

# Java语言程序设计

配套教材由清华大学出版社出版发行

## 第6章 图形用户界面程序



中國農業大學

阚道宏

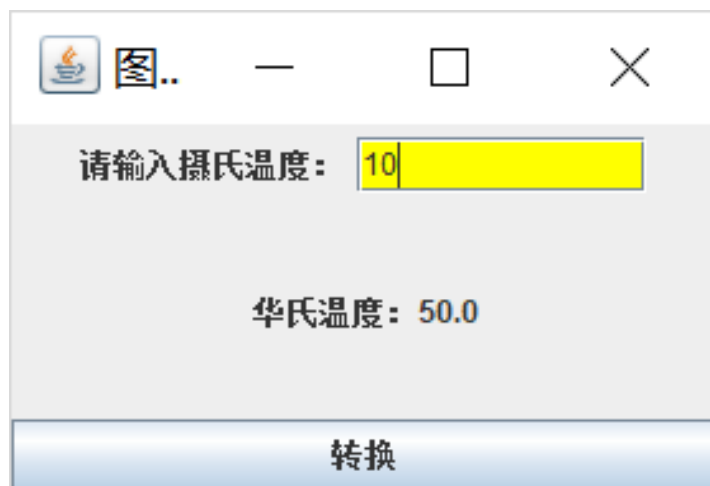
# 第6章 图形用户界面程序

- 程序执行过程中，通常需要用户输入原始数据或选择功能（称为**输入**），程序将计算得到的中间结果和最终结果反馈给用户（称为**输出**）
- 用户与程序之间的输入和输出操作统称为**人机交互**
- 人机交互的形式主要有两种
  - **命令行界面**（Command Line Interface，简称**CLI**）
  - **图形用户界面**（Graphical User Interface，简称**GUI**）



# 第6章 图形用户界面程序

- 程序的用户界面
  - 命令行界面CLI
  - 图形用户界面GUI



# 第6章 图形用户界面程序

- 本章内容
  - [6.1 图形用户界面](#)
  - [6.2 编写图形用户界面程序](#)
  - [6.3 响应用户操作](#)
  - [6.4 常用图形组件](#)
  - [6.5 对话框](#)
  - [6.6 鼠标事件和键盘事件](#)
  - [6.7 Java小应用程序Applet](#)



# 6.1 图形用户界面

- 基本概念和术语

- 屏幕坐标系

- 窗口

- 窗口位置 (location)
    - 窗口尺寸 (size)
    - 窗口标题 (title)
    - 菜单栏 (menu bar)
    - 内容面板 (content panel)
    - 组件 (component)
      - 按钮 (button)
      - 标签 (label)
      - 文本框 (text field)

- 容器 (container)

- 面板 (panel)
    - 顶层容器 (top-level container)



# 6.1 图形用户界面

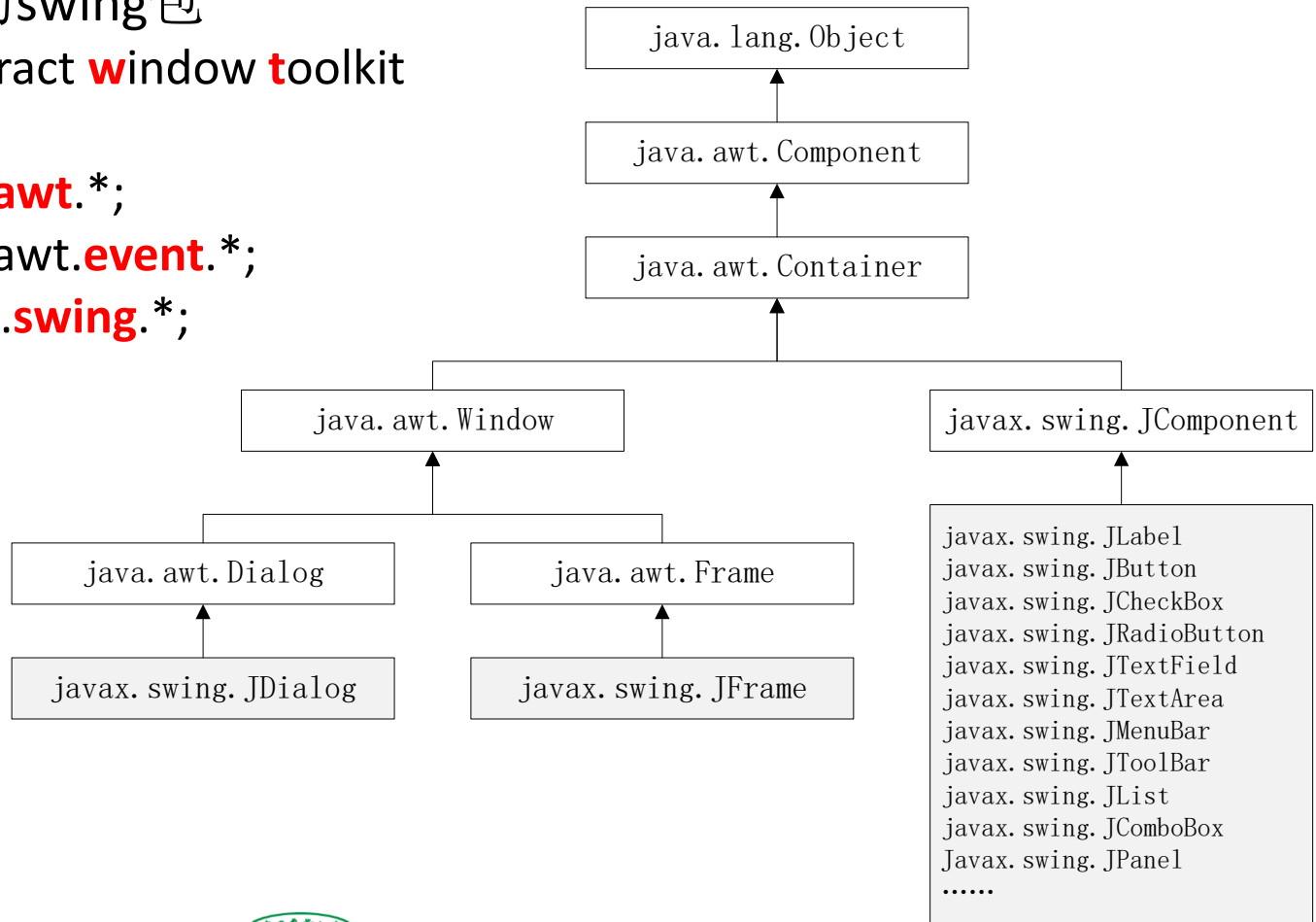
- Java API中的swing包
  - Java API将窗口、菜单栏、图形组件和容器等界面元素抽象成数据模型，并定义成一组Java类和接口
  - 这些Java类和接口被集中放在图形工具包 **javax.swing** 及其下面的子包中。它们共同组成了一个开发图形用户界面程序的框架，被称为 **swing框架**



# 6.1 图形用户界面

- Java API中的swing包
  - **awt**: **a**bstract **w**indow **t**oolkit

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



# java.awt.Component类说明文档

public abstract class **Component**

extends **Object**

implements **ImageObserver, MenuContainer, Serializable**

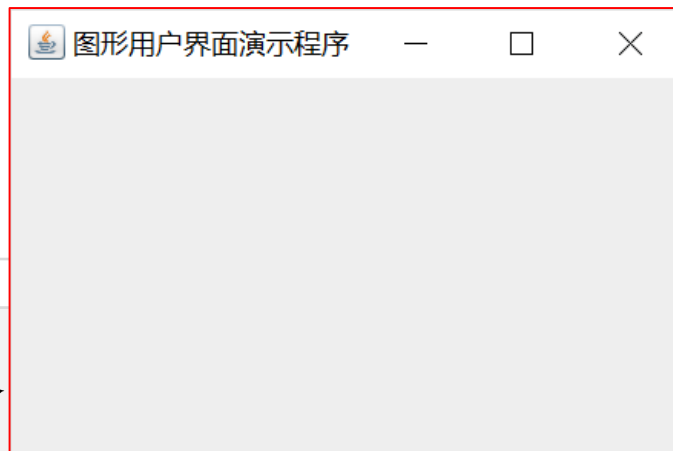
	修饰符	类成员（节选）	功能说明
1	protected	<b>Component()</b>	构造方法
2		void <b>setSize</b> (int width, int height)	设置组件尺寸
3		void <b>setLocation</b> (int x, int y)	设置组件位置
4		void <b>setBounds</b> (int x, int y, int width, int height)	移动或修改组件位置、尺寸
5		void <b>setVisible</b> (boolean b)	显示或隐藏组件
6		void <b>setEnabled</b> (boolean b)	设置组件是否可用
7		void <b>validate</b> ()	验证并重新布局组件
8		void <b>paint</b> (Graphics g)	绘制组件
9		void <b>repaint</b> ()	申请重绘组件
10		void <b>update</b> (Graphics g)	刷新并申请重绘组件
11		void <b>setFont</b> (Font f)	设置字体
12		void <b>setCursor</b> (Cursor cursor)	设置鼠标光标
13		void <b>setBackground</b> (Color c)	设置背景颜色
14		void <b>setForeground</b> (Color c)	设置前景颜色
15		void <b>requestFocus</b> ()	申请获得键盘输入焦点
16		boolean <b>hasFocus</b> ()	检查是否具有键盘输入焦点
17		int <b>getX</b> ()	获取组件左上角的x坐标
18		int <b>getY</b> ()	获取组件左上角的y坐标
19		int <b>getWidth</b> ()	获取组件的宽度
20		int <b>getHeight</b> ()	获取组件的高度
21		Container <b>getParent</b> ()	获取所在的容器
22		Graphics <b>getGraphics</b> ()	获取组件的绘图对象
23		boolean <b>contains</b> (int x, int y)	检查某个坐标点是否在本组件的显示区域内
24		void <b>addKeyListener</b> (KeyListener l)	添加键盘事件监听器
25		void <b>addMouseListener</b> (MouseListener l)	添加鼠标事件监听器
26		void <b>addMouseMotionListener</b> (MouseMotionListener l)	添加鼠标移动事件监听器
27		void <b>addMouseWheelListener</b> (MouseWheelListener l)	添加鼠标滚轮事件监听器





## 6.2 编写图形用户界面程序

- 框架窗口类JFrame
  - 创建程序窗口



例6-1 一个使用框架窗口类JFrame编写的图形用户界面演示程序

```
1 import java.awt.*;    // 导入java.awt包中的类
2 import java.awt.event.*; // 导入java.awt.event包中定义
3 import javax.swing.*;  // 导入javax.swing包中的类
4
5 public class GUITest {           // 主类
6     public static void main(String[] args) {    // 主方法
7         JFrame w = new JFrame();    // 创建框架窗口类JFrame的对象
8         w.setTitle("图形用户界面演示程序"); // 设置窗口标题
9         w.setLocation(100, 100);    // 设置窗口位置
10        w.setSize(460, 300);        // 设置窗口尺寸
11        w.setVisible(true);         // 设置窗口为可见状态
12        w.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // 关闭窗口时退出程序
13    } }
```



# 6.2 编写图形用户界面程序

- 框架窗口类**JFrame**

javax.swing.JFrame类说明文档			
public class <b>JFrame</b>			
extends <b>Frame</b>			
implements <b>WindowConstants, Accessible, RootPaneContainer</b>			
	修饰符	类成员（节选）	功能说明
1	static	int <b>EXIT_ON_CLOSE</b>	常量，关闭窗口时退出程序
2		<b>JFrame()</b>	构造方法
3		<b>JFrame</b> (String title)	构造方法
4		void <b>setSize</b> (int width, int height)	设置窗口尺寸
5		void <b>setLocation</b> (int x, int y)	设置窗口位置（左上角）
6		void <b>setTitle</b> (String title)	设置窗口标题
7		void <b>setDefaultCloseOperation</b> (int operation)	设置关闭窗口时的默认操作
8		void <b>setLayout</b> (LayoutManager manager)	设置窗口的布局管理器
9		Container <b>getContentPane</b> ()	取得窗口的内容面板
10		void <b>setContentPane</b> (Container contentPane)	设置窗口的内容面板
11		JLayeredPane <b>getLayeredPane</b> ()	取得窗口的分层面板
12		Component <b>getGlassPane</b> ()	取得窗口的透明面板
13		JRootPane <b>getRootPane</b> ()	取得窗口的根面板
14		void <b>setJMenuBar</b> (JMenuBar menubar)	设置窗口的菜单栏
15		JMenuBar <b>getJMenuBar</b> ()	取得窗口的菜单栏
16		void <b>paint</b> (Graphics g)	在窗口中绘图
17		void <b>repaint</b> ()	重绘窗口中的内容
18		void <b>setVisible</b> (boolean b)	设置窗口是否为可见状态
.....			



