9.2.2 菜单条、菜单、菜单项

- 窗口中的菜单条、菜单、菜单项是我们所熟悉的组件，菜单放在菜单条里，菜单项放在菜单里
- 1. 菜单条：JMenubar负责创建菜单条。

  JFrame类有一个将菜单条放置到窗口中的方法：

  setJMenuBar(JMenuBar bar);//将菜单条添加到窗口
- 2. 菜单：JMenu负责创建菜单
- 3. 菜单项：JMenuItem负责创建菜单项
- 4. 子菜单：JMenu是JMenuItem的子类，因此菜单本身也是一个菜单项，当把一个菜单看作菜单项添加到某个菜单中时，称这样的菜单为子菜单。
- 5. 菜单上的图标：为使菜单项有图标，用图标类Icon声明一个图标，用其子类ImageIcon类创建一个图标

9.2.2 菜单条、菜单、菜单项

```java
package shu.ces.java.chap9;

import javax.swing.*;
import java.awt.event.InputEvent;
import java.awt.event.KeyEvent;
import static javax.swing.JFrame.*;

public class WindowMenu extends JFrame {
   JMenuBar menubar;
    JMenu menu,subMenu;
    JMenuItem  itemLiterature,itemCooking;
    public WindowMenu(){}
    public WindowMenu(String s,int x,int y,int w,int h) {
       init(s);
       setLocation(x,y);
       setSize(w,h);
       setVisible(true);
       setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    }
    void init(String s){
       setTitle(s);                //设置窗口的标题
       menubar=new JMenuBar();
       menu=new JMenu("菜单");
       subMenu=new JMenu("体育话题");
       itemLiterature=new JMenuItem("文学话题",new ImageIcon("src/shu/ces/java/chap9/a.gif"));
       itemCooking=new JMenuItem("烹饪话题",new ImageIcon("src/shu/ces/java/chap9/b.gif"));
       itemLiterature.setAccelerator(KeyStroke.getKeyStroke('A'));
       itemCooking.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_S,InputEvent.CTRL_MASK));
       menu.add(itemLiterature);
       menu.addSeparator();    //在菜单之间增加分隔线
       menu.add(itemCooking);
       menu.add(subMenu);
       subMenu.add(new JMenuItem("足球",new ImageIcon("src/shu/ces/java/chap9/c.gif")));
       subMenu.add(new JMenuItem("篮球",new ImageIcon("src/shu/ces/java/chap9/d.gif")));
       menubar.add(menu);
       setJMenuBar(menubar);
    }
}
```

```java
package shu.ces.java.chap9;

public class Example9_2 {
   public static void main(String args[]) {
      WindowMenu win=new WindowMenu("带菜单的窗口",20,30,200,190);
   }
}
```

例9-2演示

第9章 GUI编程

1. Java Swing概述
2. 窗口
3. 常用组件与布局
4. 事件处理
5. 使用MVC结构
6. 对话框
7. 发布GUI程序

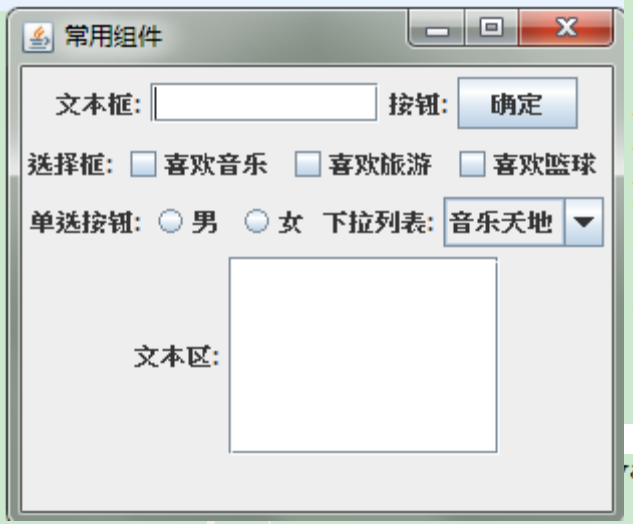
9.3 常用组件与布局

常用组件：

- 1. 文本框：JTextField类创建文本框，允许输入单行文本
- 2. 文本区：JTextArea类创建文本区，允许输入多行文本
- 3. 按钮：JButton类创建按钮，允许用户单击按钮
- 4. 标签：JLabel类创建标签，为用户提供信息提示
- 5. 复选框：JCheckBox类创建选择框，提供多项选择
- 6. 单选按钮：JRadioButton类创建单项选择框
- 7. 下拉列表：JComboBox类用来创建下拉列表
- 8. 密码框：JPasswordField创建密码框

