| Java技术

第五章（2）

Java GUI设计与事件处理

路　强

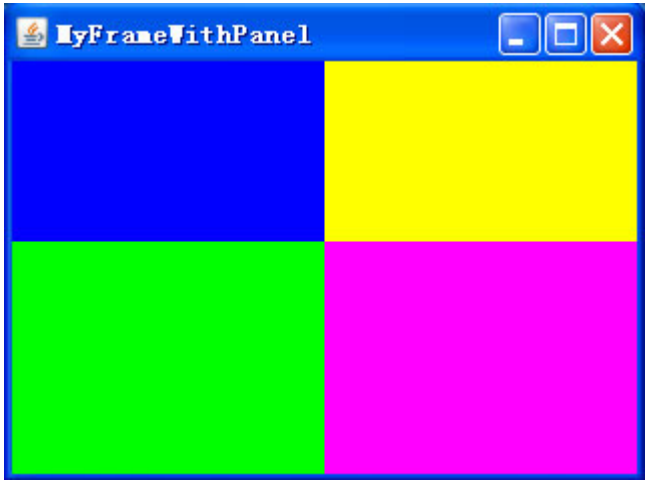
luqiang@hfut.edu.cn

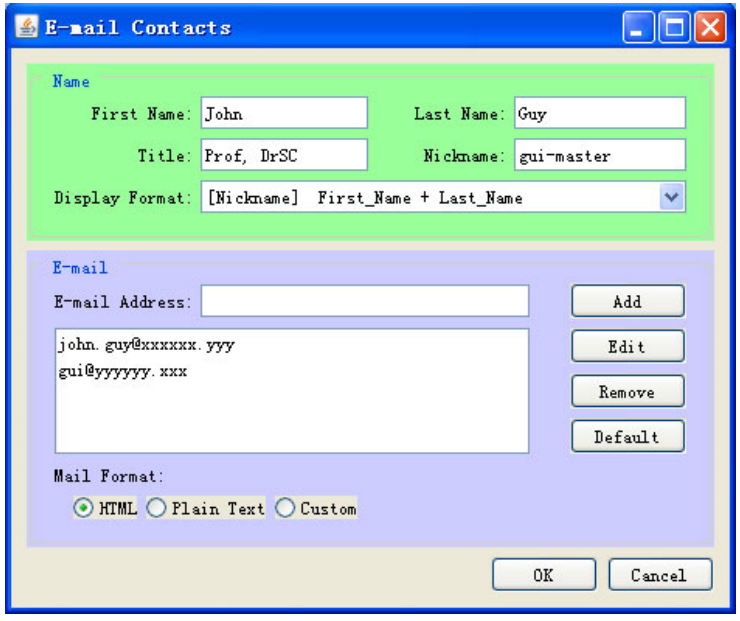
合肥工业大学计算机与信息学院

回顾 - Java的GUI设计

- 在设计和实现图形用户界面的过程中，主要完成两个任务：
  - 创建窗口并在窗口中添加各种组件，指定组件的属性和它们在窗口中的位置，从而构成图形界面的外观效果
  - 定义图形界面的事件和各种组件对不同事件的响应，从而实现图形界面与用户的交互