## 6.2 编写图形用户界面程序

例6-2 一个在窗口中绘图的Java演示程序（HelloWorld.java）

```
1 import java.awt.*;    // 导入java.awt包中的类
2 import java.awt.event.*; // 导入java.awt.event包中定义的事件类
3 import javax.swing.*;  // 导入javax.swing包中的类
4
5 public class HelloWorld {           // 主类
6     public static void main(String[] args) { // 主方法
7         JFrame w = new JFrame();        // 创建框架窗口类JFrame的对象
8         w.setTitle("图形用户界面演示程序"); // 设置窗口标题
9         w.setSize(460, 300);            // 设置窗口尺寸
10        w.setLocation(100, 100);         // 设置窗口位置
11        w.setVisible(true);             // 设置窗口为可见状态
12        w.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // 关闭窗口时退出程序
13        // 以下为在窗口中绘图的Java代码
14        Graphics g = w.getGraphics();    // 获取窗口的绘图对象
15        Font ef = new Font("TimesRoman", Font.PLAIN, 16); // 选择字体
16        g.setFont(ef);                   // 设置字体
17        g.drawString("Hello, World!", 20, 80); // 显示文字信息
18        Font cf = new Font("楷体", Font.PLAIN, 24);    // 选择字体
19        g.setFont(cf);                   // 设置字体
20        g.drawString("你好，世界！", 20, 120); // 显示文字信息
21        g.setColor( Color.BLACK );      // 设置填充颜色
22        g.fillRect(20, 150, 100, 100); // 画一个实心矩形
23        g.setColor( Color.RED );        // 设置绘图颜色
24        g.drawRect(20, 150, 100, 100); // 画一个矩形框，此处是为上面的实心矩形加框
25    } }
```



# 6.2 编写图形用户界面程序

- 框架窗口类JFrame
  - 图形类**Graphics**

java.awt. <b>Graphics</b> 类说明文档			
public abstract class <b>Graphics</b> extends <b>Object</b>			
	修饰符	类成员（节选）	功能说明
1	protected	<b>Graphics()</b>	构造方法
2	abstract	void <b>setColor</b> (Color c)	设置颜色
3	abstract	void <b>setFont</b> (Font font)	设置字体
4	abstract	void <b>drawString</b> (String str, int x, int y)	显示文本信息（字符串）
5	abstract	void <b>drawLine</b> (int x1, int y1, int x2, int y2)	画直线
6		void <b>drawRect</b> (int x, int y, int width, int height)	画矩形
7	abstract	void <b>fillRect</b> (int x, int y, int width, int height)	填充矩形
8	abstract	void <b>drawOval</b> (int x, int y, int width, int height)	画椭圆或圆形
9	abstract	void <b>fillOval</b> (int x, int y, int width, int height)	填充椭圆或圆形
10	abstract	boolean <b>drawImage</b> (Image img, int x, int y, ImageObserver observer)	显示图像
.....			



# 6.2 编写图形用户界面程序

- 框架窗口类JFrame
  - 颜色类Color

java.awt.Color类说明文档			
public class Color extends Object implements Paint, Serializable			
	修饰符	类成员（节选）	功能说明
1	static	Color <b>BLACK</b>	颜色常量，黑色
2	static	Color <b>WHITE</b>	颜色常量，白色
3	static	Color <b>RED</b>	颜色常量，红色
4	static	Color <b>GREEN</b>	颜色常量，绿色
5	static	Color <b>BLUE</b>	颜色常量，蓝色
6		<b>Color</b> (int r, int g, int b)	构造方法，各颜色分量的数值必须在0~255之间
7		<b>Color</b> (int r, int g, int b, int a)	构造方法，各颜色分量和Alpha分量（透明度）的数值必须在0~255之间
8		Color(int rgb)	构造方法，rgb的3个低字节为红绿蓝分量
9		int <b>getRed</b> ()	返回颜色的红色分量值
10		int <b>getGreen</b> ()	返回颜色的绿色分量值
11		int <b>getBlue</b> ()	返回颜色的蓝色分量值
12		int <b>getRGB</b> ()	将红绿蓝分量合并成一个int型整数
.....			



# 6.2 编写图形用户界面程序

- 框架窗口类JFrame
  - 字体类**Font**

java.awt. <b>Font</b> 类说明文档			
public class <b>Font</b> extends <b>Object</b> implements <b>Serializable</b>			
	修饰符	类成员（节选）	功能说明
1	static	int <b>BOLD</b>	字体常量，粗体
2	static	int <b>ITALIC</b>	字体常量，斜体
3	static	int <b>PLAIN</b>	字体常量，正常
4		<b>Font</b> (String name, int style, int size)	构造方法
5	static	Font <b>getFont</b> (String nm)	创建指定字体名的对象
6		Font <b>deriveFont</b> (int style, float size)	调整字体的风格和大小
.....			



## 6.2 编写图形用户界面程序

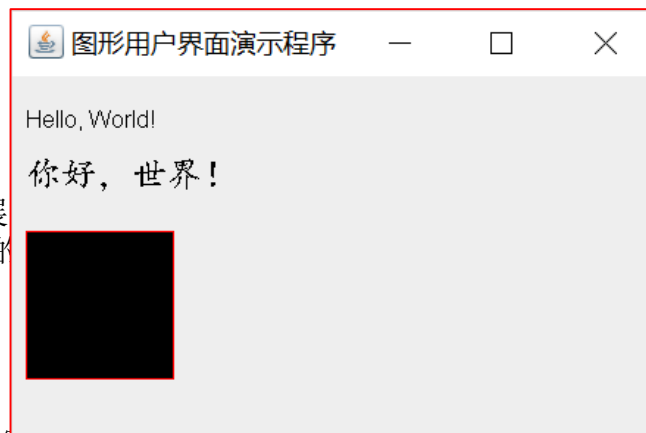
例6-2 一个在窗口中绘图的Java演示程序（HelloWorld.java）

```
1 import java.awt.*;    // 导入java.awt包中的类
2 import java.awt.event.*; // 导入java.awt.event包中定义的事件类
3 import javax.swing.*; // 导入javax.swing包中的类
4
5 public class HelloWorld {           // 主类
6     public static void main(String[] args) { // 主方法
7         JFrame w = new JFrame();        // 创建框架窗口类JFrame的对象
8         w.setTitle("图形用户界面演示程序"); // 设置窗口标题
9         w.setSize(460, 300);            // 设置窗口尺寸
10        w.setLocation(100, 100);         // 设置窗口位置
11        w.setVisible(true);              // 设置窗口为可见状态
12        w.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // 关闭窗口时退出程序
13        // 以下为在窗口中绘图的Java代码
14        Graphics g = w.getGraphics();    // 获取窗口的绘图对象
15        Font ef = new Font("TimesRoman", Font.PLAIN, 16); // 选择字体
16        g.setFont(ef);                   // 设置字体
17        g.drawString("Hello, World!", 20, 80); // 显示文字信息
18        Font cf = new Font("楷体", Font.PLAIN, 24);    // 选择字体
19        g.setFont(cf);                   // 设置字体
20        g.drawString("你好，世界！", 20, 120); // 显示文字信息
21        g.setColor( Color.BLACK );      // 设置填充颜色
22        g.fillRect(20, 150, 100, 100); // 画一个实心矩形
23        g.setColor( Color.RED );        // 设置绘图颜色
24        g.drawRect(20, 150, 100, 100); // 画一个矩形框，此处是为上面的实心矩形加框
25    } }
```



例6-3 一个继承框架窗口类JFrame并重写paint()方法的Java示例程序（HelloWorld1.java）

```
1 import java.awt.*;    // 导入java.awt包中的类
2 import java.awt.event.*; // 导入java.awt.event包中定义的事件类
3 import javax.swing.*;  // 导入javax.swing包中的类
4
5 public class HelloWorld1 {          // 主类
6     public static void main(String[] args) { // 主方法
7         MainWnd w = new MainWnd(); // 创建并显示主窗口对象
8         w.repaint();               // 调用窗口的重绘方法
9     } }
10
11 class MainWnd extends JFrame { // 定义主窗口类：继承并扩展
12     public MainWnd() {          // 构造方法：完成初始化窗口的
13         setTitle("图形用户界面演示程序"); // 设置窗口标题
14         setSize(460, 300);      // 设置窗口尺寸
15         setLocation(100, 100);  // 设置窗口位置
16         setVisible(true);       // 设置窗口时显示或英寸
17         setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE ); // 关闭窗口时退出程序
18     }
19
20     public void paint(Graphics g) { // 重写绘图方法paint()
21         super.paint( g );          // 调用超类的paint()方法
22         Font ef = new Font("TimesRoman", Font.PLAIN, 16); // 创建字体对象
23         g.setFont( ef );           // 设置字体
24         g.drawString("Hello, World!", 20, 80);           // 显示英文信息
25         Font cf = new Font("楷体", Font.PLAIN, 24);     // 选择字体
26         g.setFont( cf );           // 设置字体
27         g.drawString("你好，世界！ ", 20, 120);         // 显示中文信息
28         g.setColor( Color.BLACK ); // 设置填充颜色
29         g.fillRect(20, 150, 100, 100); // 画一个实心矩形
30         g.setColor( Color.RED );    // 设置绘图颜色
31         g.drawRect(20, 150, 100, 100); // 画一个矩形框，此处是为上面的实心矩形加框
32     } }
```





## 6.2 编写图形用户界面程序

- 在组件类**Component**上**绘图**的基本原理和编程方法
  - **绘图目的**。一是程序需要在组件中向用户显示信息，二是组件因尺寸改变等原因需重绘内容
  - **绘图过程**
    - **Java虚拟机**统一负责绘图操作的调度
    - 当需要在组件中显示信息时，程序员应当调用组件的重绘方法**repaint()**
    - Java虚拟机在接收到绘图请求后会调用组件的绘图方法**paint()**
    - 继承并扩展组件类，**重写**组件的绘图方法**paint()**
    - **请注意**：程序员不要直接调用组件的**paint()**方法，而应通过**repaint()**方法进行间接调用
  - **组件的绘图对象**。Java虚拟机在调用组件的绘图方法**paint()**时，会传递一个图形类**Graphics**的绘图对象



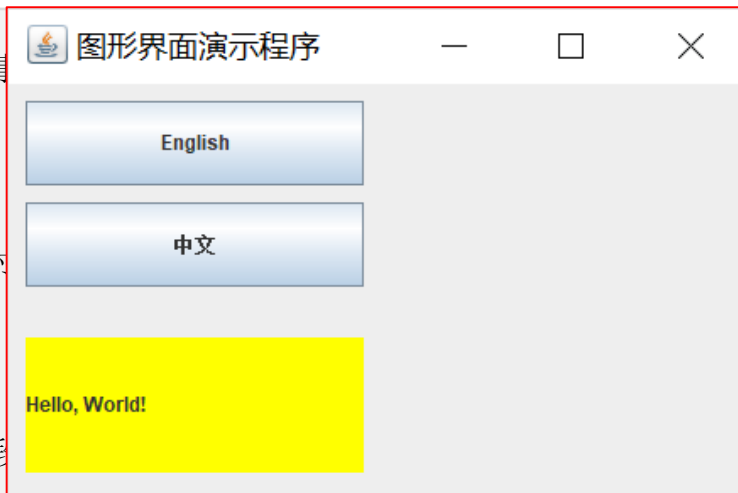
## 6.2 编写图形用户界面程序

- 在窗口中添加图形组件
  - javax.swing.**JLabel**, 标签
  - javax.swing. **JButton**, 按钮
  - javax.swing.**JCheckBox**, 复选框
  - javax.swing.**JRadioButton**, 单选按钮
  - javax.swing.**JTextField**, 单行文本框
  - javax.swing.**JTextArea**, 多行文本框
  - javax.swing.**JMenuBar**, 菜单栏
  - javax.swing.**JToolBar**, 工具栏
  - javax.swing.**JComboBox**, 下拉列表框
  - javax.swing.**JTable**, 二维表格
  - javax.swing.**JList**, 列表框
  - javax.swing.**JPanel**, 子面板（容器）
  - .....



例6-4 一个在窗口中添加图形组件的Java示例程序（JComponentTest.java）

```
1 import java.awt.*;    // 导入java.awt包中的类
2 import java.awt.event.*; // 导入java.awt.event包中定义的事件
3 import javax.swing.*; // 导入javax.swing包中的类
4
5 public class JComponentTest {    // 主类
6     public static void main(String[] args) { // 主方法
7         MainWnd w = new MainWnd(); // 创建并显示主窗口对象
8     } }
9
10 class MainWnd extends JFrame {    // 扩展JFrame
11     private JButton bEN, bCN;    // 添加两个按钮字段
12     private JLabel msg = new JLabel(); // 再添加一个标签字段
13     public MainWnd() { // 构造方法
14         // 初始化窗口
15         setTitle( "图形界面演示程序" );
16         setSize(460, 300); setLocation(100, 100);
17         setVisible(true);
18         setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
19         // 初始化图形组件
20         bEN = new JButton( "English" );    // 创建按钮对象
21         bCN = new JButton( "中文" );
22         msg.setOpaque(true);    // 设置标签背景是否透明：不透明
23         msg.setBackground(Color.YELLOW); // 设置标签背景颜色：黄色
24         msg.setText( "Hello, World!" );    // 在标签上显示文本信息
25         // 将组件添加到窗口的内容面板上
26         Container cp = getContentPane(); // 获得窗口的内容面板（容器）
27         cp.setLayout( null );    // 设置容器的布局形式：null-手工布局
28         cp.add( bEN ); cp.add( bCN ); cp.add( msg ); // 在容器中添加组件
29         bEN.setBounds(10, 10, 200, 50); // 手工设置各组件的位置和尺寸
30         bCN.setBounds(10, 70, 200, 50);
31         msg.setBounds(10, 150, 200, 80);
32         cp.validate();    // 检查并布局容器里的组件
33     } }
```



## 6.2 编写图形用户界面程序

- 在窗口中添加图形组件
  - 容器类**Container**

java.awt.Container类说明文档			
public class <b>Container</b> extends <b>Component</b>			
	修饰符	类成员（节选）	功能说明
1		<b>Container()</b>	构造方法
2		Component <b>add</b> (Component comp)	添加组件
3		Component <b>add</b> (Component comp, int index)	添加组件并指定其序号
4		void <b>doLayout</b> ()	对调整后组件重新布局
5		Component[] <b>getComponents</b> ()	获取所包含的组件
6		int <b>getComponentCount</b> ()	获取所包含的组件个数
7		void <b>remove</b> (Component comp)	删除组件
8		void <b>remove</b> (int index)	删除指定序号的组件
9		void <b>validate</b> ()	检查并重新布局容器中的组件
.....			



