计算机科学技术学院
School of Computer Science and Technology

Java 程序设计

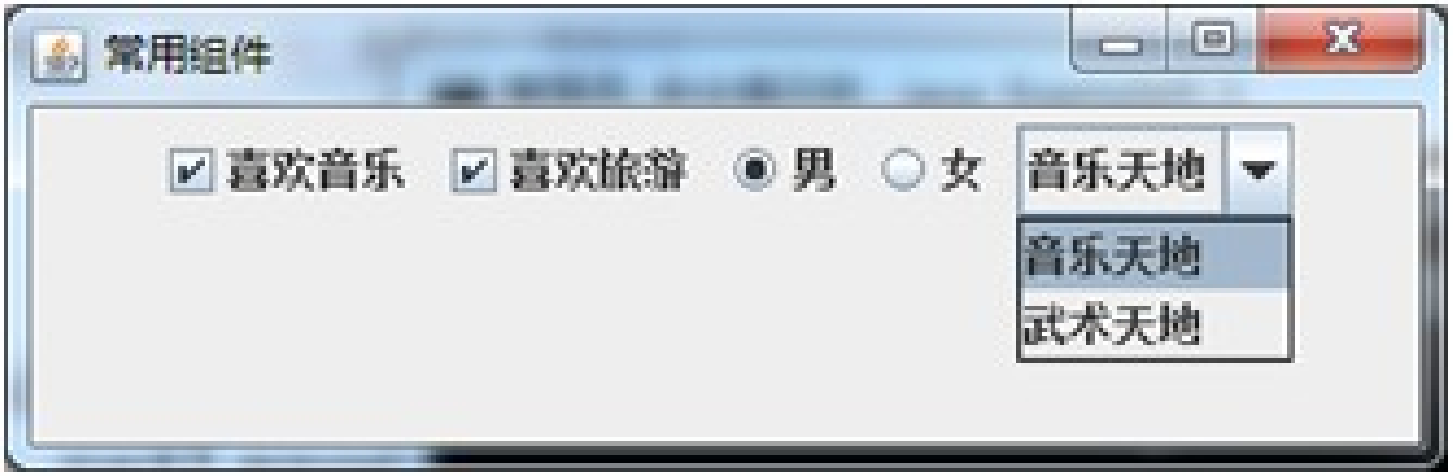
第 9 章 组件及事件处理

导读

- 主要内容
  - Java Swing 概述
  - 窗口
  - 常用组件与布局
  - 处理事件
  - 使用 MVC 结构
  - 对话框
  - 发布 GUI 程序
- 重点和难点
  - 重点： Swing 包中的各种组件，各种布局和事件处理器的应用
  - 难点：各种事件处理器的使用

9.1 Java Swing 概述

- 通过图形用户界面（GUI： Graphics User Interface），用户和程序之间可以方便地进行交互。 Java 的 Swing 工具包中包含了许多类来支持 GUI 设计。如：按钮、菜单、列表、文本框等组件类，同时它还包含窗口、面板等容器类。