回顾 - Java的GUI设计





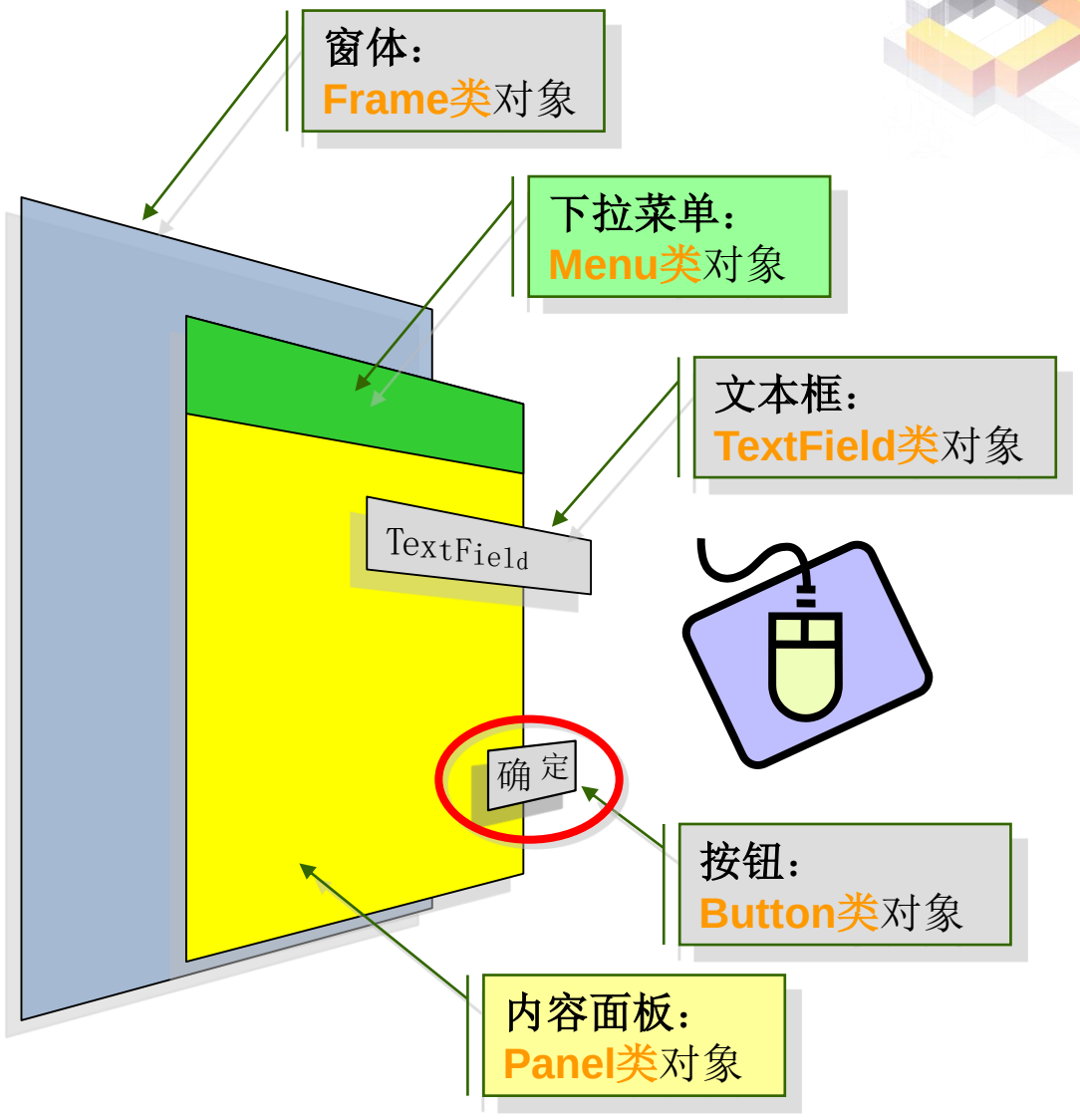
- 一般可按照下列流程进行
  1. 引入Java图形组件包 *AWT*、*Swing*
  2. 选择“外观和感觉”
  3. 设置窗体属性
  4. 设置组件布局
  5. 向窗体中添加组件
  6. 对组件进行事件处理

回顾 –实现GUI步骤示意

“交互”的实现

?如何判断“按钮”变化

?如何实现按下以后的工作



目　录

交互处理方式分析

- 处理“人-机”交互的两种方式
  1. 查询处理方式
     - C等传统语言的方式
     - 主要用于Dos等命令行环境下程序设计
  2. 事件处理机制
     - Java、Visual C++的方式
     - 主要用于Windows程序和多线程程序设计
- 两种处理方式进行比较的内容
  - 如何“查询”
  - 软件设计人员的需要完成的工作

两种方式比较 - 如何“查询”

- 看护小宝宝“换尿布”的方式
  - 传统方法：每过一会就摸摸...
    如果尿布湿了，换新的
  - 新方法：尿布自动报警器(湿敏电阻)
- 比 较
  - 工作方式：
    由 “监测->处理” 变为 “提醒告知”
  - 新方法优点：提高并发度、效率
  - 新方法缺点：提高了复杂度 (价格...)
- 程序设计的思路：查询 -> 报告

两种方式比较 - “程序员的工作”

- “查询”方式下软件设计者要完成的工作
  - ？ 例. 如何处理鼠标的多种操作

```
While (true) do{      /*查询鼠标操作*/
    if (用户单击了鼠标左键)
        Then {进行方式一处理}
    else if (用户单击了鼠标右键)
        Then {进行方式二处理}
    else if (用户双击了鼠标左键)
        Then {进行方式三处理}
    else . . .
}
```

- 实现方式
  - “查询”
  - 对操作类型逐个比较
- 设计工作
  - 判断发生交互种类
  - 设计“处理”代码
  - “交互”和”处理”代码的组织方式
- 缺点
  - 需编程判断交互类型
  - “交互”和”处理“的代码交织在一起