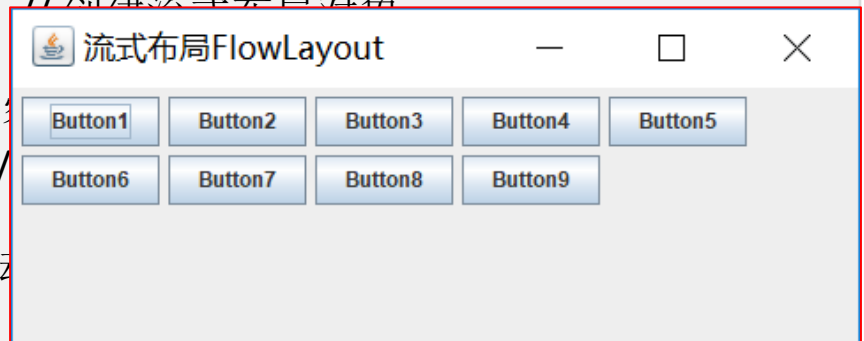
## 6.2 编写图形用户界面程序

- 容器中组件的布局管理
  - 设置容器中图形组件的位置和大小，这被称为容器的**布局管理**（**layout** management）
  - 手工布局
  - 设置布局管理策略：自动布局
    - 流式布局（**FlowLayout**）
    - 边框布局（**BorderLayout**）
    - 网格布局（**GridLayout**）
    - 卡片式布局（**CardLayout**）



### 例6-5 一个使用流式布局FlowLayout的Java示例程序（LayoutTest.java）

```
1 import java.awt.*;    // 导入java.awt包中的类
2 import java.awt.event.*; // 导入java.awt.event包中定义的事件类
3 import javax.swing.*;  // 导入javax.swing包中的类
4
5 public class LayoutTest {           // 主类
6     public static void main(String[] args) { // 主方法
7         JButton btn[] = { // 创建一个按钮对象数组，包含9个按钮
8             new JButton("Button1"), new JButton("Button2"), new JButton("Button3"),
9             new JButton("Button4"), new JButton("Button5"), new JButton("Button6"),
10            new JButton("Button7"), new JButton("Button8"), new JButton("Button9")
11        };
12        JFrame w = new JFrame();      // 创建程序窗口
13        w.setSize(500, 200); w.setLocation(100, 100); // 初始化窗口
14        w.setVisible(true);
15        w.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
16        // 下面演示：设置内容面板的布局策略，然后添加按钮并自动布局
17        w.setTitle( "流式布局FlowLayout" ); // 设置窗口标题
18        Container cp = w.getContentPane(); // 获得窗口w的内容面板
19        FlowLayout fl = new FlowLayout(); // 创建流式布局对象
20        fl.setAlignment(FlowLayout.LEFT);
21        cp.setLayout( fl ); // 将内容面板（容器）的布局策略设置为流式布局
22        for (int n = 0; n < btn.length; n++) // 将按钮对象添加到内容面板
23            cp.add( btn[n] );
24        cp.validate();           // 检查并自动布局
25    } }
```



## 6.2 编写图形用户界面程序

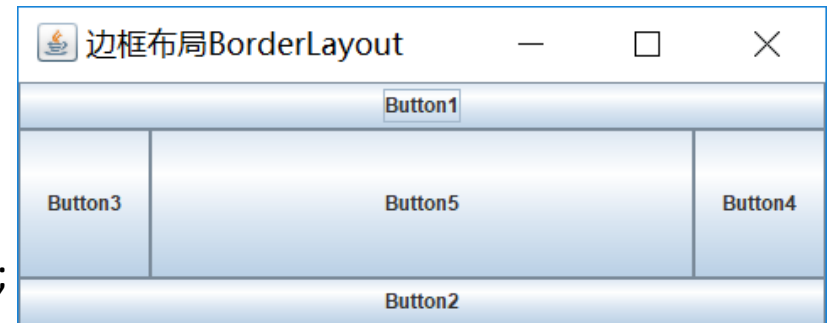
- 容器中组件的布局管理

- 边框布局BorderLayout

```
w.setTitle( "边框布局BorderLayout" ); // 设置窗口标题  
Container cp = w.getContentPane(); // 获得窗口w的内容面板
```

```
cp.setLayout( new BorderLayout() );
```

```
cp.add( btn[0], BorderLayout.NORTH );  
cp.add( btn[1], BorderLayout.SOUTH );  
cp.add( btn[2], BorderLayout.WEST );  
cp.add( btn[3], BorderLayout.EAST );  
cp.add( btn[4], BorderLayout.CENTER );
```



```
cp.validate(); // 检查并自动布局容器里的组件
```

- **注：** 边框布局最多只能添加5个组件



## 6.2 编写图形用户界面程序

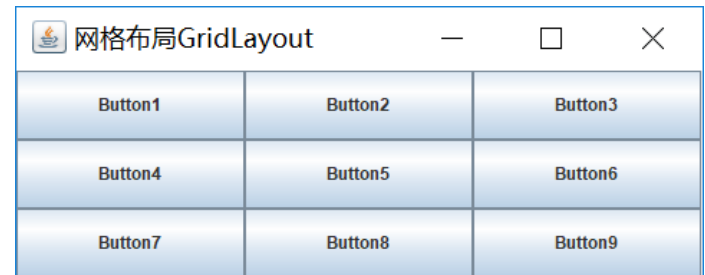
- 容器中组件的布局管理

- 网格布局GridLayout

```
w.setTitle( "网格布局GridLayout" ); // 设置窗口标题  
Container cp = w.getContentPane();
```

```
cp.setLayout( new GridLayout(3, 3) ); // 3行3列的网格
```

```
for (int n = 0; n < btn.length; n++)  
    cp.add( btn[n] );
```



```
cp.validate(); // 检查并自动布局容器里的组件
```





## 6.2 编写图形用户界面程序

- 容器中组件的布局管理

- 卡片式布局CardLayout

```
w.setTitle( "卡片式布局CardLayout" );
```

```
Container cp = w.getContentPane();
```

```
CardLayout cl = new CardLayout(); // 创建卡片式布局对象
```

```
cp.setLayout( cl ); // 将内容面板设为卡片式布局
```

```
for (int n = 0; n < btn.length; n++)
```

```
    cp.add( btn[n] );
```

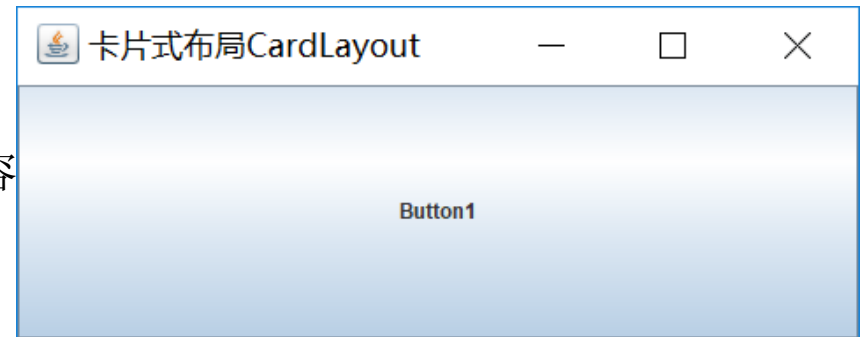
```
cp.validate(); // 检查并自动布局容
```

- 翻阅层叠在一起的组件

```
cl.first( cp ); // 显示第一张卡片
```

```
cl.next( cp ); // 显示下一张卡片
```

```
cl.previous( cp ); // 显示上一张卡片
```



## 6.3 响应用户操作

- 图形用户界面程序
  - 创建程序窗口
  - 添加组件并进行布局
  - 窗口中的每个组件都代表了某种程序功能
  - 用户操作某个组件，例如点击了某个按钮
  - 程序应当响应用户操作，完成组件的程序功能

一个具有完整交互功能的图形用户界面程序



中國農業大學

閻道宏

例6-6 一个HelloWorld程序例子 (JButtonTest.java)

```
1 import java.awt.*; // 导入java.awt包中的类
2 import java.awt.event.*; // 导入java.awt.event包中定义的事
3 import javax.swing.*; // 导入javax.swing包中的类
4
5 public class JButtonTest { // 主类
6     public static void main(String[] args) { // 主方法
7         MainWnd w = new MainWnd(); // 创建并显示程序主窗
8     } }
9
10 class MainWnd extends JFrame { // 扩展JFrame
11     private JButton bEN, bCN; // 添加两个功能按钮
12     private JLabel msg = new JLabel(); // 添加一个信息显示区
13     public MainWnd() { // 构造方法
14         // 初始化窗口
15         setTitle("图形用户界面演示程序");
16         setSize(460, 300); setLocation(100, 100);
17         setVisible(true);
18         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
19         // 初始化功能按钮，并将功能按钮放入一个子面板 (JPanel)
20         bEN = new JButton("English Button"); // 创建英文按钮
21         bCN = new JButton("中文按钮"); // 创建中文按钮
22         JPanel bp = new JPanel(); // 创建放置按钮的子面板 (默认流式布局)
23         bp.add(bEN); bp.add(bCN); // 将两个按钮放入按钮面板
24         // 初始化信息显示区标签
25         msg.setOpaque(true); // 如需设置组件的背景色，首先需将背景设为不透明
26         msg.setBackground(Color.WHITE); // 设置标签的背景色
27         msg.setText("Information area(信息显示区)"); // 设置标签里的文本内容
28         // 将按钮面板和信息标签放入窗口的内容面板
29         Container cp = getContentPane(); // 获得窗口的内容面板 (默认边框布局)
30         cp.add(bp, BorderLayout.NORTH); // 将按钮面板放在内容面板的上部
31         cp.add(msg, BorderLayout.CENTER); // 将信息标签放在内容面板的中间
32         cp.validate(); // 检查并自动布局容器里的组件
33     } }
```



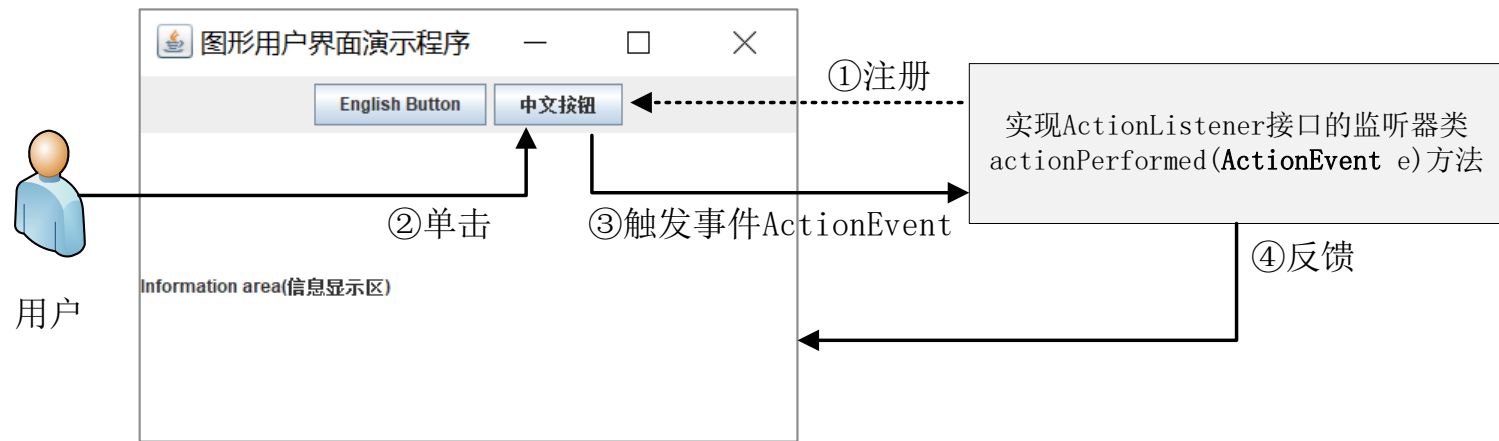
## 6.3 响应用户操作

- Java事件响应机制
  - 用户操作某个组件：用户触发了某种**事件**（event）
  - 用户所操作的组件：**事件源**（event source）
  - **事件类**：区分用户的不同操作
  - 每种事件一个算法接口：**监听器**（listener）接口



## 6.3 响应用户操作

- Java事件响应机制



- 定义实现某个监听器接口的**监听器类**
- 为图形组件**注册**（或称**添加**）一个**监听器对象**
- 当**用户**操作图形组件触发某个事件时，**Java虚拟机**会自动调用该图形组件预先注册好的**监听器对象**中的**处理方法**



## 6.3 响应用户操作

- Java事件响应机制



例6-7 在例6-6代码基础上添加事件响应机制的HelloWorld程序例子（JButtonTest.java）

```
1~9 ..... // 第1~9行与例6-6相同，此处省略
10 class MainWnd extends JFrame { // 扩展JFrame
11     private JButton bEN, bCN; // 添加两个功能按钮
12     private JLabel msg = new JLabel(); // 添加一个信息显示区标签
13     public MainWnd() { // 构造方法
14~32 ..... // 第14~32行与例6-6相同，此处省略
33
34     // 新添加代码：为“中文按钮”注册一个处理ActionEvent事件的监听器对象
35     bCN.addActionListener( new BcnClicked() );
36 } }
37
38 // 新添加代码：定义一个处理ActionEvent事件的监听器类BcnClicked
39 class BcnClicked implements ActionListener { // 需实现规定的接口ActionListener
40     public void actionPerformed(ActionEvent e) { // 实现接口的抽象方法actionPerformed()
41         msg.setText("你好，中国！"); // 在信息标签msg中显示反馈信息
42     } }
```



中國農業大學

閻道宏

## 6.3 响应用户操作

- Java事件响应机制
  - 简化事件监听器代码：**匿名类**
    - 如果一个类继承某个超类或实现了某个接口，并且这个类仅被用于创建一个对象，则可以使用**匿名类**的语法形式来简化程序代码
      - **删除**监听器类BcnClicked定义代码
      - 使用匿名类**直接**创建监听器对象

```
bCN.addActionListener( new ActionListener() {  
    public void actionPerformed(ActionEvent e) {  
        msg.setText( "你好，世界！ ");  
    }  
});
```



## 6.3 响应用户操作

- Java事件响应机制
  - 简化事件监听器代码：**匿名方法**
    - 如果一个接口只包含一个抽象方法，则接口被称为**功能接口**
    - 如果一个类只实现了一个功能接口，并且没有定义任何其他成员，则该类只包含一个方法成员，这样的类被称为**功能类**
    - 以匿名类形式实现的功能类被称为**匿名功能类**
    - 可以使用**匿名方法**（或称为**Lambda表达式**）的语法形式来简化匿名功能类的代码
  - 使用匿名方法**直接**创建监听器对象

```
bCN.addActionListener( (ActionEvent e) -> {  
    msg.setText( "你好，世界！ ");  
});
```





