollPane(picture);
    splitPane = new JSplitPane(JSplitPane.HORIZONTAL_SPLIT,
                               listScrollPane, pictureScrollPane);
    splitPane.setOneTouchExpandable(true);
    splitPane.setDividerLocation(150);
    Dimension minimumSize = new Dimension(100, 50);
    listScrollPane.setMinimumSize(minimumSize);
    pictureScrollPane.setMinimumSize(minimumSize);
    splitPane.setPreferredSize(new Dimension(400, 200));
}
```

9.2 容器组件 —— JSplitPaneDemo.java

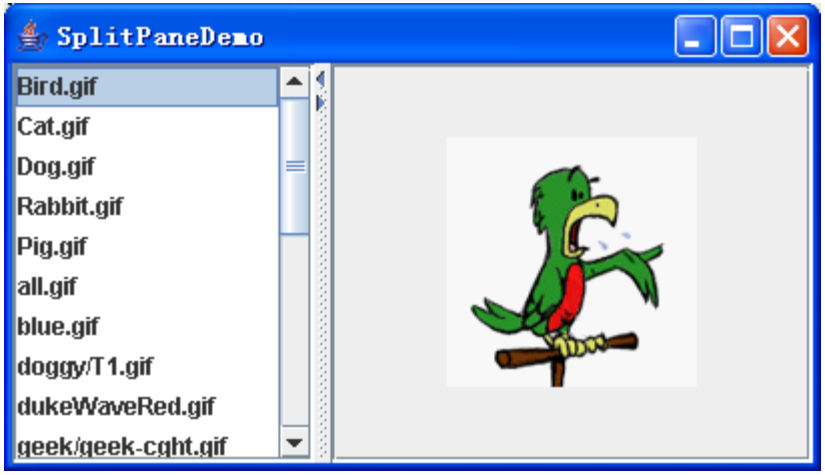
```
public void valueChanged(ListSelectionEvent e) {
    if (e.getValueIsAdjusting())
        return;
    JList theList = (JList)e.getSource();
    if (theList.isSelectionEmpty()) {
        picture.setIcon(null);
    } else {
        int index = theList.getSelectedIndex();
        ImageIcon newImage = new ImageIcon("./build/classes/"+
                             (String)imageNames.elementAt(index));
        picture.setIcon(newImage);
        picture.setPreferredSize(new Dimension(newImage.getIconWidth(),
                                 newImage.getIconHeight() ));
        picture.revalidate();
    }
}
```

9.2 容器组件 —— JSplitPaneDemo.java

```
protected static Vector parseList(String theStringList) {
        Vector v = new Vector(10);
        StringTokenizer tokenizer = new StringTokenizer(theStringList, " ");
        while (tokenizer.hasMoreTokens()) {
            String image = tokenizer.nextToken();
            v.addElement(image);
        }
        return v;
    }

    public static void main(String s[]) {
        JFrame frame = new JFrame("SplitPaneDemo");
        frame.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {System.exit(0);}
        });
        SplitPaneDemo splitPaneDemo = new SplitPaneDemo();
        frame.getContentPane().add(splitPaneDemo.getSplitPane());
        frame.pack();
        frame.setVisible(true);
    }
}
```

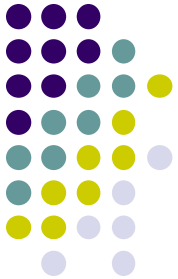
9.2 容器组件 —— JSplitPaneDemo.java

- **JSplitPaneDemo运行结果**



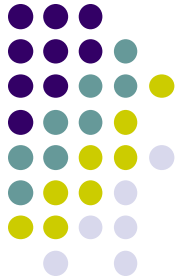
- 程序说明
  - **将一个JList组件放在一个JScrollPane容器中**
  - **将一个绘有图片的JLabel放在另一个JScrollPane容器中**
  - **将两个JScrollPane容器放到一个JSplitPane容器中**
  - **该类实现了ListSelectionListener接口，对列表选择事件可以做出反应，使JLable显示出不同的图片**

9.2 容器组件



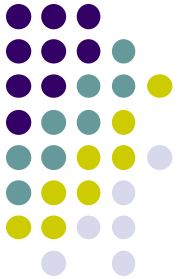
- JFrame
- JDialog
- JApplet
- JPanel
- JScrollPane
- JSplitPane
- JTabbedPane
- JToolBar

9.2 容器组件 —— JTabbedPane



- **JTabbedPane**
  - **如果一个窗口的功能有几项，可以给每项设置一个标签，每个标签下面包含为完成此功能专用的若干组件。**
  - **用户要使用哪项功能，只要单击相应的标签，就可以进入相应的页面。**
  - **这种选项卡功能的实现就需要使用JTabbedPane这个中间层容器。**

9.2 容器组件 —— JTabbedPane



- **TabbedPaneDemo.java**

```
import javax.swing.JTabbedPane;
import javax.swing.ImageIcon;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JFrame;
import java.awt.*;
import java.awt.event.*;

public class TabbedPaneDemo extends JPanel {
    public TabbedPaneDemo() {
        ImageIcon icon = new ImageIcon("images/middle.gif");
        JTabbedPane tabbedPane = new JTabbedPane();
        Component panel1 = makeTextPanel("Blah");
        tabbedPane.addTab("One", icon, panel1, "Does nothing");
        tabbedPane.setSelectedIndex(0);

        Component panel2 = makeTextPanel("Blah blah");
        tabbedPane.addTab("Two", icon, panel2, "Does twice as much nothing");
```

9.2 容器组件 —— JTabbedPane

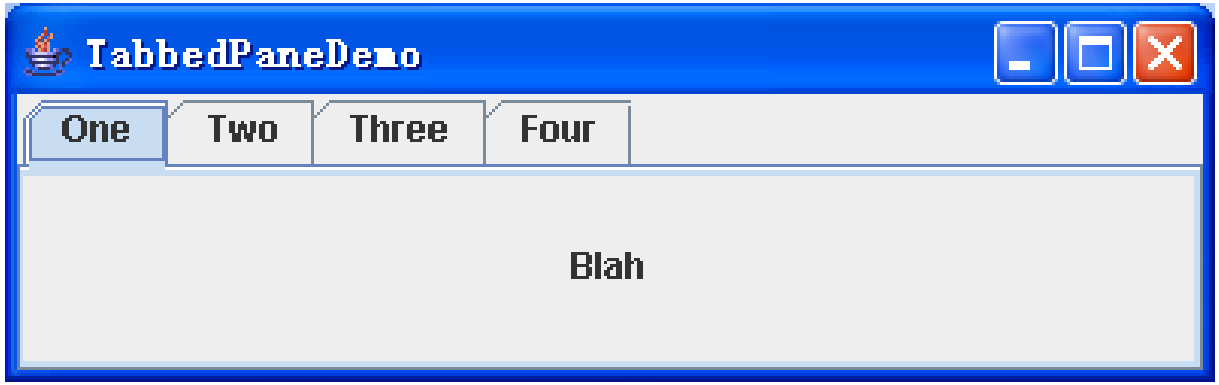
```
    Component panel3 = makeTextPanel("Blah blah blah");
    tabbedPane.addTab("Three", icon, panel3, "Still does nothing");
    Component panel4 = makeTextPanel("Blah blah blah blah");
    tabbedPane.addTab("Four", icon, panel4, "Does nothing at all");

    //Add the tabbed pane to this panel.
    setLayout(new GridLayout(1, 1));
    add(tabbedPane);
}

protected Component makeTextPanel(String text) {
    JPanel panel = new JPanel(false);
    JLabel filler = new JLabel(text);
    filler.setHorizontalAlignment(JLabel.CENTER);
    panel.setLayout(new GridLayout(1, 1));
    panel.add(filler);
    return panel;
}
```

9.2 容器组件 —— JTabbedPane

```
public static void main(String[] args) {
        JFrame frame = new JFrame("TabbedPaneDemo");
        frame.addWindowListener(new WindowAdapter()
        {   public void windowClosing(WindowEvent e)
                {System.exit(0);}
        }
         );

 frame.getContentPane().add(new TabbedPaneDemo(), BorderLayout.CENTER);
        frame.setSize(400, 125);
        frame.setVisible(true);
    }
}
```

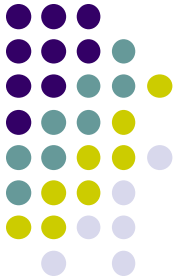
9.2 容器组件 —— JTabbedPane

- **TabbedPaneDemo.java运行结果**



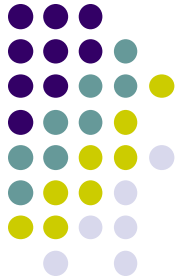
- 说明
  - **在构造函数中创建四个JPanel类型的对象**
  - **并将其依次添加到一个JTabbedPane类的对象容器中**
  - **最后再将该JTabbedPane类的对象容器放在一个JFrame对象的内容面板中**

9.2 容器组件



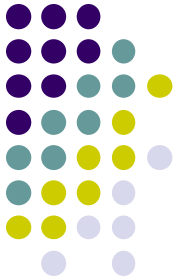
- JFrame
- JDialog
- JApplet
- JPanel
- JScrollPane
- JSplitPane
- JTabbedPane
- JToolBar

9.2 容器组件 —— JToolBar



- **JToolBar**
  - 一般在设计界面时，会将所有功能分类放置在菜单中（JMenu），但当功能相当多时，可能会使成用户为一个简单的操作反复地寻找菜单中相关的功能
  - 可以把一些常用的功能以工具栏的方式呈现在菜单下，这就是JToolBar容器类的好处
  - 可以将JToolBar设计为水平或垂直方向的，
  - 也可以以鼠标拉动的方式来改变

9.2 容器组件 —— JToolBar



- JToolBar常用API

名称	说明
JToolBar() JToolBar(int orientation) JToolBar(String name) JToolBar(String name, int orientation)	创建一个工具栏，可以指定其朝向orientation，为SwingConstants中的HORIZONTAL或VERTICLE，也可以指定游离工具栏显示的名称name。
void add(Component)	为工具栏添加一个组件
void addSeparator()	在末尾增加一个分隔线
void setFloatabled(Boolean floatabled) Boolean isFloatable()	设置或读取工具栏是否可以游离，成为一个独立的窗口

9.2 容器组件 —— ToolBarDemo.java

```
import javax.swing.JToolBar;
import javax.swing.JButton;
import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JTextArea;
import javax.swing.JScrollPane;
import javax.swing.JPanel;
import java.awt.*;
import java.awt.event.*;

public class ToolBarDemo extends JFrame {
    protected JTextArea textArea;
    protected String newline = "\n";
    public ToolBarDemo() {
        //Do frame stuff.
        super("ToolBarDemo");
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                System.exit(0);
            }
        });
```

9.2 容器组件 —— ToolBarDemo.java

```
    JToolBar toolBar = new JToolBar();  //Create the toolbar.
    addButtons(toolBar);
    //Create the text area used for output.
    textArea = new JTextArea(5, 30);
    JScrollPane scrollPane = new JScrollPane(textArea);
    //Lay out the content pane.
    JPanel contentPane = new JPanel();
    contentPane.setLayout(new BorderLayout());
    contentPane.setPreferredSize(new Dimension(400, 100));
    contentPane.add(toolBar, BorderLayout.NORTH);
    contentPane.add(scrollPane, BorderLayout.CENTER);
    setContentPane(contentPane);
}
```

9.2 容器组件 —— ToolBarDemo.java

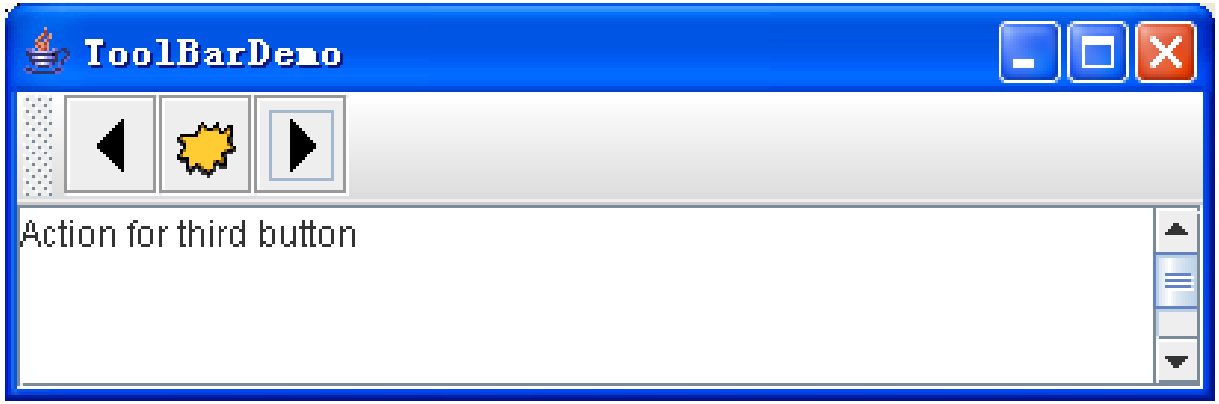
```
protected void addButtons(JToolBar toolBar) {
        JButton button = null;
        //first button
        button = new JButton(new ImageIcon("./build/classes/left.gif"));
        button.setToolTipText("This is the left button");
        button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                displayResult("Action for first button");
            }
        });
        toolBar.add(button);

       //second button
        button = new JButton(new ImageIcon("./build/classes/middle.gif"));
        button.setToolTipText("This is the middle button");
        button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                displayResult("Action for second button");
            }
        });
        toolBar.add(button);
```

9.2 容器组件 —— ToolBarDemo.java

```
        //third button
        button = new JButton(new ImageIcon("./build/classes/right.gif"));
        button.setToolTipText("This is the right button");
        button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                displayResult("Action for third button");
            }
        });
        toolBar.add(button);
    }

    protected void displayResult(String actionDescription) {
        textArea.append(actionDescription + newline);
    }

    public static void main(String[] args) {
        ToolBarDemo frame = new ToolBarDemo();
        frame.pack();
        frame.setVisible(true);
    }
}
```

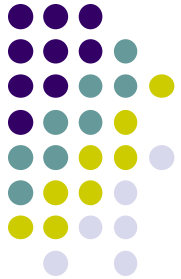
9.2 容器组件 —— ToolBarDemo.java

- ToolBarDemo运行结果



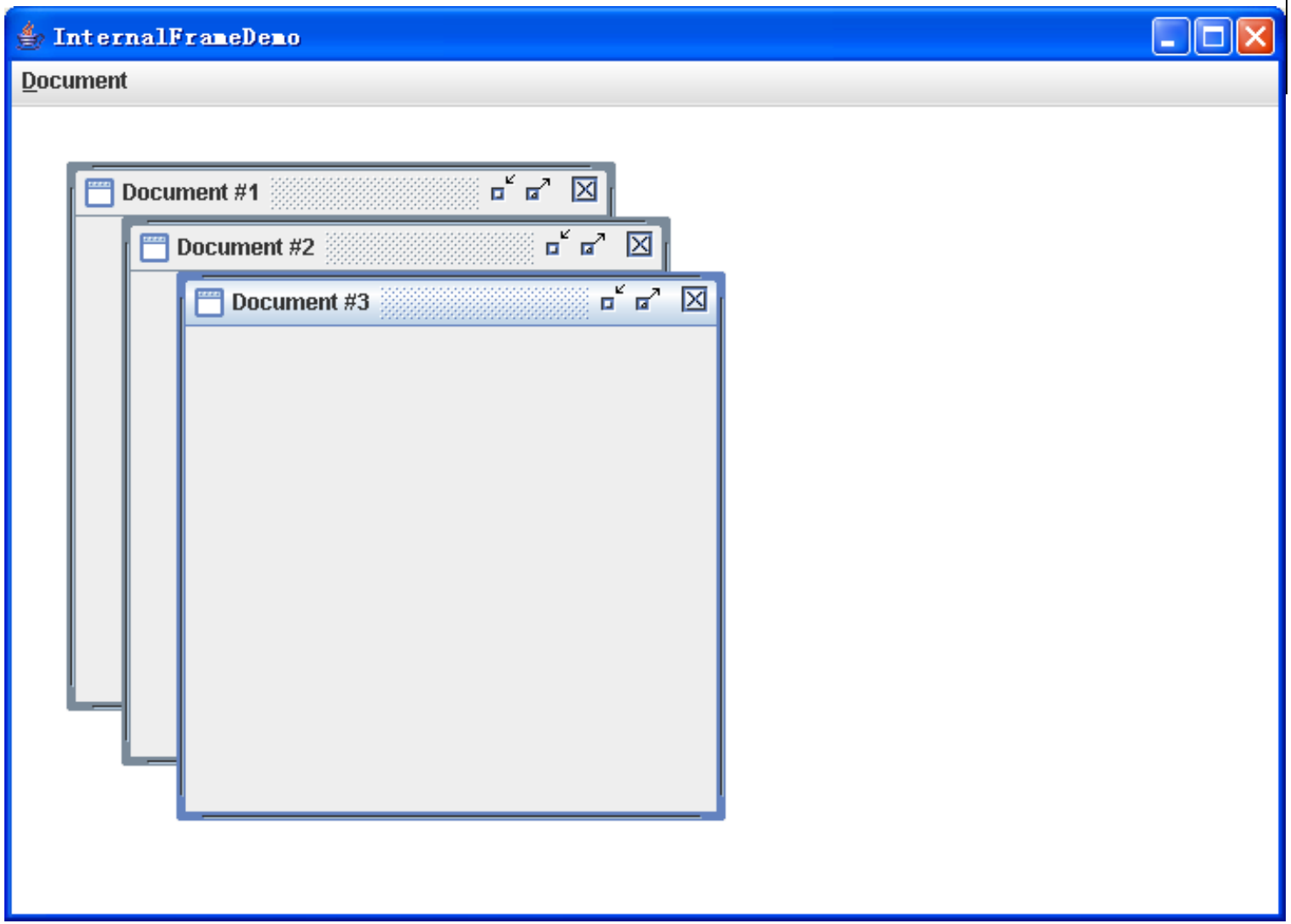
- 说明
  - 创建了三个按钮，并为每个按钮添加了事件监听器
  - 将三个按钮放在一个JToolBar容器中
  - 将JToolBar容器放在顶层容器的内容面板中
  - 将一个包含JtextArea组件的JScrollPane容器也放在顶层容器的内容面板中

9.2 容器组件 —— JInternalFram



- JInternalFrame
  - 如果要实现在一个主窗口中打开很多个文档，每个文档各自占用一个新的窗口，就需要使用JInternalFrame容器类
  - JInternalFrame的使用跟JFrame几乎一样，可以最大化、最小化、关闭窗口、加入菜单
  - 唯一不同的是JInternalFrame是轻量级组件，因此它只能是中间容器，必须依附于顶层容器上
  - 通常我们会将internal frame加入JDesktopPane类的对象来方便管理，JDesktopPane继承自JLayeredPane，用来建立虚拟桌面。
  - 它可以显示并管理众多internal frame之间的层次关系

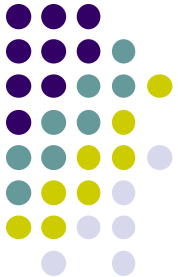
9.2 容器组件 —— JInternalFram



基于Swing的Java GUI设计思路

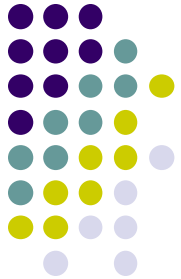
- 基本的java程序的GUI设计工具。主要包括下述几个概念：
  - 组件—Component
  - 容器—Container
  - 布局管理器—LayoutManager
  - 事件处理
- 在Java中，开发一个GUI程序，通常需要以下几步：
  - 构建一个顶层容器；通常是JFrame或JApplet
  - 构建若干个组件，组件可以是其它容器；
  - 设定容器的布局管理器；用容器的add方法将这些组件加入到这个容器中；
  - 设置组件事件；并将组件事件与代码关联。

第九章 Swing图形用户界面



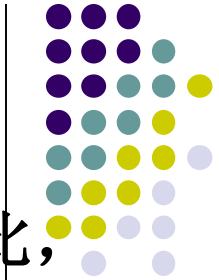
- 9.1 概述
- 9.2 容器组件
- 9.3 基本组件
- 9.4 布局管理器
- 9.5 事件处理模型
- 9.6 本章小结

容器与组件



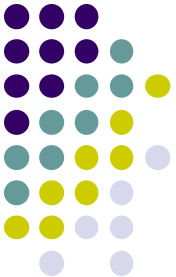
- 组件和容器
  - **容器层次结构**
    - **是一个以顶层容器为根的树状组件集合**
    - **为了显示在屏幕上，每个组件必须是一套容器层次结构的一部分**
  - **每个组件只能放置在某个容器内一次**
    - **如果某个组件已经在一个容器中，又将它加到另外一个容器中，这个组件就会从第一个容器中清除**

9.3 基本组件



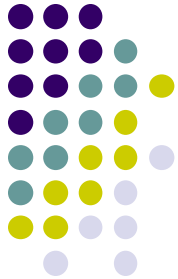
- **Swing基本组件有很多种，与顶层容器和中间容器相比，基本组件用法都比较简单**
- 可将其分为三类
  - **显示不可编辑信息**
    - **JLabel、JProgressBar、JToolTip**
  - **有控制功能、可以用来输入信息**
    - **JButton、JCheckBox、JRadioButton、JComboBox、JList、JMenu、JSlider、JSpinner、JTextComponent**
  - **能提供格式化的信息并允许用户选择**
    - **JColorChooser、JFileChooser、JTable、JTree**

9.3 基本组件——JLabel



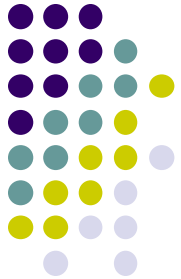
- 标签(JLabel)
  - 为GUI提供文本(主要)或图像(也可以)信息
  - 对应类(JLabel) (JComponent的子类)
  - 可以显示:
    - 单行的只读的文本信息
    - 图像
    - 同时显示文本与图像信息
  - 程序一般不修改标签的内容

9.3 基本组件——JLabel



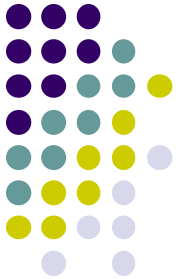
- JLabel的构造方法
  - JLabel()：该方法用来创建一个没有显示内容的对象。
  - JLabel(String label)：该方法用来创建一个显示内容为label的对象。
  - JLabel(String label, int alignment)：该方法除了用来创建一个显示内容为label的对象外，还设置了Label的对齐方式。
  - Label的对齐方式：分别用JLabel类的三个常量LEFT、CENTER和RIGHT来表示左对齐、居中对齐和右对齐。
  - JLabel (Icon ico)：建立一个JLabel对象，其初始图标为ico，没有标题。
  - JLabel (String txt, Icon ico, int halign)：功能，建立一个JLabel对象，其初始标题为txt，初始图标为ico，水平对齐方式为halign。

9.3 基本组件——JLabel



- **Jlabel的常用方法**
  - **public int getAlignment()：返回当前的对齐方式。**
  - **public String getText()：返回当前显示的字符串。**
  - **public void setAlignment(int alignment)： 设置对齐方式。**
  - **public void setText(String label)： 设置显示的字符串。**

9.3 基本组件——JLabel



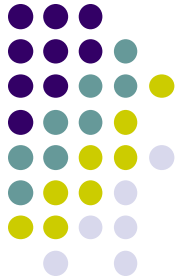
- Jlabel例子

