



北京圣思园科技有限公司
<http://www.shengsiyuan.com>

主讲人：张龙

AWT事件模型

- 课程目标
 - -编写代码来处理在图形用户界面中发生的事件
 - -描述**Adapter**类的概念，包括如何和何时使用它们
 - -根据事件对象的细节来确定产生事件的用户动作
 - -为各种类型的事件创建合适的接口和事件处理器



什么是事件

- - 事件—描述发生了什么的对象
- - 事件源—事件的产生器
- - 事件处理器—接收事件、解释事件并处理用户交互的方法
 - 如果用户在用户界面层执行了一个动作(鼠标点击和按键)，这将导致一个事件的发生。**事件是描述发生了什么的对象**。存在各种不同类型的事件类用来描述各种类型的用户交互。



什么是事件

- 事件源

- 事件源是一个事件的产生者。例如，在Button 组件上点击鼠标会产生以这个Button 为源的一个ActionEvent. 这个ActionEvent实例是一个对象，它包含关于刚才所发生的那个事件的信息的对象，这些信息包括：

- getActionCommand—返回与动作相关联的命令名称
 - getWhen—返回事件发生的时间



什么是事件

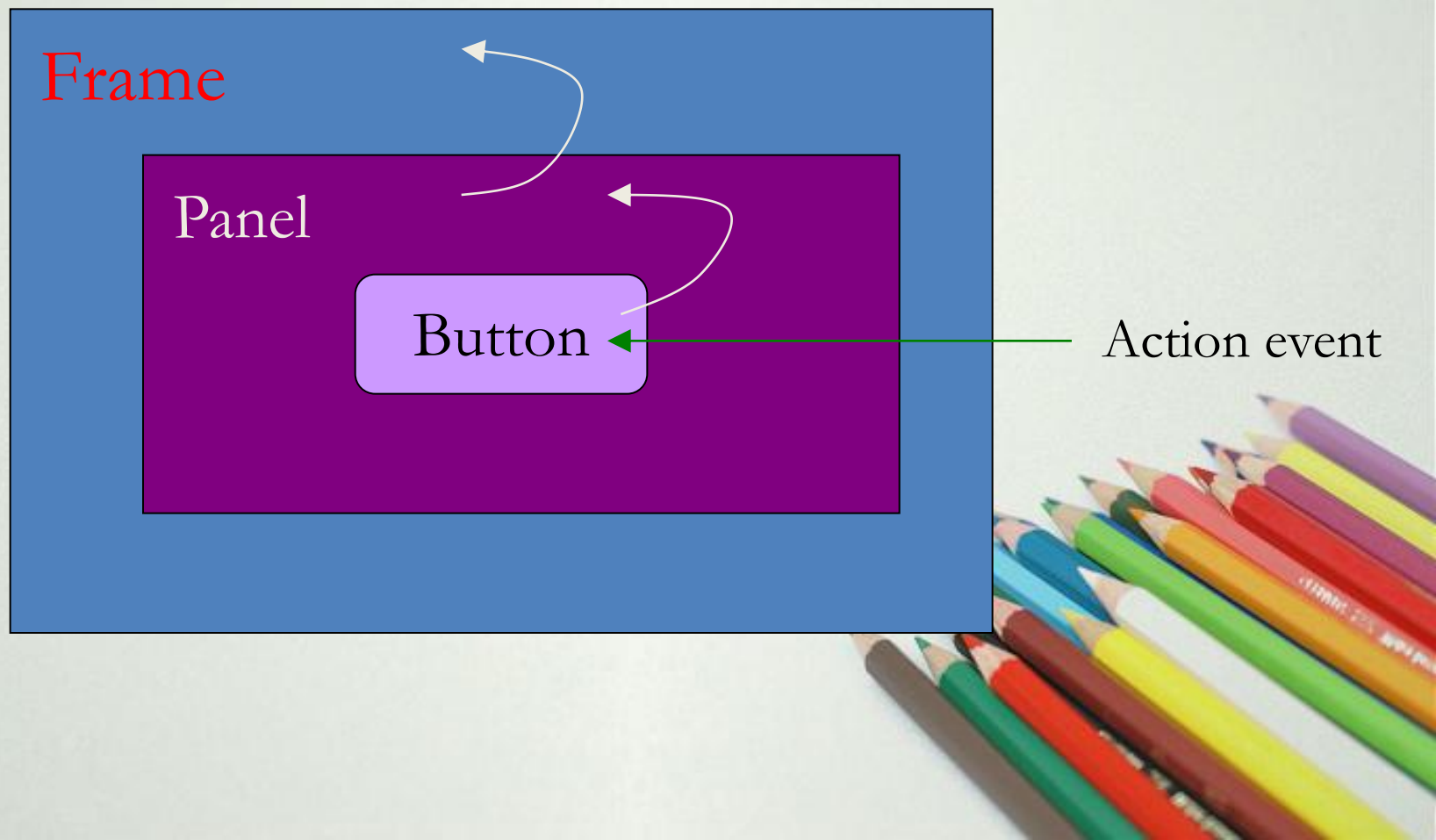
- 事件处理器

- 事件处理器就是一个接收事件、解释事件并处理用户交互的**方法**。



AWT事件模型

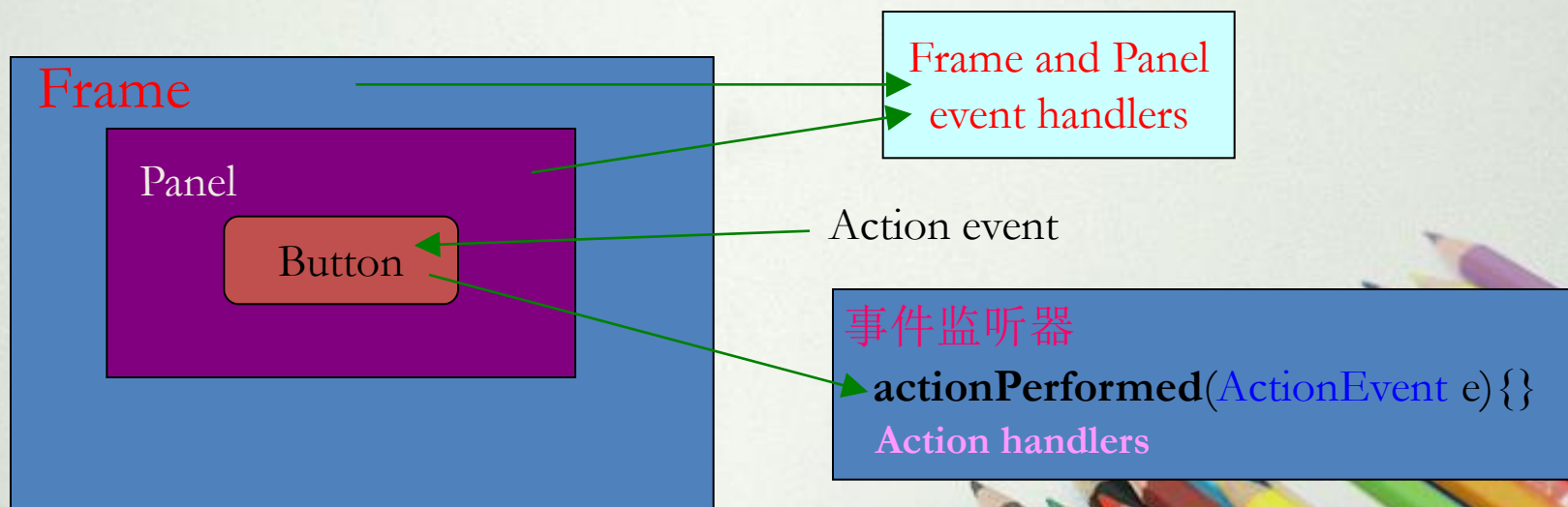
- JDK1.0的事件模型：层次模型



AWT事件模型

- JDK1.1的事件模型：委托模型

事件监听器：实现了监听器接口的类。一个监听器对象是一个实现了专门的监听器接口的类的实例。



AWT事件模型

- 参见程序 **TestButton.java**
 - 在Button对象上用鼠标进行点击时，将发送一个ActionEvent事件。这个ActionEvent事件会被使用addActionListener()方法进行注册的所有ActionListener的actionPerformed()方法接收
 - ActionEvent类的getActionCommand()方法返回与动作相关联的命令名称。
 - 以按钮的点击动作为例，将返回Button的标签。