两种方式比较 - “程序员的工作” 续

- 事件处理机制
  - Java、Visual C++的方式，应对
    ① 多道程序系统：程序执行效率
    ② 图形化界面程序设计：大量复杂交互
  - 由程序运行环境检测“事件”发生与否
    由程序运行环境自动调用相应的事件处理代码
  - 软件设计者只要关心：
    ① 会发生哪些“交互事件”
    ② 如何处理“交互事件”

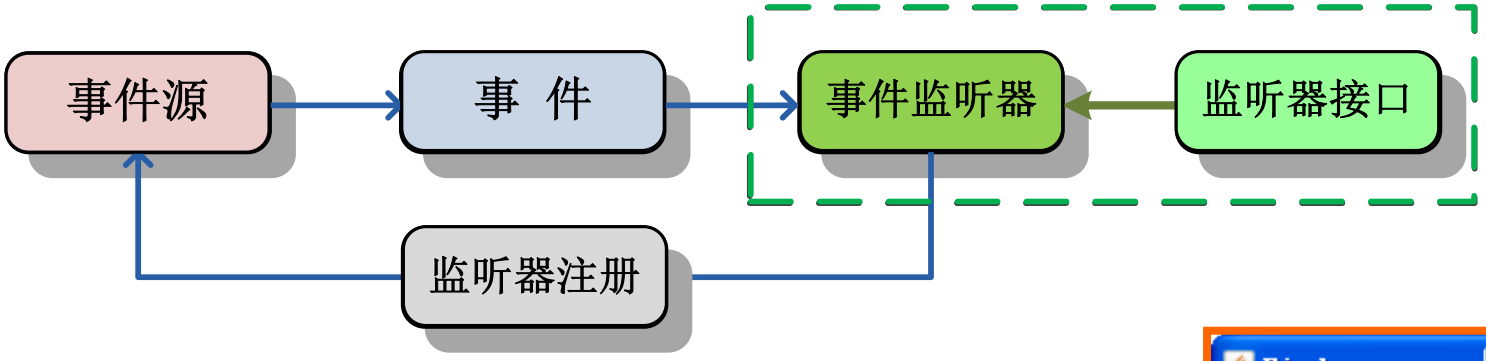
※ 优 点：

- 多道程序时，系统执行效率提高
- Java虚拟机全权处理，程序员只需专注功能设计
- 对复杂程序（复杂交互）的处理便利、结构清晰

高效率