```
import java.awt.Container;
import java.awt.FlowLayout;
import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;

public class JLabelDemo extends JFrame{
   public JLabelDemo( ){
      super( "框架和标签例程" );
      String [ ] s = {"文本标签", "文字在图标的左侧", "文字在图标的下方"};
      ImageIcon [ ] ic = {null, new ImageIcon( "img1.gif" ),
         new ImageIcon( "img2.gif" )};
      int  [ ] ih  = {0, JLabel.LEFT,   JLabel.CENTER};
      int  [ ] iv  = {0, JLabel.CENTER, JLabel.BOTTOM};
      Container c = getContentPane( );
      c.setLayout( new FlowLayout(FlowLayout.LEFT) );
```

9.3 基本组件——JLabel

```
for (int i=0; i<3; i++){
        JLabel aLabel = new JLabel( s[i] , ic[i], JLabel.LEFT);
        if (i>0){
           aLabel.setHorizontalTextPosition(ih[i]);
           aLabel.setVerticalTextPosition(iv[i]);
        }
        aLabel.setToolTipText( "第" + (i+1) + "个标签");
        c.add( aLabel );
     }
  }

 public static void main(String args[ ]){
      JLabelDemo app = new JLabelDemo( );
      app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      app.setSize( 360, 150 );
      app.setVisible( true );
  }
}
```

9.3 基本组件——JProgressBar



- **JProgressBar**
  - **在一些软件运行时，会出现一个进度条告知目前进度如何。通过使用JProgressBar组件我们可以轻松地为软件加上一个进度条**
- 提示信息
  - **通常不必直接处理JToolTip类**
  - **通常使用setToolTipText()方法为组件设置提示信息**
  - **有的组件例如JTabbedPane由多个部分组成，需要鼠标在不同部分停留时显示不同的提示信息，这时可以在其addTab()方法中设置提示信息参数，也可以通过setTooltipTextAt方法进行设置**

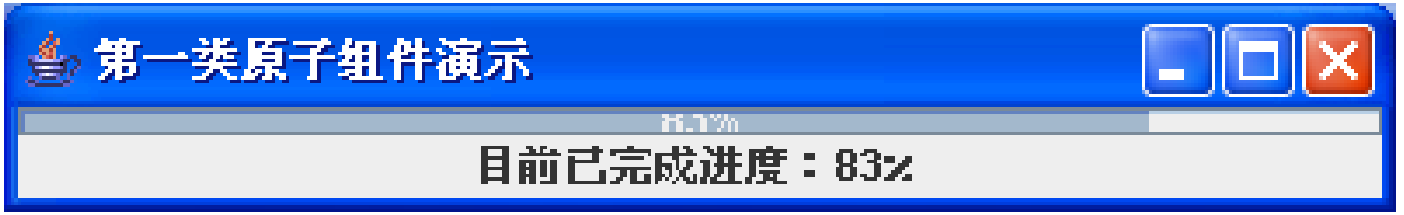
9.3 基本组件——JProgressBar

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

public class ProgressBar  implements ChangeListener {
    JLabel label;
    JProgressBar pb;
    public ProgressBar () {
        int value=0;
        JFrame f=new JFrame("第一类原子组件演示");
        Container contentPane = f.getContentPane ();
        label = new JLabel("", JLabel.CENTER);
        label.setToolTipText ("显示进度信息");
        pb=new JProgressBar();
        pb.setOrientation(JProgressBar.HORIZONTAL); //设置进度条方向
        pb.setMinimum(0);                    //设置最小值
        pb.setMaximum(100);                  //设置最大值
```

```
        pb.setValue(value);                    //初值
        pb.setStringPainted(true);             //设置进度条上显示进度
        pb.addChangeListener(this);            //增加时间监听器
        pb.setToolTipText ("进度条");          //设置提示信息

        contentPane.add(pb, BorderLayout.CENTER);
        contentPane.add(label, BorderLayout.SOUTH);
        f.setSize(400,60);
        f.setVisible(true);

        for(int i=1;i<=500000000;i++) {
            if(i%5000000==0)
            pb.setValue(++value);    //改变进度条的值，触发ChangeEvent
         }
         f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
         public static void main(String[] args) {  new ProgressBar ();  }
         public void stateChanged(ChangeEvent e)  {
               int value=pb.getValue();
               label.setText ("目前已完成进度："+value+"％");
          }
}
```

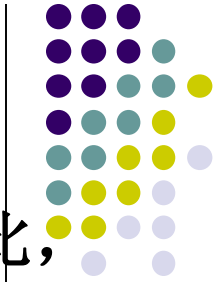
9.3 基本组件——JProgressBar

- 运行结果



- 说明
  - **ProgressBar类实现ChangeListener接口，实现了stateChanged方法**
  - **main方法中随着程序的运行，不断改变进度条的当前值，触发stateChanged方法，在JLable组件上显示当前进度信息**
  - **通过Component类的setTooltipText(String s)方法为组件添加提示信息**

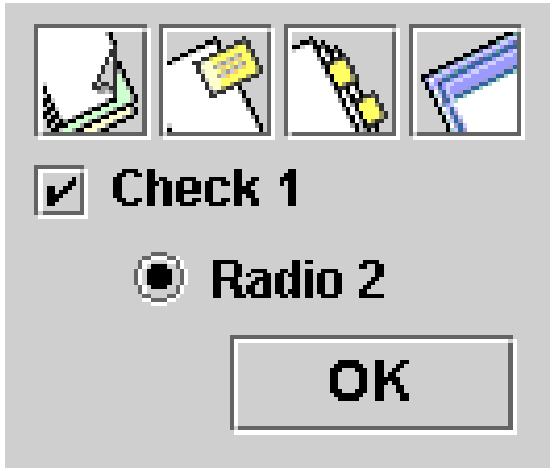
9.3 基本组件



- **Swing基本组件有很多种，与顶层容器和中间容器相比，基本组件用法都比较简单**
- 可将其分为三类
  - **显示不可编辑信息**
    - **JLabel、JProgressBar、JToolTip**
  - **有控制功能、可以用来输入信息**
    - **JButton、JCheckBox、JRadioButton、JComboBox、JList、JMenu、JSlider、JSpinner、JTextComponent**
  - **能提供格式化的信息并允许用户选择**
    - **JColorChooser、JFileChooser、JTable、JTree**

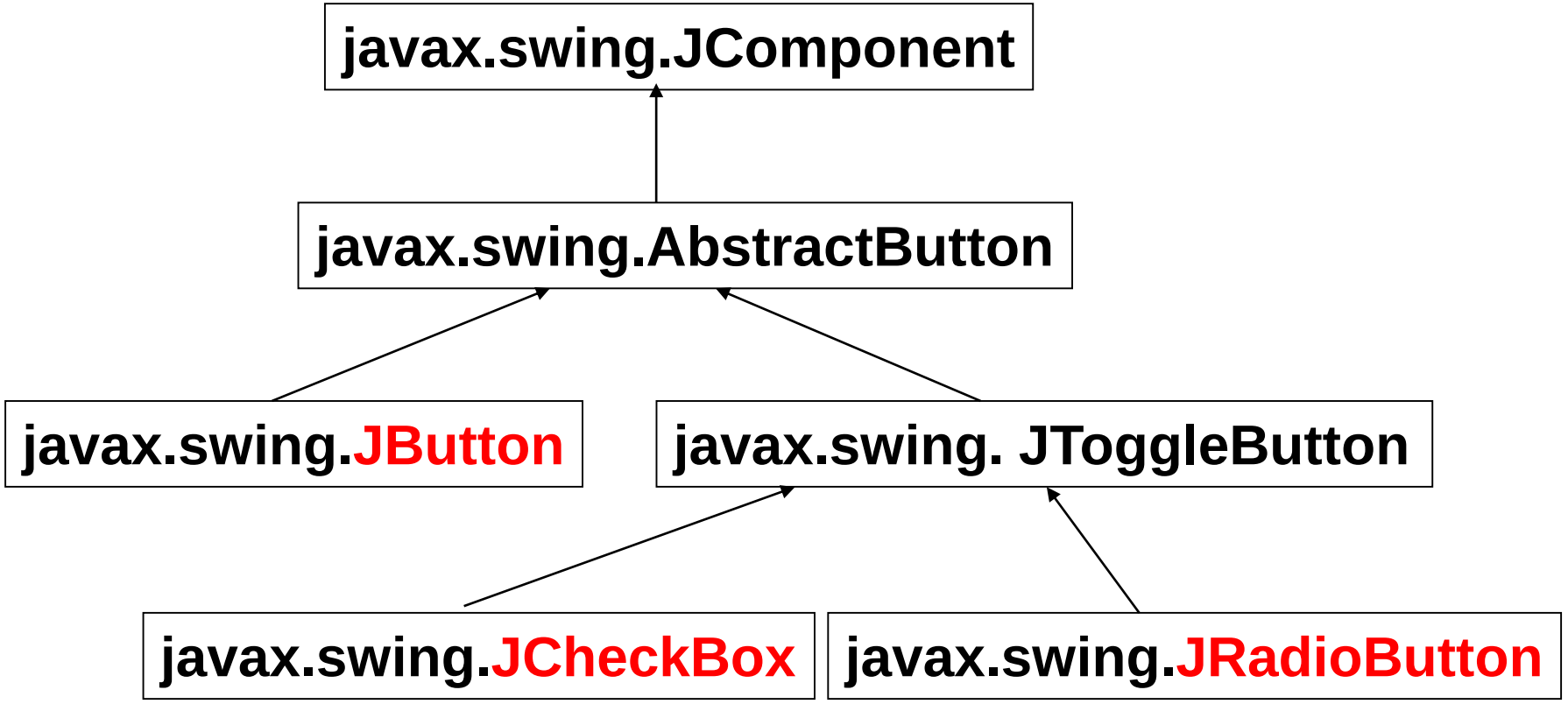
9.3 基本组件——按钮类

- 按钮类
  - **AbstractButton抽象类是众多按钮类的基类，继承它的类包括**
    - **JButton——命令式按钮**
    - **JToggleButton——表示有两个选择状态的按钮**
      - **JCheckBox（多选按钮）**
      - **JRadioButton（单选按钮）**
    - **JMenuItem——在菜单中使用**
      - **JCheckBoxMenuItem（多选按钮）**
      - **JRadioButtonMenuItem（单选按钮）**
      - **JMenu（一般的菜单项）**



9.3 基本组件——按钮类

- 按钮类的层次结构



9.3 基本组件——按钮类

```
……
 Container c = getContentPane( );
 c.setLayout( new FlowLayout( ) );
 int i;
 ImageIcon [ ] ic = {new ImageIcon("left.gif"), new ImageIcon("right.gif")};
 JButton [ ] b = {new JButton("左", ic[0]), new JButton("中间"),new JButton("右", ic[1])};
 for (i=0; i < b.length; i++)
      c.add( b[i] );

 JCheckBox [ ] ck = {new JCheckBox("左"), new JCheckBox("右")};
 for (i=0; i<ck.length; i++){
       c.add( ck[i] );
       ck[i].setSelected(true);
 }

 JRadioButton[ ] r={new JRadioButton("左"), new JRadioButton("右")};
 ButtonGroup rg = new ButtonGroup( );
 for (i=0; i < r.length; i++){
       c.add( r[i] );
       rg.add( r[i] );
 }
 r[0].setSelected(true);
 r[1].setSelected(false);
 ……
```

9.3 基本组件——JButton

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

public class ButtonDemo extends JFrame  implements ActionListener {
    protected JButton button1,button2;
    protected JLabel label1;
    protected int clickedNum = 0;

    public ButtonDemo(String title) {
      setTitle(title);
      label1 = new JLabel("共点击了"+clickedNum+"次");
       button1 = new JButton("点我");
       button2 = new JButton("清零");
       button1.addActionListener(this);
       button2.addActionListener(this);
       setLayout(new FlowLayout());
       setSize(400,200);
       add(button1);
       add(button2);
       add(label1);
    }
```