9.1 Java Swing 概述

- javax.swing 包提供了功能更为强大的用来设计 GUI 的类。 java.awt 和 javax.swing 包中一部分类的层次关系的 UML 类图

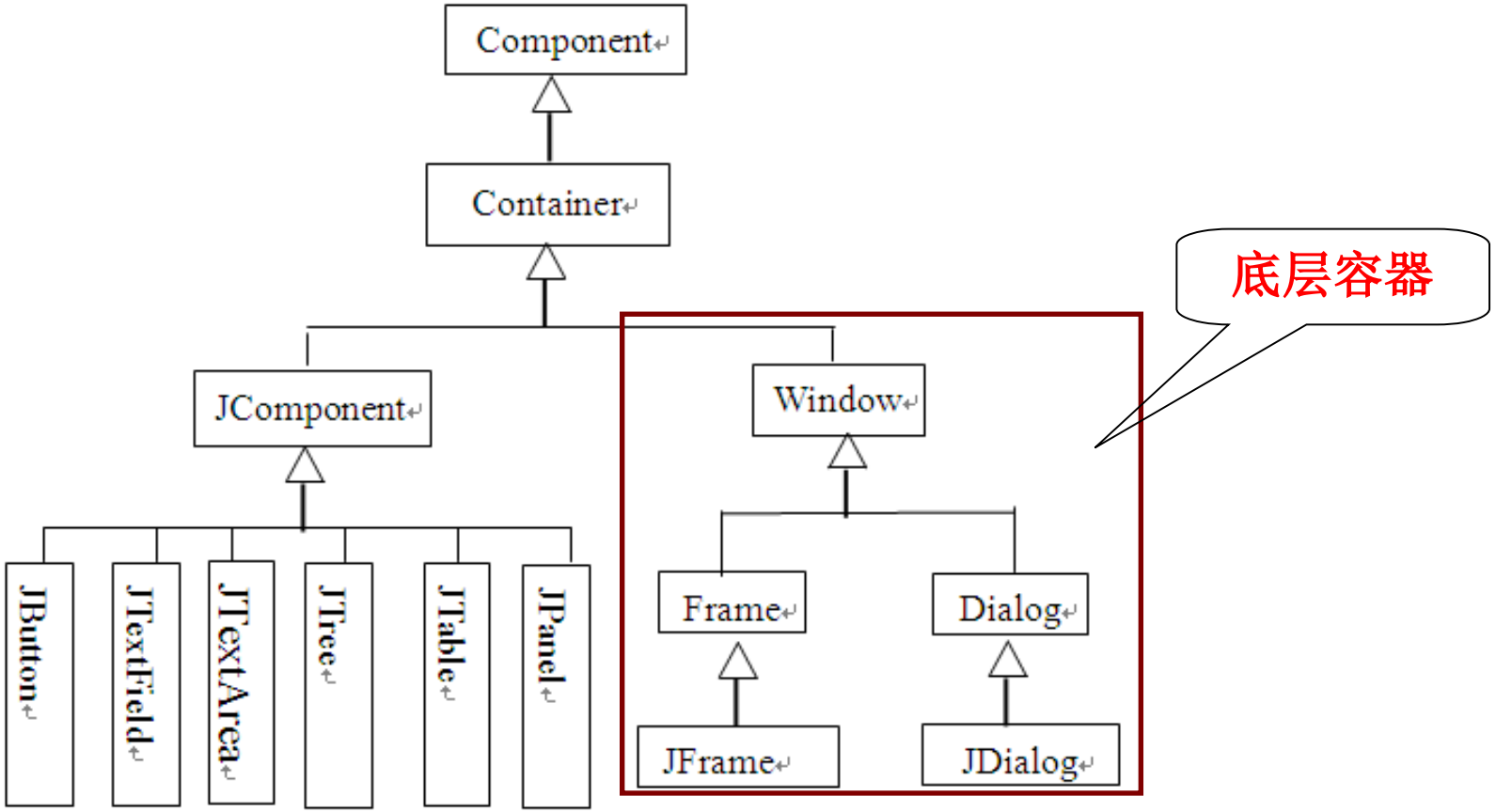


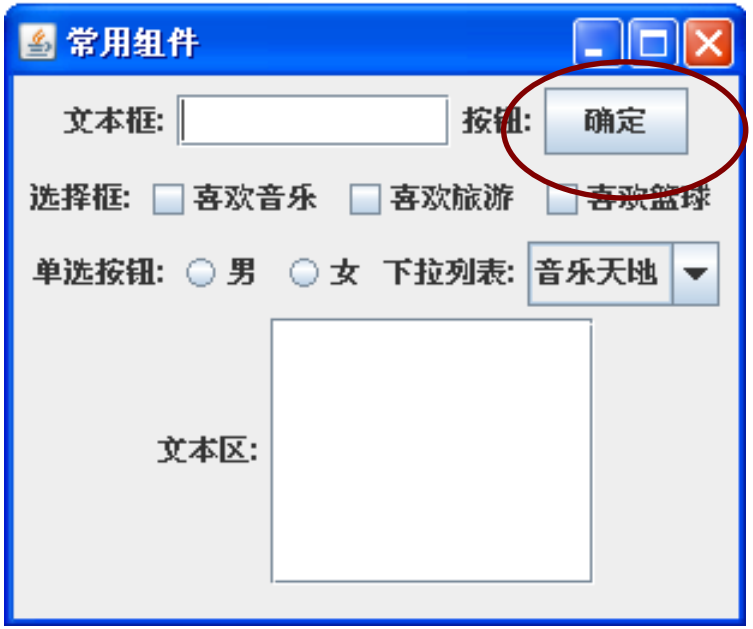
图 9.1 Component 类的部分子类

9.1 Java Swing 概述

- 在学习 GUI 编程时注意两个概念：**容器类**和**组件类**。
  1. Java 把 Component 类的子类或间接子类创建的对象称为一个组件
  2. Java 把 Container 的子类或间接子类创建的对象称为一个容器
  3. 可以向容器添加组件。 Container 类提供了一个 public 方法：add( )
  4. removeAll()/remove(Component c) 方法移掉容器中全部组件或指定组件
  5. 容器本身也是一个组件，容器可嵌套
  6. 每当容器添加新的组件或移掉组件时，应当让容器调用 validate() 方法，以保证容器中的组件能正确显示出来