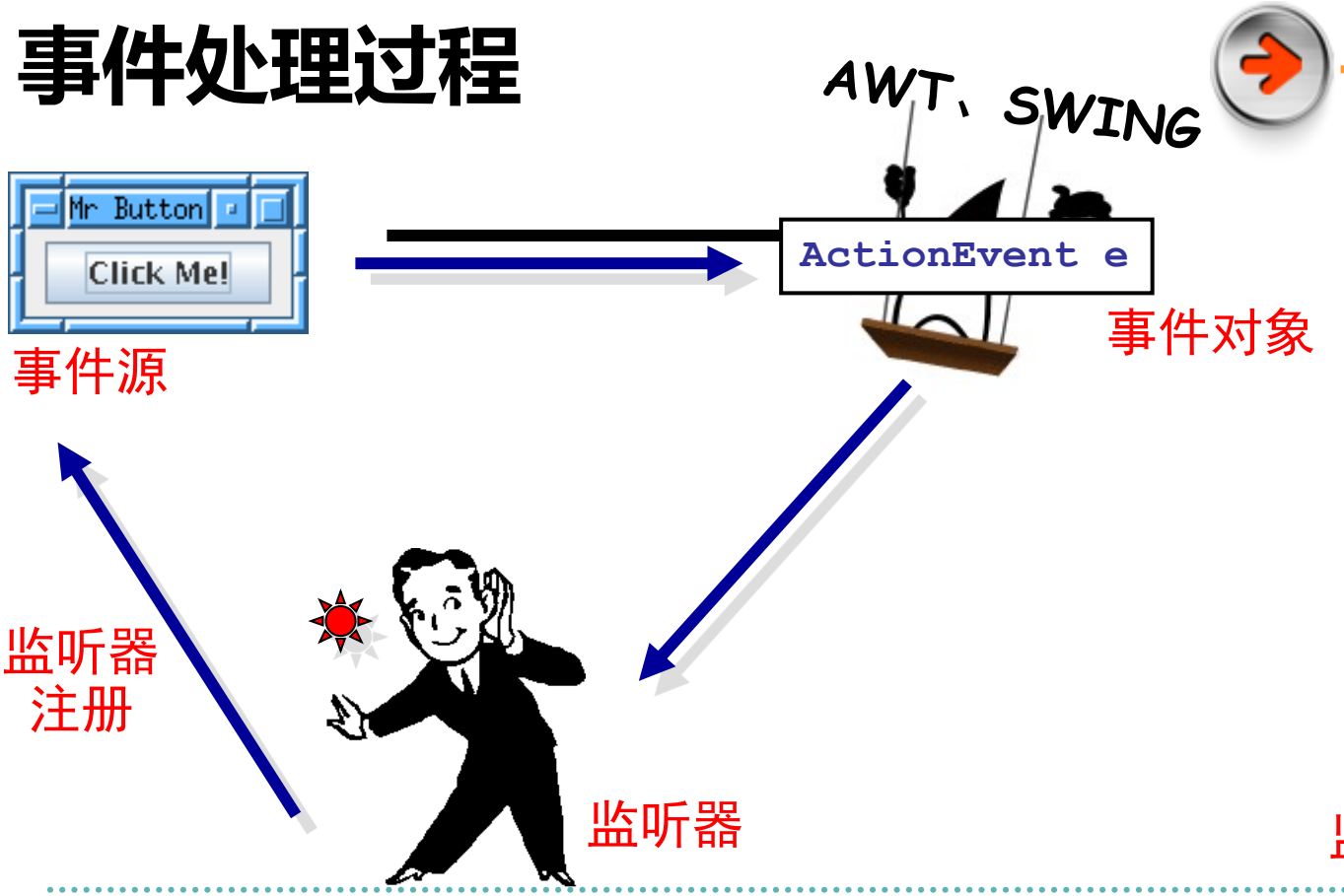
目　录

2.1 事件机制的几个概念



- **事件源** 描述人机交互中事件的来源
  (EventSource) 通常是Java图形包中组件
- **事　件** 事件源产生的交互内容，如“按下鼠标”
  (ActionEvent) 在 java.awt.event包中定义的类
- **事件监听器** 接收事件并进行处理，由程序员编写
  (ActionListener) 对应处理所监听事件源产生的事件
- **监听器接口** 编写“事件监听器”的“规定” –“抽象方法”
  必须在监听器类中实现这些方法完成事件处理
- **监听器注册** 将事件监听器对象绑定到事件源，进行监听