9.3.1 常用组件

```
1  package shu.ces.java.chap9;
2
3  import java.awt.*;
4  import javax.swing.*;
5  public class ComponentInWindow extends JFrame {
6     JTextField text;
7     JButton button;
8     JCheckBox checkBox1,checkBox2,checkBox3;
9     JRadioButton radio1,radio2;
10    ButtonGroup group;
11    JComboBox comBox;
12    JTextArea area;
13    public ComponentInWindow() {
14       init();
15       setVisible(true);
16       setDefaultCloseOperation(J
17    }
18    void init() {
19       setLayout(new FlowLayout()
20       add(new JLabel("文本框:"));
21       text=new JTextField(10);
22       add(text);
23       add(new JLabel("按钮:"));
24       button=new JButton("确定");
25       add(button);
26       add(new JLabel("选择框:"));
27       checkBox1 = new JCheckBox("喜欢音乐");
28       checkBox2 = new JCheckBox("喜欢旅游");
29       checkBox3 = new JCheckBox("喜欢篮球");
30       add(checkBox1);
31       add(checkBox2);
32       add(checkBox3);
33       add(new JLabel("单选按钮:"));
34       group = new ButtonGroup();
35       radio1 = new JRadioButton("男");
36       radio2 = new JRadioButton("女");
37       group.add(radio1);
38       group.add(radio2);
39       add(radio1);
40       add(radio2);
41       add(new JLabel("下拉列表:"));
42       comBox = new JComboBox();
43       comBox.addItem("音乐天地");
44       comBox.addItem("武术天地");
45       comBox.addItem("象棋乐园");
46       add(comBox);
47       add(new JLabel("文本区:"));
48       area = new JTextArea(6,12);
49       add(new JScrollPane(area));
```




```
a.chap9;

public class Example9_3 {
   public static void main(String args[]) {
      ComponentInWindow win=new ComponentInWindow();
      win.setBounds(100,100,310,260);
      win.setTitle("常用组件");
   }
}
```

例9-3演示

9.3.2 常用中间容器

- 1．JPanel面板：用JPanel创建一个面板，再向这个面板添加组件，然后把这个面板添加到其它容器中。
- 2．JScrollPane滚动窗格：滚动窗格只可以添加一个组件，把一个组件放到一个滚动窗格中，然后通过滚动条来操作该组件。例如，JTextArea组件不自带滚动条，因此就需要把文本区放到一个滚动窗格中。
- 3．JSplitPane拆分窗格：拆分窗格就是被分成两部分的容器。有两种类型：水平拆分和垂直拆分。

使用JPanel容器示例

```
import java.awt.*;
import javax.swing.*;
public class FrameWithPanel {
public static void main(String args[]){
    JFrame fm1=new JFrame("hello");
    JPanel pan=new JPanel();
    pan.setBackground(Color.yellow);
    fm1.add(pan);
    fm1.setVisible(true);
   }
}
```

例JPanelExp演示

（演示右上角按钮）

9.3.3 布局管理器

- 布局管理器（Layout Manager）是用来安排容器中多个组件的位置及大小，以确保GUI中各组件能安排在适当的位置。

- AWT包提供一组用来进行布局管理的类，每个布局管理类对应一种布局策略。当容器的尺寸改变时，布局管理器会自动调整组件的排列。
  - FlowLayout　　流式布局
  - BorderLayout　　边界布局
  - GridLayout　　网格布局
  - CardLayout　　卡片布局
  - BoxLayout　　盒式布局
  - GridBagLayout　　网格包布局
  - null　　空布局

如何设置布局

- 每个容器（Container对象）都有一个与它相关的默认布局管理器。
  - JFrame的缺省布局是 BorderLayout
  - JPanel的缺省布局是 FlowLayout
- 在没有设置新的布局前，在容器中添加组件都按照该容器的缺省布局排列。
- 可以通过setLayout( )方法为容器设置新的布局方式。