9.1 Java Swing 概述

```
JFrame f = new JFrame();
JButton  button=new JButton(" 确定")
f.add(button);
…….
f.remove(button);
```



9.2 窗口

- Java 提供的 JFrame 类的实例是一个底层容器，即通常所称的窗口。其他组件必须被添加到底层容器中，以便借助这个地层容器和操作系统进行信息交互。
- JFrame 类是 Container 类的间接子类。当需要一个窗口时，可使用 JFrame 或其子类创建一个对象。
- 窗口也是一个容器，可以向窗口添加组件。需要注意的是，窗口默认地被系统添加到显示器屏幕上，因此不允许将一个窗口添加到另一个容器中。

9.2.1 JFrame 常用方法

- JFrame 构造方法：

  JFrame() 创建一个无标题的窗口

  JFrame(String str) 创建标题为 str 的窗口

9.2.1 JFrame 常用方法

Jframe 常用方法：

序号	方法名	说明
1	public void setBounds(int a, int b, int width, int height)	设置窗口的初始位置是 (a,b)，即距屏幕左面 a 个像素、距屏幕上方 b 个像素；窗口的宽是 width，高是 height
2	public void setSize(int width, int height)	设置窗口的大小
3	public void setLocation(int x, int y)	设置窗口的位置，默认位置是 (0,0)
4	public void setVisible(boolean b)	设置窗口是否可见，窗口默认是不可见的
5	public void setResizable(boolean b)	设置窗口是否可调整大小，默认可调整大小
6	public void dispose()	撤消当前窗口，并释放当前窗口所使用的资源

9.2.1 JFrame 常用方法

Jframe 常用方法：

序号	方法名	说明
7	public void setExtendedState(int state)	设置窗口的扩展状态 .
8	public void setDefaultCloseOperation (int operation)	该方法用来设置单击窗体右上角的关闭图标后，程序会做出怎样的处理。如：EXIT_ON_CLOSE
9	public void setLayout (LayoutManager mgr)	设置此容器的布局管理器
10	public Component add (Component comp)	将指定组件追加到此容器的尾部
11	public void setMenuBar (MenuBar mb)	将此 frame 的菜单栏设置为指定的菜单栏
12	public void validate()	使用 validate 方法会使容器再次布置其子组件

9.2.1　JFrame 常用方法

例1　用 JFrame 创建了两个窗口，程序运行效果如图 9.2 。

图 9.2 创建窗口

9.2.1 JFrame 常用方法

```
static final int WIDTH=300;
static final int HEIGHT=200;

public static void main(String[] args) {
        JFrame jf=new JFrame(" 主窗体 ");
        jf.setSize(WIDTH,HEIGHT);
        jf.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        jf.setVisible(true);
}
```

9.3.3 常用布局_1

- 我们将分别介绍java.awt包中的FlowLayout、BorderLayout、CardLayout、GridLayout布局类和java.swing.border包中的BoxLayout布局类。

- 容器可以使用方法：

  setLayout(布局对象);

  来设置自己的布局，控制组件在容器中的位置

9.3.3 常用布局 _1

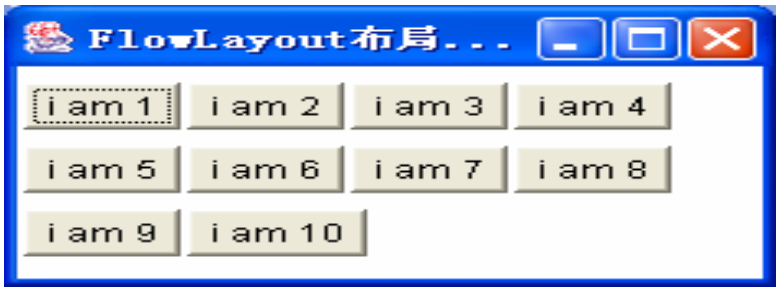
1． FlowLayout 布局： JPanel 型容器的默认布局

①创建布局对象： FlowLayout flow = new FlowLayout();

②容器 con 使用布局对象： con.setLayout(flow);

③con 可以使用 Container 类提供的 add 方法将组件顺序地添加到容器中

FlowLayout 布局对象调用相应的方法可以重新设置布局的对齐方式等． public void setAlignment(int align)



9.3.3 常用布局 _1

1． FlowLayout 布局

- 根据控件加入的先后顺序，从左到右排列。一行排满，再换到下一行，继续从左到右排列，每一行组件均为居中排列。若有些按钮看不到，可使用“pack 方法”自动调整 Frame 的大小，使所有控件都能显示出来