事件处理过程



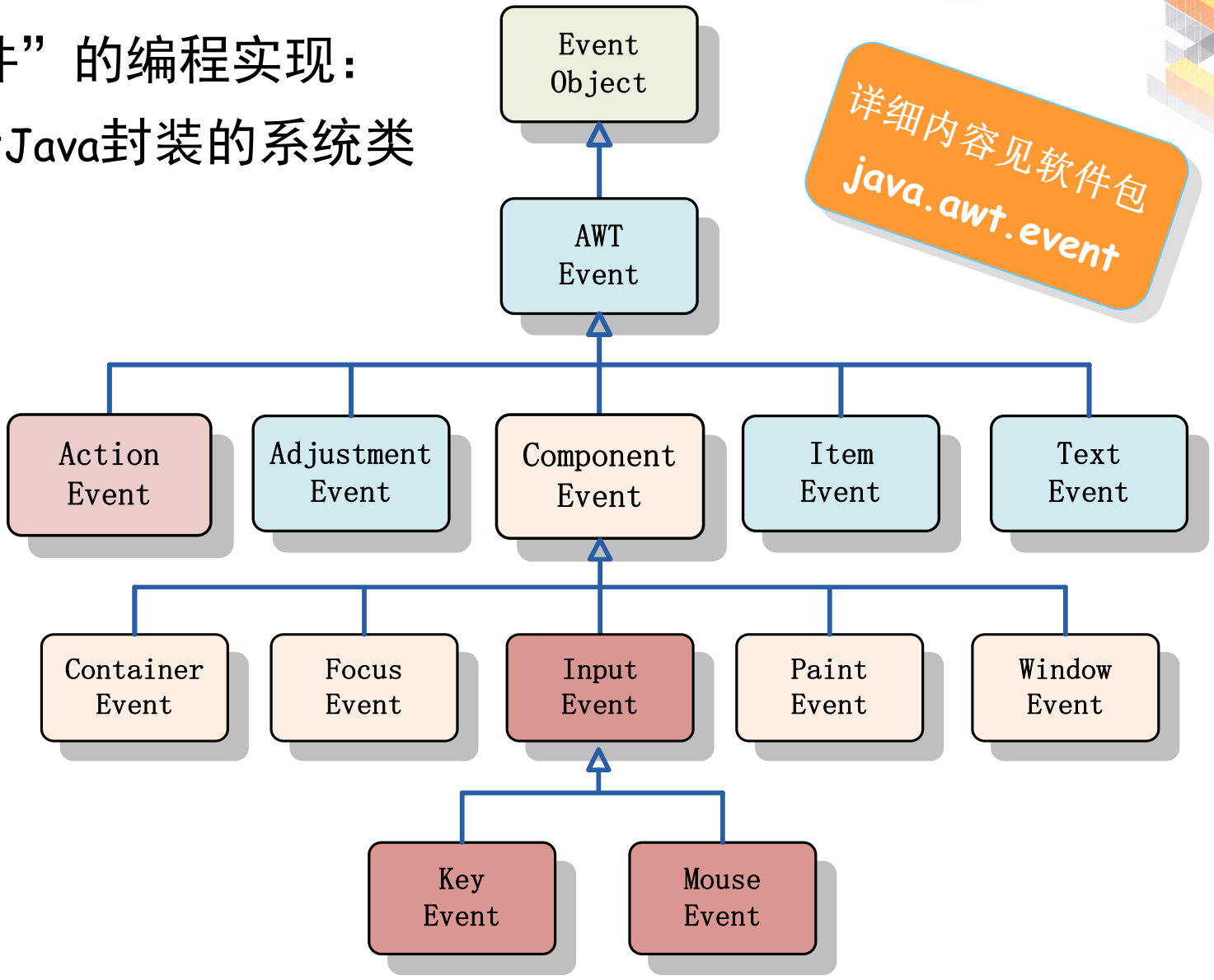

```
class ButtonListener implements ActionListener {

    public void actionPerformed(                ) {

        /**   按钮事件所进行的具体工作   **/
    }
}
```

2.2 常用事件类

事件类型	典型触发动作
ActionEvent	按钮、列表双击、单击菜单项目
KeyEvent	键盘的输入
MouseEvent	鼠标拖动、移动、单击、按下、释放或者进入、退出组件的事件
ComponentEvent	组件被隐藏、移动、尺寸调整或变为不可见的事件
FocusEvent	组件获得或失去焦点的事件
InputEvent	复选框和列表项单击、控件的选择和可选菜单项的选择事件
TextEvent	文本区域或者文本区域的值的改动
WindowEvent	窗口激活、失去活动窗口、最小化、最小化、打开、关闭或者退出的事件

事件类的继承层次关系

- “事件”的编程实现：
  基于Java封装的系统类



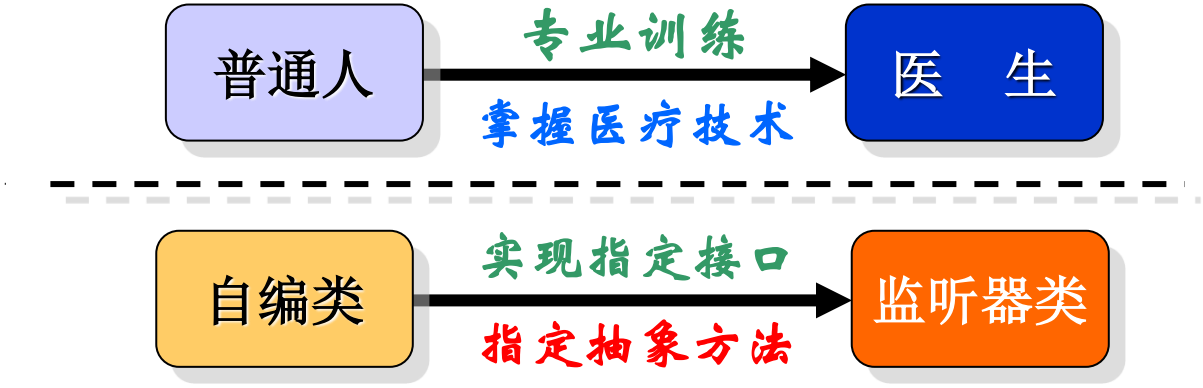
详细内容见软件包
java.awt.event

2.3 事件监听器

- “实现了某种类型的监听器接口”的类的对象

※ 如何编程实现监听器 ？

1. 每一个事件类都有唯一的事件处理方法接口，
   例如，对于处理鼠标事件“MouseEvent”类的对应接口为“MouseListener”
2. 每一个接口中都已经规定了一个空的抽象方法
   在该方法中编码实现自己想做的的工作