```
public void actionPerformed(ActionEvent e) {
        if (e.getSource() == button1) {
                clickedNum++;
        }
        if (e.getSource() == button2) {
                clickedNum = 0;
        }
        label1.setText("共点击了"+clickedNum+"次") ;
    }

    public static void main(String[] args) {
        ButtonDemo frame = new ButtonDemo("ButtonDemo1");
        frame.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                System.exit(0);
            }
        });
        frame.setVisible(true);
    }
}
```

9.3 基本组件——JRadioButton

- 单选按钮JRadioButton——构造函数
  - JRadioButton ()

    功能：建立一个无标题、无图片的单选按钮。
  - JRadioButton (String txt)

    功能：建立一个标题为txt但没有图片的单选按钮。
  - JRadioButton (Icon ico)

    功能：建立一个图片为ico但没有标题的单选按钮。
  - JRadioButton (String txt, Icon ico)

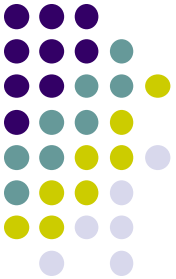
    功能：建立一个标题为txt、图片为ico的单选按钮。
  - JRadioButton (String txt, boolean stat)

    功能：建立一个标题为txt但没有图片的单选按钮，其初始选择状态由stat指定。
  - JRadioButton (Icon ico, boolean stat)

    功能：建立一个图片为ico但没有标题的单选按钮，其初始选择状态由stat指定。
  - JRadioButton (String txt, Icon ico, boolean stat)

    功能：建立一个标题为txt、图片为ico的单选按钮，其初始选择状态由stat指定。

9.3 基本组件——JCheckBox



- 复选按钮（JCheckbox）——构造函数
  - JCheckBox ()

    功能：建立一个无标题、无图片的复选按钮。
  - JCheckBox (String txt)

    功能：建立一个标题为txt但没有图片的复选按钮。
  - JCheckBox (Icon ico)

    功能：建立一个图片为ico但没有标题的复选按钮。
  - JCheckBox (String txt, Icon ico)

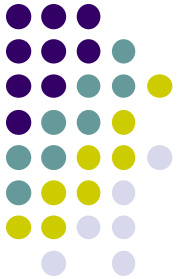
    功能：建立一个标题为txt、图片为ico的复选按钮。
  - JCheckBox (String txt, boolean stat)

    功能：建立一个标题为txt但没有图片的复选按钮，其初始选择状态由stat指定。
  - JCheckBox (Icon ico, boolean stat)

    功能：建立一个图片为ico但没有标题的复选按钮，其初始选择状态由stat指定。
  - JCheckBox (String txt, Icon ico, boolean stat)

    功能：建立一个标题为txt、图片为ico的复选按钮，其初始选择状态由stat指定。

9.3 基本组件—— ButtonGroup



- 按钮组ButtonGroup——把按钮进行分组
  - **构造函数——ButtonGroup()**
    - **功能：建立一个按钮组ButtonGroup**
  - **常用方法**
    - **表列出了按钮组ButtonGroup类的几个常用的方法。**

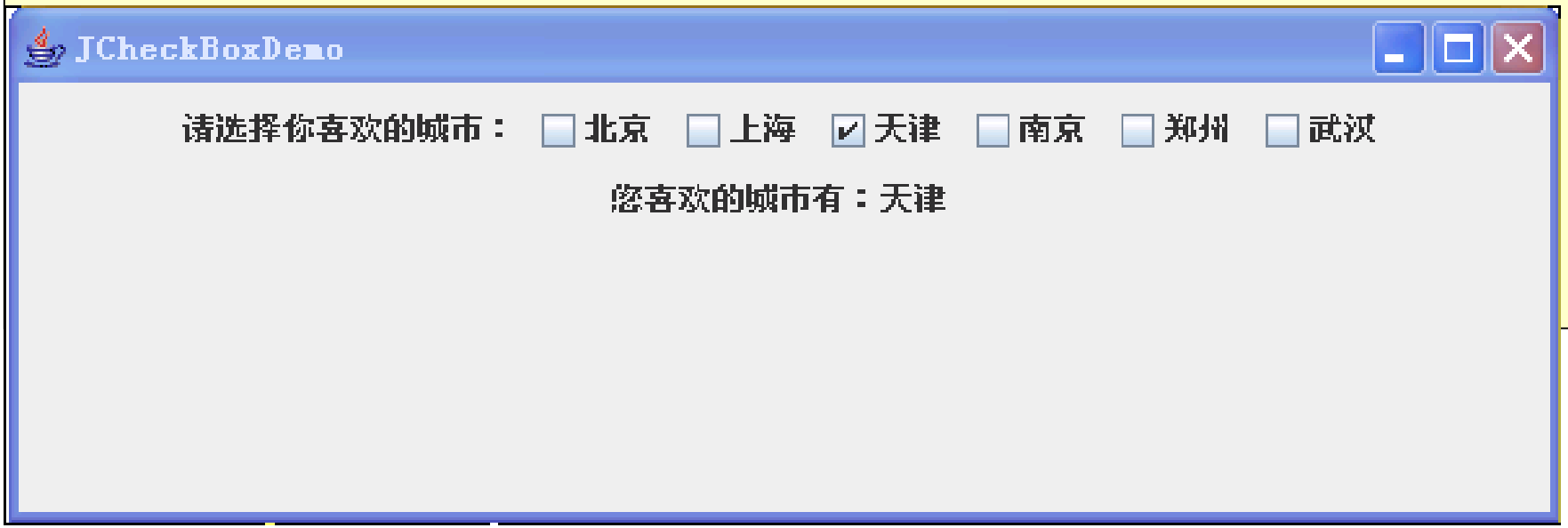
名称	说明
void add(AbstractButton b)	将按钮b加到组中
int getButtonCount()	返回组中按钮数目
void remove(AbstractButton b)	从组中删除按钮b
Enumeration getElements()	返回组中所有按钮。枚举类型Enumeration是java.util中定义的一中数据类型。

9.3 基本组件——JCheckBox例子

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

public class JCheckBoxDemo extends JFrame implements ActionListener {
    String city[]={"北京","上海","天津","南京","郑州","武汉"};
    JCheckBox c[]=new JCheckBox[city.length];
    JLabel result=new JLabel("这是一个复选框的例子");

    public JCheckBoxDemo(String title) {
            this.setTitle(title);
            this.setSize(600, 200);
            this.setLayout(new FlowLayout());
            add( new JLabel("请选择你喜欢的城市：") );
             for (int i=0; i<c.length; i++) {
                 c[i]=new JCheckBox(city[i]);
                 c[i].addActionListener(this);
                 add(c[i]);
             }
             add(result);
    }
```

9.3 基本组件——JCheckBox例子

```
public void actionPerformed(ActionEvent e) {
  String resultTemp="您喜欢的城市有：";
        for (int i = 0; i < c.length; i++){
              if (c[i].isSelected() == true)
              resultTemp = resultTemp+ c[i].getText() +" ";
        }
        result.setText(resultTemp);

}
```

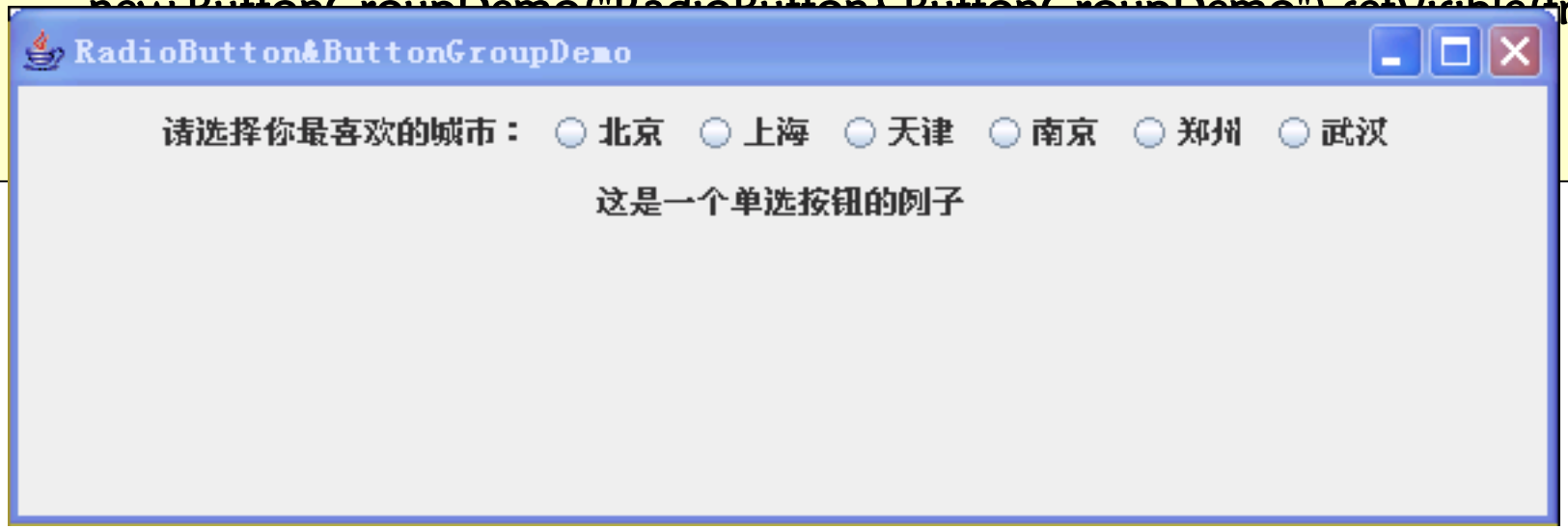


9.3 基本组件——JRadioButton例子

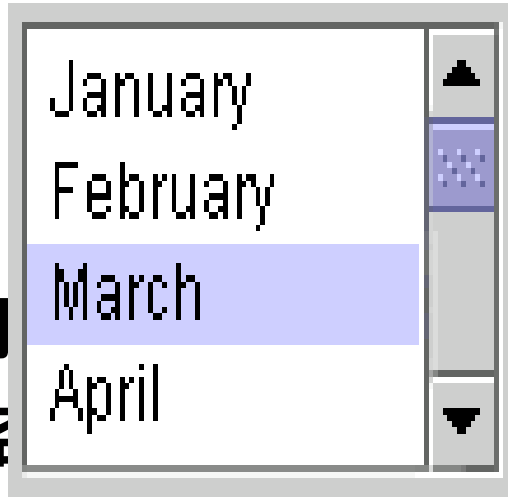
```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class ButtonGroupDemo extends JFrame implements ActionListener {
    String city[]={"北京","上海","天津","南京","郑州","武汉"};
    JRadioButton c[]=new JRadioButton[city.length];
    JLabel result=new JLabel("这是一个单选按钮的例子");
    ButtonGroup bg = new ButtonGroup();
     public ButtonGroupDemo(String title) {
            this.setTitle(title);
            this.setSize(600,200);
            this.setLayout(new FlowLayout());
            add( new JLabel("请选择你最喜欢的城市：") );
             for (int i=0; i<c.length; i++) {
                c[i]=new JRadioButton(city[i]);
                c[i].addActionListener(this);
                add(c[i]);
                bg.add(c[i]); //加到按钮组bg
            }
            add(result);
     }
```

9.3 基本组件——JRadioButton例子

```
    public void actionPerformed(ActionEvent e) {
      String resultTemp="您最喜欢的城市是：";
            for (int i = 0; i < c.length; i++){
                if (c[i].isSelected() == true)
                 resultTemp = resultTemp+ c[i].getText() +" ";
            }
            result.setText(resultTemp);

    }
    public static void main(String[] args) {
        new RadioButtonGroupDemo("RadioButton&ButtonGroupDemo").setVisible(true);
    }
}
```



9.3 基本组件——JList



- 列表框Jlist
  - 除了JCheckBox以外，JList也可以选
  - 有几种各有特色的构造方法（选项是
  - 也提供了很多API可以用来设置各选项的显示方式（在一列还是若干列）、选择模式（一次可选几项及是否可连续）等
  - 因为JList通常含有很多选项，所以经常把它放在一个JScrollPane对象里面
  - JList的事件处理一般可分为两类
    - 取得用户选取的项目，事件监听器是ListSelectionListener
    - 对鼠标事件作出响应，其事件监听器是MouseListener

9.3 基本组件——JList

- 列表框JList
  - 构造函数
    - 格式：**JList ();** 建立一个单选的JList对象。
    - 格式：**JList(Object[] data);**建立一个单选且初始数据为data的JList对象。
  - 例如：String []color={“black",“white",“blue",“yellow"};

    JList jlist=new JList(color);

    创建了一个列表框，其初始数据有4项，分别为：

    “black",“white",“blue",“yellow"。

9.3 基本组件——JList

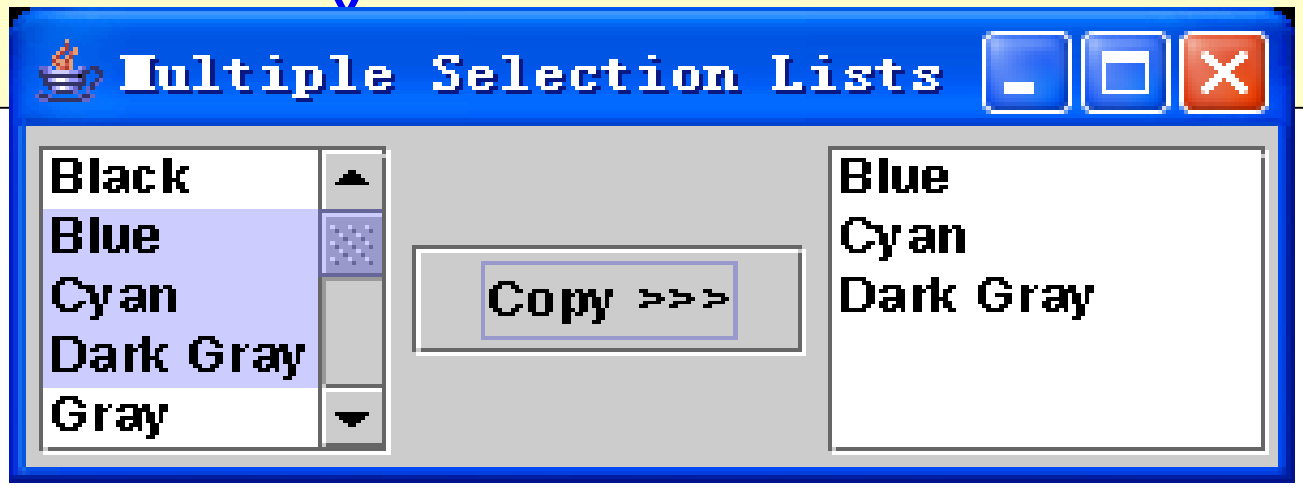
- JList显示一系列选项，用户可以从中选择一个或多项。

```
…….
String colorNames[] ={ “Black”, “Blue”, “Cyan”, “Dark Gray”, “Gray“};
colorList = new JList( colorNames );
colorList.setVisibleRowCount( 5 );
colorList.setSelectionMode(ListSelectionModel.SINGLE_SELECTION );

colorList.getSelectedIndex() ;
colorList.getSelectedValue() ;
……
```

9.3 基本组件——JList

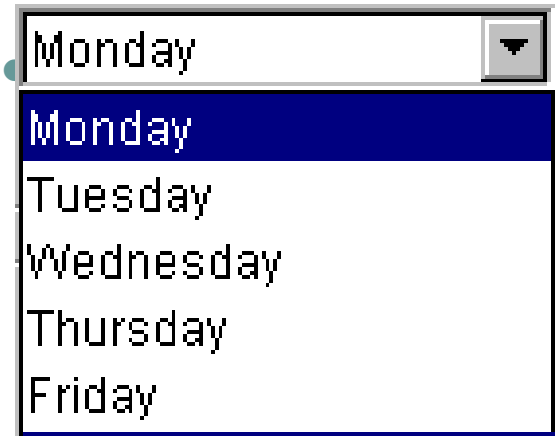
- 多选列表
  - **在JList中选择多个选项，方法是按住shift键或ctrl键。**

```
.......
String colorNames[] ={ “Black”, “Blue”, “Cyan”, “Dark Gray”, “Gray“};
JList  colorList = new JList( colorNames );
colorList.setVisibleRowCount(5);
colorList.setFixedCellHeight(15);
colorList.setSelectionMode(
                    ListSelectionModel.MULTIPLE_INTERVAL_SELECTION );
colorList.getSelectedValues()
......
```

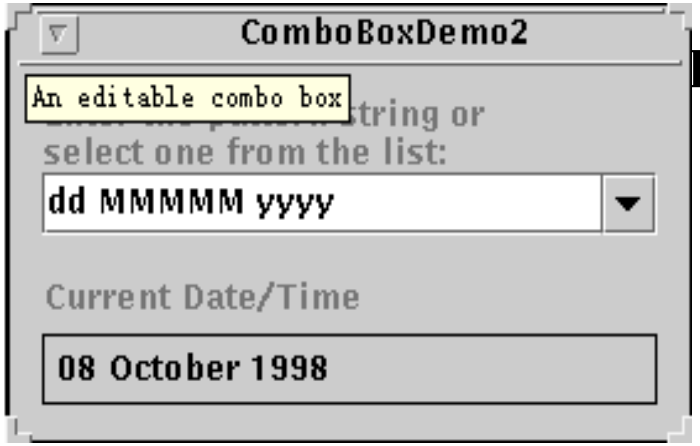


9.3 基本组件——JComboBox

- 组合框JComboBox
  - **允许用户在许多选项中选其一，可以有两种非常不一样的格式**
    - **默认状态是不可编辑的模式**，其特色是包括一个按钮和一个下拉列表，用户只能在下拉列表提供的内容中选其一。**有时也称为下拉框(drop-down list)**
    - **另一种是可编辑的模式**，其特色是多了一个文本区域，用户可以在此文本区域内填入列表中不包括的内容



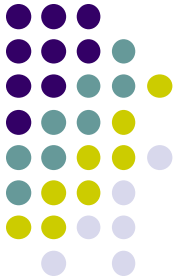
屏幕



heckBox、

大

9.3 基本组件——JComboBox



- 组合框JComboBox
  - 构造函数
    - 格式：JComboBox ();创建具有默认数据模型的组合框。
    - 格式：JComboBox(Object[] data);建立一个初始数据为data的JList对象。
  - 例如：String []color={“black",“white",“blue",“yellow"};

JComboBox jcombo=new JComboBox(color); 创建了一个组合框，其初始数据有4项，分别为：“black",“white",“blue",“yellow"。

- JComboBox中的方法
  - getSelectedIndex( )：返回当前被选中的项
  - setMaximumRowCount(n)：设置最多显示列表项的项数
  - 滚动条(Scrollbar)会自动加上

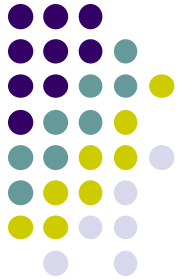
9.3 基本组件——JComboBox例子

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class JCheckBoxDemo extends JFrame implements  ItemListener {
    String city[]={"北京","上海","天津","南京","郑州","武汉"};
    JComboBox combo;
    JLabel la,result;
     public JCheckBoxDemo(String title) {
            this.setTitle(title);
            this.setSize(600, 200);
            this.setLayout(null);
            la=new JLabel("请选择你喜欢的城市：");
            la.setBounds(100,10,200,25);
            add( la);

      combo=new JComboBox(city);
      combo.setBounds(250, 10, 200, 25);
      combo.setEditable (true);
      ComboBoxEditor editor=combo.getEditor();
      combo.configureEditor (editor,"请选择或直接输入城市名称");
      combo.addItemListener(this);
      add(combo);
      result=new JLabel("这是一个复选框的例子");
      result.setBounds(100,100,200,25);
      add(result);
     }
```

9.3 基本组件——JComboBox例子