9.3.3 常用布局 _1

1． FlowLayout 布局

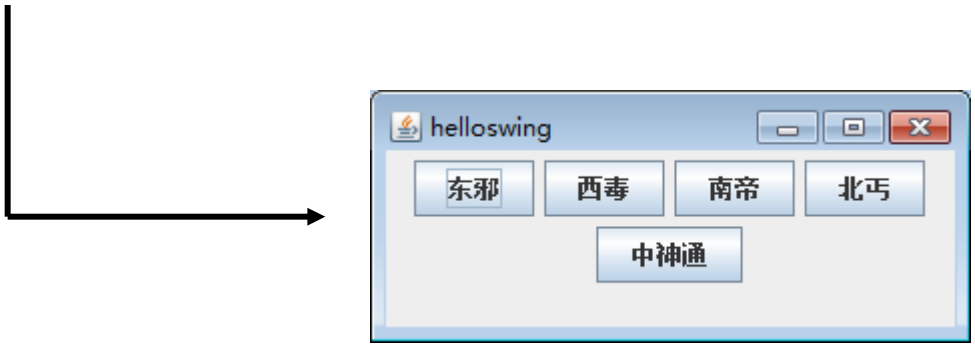
```
JPanel contentPane = new JPanel();
jf.setContentPane(contentPane);
FlowLayout lay = new FlowLayout();
contentPane.setLayout(lay);

JButton btn1 = new JButton("东邪"); contentPane.add(btn1);
JButton btn2 = new JButton("西毒"); contentPane.add(btn2);
JButton btn3 = new JButton("南帝"); contentPane.add(btn3);
JButton btn4 = new JButton("北丐"); contentPane.add(btn4);
JButton btn5 = new JButton("中神通"); contentPane.add(btn5);
```

9.3.3 常用布局 _1

FlowLayout 运行结果：

图　FlowLayout 布局管理器



9.3.3 常用布局 _1

2． BorderLayout 布局

BorderLayout 布局管理器将”容器”分为东、西、南、北、中五个区域，每个组件占据其中某个区域。

这 5 个区域分别被命名为

NORTH 、 WEST 、 EAST 、 CENTER 、 SOUTH ，均被定义为静态常量。（静态常量可以直接引用）

常量	说明
Public static final String NORTH=“North”	整个内容面板的北边
Public static final String WEST=“West”	整个内容面板的西边
Public static final String EAST=“East”	整个内容面板的东边
Public static final String CENTER=“Center”	整个内容面板的中间
Public static final String SOUTH=“South”	整个内容面板的南边

9.3.3 常用布局 _1

2． BorderLayout 布局

BorderLayout 布局是 Window 型容器的默认布局 。

使用 BorderLayout 布局的容器 con， 可以使用 add 方法将一个组件 b 添加到中心区域：

con.add(b, BorderLayout.CENTER);

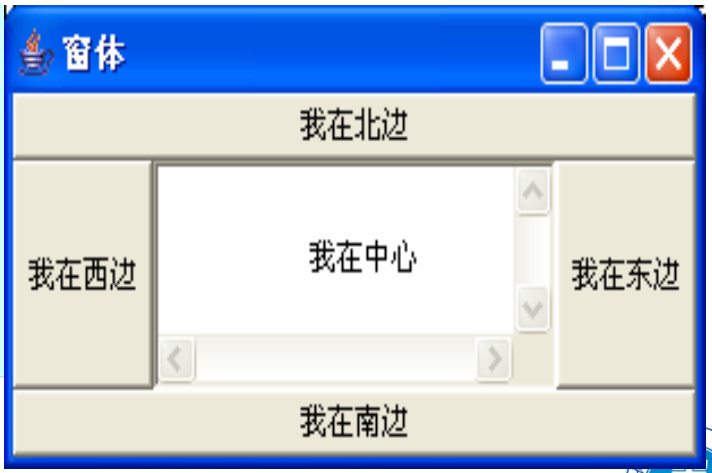
或 con.add(BorderLayour.CENTER, b);

向某个区域添加控件时：

第1个参数。

如：add(组件名称，方位)