2.3 常用事件监听器类

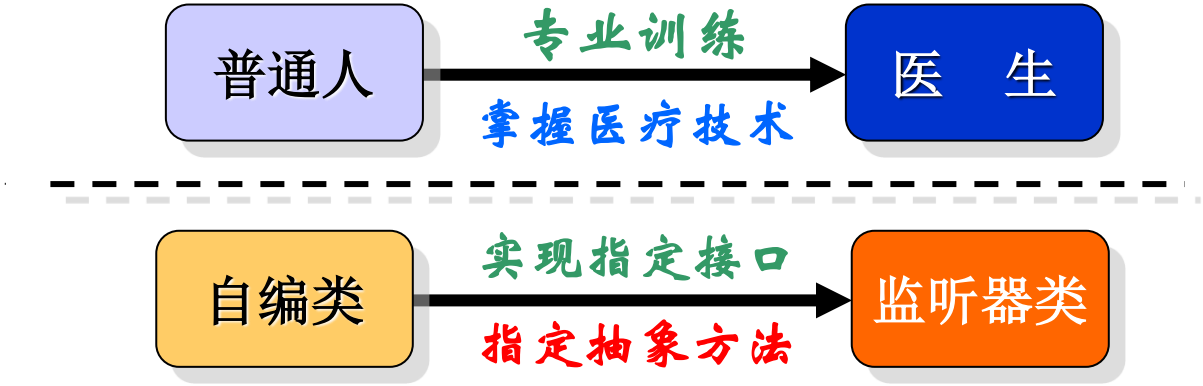
事件类型	典型动作
ActionListener	处理按钮、列表双击、单击菜单项目
KeyListener	处理键盘的输入
MouseListener	处理鼠标拖动、移动、单击、按下、释放或者进入、退出组件的事件
ComponetListener	处理组件被隐藏、移动、尺寸调整或者变为不可见的事件
FocusListener	处理组件获得或失去焦点的事件
TextListener	处理文本区域或者文本区域的值的改动
WindowListener	处理窗口激活、失去活动窗口、最小化、最小化、打开、关闭或者退出的事件

事件监听器的设计实现

- “实现了某种类型的监听器接口”的类的对象

※ 如何编程实现监听器 ？

1. 每一个事件类都有唯一的事件处理方法接口，
   例如，对于处理鼠标事件“MouseEvent”类的对应接口为“MouseListener”
2. 每一个接口中都已经规定了一个空的抽象方法
   在该方法中编码实现自己想做的的工作



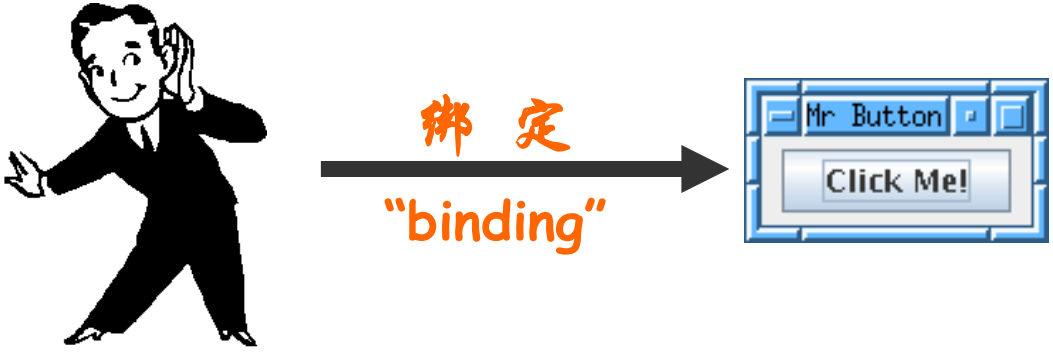
“事件源-监听器-抽象方法”对应

事件类型	监听器名称	抽象方法 （必须实现）
ActionEvent	ActionListener	actionPerformed(ActionEvent)
KeyEvent (键盘事件)	KeyListener (键盘监听器)	按 下 keyPressed(keyEvent)
		释 放 keyReleased(keyEvent)
		按+放 keyTyped(keyEvent)
MouseEvent (鼠标事件)	MouseMotionListener (鼠标移动监听器)	移 动 mouseMoved(MouseEvent)
		拖 动 mouseDragged(MouseEvent)
	MouseListener (鼠标按键监听器)	按 下 mousePressed(mouseEvent)
		释 放 mouseReleased(**mouseEvent**)
		进 入 mouseEntered(mouseEvent)
		退 出 mouseExited(mouseEvent)
. . .	. . .	. . .