```
    public void itemStateChanged(ItemEvent e) {
        String resultTemp="您最喜欢的城市是："+ e.getItem().toString();
        result.setText(resultTemp);
    }

    public static void main(String[] args) {
        new JCheckBoxDemo("JCheckBoxDemo").setVisible(true);
    }
}
```

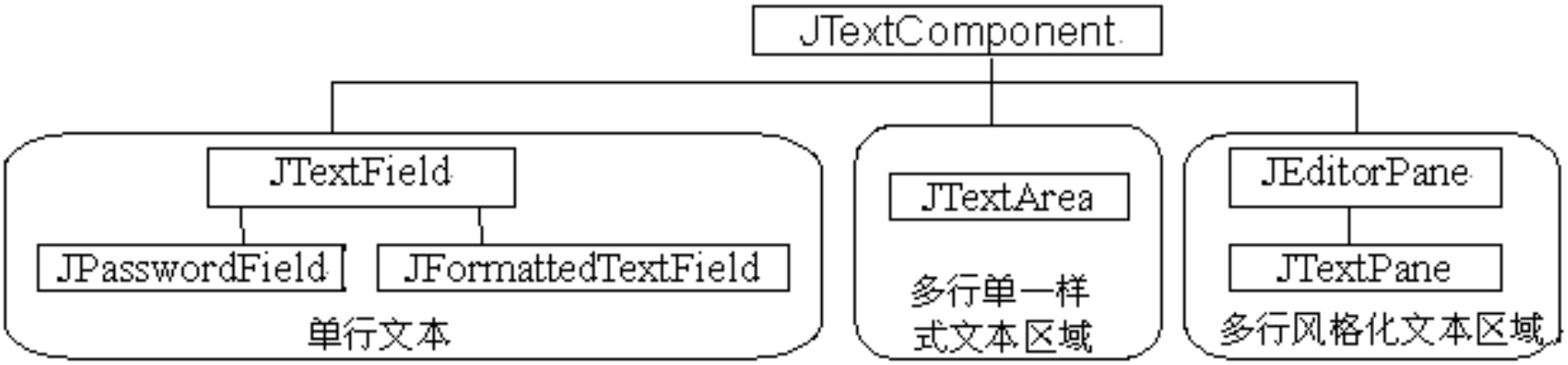
9.3 基本组件——文本组件



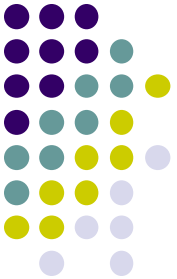
- 文本组件
  - **允许用户在里边编辑文本，能满足复杂的文本需求**
  - **都继承自JTextComponent抽象类，可分为三类**
    - **JTextField、JPasswordField、JFormattedTextField**
      - 它们只能显示和编辑一行文本，像按钮一样，它们可以产生ActionEvent事件，因此通常用来接受少量用户输入信息并在输入结束进行一些事件处理
    - **JTextArea**
      - 它可以显示和编辑多行文本，但是这些文本只能是单一风格的，因此通常用来让用户输入任意长度的无格式文本或显示无格式的帮助信息
    - **JEditorPane、JTextPane**
      - 可以显示和编辑多行多种式样的文本，嵌入图像或别的组件

9.3 基本组件——文本组件

- **JTextComponent类层次结构**

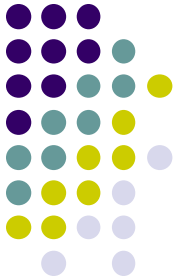


9.3 基本组件——JTextField



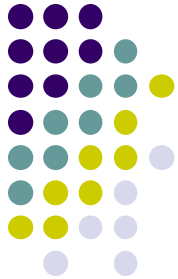
- **JTextField单行文本编辑框**
  - **JTextField 用来接受用户键盘输入的单行文本信息。**
  - **JTextField extends JTextComponent**
  - **密码输入框JPasswordField**

9.3 基本组件——JTextField



- 构造方法
  - JTextField() 创建一个缺省长度的文本框。
  - JTextField(int columns) 创建一个指定长度的文本框。
  - JTextField(String text) 创建一个带有初始文本内容的文本框。
  - JTextField(String text, int columns) 创建一个带有初始文本内容并具有指定长度的文本框。

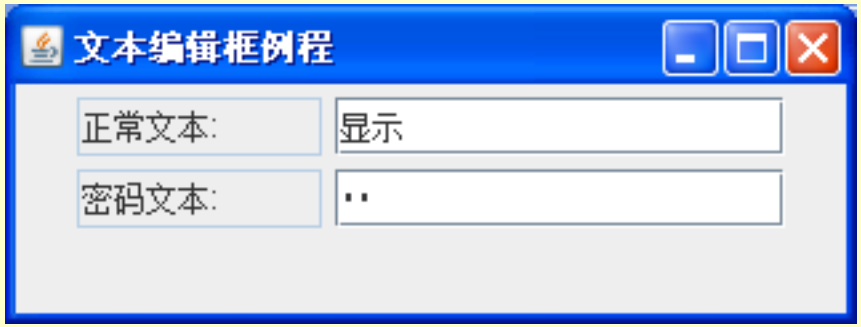
9.3 基本组件——JTextField



- 常用方法

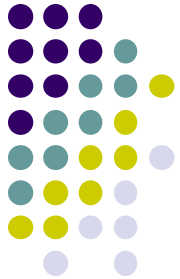
名称	说明
String getText()	返回文本编辑框JTextField中的字符串
void setText(String txt)	设定文本编辑框JTextField中的显示字符串为txt
void setToolTipText(String txt);	设定当光标落在JTextField上时显示的提示信息为txt
int getColumns()	回文本编辑框JTextField中的列数
void setColumns(int len)	设定文本编辑框JTextField中的列数为len
int getHorizontalAlignment()	返回文本编辑框JTextField中的水平对齐方式
void setHorizontalAlignment (int align)	设定文本编辑框JTextField中的水平对齐方式。align的值为：JTextField.LEFT、JTextField.CENTER、JTextField.RIGHT、JTextField.LEADING、JTextField.TRAILING

9.3 基本组件——JTextField例子



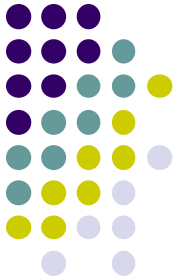

```
import java.awt.Container;
import java.awt.FlowLayout;
import javax.swing.JFrame;
import javax.swing.JPasswordField;
import javax.swing.JTextField;

public class JTextFieldDemo{
    public static void main(String args[ ]){
        JFrame app = new JFrame( "文本编辑框例程" );
        app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        app.setSize( 320, 120 );
        Container c = app.getContentPane( );
        c.setLayout( new FlowLayout( ) );
        JTextField [ ] t = {
            new JTextField("正常文本:", 8), new JTextField("显示", 15),
            new JTextField("密码文本:", 8), new JPasswordField("隐藏", 15)};
        t[0].setEditable( false );
        t[2].setEditable( false );
        for (int i=0; i<4; i++)
            c.add( t[i] );
        app.setVisible( true );
    }
}
```

9.3 基本组件——JTextArea



- **JTextArea文本区**
  - 可以显示大段的文本
- **JTextArea的构造方法**
  - **JTextArea() 创建一个缺省大小的文本区。**
  - **JTextArea(int rows, int columns) 创建一个指定行和列数的文本区。**
  - **JTextArea(String text) 创建一个带有初始文本内容的文本区。**
  - **JTextArea(String text, int rows, int columns) 创建一个带有初始文本内容并具有指定行和列数的文本区。**
  - **JTextArea(String text, int rows, int columns, int scrollbars) 在（4）的基础上添加滚动条。**

9.3 基本组件——JTextArea



- 常用方法

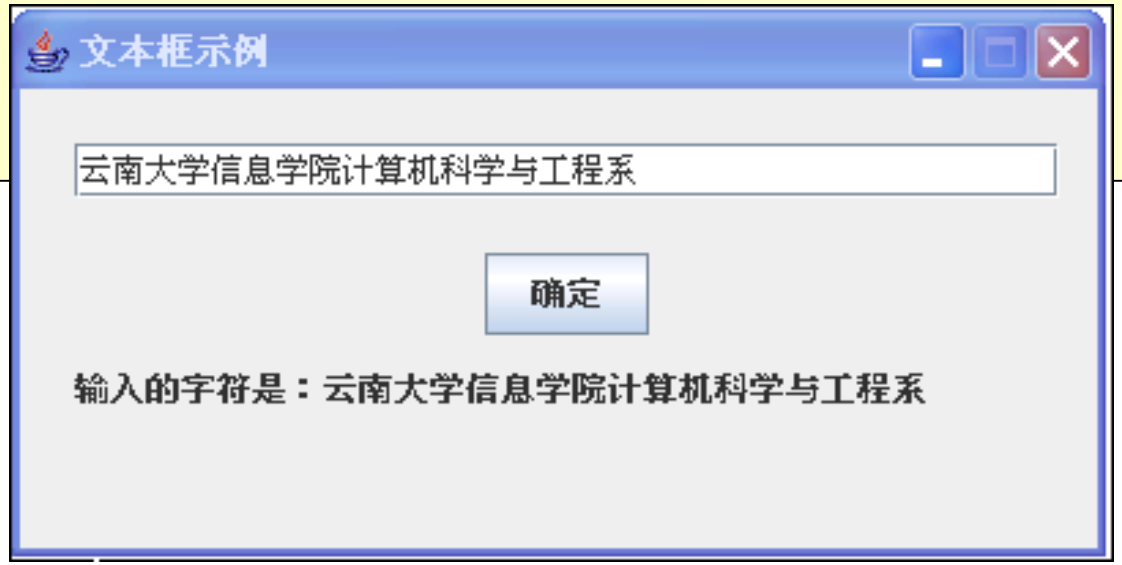
名称	说明
String getText()	返回文本编辑框JTextArea中的文本
void setText(String txt)	设定文本编辑框JTextArea中的显示文本为txt
void setToolTipText(String txt);	设定光标落在JTextArea上时显示的提示信息为txt
int getColumns()	返回文本编辑框JTextArea中的列数
void setColumns(int cols)	设定文本编辑框JTextArea中的列数为cols
int getRows()	返回文本编辑框JTextArea中的行数
void setRows(int rows)	设定文本编辑框JTextArea中的行数为rows
void append(String txt)	将字符串txt添加到文本编辑框JTextArea中后面
void insert(String txt,int pos)	将字符串txt插入到文本编辑框JTextArea中的第pos位置处
void replaceRange(String txt,int start,int end)	将文本编辑框JTextArea中的从start到end之间的字符用txt代替

9.3 基本组件——JTextArea例子

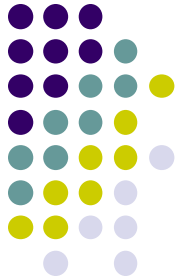
```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class JTextFieldDemo extends JFrame implements ActionListener {

    protected JTextField jtf=new JTextField(30);
    protected JButton button=new JButton("确定");
    protected JLabel label1=new JLabel("输入的字符是：");

    public JTextFieldDemo(){
        this.setLayout(null);
        this.setBounds(100,100,400,200);
        jtf.setBounds(20,20,360,20);
        button.setBounds(170,60,60,30);
        label1.setBounds(20,100,360,20);
        button.addActionListener(this);
        jtf.addActionListener(this);
        add(jtf); add(button);   add(label1);
        this.setTitle("文本框示例");
        this.setResizable(false);
        this.setVisible(true);
    }
```

9.3 基本组件——JTextArea例子

```
public void actionPerformed(ActionEvent e)
  {
     if(e.getSource()==button || e.getSource()==jtf){
        label1.setText("输入的字符是："+jtf.getText());
     }
  }

  public static void main(String[] args) {
     JTextFieldDemo frame = new JTextFieldDemo();
     frame.addWindowListener(new WindowAdapter() {
           public void windowClosing(WindowEvent e) {
              System.exit(0);
        }
     });
  }
}
```



9.3 基本组件——JTextArea



- JTextArea 是一个显示纯文本的多行区域。
- JTextArea 不管理滚动，可把它放置在 JScrollPane 的ViewPort中实现滚动，如：

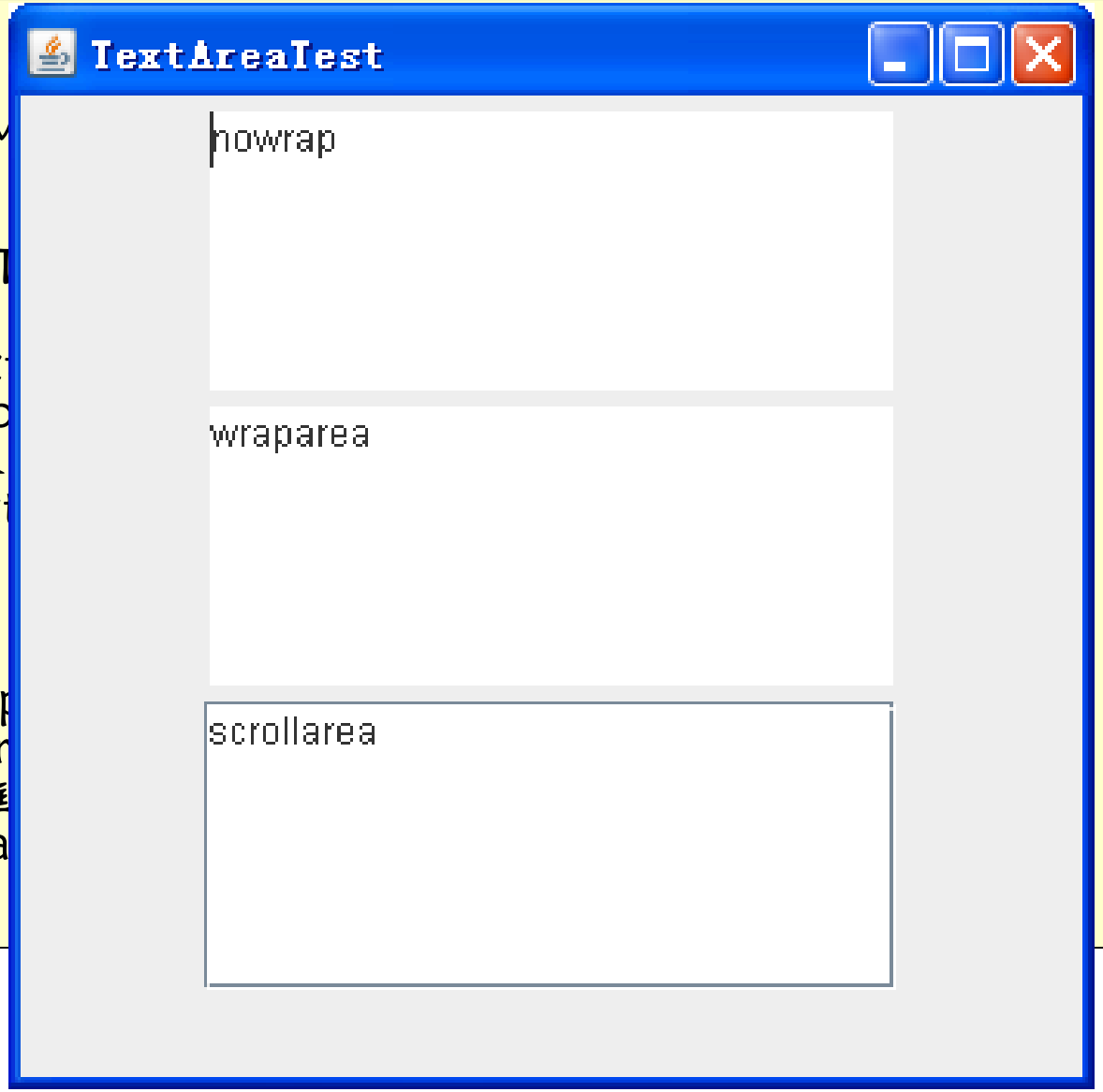
```
JTextArea textArea = new JTextArea();
JScrollPane area=new JScrollPane(textArea);
```

- TextArea 具有换行能力，JTextArea默认为不换行，需设置换行策略，如：

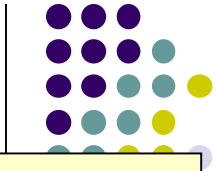
```
textArea.setLineWrap(true);
textArea.setWrapStyleWord(true);
```

9.3 基本组件——JTextArea

```
……
//建立容纳文本区的面
JPanel textPanel = new

//新建无回绕的文本区
noWrapArea = new JT
//新建有回绕的文本区
wrapArea = new JTex
wrapArea.setLineWrap
//新建带滚动条的文本
scrollArea = new JTex
//将文本区插入到滚动
 JScrollPane scrollPane

 textPanel.add(noWrap
 textPanel.add(wrapAr
 //将滚动窗格加入到框
 textPanel.add(scrollPa
 ……
```



9.3 基本组件——JTextArea



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

public class JTextAreaDemo extends JFrame implements ActionListener {
   private JPanel jp=new JPanel();
   private JButton[] jbArray=
       {new JButton("自动换行"),new JButton("不换行"),
        new JButton("单词边界"),new JButton("字符边界")};
   private JTextArea jta=new JTextArea();

   //将文本区作为被滚动控件创建滚动窗体
   private JScrollPane jsp=new JScrollPane(jta);
   public JTextAreaDemo(){
      jp.setLayout(null); //设置JPanel的布局管理器
      for(int i=0;i<4;i++){ //循环对按钮进行处理
         //设置按钮的大小和位置
         jbArray[i].setBounds(20+i*110,120,90,20);
         jp.add(jbArray[i]);
         jbArray[i].addActionListener(this);
      }
```

9.3 基本组件——JTextArea

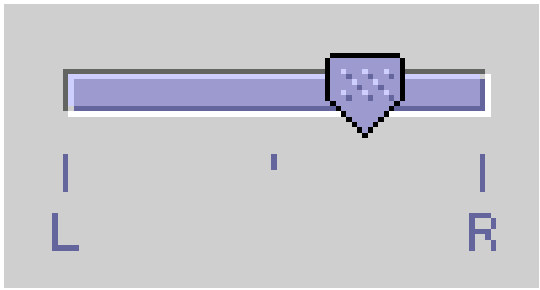
```
    jsp.setBounds(20,20,450,80); //设置JScrollPane的大小与位置
    jp.add(jsp);//将JScrollPane添加到JPanel容器中

    //设置JTextArea为自动换行
    jta.setLineWrap(true);
    for(int i=0;i<20;i++){//为JTextArea添加10条文本
       jta.append("["+i+"]Hello, this is an Example!");
    }

    //将JPanel容器添加进窗体
    this.add(jp);

    //设置窗体的标题、大小位置以及可见性
    this.setTitle("文本区与滚动条");
    this.setResizable(false);
    this.setBounds(100,100,500,180);
    this.setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);
    this.setVisible(true);
}
```

9.3 基本组件——JTextArea

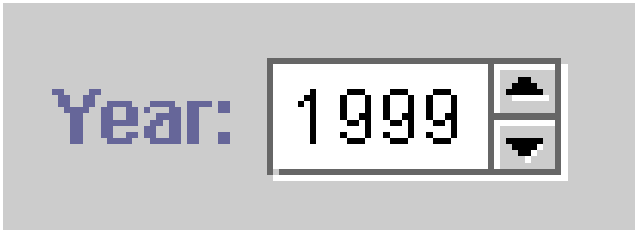
```
public void actionPerformed(ActionEvent e)
   {
      if(e.getSource()==jbArray[0]){//按下自动换行按钮
         jta.setLineWrap(true);
      }
      else if(e.getSource()==jbArray[1]) {//按下不换行按钮
         jta.setLineWrap(false);
      }
      else if(e.getSource()==jbArray[2]){//按下单词边界按钮
```

