



主讲教师 张 智  
计算机学院软件工程系  
课程群: 421694618

## 11 Java Swing编程

### 11.1 [Swing简介](#)

### 11.2 [Swing容器](#)

### 11.3 [布局管理器](#)

### 11.4 [事件监听机制](#)

### 11.5 [常用组件](#)

### 11.6 [Swing示例](#)

## 11.1 Swing简介

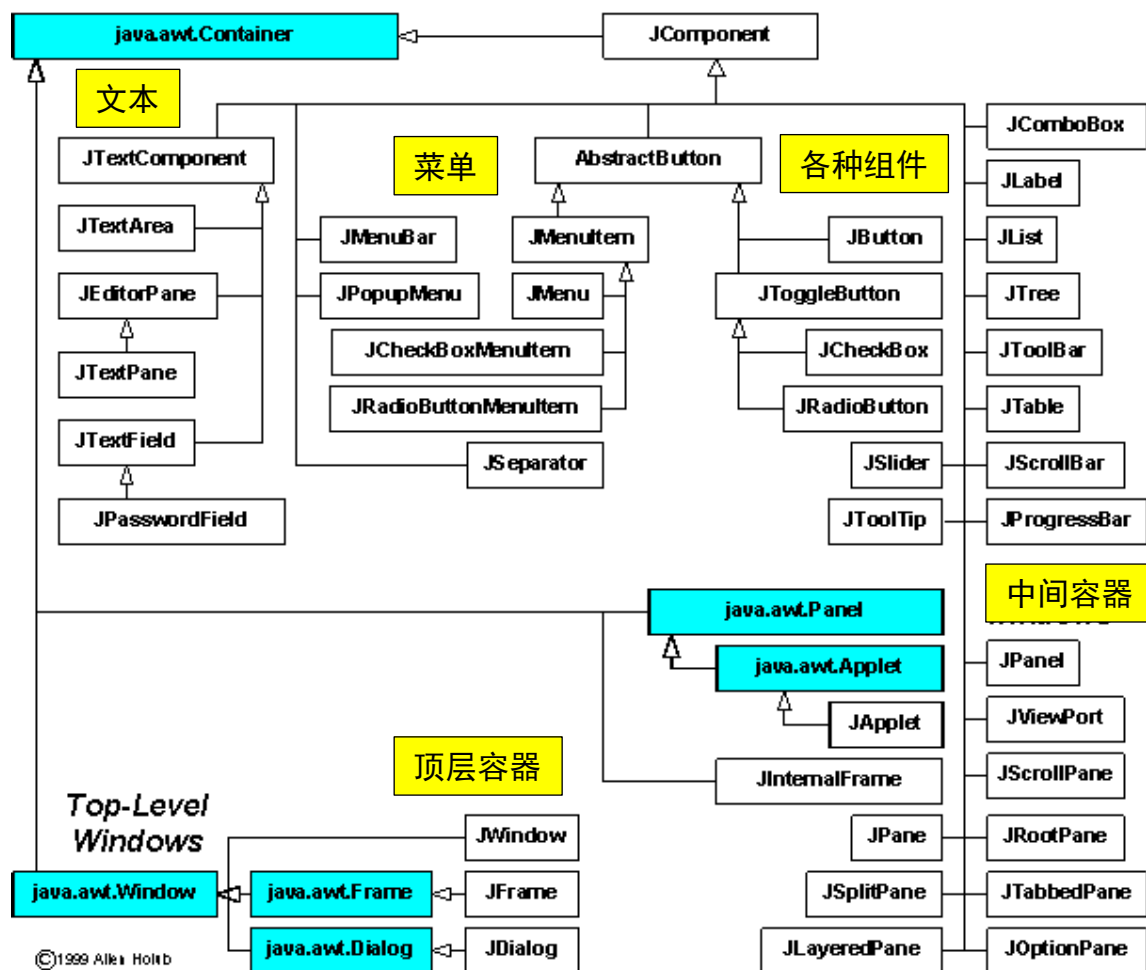
- Swing是一个用于Java **GUI** 编程的工具包(类库), 是在**AWT**的基础上发展起来的
- Swing是轻量级, 采用纯Java实现, 不再依赖本地平台图形界面, 在所有平台上能保持相同的运行效果(**跨平台**)
- Swing组件都采用**MVC**设计, 实现显示逻辑和数据逻辑的分离

图形用户界面

早期GUI工具包

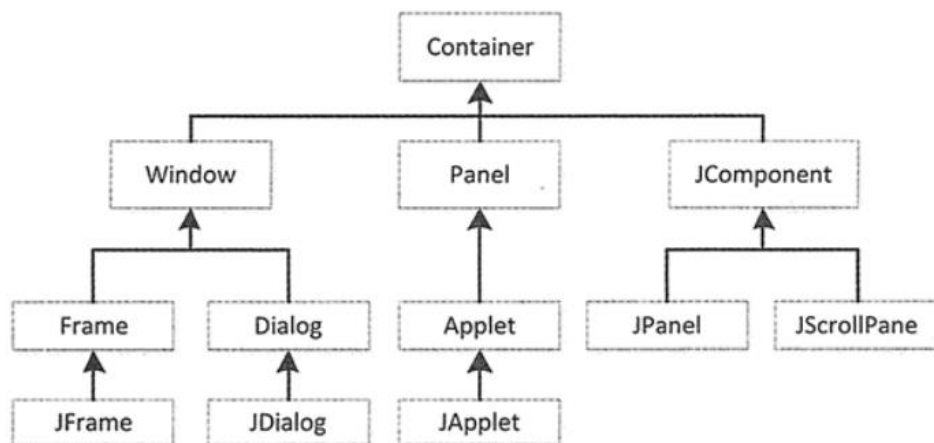
Model-View-Controller, 即模型-视图-控制器

## Swing类库结构

[【返回】](#)

## 11.2 Swing容器

- GUI程序第一步是创建一个容器以容纳其他组件，常见的窗口就是一种容器
- 容器本身也是一种组件，主要用来组织、管理和显示其他组件
- Java程序中容器类都是继承自Container类
- Swing容器可以分为两类：顶层容器和中间容器



## 1. 顶层容器

- 顶层容器是GUI主窗口，是显示并承载组件的容器组件
- 常用的顶层容器：
  - **JFrame**：窗口类，带有边框、标题、关闭和最小化图标。GUI应用程序至少应使用一个JFrame窗口
  - JDialog：对话框类
  - JApplet：运行在Web浏览器内的容器（已淘汰）

## 2. 中间容器

- 中间容器不能独立显示，必须依附于其他的顶层容器
- 常用的中间容器：
  - **JPanel**：表示一个普通面板，是最灵活、最常用的中间容器
  - **JScrollPane**：可在大的组件或可扩展组件周围提供滚动条
  - **JTabbedPane**：选项卡面板
  - **JToolBar**：工具栏

## JFrame常用方法

JFrame是一个顶层容器，JFrame支持多线程

方法	说明
JFrame()	创建一个初始时不可见的新窗体
JFrame(String title)	创建一个具有 title 标题的不可见新窗体
setTitle(String)	设置窗口标题
setDefaultCloseOperation(int option)	设置窗口单击"关闭"按钮时执行的操作，取值：
setResizable(boolean)	设置窗口是否可以调整大小，默认true可调整大小
setVisible(boolean)	true：显示窗口，false：隐藏窗口
setSize(int width, int height)	设置窗口大小，窗口的默认位置 (0,0)
setLocation(int x, int y)	设置出现在屏幕上的位置，(x,y)为坐标
setIconImage(Image image)	设置窗口图标
setJMenuBar(JMenuBar menubar)	设置此窗体菜单栏
setContentPane(JPanel)	设置窗口内容面板
setLayout(LayoutManager manager)	设置布局管理器
pack()	调整窗口大小，以适合其内部组件的首选大小和布局

- 3: JFrame.EXIT\_ON\_CLOSE (结束程序) 默认
- 2: JFrame.DISPOSE\_ON\_CLOSE (回收窗口)
- 1: JFrame.HIDE\_ON\_CLOSE (隐藏窗口)
- 0: JFrame.DO\_NOTHING\_ON\_CLOSE (无作为)



## JPanel常用方法

JPanel是一个中间容器

方法	说明
Component add(Component comp)	将指定的组件追加到此容器的尾部
void remove(Component comp)	从容器中移除指定的组件
void setFont(Font f)	设置容器的字体
void setLayout(LayoutManager mgr)	设置容器的布局管理器
void setBackground(Color c)	设置组件的背景色

## JFrame示例

```
public class Test {  
    public static void main(String[] args) {  
        JFrame frame = new JFrame("测试");  
        frame.setSize(400,300);  
        frame.setLocation(200,200);  
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
        JLabel label = new JLabel("Hello Swing!");  
        frame.add(label);  
        frame.setVisible(true);  
    }  
}
```

创建带标题的窗口

设置窗口大小

设置窗口位置

补充窗口居中显示: `frame.setLocationRelativeTo(null);`

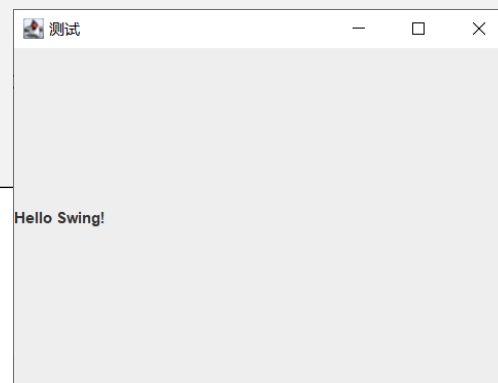
设置"关闭"按钮操作

新建一个JLabel标签组件

将JLabel组件添加到窗口中

设置窗口可见（放在最后面）

运行结果



## 代码改进:

GUI是单线程设计，应该使用事件派发机制调度线程去执行，避免造成死锁

```
public class Test {  
    public static void main(String[] args) {  
        EventQueue.invokeLater( new Runnable() {  
            @Override  
            public void run() {  
                JFrame frame = new JFrame("测试");  
                frame.setSize(400, 300);  
                frame.setLocation(200, 200);  
                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
                JLabel label = new JLabel("Hello Swing!");  
                frame.add(label);  
                frame.setVisible(true);  
            }  
        });  
    }  
}
```

使用事件派发机制启动JFrame更安全



## JFrame另一种写法

```
public class Test extends JFrame {  
    JLabel label = new JLabel("Hello Swing!");  
  
    public Test() {  
        this.setSize(400,300);  
        this.setLocationRelativeTo(null); //窗口在屏幕居中对齐  
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
        this.add(label);  
    }  
  
    public static void main(String[] args) {  
        new Test().setVisible(true);  
    }  
}
```

继承JFrame

组件作为类成员

在构造函数中进行初始化，或单独定义一个方法（this可省略）

新建JFrame对象

运行结果

## JPanel示例

```
public class Test {  
    public static void main(String[] args) {  
        JFrame frame = new JFrame("测试");  
        frame.setSize(400, 300);  
        frame.setLocation(200, 200);  
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
  
        JLabel label = new JLabel("这是放在JPanel上的标签");  
        JPanel panel=new JPanel(); ← 创建一个JPanel面板对象  
        panel.setBackground(Color.YELLOW); ← 设置JPanel背景色  
        panel.add(label); ← 将标签添加到JPanel面板  
        frame.add(panel); ← 将JPanel面板添加到窗口 或者使用 frame.setContentPane(panel);  
  
        frame.setVisible(true);  
    }  
}
```

[【返回】](#)

## 11.3 布局管理器

- 容器可以容纳组件，向容器添加组件时，需要考虑组件位置和大小
- 如果不使用布局管理器，则需要事先计算好位置和距离，这种绝对定位实现非常麻烦
- **Java布局管理器：为实现跨平台特性并获得动态布局效果，从而加快开发速度**

## Java布局管理器类

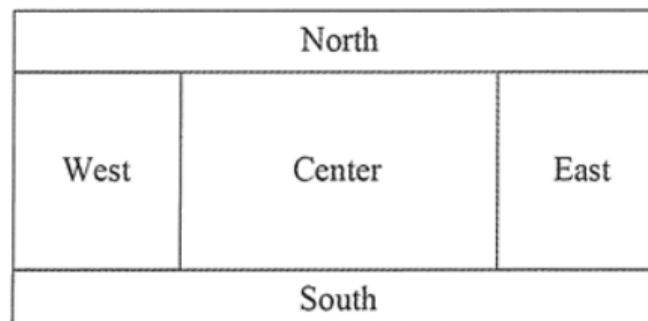
- 所有布局都实现 `LayoutManager` 接口：

- `BorderLayout`
- `FlowLayout`
- `CardLayout`
- `GridLayout`
- `BoxLayout`
- `GridBagLayout` (重要)

**【[返回](#)】**

## 1. BorderLayout (边框布局管理器)

- BorderLayout是JFrame、JDialog默认的布局管理器
- BorderLayout 把容器空间划分为东、西、南、北、中5个区域：
  - BorderLayout.EAST
  - BorderLayout.WEST
  - BorderLayout.SOUTH
  - BorderLayout.NORTH
  - BorderLayout.CENTER



- 添加组件时, 如果不指定区域, 则默认放在Center区域
- BorderLayout不要求所有区域都必须有组件, 如果四周区域(North、South、East、West)没有组件, 则由Center区域去补充
- 如果单个区域中添加的不只一个组件, 那么后来添加的组件将覆盖原来的组件, 所以, 区域中只显示最后添加的一个组件



## BorderLayout构造器

- BorderLayout(): 创建一个Border布局, 组件之间没有间隙
- BorderLayout(int hgap,int vgap): 创建一个Border布局, hgap表示组件间横向间隔; vgap表示组件间纵向间隔, 单位是像素