FlowLayout

- 流式布局以行为单位依次排列各组件，一行排不下，另起一行排列组件。
- JPanel的默认布局是FlowLayout

- **构造方法**
  - FlowLayout();
  - FlowLayout(int align);
    //align一般取值有：CENTER、LEFT、RIGHT
  - FlowLayout(int align,int hgap,int vgap);
    //hgap和vgap指定组件与容器起始边界以及组件间的水平和垂直间距，默认值为5个像素

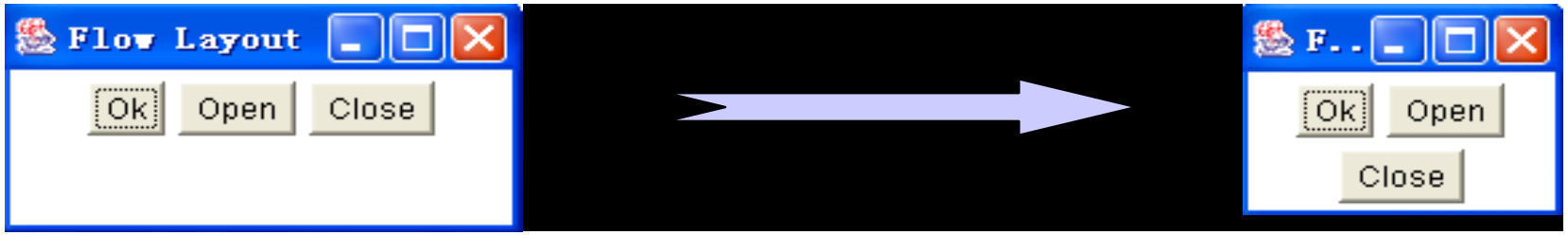
例如：`FlowLayout layout = new FlowLayout(FlowLayout.LEFT, 10, 10);`

FlowLayout布局的使用

- 创建FlowLayout布局对象
  FlowLayout f = new FlowLayout();
- 创建容器对象
  JPanel panel = new JPanel();
- 设置容器对象的布局
  panel.setLayout(f);
- 向容器中添加组件对象
  panel.add(组件对象);

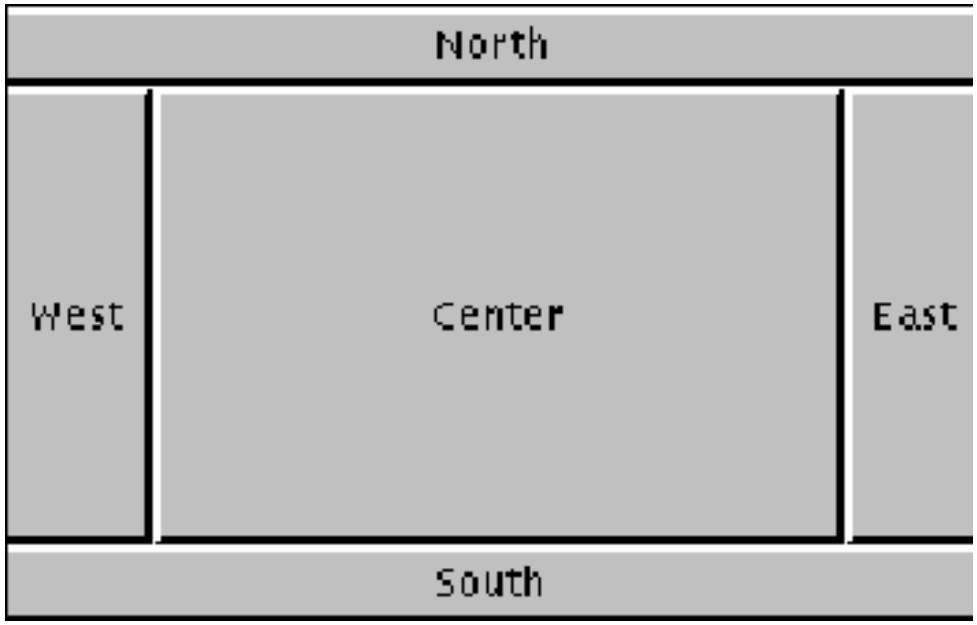
FlowLayout布局的使用

- ✓ **优点：** 能够按照应用需求，一次性将多个基本组件放置于界面中。
- ✓ **缺点：** 当用户对由FlowLayout布局管理的区域进行缩放时，布局发生变化。