委托模型(JDK1.1或更高版本)

- 优点

- 事件不会被意外地处理。
- 有可能创建并使用适配器 (**adapter**)类对事件动作进行分类。
- 委托模型有利于把工作分布到各个类中。




AWT事件处理

事件处理机制

几类具有典型代表意义的事件：

类	对应事件	说明
MouseEvent	鼠标事件	鼠标按下，鼠标释放，鼠标点击等
WindowEvent	窗口事件	点击关闭按钮，窗口得到与失去焦点，窗口最小化等
ActionEvent	动作事件	<p>不代表具体的动作，是一种语义，如按钮或菜单被鼠标单击，单行文本框中按下回车键等都可以看作是ActionEvent事件</p> <p>可以这么理解，如果用户的一个动作导致了某个组件本身最基本的动作发生了，这就是ActionEvent事件</p>



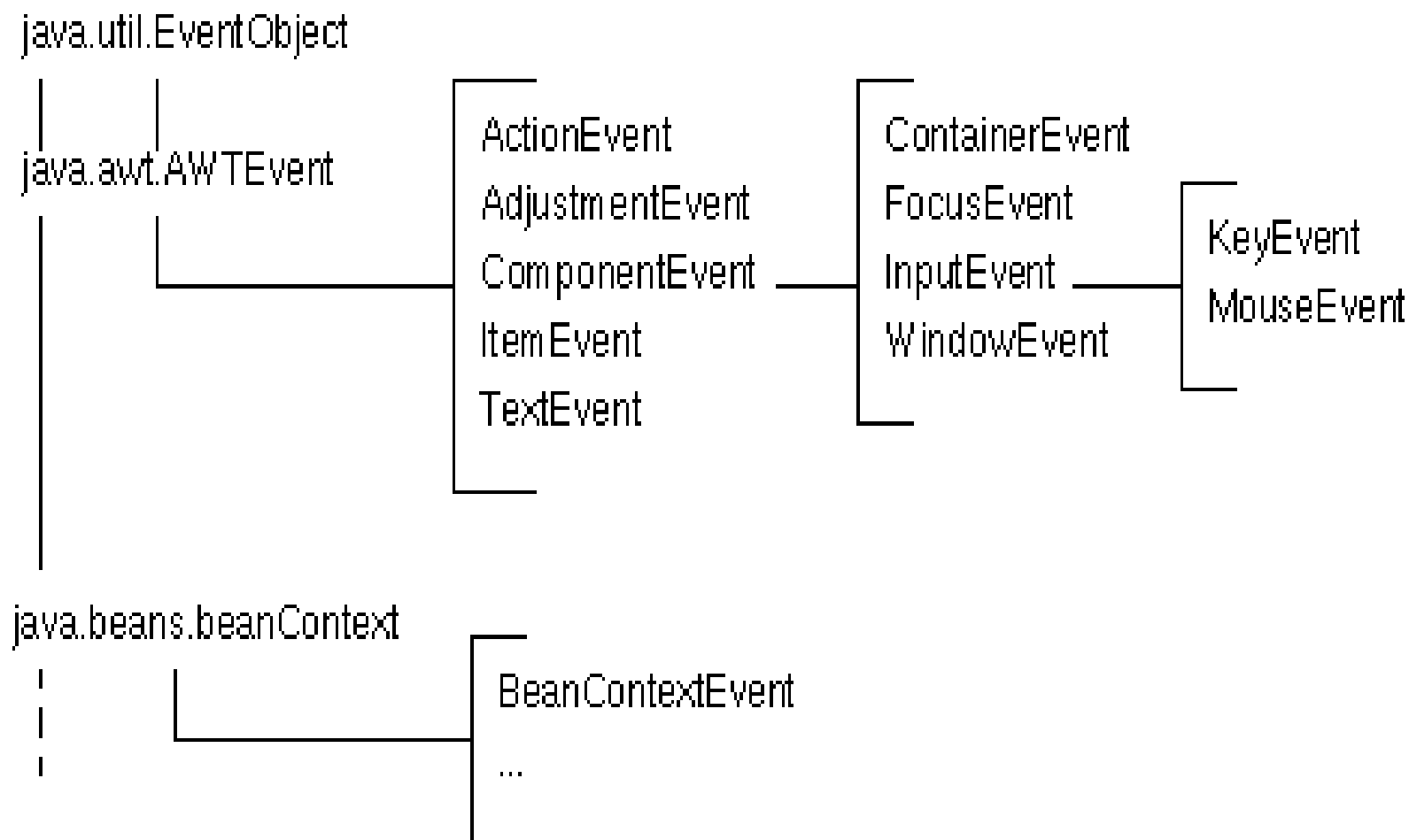
图形用户界面的行为

- 事件类型

- 事件类的层次结构图如下所示。许多事件类在 `java.awt.event` 包中，也有一些事件类在API的其他地方
- 对于每类事件，都有一个接口，这个接口必须由想接收这个事件的类的对象实现。这个接口还要求定义一个或多个方法。当发生特定的事件时，就会调用这些方法



图形用户界面的行为



方法类型和接口

Category	Interface Name	Methods
Action	ActionListener	actionPerformed (ActionEvent)
Item	ItemListener	itemStateChanged (ItemEvent)
Mouse motion	MouseMotionListener	mouseDragged (MouseEvent) mouseMoved (MouseEvent)
Mouse button	MouseListener	mousePressed (MouseEvent) mouseReleased (MouseEvent) mouseEntered (MouseEvent) mouseExited (MouseEvent) mouseClicked (MouseEvent)
Key	KeyListener	keyPressed (KeyEvent) keyReleased (KeyEvent) keyTyped (KeyEvent)
Focus	FocusListener	focusGained (FocusEvent) focusLost (FocusEvent)
Adjustment	AdjustmentListener	adjustmentValueChanged (AdjustmentEvent)
Component	ComponentListener	componentMoved (ComponentEvent) componentHidden (ComponentEvent) componentResized (ComponentEvent) componentShown (ComponentEvent)

方法类型和接口

Category	Interface Name	Methods