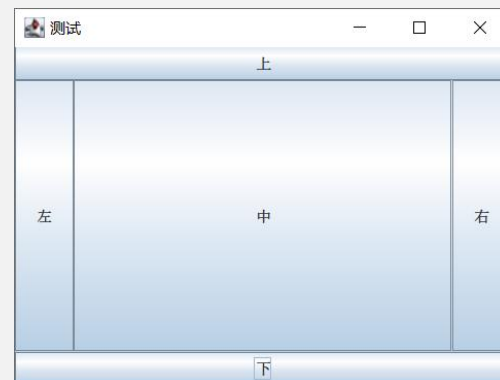
## BorderLayout示例

```
JFrame frame = new JFrame("测试");
frame.setSize(400, 300);
frame.setLocation(200, 200);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setLayout( new BorderLayout() ); ← JFrame设置BorderLayout布局
JButton button1=new JButton ("上");
JButton button2=new JButton("下");
JButton button3=new JButton("左");
JButton button4=new JButton("右");
JButton button5=new JButton("中");

frame.add( button1, BorderLayout.NORTH );
frame.add( button2, BorderLayout.SOUTH );
frame.add( button3, BorderLayout.WEST );
frame.add( button4, BorderLayout.EAST );
frame.add( button5, BorderLayout.CENTER );

frame.setVisible(true);
```

运行结果



[【返回】](#)

## 2. FlowLayout（流式布局管理器）

- FlowLayout是JPanel默认的布局管理器
- FlowLayout会将组件按照从上到下、从左到右的放置规律逐行进行定位，默认情况下，容器中的每一行组件都居中对齐
- FlowLayout不限制组件的大小，允许它们有自己的最佳大小

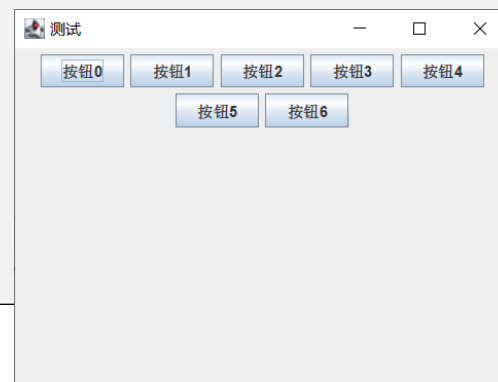
## FlowLayout构造器

- `FlowLayout()`: 创建一个Flow布局管理器，默认使用居中对齐和5像素的水平和垂直间隔
- `FlowLayout(int align)`: `align`表示组件对齐方式，取值：`FlowLayout.LEFT`、`FlowLayout.RIGHT`和`FlowLayout.CENTER`
- `FlowLayout(int align, int hgap,int vgap)`: `align`表示组件对齐方式，`hgap`表示组件间横向间隔；`vgap`表示组件间纵向间隔，单位是像素

## FlowLayout示例

```
JFrame frame = new JFrame("测试");  
frame.setSize(400, 300);  
frame.setLocation(200, 200);  
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
frame.setLayout( new FlowLayout() ); ← JFrame设置BorderLayout布局  
for(int i=0;i<7;i++){  
    frame.add( new JButton("按钮"+i) );  
}  
frame.setVisible(true);
```

运行结果



[【返回】](#)

### 3. CardLayout（卡片布局管理器）

- 实现多个成员共享同一显示空间，且一次只显示一个容器组件的内容
- CardLayout布局管理器将容器分成许多层，每层的显示空间占据整个容器的大小

## CardLayout常用方法

方法	说明
<code>CardLayout()</code>	创建一个Card布局，默认间隔为0
<code>CardLayout(int hgap, int vgap)</code>	创建一个Card布局，hgap:水平间隔，vgap:垂直间隔
<code>first(Container)</code>	显示第一张卡片
<code>last(Container)</code>	显示最后一张卡片
<code>next(Container)</code>	显示下一张卡片（循环）
<code>previous(Container)</code>	显示上一张卡片（循环）
<code>show(Container parent, String name);</code>	显示指定名称的组件（添加组件到容器时，可同时添加组件名称）

```
JFrame frame = new JFrame("测试");
frame.setSize(400, 300);
frame.setLocation(200, 200);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
JPanel p1=new JPanel(); //面板1，默认流式布局
p1.add(new JButton("登录"));
p1.add(new JButton("注册"));
p1.add(new JButton("找回密码"));
```

```
JPanel p2=new JPanel(); //面板2，默认流式布局
p2.add(new JTextField("用户名",20));
p2.add(new JTextField("密码",20));
```

```
JPanel cardPanel = new JPanel( new CardLayout() ); //创建使用card布局的面板
cardPanel.add( p1, "card1" ); //在card布局面板中添加面板1,并设置卡片名称
cardPanel.add( p2, "card2" ); //在card布局面板中添加面板2,并设置卡片名称
```

// 先显示第1个card

```
CardLayout layout = (CardLayout) ( cardPanel.getLayout() ); //获得cardPanel容器的布局
layout.show(cardPanel,"card1"); //调用show()方法显示面板1
```

```
frame.add(cardPanel); //将card布局的面板放到窗口中
frame.setVisible(true);
```

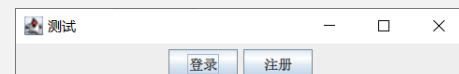
// 每间隔2秒切换显示下一个card（循环）

```
new Timer(2000, new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        layout.next( cardPanel );
    }
}).start();
```

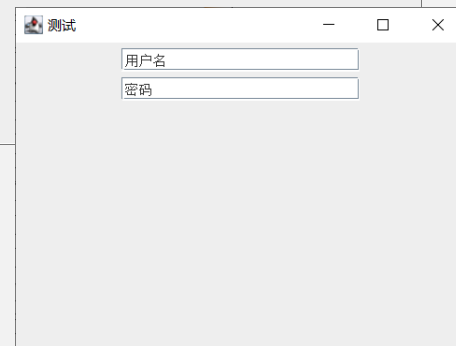
Lambda表达式实现：

```
new Timer(2000, (e)->layout.next(cardPanel)).start();
```

初始显示第一张card



2秒之后,切换到第二张card



[【返回】](#)



## 4. GridLayout（网格布局管理器）

- 将区域分割成行数(rows)和列数(columns)的网格状布局，组件按照由左至右、由上而下的次序排列填充到各个单元格中
- GridLayout所有单元格都是矩形的，且大小相等(自动平分)，每个格子只能存放一个组件，组件大小一样

## GridLayout构造器

- `GridLayout(int rows,int cols)`: 创建一个指定行数(rows)和列数(cols)的网格布局, 布局中所有组件的大小一样, 组件之间没有间隔
- `GridLayout(int rows,int cols,int hgap,int vgap)`: `hgap`表示组件间横向间隔; `vgap`表示组件间纵向间隔, 单位是像素

```
JFrame frame = new JFrame("测试");
frame.setSize(400, 300);
frame.setLocation(200, 200);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

//创建一个panel面板，设置布局为GridLayout，4行4列，间隙为5

```
JPanel panel = new JPanel( new GridLayout(4,4,5,5) );
```

```
panel.add( new JButton("7") );    //添加按钮
```

```
panel.add( new JButton("8") );
```

```
panel.add( new JButton("9") );
```

```
panel.add( new JButton("/") );
```

```
panel.add( new JButton("4") );
```

```
panel.add( new JButton("5") );
```

```
panel.add( new JButton("6") );
```

```
panel.add( new JButton("*") );
```

```
panel.add( new JButton("1") );
```

```
panel.add( new JButton("2") );
```

```
panel.add( new JButton("3") );
```

```
panel.add( new JButton("-") );
```

```
panel.add( new JButton("0") );
```

```
panel.add( new JButton(".") );
```

```
panel.add( new JButton("=") );
```

```
panel.add( new JButton("+") );
```

```
JPanel panel2 = new JPanel();
```

```
panel2.add( new JLabel("运算结果： ") );
```

```
panel2.add( new JTextField(20) );
```

```
frame.add( panel, BorderLayout.CENTER ); //添加面板1到窗口Center
```

```
frame.add( panel2, BorderLayout.NORTH ); //添加面板2到窗口North
```

```
frame.setVisible(true);
```

运行结果



[【返回】](#)

## 5. BoxLayout（盒子布局管理器）

- BoxLayout通常和Box容器联合使用
- Box类两个常用方法：
  - Box.createHorizontalBox(): 创建一个水平盒子布局的Box对象，组件在容器内从左到右排列
  - Box.createVerticalBox(): 创建一个垂直盒子布局的Box对象，组件在容器内从上到下进行排列

## Box内部组件间的空隙

- Box内部组件之间默认没有空隙并居中，如果想在组件之间（或头部/尾部）添加空隙，可以在其中添加一个影响布局的不可见组件
- Box提供三种用于填充空隙的不可见组件：`glue`、`struts`、`rigidAreas`

↪ 具体用法见下页

创建glue(胶水)的方法--宽或高自动值	说明
<code>Component hGlue = Box.createHorizontalGlue();</code>	创建一个水平方向胶状的不可见组件，用于撑满水平方向剩余的空间（如果有多个该组件，则平分剩余空间）
<code>Component vGlue = Box.createVerticalGlue();</code>	创建一个垂直方向胶状的不可见组件，用于撑满垂直方向剩余的空间（如果有多个该组件，则平分剩余空间）
<code>Component glue = Box.createGlue();</code>	创建一个水平和垂直方向胶状的不可见组件，用于撑满水平和垂直方向剩余的空间（如果有多个该组件，则平分剩余空间）

创建struts(框架)的方法--宽或高固定值	说明
<code>Component hStrut = Box.createHorizontalStrut(int width);</code>	创建一个固定宽度的不可见组件
<code>Component vStrut = Box.createVerticalStrut(int height);</code>	创建一个固定高度的不可见组件

创建rigidAreas的方法-宽高固定值	说明
<code>Component rigidArea = Box.createRigidArea(new Dimension(int width, int height));</code>	创建固定宽高的不可见组件

```
JFrame frame = new JFrame("测试");
frame.setSize(600,300);
frame.setLocationRelativeTo(null); //窗口居中
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
JButton btn1 = new JButton("Button01");
JButton btn2 = new JButton("Button02");
JButton btn3 = new JButton("Button03");
JButton btn4 = new JButton("Button04");
JButton btn5 = new JButton("Button05");
```

// 创建一个水平盒子容器

```
Box hBox01 = Box.createHorizontalBox();
```

```
hBox01.add(btn1);
```

```
hBox01.add(btn2);
```

```
hBox01.add( Box.createHorizontalGlue() ); // 添加一个水平胶水
```

```
hBox01.add(btn3);
```

```
hBox01.add( Box.createHorizontalStrut(20) ); // 添加一个宽度为20的水平框架
```

// 创建一个垂直盒子容器

```
Box vBox01 = Box.createVerticalBox();
```

```
vBox01.add(btn4);
```

```
vBox01.add( Box.createVerticalStrut(50) ); // 添加一个高度为50的垂直框架
```

```
vBox01.add(btn5);
```

// 创建一个水平盒子容器，放置上面两个Box

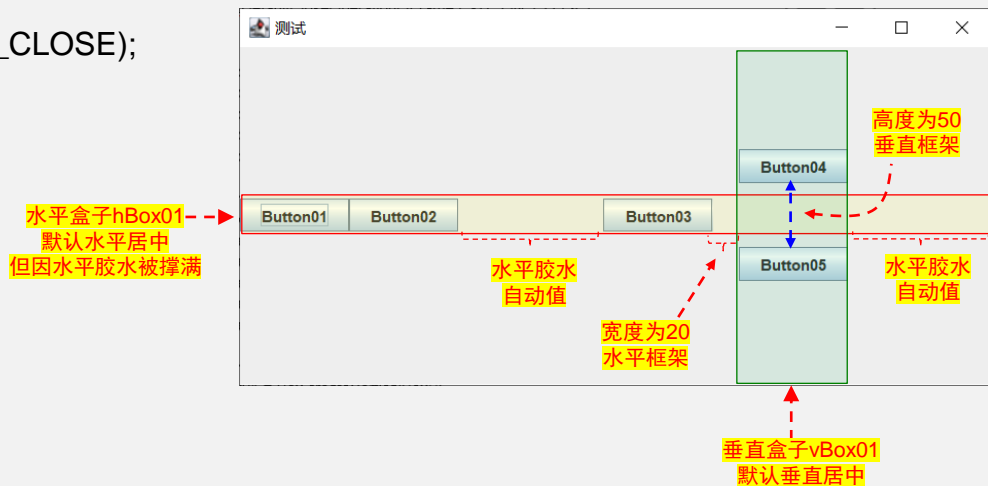
```
Box hBox = Box.createHorizontalBox();
```