事件监听器.续

※ 如何将监听器绑定到组件？

- 每个组件都提供了用于绑定监听器的方法
- 通过观察 “addxxxListener” 方法的名称，可以很容易地知道其能够处理的事件类型



单事件源的“多个事件”

1. 学生可发生多种事件
2. 教师、辅导员、医生的共同点：

   都受过“专业培训”

目　录

程序设计.例一

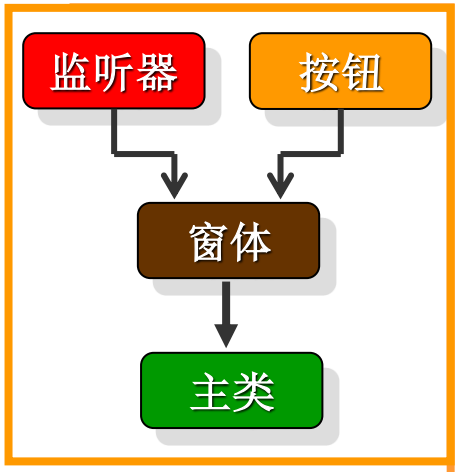
程序功能需求

- 绘制一个窗体
- 窗体内绘制一个按钮，名为“*点我*”
- 按下按钮时，在命令行打印信息
  打印内容为 “*我知道你按下按钮啦！*”

解决方法

- 设计自己的带有按钮的窗体类
  (派生自Java.awt.Frame类)
- 设计自己的按钮事件监听器类
  (实现ActionListener接口、并完成ActionPerformed方法)
- 创建按钮事件监听器类对象，并绑定到“按钮”上
  (调用按钮类的 addActionListener方法)

事件处理例1.代码



监听器类

设计自己的窗体类

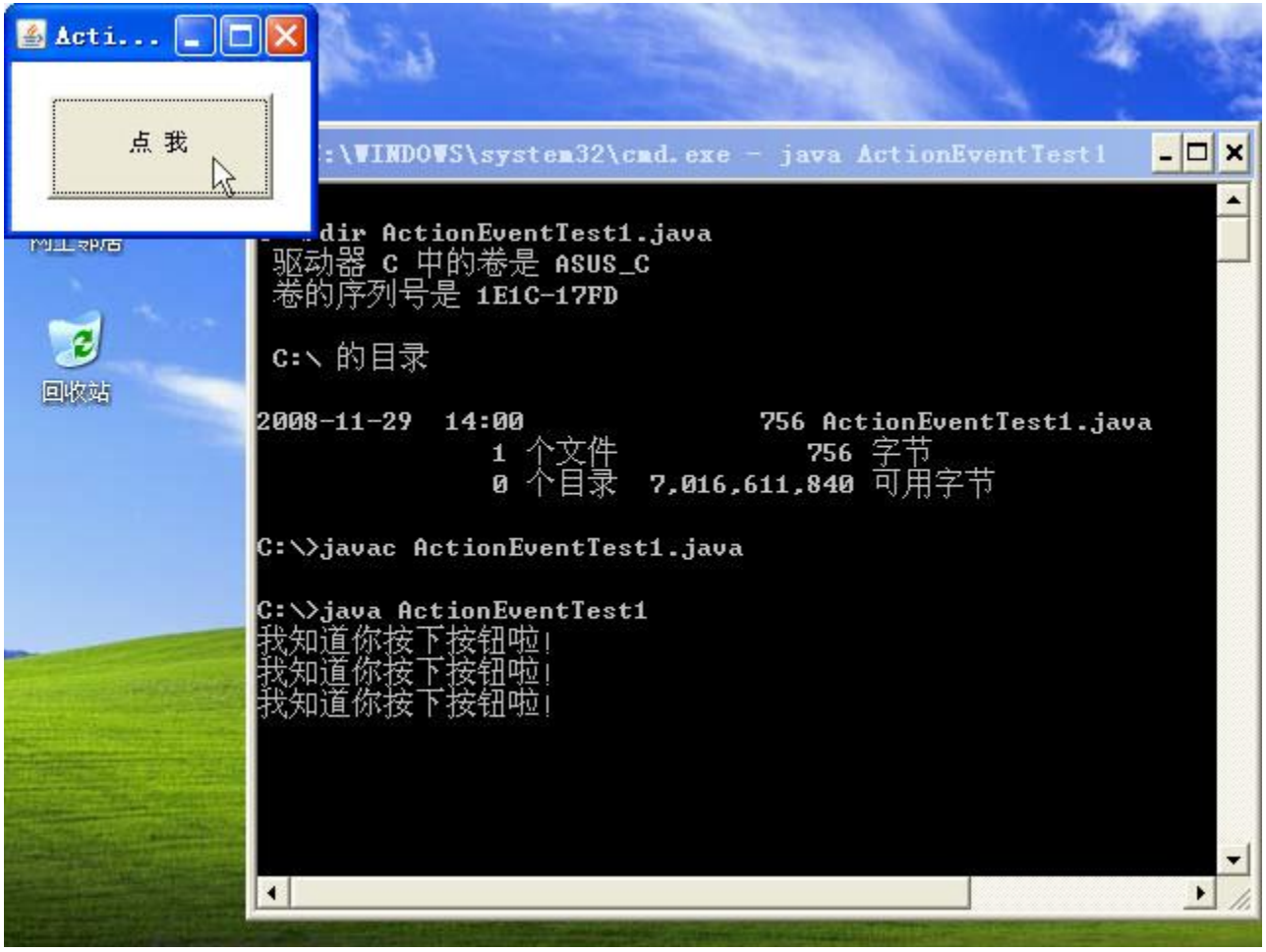
主类

```
1.  import java.awt.*;     import java.awt.event.*;
2.  class ButtonListener implements ActionListener {
3.          public void actionPerformed ( ActionEvent e ) {
4.                  System.out.println ( “我知道你按下按钮啦“ );
5.          }
6.  }
7.  class myButtonFrame extends Frame {
8.          Button btn;
9.          myButtonFrame(String s) {  //构造函数
1.                  super(s);
2.                  this.setSize(200,120);
3.   /* 创建按钮*/ btn = new Button("点击");
4.                  this.add(btn);
5.
6.
7.
8.          }
9.  }
19. public class ActionEventTest {     // 主类
20.         public static void main(String args[]){
21.                 myButtonFrame frm = new myButtonFrame("ActionEventTest");
22.                 frm.show();        //   显示窗体
23.             }
24.         }
```