## 6.3 响应用户操作

- Java事件响应机制
  - 多个图形组件共用监听器对象

```
ActionListener bl = new ActionListener() {           // 创建监听器对象bl
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == bEN)
            msg.setText( "Hello, World!");
        else if (e.getSource() == bCN)
            msg.setText( "你好， 中国！ ");
    }
};
```

```
bEN.addActionListener( bl ); // 中英文按钮共用监听器对象bl
bCN.addActionListener( bl );
```



中國農業大學

阚道宏

# 6.3 响应用户操作

- 常用事件类及其监听器接口
  - Java API中常用的事件类

表6-1 常用事件类（java.awt.event包）

事件类	说明	监听器接口
ActionEvent	点击按钮、选择菜单、在编辑框按回车键确认输入等操作将触发ActionEvent事件	ActionListener
ItemEvent	点击单选按钮、勾选复选框、在下拉列表中选择列表选项等操作将触发ItemEvent事件	ItemListener
TextEvent	在编辑框中编辑文本将触发TextEvent事件	TextListener
AdjustmentEvent	拖动卷滚条将触发AdjustmentEvent事件	AdjustmentListener
ComponentEvent	改变组件位置或大小、显示或隐藏组件等操作将触发ComponentEvent事件	ComponentListener
MouseEvent	移动鼠标、按压鼠标按键等操作将触发MouseEvent事件，所有图形组件都有这个事件	MouseListener MouseMotionListener
KeyEvent	按压键盘按键将触发KeyEvent事件，所有图形组件都有这个事件	KeyListener



## 6.3 响应用户操作

- 常用事件类及其监听器接口
  - Java API中各事件类的监听器接口

java.awt.event. <b>ActionListener</b> 接口说明文档			
public interface <b>ActionListener</b> extends <b>EventListener</b>			
	修饰符	接口成员（功能接口）	功能说明
1		void <b>actionPerformed</b> (ActionEvent e)	处理ActionEvent事件的方法

java.awt.event. <b>ItemListener</b> 接口说明文档			
public interface <b>ItemListener</b> extends <b>EventListener</b>			
	修饰符	接口成员（功能接口）	功能说明
1		void <b>itemStateChanged</b> (ItemEvent e)	处理ItemEvent事件的方法



# 6.3 响应用户操作

- 常用事件类及其监听器接口
  - Java API中各事件类的监听器接口

java.awt.event.MouseListener接口说明文档			
public interface <b>MouseListener</b> extends <b>EventListener</b>			
	修饰符	接口成员（全部）	功能说明
1		void <b>mouseClicked</b> (MouseEvent e)	处理单击鼠标按键事件的方法
2		void <b>mouseEntered</b> (MouseEvent e)	处理鼠标进入组件事件的方法
3		void <b>mouseExited</b> (MouseEvent e)	处理鼠标退出组件事件的方法
4		void <b>mousePressed</b> (MouseEvent e)	处理鼠标键被按下事件的方法
5		void <b>mouseReleased</b> (MouseEvent e)	处理鼠标键被松开事件的方法

java.awt.event.KeyListener接口说明文档			
public interface <b>KeyListener</b> extends <b>EventListener</b>			
	修饰符	接口成员（全部）	功能说明
1		void <b>keyPressed</b> (KeyEvent e)	处理按下键盘按键事件的方法
2		void <b>keyReleased</b> (KeyEvent e)	处理松开键盘按键事件的方法
3		void <b>keyTyped</b> (KeyEvent e)	处理敲击键盘按键事件的方法



# 6.4 常用图形组件

- 超类**JComponent**

javax.swing.JComponent类说明文档			
public abstract class JComponent			
extends Container			
implements Serializable			
	修饰符	类成员（节选）	功能说明
1		void grabFocus()	申请键盘输入焦点
2		void setOpaque(boolean isOpaque)	设置是否不透明，即是否启用背景色
3		void setBorder(Border border)	设置边框
4		void setAutoscrolls(boolean autoscrolls)	设置是否自动滚动
5		void setEnabled(boolean enabled)	设置组件是否可用，若不可用则变灰
.....			



# 6.4 常用图形组件

- 按钮类 **JButton**

javax.swing.JButton类说明文档			
public class <b>JButton</b> extends <b>AbstractButton</b> implements <b>Accessible</b>			
	修饰符	类成员（节选）	功能说明
1		<b>JButton()</b>	构造方法
2		<b>JButton</b> (String text)	构造方法（按钮名称）
3		<b>JButton</b> (Icon icon)	构造方法（按钮的图标）
4		<b>JButton</b> (String text, Icon icon)	构造方法（名称和图标）
5		void <b>setText</b> (String text)	设置按钮名称
6		String <b>getText</b> ()	读取按钮名称
7		void <b>setHorizontalTextPosition</b> (int textPosition)	设置名称的水平对齐方式
8		void <b>setVerticalTextPosition</b> (int textPosition)	设置名称的垂直对齐方式
9	protected	void <b>fireActionPerformed</b> (ActionEvent event)	触发ActionEvent事件
10		void <b>addActionListener</b> (ActionListener l)	添加ActionEvent监听器
.....			



# 6.4 常用图形组件

- 标签类**JLabel**

javax.swing.JLabel类说明文档			
public class JLabel			
extends JComponent			
implements SwingConstants, Accessible			
	修饰符	类成员（节选）	功能说明
1		<b>JLabel()</b>	构造方法
2		<b>JLabel(String text)</b>	构造方法（文本信息）
3		void <b>setText</b> (String text)	在标签上显示文本信息
4		String <b>getText</b> ()	读取标签上的文本信息
5		void <b>setHorizontalTextPosition</b> (int textPosition)	设置文本的水平对齐方式
6		void <b>setVerticalTextPosition</b> (int textPosition)	设置文本的垂直对齐方式
7		void <b>setIcon</b> (Icon icon)	在标签上显示图标（图像）
8		Icon <b>getIcon</b> ()	读取标签上的图标（图像）
.....			



## 6.4 常用图形组件

- 文本组件类
  - 使用文本编辑框接收用户的**键盘输入**，输入结果为**字符串**（String）类型
  - Java API为文本编辑框定义了两个类
    - 文本字段类**JTextField**，用于**单行**输入
    - 文本区域类**JTextArea**，用于编辑**多行**文本
  - 这两个类是从同一个文本组件类**JTextComponent**继承并扩展而来的





# 6.4 常用图形组件

- 文本组件类
  - 文本组件类 **JTextComponent**

javax.swing.text.JTextComponent类说明文档			
public abstract class JTextComponent			
extends JComponent			
implements Scrollable, Accessible			
	修饰符	类成员（节选）	功能说明
1		<b>JTextComponent()</b>	构造方法
2		String <b>getText()</b>	读出文本字符串
3		void <b>setText</b> (String t)	设置文本字符串
4		void <b>setEditable</b> (boolean b)	设置是否可编辑
5		void <b>select</b> (int selectionStart, int selectionEnd)	选中指定范围内的文本
6		void <b>selectAll</b> ()	选中全部文本
7		String <b>getSelectedText</b> ()	读出选中的文本
8		void <b>replaceSelection</b> (String content)	替换选中的文本
9		void <b>copy</b> ()	将选中的文本复制到剪贴板
10		void <b>cut</b> ()	将选中的文本剪切到剪贴板
11		void <b>paste</b> ()	粘贴剪贴板里的文本



## 6.4 常用图形组件

- 文本组件类
  - 文本字段类 **JTextField**
    - 使用文本字段类 **JTextField** 实现 **单行** 文本编辑框功能
    - 用户在文本编辑框中输入内容，按 **回车键**（Enter）表示结束输入，此时将触发 **ActionEvent** 事件
    - 程序员需使用文本字段类 **JTextField** **创建** 单行编辑框对象，并为其 **添加** 处理 **ActionEvent** 事件的 **ActionListener** 监听器



## 6.4 常用图形组件

例6-8 一个文本字段类JTextField的Java演示程序（JTextFieldTest.java）

```
1 import java.awt.*; // 导入java.awt包中的类
2 import java.awt.event.*; // 导入java.awt.event包中定义的事件类
3 import javax.swing.*; // 导入javax.swing包中的类
4
5 public class JTextFieldTest { // 主类
6     public static void main(String[] args) { // 主方法
7         MainWnd w = new MainWnd(); // 创建并显示程序主窗口
8     }
9
10    class MainWnd extends JFrame { // 扩展JFrame
11        JTextField tf = new JTextField(); // 添加一个单行文本编辑框
12        JLabel msg = new JLabel("Hello, World!"); // 添加一个显示信息的标签
13        public MainWnd() { // 构造方法
14            setTitle("图形界面演示程序"); // 初始化窗口
15            setSize(300, 200); setLocation(100, 100); setVisible(true);
16            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
17            // 设置单行文本编辑框tf
18            tf.setBackground(Color.YELLOW); // 设置背景色
19            tf.addActionListener(new ActionListener() { // 添加ActionEvent事件监听器
20                public void actionPerformed(ActionEvent e) {
21                    msg.setText("Hello, " + tf.getText());
22                }
23            });
24            // 在窗口的内容面板上添加组件tf和msg
25            Container cp = getContentPane(); // 获得窗口的内容面板（默认边框布局）
26            cp.add(tf, BorderLayout.NORTH); cp.add(msg, BorderLayout.CENTER);
27            cp.validate(); // 检查并自动布局容器里的组件
28        }
29    }
30 }
```



## 6.4 常用图形组件

- 文本组件类
  - 文本字段类 **JTextField**

javax.swing.JTextField类说明文档			
public class <b>JTextField</b> extends <b>JTextComponent</b> implements <b>SwingConstants</b>			
	修饰符	类成员（节选）	功能说明
1		<b>JTextField()</b>	构造方法
2		<b>JTextField(int columns)</b>	构造方法
3		<b>JTextField(String text)</b>	构造方法
4		<b>JTextField(String text, int columns)</b>	构造方法
5	protected	void <b>fireActionPerformed()</b>	触发ActionEvent事件
6		void <b>addActionListener(ActionListener l)</b>	添加ActionEvent监听器
.....			



## 6.4 常用图形组件

- 文本组件类
  - 文本区域类 **JTextArea**
    - 使用文本区域类 **JTextArea** 实现 **多行** 文本编辑框的功能
    - 在多行文本编辑框中按回车键 **不会** 触发 **ActionEvent** 事件。程序员应 **另外** 添加组件（例如按钮），为用户提供操作多行文本编辑框的功能
    - 通常将多行文本编辑框放入一个 **滚动面板**（**JScrollPane**）。当文本内容超出显示区域时，编辑框将自动显示出卷滚条进行 **滚动**



## 6.4 常用图形组件

例6-9 一个文本区域类JTextArea的Java演示程序 (JTextAreaTest.java)

```
1~9 ..... // 第1~9行与例6-8相同, 此处省略。注: 将主类名改为JTe
10 class MainWnd extends JFrame { // 扩展JFrame
11     JTextArea ta = new JTextArea(2, 10); // 添加一个2行10列的
12     JLabel msg = new JLabel(); // 添加一个显示信息的标签
13     JButton b = new JButton("显示文本"); // 添加一个按钮
14     public MainWnd() { // 构造方法
15         setTitle("图形界面演示程序"); // 初始化窗口
16         setSize(300, 200); setLocation(100, 100); setVisible(true);
17         setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
18         // 设置多行文本编辑框ta
19         ta.setBackground(Color.YELLOW); // 设置多行文本编辑框的背景色
20         JScrollPane taScroller = new JScrollPane(ta); // 将编辑框放入一个滚动面板
21         b.addActionListener( new ActionListener() { // 为按钮添加ActionEvent事件监听器
22             public void actionPerformed(ActionEvent e) {
23                 msg.setText( ta.getText() ); // 取出编辑框里的内容, 并显示到标签中
24             }
25         });
26         // 在窗口的内容面板上添加滚动面板taScroller、标签msg和按钮b
27         Container cp = getContentPane(); // 获得窗口的内容面板 (默认边框布局)
28         cp.add( taScroller, BorderLayout.NORTH );
29         cp.add( msg, BorderLayout.CENTER ); cp.add( b, BorderLayout.SOUTH );
30         cp.validate(); // 检查并自动布局容器里的组件
31     } }
```



# 6.4 常用图形组件

- 文本组件类
  - 文本区域类 **JTextArea**

javax.swing.JTextArea类说明文档

public class **JTextArea**

extends **JTextComponent**

	修饰符	类成员（节选）	功能说明
1		<b>JTextArea()</b>	构造方法
2		<b>JTextArea</b> (int rows, int columns)	构造方法
3		<b>JTextArea</b> (String text)	构造方法
4		<b>JTextArea</b> (String text, int rows, int columns)	构造方法
5		void <b>append</b> (String str)	在末尾追加文本
6		void <b>insert</b> (String str, int pos)	在指定位置插入文本
.....			