BorderLayout

- 边界布局按照东、西、南、北、中5个方位排列组件
- 底层容器JFrame、JDialog、JWindow的默认的布局是BorderLayout



BorderLayout

- **构造方法**
  - BorderLayout();
  - BorderLayout(int hgap,int vgap);
    //hgap和vgap指定组件间水平和垂直间距

例如： BorderLayout lay1 = new BorderLayout();
BorderLayout lay2 = new BorderLayout(10, 10);

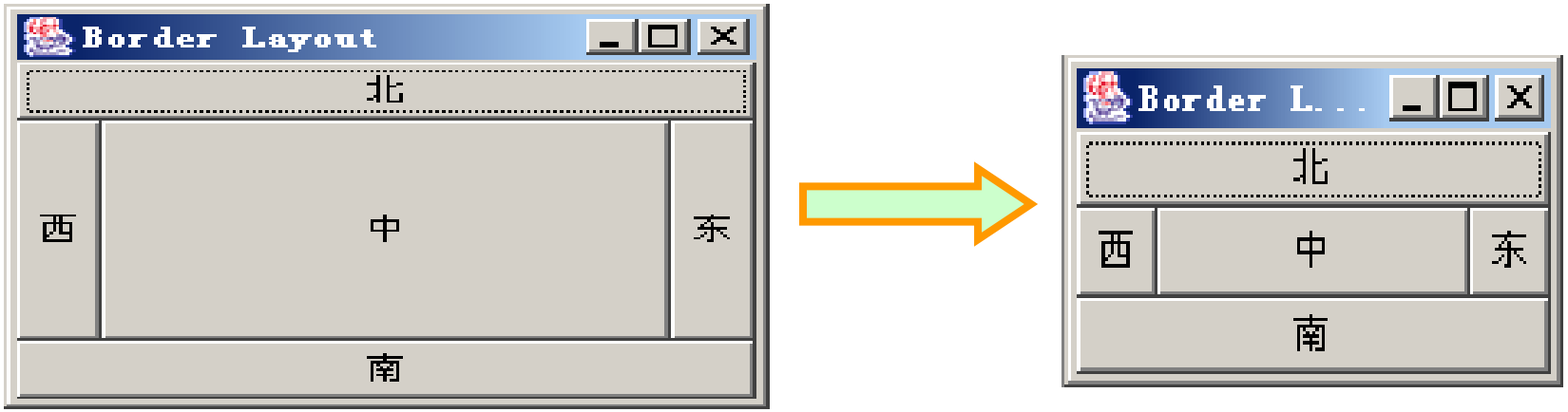
BorderLayout布局的使用

- 创建BorderLayout布局对象
  BorderLayout bl = new BorderLayout();
- 创建容器对象
  JPanel panel = new JPanel();
- 设置容器对象的布局
  panel.setLayout(bl);
- 向容器中添加组件对象
  panel.add(组件对象,方位);

//方位的取值为:
BorderLayout.EAST
BorderLayout.WEST
BorderLayout.SOUTH
BorderLayout.NORTH
BorderLayout.CENTER

BorderLayout布局的使用

- 优点：当容器缩放时，组件相应的位置不变化，但大小改变。
- 缺点：当加入的组件超过5个时，就必须使用容器的嵌套或其它布局。



GridLayout

❖用于将容器区域划分为一个矩形网格，组件按行和列排列

- GridLayout的构造函数如下所示：

- GridLayout():生成一个单列的网格布局
- GridLayout(int row,int col):生成一个设定行数和列数的网格布局
- GridLayout(int row,int col,int horz,int vert): 可以设置组件之间的水平和垂直间隔

```
. . .
JButton btn[]; // 声明按钮数组
String str[]={"1","2","3","4","5","6","7","8","9"};
setLayout(new GridLayout(3,3));
btn=new JButton[str.length]; // 创建按钮数组
for(int i=0;i<str.length;i++){
  btn[i]=new JButton(str[i]);  add(btn[i]);
}
```