9.3 基本组件——连续数值选择

- 连续数值选择

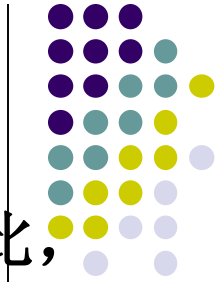


  - **JSlider**
    - 空间大
    - 好像电器用品上机械式的控制杆，可以设置它的最小、最大、初始刻度；还可以设置它的方向；还可以为其标上刻度或文本
    - 在JSlider上移动滑动杆，会产生ChangeEvent事件



  - **JSpinner**
    - 空间小
    - 类似于可编辑的JComboBox，是种复合组件，由三个部分组成：向上按钮、向下按钮和一个文本编辑区
    - 可以通过按钮来选择待选项，也可以直接在文本编辑区内输入
    - 和JComboBox不同的是，它的待选项不会显示出来

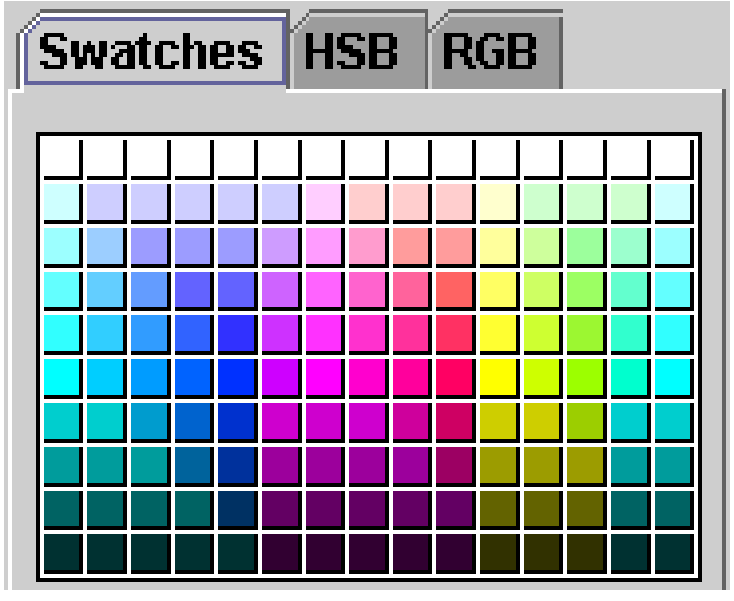
9.3 基本组件



- **Swing基本组件有很多种，与顶层容器和中间容器相比，基本组件用法都比较简单**
- 可将其分为三类
  - **显示不可编辑信息**
    - **JLabel、JProgressBar、JToolTip**
  - **有控制功能、可以用来输入信息**
    - **JButton、JCheckBox、JRadioButton、JComboBox、JList、JMenu、JSlider、JSpinner、JTextComponent**
  - **能提供格式化的信息并允许用户选择**
    - **JColorChooser、JFileChooser、JTable、JTree**

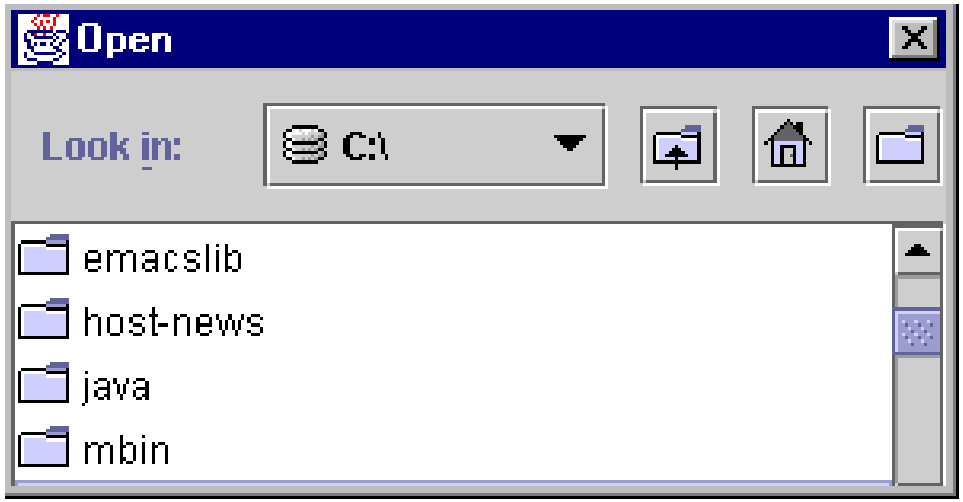
9.3 基本组件——JColorChooser

- JColorChooser颜色选择对话框
  - 可以让用户选择所需要的颜色
  - 通常使用这个类的静态方法showDialog()来输出标准的颜色选择对话框，其返回值就是选择的颜色
  - 也可以通过静态方法createDialog()方式输出个性化的颜色选择对话框，例如为其添加菜单、定义其事件处理程序，这个方法的返回值就是一个对话框



9.3 基本组件——JFileChooser

- JFileChooser文件选择对话框
  - 让用户选择一个已有的文件或者新建一个文件
  - 可以使用JFileChooser的showDialog、showOpenDialog或showSaveDialog()方法来打开文件对话框，但是它仅仅会返回用户选择的按钮（确认还是取消）和文件名（如果确认的话），接下来的要实现的例如存盘或者打开的功能还需要程序员自己编写
  - 这个类提供了专门的方法用于设置可选择的文件类型，还可以指定每类文件使用的类型图标

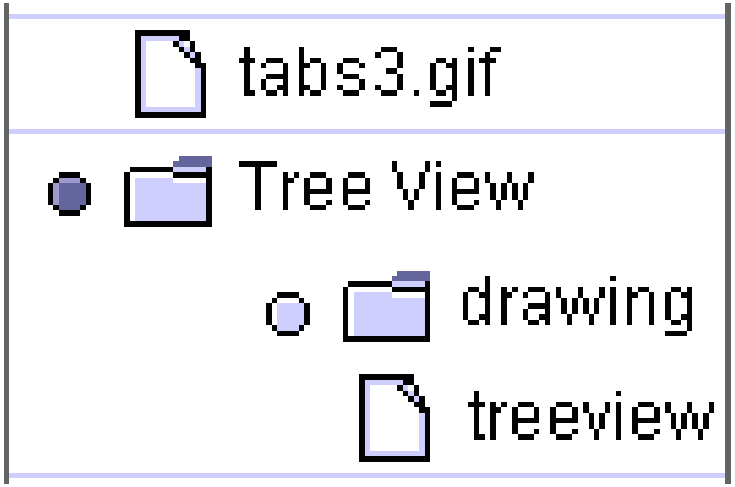


9.3 基本组件——JTable

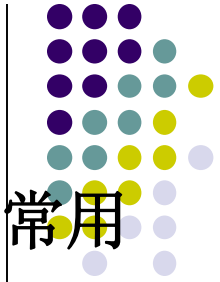
- JTable表格
  - 和其相关的还有一些接口和类，包括TableModel、AbstractTableModel、DefaultTableModel、SelectionModel、TableColumnModel
  - 可为表格设计显示外观（是否有滚动条、调整某一列宽其它列宽变化情形）、显示模式（根据数据类型的不同有不同的排列显示方式、为某一字段添加复选框JComboBox）、选择模式（单选、多选、连续选、任意选）
  - 在JTable的事件大致均针对表格内容的异操作处理，我们称为TableModelEvent事件。通过addTableModelListener方法为表格添加此种事件监听器

9.3 基本组件——JTree

- JTree
  - 用来产生树状结构来直观地表现层次关系，有根节点、树枝节点、树叶节点
  - 构造方法有多种，参数可以是一个Hashtable，也可以是TreeNode或TreeModel对象
  - 可以使用JComponent提供的putClientProperty方法来设置JTree外观，也可以使用TreeCellRenderer来个性化各类节点的显示样式

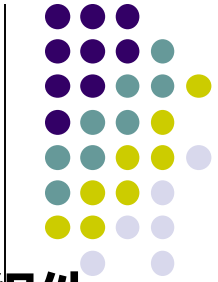


9.3 基本组件——JComponent类



- JComponent类是所有组件的父类，JComponent类的常用方法
- 1. 组件的颜色
  - public void setBackground(Color c) 设置组件的背景色。
  - public void setForeground(Color c) 设置组件的前景色。
  - public Color getBackground() 获取组件的背景色。
  - public Color getForeground() 获取组件的前景色。
- 2．组件的字体
  - public void setFont(Font f) 组件调用该方法设置组件上的字体。例如，文本组件调用该方法可以设置文本组件中的字体。
  - public Font getFont() 组件调用该方法获取组件上的字体。

9.3 基本组件——JComponent类



3．组件的激活与可见性

- **public void setEnabled(boolean b)** 组件调用该方法可以设置组件是否可被激活，当参数b取值true时，组件可以被激活，当参数b取值false 时，组件不可激活。默认情况下，组件是可以被激活的。
- **public void setVisible(boolean)** 设置组件在该容器中的可见性，当参数b取值true时，组件在容器中可见，当参数b取值false 时，组件在容器中不可见。除了Window型组件外，其它类型组件默认是可见的。

4．组件上的光标

- **public void setCursor(Cursor c)** 设置鼠标指向组件时的光标形状。可以使用 Cursor类的类方法直接获得一个光标对象，例如：

```
Cursor c=Cursor.getPredefinedCursor(Cursor.HAND_CURSOR);
```

9.3 基本组件——JComponent类

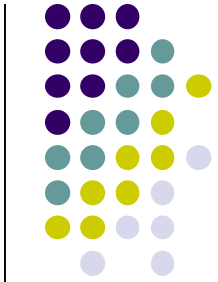
5、组件的大小与位置

- public void setSize(int width,int height) 组件调用该方法设置组件的大小。
- public void setLocation(int x,int y) 组件调用该方法设置组件在容器中的位置，包含该组件的容器都有默认的坐标系，容器的坐标系的左上角的坐标是(0,0)，参数x, y指定该组件的左上角在容器的坐标系中的坐标，即组件距容器的左边界 x个像素，距容器的上边界 y 个像素。
- public Dimension getSize() 组件调用该方法返回一个Dimension对象的引用，该对像实体中含有名字是width 和height的成员变量，方法返回的Dimension对象的width的值就是组件的宽度、height的值就是当前组件的高度。
- public Point getLocation() 组件调用该方法返回一个Point对象的引用，该对像实体中含有名字是x 和y的成员变量，方法返回的Point对象的x，y的值就是组件的左上角在容器的坐标系中的x坐标和y坐标。
- public void setBounds(int x,int y,int width,int height) 组件调用该方法设置组件在容器中的位置和组件的大小。该方法相当于setSize方法和setLocation方法的组合。
- public Rectangle getBounds() 组件调用该方法返回一个Rectangle对象的引用，该对象实体中含有名字是x、y、width 和height的成员变量，方法返回的Rectangle对象的x,y的值就是组件的左上角在容器的坐标系中的x坐标和y坐标，width和height的值就是当前组件的宽度和高度。

基于Swing的Java GUI设计思路

- 基本的java程序的GUI设计工具。主要包括下述几个概念：
  - 组件—Component
  - 容器—Container
  - 布局管理器—LayoutManager
  - 事件处理
- 在Java中，开发一个GUI程序，通常需要以下几步：
  - 构建一个顶层容器；通常是JFrame或JApplet
  - 构建若干个组件，组件可以是其它容器；
  - 设定容器的布局管理器；用容器的add方法将这些组件加入到这个容器中；
  - 设置组件事件；并将组件事件与代码关联。

第九章 Swing图形用户界面

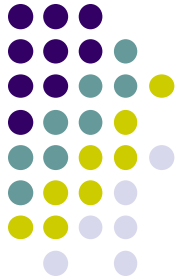


- 9.1 概述
- 9.2 容器组件
- 9.3 基本组件
- 9.4 布局管理器
- 9.5 事件处理模型
- 9.6 本章小结

基于Swing的Java GUI设计思路

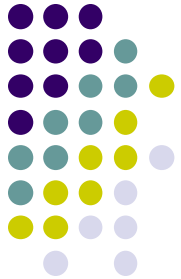
- 基本的java程序的GUI设计工具。主要包括下述几个概念：
  - 组件—Component
  - 容器—Container
  - 布局管理器—LayoutManager
  - 事件处理
- 在Java中，开发一个GUI程序，通常需要以下几步：
  - 构建一个顶层容器；通常是JFrame或JApplet
  - 构建若干个组件，组件可以是其它容器；
  - 设定容器的布局管理器；用容器的add方法将这些组件加入到这个容器中；
  - 设置组件事件；并将组件事件与代码关联。

9.4 布局管理器



- 决定组件在界面中所处的位置和大小
- 如何将下级组件有秩序地摆在上一级容器中
- 六种布局管理器(Interface LayoutManager)
  - 两种简单布局
    - java.awt.FlowLayout　顺序布局
    - java.awt.GridLayout　网格布局
  - 两种特定用途布局
    - java.awt.BorderLayout　边界布局
    - java.awt.CardLayout　卡片布局
  - 两种灵活布局
    - java.awt.GridBagLayout　复杂网格布局
    - javax.swing.BoxLayout
    - javax.swing.SpringLayout

9.4 布局管理器

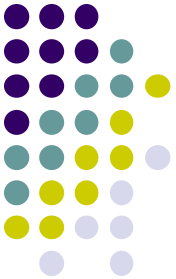


- 布局管理器
  - **使用方法是通过调用容器对象的setLayout方法，并以某种布局管理器对象为参数，例如：**

    ```
    Container contentPane = frame.getContentPane();
    contentPane.setLayout(new FlowLayout());
    ```

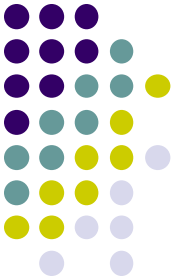
  - **使用布局管理器可以更容易地进行布局，而且当改变窗口大小时，它还会自动更新版面来配合窗口的大小，不需要担心版面会因此混乱**

9.4 布局管理器 —— FlowLayout



- 流式布局FlowLayout
  - 是一种最基本的布局管理器
  - 是 java.awt.Applet、java.awt.Panel 和 javax.swing.JPanel的默认布局方式
  - 在容器中，从左到右依次放置GUI组件，当组件排到容器一行的末尾时，则从下一行开始接着排列组件
  - 它将根据容器的大小随时调整它里面的组件的大小，包括高度和宽度，这个管理器不会约束组件的大小，而是允许他们获得自己的最佳大小。
  - 每行组件的对齐方式可以是:
    - 左对齐、中间(默认对齐方式)和右对齐

9.4 布局管理器 —— FlowLayout



- 构造方法
  - **public FlowLayout()**
  - **public FlowLayout(int align)**
  - **public FlowLayout(int alignment, int hgap, int vgap)**

参数说明：

align：对齐方式 （默认为FlowLayout.CENTER）

FlowLayout.LEFT 左对齐

FlowLayout.RIGHT 右对齐

FlowLayout.CENTER 居中对齐

hgap, vgap: 组件的水平间距和垂直间距

单位为象素，默认值为5

加入组件的方法：

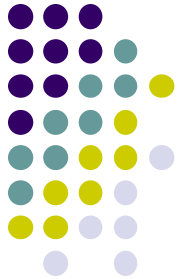
add(<组件名称>);

9.4 布局管理器 —— FlowLayout例子

```
import java.awt.*;
import javax.swing.*;
public class FlowLayoutDemo extends JFrame {
   public FlowLayoutDemo() {
      Container c=getContentPane();
      c.setLayout(new FlowLayout());
      c.add(new JLabel("Buttons:"));
      c.add(new JButton("Button 1"));
      c.add(new JButton("2"));
      c.add(new JButton("Button 3"));
      c.add(new JButton("Long-Named Button 4"));
      c.add(new JButton("Button 5"));
   }
   public static void main(String args[]) {
      FlowLayoutDemo window = new FlowLayoutDemo();
```

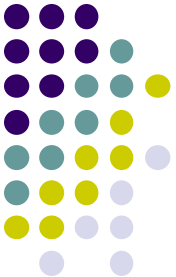

```
}
```

9.4 布局管理器 —— BorderLayout



- 是容器JFrame和JApplet的默认布局方式
- 将容器分成五个区域，
  - NORTH (顶部)
  - SOUTH (底部)
  - WEST (左侧)
  - EAST (右侧)
  - CENTER (中间)
- 每个区域最多只能1个组件，所以最多只能容纳5个组件，否则要采取其它布局方式。

9.4 布局管理器 —— BorderLayout



BorderLayout的构造方法：

(1) BorderLayout();

(2) BorderLayout(int hgap, int vgap);

参数说明：

hgap, vgap: 组件的水平间距和垂直间距

单位为象素，默认值为0

加入组件的方法：

add(<组件名称>,<位置参数>);

add(<位置参数>,<组件名称>);

其中<位置参数>是BorderLayout.CENTER, BorderLayout.NORTH, BorderLayout.SOUTH, BorderLayout.WEST, BorderLayout.EAST等5个静态变量

或直接写"Center", "North", "South", "West", "East"等5个字符串

9.4 布局管理器 —— BorderLayout例子

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
class BorderLayoutDemo extends JFrame implements ActionListener{
    String names[]={"北","南","东","西","中"};
   Button but[]=new Button[names.length];
    public BorderLayoutDemo() {
            setTitle("BorderLayoutDemo");
            setSize(400,300);
            setLayout(new BorderLayout(10,10));
             for (int i = 0; i<names.length; i++){
                but[i]=new Button(names[i]);
                but[i].addActionListener(this);
             }
             add("North", but[0]);
             add(but[1],"South");
             add(BorderLayout.EAST,  but[2]);
             add(but[3], BorderLayout.WEST);
             add("Center", but[4]);
            setVisible(true);
            setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);
    }
```

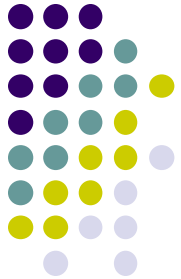
9.4 布局管理器 —— BorderLayout例子