参数1 参数2



9.3.3 常用布局 _1

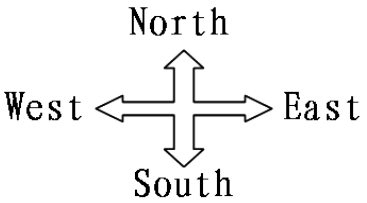
2． BorderLayout 布局

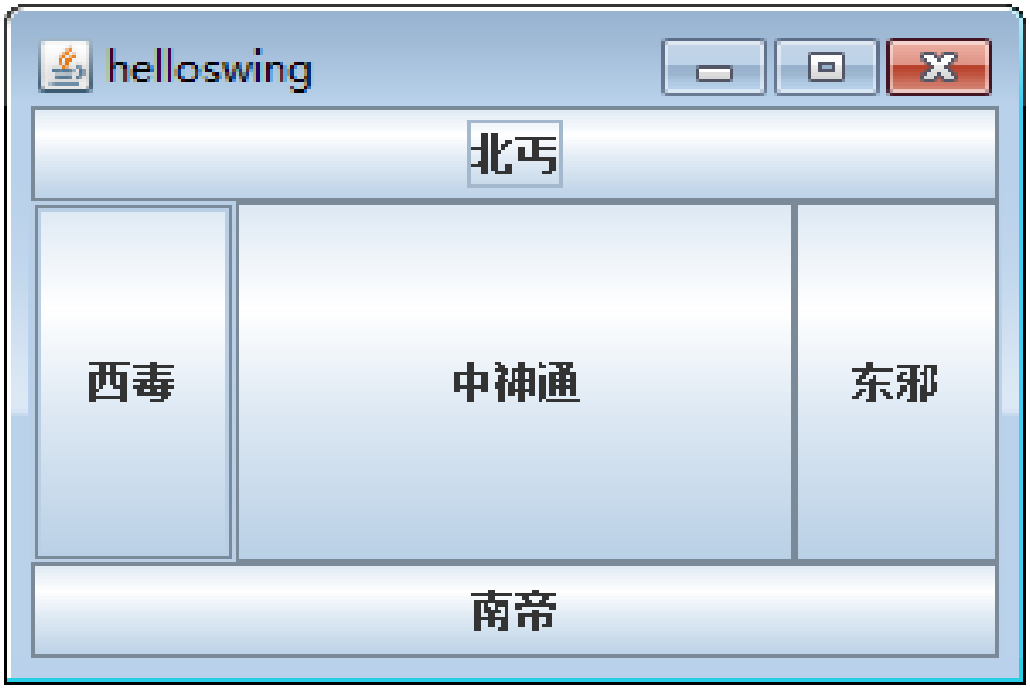
```
JPanel contentPane = new JPanel();
jf.setContentPane(contentPane);
BorderLayout lay = new BorderLayout();
contentPane.setLayout(lay);

JButton btn1 = new JButton(" 北丐 "); contentPane.add(btn1, "North");
JButton btn2 = new JButton(" 西毒 "); contentPane.add(btn2, "West");
JButton btn3 = new JButton(" 南帝 "); contentPane.add(btn3, "South");
JButton btn4 = new JButton(" 东邪 "); contentPane.add(btn4, "East");
JButton btn5 = new JButton(" 中 神 通 "); contentPane.add(btn5,
BorderLayout.CENTER);
```