例GridLayoutTest演示

CardLayout

- 卡片布局管理器可以让一群组件使用相同的空间，如同一叠卡片，只有最上边的可以被看到。
- 常用的构造函数有：
  - CardLayout():生成一个卡片布局管理器。
- 其他常用的方法如下：

  first(Container parent):显示容器的第一个组件。
  last(Container parent):显示容器的最后一个组件。
  next(Container parent):显示容器的下一个组件。
  previous(Container parent):显示容器的前一个组件。
  show(Container parent, String name):显示容器中名字为name的组件。必须先指定各组件的名字。

CardLayout的使用

- 创建CardLayout布局对象

  CardLayout l = new CardLayout();
- 创建容器对象

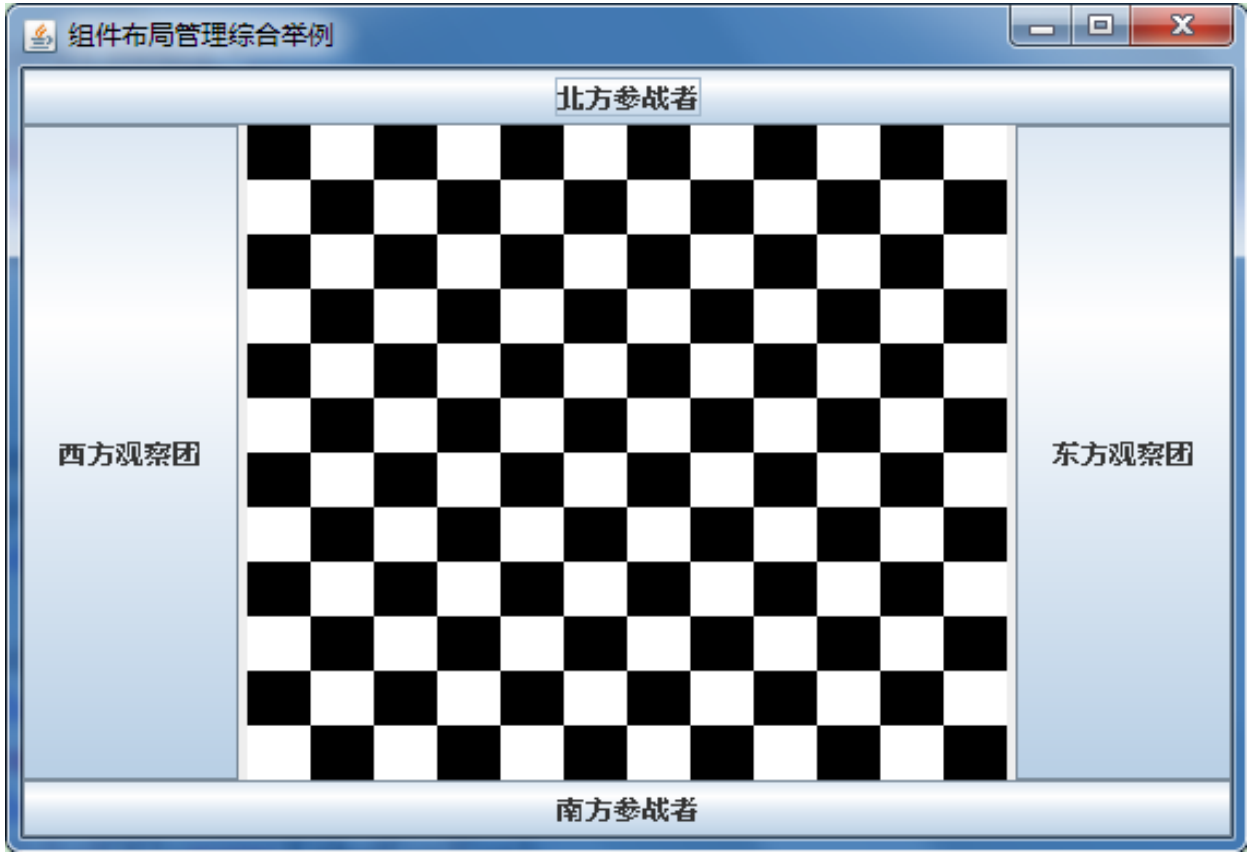
  JPanel panel = new JPanel();
- 设置容器对象的布局

  panel.setLayout(l);
- 向容器中添加组件对象

  panel.add(组件对象，“名称”);
- 显示组件
  - l.show(panel,“名称”);
  - l.first(panel);　　l.last(panel);
  - l.next(panel);　　l.previous(panel);