中國農業大學

閻道宏

# 6.4 常用图形组件

- 单选按钮类与复选框类
  - 如果程序有一组选项，程序员通常会以**单选按钮**或**复选框**的形式让用户进行选择
    - 单选按钮通常为**圆形**，一组单选按钮同时只能选中其中的一项，程序员需使用单选按钮类**JRadioButton**创建单选按钮对象
    - 复选框通常为**方形**，在一组复选框中可以同时勾选多项，程序员需使用复选框类**JCheckBox**创建复选框对象
  - 用户点击单选按钮或勾选复选框时会同时触发**ItemEvent**事件和**ActionEvent**事件。程序员可为单选按钮、复选框添加
    - 响应**ItemEvent**事件的**ItemListener**监听器
    - 或添加响应**ActionEvent**事件的**ActionListener**监听器
    - 二者任选其一
  - 单选按钮类**JRadioButton**和复选框类**JCheckBox**都是从抽象按钮类**AbstractButton**继承并扩展而来的
  - **注：**按钮类**JButton**也是从抽象按钮类**AbstractButton**继承并扩展而来的





# 6.4 常用图形组件

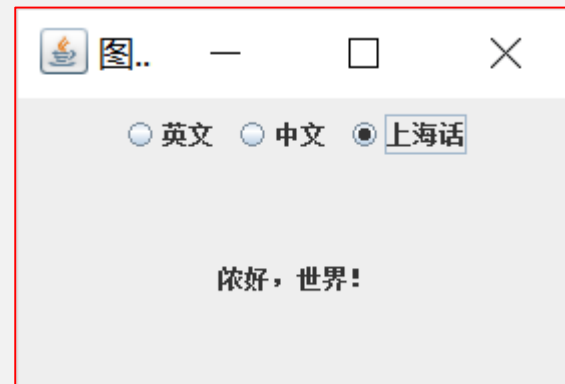
- 单选按钮类与复选框类
  - 抽象按钮类 **AbstractButton**

javax.swing. <b>AbstractButton</b> 类说明文档			
public abstract class <b>AbstractButton</b> extends <b>JComponent</b> implements <b>ItemSelectable</b> , <b>SwingConstants</b>			
	修饰符	类成员（节选）	功能说明
1		<b>AbstractButton()</b>	构造方法
2		void <b>setText</b> (String text)	设置按钮名称
3		String <b>getText</b> ()	读取按钮名称
4		void <b>setHorizontalTextPosition</b> (int textPosition)	设置名称的水平对齐方式
5		void <b>setVerticalTextPosition</b> (int textPosition)	设置名称的垂直对齐方式
6		void <b>setEnabled</b> (boolean b)	设置是否可用（是否可选）
7		void <b>setSelected</b> (boolean b)	设置选中状态
8		boolean <b>isSelected</b> ()	检查是否被选中
9	protected	void <b>fireActionPerformed</b> (ActionEvent event)	触发ActionEvent事件
10		void <b>addActionListener</b> (ActionListener l)	添加ActionEvent监听器
11	protected	void <b>fireItemStateChanged</b> (ItemEvent event)	触发ItemEvent事件
12		void <b>addItemListener</b> (ItemListener l)	添加ItemEvent监听器
.....			



### 例6-10 一个单选按钮类JRadioButton的Java演示程序（JRadioButtonTest.java）

```
1~9 ..... // 第1~9行与6.4.3小节例6-8相同，此处省略。注：将主类名改为JRadioButtonTest
10 class MainWnd extends JFrame {           // 扩展JFrame
11     JRadioButton cbEN = new JRadioButton("英文", true); // 单选按钮：英文
12     JRadioButton cbCN = new JRadioButton("中文");      // 单选按钮：中文
13     JRadioButton cbSH = new JRadioButton("上海话");    // 单选按钮：上海话
14     JLabel hello = new JLabel("Hello, World!", SwingConstants.CENTER); // 信息标签
15
16     public MainWnd() {           // 构造方法
17         setTitle("图形界面演示程序"); // 初始化窗口
18         setSize(300, 200); setLocation(100, 100); setVisible(true);
19         setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
20         // 新建一个按钮面板，放入3个单选按钮
21         JPanel bp = new JPanel( );           // 创建子面板（默认流式布局）
22         bp.add(cbEN); bp.add(cbCN); bp.add(cbSH); // 添加单选按钮组件
23         // 将3个互斥的单选按钮合成一组，同时只会有一个被选中
24         ButtonGroup group = new ButtonGroup(); // 创建组对象
25         group.add(cbEN); group.add(cbCN); group.add(cbSH); // 加入组中
26         // 在主窗口的内容面板中放入按钮面板bp和标签hello
27         Container cp = getContentPane(); // 获得窗口的内容面板（默认边框布局）
28         cp.add( bp, BorderLayout.NORTH ); // 添加按钮面板bp
29         cp.add( hello, BorderLayout.CENTER ); // 添加信息标签hello
30         cp.validate(); // 检查并自动布局容器里的组件
31         // 处理ItemEvent事件的监听器：根据单选按钮状态来显示对应的信息
32         ItemListener il = new ItemListener() { // 匿名类
33             public void itemStateChanged(ItemEvent e) { // 处理ItemEvent事件的方法
34                 String msg = null;
35                 if ( cbEN.isSelected() ) msg = "Hello, World!";
36                 else if ( cbCN.isSelected() ) msg = "你好，世界！";
37                 else if ( cbSH.isSelected() ) msg = "侬好，世界！";
38                 hello.setText( msg );
39             } };
40         // 三个单选按钮对象共用同一个监听器对象il
41         cbEN.addItemListener(il); cbCN.addItemListener(il); cbSH.addItemListener(il);
42     } }
```

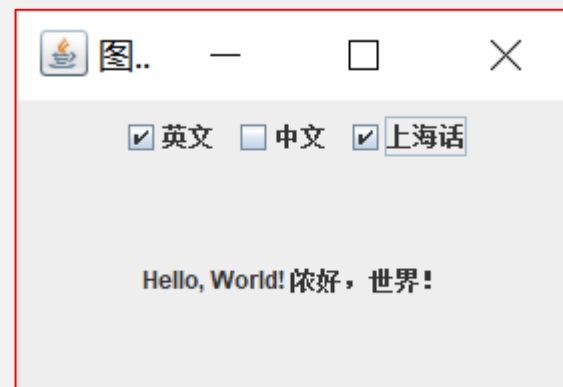


例6-11 一个复选框类JCheckBox的Java演示程序（JCheckBoxTest.java）

```

1~9 ..... // 第1~9行与6.4.3小节例6-8相同，此处省略。注：将主类名改为JCheckBoxTest
10 class MainWnd extends JFrame {           // 扩展JFrame
11     JCheckBox cbEN = new JCheckBox("英文"); // 复选框：英文
12     JCheckBox cbCN = new JCheckBox("中文"); // 复选框：中文
13     JCheckBox cbSH = new JCheckBox("上海话"); // 复选框：上海话
14     JLabel hello = new JLabel("", SwingConstants.CENTER); // 信息标签
15
16     public MainWnd() {                     // 构造方法
17         setTitle("图形界面演示程序"); // 初始化窗口
18         setSize(300, 200); setLocation(100, 100); setVisible(true);
19         setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
20         // 新建一个复选框面板，放入3个复选框
21         JPanel bp = new JPanel( );         // 创建子面板（默认流式布局）
22         bp.add(cbEN); bp.add(cbCN); bp.add(cbSH); // 添加复选框组件
23         // 在主窗口的内容面板中放入复选框面板bp + 标签hello
24         Container cp = getContentPane(); // 获得窗口的内容面板（默认边框布局）
25         cp.add( bp, BorderLayout.NORTH ); // 添加复选框面板bp
26         cp.add( hello, BorderLayout.CENTER ); // 添加信息标签hello
27         cp.validate();                     // 检查并自动布局容器里的组件
28         // 处理ItemEvent事件的监听器：根据复选框状态来显示对应的信息
29         ItemListener il = new ItemListener() { // 匿名类
30             public void itemStateChanged(ItemEvent e) { // 处理ItemEvent事件的方法
31                 String msg = "";
32                 if ( cbEN.isSelected() ) msg += "Hello, World! ";
33                 if ( cbCN.isSelected() ) msg += "你好，世界！ ";
34                 if ( cbSH.isSelected() ) msg += "侬好，世界！ ";
35                 if ( msg.equals("") ) hello.setText("没有勾选项！ ");
36                 else hello.setText( msg );
37             } };
38         // 三个复选框对象共用一个事件监听器
39         cbEN.addItemListener(il); cbCN.addItemListener(il); cbSH.addItemListener(il);
40     } }

```



## 6.4 常用图形组件

- 列表类

- 图形用户界面也会以**列表**或**下拉列表**的形式让用户在**一组选项**中进行选择。列表、下拉列表比单选按钮或复选框节省屏幕空间
  - 程序员使用列表类**JList<E>**创建列表对象
  - 使用下拉列表类**JComboBox<E>**创建下拉列表对象
- 列表类、下拉列表类采用**泛型编程**，其中的列表选项可以是任意引用类型。程序通常使用字符串形式描述列表选项，例如**JList<String>**、**JComboBox<String>**
- 列表类**JList<E>**以列表格式接收用户**输入**，这时列表类**JList<E>**被当作输入组件使用
- 列表类**JList<E>**也可以被当作输出组件使用，即以列表格式向用户**输出**信息。输出组件通常不需要响应事件
- 另一种常用的输出组件是表格类**JTable**



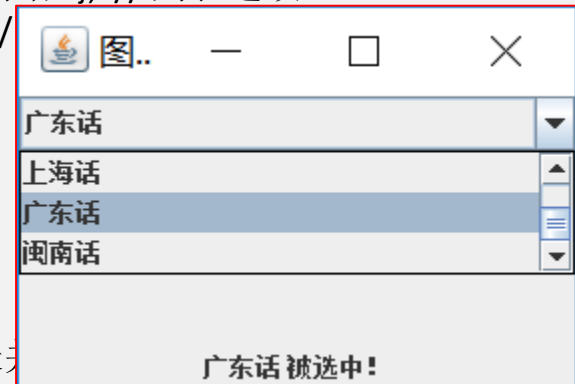
## 6.4 常用图形组件

- 列表类
  - 下拉列表类 **JComboBox<E>**
    - 用户选择下拉列表中的选项，这会同时触发 **ItemEvent** 事件和 **ActionEvent** 事件
  - 程序员可为下拉列表添加
    - 响应 **ItemEvent** 事件的 **ItemListener** 监听器
    - 或添加响应 **ActionEvent** 事件的 **ActionListener** 监听器
    - 二者任选其一



### 例6-12 一个下拉列表类JComboBox<E>的Java演示程序（JComboBoxTest.java）

```
1~9 ..... // 第1~9行与6.4.3小节例6-8相同，此处省略。注：将主类名改为JComboBoxTest
10 class MainWnd extends JFrame { // 扩展JFrame
11     JComboBox<String> list; // 字符串型下拉列表
12     String listItems[] = { "英文", "中文", "上海话", "广东话", "闽南话" }; // 列表选项
13     JLabel info = new JLabel("", SwingConstants.CENTER); //
14
15     public MainWnd() { // 构造方法
16         setTitle( "图形界面演示程序" ); // 初始化窗口
17         setSize(300, 200); setLocation(100, 100); setVisible(true);
18         setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
19         // 设置下拉列表list
20         list = new JComboBox<String>( listItems ); // 创建下拉列表
21         list.setMaximumRowCount( 3 ); // 设置下拉行数
22         list.setSelectedIndex( 1 ); // 设置初始选中的列表选项（编号从0开始）
23         // 在窗口的内容面板上添加组件
24         Container cp = getContentPane(); // 获得窗口的内容面板（默认边框布局）
25         cp.add( list, BorderLayout.NORTH ); // 将下拉列表添加到主窗口
26         cp.add( info, BorderLayout.SOUTH ); // 将信息显示标签添加到主窗口
27         cp.validate(); // 检查并自动布局容器里的组件
28         // 处理ItemEvent事件的监听器：根据下拉列表选中的选项来显示对应的信息
29         ItemListener il = new ItemListener() { // 匿名类
30             public void itemStateChanged(ItemEvent e) { // 处理ItemEvent事件的方法
31                 JComboBox cb = (JComboBox)e.getSource(); // 获取事件源
32                 String item = (String)cb.getSelectedItem(); // 获取被选中的选项
33                 info.setText( item + " 被选中！" );
34             } };
35         list.addItemListener(il); // 为下拉列表添加ItemListener监听器
36     } }
```



## 6.4 常用图形组件

- 列表类

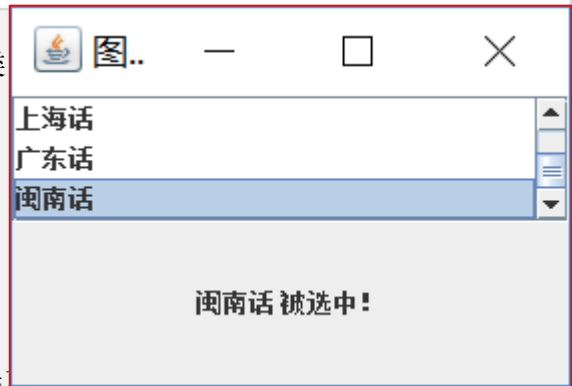
- 列表类 **JList**<E>

- 用户选择列表里的选项时会触发 **ListSelectionEvent** 事件
    - 程序员需为列表添加响应这个事件的 **ListSelectionListener** 监听器
  - 之前用到的事件类都是 **awt** 定义的，而列表类 **JList**<E> 用到的事件类 **ListSelectionEvent** 则是 **swing** 新增加的
  - 处理 **ListSelectionEvent** 事件需导入 **javax.swing.event** 包中定义的事件类