```
hBox.add(hBox01);
```

```
hBox.add(vBox01);
```

```
frame.add( hBox ); //把水平盒子放到窗口，默认Center位置
frame.setVisible(true);
```

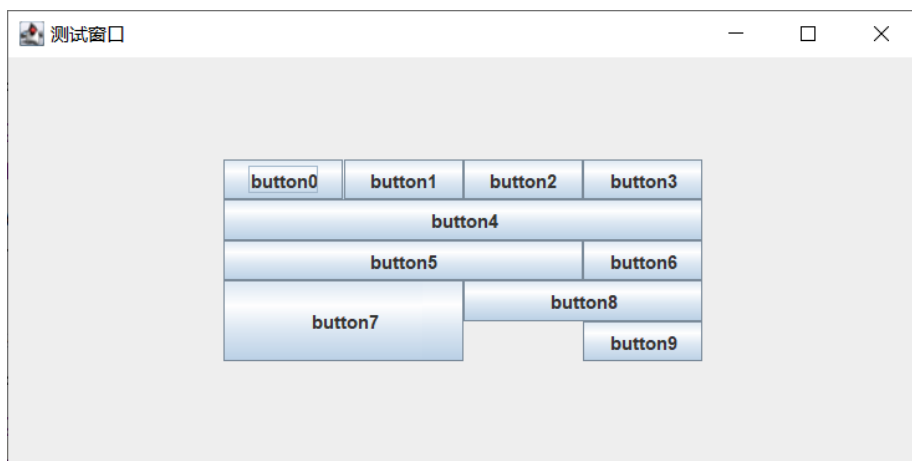
运行结果（窗口宽度600，高度300）



[【返回】](#)

## 6. GridBagLayout（网格袋布局管理器）

- 更灵活，更复杂的布局，允许组件扩展到多行多列
- GridBagLayout所管理的每个组件都与一个GridBagConstraints约束对象相关，这个约束对象指定了组件在网格中的位置、填充方式等





# GridBagConstraints常用属性(约束对象)

常用属性	说明
gridx、gridy	用来指定组件在网格中的列和行(坐标索引)，第一列gridx=0，第一行gridy=0，例如：gridx=2 gridy=3表示横向第3个纵向第4个网格。 注：默认gridx=-1(RELATIVE值)，表示紧跟在上一个组件之后 默认gridy=-1(RELATIVE值)，表示紧跟在上一个组件之下一行
gridwidth、gridheight	用来指定组件显示区域所占的列数(宽度)和行数(高度)，以网格单元为单位，默认值为 1 <b>注意：0值(REMAINDER值)</b> ，如 gridwidth=0，表示组件是横向最后一个组件，通常用来控制换行 <b>-1值(RELATIVE)</b> ，如gridwidth=-1，表示组件是横向倒数第二个组件（较少使用）
fill	指定组件填充网格的方式，取值：GridBagConstraints.NONE（默认值，不拉伸，组件保持原来大小） GridBagConstraints.HORIZONTAL（组件横向充满显示区域，但是不改变组件高度） GridBagConstraints.VERTICAL（组件纵向充满显示区域，但是不改变组件宽度） GridBagConstraints.BOTH（组件横向、纵向充满其显示区域） <b>注：当格子大于组件时才控制拉伸</b>
ipadx、ipady	指定组件显示区域的内部填充，即在组件最小尺寸之外需要附加的像素数，默认值为0
insets	指定组件显示区域的外部填充，即组件与其显示区域边缘之间的空间，默认值为0
anchor	指定组件在显示区域中的摆放位置，取值：总共有9个方位，分别是东、南、西、北、中（GridBagConstraints.CENTER默认值）、东北、西北、东南、西南，英文略
weightx、weighty	用来指定在容器大小改变时，增加或减少的空间如何在组件间分配，默认值为 0，即所有的组件将聚拢在容器的中心，多余的空间将放在容器边缘与网格单元之间。weightx 和 weighty 的取值一般在 0.0 与 1.0 之间，数值大表明组件所在的行或者列将获得更多的空间(如=1，则沾满剩余空间)

```

JFrame frame = new JFrame("测试窗口");
frame.setSize(600, 300);
frame.setLocationRelativeTo(null);
frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);

```

// 10个按钮

```

JButton[] btns = new JButton[10];
for (int i = 0; i < 10; i++) {
    btns[i] = new JButton("button" + i);
}

```

//将设计为5\*4表格

```

GridBagLayout layout = new GridBagLayout(); //新建GridBagLayout布局对象
JPanel panel = new JPanel(layout); //panel设置布局
GridBagConstraints c = null; //定义GridBagConstraints约束对象

```

// 第1行4个按钮

```

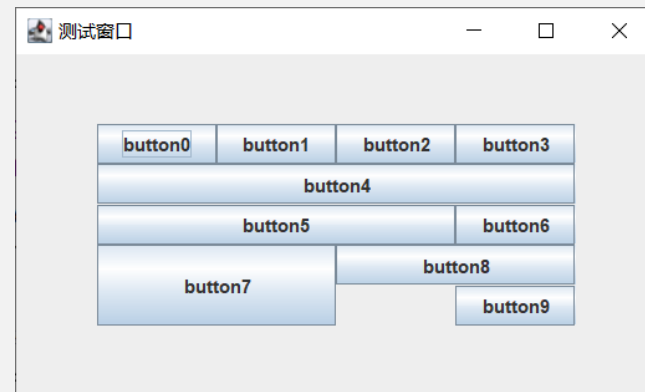
c = new GridBagConstraints(); //约束对象每次重新初始化一次，以免相互影响
c.gridwidth = 1; //宽度为1
layout.setConstraints(btns[0], c); //给组件添加约束
panel.add(btns[0]); //将组件放到panel

```

```

c = new GridBagConstraints(); //约束对象重新初始化
c.gridwidth = 1; //宽度为1
layout.setConstraints(btns[1], c); //给组件添加约束
panel.add(btns[1]); //将组件放到panel

```



```

c = new GridBagConstraints();    //约束对象重新初始化
c.gridwidth = 1;                //宽度为1
layout.setConstraints(btns[2], c); //给组件添加约束
panel.add(btns[2]);              //将组件放到panel

```

```

c = new GridBagConstraints();    //约束对象重新初始化
c.gridwidth = 0;                //0表示最后一个
layout.setConstraints(btns[3], c);
panel.add(btns[3]);

```

// 第2行1个按钮

```

c = new GridBagConstraints();    //约束对象重新初始化
c.gridwidth = 0;                // 0表示最后一个
c.fill = GridBagConstraints.HORIZONTAL; //水平拉伸，默认是不拉伸(保持原有宽度)
//c.anchor=GridBagConstraints.EAST;    //将上一行c.fill注释后测试，结果是东边定位(靠右)
layout.setConstraints(btns[4], c);
panel.add(btns[4]);

```

// 第3行2个按钮

```

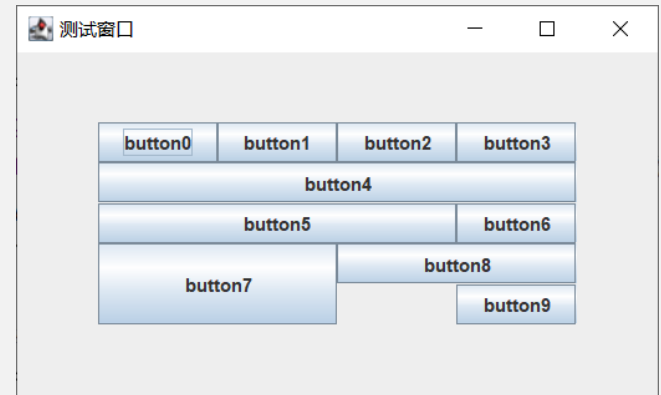
c = new GridBagConstraints();    //约束对象重新初始化
c.gridwidth = 3;                //占3格
c.fill = GridBagConstraints.HORIZONTAL; //水平拉伸
layout.setConstraints(btns[5], c);
panel.add(btns[5]);

```

```

c = new GridBagConstraints();    //约束对象重新初始化
c.gridwidth = 0;                // 0表示最后一个
layout.setConstraints(btns[6], c);
panel.add(btns[6]);

```



// 第4行2个按钮

// 第1个按钮横纵各占2格, 2\*2大小

```
c = new GridBagConstraints(); //约束对象重新初始化
```

```
c.gridwidth = 2; //宽度占2格
```

```
c.gridheight = 2; //高度占2格
```

```
c.fill = GridBagConstraints.BOTH; //两个方向都拉伸
```

```
layout.setConstraints(btns[7], c);
```

```
panel.add(btns[7]);
```

// 第2个按钮横向满2个格子

```
c = new GridBagConstraints(); //约束对象重新初始化
```

```
c.gridwidth = 0; //0表示最后一个
```

```
c.fill = GridBagConstraints.HORIZONTAL; //水平拉伸
```

```
layout.setConstraints(btns[8], c);
```

```
panel.add(btns[8]);
```

//第5行1个按钮

```
c = new GridBagConstraints(); //约束对象重新初始化
```

```
c.gridwidth = 1; //占1格
```

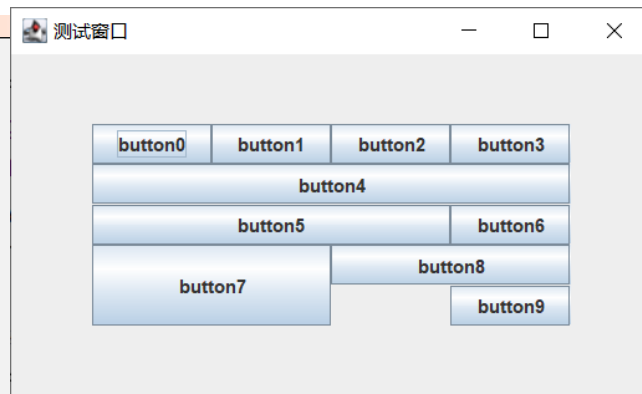
```
c.gridx = 3; //从第4列开始
```

```
layout.setConstraints(btns[9], c);
```

```
panel.add(btns[9]);
```

```
frame.add(panel);
```

```
frame.setVisible(true);
```



```

JFrame frame = new JFrame("测试");
frame.setSize(500, 300);
frame.setLocationRelativeTo(null);
frame.setDefaultCloseOperation(frame.EXIT_ON_CLOSE);

//创建一些组件
JButton btn1 = new JButton("打开");
JButton btn2 = new JButton("保存");
JButton btn3 = new JButton("退出");

String[] str = {"Java", "C#", "HTML5", "CSS", "JS"};
JComboBox choose = new JComboBox(str);

JTextField text = new JTextField(10);
JButton btn4 = new JButton("清空");

JList list = new JList(str);
JScrollPane scrollPane = new JScrollPane(list); //加滚动条
scrollPane.setPreferredSize(new Dimension(80, 60)); //设置滚动条大小，后面会控制拉伸

JTextArea textArea = new JTextArea(3, 10);
textArea.setBackground(Color.GREEN); //为了看出效果，设置了颜色
JScrollPane scrollPane2 = new JScrollPane(textArea); //加滚动条

//假设3*5表格

GridBagLayout layout = new GridBagLayout();
JPanel panel = new JPanel(layout);
GridBagConstraints s = null;

//第1行3个按钮
s = new GridBagConstraints();
s.gridwidth = 1;
layout.setConstraints(btn1, s);
panel.add(btn1);

s = new GridBagConstraints();
s.gridwidth = 1;
layout.setConstraints(btn2, s);
panel.add(btn2);

s = new GridBagConstraints();
s.gridwidth = 0; //表示最后一个
s.anchor = GridBagConstraints.WEST; //靠左（默认是居中）
layout.setConstraints(btn3, s);
panel.add(btn3);

//第2行3个组件
s = new GridBagConstraints();
s.gridwidth = 2;
s.fill = GridBagConstraints.HORIZONTAL; //水平拉伸
layout.setConstraints(choose, s);
panel.add(choose);

s = new GridBagConstraints();
s.gridwidth = 1;

layout.setConstraints(text, s);
panel.add(text);

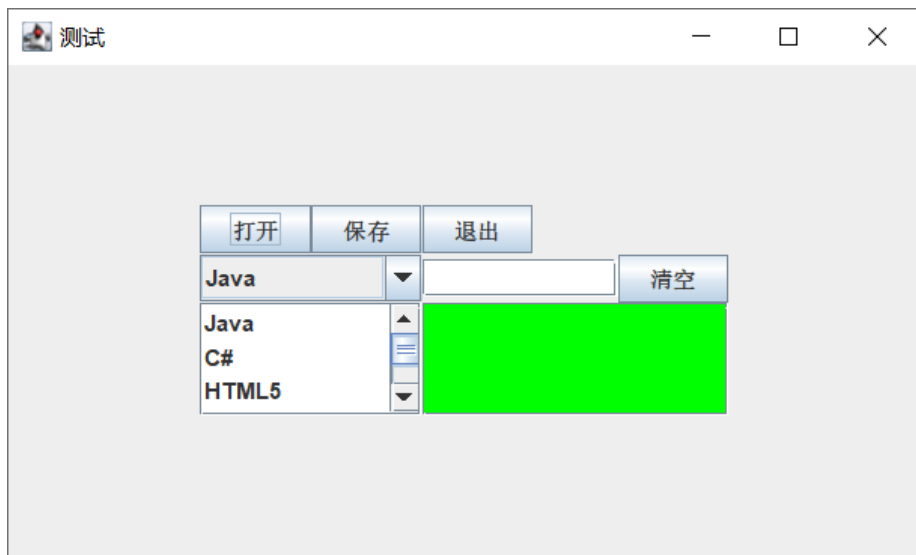
s = new GridBagConstraints();
s.gridwidth = 0; //表示最后一个
s.gridheight = 2;
s.fill = GridBagConstraints.BOTH; //水平和垂直都拉伸
layout.setConstraints(scrollPane, s);
panel.add(scrollPane);

s = new GridBagConstraints();
s.gridwidth = 3;
s.gridheight = 2;
s.fill = GridBagConstraints.BOTH;
layout.setConstraints(scrollPane2, s);
panel.add(scrollPane2);