9.3.3 常用布局 _1

BorderLayout 运行结果：





9.3.3 常用布局 _1

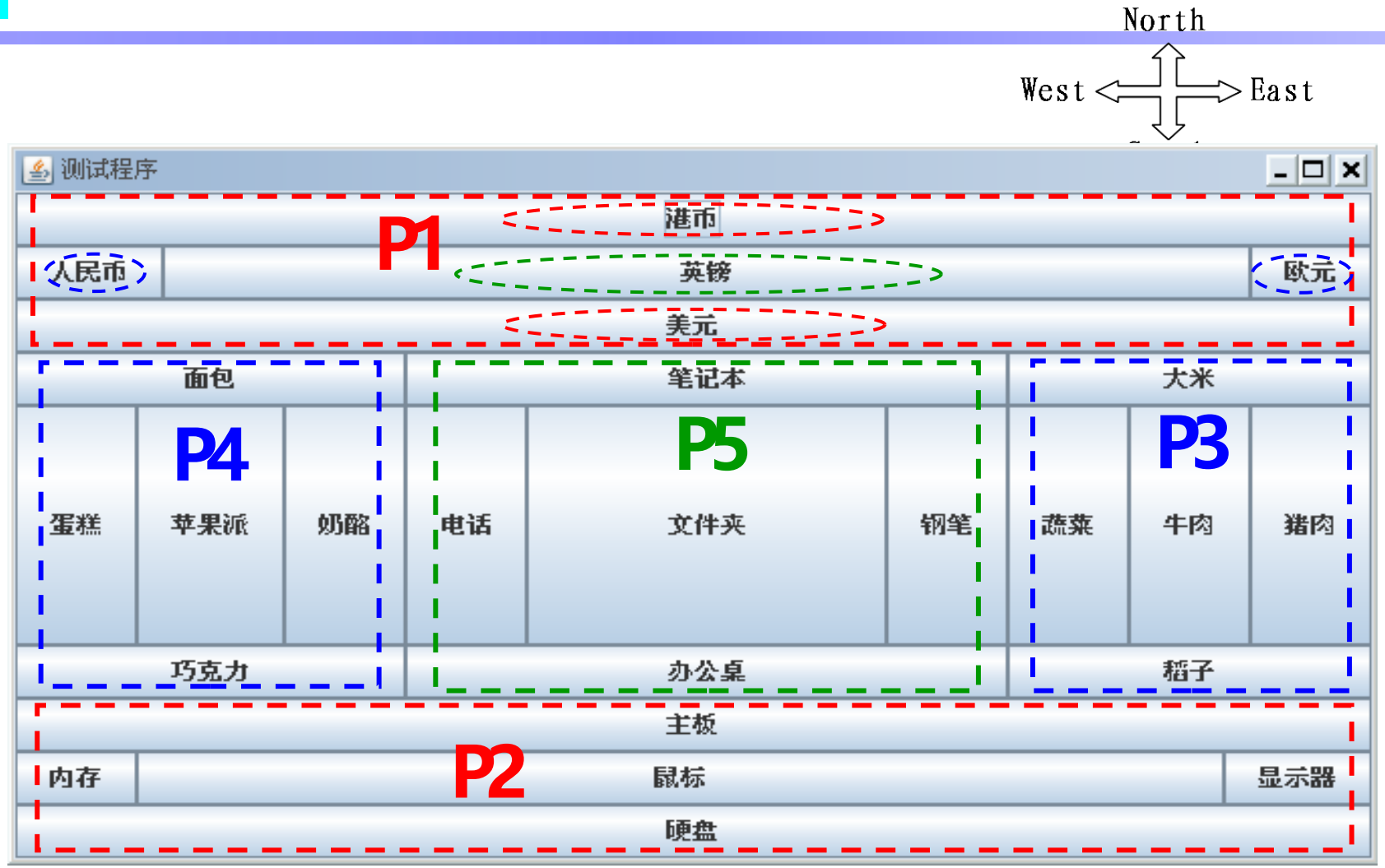


图 复杂的 BorderLayout 布局管理器

9.3.3 常用布局 _1

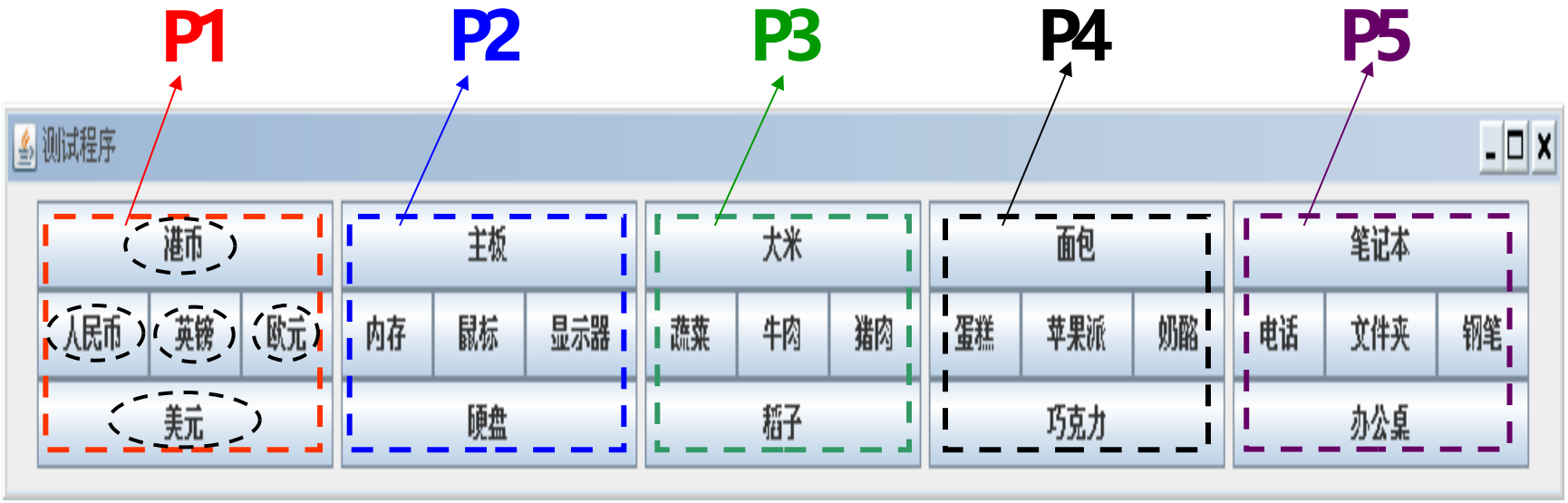


图 5.4 BorderLayout 和 FlowLayout 布局管理器相结合

说明：此例给出两个不同布局管理器，在同一个内容面板中的不同表现形式。也可尝试对调两个布局管理器，看看结果如何

9.3.3　常用布局 _2

3．CardLayout 布局

- 使用 CardLayout 的一般步骤如下：

①创建 CardLayout 对象　CardLayout card=new CardLayout();

②为容器设置布局　con.setLayout(card);

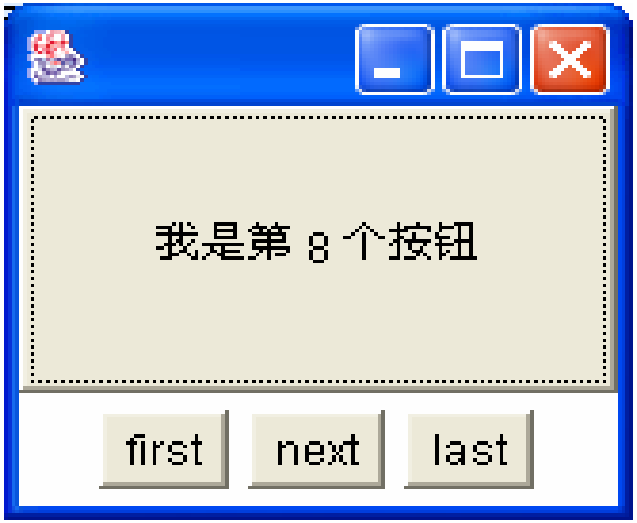
③容器调用 add(String s, Component b) 将组件 b 加入容器，并给出了显示该组件的代号 s 。

④布局对象 card 用 CardLayout 类提供的 show() 方法，显示容器 con 中组件代号为 s 的组件：　card.show(con, s);

9.3.3 常用布局 _2

3． CardLayout 布局

- 使用 CardLayout 的容器可以容纳多个组件，但是实际上同一时刻容器只能从这些组件中选出一个来显示，就像一叠“扑克牌”每次只能显示最上面一张一样，这个被显示的组件将占据所有的容器空间，依次排序



9.3.3 常用布局 _2

4． GridLayout 布局

- GridLayout 布局策略是把容器划分成若干行乘若干列的网格区域，组件就位于这些划分出来的小格中
- GridLayout 布局编辑器的一般步骤如下：

1) 创建布局对象，指定划分网格的行数 m 和列数 n

```
GridLayout grid=new new GridLayout(10, 8);
```

2) 使用 GridLayout 布局的容器调用方法 add(Component c) 将组件 c 加入容器。

9.3.3 常用布局 _2

4． GridLayout

```
GridLayout lay = new GridLayout(3, 3); contentPane.setLayout(lay);
JButton btn1 = new JButton(" 理学 "); contentPane.add(btn1);
JButton btn2 = new JButton(" 光电 "); contentPane.add(btn2);
JButton btn3 = new JButton(" 机电 "); contentPane.add(btn3);
JButton btn4 = new JButton(" 电信 ");   contentPane.add(btn4);
JButton btn5 = new JButton(" 计算机 ");contentPane.add(btn5);
JButton btn6 = new JButton(" 生命 ");contentPane.add(btn6);
JButton btn7 = new JButton(" 经管 ");contentPane.add(btn7);
JButton btn8 = new JButton(" 化工 ");contentPane.add(btn8);
JButton btn9 = new JButton(" 文学 ");contentPane.add(btn9);
// jf.pack();
```

