



# 第九章 图形用户界面编程

孙国道, 博士, 副教授, 博士生导师

13757125851(685851) 计C516

Homepage: <http://godoorsun.org>

E-Mail: [godoor.sun@gmail.com](mailto:godoor.sun@gmail.com)  
[guodao@zjut.edu.cn](mailto:guodao@zjut.edu.cn)



计 算 机 科 学 与 技 术 学 院  
浙 江 工 业 大 学

<https://stackoverflow.com/questions/2598006/will-vc-mfc-become-obsolete-in-near-future>

<https://stackoverflow.com/questions/8614995/is-mfc-deprecated>

<https://www.zhihu.com/question/60773898>

<https://www.zhihu.com/question/19938661>

<https://www.zhihu.com/question/43824189>

<https://www.zhihu.com/question/60773898>

<https://www.quora.com/Which-is-better-MFC-or-NET>

<https://www.quora.com/Should-I-use-MFC-or-QT>



**刘天一**

仪器工程师。小于10人关注的问题不回答。

4 人赞同了该回答

纯软件的产品，用MFC无疑是过时的。

但是对于需要与硬件结合的软件，MFC并不过时。很多工控产品，很多板卡、工控设备的示例代码，往往都有MFC的示例。稍好一些的工控产品可能有C#、VB的示例，但也不是全有。而更先进一些的图形界面示例，就几乎没有了。

<https://www.zhihu.com/question/36625877>

<https://www.zhihu.com/question/37236236>

<https://stackoverflow.com/questions/889097/what-advantages-are-to-be-had-by-using-javaswing-over-cwinforms-wpf>

<https://www.quora.com/In-your-opinion-what-is-better-for-a-GUI-Java-or-c>

HTML/CSS/Javascript

# C/S v.s. B/S

**回顾关键词：**字节流(InputStream\OutputStream)、字符流(Reader\Writer)、File、对象序列化(java.io.Serializable)

**本章的目的：**图形用户界面(GUI)程序包括哪些元素？如何开发图形用户界面程序？如何让界面上的按钮、下拉框、树等响应鼠标、键盘等？

### 图形用户界面编程

9.1、AWT与Swing

9.2、容器：JFrame\JPanel\JScrollPane\JSplitPane

9.3、菜单和工具条

9.4、基本组件：JLabel\JButton\JComboBox\JTree等

9.5、组件常用方法

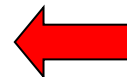
9.6、布局管理器

9.7、事件处理模型

9.8、鼠标事件处理

9.9、事件适配器类

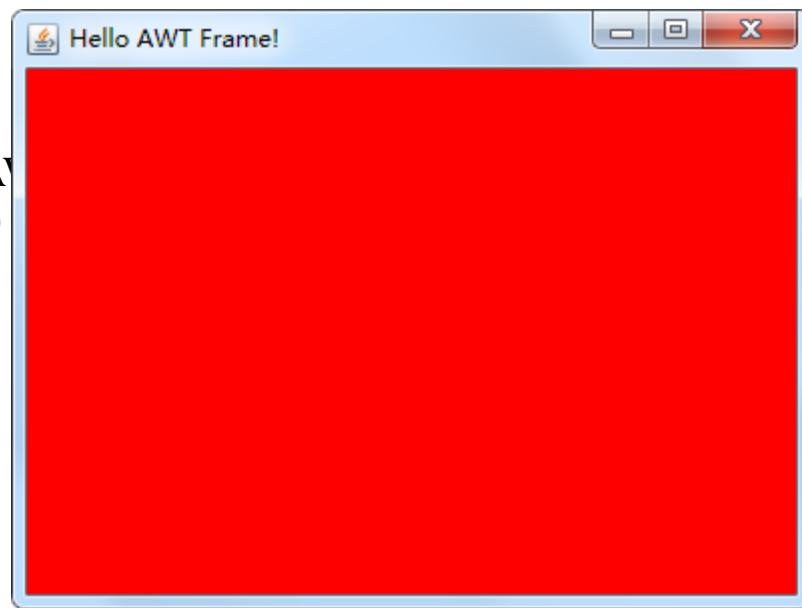
9.10、键盘事件处理



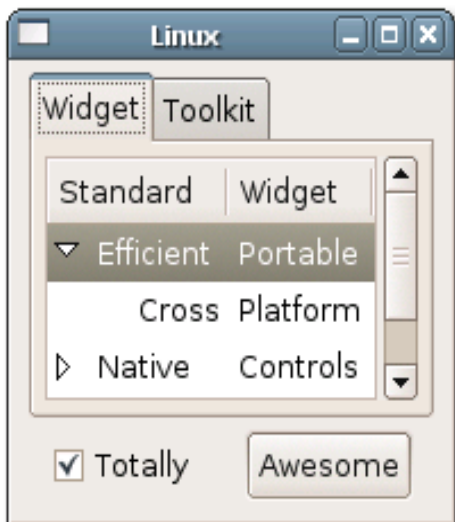


- AWT(Abstract Windowing Toolkit)是Java早期(JDK1.0)的GUI标准API
- 重量级组件
- 在90年代，程序员中流传着一个笑话: Java的真正信条是"一次编写，到处测试(Write Once, Test Everywhere)"。导致这种糟糕局面的一个可能原因据说是AWT从概念产生到完成实现只用了一个月。

```
import java.awt.Frame;  
import java.awt.Color;  
public class TestAWT extends Frame{  
    public TestAWT (String str){  
        super(str);  
    }  
    public static void main(String args[ ]){  
        TestAWT fr = new TestAWT ("Hello AWT");  
        //设置Frame的大小，缺省为 (0, 0)  
        fr.setSize(400,300);  
        //设置Frame的背景为红色  
        fr.setBackground(Color.red);  
        //设置Frame为可见，缺省为不可见  
        fr.setVisible(true);  
    }  
}
```



- Java 1.2开始，AWT被Swing替代
- Swing包含的组件的平台相关性较小，所以称为轻量级组件
- 所在的包：javax.swing
- 由100%纯java实现的