frame.add(panel);
frame.setVisible(true);

```

## GridBagLayout示例2 -- 运行结果



[【返回】](#)

## 11.4 事件监听机制

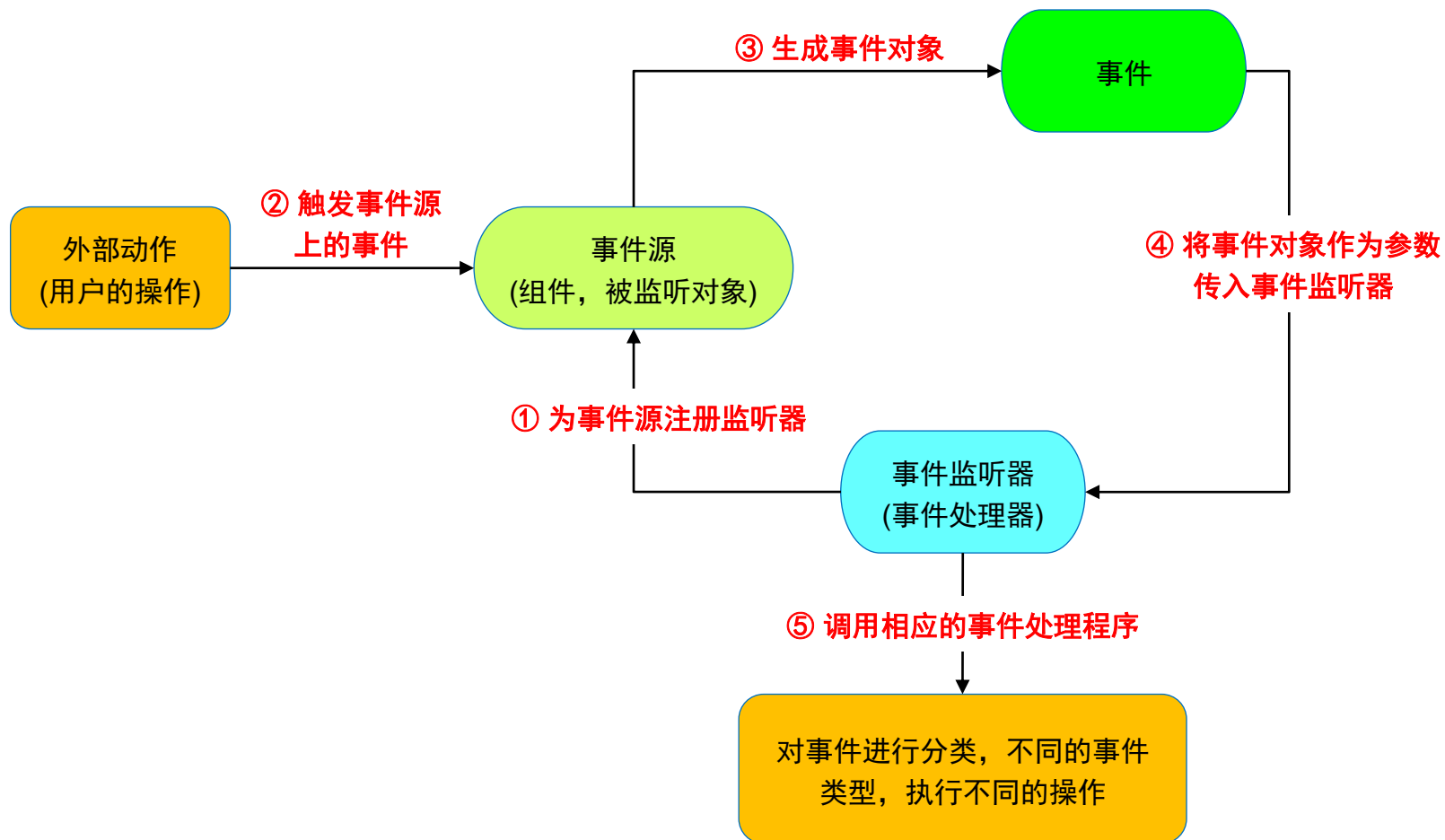
### ■ 事件表示程序和用户之间的交互

- 例如：单击按钮、在文本框中输入、选中复选框或单选框、在列表框中选择等

### ■ 事件处理表示程序对事件的响应

- 当事件发生时，系统会自动捕捉这一事件，同时创建事件对象并把它们传给事件监听器，后者调用相应的事件处理程序代码，这种代码确定了如何处理此事件以使用户得到相应的回答

## 事件监听机制流程





## 事件处理的相关对象

- 在事件处理的过程中，主要涉及三类对象
  - Event(事件): 用户对组件的一次操作称为一个事件，例如：按钮单击事件ActionEvent
  - Event Source(事件源): 事件发生的场所，通常是各个组件，例如：按钮JButton
  - Event Handler(事件处理器): 接收事件对象并对其进行处理，通常是调用某个处理事件的成员方法，例如：actionPerformed()处理方法

## 事件处理框架 -- 以按钮单击事件为例

```
 JButton btn1 = new JButton("测试");  
 btn1.addActionListener( new ActionListener() {  
     @Override                为按钮添加事件监听器(注册)    将事件传入监听器(匿名类实现事件接口)  
     public void actionPerformed(ActionEvent e) {  
         // 代码处理                实现事件处理方法    相应的事件对象  
     }  
 });
```

事件处理程序

## 示例

```
JFrame frame = new JFrame("测试窗口");
frame.setSize(400, 300);
frame.setLocationRelativeTo(null);
frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
JPanel panel = new JPanel(); //创建JPanel对象

JLabel label = new JLabel("hello java!");
label.setFont(new Font("宋体",Font.BOLD,16)); //修改字体样式

JButton btn1 = new JButton("测试");
btn1.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        label.setText("武汉科技大学");
    }
});

panel.add(label);
panel.add(btn1);
frame.add(panel);
frame.setVisible(true);
```

运行结果



[【返回】](#)

## 11.5 常用组件

- [JLabel](#)
- [JButton](#)
- [JTextField](#)    [JPasswordField](#)    [JTextArea](#)
- [JCheckBox](#)    [JRadioButton](#)
- [JComboBox](#)    [JList](#)
- [菜单组件](#)
- [JTable和JTree](#)

**【[返回](#)】**

## 1. JLabel

■ 标签是一种可以包含文本和图片的非交互组件

■ 构造函数： ———→ 创建标签的方法

■ JLabel(): 创建无图像并且标题为空字符串的 JLabel

■ JLabel(Icon image): 创建具有指定图像的 JLabel

■ JLabel(String text): 创建具有指定文本的 JLabel

■ JLabel(String text, Icon image, int hAlign): 创建具有指定文本、图像和水平对齐方式的JLabel, hAlign取值: JLabel.LEFT、JLabel.RIGHT、JLabel.CENTER

## JLabel常用方法

方法	说明
<code>void setText(String text)</code>	设置要显示的单行文本
<code>String getText()</code>	返回所显示的文本字符串
<code>void setIcon(Image image)</code>	定义要显示的图标
<code>Image getIcon()</code>	返回显示的图形图像
<code>void setHorizontalTextPosition(int textPosition)</code>	设置文本相对其图像的水平位置（文本默认在图片右边垂直居中）
<code>void setHorizontalAlignment(int alignment)</code>	设置标签内容沿X轴的对齐方式
<code>void setVerticalTextPosition(int textPosition)</code>	设置文本相对其图像的垂直位置
<code>void setVerticalAlignment(int alignment)</code>	设置标签内容沿Y轴的对齐方式
<code>void setIconTextGap(int iconTextGap)</code>	设置图片和文本之间的间隙
<code>void setFont(Font font)</code>	设置文本的字体类型、样式和大小
<code>void setForeground(Color fg)</code>	设置字体颜色
<code>void setBackground(Color bg)</code>	设置组件的背景
<code>void setToolTipText(String text)</code>	当鼠标移动到组件上时显示的提示文本
<code>void setVisible(boolean visible)</code>	设置组件是否可见
<code>void setOpaque(boolean isOpaque)</code>	设置组件是否为不透明，默认为透明(true)

## JLabel示例

```
JLabel label1 = new JLabel("普通标签");    // 创建带文本的标签
```

```
JLabel label2 = new JLabel();
```

```
label2.setText("调用setText()方法");    // 调用setText()设置文本
```

images文件夹和src在同一个目录

```
ImageIcon img = new ImageIcon("images/login.jpg");
```

```
JLabel label3=new JLabel(img);    // 创建带图片标签
```

```
JLabel label4 = new JLabel();
```

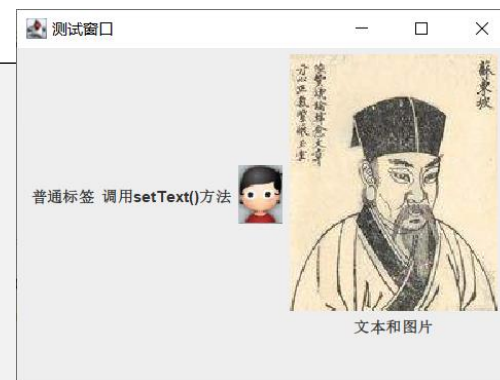
```
label4.setText("文本和图片");
```

```
label4.setIcon(new ImageIcon("images/苏轼.jpg"));
```

```
label4.setHorizontalTextPosition(SwingConstants.RIGHT);    // 水平方向文本居中对齐
```

```
label4.setVerticalTextPosition(SwingConstants.BOTTOM);    // 垂直方向文本在图片下方
```

运行结果



【返回】

## 2. JButton

### ■ 构造函数：

- JButton(): 创建一个无标签文本、无图标的按钮
- JButton(Icon icon): 创建一个无标签文本、有图标的按钮
- JButton(String text): 创建一个有标签文本、无图标的按钮
- JButton(String text,Icon icon): 创建一个有标签文本、有图标的按钮



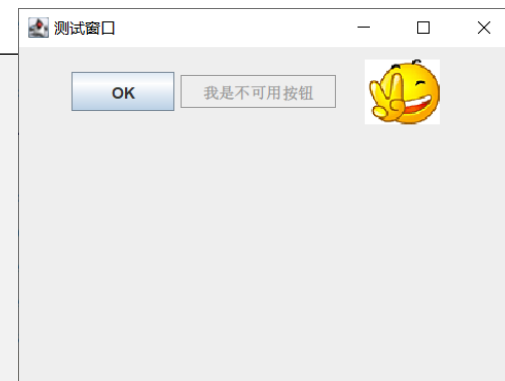
## JButton常用方法

方法	说明
<code>addActionListener(ActionListener listener)</code>	为按钮添加点击事件
<code>void removeActionListener(ActionListener listener)</code>	移除按钮的点击事件
<code>void setText(String text)</code>	设置按钮的文本
<code>void setFont(Font font)</code>	设置按钮的字体
<code>void setForeground(Color fg)</code>	设置按钮的字体颜色
<code>void setIcon(Icon icon)</code>	设置按钮的默认图标
<code>void setPressedIcon(Icon icon)</code>	设置按下按钮时的图标
<code>void setRolloverIcon(Icon icon)</code>	设置鼠标移动到按钮区域时的图标
<code>void setDisabledIcon(Icon icon)</code>	设置按钮无效状态下的图标
<code>void setEnabled(boolean flag)</code>	启用或禁用按钮
<code>setPreferredSize(new Dimension(w,h))</code>	设置组件大小（不建议使用 <code>setSize()</code> ）
<code>void setBorderPainted(boolean b)</code>	是否绘制边框（默认有边框）
<code>setContentAreaFilled(boolean b)</code>	是否绘制默认按钮背景
<code>setFocusPainted(boolean b)</code>	是否绘制图片或文字周围的焦点虚框

## JButton示例

```
JButton btn1=new JButton("OK");  
Dimension preferredSize=new Dimension(80, 30);    //设置尺寸  
btn1.setPreferredSize(preferredSize);  
  
JButton btn2=new JButton("我是不可用按钮");  
tn2.setEnabled(false);    //设置按钮不可用  
  
JButton btn3=new JButton(new ImageIcon("images/1.gif"));  
btn3.setRolloverIcon(new ImageIcon("images/2.gif"));    // 设置鼠标移动到按钮时的图标  
btn3.setBorderPainted(false);    //不绘制边框  
btn3.setContentAreaFilled(false);    //不绘制默认按钮背景  
btn3.setFocusPainted(false);    //不绘制图片或文字周围的焦点虚框
```

运行结果



[【返回】](#)

### 3. JTextField

- 单行文本框，它允许用户输入单行的文本信息
- 构造函数：
  - JTextField(): 创建一个默认的文本框
  - JTextField(String text): 创建一个指定初始文本信息的文本框
  - JTextField(int columns): 创建一个指定列数的文本框
  - JTextField(String text, int columns): 创建一个既指定初始化文本信息，又指定列数的文本框

## JTextField常用方法

方法	说明
String getText()	获取文本框中的文本
void setText(String text) void setFont(Font font) void setForeground(Color fg)	设置文本框的文本、字体 和 字体颜色
void setCaretColor(Color c) void setSelectionColor(Color c) void setSelectedTextColor(Color c) void setDisabledTextColor(Color c)	设置颜色，分别为: 光标颜色、呈现选中部分的背景颜色、选中部分文本的颜色、不可用时文本的颜色
void setHorizontalAlignment(int alignment)	设置文本内容的水平对齐方式
void setEditable(boolean b)	设置文本框是否可编辑
boolean isFocusOwner()	判断组件当前是否拥有焦点
void setEnabled(boolean b)	设置组件是否可用
boolean isFocusOwner()	判断组件当前是否拥有焦点

## JTextField常用方法（续）

复制粘贴相关方法	说明
<code>void setSelectionStart(int selectionStart)</code>	设置光标开始位置, <code>selectionStart &gt;= 0</code>
<code>void setSelectionEnd(int selectionEnd)</code>	设置光标结束位置, <code>selectionEnd &gt;= selectionStart</code>
<code>void copy()</code>	复制选中部分文本
<code>void cut()</code>	剪切选中部分文本
<code>void paste()</code>	粘贴文本到文本框

常用监听器方法	说明
<code>void addFocusListener(FocusListener listener)</code>	添加焦点事件监听器
<code>void addKeyListener(KeyListener listener)</code>	添加按键监听器
<code>textField.getDocument().addDocumentListener(DocumentListener listener)</code>	添加文本框内的文本改变监听器

## JTextField示例

```
JLabel label=new JLabel("输入信息");  
JTextField txtfield = new JTextField(10);    //10列的文本框  
txtfield.setFont(new Font("黑体", Font.BOLD, 12));  
txtfield.setHorizontalAlignment(JTextField.CENTER);    //居中
```

// 文本框添加键盘按键事件