9.3.3 常用布局 _2

程序运行结果：

图 GridLayout 布局管理器

9.3.3 常用布局 _2

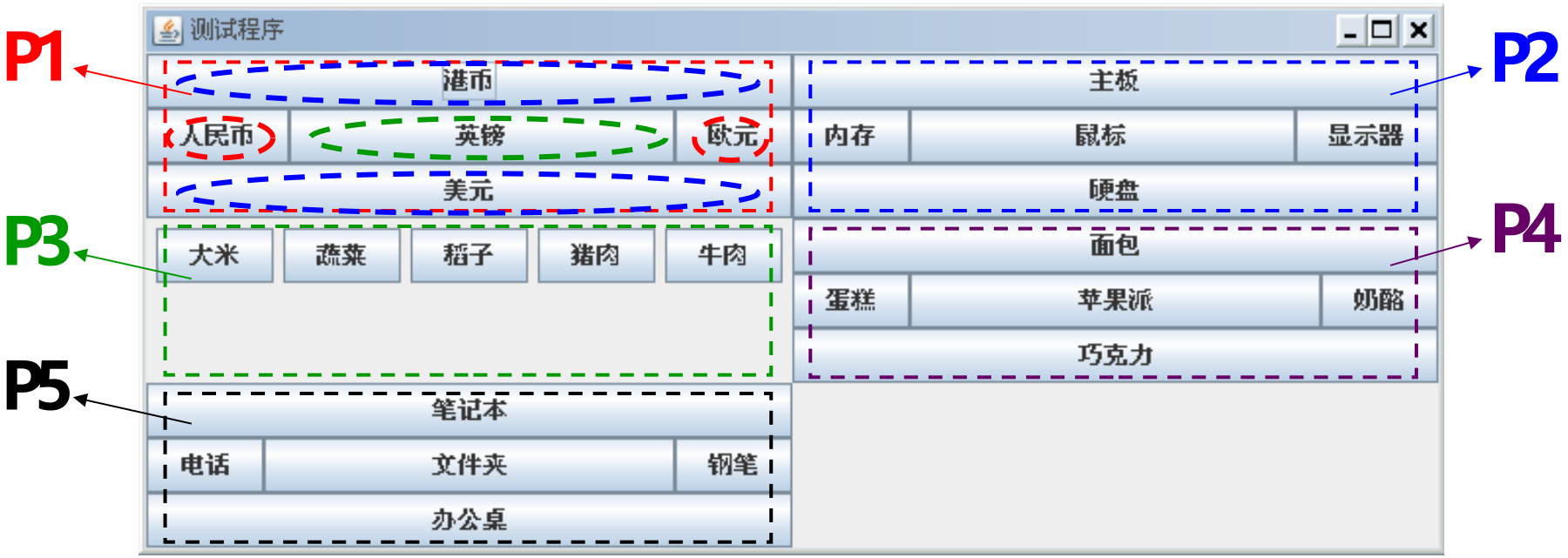


图 GridLayout 、 BorderLayout 、 FlowLayout 布局管理器相结合

9.3.3　常用布局 _2

5．null 布局

- 可以把一个容器的布局设置为 null 布局 (空布局) 。空布局容器可以准确地定位组件在容器的位置和大小

例如， p 是某个容器， c 是要添加的组件

```
p.setLayout(null);
```

①容器 p 使用 add(c) 方法添加组件

②组件 c 再调用 setBounds(int a,int b,int width,int height) 方法设置该组件在容器 p 中的位置和本身的大小

9.3.3　常用布局 _3

6．BoxLayout 布局

- Box 类的类（静态）方法　createHorizontalBox() 获得一个行型盒式容器；
- 使用 Box 类的类（静态）方法　createVerticalBox() 获得一个列型盒式容器。
- 想控制盒式布局容器中组件之间的距离，需使用水平支撑组件或垂直支撑。

例 5，有两个列型盒式容器 boxVOne、boxVTwo 和一个行型盒式容器 boxH。将 boxVOne、boxVTwo 添加到 boxH 中，并在它们之间添加了水平支撑。

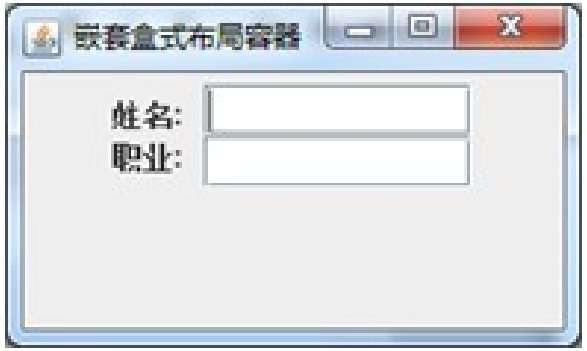


图 9.6　嵌套 Box 容器的窗口

9.3.3 常用布局 _3

```
JFrame jf=new JFrame("BoxLayout 练习 ");
jf.setSize(WIDTH,HEIGHT);
jf.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
jf.setVisible(true);
jf.setLayout(new FlowLayout());
Box b1 = Box.createHorizontalBox();
Box b2 = Box.createHorizontalBox();
b1.add(new JTextField(10));
b1.add(new JButton(" 确定 "));
b2.add(new JTextField(10));
b2.add(new JButton(" 取消 "));
Box b3 = Box.createVerticalBox();
b3.add(b1);
b3.add(b2);
jf.add(b3);
```