(a) Linux下

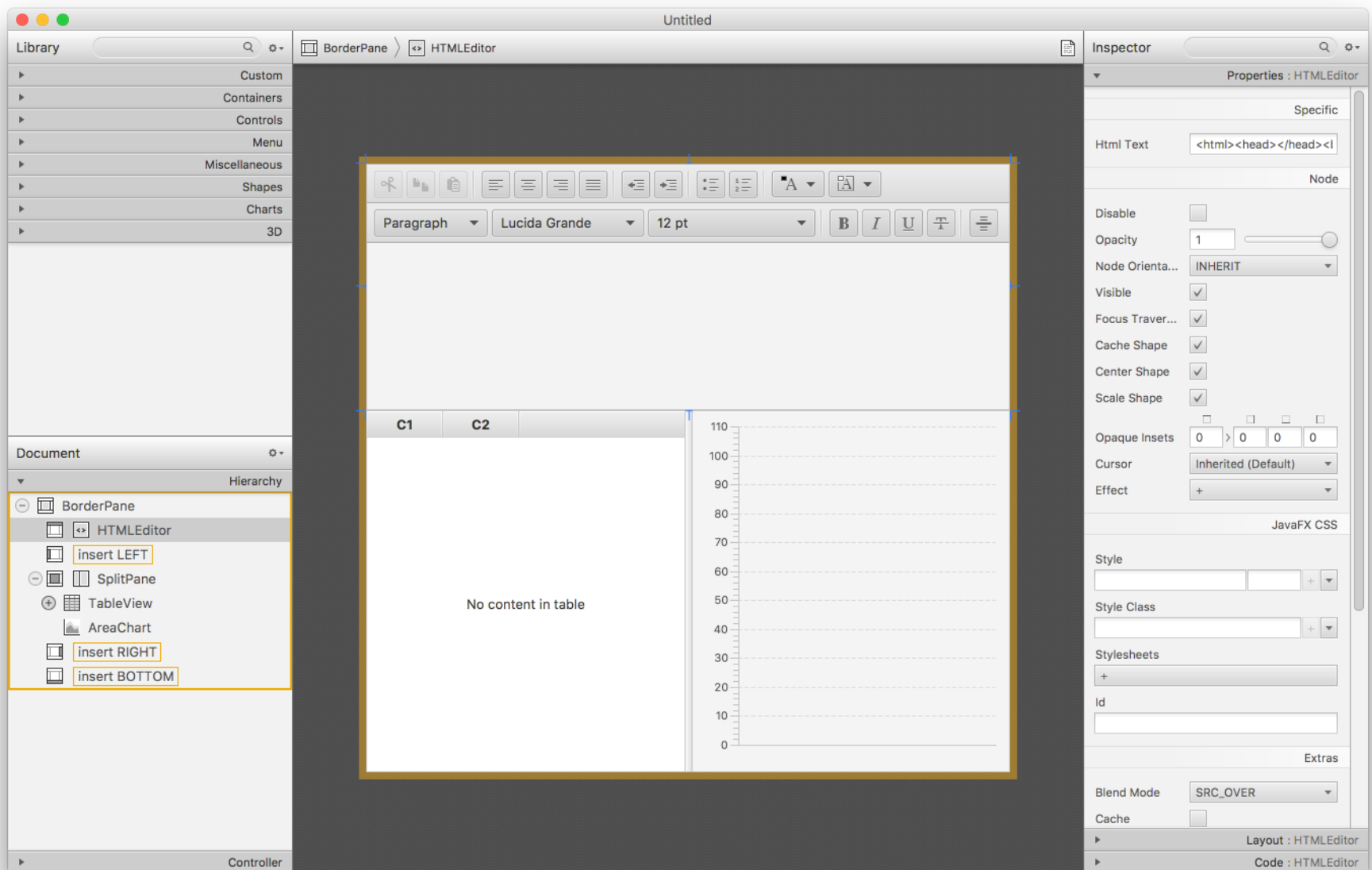


(b) Mac OS下



(c) Windows下

同样的java代码在不同操作系统下的表现



### Swing组件

#### 容器

顶层容器：JFrame, JApplet, JDialog和JWindow

其他容器：JPanel, JScrollPane, JSplitPane, JToolBar

#### 组件

基本控制组件：JButton, JComboBox, JList, JMenu, JSlider, JTextField等

不可编辑的信息显示组件：JLabel, JProgressBar, JToolTip等

可编辑的信息显示组件：JColorChooser, JFileChooser, JTable, JTextArea等

## 9.1.2、AWT和Swing

java.lang.Object

└─ java.awt.Component

└─ java.awt.Container

└─ java.awt.Window

└─ java.awt.Frame

└─ javax.swing.JFrame

└─ java.awt.Dialog

└─ javax.swing.JDialog

└─ javax.swing.JWindow

└─ java.awt.Panel

└─ java.applet.Applet

└─ javax.swing.JApplet

└─ javax.swing.JComponent

└─ javax.swing.JPanel

└─ javax.swing.JScrollPane

└─ javax.swing.JSplitPane

└─ javax.swing.JToolBar

└─ javax.swing.AbstractButton

└─ javax.swing.JButton

└─ javax.swing.JToggleButton

└─ javax.swing.JCheckBox

└─ javax.swing.JRadioButton

└─ javax.swing.JMenuItem

└─ javax.swing.JMenu

└─ javax.swing.JComboBox

└─ javax.swing.JList

└─ javax.swing.JSlider

└─ javax.swing.text.JTextComponent

└─ javax.swing.JTextField

└─ javax.swing.JTextArea

└─ javax.swing.JLabel

└─ javax.swing.JProgressBar

└─ javax.swing.JToolTip

└─ javax.swing.JColorChooser

└─ javax.swing.JFileChooser

└─ javax.swing.JTable

### 图形用户界面编程

9.1、AWT与Swing

9.2、容器：JFrame\JPanel\JScrollPane\JSplitPane

9.3、菜单和工具条

9.4、基本组件：JLabel\JButton\JComboBox\JTree等

9.5、组件常用方法

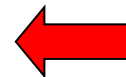
9.6、布局管理器

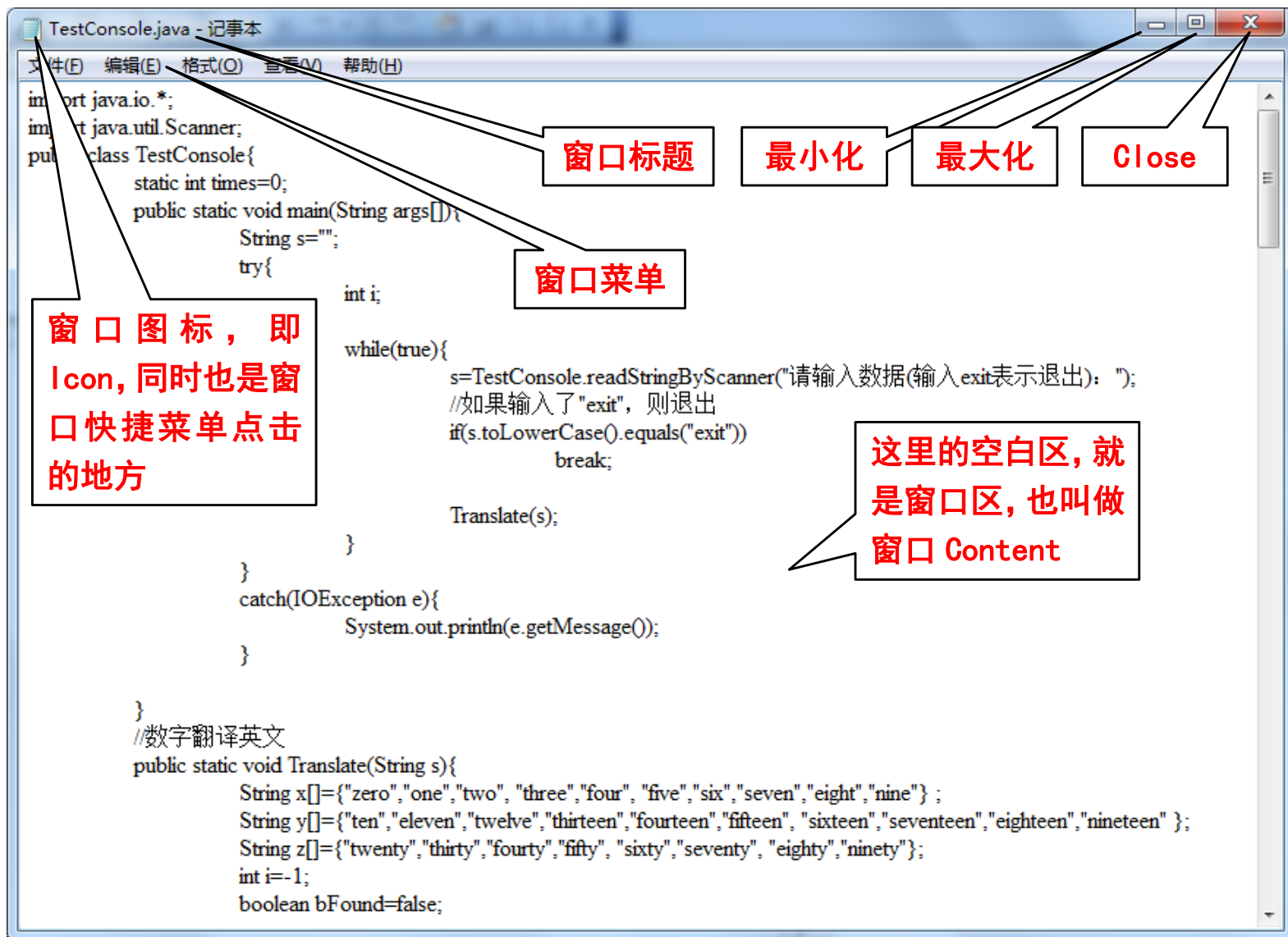
9.7、事件处理模型

9.8、鼠标事件处理

9.9、事件适配器类

9.10、键盘事件处理





/\*我的第一个java窗口程序，采用直接在main中创建窗口的方法\*/

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

```
import javax.swing.*;
```

```
public class TestJFrameDirect{
```

```
    /*******
```

```
    //以下为成员变量（对象）的定义
```

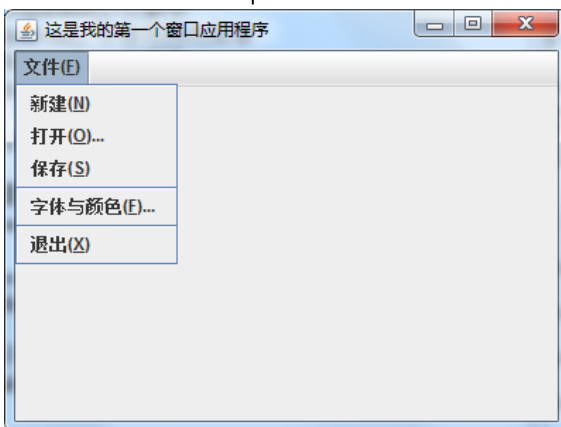
```
    //定义菜单
```

```
    static JMenuBar mb=new JMenuBar();//菜单栏
```

```
    static FgMenu mFile=new FgMenu("文件(F)",KeyEvent.VK_F);//"文件"菜单
```

```
    static JMenuItem miNew=new JMenuItem("新建(N)",KeyEvent.VK_N),  
        miOpen=new JMenuItem("打开(O)...",KeyEvent.VK_O),  
        miSave=new JMenuItem("保存(S)",KeyEvent.VK_S),  
        miFont=new JMenuItem("字体与颜色(F)...",KeyEvent.VK_F),  
        miQuit=new JMenuItem("退出(X)",KeyEvent.VK_X);
```

```
    /*******
```