```
txtfield.addKeyListener(new KeyListener() {
```

```
    @Override
```

```
    public void keyTyped(KeyEvent e) {
```

```
        //按键按下后，放开前时发生
```

```
    }
```

```
    @Override
```

```
    public void keyPressed(KeyEvent e) {
```

```
        //按下按键时发生
```

```
    }
```

```
    @Override
```

```
    public void keyReleased(KeyEvent e) {    //松开按键时发生
```

```
        label.setText( txtfield.getText() );
```

```
    }
```

```
});
```

// 按钮点击后获取文本框中的文本

```
JButton btn = new JButton("提交");
```

```
btn.addActionListener(new ActionListener() {
```

```
    @Override
```

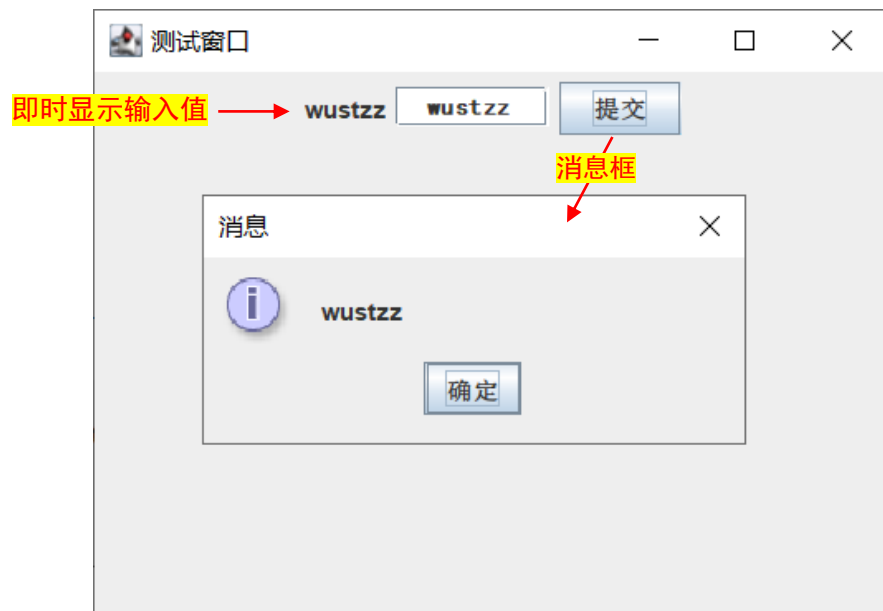
```
    public void actionPerformed(ActionEvent e) {
```

```
        JOptionPane.showMessageDialog(null,txtfield.getText());
```

```
    }
```

```
});
```

## 运行结果



## 4. JPasswordField

- 密码框，继承自JTextField，输入内容时用特定的字符替换显示，用法和 JTextField 基本一致
- 构造函数：
  - JPasswordField(): 创建一个默认密码框
  - JPasswordField(String text): 创建一个指定初始文本信息的密码框
  - JPasswordField(int columns): 创建一个指定列数的密码框
  - JPasswordField(String text, int columns): 创建一个既指定初始文本信息，又指定列数的密码框



## JPasswordField常用方法

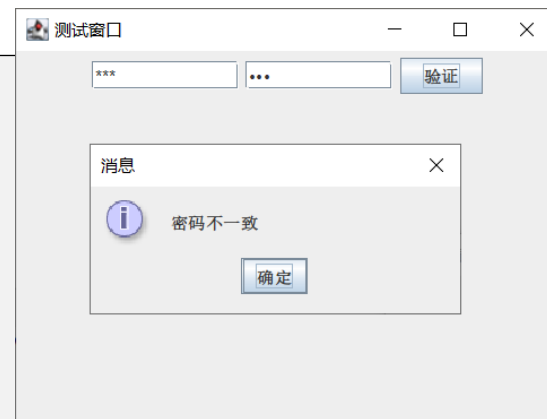
方法	说明
<code>char[] getPassword()</code>	获取密码框输入的密码
<code>void setEchoChar(char c)</code>	设置密码框默认显示的密码字符

## JPasswordField示例

```
JPasswordField pwdField1 = new JPasswordField(10); //10列
pwdField1.setEchoChar('*'); //设置回显字符为'*'
JPasswordField pwdField2= new JPasswordField(10);

// 验证密码是否一致
JButton btn = new JButton("验证");
btn.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        String s1 = new String(pwdField1.getPassword());
        String s2 = new String(pwdField2.getPassword());
        JOptionPane.showMessageDialog(null, s1.equals(s2)?"密码一致":"密码不一致" );
    }
});
```

运行结果



[【返回】](#)

## 5. JTextArea

- 文本域与文本框的最大区别就是文本域允许用户输入多行文本信息
- 构造函数：
  - JTextArea(): 创建一个默认的文本域
  - JTextArea(int rows,int columns): 创建一个具有指定行数和列数的文本域
  - JTextArea(String text): 创建一个包含指定文本的文本域
  - JTextArea(String text,int rows,int columns): 创建一个既包含指定文本，又包含指定行数和列数的多行文本域

## JTextArea常用方法

方法	说明
String getText()	获取文本域中的文本
void setLineWrap(boolean wrap)	设置文本域的换行策略
void setColumns(int columns)	设置文本域的行数
void setRows(int rows)	设置文本域的列数
int getLineCount()	获取内容的行数（以换行符计算，满行自动换下一行不算增加行数）
int getLineEndOffset(int line)	获取指定行（行数从0开始）的行尾（包括换行符）在全文中的偏移量
int getLineOfOffset(int offset)	获取指定偏移量所在的行数（行数从0开始）
void setCaretColor(Color c) void setSelectionColor(Color c) void setSelectedTextColor(Color c) void setDisabledTextColor(Color c)	设置颜色，分别为：光标颜色、呈现选中部分的背景颜色、选中部分文本的颜色、不可用时文本的颜色
void append(String str)	将字符串 str 添加到文本域的最后位置
void insert(String str,int position)	插入指定的字符串到文本域的指定位置
void replaceRange(String str,int start,int end)	将指定的开始位 start 与结束位 end 之间的字符串用指定的字符串 str 取代

## JTextArea常用方法（续）

复制粘贴相关方法	说明
<code>void setSelectionStart(int selectionStart)</code>	设置光标开始位置, <code>selectionStart &gt;= 0</code>
<code>void setSelectionEnd(int selectionEnd)</code>	设置光标结束位置, <code>selectionEnd &gt;= selectionStart</code>
<code>void copy()</code>	复制选中部分文本
<code>void cut()</code>	剪切选中部分文本
<code>void paste()</code>	粘贴文本到文本框

常用监听器方法	说明
<code>void addFocusListener(FocusListener listener)</code>	添加焦点事件监听器
<code>void addKeyListener(KeyListener listener)</code>	添加按键监听器
<code>textField.getDocument().addDocumentListener(DocumentListener listener)</code>	添加文本框内的文本改变监听器

## JTextArea示例

```
JTextArea textArea = new JTextArea("请输入内容", 7, 30); //7行30列
textArea.setLineWrap(true); //设置文本域中的文本为自动换行
textArea.setForeground(Color.BLACK); //设置文字颜色
textArea.setFont(new Font("楷体", Font.BOLD, 16)); //设置字体样式
textArea.setBackground(Color.YELLOW); //设置背景色
```

//添加滚动条

滚动条会自动显示

```
JScrollPane jsp = new JScrollPane(textArea); //将文本域放入滚动窗口
panel.add( jsp ); //将JScrollPane添加到JPanel容器中
JButton btn = new JButton("提交");
```

```
btn.addActionListener(new ActionListener() {
```

@Override

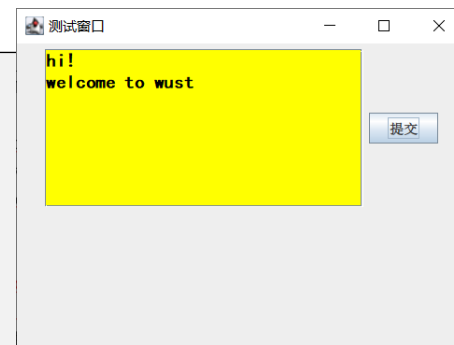
```
public void actionPerformed(ActionEvent e) {
```

```
    System.out.println("提交: " + textArea.getText() ); //控制台输出
```

```
}
```

```
});
```

运行结果



注意：如用 panel.add(textArea); 不会显示滚动条

[【返回】](#)

## 6. JCheckBox

- 复选框，有选中和未选中两种状态，并且可同时选定多个复选框
- 构造函数：
  - JCheckBox(): 创建一个默认的复选框，在默认情况下既未指定文本，也未指定图像，并且未被选择
  - JCheckBox(String text): 创建一个指定文本的复选框，默认未选中
  - JCheckBox(String text,boolean selected): 创建一个指定文本和选择状态的复选框

## JCheckBox常用方法

方法	说明
<code>void setSelected(boolean b)</code>	设置复选框是否选中状态
<code>boolean isSelected()</code>	判断复选框是否选中
<code>void setText(String text)</code>	设置复选框的文本
<code>void setFont(Font font)</code>	设置复选框的字体
<code>void setForeground(Color fg)</code>	设置字体颜色
<code>void setEnabled(boolean enable)</code>	设置复选框是否可用
<code>void setIconTextGap(int iconTextGap)</code>	设置图片和文本的间距
<code>void addItemListener(ItemListener l)</code>	添加Item监听器（选中状态改变时触发）

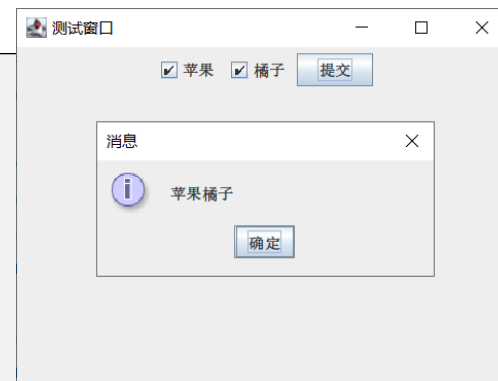


## JCheckBox示例1

```
JCheckBox checkBox01 = new JCheckBox("苹果");  
JCheckBox checkBox02 = new JCheckBox("橘子");  
checkBox01.setSelected(true); // 设置默认第一个复选框选中
```

```
JButton btn = new JButton("提交");  
btn.addActionListener(new ActionListener() {  
    @Override  
    public void actionPerformed(ActionEvent e) {  
        StringBuffer msg = new StringBuffer("");  
        if(checkBox01.isSelected()){  
            msg.append(checkBox01.getText());  
        }  
        if(checkBox02.isSelected()){  
            msg.append(checkBox02.getText());  
        }  
        JOptionPane.showMessageDialog( null, msg );  
    }  
});
```

运行结果

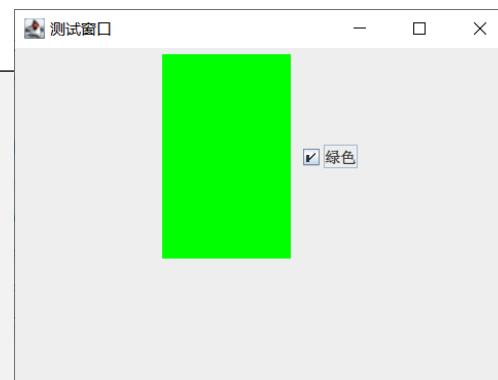


## JCheckBox示例2

```
JTextArea textArea = new JTextArea(10, 10);
JCheckBox checkBox01 = new JCheckBox("绿色");

checkBox01.addItemListener(new ItemListener() {
    @Override
    public void itemStateChanged(ItemEvent e) {
        JCheckBox ck = (JCheckBox) e.getSource(); // 获取事件源（即复选框本身）
        if ( ck.isSelected() )
            textArea.setBackground(Color.GREEN);
        else
            textArea.setBackground(Color.WHITE);
    }
});
```

运行结果



[【返回】](#)

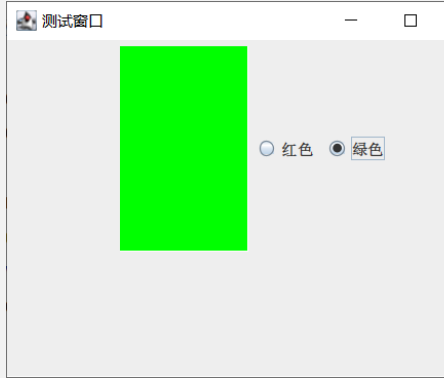
## 7. JRadioButton

- 单选按钮，用法与复选框类似，区别是单选按钮通常位于一个 `ButtonGroup` 按钮组中，同一组的单选按钮中只能有一个被选中
- 构造函数：
  - `JRadioButton()`：创建一个默认的单选按钮，在默认情况下既未指定文本，且未被选择
  - `JRadioButton(String text)`：创建一个指定文本的单选按钮，默认未选中
  - `JRadioButton(String text,boolean selected)`：创建一个指定文本和选择状态的单选按钮

## JRadioButton示例



```
JTextArea textArea = new JTextArea(10, 10);  
// 创建两个单选按钮  
JRadioButton radio01 = new JRadioButton("红色");  
JRadioButton radio02 = new JRadioButton("绿色");  
  
// 创建按钮组，把两个单选按钮添加到该组  
ButtonGroup btnGroup = new ButtonGroup();  
btnGroup.add(radio01);  
btnGroup.add(radio02);  
  