布局管理器综合应用演示

例9_4综合布局演示



第9章 GUI编程

1. Java Swing概述
2. 窗口
3. 常用组件与布局
4. 事件处理
5. 使用MVC结构
6. 对话框
7. 发布GUI程序

9.4事件处理

- 事件处理机制
  - **事件：**Java语言将每一个鼠标或键盘的操作定义为一个“事件”。

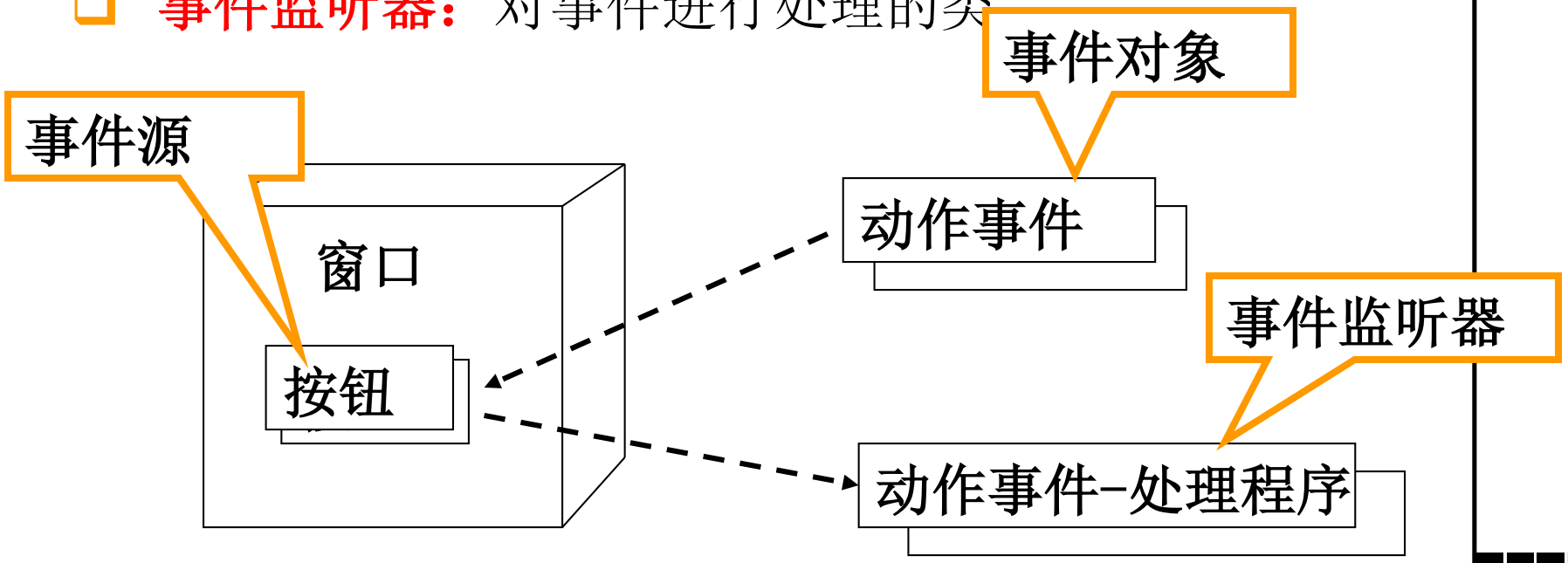
当用户点击了一个按钮，意味着一个按钮事件的发生。

事件处理方法对该事件进行响应

  - **事件响应：**当事件发生时程序应该作出何种响应。

9.4.1 事件处理模式

- 事件处理模式
  - **事件源：**产生事件的组件叫事件源，如文本框、按钮、下拉式列表框等。
  - **事件对象：**描述系统中发生了什么的对象。
  - **事件监听器：**对事件进行处理的类