例6-13 一个列表类JList<E>的Java演示程序（JListTest.java）

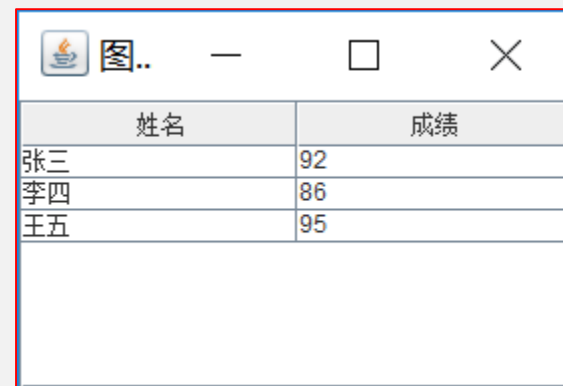
```
1 import java.awt.*; // 导入java.awt包中定义的类
2 //import java.awt.event.*; // 本例不需要导入java.awt.event包中定义的事件类
3 import javax.swing.*; // 导入javax.swing包中定义的类
4 import javax.swing.event.*; // 导入javax.swing.event包中定义的事件类
5 public class JListTest { // 主类
6     public static void main(String[] args) { // 主方法
7         MainWnd w = new MainWnd(); // 创建并显示主窗口对象
8     } }
9 class MainWnd extends JFrame { // 扩展JFrame
10     JList<String> list; // 字符串型下拉列表
11     String listItems[] = { "英文", "中文", "上海话", "广东话", "闽南话" }; // 列表选项
12     JLabel info = new JLabel("", SwingConstants.CENTER); // 信息标签
13     public MainWnd() { // 构造方法
14         setTitle( "图形界面演示程序" ); // 初始化窗口
15         setSize(300, 200); setLocation(100, 100); setVisible(true);
16         setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
17         // 设置列表list
18         list = new JList<String>( listItems ); // 创建列表并初始化列表选项
19         list.setSelectionMode( ListSelectionModel.SINGLE_SELECTION ); // 设置单选或多选
20         list.setLayoutOrientation( JList.VERTICAL ); // 设置纵向或横向布局
21         list.setVisibleRowCount( 3 ); // 设置行数
22         list.setSelectedIndex( 1 ); // 设置初始选中的列表选项（编号从0开始）
23         // 如果列表选项比较多，需将列表放入一个卷滚面板
24         JScrollPane listScroller = new JScrollPane(list);
25         // 在窗口的内容面板上添加组件
26         Container cp = getContentPane(); // 获得窗口的内容面板（默认边框布局）
27         cp.add( listScroller, BorderLayout.NORTH ); // 将列表卷滚面板添加到主窗口
28         cp.add( info, BorderLayout.CENTER ); // 将信息显示标签添加到主窗口
29         cp.validate(); // 检查并自动布局容器里的组件
30         // 处理ListSelectionEvent事件的监听器：根据列表选中的选项来显示对应的信息
31         ListSelectionListener lsl = new ListSelectionListener () { // 匿名类
32             public void valueChanged(ListSelectionEvent e) { // 处理ListSelectionEvent事件
33                 int index = list.getSelectedIndex(); // 获取被选中选项的序号
34                 if (index == -1) info.setText( "无" ); // 没有选项被选中
35                 else info.setText( listItems[index] + " 被选中！" ); // 显示被选中的选项
36             } };
37         list.addListSelectionListener( lsl ); // 为列表list添加选择事件监听器对象lsl
38     } }
```





### 例6-14 一个表格类JTable的Java演示程序（JTableTest.java）

```
1 import java.awt.*; // 导入java.awt包中定义类
2 //import java.awt.event.*; // 本例不需要导入java.awt.event包中的事件类
3 import javax.swing.*; // 导入javax.swing包中定义类
4
5 public class JTableTest { // 主类
6     public static void main(String[] args) { // 主方法
7         MainWnd w = new MainWnd(); // 创建并显示主窗口对象
8     } }
9
10 class MainWnd extends JFrame { // 扩展JFrame
11     JTable list; // 二维表格
12     public MainWnd() { // 构造方法
13         setTitle("图形界面演示程序"); // 初始化窗口
14         setSize(300, 200); setLocation(100, 100); setVisible(true);
15         setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
16         // 设置表格以及所要显示的数据
17         String columnNames[] = { "姓名", "成绩" }; // 表格头（表格的列名）
18         Object data[][] = { { "张三", 92 }, { "李四", 86 }, { "王五", 95 } }; // 表格数据
19         list = new JTable(data, columnNames); // 创建表格对象，初始化表格头和数据
20         // 如果表格行数较多，需将表格放入一个卷滚面板
21         JScrollPane listScroller = new JScrollPane(list);
22         list.setFillsViewportHeight(true); // 设置是否填满显示区域
23         // 在窗口的内容面板上添加组件
24         Container cp = getContentPane(); // 获得窗口的内容面板（默认边框布局）
25         cp.add( listScroller, BorderLayout.CENTER ); // 将表格卷滚面板添加到主窗口
26         cp.validate(); // 检查并自动布局容器里的组件
27     } }
```



## 6.4 常用图形组件

- 菜单类
  - 图形用户界面可以让用户选择程序功能。**菜单**是占用屏幕空间最少的一种形式
  - 在框架窗口JFrame中**添加菜单**需分如下四步完成
    - 在框架窗口中添加一个菜单栏类**JMenuBar**的对象
    - 在菜单栏**JMenuBar**对象中添加一级菜单类**JMenu**的对象
    - 在一级菜单**JMenu**对象中添加二级菜单项类**JMenuItem**的对象
    - 为二级菜单项**JMenuItem**对象添加**事件监听器**，用于实现菜单所对应的程序功能
  - 用户选择菜单会同时触发**ItemEvent**事件和**ActionEvent**事件。程序员需要为每个二级菜单项添加
    - 响应**ItemEvent**事件的**ItemListener**监听器
    - 或添加响应**ActionEvent**事件的**ActionListener**监听器
    - 二者任选其一



例 6-15 一个

```

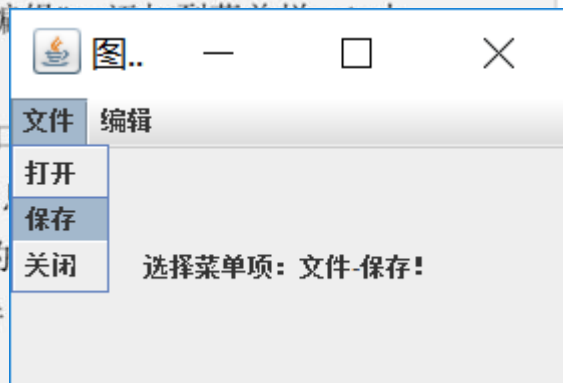
1~9 .....
10 class
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33

```

```

34 // 在框架窗口中添加菜单
35 setJMenuBar( mb ); // ①在框架窗口中添加菜单栏对象 mb
36 JMenu m; JMenuItem mi; // 定义局部引用变量 m 和 mi
37 // 新建并添加一级菜单"文件"
38 m = new JMenu("文件"); // ②新建一级菜单"文件"
39 mi = new JMenuItem("打开"); // ③二级菜单项"打开"
40 mi.addActionListener( al ); // ④为二级菜单项 mi 添加动作事件监听器 al
41 m.add( mi ); // ⑤将二级菜单项 mi 添加到一级菜单 m 中
42 mi = new JMenuItem("保存"); mi.addActionListener(al); m.add(mi); // "保存"
43 mi = new JMenuItem("关闭"); mi.addActionListener(al); m.add(mi); // "关闭"
44 mb.add( m ); // ⑥将一级菜单"文件"m 添加到菜单栏 mb 中
45 // 新建并添加一级菜单"编辑"
46 m = new JMenu("编辑"); // 新建一级菜单"编辑"
47 mi = new JMenuItem("复制"); mi.addActionListener(al); m.add(mi); // "复制"
48 mi = new JMenuItem("剪切"); mi.addActionListener(al); m.add(mi); // "剪切"
49 m.addSeparator(); // 可在菜单中增加分隔线，用于功能分组
50 mi = new JMenuItem("粘贴"); mi.addActionListener(al); m.add(mi); // "粘贴"
51 mb.add( m ); // 将一级菜单"编辑"添加到菜单栏 mb 中
52 // 窗口及其内容面板的布局
53 Container cp = getContentPane(); // 获得窗口内容面板
54 cp.add( info, BorderLayout.CENTER ); // 添加信息面板
55 cp.validate(); // 检查并自动布局内容面板里的组件
56 validate(); // 检查并自动布局窗口里的组件
57 } }

```



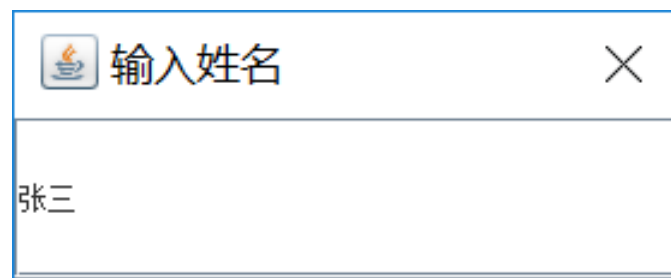
## 6.5 对话框

- 程序运行过程中，如果中途需要接收**用户**的指令或信息，然后再继续下一步运行，此时程序主窗口可以**单独弹出一个对话框**（dialog）窗口来接收用户的输入
- **对话框窗口**通常由框架窗口（JFrame，即程序主窗口）弹出，是隶属于框架窗口的**子窗口**
- 换句话说，**框架窗口**是对话框窗口的**父窗口**。关闭框架窗口会同时关闭其所属的对话框子窗口
- 和框架窗口一样，对话框窗口也是**顶级容器**，可以包含其他组件，但对话框窗口**不能**添加**菜单栏**
- 将程序中的部分界面功能**独立**出来，单独设计一个对话框窗口，这样可以减轻程序主窗口的负担



# 6.5 对话框

- 对话框类 **JDialog**



javax.swing.JDialog类说明文档

public class **JDialog**

extends **Dialog**

implements **WindowConstants, Accessible, RootPaneContainer**

	修饰符	类成员（节选）	功能说明
1		<b>JDialog()</b>	构造方法
2		<b>JDialog</b> (Frame owner, String title)	构造方法
3		<b>JDialog</b> (Frame owner, String title, boolean modal)	构造方法
4		void <b>setTitle</b> (String title)	设置对话框标题
5		void <b>setModal</b> (boolean modal)	设置是否模式对话框
6		Window <b>getOwner</b> ()	获取父窗口
7		Container <b>getContentPane</b> ()	获取内容面板
8		void <b>setLayout</b> (LayoutManager manager)	设置布局管理器
9	protected	void <b>dialogInit</b> ()	初始化对话框的设置
10		void <b>setDefaultCloseOperation</b> (int operation)	设置关闭对话框时的默认操作
11		void <b>dispose</b> ()	释放资源，关闭对话框

.....



# 6.5 对话框

- 对话框类**JDialog**
  - 对话框分**模式**（modal，或称模态）和**非模式**（modeless，或称非模态）两种
    - **模式对话框**打开后，用户**必须完成**对话框所规定的的数据输入或功能选择，然后才能返回父窗口，继续操作程序。简单地说，模式对话框打开后，本程序中的其他窗口都被禁止操作
    - 打开**非模式对话框**并不会影响用户操作本程序中的其他窗口，非模式对话框可以与其他窗口**同时操作**
    - 模式对话框和非模式对话框在**界面设计**和**代码实现**上有一些区别




例 6-16 使用模式对话框实现

```

1~9 ..... // 第 1~9 行
10 class MainWnd e
11     JButton btn =
12     JLabel info = r
13     JFrame fWin;
14     public MainW
15         setTitle(
16         setSize(
17         setDefa
18         fWin = t
19         // 主窗
20         Contain
21         cp.add(
22         cp.valid
23         btn.add
24         pu
25
26

```

 输入姓名

张三

确定

```

29
30 class NameDialogM extends JDialog {           // 对话框类：扩展 JDialog
31     JTextField name = new JTextField("张三"); // 单行文本框：输入姓名
32     JButton ok = new JButton("确定");         // 按钮：确认输入
33     JDialog dWin; // 保存对话框窗口的引用，事件监听器需要访问该字段
34     public NameDialogM(JFrame pw) {          // 构造方法，接收参数：父窗口
35         super(pw);                             // 调用超类 JDialog 的构造方法，传递父窗口
36         setModal(true);                         // 设置为模式对话框
37         setTitle("输入姓名"); setSize(300, 150); setLocation(200, 200);
38         setDefaultCloseOperation( WindowConstants.HIDE_ON_CLOSE );
39         dWin = this;                             // 保存当前对话框窗口的引用
40         // 对话框内容面板：单行文本框 +按钮
41         Container cp = getContentPane(); // 获得对话框的内容面板（默认边框布局）
42         cp.add( name, BorderLayout.CENTER ); cp.add( ok, BorderLayout.SOUTH );
43         cp.validate();                          // 检查并自动布局对话框内容面板里的组件
44         // 为单行编辑框和按钮添加事件监听器：接收输入，关闭对话框
45         ActionListener al = new ActionListener() { // 匿名类
46             public void actionPerformed(ActionEvent e) { // 处理 ActionEvent 事件的方法
47                 MainWnd w = (MainWnd)dWin.getOwner(); // 获取父窗口
48                 w.info.setText("Hello, " +name.getText()); // 读取并显示姓名
49                 dWin.setVisible(false);                // 隐藏对话框窗口
50                 dWin.dispose();                          // 释放资源，关闭对话框
51             } };
52         name.addActionListener( al ); // 输入姓名时按回车键会触发 ActionEvent 事件
53         ok.addActionListener( al );   // 点击“确认”按钮会触发 ActionEvent 事件
54     } }