ItemListener itemListener = new ItemListener() {  
    @Override  
    public void itemStateChanged(ItemEvent e) {  
        if( e.getSource()==radio01 ) {    // 对事件源进行判断  
            textArea.setBackground(Color.RED);  
        }  
        else{  
            textArea.setBackground(Color.GREEN);  
        }  
    }  
};  
radio01.addItemListener(itemListener);  
radio02.addItemListener(itemListener);  
  
panel.add(textArea);  
panel.add(radio01);    //注意不能用 panel.add(btnGroup)  
panel.add(radio02);
```

运行结果



【返回】

## 8. JComboBox

- 下拉列表框，以下拉列表的形式展示多个选项，用户可以选择一个值
- 构造函数：
  - JComboBox(): 创建一个空的JComboBox对象
  - JComboBox(ComboBoxModel aModel): 创建一个JComboBox，其选项取自现有的 ComboBoxModel  选项内容  由一个 ComboBoxModel 实例来维护
  - JComboBox(Object[] items): 创建包含指定数组中元素的JComboBox
  - JList(Vector<?> listData): 创建包含指定数组中元素的JComboBox

## JComboBox常用方法

方法	说明
<code>void setSelectedIndex(int index)</code>	设置选中的索引，索引从 0 开始
<code>int getSelectedIndex()</code>	获取当前选中的索引
<code>Object getSelectedItem()</code>	获取当前选中的数据项
<code>void addItem(Object anObject)</code>	将指定的对象作为选项添加到下拉列表框中
<code>void insertItemAt(Object anObject,int index)</code>	在下拉列表框中的指定索引处插入项
<code>void removeItem(Object anObject)</code>	在下拉列表框中删除指定的对象项
<code>void removeItemAt(int anIndex)</code>	在下拉列表框中删除指定位置的对象项
<code>void removeAllItems()</code>	从下拉列表框中删除所有项
<code>int getItemCount()</code>	返回下拉列表框中的项数
<code>Object getItemAt(int index)</code>	获取指定索引的列表项
<code>void setModel(ComboBoxModel&lt;?&gt; model)</code>	设置选项数据模型
<code>ComboBoxModel&lt;?&gt; getModel()</code>	获取选项数据模型（再通过model.getElementAt(int index)和model.getSize()可获取指定索引的选项数据和选项数量）
<code>void setEditable(boolean flag)</code>	设置下拉列表框是否可编辑，默认不可编辑，如果设置为可编辑，则除了选择指定的选项值外，还允许用户自行输入值（自行输入的值索引为-1）
<code>void addItemListener(ItemListener l)</code>	添加Item监听器（选中状态改变时触发）

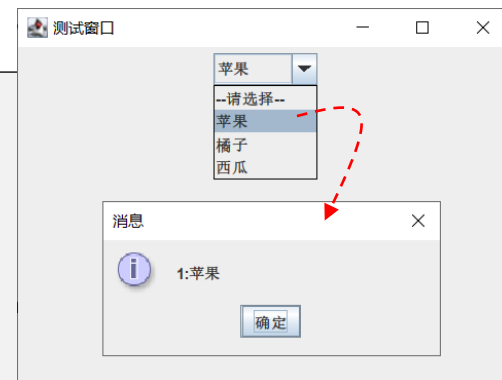
## JComboBox示例1

```
String[] data = new String[]{"苹果", "橘子"};
JComboBox cmb = new JComboBox(data); //创建JComboBox

cmb.insertItemAt("--请选择--", 0);    //插入一项
cmb.addItem("西瓜");                  //添加一项
cmb.setSelectedIndex(0);               //设置选中项

cmb.addItemListener(new ItemListener() {
    @Override
    public void itemStateChanged(ItemEvent e) {
        if ( e.getStateChange() == ItemEvent.SELECTED ) { // 只处理选中的状态
            JOptionPane.showMessageDialog(null, cmb.getSelectedIndex() + ":" + cmb.getSelectedItem());
        }
    }
});
```

运行结果



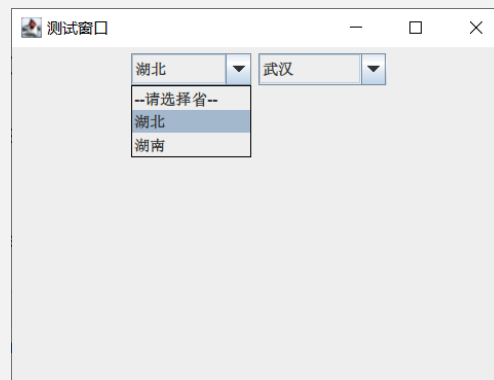
## JComboBox示例2 -- 级联

```
String[] data = new String[]{"-请选择省-", "湖北", "湖南"};
JComboBox province = new JComboBox(data);
JComboBox city = new JComboBox();
city.setPreferredSize(new Dimension(100,25));

province.addItemListener(new ItemListener() {
    @Override
    public void itemStateChanged(ItemEvent e) {
        int index = province.getSelectedIndex();
        switch (index) {
            case 0:
                city.removeAllItems(); //清除所有项
                break;
            case 1:
                city.removeAllItems();
        }
    }
});
```

```
city.addItem("武汉");
city.addItem("黄石");
break;
case 2:
    city.removeAllItems();
    city.addItem("长沙");
    city.addItem("湘潭");
    break;
```


运行结果



[【返回】](#)



## 9. JList

- 列表框，能展示多个选项，允许用户选择一个或多个选项
- JList不具有滚动效果，需要结合JScrollPane实现滚动
- 构造函数：
  - JList(): 构造一个空的只读模型的列表框
  - JList(ListModel dataModel): 根据指定的非 null 模型对象构造一个显示元素的列表框。 选项内容<sub>由一个</sub> ListModel 实例来维护
  - JList(Object[] listData): 使用 listData 指定的元素构造一个列表框。
  - JList(Vector<?> listData): 使用 listData 指定的元素构造一个列表框。

## JList常用方法

方法	说明
<code>void setSelectionMode(int selectionMode)</code>	设置选择模式: 单选SINGLE_SELECTION、多选 MULTIPLE_INTERVAL_SELECTION (配合Shift或Ctrl)
<code>void setSelectedIndex(int index)</code>	设置某个选项选中
<code>void setSelectedIndices(int[] indices)</code>	设置某一些选项选中
<code>int getSelectedIndex()</code>	获取第一个选中的选项索引
<code>int[] getSelectedIndices()</code>	获取所有选中的选项索引
<code>java.util.List list.getSelectedValuesList()</code>	获取所有选中项对象
<code>void setListData(Object[] listData)</code>	以数组形式设置选项数据, 内部将自动封装成 ListModel
<code>void setListData(Vector&lt;?&gt; listData)</code>	以集合形式设置选项数据, 内部将自动封装成 ListModel
<code>void setModel(ListModel&lt;?&gt; model)</code>	直接设置选项数据的 ListModel
<code>ListModel&lt;?&gt; getModel()</code>	获取维护选项数据的 ListModel (再通过 <code>listModel.getElementAt(int index)</code> 和 <code>listModel.getSize()</code> 可获取指定索引的选项数据和选项数量
<code>void setSelectionForeground(Color)</code>	设置选中选项的字体颜色
<code>void setSelectionBackground(Color)</code>	设置选中项背景色
<code>void addListSelectionListener(ListSelectionListene)</code>	添加选项选中状态被改变的监听器

## JList示例

```
String[] data = new String[]{"苹果", "橘子", "西瓜", "葡萄"};
Vector<String> vector = new Vector<String>();
Collections.addAll(vector, data); //数组转存到集合

JList list = new JList(vector); //使用vector创建JList
list.setSelectionMode(ListSelectionModel.MULTIPLE_INTERVAL_SELECTION); //允许多选

JScrollPane scrollPane = new JScrollPane(list); //添加滚动条
scrollPane.setPreferredSize(new Dimension(100, 80)); //设置滚动条大小
panel.add(scrollPane); //不要用panel.add(list); 不然滚动条不会显示

//再添加几项
vector.add("西瓜");
vector.add("荔枝");
list.setListData(vector); //为列表填充数据
```

// 添加选项选中状态被改变的监听器

```
list.addListSelectionListener(new ListSelectionListener() {
```

```
    @Override
```

```
    public void valueChanged(ListSelectionEvent e) {
```

```
        // 测试表明，每当进行选择时，valueChanged方法都会被激活多次
```

```
        // 在响应代码中需要注意的是getValuesAdjusting值的判断。
```

```
        boolean adjusting = e.getValuesAdjusting();    // 点击下去时为true，松开鼠标为false
```

```
        if( !adjusting ) { // 松开鼠标时候
```

```
            int[] indices = list.getSelectedIndices();    // 获取所有被选中的选项索引
```

```
            ListModel<String> listModel = list.getModel(); // 获取选项数据的 ListModel
```

```
            // 输出选中的选项
```

```
            for (int index : indices) {
```

```
                System.out.println("选中: " + index + " = " + listModel.getElementAt(index));
```

```
            }
```

```
            //或者获取所有选中项对象
```

```
            java.util.List values = list.getSelectedValuesList();
```

```
            System.out.println( values.toString() );
```

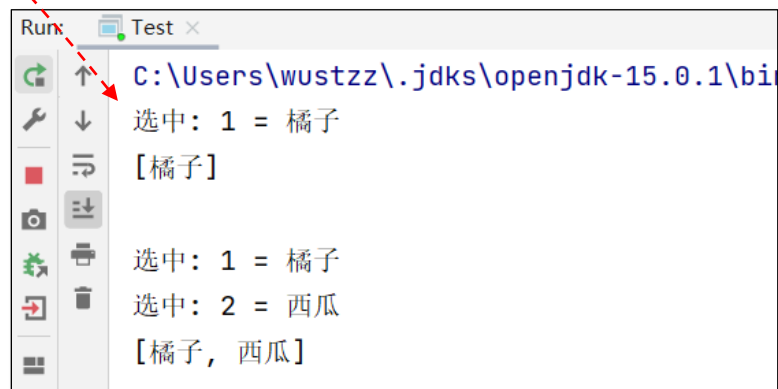
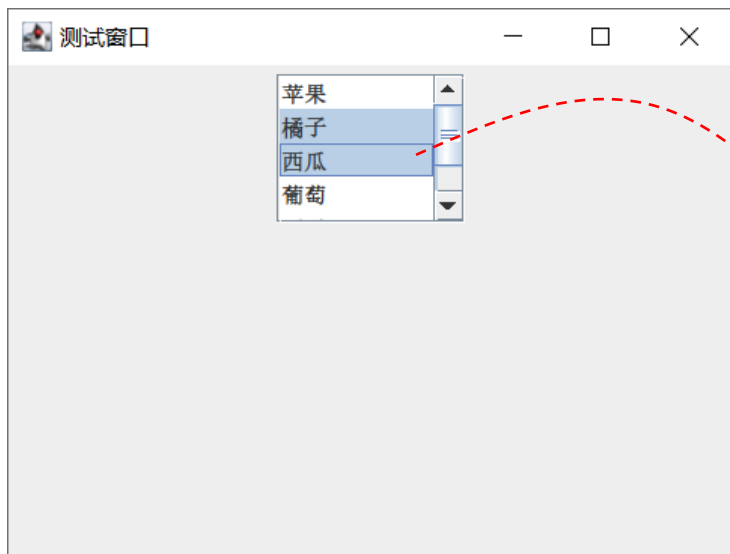
```
            System.out.println();
```

```
        }
```

```
    }
```

```
});
```

## 运行结果

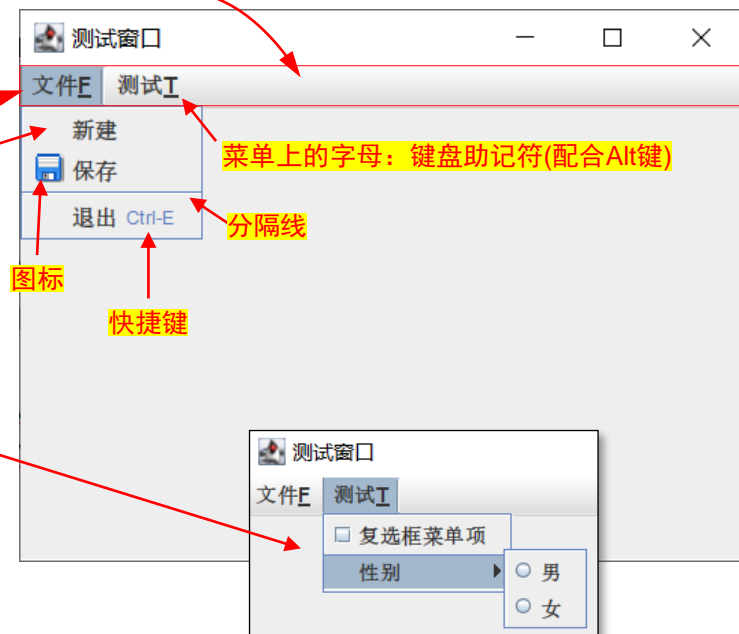


[【返回】](#)

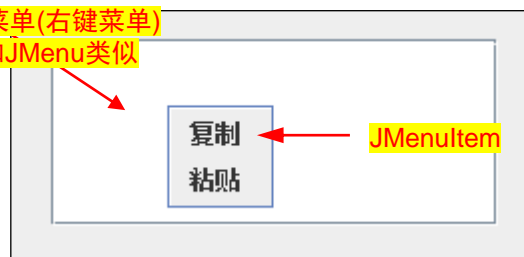
## 10. 菜单组件

- JMenuBar (菜单栏)
- JMenu (菜单)
  - 主菜单 (一级菜单) 或  
有子菜单的菜单 (嵌套关系)
- JMenuItem (普通菜单项)
- JRadioButtonMenuItem (带单选按钮的菜单项)
- JCheckBoxMenuItem (带复选框的菜单项)
- JSeparator (分隔线)
- JPopupMenu (弹出式菜单)

JMenuBar添加到JFrame后, 将在窗口顶部出现



弹出式菜单(右键菜单)  
用法和JMenu类似



## 创建菜单主要代码框架

1. 创建菜单栏: `JMenuBar menuBar = new JMenuBar();`  
将菜单栏添加到窗口: `frame.setJMenuBar(menuBar);`
2. 创建菜单: `JMenu menu = new JMenu("菜单名");`  
将菜单添加到菜单栏: `menuBar.add(menu);`
3. 创建菜单项: 以普通菜单项为例  
`JMenuItem menuItem = new JMenuItem("子菜单名");`  
将菜单项添加到菜单: `menu.add(menuItem);`
4. 为菜单项添加监听事件: 例如 `ActionListener`  
`menuItem.addActionListener(...)`

## JMenu常用方法

方法	说明
JMenu()	创建一个无文本的 JMenu 对象
JMenu(String s)	创建一个带有指定文本的 JMenu 对象
JMenuItem add(JMenuItem menuItem)	添加菜单项到JMenu中
void setMnemonic(int mnemonic)	设置菜单的键盘助记符（通常是 Alt结合）
void addSeparator()	添加一个菜单分割线
void insertSeparator(int index)	在指定的位置插入分隔符
int getItemCount()	返回菜单上的项数，包括分隔符
JMenuItem getItem(int pos)	返回指定位置的菜单项
JMenuItem insert(JMenuItem mi,int pos)	在给定位置插入菜单项
isSelected()	如果菜单是当前选择的（即高亮显示的）菜单，则返回 true
isTopLevelMenu()	如果菜单是“顶层菜单”（即菜单栏的直接子级），则返回 true
setSelected(boolean b)	设置菜单的选择状态



## JMenuItem常用方法

方法	说明
JMenuItem(String text)	创建带有指定文本的 JMenuItem
JMenuItem(String text,Icon icon)	创建带有指定文本和图标的 JMenuItem。
JMenuItem(String text,int mnemonic)	创建带有指定文本和键盘助记符的 JMenuItem
void setText(String text)	设置菜单显示的文本
void setIcon(Icon defaultIcon)	设置菜单显示的图标
void setMnemonic(int mnemonic)	设置菜单的键盘助记符（通常是 Alt结合）
void setAccelerator(KeyStroke keyStroke)	设置快捷键，快捷键能直接触发菜单项的动作
void setActionCommand(String actionCommand)	绑定菜单项的动作命令名称(如果所有菜单项使用同一个监听器，通过命令名称可以区别是哪个菜单项触发的动作)
void addActionListener(ActionListener l)	添加菜单项被点击的监听器

## JCheckBoxMenuItem、JRadioButtonMenuItem 常用方法

方法	说明
void setSelected(boolean b)	设置 复选框/单选按钮 是否选中
boolean isSelected()	判断 复选框/单选按钮 是否选中
void addItemListener(ItemListener)	添加 复选框/单选按钮 状态被改变的监听器

## 菜单示例

//创建菜单栏