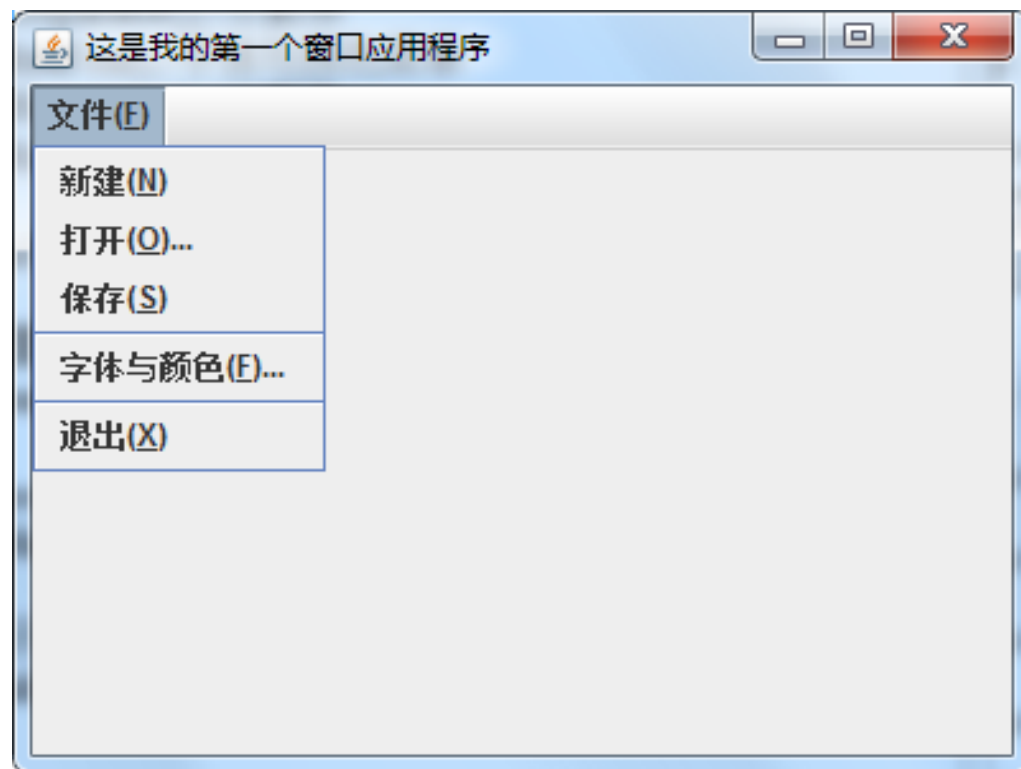
TestJFrameDirect.java



```
import ...
public class TestJFrameDirect{
    JMenuBar mb...//定义变量(菜单)部分
    public static void main(String args[]){
        //①：创建窗口对象，窗口标题通过构造方法传递进去
        JFrame frm=new JFrame("这是我的第一个窗口应用程序");
        //②：添加组件。本例中直接添加菜单
        frm.setJMenuBar(mb);
        mFile.add(miNew);//新建
        mFile.add(miOpen);//打开
        mFile.add(miSave);//保存
        mFile.addSeparator();//分割条
        mFile.add(miFont);//字体与颜色菜单
        mFile.addSeparator();//分割条
        mFile.add(miQuit);//退出
        mb.add(mFile); //将"文件"菜单添加到菜单栏上
        //③：设置窗口位置和大小
        frm.setBounds(10, 10, 400, 300);
        //设置close按钮的操作
        frm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        //④：显示窗口
        frm.setVisible(true);
    }
}
```

//自定义菜单

```
class FgMenu extends JMenu{  
    public FgMenu(String label){  
        super(label);  
    }  
    public FgMenu(String label,int nAccelerator){  
        super(label);  
        setMnemonic(nAccelerator);  
    }  
}
```



```
import java.awt.Color;
import java.awt.Container;
import javax.swing.*;
public class TestJPanel extends JFrame{
    public TestJPanel(String sTitle) {
        super(sTitle);
        setSize(400,300);//设置大小
        //获取窗口面板;
        Container c=getContentPane();
        c.setBackground(Color.RED); //窗口背景红色
        c.setLayout(null);//取消布局器
        JPanel pan=new JPanel();
        pan.setBackground(Color.YELLOW); // pan背景黄色
        pan.setSize(200,100);
        add(pan);//用add方法把面板pan添加到窗口中
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
    public static void main(String args[]) {
        TestJPanel frm = new TestJPanel("JFrame with JPanel");
        frm.setVisible(true);
    }
}
```

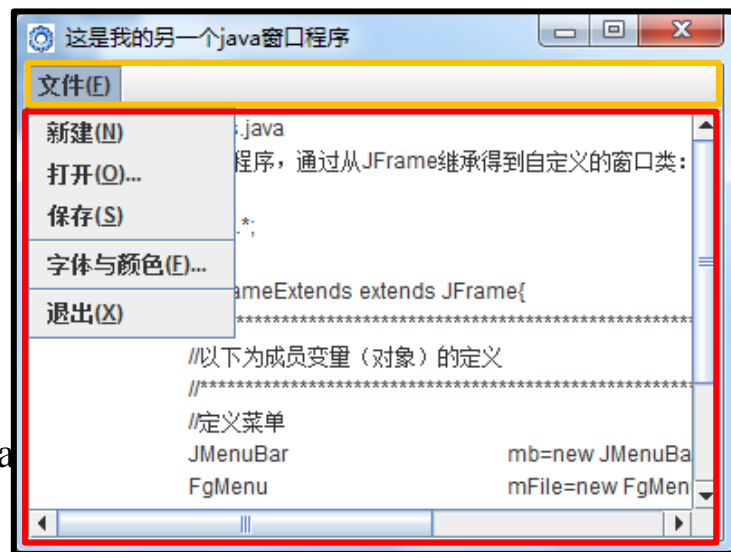


JPanel是一种轻量级的中间容器，称为**面板组件**，可以在它上面添加其他组件（包括其他面板组件）。面板(JPanel)的**大小**随着其包含的组件多少而变大变小。

TestJPanel.java

import ...

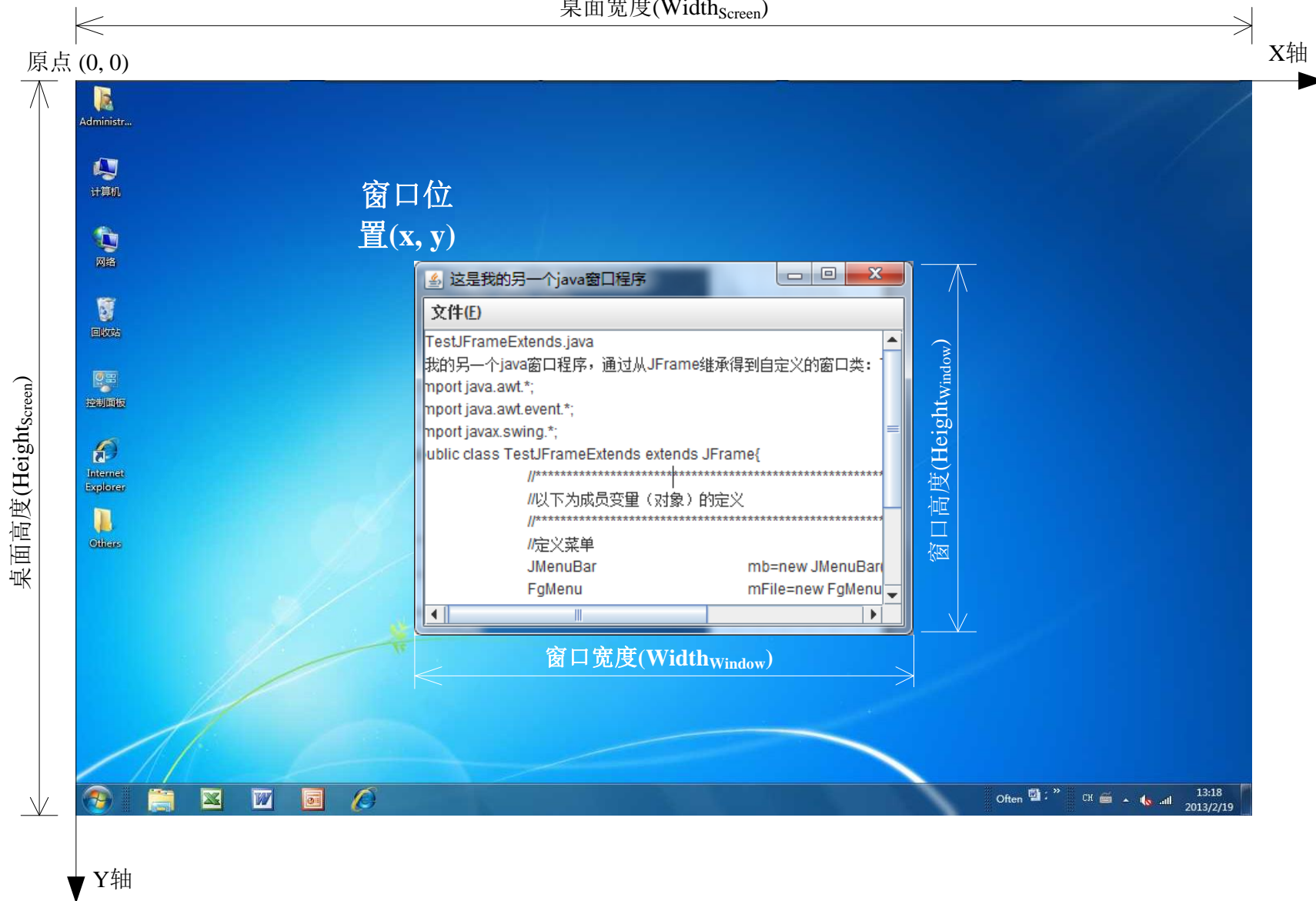
```
public class TestJFrameExtends extends JFrame{
    //以下为成员变量（对象）的定义
    //此处定义菜单(与TestJFrameDirect.java中一样)
    JTextArea      ta=new JTextArea();//文本框
    TestJFrameExtends(String sTitle){
        super(sTitle);
        //②：添加组件。本例直接添加菜单与JTextArea
        addMenus();
        //添加带滚动条(JScrollPane)的文本编辑框JTextArea
        JScrollPane sp=new JScrollPane(ta);
        add(sp);
        //③：设置窗口大小
        setSize(400, 300);
        //设置close按钮的操作
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        //使窗口在显示屏居中显示
        centerWindow();
        //改变窗口图标
        Toolkit tk=getToolkit(); //得到一个Toolkit对象
        Image icon=tk.getImage("online.gif"); //获取图标
        setIconImage(icon);
    }
}
```



```
//添加菜单
private void addMenus(){
    setJMenuBar(mb);
    mFile.add(miNew);//新建
    //.....
}
//窗口居中
public void centerWindow(){
    //获得显示屏桌面窗口的大小
    Toolkit tk=getToolkit();
    Dimension dm=tk.getScreenSize();
    //让窗口居中显示
    setLocation((int)(dm.getWidth()-getWidth())/2,(int)(dm.getHeight()-getHeight())/2);
}
public static void main(String args[]){
    //①： 创建窗口对象
    TestJFrameExtends frm=new TestJFrameExtends ("这是我的另一个java窗口程序");
    //④： 显示窗口
    frm.setVisible(true);
}
}
//这里省略了FgMenu类的定义
```