Window	WindowListener	<code>windowClosing(WindowEvent)</code>
		<code>windowOpened(WindowEvent)</code>
		<code>windowIconified(WindowEvent)</code>
		<code>windowDeiconified(WindowEvent)</code>
		<code>windowClosed(WindowEvent)</code>
		<code>windowActivated(WindowEvent)</code>
		<code>windowDeactivated(WindowEvent)</code>
Container	ContainerListener	<code>componentAdded(ContainerEvent)</code>
		<code>componentRemoved(ContainerEvent)</code>
Text	TextListener	<code>textValueChanged(TextEvent)</code>

范例

- 参见程序 `TwoListen.java`



多监听者

- 多监听者可以使一个程序的不相关部分执行同样的动作
- 事件发生时，所有被注册的监听者的处理器都会被调用



事件Adapters(适配器)

- 你定义的Listener可以继承Adapter类，而且只需**重写**你所需要的方法
- 为了方便起见，Java语言提供了Adapters类，用来实现含有多个方法的类。**这些Adapters类中的方法是空的。**
- **你可以继承Adapters类，而且只需重写你所需要的方法**



事件Adapters(适配器)

为简化编程，JDK针对大多数事件监听器接口定义了相应的实现类，我们称之为事件适配器（Adapter）类

在适配器类中，实现了相应监听器接口所有方法，但不做任何事情，只要继承适配器类，就等于实现了相应的监听器接口

如果要对某类事件的某种情况进行处理，只要覆盖相应的方法就可以，其他的方法再也不用“简单实现”了

如果想用作事件监听器的类已经继承了别的类，
就不能再继承适配器类了，
只能去实现事件监听器接口了



事件Adapters(适配器)

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

public class MouseClickHandler extends MouseAdapter {

    // We just need the mouseClicked handler, so we use
    // the an adapter to avoid having to write all the
    // event handler methods

    public void mouseClicked (MouseEvent e) {
        // Do stuff with the mouse click...
    }
}
```