```
JMenuBar menuBar = new JMenuBar();
```

//将菜单栏添加到窗口

```
frame.setJMenuBar(menuBar);
```

//创建主菜单

```
JMenu fileMenu = new JMenu("文件F");
```

```
JMenu testMenu = new JMenu("测试T");
```

//设置键盘助记符

```
fileMenu.setMnemonic(KeyEvent.VK_F); //Alt+F
```

```
testMenu.setMnemonic(KeyEvent.VK_T); //Alt+T
```

//将主菜单添加到菜单栏

```
menuBar.add(fileMenu);
```

```
menuBar.add(testMenu);
```

//(1)创建"文件"主菜单下的菜单项

```
JMenuItem newMenuItem = new JMenuItem("新建");
```

```
JMenuItem saveMenuItem = new JMenuItem("保存");
```

```
JMenuItem exitMenuItem = new JMenuItem("退出");
```

//给"保存"菜单项添加图标

```
saveMenuItem.setIcon(new ImageIcon("images/save_small.png"));
```

//给"退出"菜单项添加快捷键，假设是 Ctrl+E

```
exitMenuItem.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_E, ActionEvent.CTRL_MASK));
```

//将菜单项添加到"文件"菜单中

```
fileMenu.add(newMenuItem);
```

```
fileMenu.add(saveMenuItem);
```

```
fileMenu.addSeparator();    // 添加分割线
```

```
fileMenu.add(exitMenuItem);
```

// "新建" 菜单项添加事件

```
newMenuItem.addActionListener(new ActionListener() {  
    @Override  
    public void actionPerformed(ActionEvent e) {  
        JOptionPane.showMessageDialog(null, "新建菜单被点击");  
    }  
});
```

// "退出" 菜单项添加事件

// 使用快捷键也可以激活事件处理代码

```
exitMenuItem.addActionListener(new ActionListener() {  
    @Override  
    public void actionPerformed(ActionEvent e) {  
        int isExit = JOptionPane.showConfirmDialog(null, "确认退出吗?",  
                                                    "标题提示", JOptionPane.YES_NO_OPTION);  
        if (isExit == JOptionPane.YES_OPTION) {  
            frame.dispose(); // 退出前回收frame (可选)  
            System.exit(0);  
        }  
    }  
});
```

//(2)创建"Test"主菜单下的菜单项

```
JCheckBoxMenuItem checkBoxMenuItem = new JCheckBoxMenuItem("复选框菜单项");
```

```
JMenu genderMenu = new JMenu("性别");    //创建带有子菜单的二级菜单，仍然用JMenu
```

```
JRadioButtonMenuItem radioButtonMenuItem01 = new JRadioButtonMenuItem("男");
```

```
JRadioButtonMenuItem radioButtonMenuItem02 = new JRadioButtonMenuItem("女");
```

//把radio菜单项放到ButtonGroup中（实现互斥，可选）

```
ButtonGroup buttonGroup = new ButtonGroup();
```

```
buttonGroup.add(radioButtonMenuItem01);
```

```
buttonGroup.add(radioButtonMenuItem02);
```

//将radio菜单项添加到"性别"菜单中

```
genderMenu.add(radioButtonMenuItem01);
```

```
genderMenu.add(radioButtonMenuItem02);
```

//将菜单项添加到"Test"菜单中

```
testMenu.add(checkBoxMenuItem);
```

```
testMenu.add(genderMenu);
```

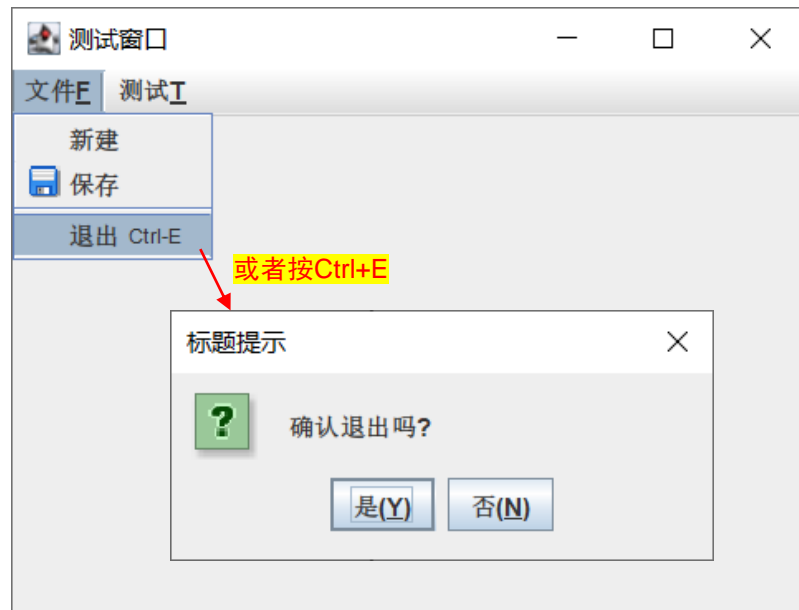
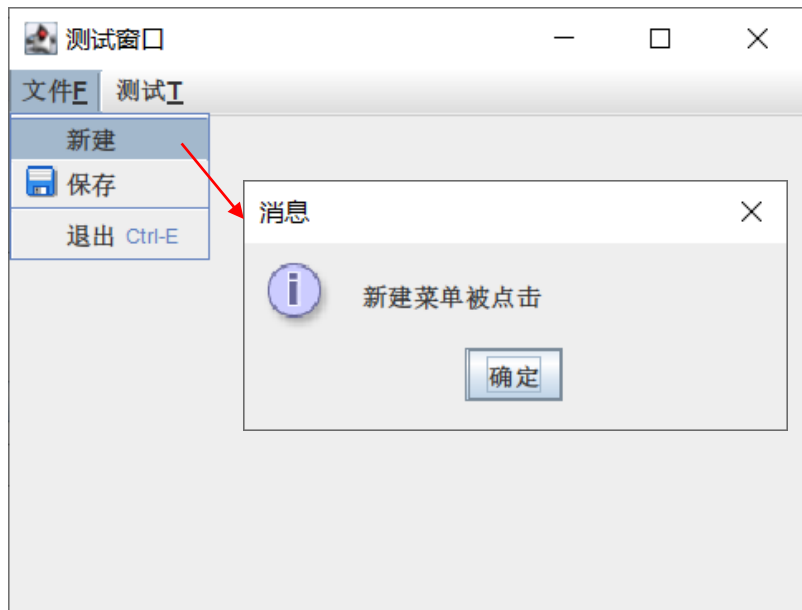
// 设置复选框子菜单状态改变监听

```
checkBoxMenuItem.addItemListener(new ItemListener() {  
    @Override  
    public void itemStateChanged(ItemEvent e) {  
        JOptionPane.showMessageDialog(null, "复选框是否被选中: " + checkBoxMenuItem.isSelected());  
    }  
});
```

// 设置单选按钮子菜单状态改变监听

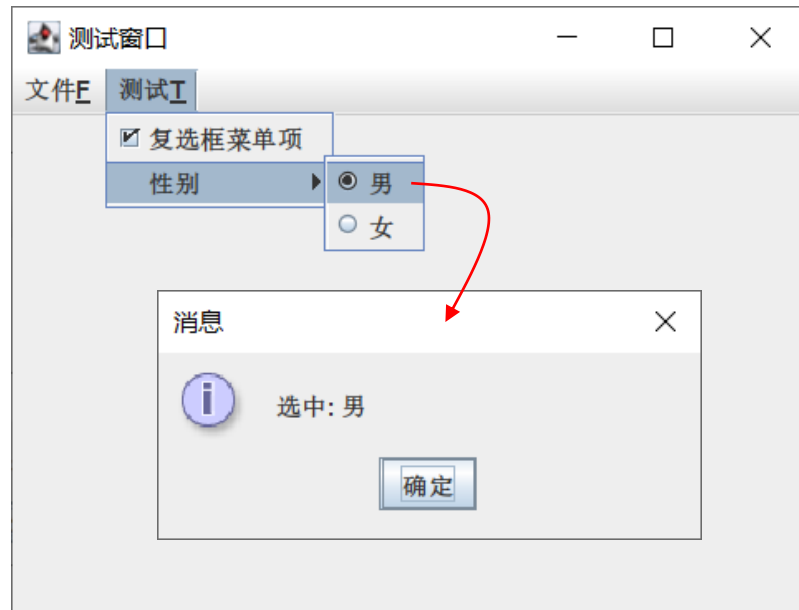
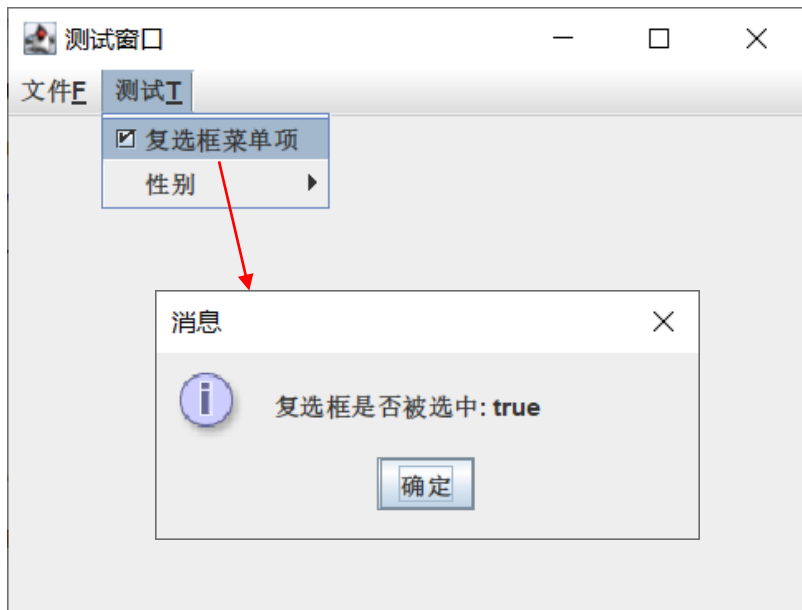
```
ItemListener itemListener=new ItemListener() {  
    @Override  
    public void itemStateChanged(ItemEvent e) {  
        JRadioButtonMenuItem menuItem=(JRadioButtonMenuItem) e.getSource();  
        if( menuItem.isSelected() ){  
            JOptionPane.showMessageDialog(null, "选中: " + menuItem.getText() );  
        }  
    }  
};  
radioButtonItem01.addItemListener(itemListener);  
radioButtonItem02.addItemListener(itemListener);
```

## 运行结果





## 运行结果（续）



## 创建弹出式菜单

### ■ 主要步骤:

1. 新建弹出菜单JPopupMenu, 添加菜单项JMenuItem
2. 为各菜单项创建监听事件
3. 为某个组件设置弹出菜单:

组件.setComponentPopupMenu( JPopupMenu )

## 弹出式菜单示例

```
// 创建弹出式菜单
JPopupMenu popupMenu = new JPopupMenu();

// 创建菜单项
JMenuItem copyMenuItem = new JMenuItem("复制");
JMenuItem pasteMenuItem = new JMenuItem("粘贴");