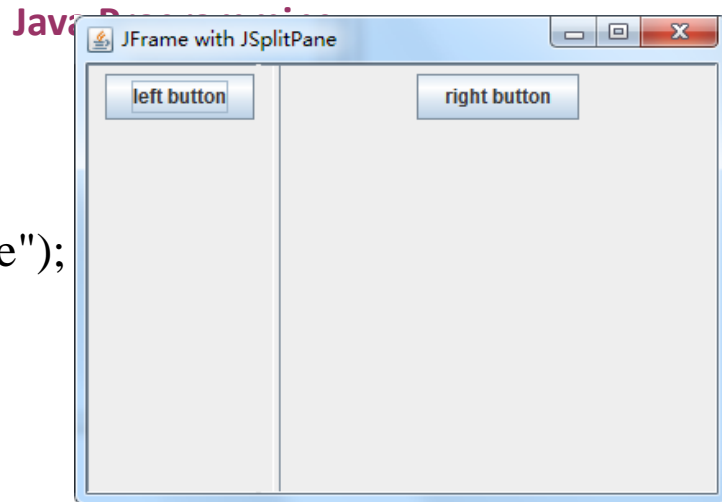
## 9.2.3、JScrollPane

桌面宽度(Width<sub>Screen</sub>)



## 9.2.4、JSplitPane

```
import ...
public class TestJSplitPane{
    public static void main(String args[]){
        JFrame fr = new JFrame("JFrame with JSplitPane");
        Container c = fr.getContentPane();
        JPanel leftPane = new JPanel();//左面板
        JPanel rightPane = new JPanel();//右面板
        //创建水平分割条，即分为左右两部分
        JSplitPane sp = new JSplitPane(JSplitPane.HORIZONTAL_SPLIT,
            leftPane,rightPane);
        sp.setDividerSize(5); //设置分割条本身的宽度为5个像素
        leftPane.add(new JButton("left button"));    //将按钮添加到左边的面板
        rightPane.add(new JButton("right button")); //将按钮添加到右边的面板
        c.add(sp);//将分割条(含左右两个带按钮的面板)添加到窗口上
        fr.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        fr.setSize(400,300);
        fr.setVisible(true);
        //以下语句须放在setVisible之后，否则不会起到效果
        sp.setDividerLocation(0.3); //左边占0.3(30%)，右边占0.7(70%)
    }
}
```



### 图形用户界面编程

9.1、AWT与Swing

9.2、容器：JFrame\JPanel\JScrollPane\JSplitPane

9.3、菜单和工具条

9.4、基本组件：JLabel\JButton\JComboBox\JTree等

9.5、组件常用方法

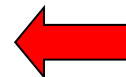
9.6、布局管理器

9.7、事件处理模型

9.8、鼠标事件处理

9.9、事件适配器类

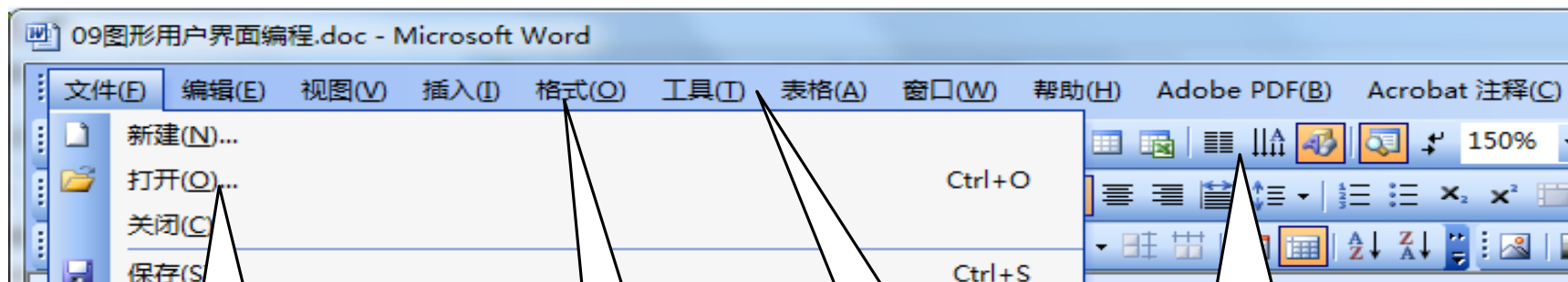
9.10、键盘事件处理





## 9.3.1、菜单组件：JMenuBar、JMenu、JMenuItem

Java Programming



菜单条  
**JMenuItem**

菜单  
**JMenu**

菜单栏  
**JMenuBar**

工具栏(条)  
**JToolBar**

### ❖ 菜单中的加速键

- 菜单(JMenu)或菜单项(JMenuItem)中所显示的带下划线的字母即为对应的加速键字母。如前图所示，如果要通过加速键访问“保存(S)...”菜单，首先按下ALT+F（文件菜单的加速键，JMenu），然后再按下S字母键即可(保存菜单项的加速键，JMenuItem)。
- 父类javax.swing.AbstractButton的如下方法进行设置：  
    public void **setMnemonic**(int mnemonic)  
    其中 mnemonic为java.awt.event.KeyEvent.VK\_XXX，  
    XXX对应于相应的字母，如A、B、...等。
- 源文件加入：import java.awt.event.\*;

例如：

```
Import java.awt.event.*;
...
//定义菜单变量(对象)
JMenu      miFile=new JMenu("文件(F) ");
JMenuItem  miNew=new JMenuItem("新建(N) ", KeyEvent.VK_N),
miOpen=new JMenuItem("打开(O)... ");

...
miFile.setMnemonic(KeyEvent.VK_F);
miFile.add(miNew);
miFile.add(miOpen);
miOpen.setMnemonic(KeyEvent.VK_O);
```

### ❖ 菜单中的快捷键和图标

- 带加速键的菜单需要两步才能操作，快速键一次完成，如按下CTRL+S键即可触发"保存"菜单的功能
- 设置快捷键的方法定义在JMenuItem中，如下：

```
public void setAccelerator(KeyStroke keyStroke)
```

其中keyStroke则通常采用KeyStroke的如下静态方法获得。

```
public static KeyStroke getKeyStroke(int keyCode, int modifiers)
```

其中，keyCode是定义在java.awt.event.KeyEvent中的虚拟键常量：

```
java.awt.event.KeyEvent.VK_ENTER
```

```
java.awt.event.KeyEvent.VK_N
```

```
...
```

而modifiers则是定义在java.awt.event.InputEvent中的修饰符常量：

```
java.awt.event.InputEvent.SHIFT_DOWN_MASK
```

```
java.awt.event.InputEvent.CTRL_DOWN_MASK
```

```
...
```

- 源文件加入：import java.awt.event.\*;

例如：

```
miNew.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_N,  
                                           InputEvent.CTRL_DOWN_MASK));  
miOpen.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_O,  
                                           InputEvent.CTRL_DOWN_MASK| InputEvent.SHIFT_DOWN_MASK));
```

```
ImageIcon icon=new ImageIcon("online.gif");  
miOpen.setIcon(icon);
```



暴走的快捷键。。

(1)在JMenuBar对象创建的下一行创建工具栏对象：JToolBar   mtb=new JToolBar();