```
    public void actionPerformed(ActionEvent e) {
            for (int i = 0; i<but.length; i++){
                if(e.getSource() == but[i])
                    but[i].setVisible(false);
                else
                    bu
            }
    }

    public static voi
            BorderLay
    }
}
```

9.4 布局管理器 —— BorderLayout例子

```
import java.awt.*;
import javax.swing.*;
public class BorderLayoutDemo extends JFrame {
   public BorderLayoutDemo() {
    Container c=getContentPane();
      c.setLayout(new BorderLayout(5,5));
      c.add(new JButton("North"), "North");
      c.add( new JButton("South"),"South");
      //c.add("South",new Button("South"));
      //c.add(new Button(" South "), BorderLayout.SOUTH);
      c.add( new JButton("East"),"East");
      c.add( new JButton("West"),"West");
      c.add( new JButton("Center"),"Cen
   }
   public static void main(String args[])
      BorderLayoutDemo window = ne
      window.setTitle("BorderLayout D
      window.pack();
      window.setVisible(true);
   }
}
```

9.4 布局管理器 ——BoxLayout



- 盒式布局管理器**BoxLayout**
  - **允许多个组件在容器中沿水平方向或竖直方向排列**
  - **容器的大小发生变化时，组件占用的空间不会发生变化**
- 构造方法：

BoxLayout(Container target, int axis)

创建一个将沿给定轴放置组件的布局管理器。

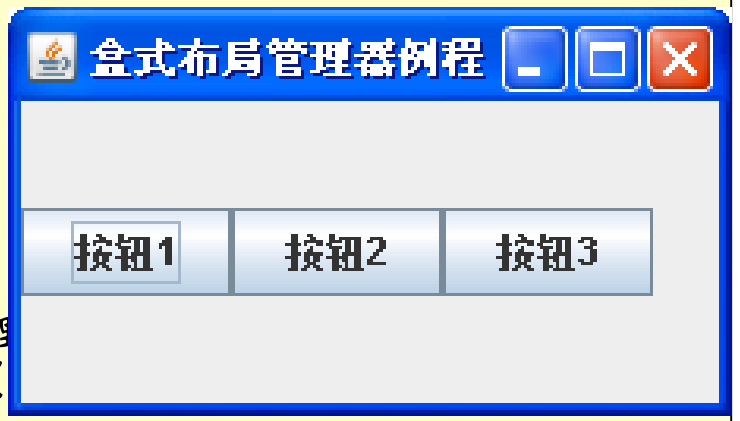
- **轴的方向：**

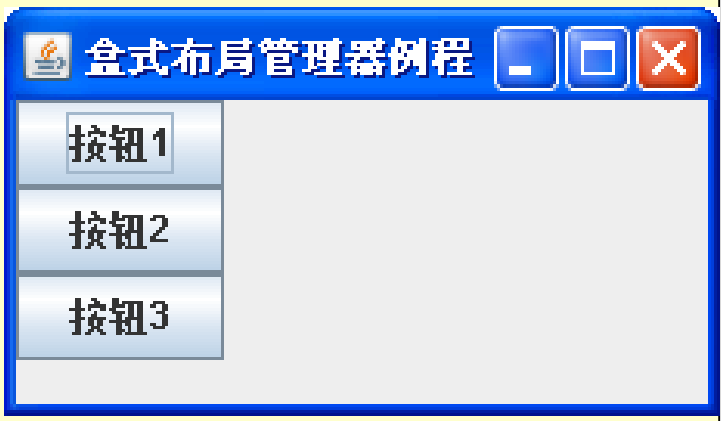
  BoxLayout.X_AXIS: 指定组件应该从左到右放置

  BoxLayout.Y_AXIS: 指定组件应该从上到下放置

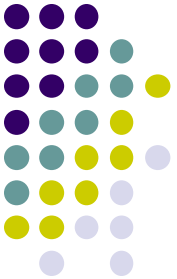
9.4 布局管理器 ——BoxLayout例子

```
import java.awt.Container;
import javax.swing.BoxLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
public class BoxLayoutDemo{
   public static void main(String args[ ]){
      JFrame app = new JFrame("盒式布局管理
      app.setDefaultCloseOperation(JFrame.EX
      app.setSize( 220, 130 );
      Container c = app.getContentPane( );
      c.setLayout( new BoxLayout( c, BoxLayout.X_AXIS ) );
      String s;
      JButton b;
      for (int i=0; i<3; i++){
         s = "按钮" + (i+1);
         b = new JButton( s );
         c.add( b );
      }
      app.setVisible( true );
   }
}
```



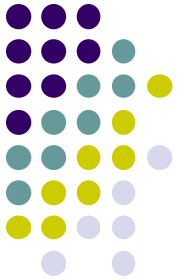


9.4 布局管理器 ——GridLayout



- 布局管理器GridLayout按行与列将容器等分成网格
  - **每个组件占用具有相同宽度和高度的网格**
  - **添加组件占用网格的顺序: 从上到下，从左到右**
  - **当一行满了，则继续到下一行，仍然是从左到右**

9.4 布局管理器 ——GridLayout



- GridLayout类的构造方法
  - GridLayout()
    创建默认值的网格布局，即每个组件占据一行一列。
  - GridLayout(int rows, int cols)
    创建具有指定行数和列数的网格布局
  - GridLayout(int rows, int cols, int hgap, int vgap)
    创建具有指定行数和列数的网格布局。

9.4 布局管理器 ——GridLayout例子

```
import java.awt.*;
import javax.swing.*;
public class GridLayoutDemo {
    JFrame f;
    public GridLayoutDemo(String str){
        f=new JFrame(str);
        Container c=f.getContentPane();
        c.setLayout(new GridLayout(3,2));
        for(int i=1;i<=6;i++){
              c.add(new JButton(i+""));
        }
        f.pack( ) ;
        f.setVisible(true);
    }
    public static void main(String args[]){
      new GridLayoutDemo("GridLayout Demo");
   }
}
```

9.4 布局管理器 ——GridLayout例子

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class Test{
     public static void main(String args[]){
       Calculator myCal = new Calculator();
    }
}

class Calculator extends JFrame implements ActionListener, WindowListener{
  static String names[]=
  {"7", "8", "9", "/", "CE",
   "4", "5", "6", "*", "C",
   "1", "2", "3", "-", " 退格",
   "0", "+/-", ".", "+", "="};
  String resultStr;
  JButton buts[] = new JButton[names.length];
  JPanel pa1;
  JTextField text;
```

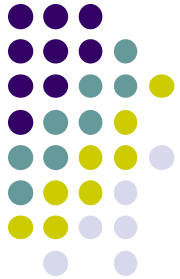
9.4 布局管理器 ——GridLayout例子

```
public Calculator() {
        super("我的计算器");
        this. resultStr=new String();
        this.setSize(350,300);
         this.setLayout(new BorderLayout());
         this.pa1=new JPanel();
         this.text=new JTextField("0",12);
         pa1.setLayout(new GridLayout(4, 5, 5, 5));
        this. setFont(new Font("Arial",Font.BOLD,12));
       for (int i=0;i<buts.length;i++){
         buts[i]=new JButton(names[i]);
         buts[i].addActionListener(this);
         pa1.add(buts[i]);
      }
      this.add("North",text);
      this.add("Center",pa1);
      this.addWindowListener(this);
     this. setVisible(true);
  }
```

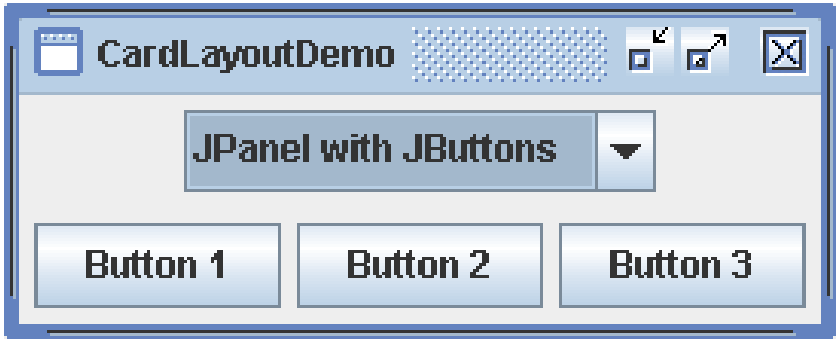
9.4 布局管理器 ——GridLayout例子

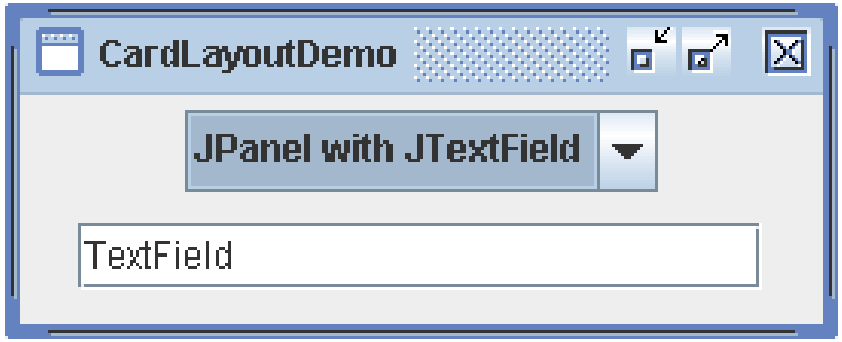
```
public void actionPerformed(ActionEvent e) {
    String str1=new String();
    for (int i = 0; i<buts.length; i++){
            if(e.getSource()==buts[i]){  str1 = buts[i].getLabel();
            }
    }
    resultStr+=str1;
    text.setText(resultStr);
 }
    public void windowOpened(Window
    public void windowClosing(Windowl
            System.exit(0);
    }
    public void windowClosed(WindowE
    public void windowIconified(Windov
    public void windowDeiconified(Winc
    public void windowActivated(Windo
    public void windowDeactivated(Win
}
```

9.4 布局管理器 ——CardLayout

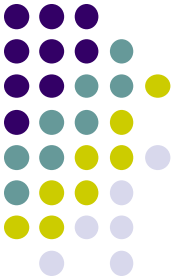


- **CardLayout**的布局方式有点象码“扑克牌”
- 一个组件压在另一个组件的上面，所以每次一般只能看到一个组件。。它经常由一个复选框控制这个区域显示哪一组组件，可通过组合框像选择卡片一样选择某一种布局。
- 这个被显示的组件占据所有的容器区域。





9.4 布局管理器 ——CardLayout



CardLayout的构造方法：

CardLayout();

加入组件的方法：

add(<组件代号>,<组件名称>);

<组件代号>为一个字符串

常用方法：

show(<容器名称>,<组件名称>);　//显示组件

first(<容器名称>);　//显示容器中的第一个组件

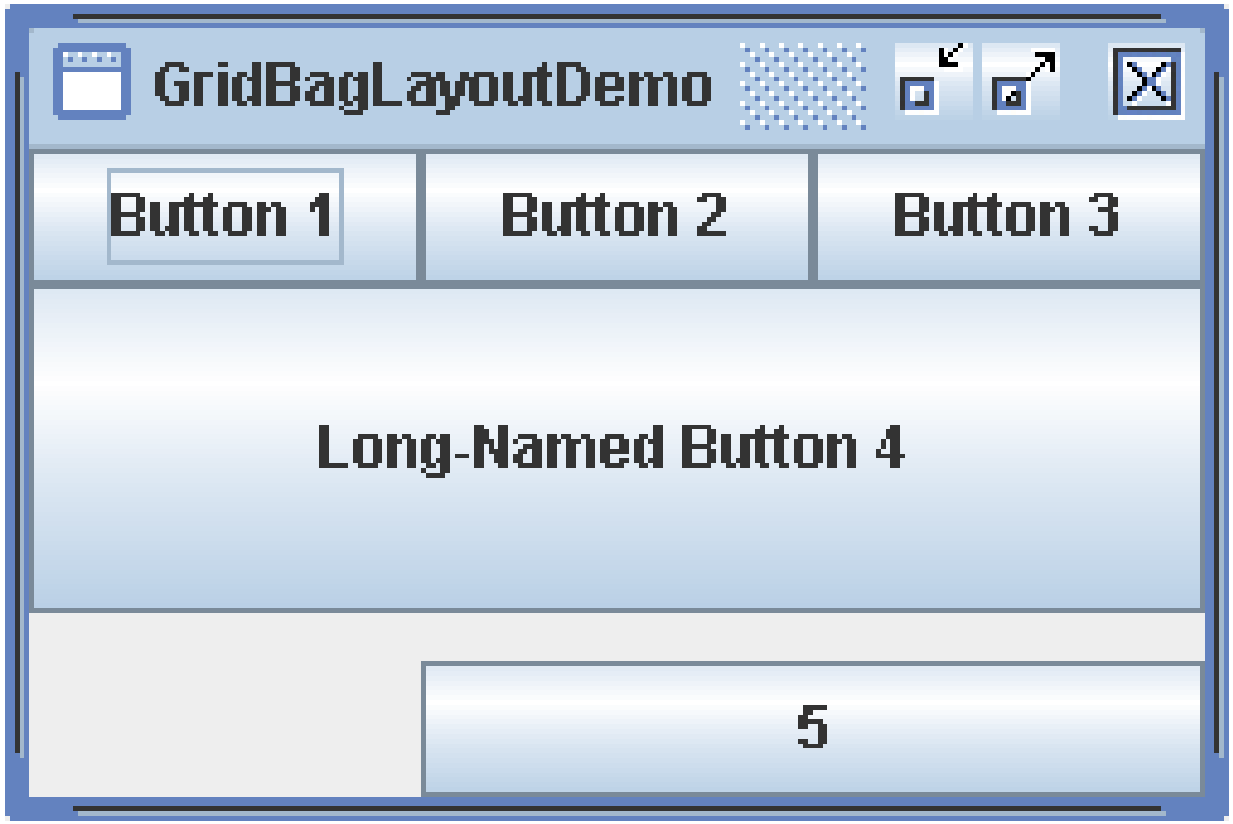
last(<容器名称>);　//显示容器中的最后一个组件

next(<容器名称>);　//显示容器中的下一个组件

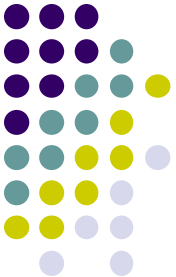
9.4 布局管理器 —— GridBagLayout

- 在网格布局(GridLayout)中，网格大小是相同的，因此放置组件的大小也是相同的，分布呆板，不够灵活。于是，引入了复杂网格布局。
- 复杂网格布局（GridBagLayout）可以给每个组件指定所包含的网格个数，可以任意顺序将组件加入到容器的任何位置。
- GridBagLayout是AWT包中提供的最灵活、最复杂的布局管理器。
- GridBagLayout将组件以多行多列放置，允许指定的组件跨多行或多列。
- 每个 GridBagLayout 对象保留一个动态的矩形单元网格，每个组件占用一个或多个单元，称为它的显示区域。
- 每个由一个网格元包布局管理的组件都与一个GridBagConstraints的实例相关，它指定了组件在它的显示区域是如何放置的。

9.4 布局管理器 —— GridBagLayout



9.4 布局管理器 ——空布局

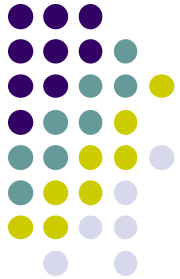


- **空布局**

  setLayout(null);

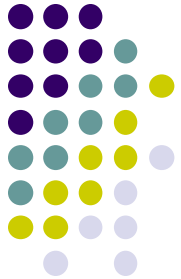
- **直接用setBounds(int, int, int, int)指定各个组件的位置和大小，前两个参数指定位置，窗口左上角是坐标原点，后两个参数是组件的大小，即宽和高。**

9.4 布局管理器



- 使用布局管理器的基本规则
  - **若组件尽量充满容器空间，可以考虑使用BorderLayout和GridBagLayout。**
  - **若用户需要在紧凑的一行中以组件的自然尺寸显示较少的组件，用户可以考虑用面板容纳组件，并使用面板的默认布局管理器FlowLayout。**
  - **若用户需要在多行或多列中显示一些同样尺寸的组件，GridLayout最适合此情况。**
  - **若界面较为复杂，可先使用面板来容纳组件，然后选用适当的布局管理器。**

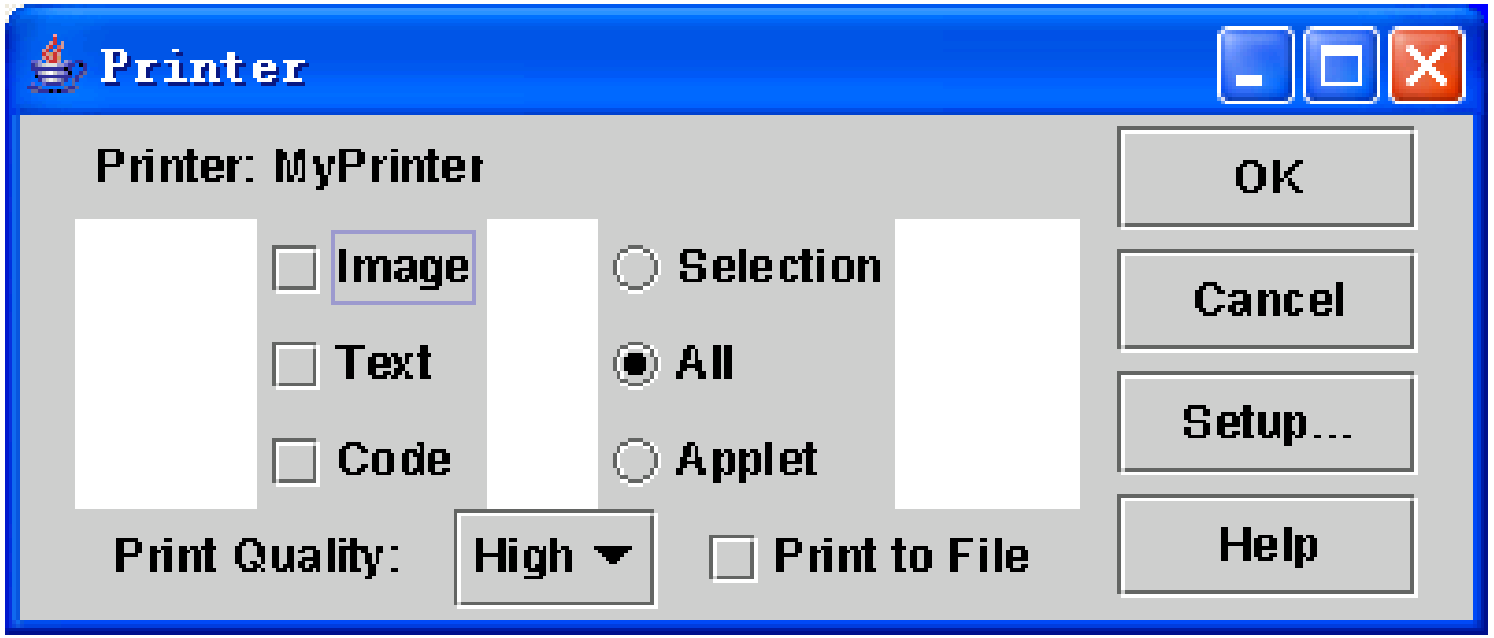
9.4 布局管理器



- 布局管理器嵌套
- 实际上是容器的嵌套，被嵌套的容器可以具有不同的布局管理器
- 在嵌套的布局管理器中
  - **JPanel 通常起到了 “桥” 的作用**