// 把菜单项添加到弹出菜单
popupMenu.add(copyMenuItem);
popupMenu.add(pasteMenuItem);
```

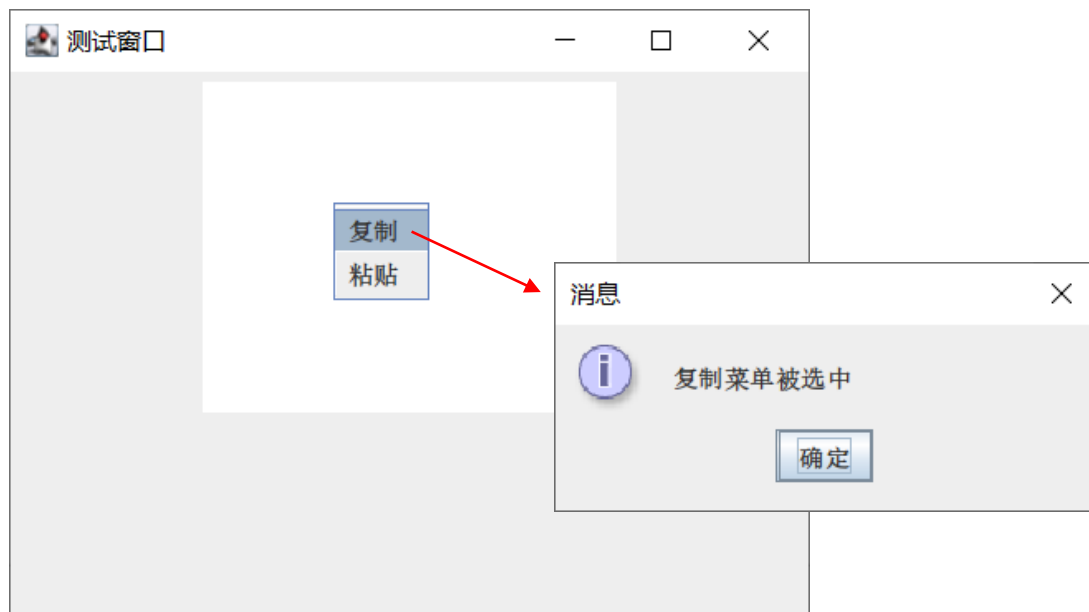
// 添加菜单项的点击监听器

```
copyMenuItem.addActionListener(new ActionListener() {  
    @Override  
    public void actionPerformed(ActionEvent e) {  
        JOptionPane.showMessageDialog(null,"复制菜单被选中");  
    }  
});
```

//将弹出菜单设置为JTextArea组件

```
JTextArea jta=new JTextArea(10,20);  
jta.setComponentPopupMenu(popupMenu);
```

## 运行结果



[【返回】](#)

## 11. JTable和JTree

- JTable: 表格, 是最复杂的组件之一
- JTree: 树, 把数据按照树状进行显示

## JTable示例

```
JPanel panel = new JPanel();
```

```
//表格数据
```

```
Object[][] tableDate = new Object[5][8];
```

```
for (int i = 0; i < 5; i++) {
```

```
    tableDate[i][0] = "1000" + i;
```

```
    for (int j = 1; j < 8; j++) {
```

```
        tableDate[i][j] = 0;
```

```
    }
```

```
}
```

```
//表头
```

```
String[] header = {"学号", "软件工程", "Java", "网络", "数据结构", "数据库", "总成绩", "平均成绩"};
```

```
JTable table = new JTable(tableDate, header);
```

```
JScrollPane scrollPane = new JScrollPane(table); //不添加，则表头不会显示
```

```
panel.add(scrollPane);
```



学号	软件工程	Java	网络	数据结构	数据库	总成绩	平均成绩
10000	0	0	0	0	0	0	0
10001	0	0	0	0	0	0	0
10002	0	0	0	0	0	0	0
10003	0	0	0	0	0	0	0
10004	0	0	0	0	0	0	0

# JTree示例

```
JPanel panel = new JPanel();  
// 创建根节点  
DefaultMutableTreeNode rootNode = new DefaultMutableTreeNode("中国");  
  
// 创建二级节点  
DefaultMutableTreeNode hbNode = new DefaultMutableTreeNode("湖北");  
DefaultMutableTreeNode hnNode = new DefaultMutableTreeNode("湖南");  
// 把二级节点作为子节点添加到根节点  
rootNode.add(hbNode);  
rootNode.add(hnNode);  
  
// 创建三级节点  
DefaultMutableTreeNode whNode = new DefaultMutableTreeNode("武汉");  
DefaultMutableTreeNode hsNode = new DefaultMutableTreeNode("黄石");  
DefaultMutableTreeNode csNode = new DefaultMutableTreeNode("长沙");  
DefaultMutableTreeNode xtNode = new DefaultMutableTreeNode("湘潭");  
// 把三级节点作为子节点添加到相应的二级节点  
hbNode.add(whNode);  
hbNode.add(hsNode);  
hnNode.add(csNode);  
hnNode.add(xtNode);
```



// 使用根节点创建树组件

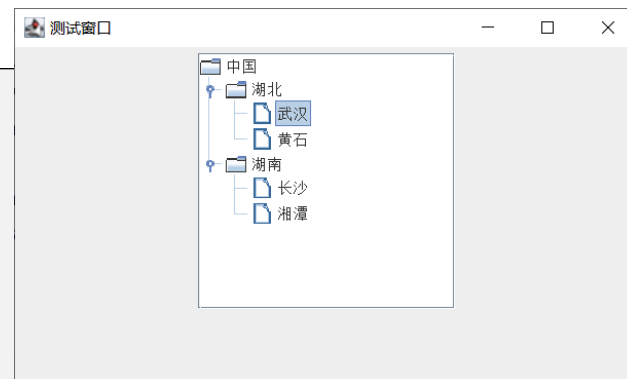
```
JTree tree = new JTree(rootNode);
```

// 设置节点选中监听器

```
tree.addTreeSelectionListener(new TreeSelectionListener() {  
    @Override  
    public void valueChanged(TreeSelectionEvent e) {  
        System.out.println("当前被选中的节点: " + e.getPath());  
    }  
});
```

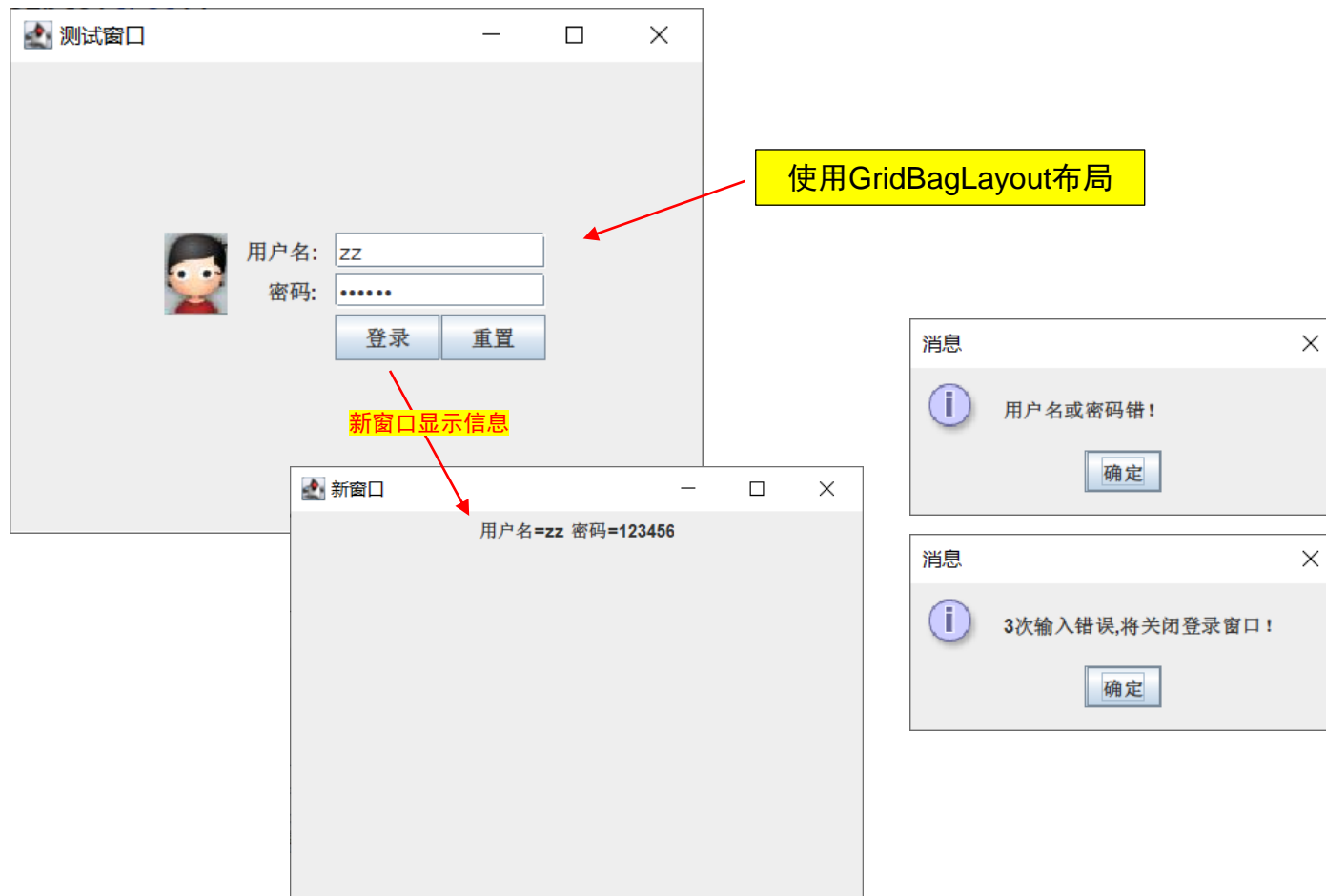
// 滚动条

```
JScrollPane scrollPane = new JScrollPane(tree);  
scrollPane.setPreferredSize(new Dimension(200,200));  
panel.add(scrollPane);
```



[【返回】](#)

## 11.6 Swing示例



## 主要代码

User类

```
class User {  
    String username;  
    String pwd;  
  
    public User() {  
    }  
  
    public User(String username, String pwd) {  
        this.username = username;  
        this.pwd = pwd;  
    }  
  
    public String getUsername() {  
        return username;  
    }  
  
    public void setUsername(String username) {  
        this.username = username;  
    }  
  
    public String getPwd() {  
        return pwd;  
    }  
  
    public void setPwd(String pwd) {  
        this.pwd = pwd;  
    }  
}
```

Login类  
登录主窗口

```
class Login {  
    static int errCount = 0;  
    public static void main(String[] args) {  
        JFrame frame = new JFrame("测试窗口");  
        frame.setSize(400, 300);  
        frame.setLocationRelativeTo(null);  
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
  
        JLabel img = new JLabel(new ImageIcon("images/login.jpg"));  
        JLabel label1 = new JLabel("用户名:");  
        JLabel label2 = new JLabel("密码:");  
        JTextField usernameFiled = new JTextField();  
        JPasswordField passwordField = new JPasswordField();  
        JButton btn01 = new JButton("登录");  
        JButton btn02 = new JButton("重置");  
  
        GridBagLayout layout = new GridBagLayout();  
        JPanel panel = new JPanel(layout);  
        GridBagConstraints c = null; // 约束  
  
        //开始布局
```

//第1行

//图片

```
c = new GridBagConstraints();
c.gridwidth = 1;
c.gridheight = 2;
c.fill = GridBagConstraints.BOTH;
layout.setConstraints(img, c);
panel.add(img);
```

//用户名标签

```
c = new GridBagConstraints();
c.gridwidth = 1;
c.insets = new Insets(0, 10, 0, 10);
layout.setConstraints(label1, c);
panel.add(label1);
```

//用户名文本框

```
c = new GridBagConstraints();
c.gridwidth = 0; //换行
c.fill = GridBagConstraints.HORIZONTAL;
layout.setConstraints(usernameFiled, c);
panel.add(usernameFiled);
```

//第2行

//密码标签

```
c = new GridBagConstraints();
```

```
c.gridwidth = 1;
```

```
c.insets = new Insets(0, 10, 0, 10);
```

```
c.anchor = GridBagConstraints.EAST;
```

```
layout.setConstraints(label2, c);
```

```
panel.add(label2);
```

//密码文本框

```
c = new GridBagConstraints();
```

```
c.gridwidth = 0; //换行
```

```
c.fill = GridBagConstraints.HORIZONTAL;
```

```
layout.setConstraints(passwordField, c);
```

```
panel.add(passwordField);
```

//第3行

//登录按钮

```
c = new GridBagConstraints();
```

```
c.gridwidth = 1;
```

```
c.gridx = 2; //从第3列开始
```

```
layout.setConstraints(btn01, c);
```

```
panel.add(btn01);
```

//重置按钮

```
c = new GridBagConstraints();
```

```
c.gridwidth = 0;
```

```
layout.setConstraints(btn02, c);
```

```
panel.add(btn02);
```

```
btn01.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        String username = usernameFiled.getText();
        String password = new String(passwordField.getPassword());
        if (username.equals("zz") && password.equals("123456")){
            JOptionPane.showMessageDialog(null, "欢迎" + username);
            User user=new User();
            user.setUsername(username);
            user.setPwd(password);

            MyJFrame tmp=new MyJFrame(user);
            tmp.setVisible(true);

            frame.setVisible(false);
            frame.dispose();
        }
        else {
            JOptionPane.showMessageDialog(null, "用户名或密码错！");
            errCount++;
        }
    }
});
```

```
        if (errCount == 3) {
            JOptionPane.showMessageDialog(null,
                "3次输入错误,将关闭登录窗口！");
            frame.dispose();
            System.exit(0);
        }
    }
});

btn02.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        usernameFiled.setText("");
        passwordField.setText("");
    }
});

frame.add(panel);
frame.setVisible(true);
}
}
```

MyJFrame类  
显示信息的新窗口

```
class MyJFrame extends JFrame {  
  
    JLabel label1 = new JLabel("用户名");  
    JLabel label2 = new JLabel("密码");  
    JPanel panel=new JPanel();  
  
    User user;  
  
    public MyJFrame(User user) {  
        this.user=user;  
        init();  
    }  
  
    public void init(){  
        this.setTitle("新窗口");  
        this.setSize(400,300);  
  
        this.setLocationRelativeTo(null);  
  
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
        panel.add(label1);  
        panel.add(label2);  
        this.setContentPane(panel);  
        showInfo();  
    }  
  
    public void showInfo(){  
        label1.setText("用户名="+user.getUsername());  
        label2.setText("密码="+user.getPwd());  
    }  
}
```

【完】