(2)自定义类FgButton如下：

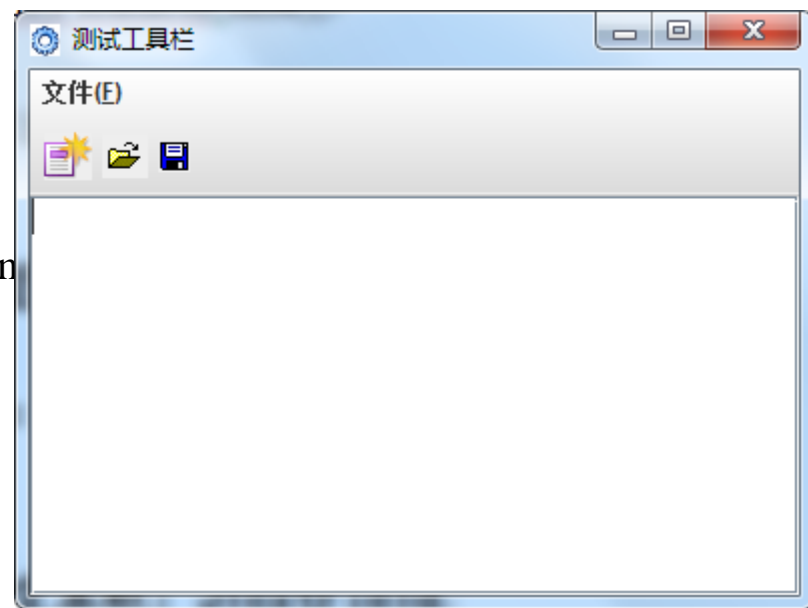
```
class FgButton extends JButton{
    public FgButton(){
        super();
    }
    public FgButton(Icon icon){
        super(icon);
    }
    public FgButton(Icon icon,String strToolTipText){
        super(icon);
        setToolTipText(strToolTipText);
    }
    public FgButton(String text){
        super(text);
    }
    public FgButton(String text, Icon icon, String strToolTipText){
        super(text, icon);
        setToolTipText(strToolTipText);
    }
}
```

FgButton.java

(3)添加addToolBar()方法:

```
private void addToolBar(){  
    //工具条  
    Container c=getContentPane();  
    c.add(BorderLayout.NORTH, mtb);  
  
    mtb.setLayout(new FlowLayout(FlowLayout.LEFT));  
    FgButton[] btn={new FgButton(new ImageIcon ("New.gif"),  
                                "新建文件"),  
                    new FgButton(new ImageIcon("open.gif"),  
                                "打开文件"),  
                    new FgButton(new ImageIcon("save.gif"),  
                                "保存文件")};  
  
    for(int i=0;i<btn.length;i++){  
        btn[i].setBorder(BorderFactory.createEtchedBorder());  
        mtb.add(btn[i]);  
    }  
    //设置不可浮动  
    mtb.setFloatable(false);  
}
```

(4)在构造方法中添加如下代码: addToolBar();





### 图形用户界面编程

9.1、AWT与Swing

9.2、容器：JFrame\JPanel\JScrollPane\JSplitPane

9.3、菜单和工具条

9.4、基本组件：JLabel\JButton\JComboBox\JTree等

9.5、组件常用方法

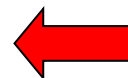
9.6、布局管理器

9.7、事件处理模型

9.8、鼠标事件处理

9.9、事件适配器类

9.10、键盘事件处理



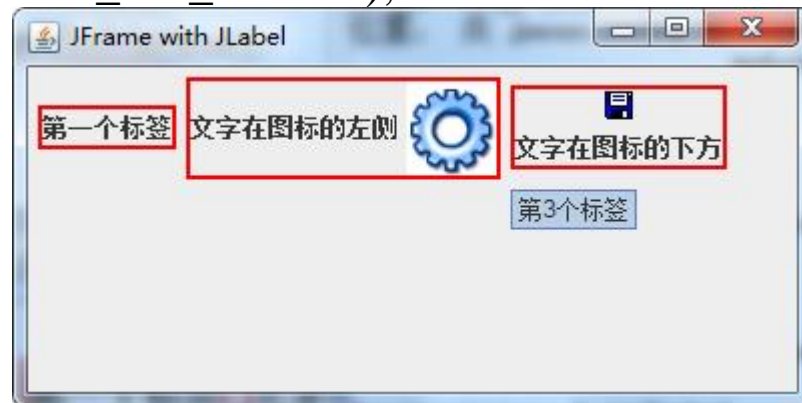
## TestJLabel.java

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

public class TestJLabel extends JFrame{
    public TestJLabel(){
        super( "JFrame with JLabel");
        //三个标签上的文字
        String [] s = {"第一个标签",
                        "文字在图标的左侧",
                        "文字在图标的下方"};
        ImageIcon[] ic = {null,
                           new ImageIcon("online.gif"),
                           new ImageIcon("save.gif" )};
        //三个标签在水平方向上的对齐方式
        int [] ih = {0, JLabel.LEFT, JLabel.CENTER};
        //三个标签在垂直方向上的对齐方式
        int [] iv = {0, JLabel.CENTER, JLabel.BOTTOM};
        Container c = getContentPane( );//取得窗口的内容面板
        c.setLayout( new FlowLayout(FlowLayout.LEFT) );//设置布局管理器
```



```
for (int i=0; i<3; i++){  
    //创建三个标签  
    JLabel myLabel = new JLabel( s[i] , ic[i], JLabel.LEFT);  
    if (i>0){  
        myLabel.setHorizontalTextPosition(ih[i]);  
        myLabel.setVerticalTextPosition(iv[i]);  
    }  
    //设置边框, setBorder来自JLabel的父类JComponent  
    myLabel.setBorder(BorderFactory.createLineBorder (Color.RED, 2));  
    myLabel.setToolTipText("第" + (i+1) + "个标签");  
    //加入到窗口的内容面板中  
    c.add(myLabel);  
}  
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
setSize(400, 300);  
}  
public static void main(String args[ ]){  
    TestJLabel frm = new TestJLabel();  
    frm.setVisible(true);  
}  
}
```



## 9.4.2、单行文本框：JTextField和JPasswordField

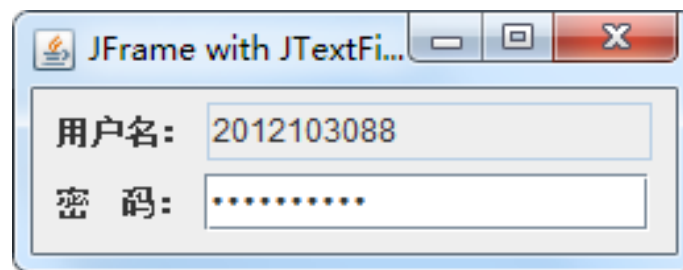
Java Programming

TestJTextField.java

```
import java.awt.*;
import javax.swing.*;
public class TestJTextField{
    public static void main(String args[ ]){
        JFrame frm= new JFrame( "JFrame with JTextField" );
        frm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frm.setSize(260, 100 );
        Container c = frm.getContentPane( );
        c.setLayout(new FlowLayout( ) );

        JTextField[] t={new JTextField("2012103088", 15),
                        new JPasswordField("1234567890", 15)};
        c.add(new JLabel("用户名: "));
        c.add(t[0]);
        c.add(new JLabel("密 码: "));
        c.add(t[1]);

        t[0].setEditable( false );//用户名设置为只读
        frm.setVisible( true );
    }
}
```



JTextArea类的常用public方法

方法定义	功能说明
<code>void setLineWrap(boolean wrap)</code>	设置是否自动换行，默认值为false
<code>void setWrapStyleWord(boolean word)</code>	设置换行方式，若为true，当超过文本框边界时，则以单词边界为换行界线；若为false，则将在字符边界处换行。默认为 false
<code>int getLineCount()</code>	返回文本区中所包含的行数，注意它是根据回车来判断行数的，实际上并不能反映界面上所看到的行数，因为这与文本框的大小有关。
<code>void insert(String str, int pos)</code>	将str插入指定位置pos的前面
<code>void append(String str)</code>	将str追加到文档结尾
<code>void replaceRange(String str, int start, int end)</code>	用str替换索引位置从start到end之间的文本
<code>void setFont(Font f)</code>	设置当前字体

# 多行文本框：JTextArea

JTextArea类的其他方法(直接继承自JTextComponent类)

方法定义	功能说明
void setSelectionColor(Color c)	设置选定文字的背景颜色为c
void setSelectedTextColor(Color c)	设置选定文字的颜色为c
void setDisabledTextColor(Color c)	设置当文本框被禁用时的文本颜色
void replaceSelection(String content)	用content替换选定的内容；如果没有选择的内容，则该操作插入给定的文本；如果没有替换文本，则该操作移除当前选择的内容
String getText(int offset, int len)	获取从offset开始的len个字符
String getText()	返回文本框中所有的文本
void cut()	将选定文字传输到系统剪贴板同时从文本框中将这些文字移除
void copy()	将选定文字传输到系统剪贴板
void paste()	将系统剪贴板的内容传输到文本框中，如果在文本框有选定的内容，则使用剪贴板的内容替换它
void setFocusAccelerator(char aKey)	设置将导致接收的文本组件获取焦点的加速键
void setText(String t)	将此文本设置为t
String getSelectedText()	返回选定的文本。如果选定为 null 或文档为空，则返回 null。
boolean isEditable()	返回是否可编辑
void setEditable(boolean b)	设置文本框是否可编辑
void select(int start, int end)	选定从start开始到end结束的文本
void selectAll()	选中所有文本

## 9.4.3、按钮：JButton、JCheckBox和JRadioButton

Java Programming