9.4.1 事件处理模式

Step One：　**创建事件源对象**

创建将要产生事件的组件对象

Step Two：　**创建监听器对象**

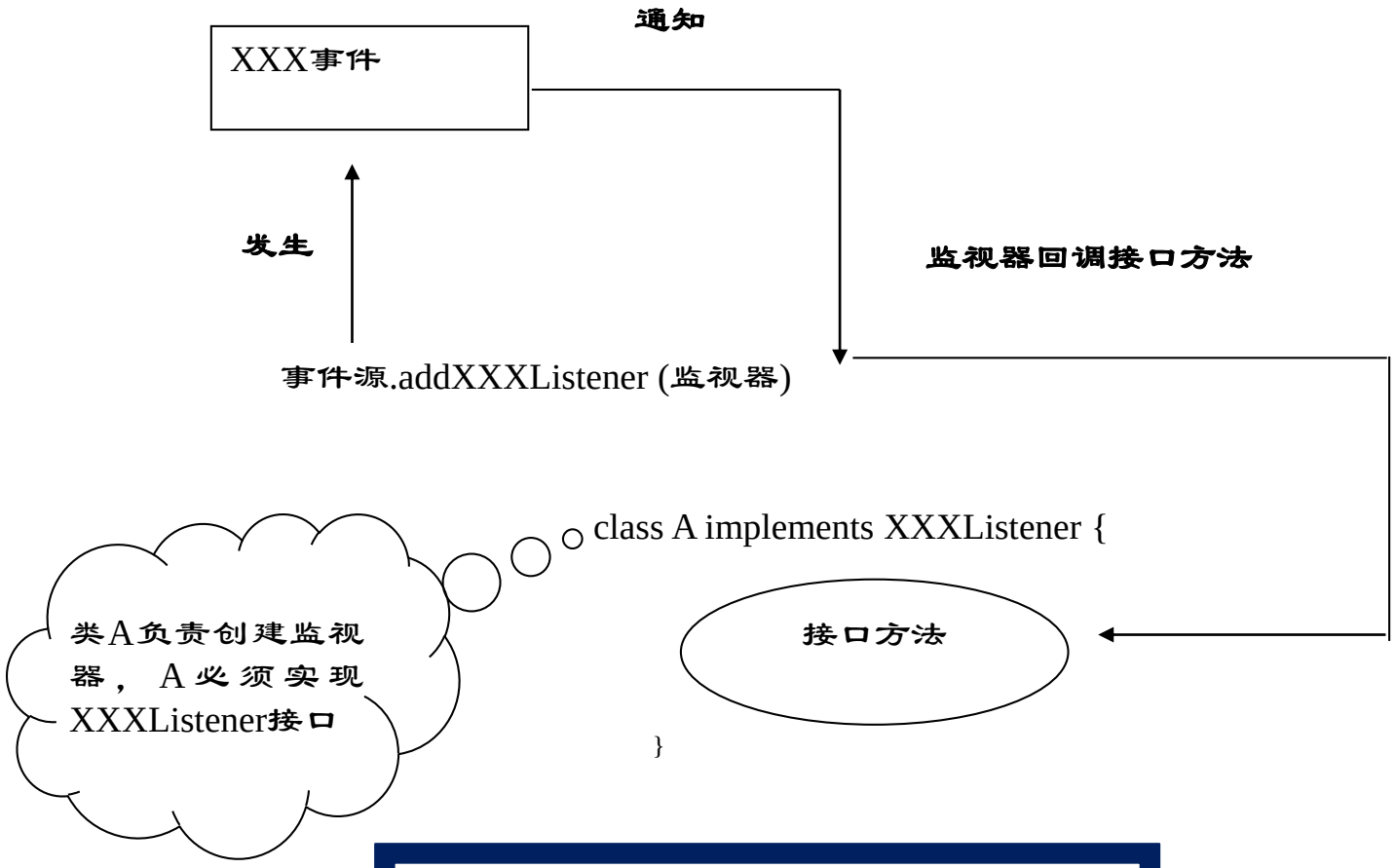
构造实现相应事件监视器接口的类，并创建事件监听器对象

Step Three：**建立事件源与监听器的关联**

为组件对象增加事件监视器对象：

组件对象.addXXXListener(事件监视器对象);

9.4.1 事件处理模式



有哪些XXXListener？

9.4.1 事件监听接口

ActionListener	动作事件监视器接口
ItemListener	选项事件监视器接口
DocumentListener	文档事件监视器接口
MouseListener	鼠标事件监视器接口
MouseMotionListener	鼠标移动事件监视器接口
FocusListener	焦点事件监视器接口
KeyListener	键盘事件监视器接口
ComponentListener	组件事件监视器接口
ContainerListener	容器事件监视器接口
WindowListener	窗口事件监视器接口