```



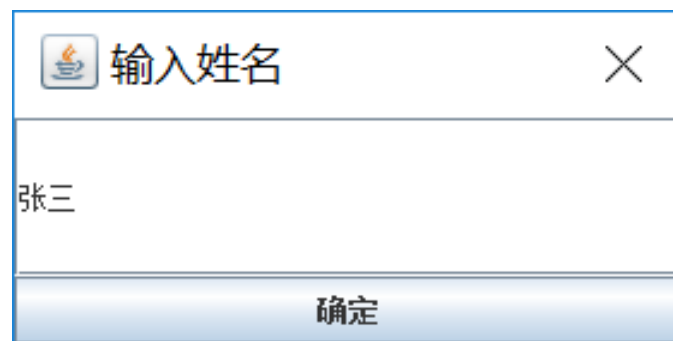
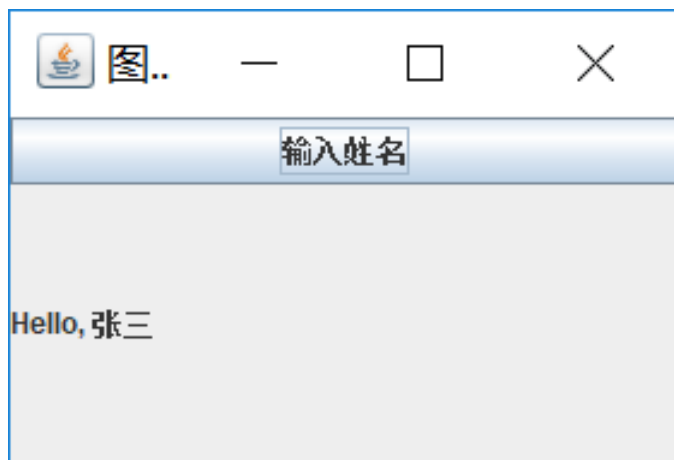
例 6-17 使用非模式对话框实现类似图 6-23 所示程序功能的 Java 示例程序 (JDialogModalelessTest.java)

```
1~9      31  class NameDialogMless extends JDialog {    // 对话框类：扩展 JDialog
10      32      JTextField name = new JTextField("张三"); // 单行文本框：输入姓名
11      33      JDialog dWin; // 保存对话框窗口的引用，事件监听器需要访问该字段
12      34      public NameDialogMless(JFrame pw) {    // 构造方法
13      35          super(pw);    // 调用超类 JDialog 的构造方法，传递父窗口
14      36          setModal(false); // 设置为非模式对话框
15      37          setTitle("输入姓名"); setSize(300, 100); setLocation(400, 200);
16      38          setDefaultCloseOperation( WindowConstants.HIDE_ON_CLOSE );
17      39          dWin = this;    // 保存当前对话框窗口的引用
18      40          // 对话框内容面板：只有一个单行文本框
19      41          Container cp = getContentPane(); // 获得对话框的内容面板（默认边框布局）
20      42          cp.add( name, BorderLayout.CENTER ); cp.validate();
21      43          // 为单行编辑框添加事件监听器：接收输入，输入后不需要关闭对话框
22      44          ActionListener al = new ActionListener() { // 匿名类
23      45              public void actionPerformed(ActionEvent e) {
24      46                  MainWnd w = (MainWnd)dWin.getOwner(); // 获取父窗口
25      47                  w.info.setText("Hello, " +name.getText()); // 读取并显示姓名
26      48                  // 非模式对话框不是必须关闭，可与程序主窗口并存
27      49              } };
28      50          name.addActionListener( al ); // 输入姓名时按回车键会触发 ActionEvent 事件
29      51      } }
```



## 6.5 对话框

- 对话框类 **JDialog**
  - 非模式对话框



# 6.5 对话框

- 常用对话框
  - 消息（message）对话框
  - 确认（confirm）对话框
  - 输入（input）对话框
  - 选项（option）对话框
- Java API预先设计好了这些对话框功能，并以静态方法的形式提供给程序员使用
- 这些静态方法被统一定义在选项面板类JOptionPane中
- 程序员只要调用选项面板类JOptionPane中的静态方法，就能很方便地实现上述常用的对话框功能
- 另外还有两个比较常用的对话框
  - 文件选择类JFileChooser
  - 颜色选择类JColorChooser



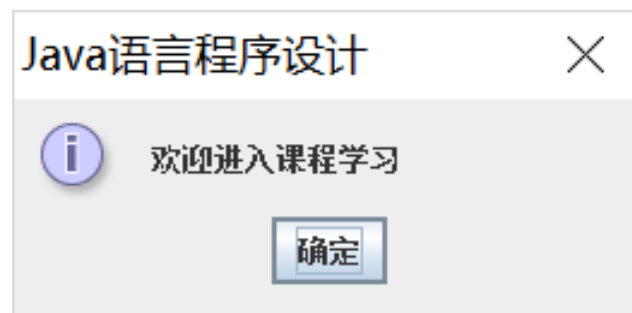
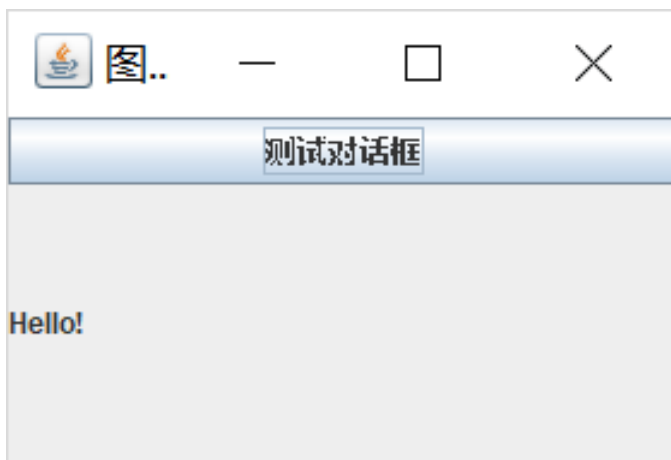
# 6.5 对话框

- 常用对话框
  - 选项面板类 **JOptionPane**

javax.swing.JOptionPane类说明文档			
public class <b>JOptionPane</b> extends <b>JComponent</b> implements <b>Accessible</b>			
	修饰符	类成员（节选）	功能说明
1	static	void <b>showMessageDialog</b> (Component parentComponent, Object message)	消息对话框
2	static	void <b>showMessageDialog</b> (Component parentComponent, Object message, String title, int messageType)	消息对话框
3	static	int <b>showConfirmDialog</b> (Component parentComponent, Object message)	确认对话框
4	static	int <b>showConfirmDialog</b> (Component parentComponent, Object message, String title, int optionType)	确认对话框
5	static	String <b>showInputDialog</b> (Component parentComponent, Object message)	输入对话框
6	static	String <b>showInputDialog</b> (Component parentComponent, Object message, Object initialSelectionValue)	输入对话框
7	static	int <b>showOptionDialog</b> (Component parentComponent, Object message, String title, int optionType, int messageType, Icon icon, Object[] options, Object initialValue)	选项对话框
.....			

## 6.5 对话框

- 常用对话框
  - 选项面板类 **JOptionPane**
    - **消息**对话框



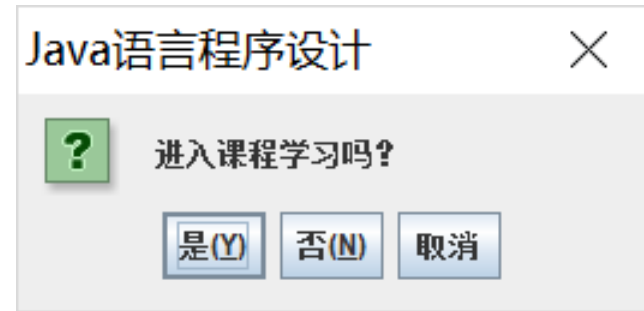
### 例6-18 一个弹出消息对话框的Java演示程序（JOptionPaneTest.java）

```
1 import java.awt.*;    // 导入java.awt包中的类
2 import java.awt.event.*; // 导入java.awt.event包中定义的事件类
3 import javax.swing.*; // 导入javax.swing包中的类
4
5 public class JOptionPaneTest {    // 测试类
6     public static void main(String[] args) { // 主方法
7         JFrame w = new JFrame();    // 创建并显示主窗口对象
8         w.setTitle( "图形界面演示程序" ); // 初始化主窗口
9         w.setSize(300, 200); w.setLocation(100, 100); w.setVisible(true);
10        w.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
11        // 主窗口内容面板：按钮 + 标签
12        Container cp = w.getContentPane(); // 获得窗口的内容面板（默认边框布局）
13        JButton btn = new JButton("测试对话框"); // 按钮：单击按钮弹出对话框
14        JLabel info = new JLabel("Hello!");    // 标签：显示对话框返回的结果
15        cp.add( btn, BorderLayout.NORTH ); cp.add( info, BorderLayout.CENTER );
16        cp.validate();    // 检查并自动布局内容面板里的组件
17        // 为按钮添加事件监听器：用户点击按钮，弹出对话框
18        btn.addActionListener( new ActionListener() { // 匿名类
19            public void actionPerformed(ActionEvent e) { // 弹出消息对话框
20                JOptionPane.showMessageDialog(w, "欢迎进入课程学习",
21                    "Java语言程序设计", JOptionPane.INFORMATION_MESSAGE);
22            } } );
23    } }
```



## 6.5 对话框

- 常用对话框
  - 选项面板类 **JOptionPane**
    - 确认对话框

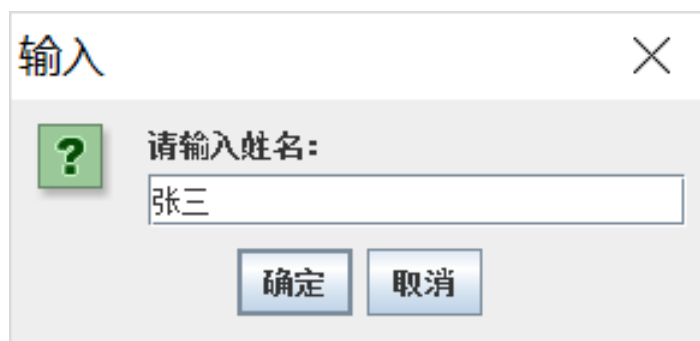


```
int opt;  
opt = JOptionPane.showConfirmDialog(w, "进入课程学习吗? ",  
    "Java语言程序设计", JOptionPane.INFORMATION_MESSAGE);  
if (opt == JOptionPane.YES_OPTION)  
    info.setText("您选择了 是");  
else if (opt == JOptionPane.NO_OPTION)  
    info.setText("您选择了 否");  
else if (opt == JOptionPane.CANCEL_OPTION)  
    info.setText("您选择了 取消");
```



## 6.5 对话框

- 常用对话框
  - 选项面板类 **JOptionPane**
    - **输入**对话框



```
String str;
```

```
str = JOptionPane.showInputDialog(w, "请输入姓名: ");
```

```
info.setText("您输入的是 " +str);
```

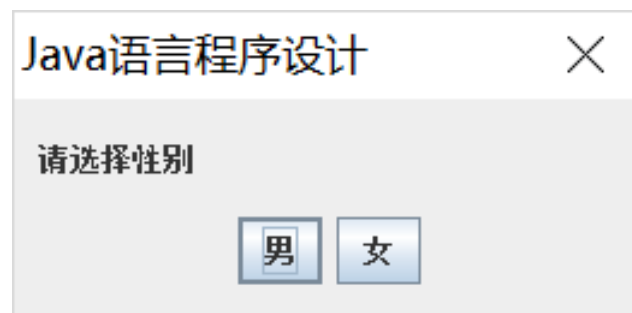


中國農業大學

閻道宏

## 6.5 对话框

- 常用对话框
  - 选项面板类 **JOptionPane**
    - 选项对话框



```
String gender[] = { "男", "女" };  
int opt = JOptionPane.showOptionDialog(  
    w, "请选择性别", "Java语言程序设计",  
    JOptionPane.DEFAULT_OPTION, JOptionPane.PLAIN_MESSAGE,  
    null, gender, "男");  
info.setText("您选择的是 " + gender[opt]);
```





# 6.5 对话框

- 文件选择类**JFileChooser**

javax.swing.JFileChooser类说明文档			
public class <b>JFileChooser</b> extends <b>JComponent</b> implements <b>Accessible</b>			
	修饰符	类成员（节选）	功能说明
1		<b>JFileChooser()</b>	构造方法
2		<b>JFileChooser</b> (String currentDirectoryPath)	构造方法
3		void <b>setCurrentDirectory</b> (File dir)	设置当前目录
4		void <b>setFileFilter</b> (FileFilter filter)	设置文件过滤器，例如只显示.jpg 图像文件
5		void <b>setMultiSelectionEnabled</b> (boolean b)	设置是否可以多选
6		int <b>showOpenDialog</b> (Component parent)	弹出一个打开文件对话框
7		int <b>showSaveDialog</b> (Component parent)	弹出一个保存文件对话框
8		int <b>showDialog</b> (Component parent, String approveButtonText)	弹出一个文件选择对话框
9		File <b>getSelectedFile</b> ()	获取所选择的文件
10		File[] <b>getSelectedFiles</b> ()	获取所选择的文件列表
11		String <b>getName</b> (File f)	获取文件f的文件名

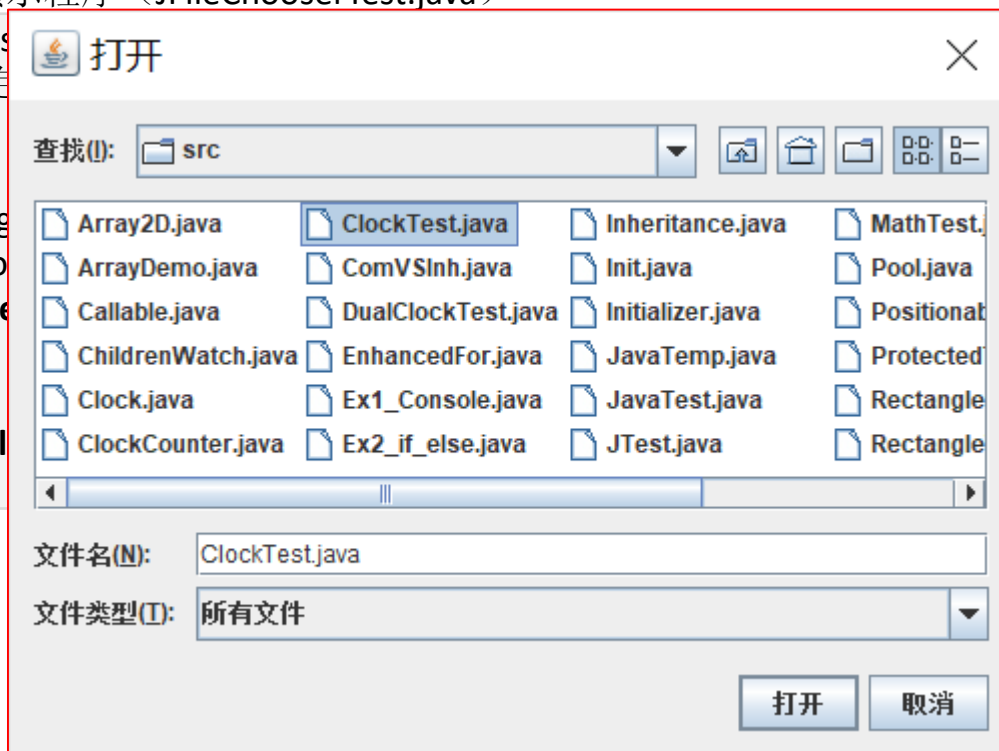


## 6.5 对话框

- 文件选择类 **JFileChooser**

例6-19 一个文件选择类JFileChooser的Java演示程序（JFileChooserTest.java）

```
1 import javax.swing.*; // 导入javax.s
2 import java.io.File; // 导入java.io包
3
4 public class JFileChooserTest {
5     public static void main(String[] arg
6         JFileChooser fc = new JFileChoo
7         fc.setCurrentDirectory( new File
8         fc.showOpenDialog(null);
9         File f = fc.getSelectedFile();
10        JOptionPane.showMessageDialog
11    } }
```



# 6.5 对话框

- 颜色选择类**JColorChooser**

javax.swing.JColorChooser类说明文档			
public class JColorChooser extends JComponent implements Accessible			
	修饰符	类成员（节选）	功能说明
1		JColorChooser()	构造方法
2		JColorChooser(Color initialColor)	构造方法
3		void setColor(Color color)	设置当前颜色
4		Color getColor()	获取当前颜色
5	static	Color showDialog(Component component, String title, Color initialColor)	显示对话框，选择颜色
.....			