```
import java.awt.*;
import javax.swing.*;
public class TestJButton extends JFrame{
    TestJButton(String sTitle){
        super(sTitle);
        Container c = getContentPane();
        c.setLayout( new FlowLayout());
        //两个按钮上的图标
        ImageIcon[] ic = {new ImageIcon("new.gif"), new ImageIcon("open.gif")};
        //三个按钮
        JButton[] btn = {new JButton("新建", ic[0]), new JButton("中间"),
                        new JButton("打开", ic[1])};

        int i;
        for (i=0; i<btn.length; i++)
            c.add(btn[i]);
        //btn[0]的文字在图标左侧
        btn[0].setHorizontalTextPosition(SwingConstants.LEFT);
        //两个复选框
        JCheckBox[] ck = {new JCheckBox("左"), new JCheckBox("右")};
        for (i=0; i<ck.length; i++){
            c.add(ck[i]);
            ck[i].setSelected(true); //将复选框设置为选中状态
        }
    }
}
```



TestJButton.java

## 9.4.3、按钮：JButton、JCheckBox和JRadioButton

Java Programming

//两个单选框

```
JRadioButton[] r={new JRadioButton("左"), new JRadioButton("右")};
```

```
ButtonGroup rg = new ButtonGroup( );
```

```
for (i=0; i < r.length; i++){
```

```
    c.add( r[i] );
```

```
    rg.add( r[i] );//组成ButtonGroup, 这样二者只能同时选中一项
```

```
}
```

//设置单选框的选择状态

```
r[0].setSelected(true);
```

```
r[1].setSelected(false);
```

```
setSize(300,150);
```

```
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
}
```

```
public static void main(String args[]){
```

```
    TestJButton frm=new TestJButton("JFrame with JButton");
```

```
    frm.setVisible(true);
```

```
}
```

```
}
```



如果改为:

```
r[0].setSelected(true);
```

```
r[1].setSelected(true);
```



## 9.4.4、下拉框：JComboBox

### TestJComboBox.java

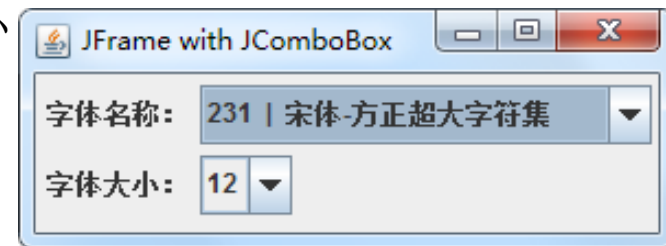
```
import java.awt.*;
import javax.swing.*;
public class TestJComboBox extends JFrame{
    //字体与大小下拉框
    JComboBox cbxFont=new JComboBox();
    JComboBox cbxFontSize=new JComboBox();//字体大小
    TestJComboBox(String sTitle){
        super(sTitle);

        Container c = getContentPane( );
        c.setLayout(new FlowLayout(FlowLayout.LEFT));

        c.add(new JLabel("字体名称: "));
        c.add(cbxFont);
        c.add(new JLabel("字体大小: "));
        c.add(cbxFontSize);

        //初始化字体与大小下拉框
        InitFonts();

        setSize(300,120);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```



## 9.4.4、下拉框：JComboBox

//初始化字体框

```
private void InitFonts(){
```

```
    //获得系统的字体数组
```

```
    GraphicsEnvironment ge= GraphicsEnvironment.getLocalGraphicsEnvironment();
```

```
    String[] fontList=ge.getAvailableFontFamilyNames();
```

```
    int i;
```

```
    //添加字体名称
```

```
    for(i=0;i<fontList.length;i++)
```

```
        cbxFont.addItem(String.valueOf(i)+" | "+fontList[i]);
```

```
    cbxFont.setSelectedIndex(231);//选择index为231的项
```

```
    //添加字体大小
```

```
    for(i=9;i<=72;i++)
```

```
        cbxFontSize.addItem(new Integer(i).toString());
```

```
    cbxFontSize.setSelectedIndex(3);//选择index为3的项
```

```
}
```

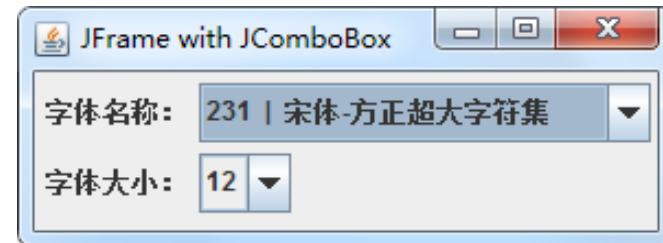
```
public static void main(String args[]){
```

```
    TestJComboBox frm=new TestJComboBox("JFrame with JComboBox");
```

```
    frm.setVisible(true);
```

```
}
```

```
}
```



```
import java.awt.*;
import javax.swing.*;
public class TestJList extends JFrame{
    //声明列表框对象
    JList listNames=new JList();
    TestJList(String sTitle){
        super(sTitle);
```

```
        Container c = getContentPane( );
        //以下语句保证列表框数据较多时会出现滚动条
        JScrollPane scrollPane = new JScrollPane(listNames);
        c.add(scrollPane);
        //初始化列表框
        InitList();
        setSize(250,150);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
```



TestJList.java

//初始化列表框

```
private void InitList(){
```

```
    String[] names={"201126100101 - 曹帝胄","201126100111 - 洪峰",  
                    "201126100128 - 徐华鹏","201126100131 - 姚臻平",  
                    "201126100202 - 陈思行","201126100207 - 姜楠",  
                    "201126100210 - 林一民","201126100211 - 林泽伟"};
```

//用数组填充列表框

```
    listNames.setListData(names);
```

//将1、3的项(201126100111和201126100131)设置为选择状态

```
    listNames.setSelectedIndices(new int[]{1, 3});
```

```
}
```

```
public static void main(String args[]){
```

```
    TestJList frm=new TestJList("JFrame with JList");
```

```
    frm.setVisible(true);
```

```
}
```

```
}
```



## 9.4.7、表格组件：JTable

```
import java.awt.Dimension;
import javax.swing.JFrame;
import javax.swing.JScrollPane;
import javax.swing.JPanel;
import javax.swing.JTable;
import java.awt.Color;
import java.awt.GridLayout;
import javax.swing.table.TableColumn;
public class TestJTable{
```

```
    public static void main (String[] args) {
```

```
        JTable table1 = new JTable (12, 6); //12行6列的空表格
```

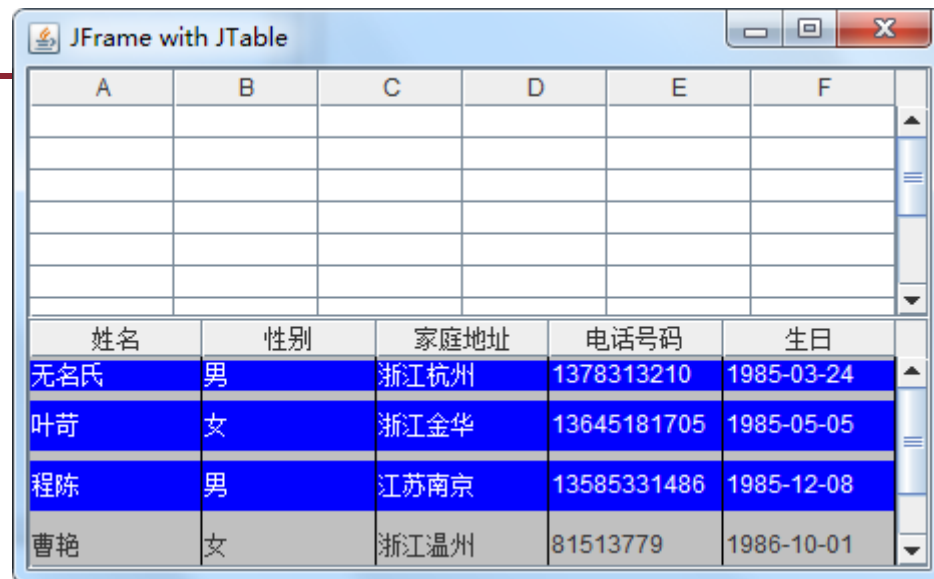
```
        //定义列名与行数据，其中列名最好用final修饰
```

```
        final Object[] columnNames = {"姓名", "性别", "家庭地址", "电话号码", "生日"};
```

```
        Object[][] rowData = {{"张国伟", "男", "浙江杭州", "1378313210", "1985-03-24"},
                                {"叶苛", "女", "浙江金华", "13645181705", "1985-05-05"},
                                {"程陈", "男", "江苏南京", "13585331486", "1985-12-08"},
                                {"曹艳", "女", "浙江温州", "81513779", "1986-10-01"},
                                {"刘飞", "男", "浙江宁波", "13651545936", "1985-12-25"}};
```

```
        //创建表格
```

```
        JTable table2= new JTable (rowData, columnNames);
```