9.4.1 事件类型

ActionEvent	动作事件
ItemEvent	选项事件
DocumentEvent	文档事件
MouseEvent	鼠标事件
FocusEvent	焦点事件
KeyEvent	键盘事件

9.4.2 ActionEvent事件

1. ActionEvent事件源

- 文本框、按纽、菜单项、密码框和单选按纽都可以触发ActionEvent事件，即都可以成为ActionEvent事件的事件源。
- 注册了监视器的文本框，在文本框获得输入焦点后，如果用户按回车键，Java运行环境用ActionEvent类创建一个对象，即触发ActionEvent事件。
- 注册了监视器的按纽，如果用户按单击按纽，会触发ActionEvent事件。
- 注册了监视器的菜单项，如果用户按选中该菜单项，就会触发ActionEvent事件。

9.4.2 ActionEvent事件

2．注册监视器

能触发ActionEvent事件源组件用addActionListener(ActionListener listener)将实现ActionListener接口的类创建的对象注册为事件源的监视器。

3．ActionListener接口（实现监视器）

该接口中只有一个方法：

```
public void actionPerformed(ActionEvent e)
```

- 事件源触发ActionEvent事件后，监视器发现触发的ActionEvent事件，然后调用接口中该方法对发生的事件作出处理。
- 当监视器调用actionPerformed(ActionEvent e)方法时，ActionEvent类事先创建的事件对象就会传递给该方法的参数e。

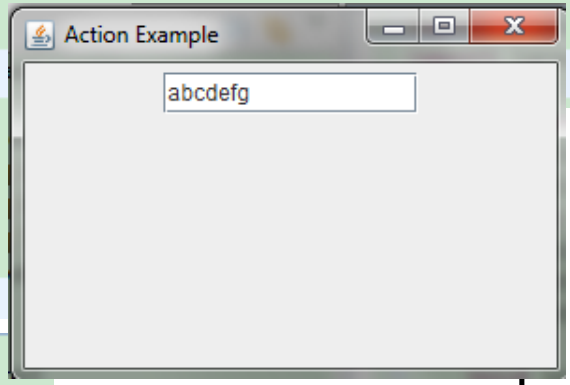
9.4.2 ActionEvent事件

```java
package shu.ces.java.chap9;

import java.awt.*;
public class WindowActionEvent extends JFrame {
    JTextField text;
    ActionListener listener;                //listener是
    public WindowActionEvent() {
        setLayout(new FlowLayout());
        text = new JTextField(10);
        add(text);
        listener = new ReaderListen();        //创建责计算
        text.addActionListener(listener);    //text是事件
        setVisible(true);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

```java
package shu.ces.java.chap9;

public class Example9_6 {
    public static void main(String args[]) {
        WindowActionEvent win=new WindowActionEvent();
        win.setTitle("处理ActionEvent事件");
        win.setBounds(100,100,310,260);
    }
}
```




```java
package shu.ces.java.chap9;

import java.awt.event.*;
public class ReaderListen implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        String str=e.getActionCommand();    //获取封装在事件中的"命令"字符串
        System.out.println(str+":"+str.length());
    }
}
```

例9-6演示

9.4.3 ItemEvent事件

1．ItemEvent事件源

选择框、下拉列表都可以触发ItemEvent事件。

- 对于注册了监视器的选择框，当用户的操作使得选择框从未选中状态变成选中状态，或者从选中状态变成未选中状态时触发ItemEvent事件。
- 对于注册了监视器的下拉列表，当用户选中下拉列表的某个选项，触发ItemEvent事件。

2．注册监视器

能触发ItemEvent事件的组件使用addItemListener(ItemListener listener)将实现ItemListener接口的类的对象，注册为事件源的监视器。

9.4.3 ItemEvent事件

3．ItemListener接口（实现监视器）

java.awt.event包中该接口中只有一个方法：

public void itemStateChanged(ItemEvent e)

- 事件源触发ItemEvent事件后，监视器将发现触发的ItemEvent事件，然后调用接口中该方法对发生的事件作出处理。ItemEvent类事先创建的事件对象传递给该方法的参数e。
- ItemEvent事件对象除了可以使用getSource()方法返回发生ItemEvent事件的事件源外，使用getSelectedItem()方法返回发生事件的事件源选项。

例9-8演示