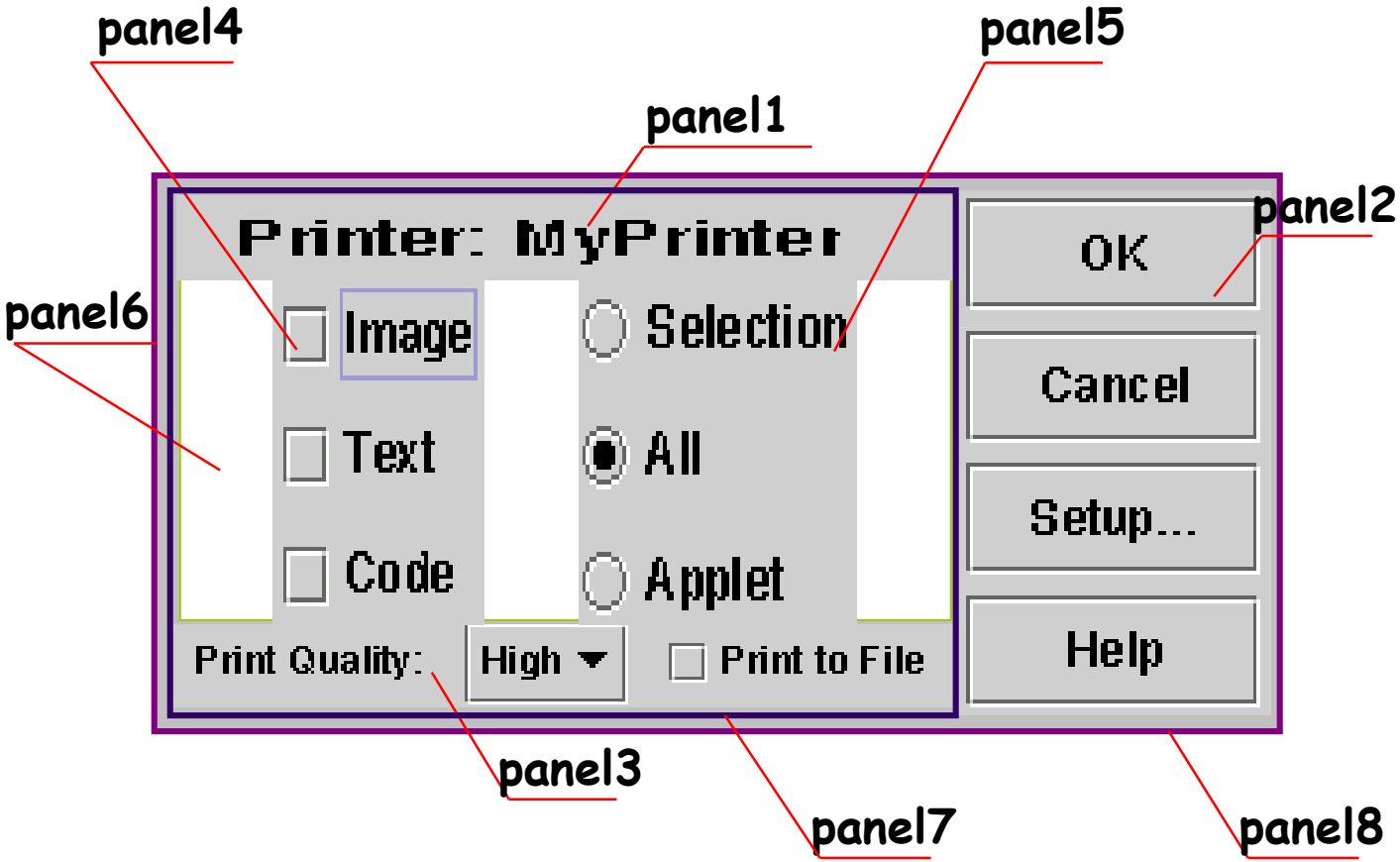
9.4 布局管理器

- 复杂GUI的布局



9.4 布局管理器

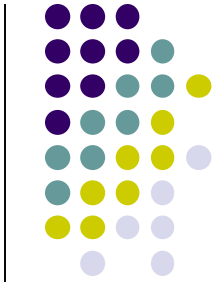
- 实现GUI的步骤分析



基于Swing的Java GUI设计思路

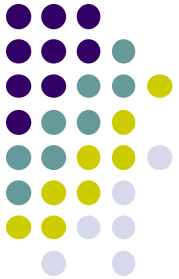
- 基本的java程序的GUI设计工具。主要包括下述几个概念：
  - 组件—Component
  - 容器—Container
  - 布局管理器—LayoutManager
  - 事件处理
- 在Java中，开发一个GUI程序，通常需要以下几步：
  - 构建一个顶层容器；通常是JFrame或JApplet
  - 构建若干个组件，组件可以是其它容器；
  - 设定容器的布局管理器；用容器的add方法将这些组件加入到这个容器中；
  - 设置组件事件；并将组件事件与代码关联。

第九章 Swing图形用户界面



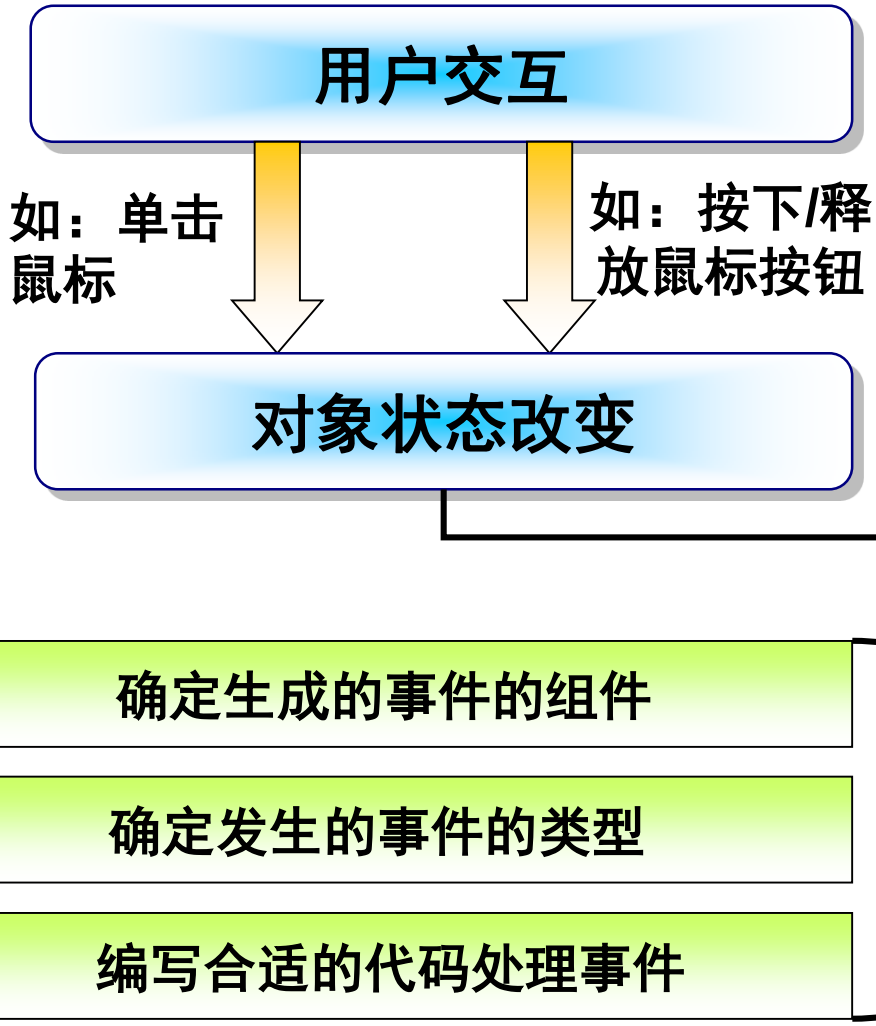
- 9.1 概述
- 9.2 容器组件
- 9.3 基本组件
- 9.4 布局管理器
- 9.5 事件处理
- 9.6 本章小结

9.5 事件处理



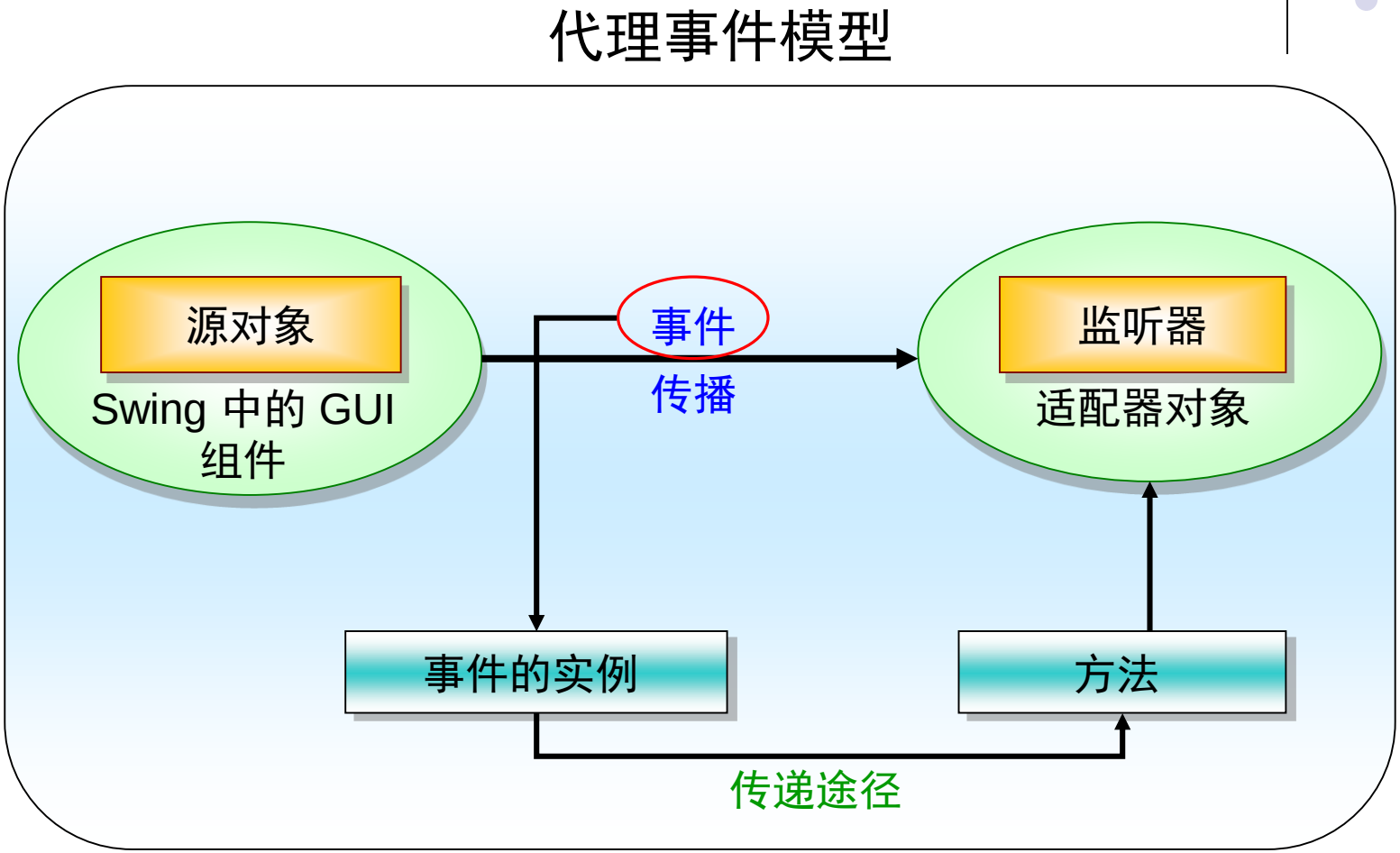
- 界面设计 (静态)
- 界面动起来
  - 通过事件触发对象的响应机制
- 一些常见的事件包括：
  - 移动鼠标
  - 单双击鼠标各个按钮
  - 单击按钮
  - 在文本字段输入
  - 在菜单中选择菜单项
  - 在组合框中选择、单选和多选
  - 拖动滚动条
  - 关闭窗口
  - ……
- Swing通过事件对象来包装事件，程序可以通过事件对象获得事件的有关信息

事件处理



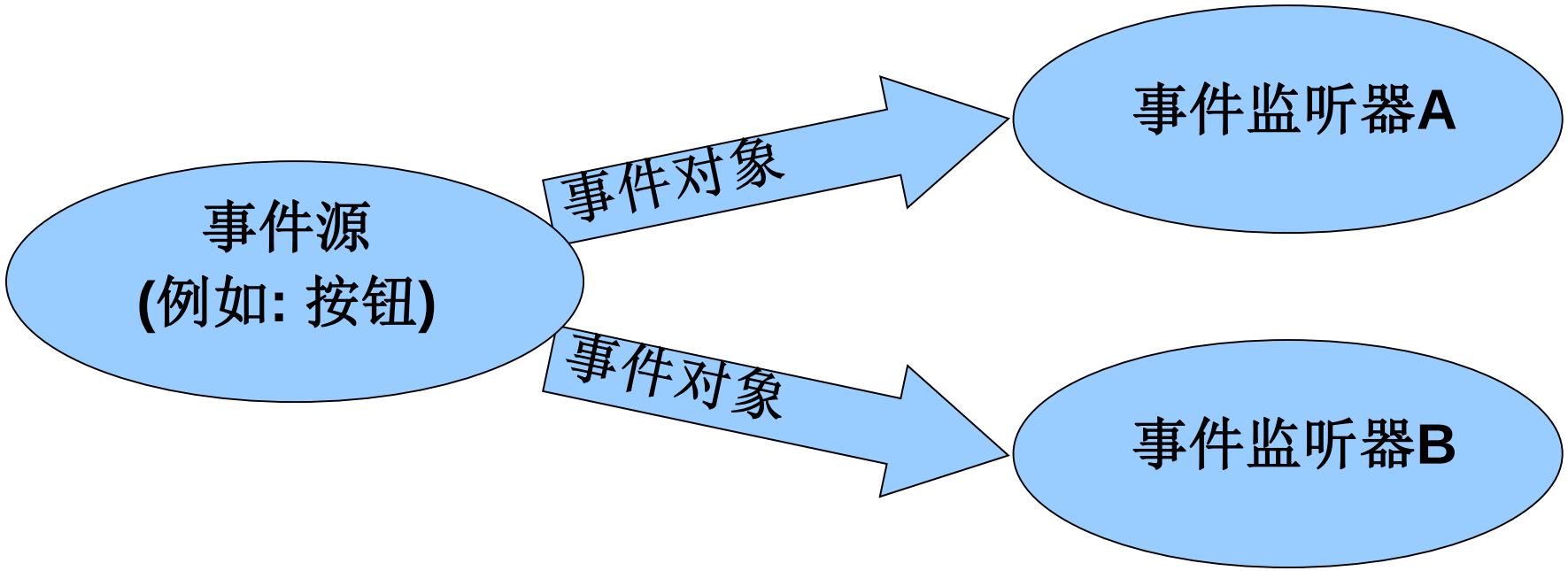
- 事件作用于对象。
- 对象识别事件并做出响应，不同的对象能识别的事件不全相同。
- 可以针对相应的事件编写相应的代码，在事件发生时，程序执行事件里的代码。

事件处理模型



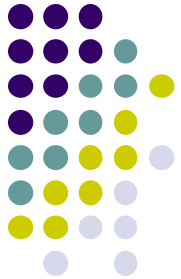
事件处理机制

- 主要涉及三种对象
  - **事件源(An event source)**
  - **事件对象(An event object)**
  - **事件监听器(event listener(s))**



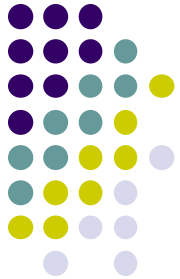
164

9.5 事件处理



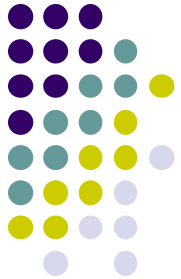
- 事件处理机制
  - **事件源**
    - 与用户进行交互的GUI组件，表示事件来自于哪个组件或对象
    - 比如要对按钮被按下这个事件编写处理程序，按钮就是事件源
  - **事件监听器**
    - 负责监听事件并做出响应
    - 一旦它监视到事件发生，就会自动调用相应的事件处理程序作出响应
  - **事件对象**
    - 封装了有关已发生的事件的信息
    - 例如按钮被按下就是一个要被处理的事件，当用户按下按钮时，就会产生一个事件对象。事件对象中包含事件的相关信息和事件源

9.5 事件处理



- 程序员应完成的任务
  - 实现事件监听的接口类
  - 为事件源注册一个事件监听器
  - 实现事件处理方法

9.5 事件处理



包含事件处理的程序应该包括以下四部分内容：

1、引入系统事件类包，如**import java.awt.event.***。

2、在事件处理类的声明中指定要实现的监听器名，如：

```
public class MyClass implements ActionListener { …}
```

3、注册事件源对象的事件监听者，如

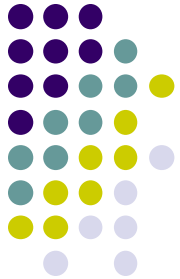
```
btn.addActionListener (this)。
```

4、实现监听器中的接口

如实现按钮事件监听接口**ActionListener** ：

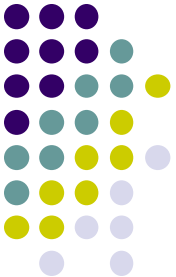
```
public void actionPerformed(ActionEvent e) {
 ...//响应某个动作的代码...
}
```

9.5 事件处理——事件分类与监听器接口



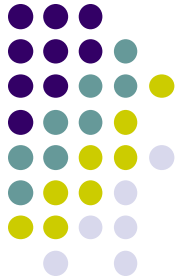
- 事件源
  - **提供注册监听器或取消监听器的方法**
  - **维护一个已注册的监听器列表，如有事件发生，就会通知每个已注册的监听器**
- 一个事件源可以注册多个事件监听器，每个监听器又可以对多种事件进行相应，例如一个**JFrame事件源**上可以注册
  - **窗口事件监听器，响应**
    - **窗口关闭**
    - **窗口最大化**
    - **窗口最小化**
  - **鼠标事件监听器，响应**
    - **鼠标点击**
    - **鼠标移动**