TestJTable.java

//设置表格属性

table2.setRowHeight (30);//设置每行的高度为30

table2.setRowHeight (0, 20);//设置第1行的高度为20， 作为区别

table2.setRowMargin (5);//设置相邻两行的距离

table2.setRowSelectionAllowed (true);//设置可否被选择，默认为false

table2.setSelectionBackground (Color.BLUE);//设置所选择行的背景颜色

table2.setSelectionForeground (Color.WHITE);//设置所选择行的前景色

table2.setGridColor (Color.BLACK);//设置网格线的颜色

table2.setRowSelectionInterval (0,2);//设置选择行的范围

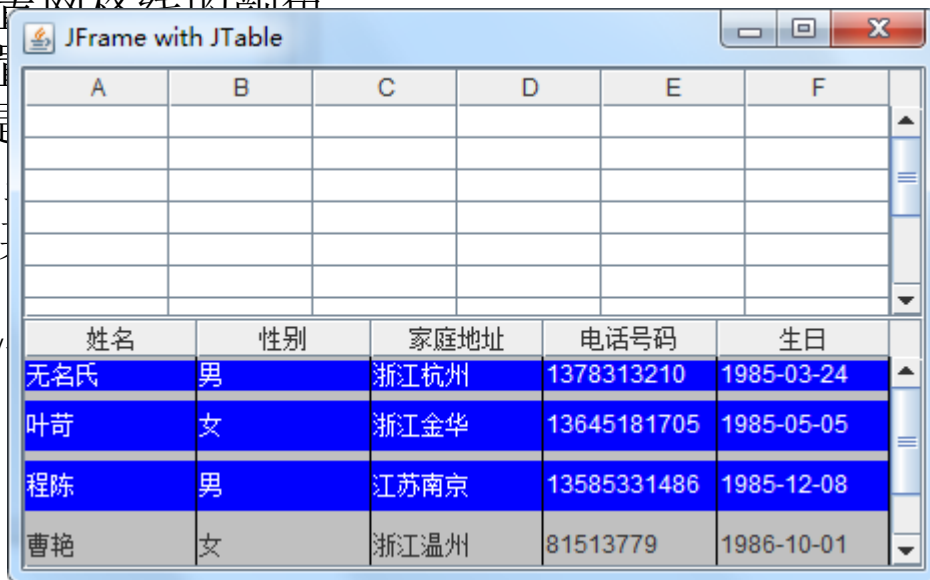
table2.setShowHorizontalLines (false);//是否显示水平线

table2.setShowVerticalLines (true);//是否显示垂直线

table2.setValueAt ("无名氏", 0, 0);//设置表格数据

table2.doLayout ();

table2.setBackground (Color.lightGray);//设置表格背景色



A	B	C	D	E	F

姓名	性别	家庭地址	电话号码	生日
无名氏	男	浙江杭州	1378313210	1985-03-24
叶苛	女	浙江金华	13645181705	1985-05-05
程陈	男	江苏南京	13585331486	1985-12-08
曹艳	女	浙江温州	81513779	1986-10-01

//设置表格的大小

```
table2.setPreferredScrollableViewportSize(new Dimension(600, 100));
```

## //创建窗口中将要用到的面板

```
JScrollPane panel1 = new JScrollPane (table1);
```

```
JScrollPane pane2 = new JScrollPane (table2);
```

```
JPanel pan = new JPanel (new GridLayout (0, 1));
```

```
pan.setPreferredSize (new Dimension (600,250));
```

```
pan.setBackground (Color.black);
```

```
pan.add (pane1);
```

```
pan.add (pane2);
```

```
//创建窗口
```

```
JFrame frm = new JFrame ("JFrame")
```

## frm.setDefaultCloseOperation

```
frm.setContentPane (pan);
```

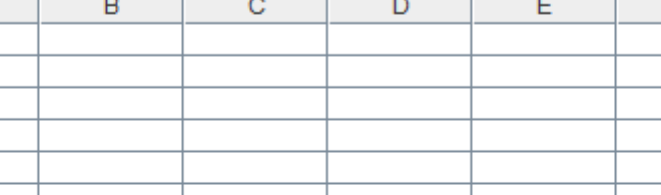
```
frm.pack();
```

```
frm.setVisible(true);
```

}

}

# TestJTable.java



A	B	C	D	E	F
姓名	性别	家庭地址	电话号码	生日	
无名氏	男	浙江杭州	1378313210	1985-03-24	
叶苛	女	浙江金华	13645181705	1985-05-05	
程陈	男	江苏南京	13585331486	1985-12-08	
曹艳	女	浙江温州	81513779	1986-10-01	

### 图形用户界面编程

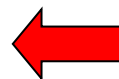
9.1、AWT与Swing

9.2、容器：JFrame\JPanel\JScrollPane\JSplitPane

9.3、菜单和工具条

9.4、基本组件：JLabel\JButton\JComboBox\JTree等

9.5、组件常用方法



9.6、布局管理器

9.7、事件处理模型

9.8、鼠标事件处理

9.9、事件适配器类

9.10、键盘事件处理



JComponent是常用组件的直接或间接父类，它封装了这些组件通用的一些方法，如颜色、透明性、边框、字体、大小和位置等相关方法

### ❖ 颜色

#### 颜色相关的public方法

方法定义	功能说明
void setBackground(Color c)	设置组件的背景色为c
void setForeground(Color c)	设置组件的前景色为c
Color getBackground()	获取组件的背景色
Color getForeground()	获取组件的前景色

例如：

//下面代码片段用于设置JLabel的背景色以及字体颜色

```
import java.awt.Color;
```

```
...
```

```
JLabel myLabel = new JLabel("测试颜色的标签");
```

```
//将背景设置为蓝色(红、绿分量为0)
```

```
myLabel.setBackground(new Color(0,0,255));
```

```
//将前景即字体颜色设置为白色
```

```
myLabel.setForeground(Color.WHITE);
```

### ❖ 透明性

#### 透明性相关的public方法

方法定义	功能说明
<code>void setOpaque(boolean isOpaque)</code>	设置组件是否不透明，当参数 <code>isOpaque</code> 取false时，组件被设置为 透明；否则，为不透明
<code>boolean isOpaque()</code>	组件不透明，返回true；否则，返 回false

### ❖ 边框

`void setBorder(Border border)` // 设置组件的边框  
`Border getBorder()` // 返回边框

例如：

```
//JLabel默认没有边框，通过下面语句设置线宽为2像素的红色边框  
JLabel myLabel = new JLabel("测试边框的标签");  
myLabel.setBorder(BorderFactory.createLineBorder (Color.RED, 2));
```

E.g.,

PPT图片阴影

### ❖ 字体

`void setFont(Font f)` //设置字体，f封装了字体的名称、样式等  
`Font getFont()` //返回组件上当前使用的字体

例如：

```
//
JLabel label1 =new JLabel("标签1");
JLabel label2 =new JLabel("标签2");
JLabel label3 =new JLabel("标签3");

label1.setFont(new Font("宋体",Font.PLAIN,14));
label2.setFont(new Font("Times New Roman",Font.BOLD,15));
label3.setFont(new Font("Arial",Font.BOLD | Font.ITALIC,16));
```

### ❖ 大小和位置

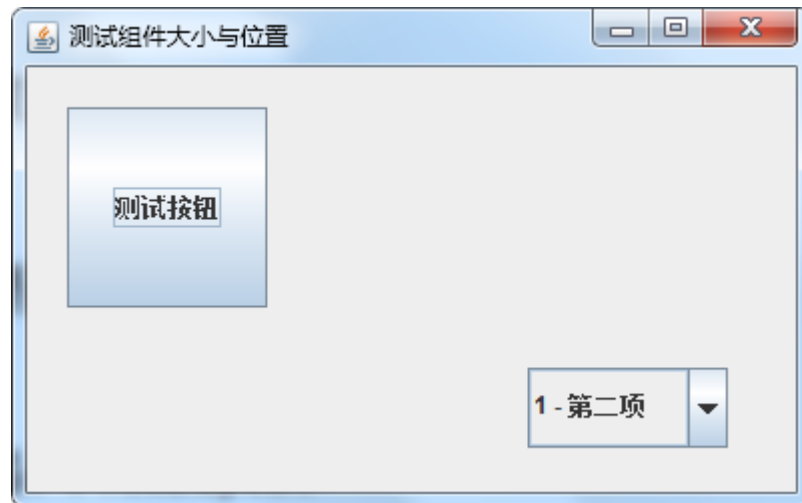
大小与位置相关的public方法

方法定义	功能说明
<code>void setSize(int width, int height)</code>	将组件的宽设置为width，高height，像素
<code>void setLocation(int x, int y)</code>	将组件左上角定位在所在容器的(x, y)处。包含组件的容器都有默认的坐标系，通常其左上角的坐标是(0, 0)，参数x和y指定该组件的左上角在容器的坐标系中的坐标。
<code>Dimension getSize()</code>	返回一个Dimension对象的引用，该对象包含有名字是width和height的成员变量，分别表示组件的宽和高，单位为像素
<code>void setBounds(int x, int y, int width, int height)</code>	设置组件在容器中的位置和大小
<code>Rectangle getBounds()</code>	返回组件在容器中的位置和大小

```
import java.awt.*;
import javax.swing.*;
public class TestSize extends JFrame{
    TestSize(String sTitle){
        super(sTitle);
        Container c=getContentPane();
        c.setLayout(null);//将布局设置空，否则setBounds不起作用
        JComboBox jcb=new JComboBox();
        JButton btnTest=new JButton("测试按钮");

        jcb.addItem("0 - 第一项");
        jcb.addItem("1 - 第二项");
        jcb.setSelectedIndex(1);

        c.add(btnTest);
        c.add(jcb);
        //设置组件位置与大小
        jcb.setBounds(250,150, 100,40);
        btnTest.setBounds(20,20, 100,100);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(400, 250);
    }
    public static void main(String[] args){
        TestSize frm=new TestSize("测试组件大小与位置");
        frm.setVisible(true);
    }
}
```