# 6.5 对话框

- 颜色选择类 **JColorChooser**

例6-20 一个颜色选择类JColorChooser的Java演示程序（JColorChooserTest.java）

```
1 import javax.swing.*; // 导入
2 import java.awt.Color; // 导入
3
4 public class JColorChooserTest
5 {
6     public static void main(String[] args)
7     {
8         JColorChooser cc = new JColorChooser();
9         Color c = cc.showDialog(new JFrame(), "请选择颜色");
10        JOptionPane.showMessageDialog(new JFrame(), c);
11    }
12 }
```



中國農業大學

閻道宏

## 6.6 鼠标事件和键盘事件

- 图形用户界面中的所有图形组件都有
  - 鼠标事件 (**MouseEvent**)
  - 键盘事件 (**KeyEvent**)
  - 这是两个底层事件 (low-level event)
- 为了方便程序员，Java API将底层事件提炼成适合编程使用的**高层语义事件** (semantic event)
  - **ActionEvent**、**ItemEvent**、**ListSelectionEvent**等
  - 通常，程序员不需要处理鼠标或键盘事件
- 程序员也可以为图形组件添加**监听器**来处理底层的鼠标和键盘事件，其代表性应用场合是**计算机绘图程序**



# 6.6 鼠标事件和键盘事件

- 响应鼠标和键盘事件
  - Java API为**鼠标**事件设计了两个**监听器接口**
    - 鼠标监听器接口**MouseListener**，可以处理鼠标进出组件或鼠标按键操作事件  
mouseClicked()、mouseEntered()、mouseExited()、mousePressed()、mouseReleased()
    - 鼠标**移动**监听器接口**MouseMotionListener**，可以处理鼠标移动或拖动（按下按键的同时移动鼠标）事件  
mouseMoved()、mouseDragged()
  - Java API为**键盘**事件设计了键盘监听器接口**KeyListener**，可以处理键盘按键操作事件  
keyPressed()、keyReleased()、keyTyped()
  - 如果需要处理图形组件的鼠标和键盘事件，程序员需为组件添加相应的事件监听器



例 6-21 一个响应底层鼠标和键盘事件的 Java 演示程序 (JMouseEventTest.java)

```
1  import java.awt.*;           // 导入 java.awt 包中的类
2  import java.awt.event.*;     // 导入 java.awt.event 包中定义的事件类
3  import javax.swing.*;       // 导入 javax.swing 包中的类
4
5  public class JMouseEventTest { // 测试类
6      public void main(String[] args) {
7          // 为内容面板容器添加鼠标移动事件监听器
8          cp.addMouseMotionListener( new MouseMotionListener() {
9              public void mouseMoved(MouseEvent e) {
10                 String str = String.format("鼠标在移动: %d, %d", e.getX(), e.getY());
11                 info.setText(str);           // 显示当前鼠标的位置坐标
12             }
13             public void mouseDragged(MouseEvent e) { // 鼠标被拖动
14                 String str = String.format("鼠标在拖动: %d, %d", e.getX(), e.getY());
15                 info.setText(str);           // 显示当前鼠标的位置坐标
16             }
17         } });
18         // 为内容面板容器添加键盘事件监听器
19         cp.requestFocus(); // 获得键盘输入焦点
20         cp.addKeyListener( new KeyListener() { // 匿名类
21             public void keyTyped(KeyEvent e) { // 敲击了某个键盘按键
22                 String str = String.format("敲击了按键: %c", e.getKeyChar());
23                 info.setText(str);           // 显示被敲击的按键
24             }
25             public void keyPressed(KeyEvent e) {} // 某个按键被按下, 未响应
26             public void keyReleased(KeyEvent e) {} // 某个按键被松开, 未响应
27         } });
28     }
29 }
```



图..



鼠标在移动: 194, 118

## 6.6 鼠标事件和键盘事件

- 在画布上绘图
  - 画布类**Canvas**
    - 是一个图形组件，描述了一个可以绘图的**矩形区域**
  - 使用画布类Canvas编写绘图程序的基本步骤
    - **继承**并扩展画布类Canvas，**重写**其中的绘图方法**paint()**
    - 使用扩展后的画布类**创建**画布对象
    - 为画布对象**添加**响应鼠标和键盘事件的监听器，**记录**用户在画布上的操作
    - 每次记录完用户操作之后即**调用**画布类的重画方法**repaint()**，对画布进行刷新重绘



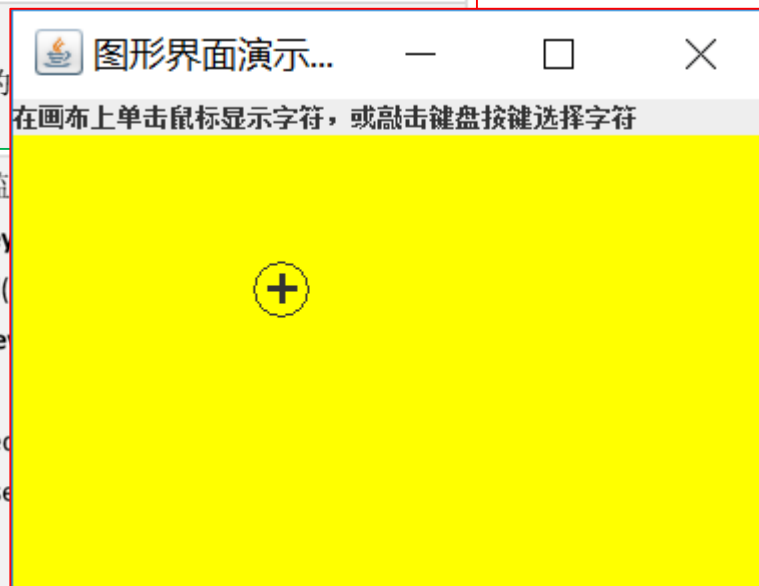


例 6-22 一个使用画布类 Canvas 编写的绘图演示程序 (JCanvasTest.java)

```

1  import java.awt.*;           // 导入 java.awt 包中的类
2  import java.awt.event.*;     // 导入 java.awt.event 包中定义的
3  import javax.swing.*;        // 导入 javax.swing 包中的类
4
5  public class JCanvasTest {   // 为画布添加键盘事件监听器
6      public void addKeyListener( new KeyListener() {
7          public void keyTyped(KeyEvent e) {
8              cv.key = e.getKeyChar();
9          }
10         public void keyPressed(KeyEvent e) {}
11         public void keyReleased(KeyEvent e) {}
12     });
13
14     class MyCanvas extends Canvas { // 定义自己的画布类，继承 Canvas 类
15         public int x = -1, y = -1; // 添加记录鼠标位置的字段
16         public char key = '+';     // 添加记录键盘按键的字段，初始值为“+”
17         private Font ef = new Font("TimesRoman", Font.PLAIN, 32); // 字体
18         public MyCanvas() {        // 构造方法
19             setBackground(Color.YELLOW); // 设置画布背景色
20         }
21         public void paint(Graphics g) { // 重写 paint 方法
22             if (x == -1 || y == -1) return; // 未点击过鼠标，直接返回
23             g.drawOval(x-5, y-25, 27, 27); // 在点击鼠标的位置画一个圆
24             g.setFont( ef );              // 设置字体
25             g.drawString( String.valueOf(key), x, y ); // 在圆中显示一个字符
26         }
27     }
28 }

```



# 6.7 Java小应用程序Applet

- 小应用程序类**Applet**

例6-23 一个简单的Java小应用程序示例代码（AppletDemo.java）

```
1 import java.applet.*; // 导入java.applet包中与小应用程序相关的类和接口
2 import java.awt.*;    // 导入java.awt包中的类
3
4 public class AppletDemo extends Applet { // 定义一个小应用程序类
5     String msg;        // 显示用的文字信息
6     Font f;            // 显示用的字体
7     Color c;           // 绘图用的颜色
8     public void init() { // 重写超类Applet的初始化方法
9         msg = "Hello, World"; // 将在组件上显示该字符串
10        f = new Font("TimesRoman", Font.PLAIN, 16); // 字体
11        c = Color.RED;      // 颜色
12    }
13    public void paint(Graphics g) { // 重写绘制方法：显示文字、绘制图形
14        super.paint(g);        // 调用超类的paint()方法
15        g.setFont(f);          // 设置字体
16        g.drawString(msg, 20, 40); // 显示文字信息 “Hello, World”
17        g.setColor(c);         // 设置绘图颜色
18        g.fillRect(20, 60, 100, 100); // 填充一个实心的正方形
19    }
20 }
```



中國農業大學

閻道宏

## 6.7 Java小应用程序Applet

- 小应用程序类**Applet**

- 小应用程序查看器

\JDK安装目录\bin\ **appletviewer**.exe



**appletviewer** d:\javatest\ **AppletDemo.class** <回车键>

- 在Eclipse中运行（**Run**）Java小应用程序



中國農業大學

閻道宏

# 6.7 Java小应用程序Applet

- 小应用程序类**Applet**

- 小应用程序查看器

- 在**浏览器**中运行Java小应用程序

- Java语言提供小应用程序的目的是为了在浏览器（browser）中运行，用于增强HTML网页的表现力

- 在HTML网页中嵌入Java小应用程序

- `<applet code=小应用程序文件名 height=高度 width=宽度>`  
`</applet>`

- `<html>`

- `<applet code=AppletDemo.class height=300 width=200>`

- `</applet>`

- `</html>`



中國農業大學

閻道宏

# 6.7 Java小应用程序Applet

- 小应用程序类**Applet**

java.applet. <b>Applet</b> 类说明文档			
public class <b>Applet</b> extends <b>Panel</b>			
	修饰符	类成员（节选）	功能说明
1		<b>Applet()</b>	构造方法
2		void <b>init()</b>	浏览器加载 <b>Applet</b> 后自动调用该方法，用于定义 <b>Applet</b> 的初始化代码
3		void <b>start()</b>	浏览器在调用完 <b>init()</b> 方法后自动调用该方法，用于定义 <b>Applet</b> 的功能代码
4		void <b>stop()</b>	在退出 <b>Applet</b> 所在页面时浏览器自动调用该方法，用于定义 <b>Applet</b> 的暂停代码
5		void <b>destroy()</b>	浏览器在删除 <b>Applet</b> 时自动调用该方法，用于定义 <b>Applet</b> 的善后处理代码
6		Image <b>getImage</b> (URL url)	加载url指定的图像文件，用于后续显示
7		AudioClip <b>getAudioClip</b> (URL url)	加载url指定的音频文件，用于后续播放
8		void <b>play</b> (URL url)	直接播放url指定的音频文件
.....			



# 6.7 Java小应用程序Applet

- 小应用程序类**Applet**
  - **getImage()**

java.awt. <b>Image</b> 类说明文档			
public abstract class <b>Image</b> extends Object			
	修饰符	类成员（节选）	功能说明
1		<b>Image()</b>	构造方法
2	abstract	int <b>getWidth</b> (ImageObserver observer)	获取图像宽度
3	abstract	int <b>getHeight</b> (ImageObserver observer)	获取图像高度
.....			

- **getAudioClip()**

java.applet. <b>AudioClip</b> 接口说明文档			
public interface <b>AudioClip</b>			
	修饰符	接口成员（节选）	功能说明
1		void <b>play</b> ()	播放音频
2		void <b>stop</b> ()	停止播放
3		void <b>loop</b> ()	循环播放
.....			



# 6.7 Java小应用程序Applet

- 小应用程序类Applet
  - swing框架：新的小应用程序类**JApplet**
    - JApplet是Applet的子类，两者的用法基本相同
    - JApplet与Applet的最大区别是，使用Applet编写小应用程序时只能使用老的AWT图形组件，而新的JApplet则可以使用新的swing图形组件
    - JApplet与JFrame、JDialog一样，它们是swing框架中的**三大顶层容器**
  - Java小应用程序是一种早期用于**增强**HTML网页表现力的技术
    - 现在，**HTML语言**本身也在不断发展，其表现能力不断提高，例如**HTML 5**
    - **JavaScript**脚本语言也可以在很大程度上替代Java小应用程序
    - 目前，Java小应用程序**已经**很少使用了，读者只需简单了解即可
- **注：**全国计算机等级考试二级Java语言程序设计考试大纲（**2018年版**）中还包含Java小应用程序（Applet）这个知识点。



# 第6章 图形用户界面程序

- 本章学习要点

- 了解**Java API**中各图形组件之间的关系

- 框架窗口**JFrame**和对话框窗口**JDialog**是顶级容器，其中包含内容面板
    - 可以在**内容面板**中添加**组件**，并可设置不同的**布局管理**策略
    - 内容面板中可使用**JPanel**划分出子面板，子面板**独立布局**，可实现比较复杂的图形界面

- 了解Java图形用户界面程序的**事件响应机制**

- 通过**编程练习**掌握常用组件的用法，并能根据程序功能要求设计图形用户界面

- 在掌握上述图形用户界面基本编程原理之后，可通过**Java API文档**自行研究javax.swing包中其他各种不同功能的图形组件。例如，JSplitPane、JTabbedPane、JEditorPane、JPasswordField、JPopupMenu、JToolBar、JToolTip、JProgressBar、JScrollBar、JSlider、JSpinner、JTree等