9.5 事件处理——事件分类与监听器接口



- 事件监听器
  - **是一个对象，通过事件源的add×××Listener方法被注册到某个事件源上**
  - **不同的Swing组件可以注册不同的事件监听器**
  - **一个事件监听器中可以包含有对多种具体事件的专用处理方法**
    - **例如用于处理鼠标事件的监听器接口MouseListener中就包含有对应于鼠标压下、放开、进入、离开、敲击五种事件的相应方法mousePressed、mouseReleased、mouseEntered、mouseExited、mouseClicked，这五种方法都需要一个事件对象作为参数**

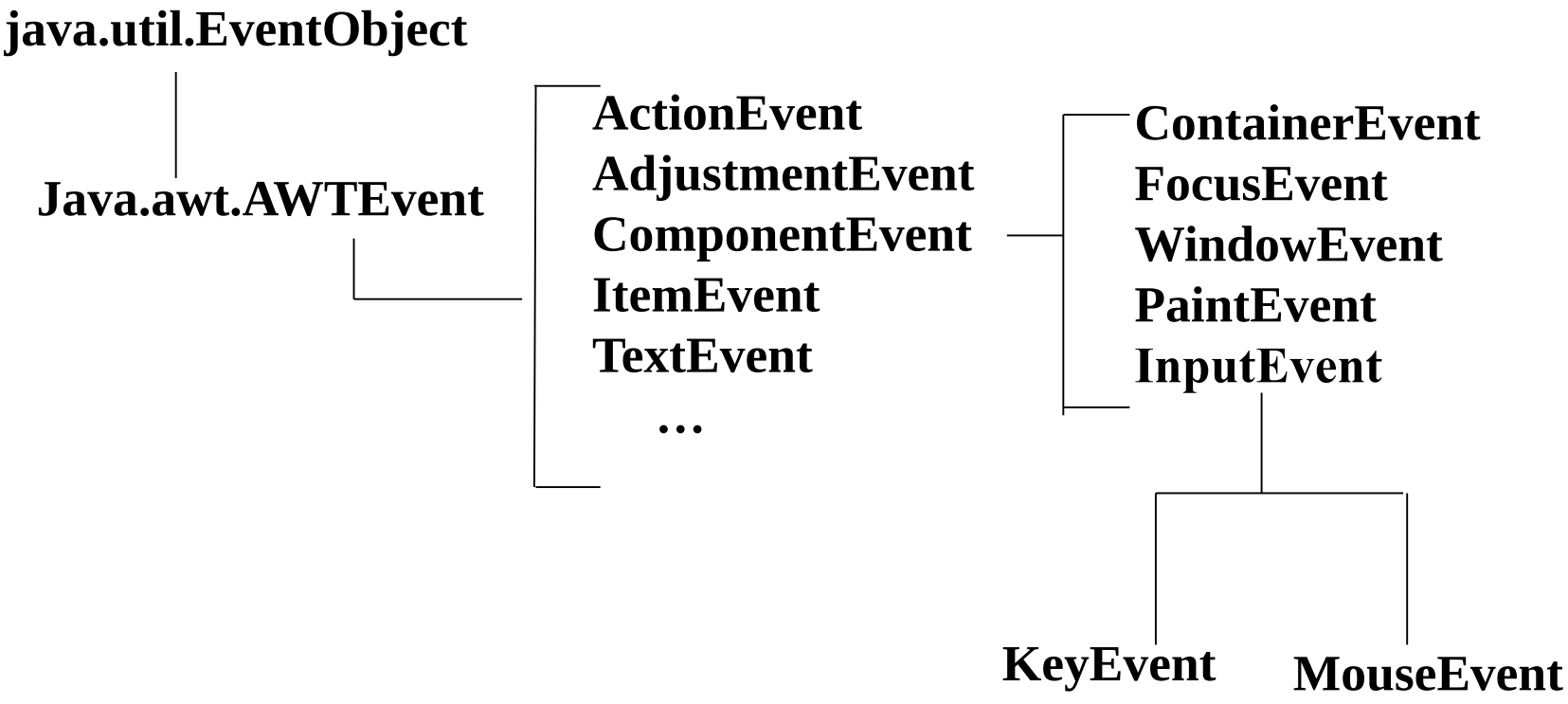
9.5 事件处理——事件分类与监听器接口



- 通常我们用到的事件对象有
  - **ActionEvent**
    - **发生在按下按钮、选择了一个项目、在文本框中按下回车键**
  - **ItemEvent**
    - **发生在具有多个选项的组件上，如JCheckBox、JComboBox**
  - **ChangeEvent**
    - **用在可设定数值的拖曳杆上，例如JSlider、JProgressBar等**
  - **WindowEvent**
    - **用在处理窗口的操作**
  - **MouseEvent**
    - **用于鼠标的操作**

9.5 事件处理——事件分类与监听器接口

- 事件分类



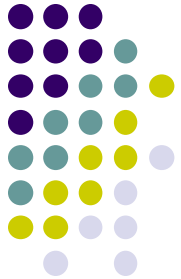
9.5 事件处理——事件分类与监听器接口

- 监听器接口
  - **对于每种类型的事件，都定义了相应的事件处理接口；**
  - **XXXEvent对应的事件处理接口通常为XXXListener。**

java.util.EventListener

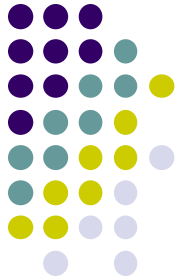
- **ActionListener**
- **ItemListener**
- **WindowListener**
- **KeyListener**
- **MouseListener**
- **MouseMotionListener**
- **…**

9.5 事件处理



- 事件监听器接口和事件监听器适配器类
- 事件监听器接口
  - 例如MouseListener是一个接口，为了在程序中创建一个鼠标事件监听器的对象，我们需要实现其所有五个方法，在方法体中，我们可以通过鼠标事件对象传递过来的信息（例如点击的次数，坐标），实现各种处理功能
- 事件监听器适配器类
  - 有时我们并不需要对所有事件进行处理，为此Swing提供了一些适配器类×××Adapter，这些类含有所有×××Listener中方法的默认实现（就是什么也不做），因此我们就只需编写那些需要进行处理的事件的方法。例如，如果只想对鼠标敲击事件进行处理，如果使用MouseAdapter类，则只需要重写mouseClicked方法就可以了

9.5 事件处理



三种实现事件处理的方法

- 实现事件监听器接口
  - 这种方法需要实现接口中所有的方法，对我们不需要进行处理的事件方法，也要列出来，其方法体使用一对空的花括号
- 继承事件监听器适配器类
  - 只需要重写我们感兴趣的事件
- 使用匿名内部类
  - 特别适用于已经继承了某个父类（例如主类必须继承JFrame类），则根据Java语法规则，就不能再继承适配器类的情况，而且使用这种方法程序看起来会比较清楚明了

9.5 事件处理——例子

- 当鼠标在窗口中点击时，在窗口标题栏中显示点击位置坐标。

```
import java.awt.event.*; //载入MouseListener类所在的包
import javax.swing.*;  //载入JFrame所在的包
public class ActionDemo implements MouseListener{
   JFrame f;
   public ActionDemo() {
      f=new JFrame(); //新建一窗口
      f.setSize(400,300);
      f.setVisible(true);
      f.addMouseListener(this); //为窗口增加鼠标事件监听器
      f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   }
    public void mousePressed(MouseEvent e){}
    public void mouseReleased(MouseEvent e){}
    public void mouseEntered(MouseEvent e){}
    public void mouseExited(MouseEvent e){}
    public void mouseClicked(MouseEvent e){
      f.setTitle("点击坐标为 ("+e.getX()+", "+e.getY()+")");
    }
   public static void main(String[] args){ new ActionDemo();}
}
```

方法1：采用同一个类中实现事件接口的方法

9.5 事件处理——例子

- 当鼠标在窗口中点击时，在窗口标题栏中显示点击位置坐标。

```
import java.awt.event.*; //载入MouseAdapter所在的包
import javax.swing.*;
public class ActionDemo extends MouseAdapter{
   JFrame f;
   public ActionDemo() {
      f=new JFrame();
      f.setSize(300,150);
      f.setVisible(true);
      f.addMouseListener(this);
      f.setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);
   }
    public void mouseClicked(MouseEvent e){
       f.setTitle("点击坐标为 ("+e.getX()+", "+e.getY()+")");
    }
   public static void main(String[] args){ new ActionDemo();}
}
```

方法2：继承MouseAdapter类

9.5 事件处理——例子

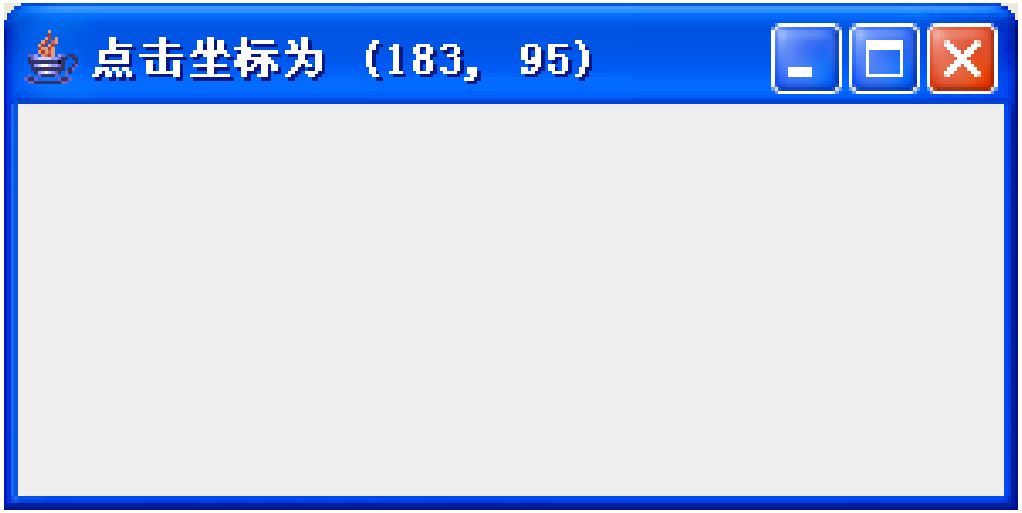
- 当鼠标在窗口中点击时，在窗口标题栏中显示点击位置坐标。

```
import java.awt.event.*;
import javax.swing.*;
public class ActionDemo{
   JFrame f;
   public ActionDemo() {
      f=new JFrame();
      f.setSize(300,150);
      f.setVisible(true);
      f.addMouseListener(new MouseAdapter(){
          public void mouseClicked(MouseEvent e){
             f.setTitle("点击坐标为 ("+e.getX()+", "+e.getY()+")");
          }
      }); //为窗口添加鼠标事件监听器语句结束
      f.addWindowListener(new WindowAdapter(){
         public void windowClosing(WindowEvent e){
            System.exit(0);
         }
      });//为窗口添加窗口事件监听器语句结束
   }
   public static void main(String[] args){ new ActionDemo(); }
}
```

方法3：使用匿名内部类

9.5 事件处理——例子

- 采用三种不同方法的程序，其运行效果都是一样的，当鼠标在窗口中点击的时候，窗口标题栏将出现所点位置的坐标信息



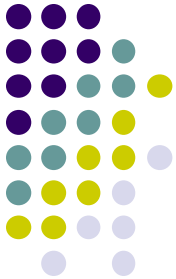
9.5 事件处理

- 事件接口及处理方法

事件描述信息	接口名称	方法（事件）
点击按钮、点击菜单项、文本框按回车等动作	ActionListener	actionPerformed(ActionEvent)
选择了可选项的项目，如复选框、单选按钮、下拉选项框	ItemListener	itemStateChanged(ItemEvent)
文本组件内容改变	TextListener	textValueChanged(TextEvent)
移动了滚动条等组件	AdjustmentListener	adjustmentVlaueChanged (AdjustmentEvent)
鼠标移动	MouseMotionListener	mouseDragged(MouseEvent) mouseMoved(MouseEvent)

9.5 事件处理

- 事件接口及处理方法



事件描述信息	接口名称	方法（事件）
鼠标点击等	MouseListener	mousePressed(MouseEvent) mouseReleased(MouseEvent) mouseEntered(MouseEvent) mouseExited(MouseEvent) mouseClicked(MouseEvent)
键盘输入	KeyListener	keyPressed(KeyEvent) keyReleased(KeyEvent) keyTyped(KeyEvent)
组件收到或失去焦点	FocusListener	focusGained(FocusEvent) focusLost(FocusEvent)
组件移动、缩放、显示/隐藏等	ComponentListener	componentMoved(ComponentEvent) componentHidden(ComponentEvent) componentResized(ComponentEvent) componentShown(ComponentEvent)

9.5 事件处理——按钮事件的示例

```
1.  import java.awt.*;
2.  import java.awt.event.* ;
3.  import javax.swing.*;
4.  public class TestJButton implements ActionListener{
5.    public TestJButton(){
6.      JFrame f = new JFrame("Test Button Event!");
7.      Container c=f.getContentPane();
8.      JButton b = new JButton("Press Me!");
9.      b.addActionListener(this);
10.     c.add(b, "Center");
11.     f.setSize(200,100);
12.     f.setVisible(true);
13.   }
14.   public void actionPerformed(ActionEvent e){
15.     System.out.println("Action occurred");
16.     System.out.println("Button's label is:"+
17.     e.getActionCommand());
18.   }
19.  public static void main(String args[ ]){
20.    new TestJButton();
21.   }
22. }
```

方法1：采用同一个类中实现事件接口的方法

9.5 事件处理——如何监听多个组件事件？

- 编写一个允许学生在文本字段中输入一个数的程序。创建一个每当用户单击一次就将此数加一的按钮。创建另一个每当用户单击一次就将此数减一的按钮。 界面效果如下图所示。

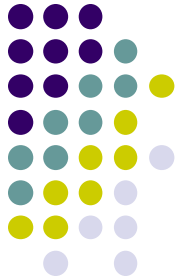
9.5 事件处理——如何监听多个组件事件？

```
1.  import java.awt.*;
2.  import java.awt.event.*;
3.  import javax.swing.*;
4.  class Incrementor implements ActionListener{
5.      JTextField numberTxf;
6.      JButton incrementBtn,decrementBtn;
7.      public void makeGUI(){
8.          JFrame frm = new JFrame("Incrementor");
9.          Container c=frm.getContentPane();
10.         c.setLayout(new FlowLayout());
11.         numberTxf = new JTextField("0",5);
12.         c.add(numberTxf);
13.         incrementBtn = new JButton("Increment");
14.         c.add(incrementBtn);
15.         incrementBtn.addActionListener(this);
16.         decrementBtn= new JButton("Decrement");
17.         c.add(decrementBtn);
18.         decrementBtn.addActionListener(this);
19.         frm.setSize(300,100);
20.         frm.setVisible(true);
21.     }
```

9.5 事件处理——如何监听多个组件事件？

```
22  public void actionPerformed(ActionEvent e) {
23          int oldNum = Integer.parseInt(numberTxf.getText());
24          int newNum = oldNum;
25          if(e.getSource()==incrementBtn){
26             newNum++;
27          }
28          else if(e.getSource()==decrementBtn){
29             newNum--;
30          }
31        numberTxf.setText(String.valueOf(newNum));
32      }
33      public static void main(String args[]) {
34       Incrementor i = new Incrementor();
35       i.makeGUI();
36      }
37  }
```

9.5 事件处理——选项事件ItemEvent



- 可触发选项事件的组件有：
  JCheckBox,JRadioButton,JComboBox

- 注册事件的方法：
  public void addItemListener(ItemListener e)

- 处理事件的接口**ItemListener**，仅含有方法：
  public void itemStateChanged(ItemEvent e)

9.5 事件处理——选项事件ItemEvent

- 用JComboBox和JCheckBox来演示可选项目事件接口，界面如下图所示：

```
1.import java.awt.*;
2.import java.awt.event.* ;
3.import javax.swing.*;
4.public class ItemDemo implements ItemListener{
5.     JFrame f;
6.     JPanel p1,p2,p3;
7.     JLabel birthPlace,hobby;
8.     JComboBox place;
9.     JCheckBox hobby1,hobby2,hobby3;
10.     public ItemDemo(String title){
11.         f=new JFrame(title);
12.         p1=new JPanel();
13.         birthPlace=new JLabel("出生地:");
14.         place=new JComboBox();
15.         place.addItemListener(this);
16.         place.addItem("杭州");
17.         place.addItem(“上海");
18.         place.addItem(“南京");
19.         place.addItem(“苏州");
20.         p1.add(birthPlace);
21.         p1.add(place);
22.         f.add(p1,"North");
```

```
23. p2=new JPanel();
24.         p2.setLayout(new GridLayout(3,1));
25.         hobby=new JLabel("业余爱好:");
26.         hobby1=new JCheckBox("运动");
27.         hobby1.addItemListener(this);
28.         hobby2=new JCheckBox("旅游");
29.         hobby2.addItemListener(this);
30.         hobby3=new JCheckBox("上网");
31.         hobby3.addItemListener(this);
32.         p2.add(hobby1);
33.         p2.add(hobby2);
34.         p2.add(hobby3);
35.         p3=new JPanel();
36.         p3.add(hobby);
37.         p3.add(p2);
38.         f.add(p3,"Center");
39.
40.         f.pack();
41.         f.setVisible(true);
42.     }
```

```
43. public void itemStateChanged(ItemEvent e){
44.   if(e.getSource()==place&&e.getStateChange()==ItemEvent.SELECTED){
45.         System.out.println("您当前选择的是"+place.getSelectedItem());
46.       }
47.       else if(e.getSource()==hobby1 &&hobby1.isSelected() ){
48.         System.out.println("您的业余爱好有： "+hobby1.getLabel());
49.       }
50.       else if(e.getSource()==hobby2 &&hobby2.isSelected()){
51.         System.out.println("您的业余爱好有： "+hobby2.getLabel());
52.       }
53.       else if(e.getSource()==hobby3 &&hobby3.isSelected()){
54.         System.out.println("您的业余爱好有： "+hobby3.getLabel());
55.       }
56.    }
57.    public static void main(String args[ ]
58.       new ItemDemo("测试ItemListene
59.   }
60. }
```

本章小结

- **Swing组件包含了Windows平台中所见的几乎所有的组件，完全由Java实现（轻量组件），且功能更强大。事件处理与AWT一样。**
- 编程步骤：
  - 导入Swing包
  - 设置顶层容器(布局管理器)
  - 设置组件
  - 增加组件到容器
  - 事件处理
- **掌握Swing的结构和特点，学会使用布局管理、事件处理，以及常用的Swing组件**