按钮 事件源

监听器

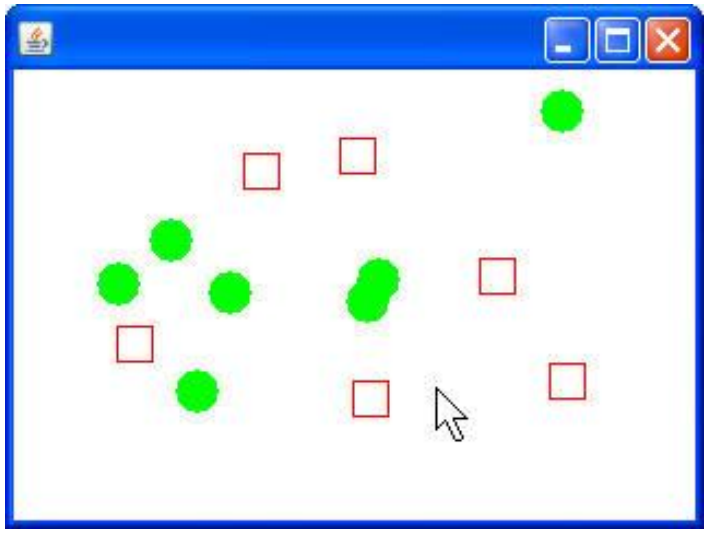
程序设计. 例一. 运行结果



代码见

ActionEventTest1.java

程序设计.例二



程序功能需求

- 绘制一个窗体
- 窗体内“单击鼠标左键”，在鼠标处绘制一个“绿圆”
- 窗体内“单击鼠标右键”，在鼠标处绘制一个“红色方框”
- 窗体内“双击鼠标左键”，清空所有已画“圆”和“方”

本节课小结

本节课我们学习了

- 事件机制的功能 (**W**hy： 为什么需要…)
- 事件机制架构和流程 (**W**hat：什么是…)
- 事件机制的编程实现 (Ho**w**： 怎么创建…)

下一节课将要学习

- Java Swing图形包
- 复杂交互程序的设计与实现

经验与建议

- “**最有用**” 的Java资料 – JDK API手册
- “**最有效**” 的学习方法 – 亲手编程、调试

作业3

作业3：见作业文档。


Thank You !