### ❖ 激活与可见性

`void setEnabled(boolean b)` //设置组件是否可以激活

`void setVisible(boolean b)` //设置组件是否可见

例如：

```
JButton btnStop=new JButton("停止");
```

//设置非激活状态，鼠标点击无反应，其背景与文字颜色均为灰色

```
btnStop.setEnabled(false);
```



### 图形用户界面编程

9.1、AWT与Swing

9.2、容器：JFrame\JPanel\JScrollPane\JSplitPane

9.3、菜单和工具条

9.4、基本组件：JLabel\JButton\JComboBox\JTree等

9.5、组件常用方法

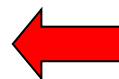
9.6、布局管理器

9.7、事件处理模型

9.8、鼠标事件处理

9.9、事件适配器类

9.10、键盘事件处理



### ❖ 事件处理机制

- **事件源**：事件源回答事件是由谁发生的，也就是事件发生的场所或者来源，通常是组件的对象，例如按钮JButton、下拉框JComboBox、列表框JList、树JTree等。
- **事件对象**：事件对象主要回答发生了什么事情。事件对象本身封装了包含所发生的各种事件的有效信息，包括事件源对象以及处理该事件所需要的其它各种信息(如鼠标点击时的坐标等)，这些有效信息被封装在类AWTEvent或其子类的实例对象中。
- **事件监听器**：事件监听器主要回答当某个事件发生由谁处理以及怎么处理。一旦注册完成一个事件监听器，它将能接受事件对象并进行处理。

### ❖ 事件对象

- 所有事件类均在**java.awt.event**包中
- 常用的事件类包括ActionEvent、AdjustmentEvent、FocusEvent、InputEvent、KeyEvent、MouseEvent、WindowEvent、ItemEvent、TextEvent等
- 继承于**java.awt.AWTEvent**类

### ❖ 监听器接口

- 要处理某个事件，必须实现某个监听器接口。这类似于交通事故是一种事件对象，而成立交通部门并完善处理制度相当于成立了事件监听器(实现了某个监听器接口)，而对外发布公文规定：凡是交通事故即交给该交通部门进行处理，这个过程相当于注册完成事件监听器。
- 每种类型的事件，都定义了相应的事件处理(监听器)接口，其命名规则是**XXXEvent**事件对应的事件处理(监听器)接口通常命名为**XXXListener**。这相当于整个java体系对每种类型的事件规定了必须成立什么部门进行处理，但如何处理则交给程序员去设计，即编写事件处理程序去覆盖对应接口中的所有方法。

## 9.7、事件处理模型

常用事件接口与对应的事件(动作)描述

事件描述信息	监听器接口	监听器接口中的方法
单击按钮、菜单项等动作	ActionListener	actionPerformed(ActionEvent e)
选择了复选框、单选按钮、下拉框或列表框	ItemListener	itemStateChanged(ItemEvent e)
移动了滚动条等组件	AdjustmentListener	adjustmentVlaueChanged (AdjustmentEvent e)
鼠标移动	MouseMotionListener	mouseDragged(MouseEvent e) mouseMoved(MouseEvent e)
按下或释放鼠标按键	MouseListener	mousePressed(MouseEvent e) mouseReleased(MouseEvent e) mouseEntered(MouseEvent e) mouseExited(MouseEvent e) mouseClicked(MouseEvent e)
键盘输入	KeyListener	keyPressed(KeyEvent e) keyReleased(KeyEvent e) keyTyped(KeyEvent e)
组件收到或失去焦点	FocusListener	focusGained(FocusEvent e) focusLost(FocusEvent)
组件移动、缩放、显示/隐藏等	ComponentListener	componentMoved(ComponentEvent e) componentHidden(ComponentEvent e) componentResized(ComponentEvent e) componentShown(ComponentEvent e)

要编写一个完整的事件处理程序，通常包含如下四部分的内容：

①引入系统事件类包：**import java.awt.event.\***

②自定义事件处理类，即加上**implements XXXListener**，如：

```
public class MyFrame extends JFrame implements ActionListener{  
    ...  
}
```

③注册事件源对象的监听者，即告诉程序一旦发生相应的事件后，由谁处理，如：

```
public class MyFrame extends JFrame implements ActionListener{  
    MyFrame(String sTitle){  
        super(sTitle);  
        JButton btn=new JButton("确定");  
        ...  
        btn.addActionListener (this);  
        ...  
    }  
}
```

这个例子中，`btn.addActionListener(this)`即注册监听者，它相当于宣布一旦发生点击事件(`ActionEvent`，对应的事件监听器接口为 `ActionListener`，而 `MyFrame` 已经 `implements` 了该接口)由 `this` 处理，`this` 即为 `MyFrame`。

④注册了监听者还不能响应相应的事件，还需要实现监听器接口中的所有方法。例如，在上例中，需加入如下代码：

```
//实现ActionListener接口中的方法  
public void actionPerformed(ActionEvent e) {  
    ...//响应某个动作的代码...  
}
```



## 9.7.4、编写事件处理程序

import java.awt.event.\* ;//第①步，引入事件包

import java.awt.\*;

import javax.swing.\*;

//第②步，即声明implements某个监听器接口

public class TestJButtonClick implements ActionListener{//事件监听者与事件源属于同一个类

public TestJButtonClick(){

    JFrame f = new JFrame("单击按钮事件");

    Container c=f.getContentPane();

    JButton b = new JButton("Press Me!");

    b.addActionListener(this);//第③步，注册监听者

    c.add(b, "Center");

    f.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

    f.setSize(200,100);

    f.setVisible(true);

}

public void actionPerformed(ActionEvent e){//第④步，监听者如何监听

    //e.getActionCommand()方法返回事件源的名称

    JOptionPane.showMessageDialog(null,

        "你点击了按钮\""+e.getActionCommand()+"\"",  
        "提示",

        JOptionPane.INFORMATION\_MESSAGE);

}

public static void main(String args[ ]){

    new TestJButtonClick();

}

}



TestJButtonClick.java

```
import java.awt.*;
import java.awt.event.*; //第①步，引入事件包
import javax.swing.*;

public class TestJButtonClick1 { //事件监听者与事件源不属于同一类
    TestJButtonClick1() {
        JFrame f = new JFrame("事件监听者与事件源所在的类分离");
        Container c = f.getContentPane();
        JButton b = new JButton("Press Me!");

        b.addActionListener(new JButtonHandler()); //第③步，注册监听者
        c.add(b, "Center");

        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setSize(200, 100);
        f.setVisible(true);
    }
    public static void main(String args[]) {
        new TestJButtonClick1();
    }
}
```

TestJButtonClick1.java

**//第②步，单独定义一个类声明implements某个监听器接口**

```
class JButtonHandler implements ActionListener{
```

**//第④步，监听者如何监听**

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

```
        JOptionPane.showMessageDialog(null,
```

```
            "你点击了按钮\""+e.getActionCommand()+"\",
```

```
            "提示",
```

```
            JOptionPane.INFORMATION_MESSAGE);
```

```
    }
```

```
}
```

```
import java.awt.*;
import java.awt.event.*; //第①步，引入事件包
import javax.swing.*;
```

```
public class TestJButtonClick2{ //利用匿名类实现监听
```

```
    TestJButtonClick2(){
```

```
        JFrame f = new JFrame("匿名类实现监听");
```

```
        Container c=f.getContentPane();
```

```
        JButton b = new JButton("Press Me!");
```

```
        //以下利用匿名类实现第②③④步
```

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

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

```
                JOptionPane.showMessageDialog(null,
```

```
                    "你点击了按钮\""+e.getActionCommand()+"\"",
```

```
                    "提示",
```

```
                    JOptionPane.INFORMATION_MESSAGE);
```

```
            }
```

```
        });
```

```
        c.add(b, "Center");
```

```
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
        f.setSize(200,100);
```

```
        f.setVisible(true);
```

```
    }
```

```
    public static void main(String args[ ]){
```

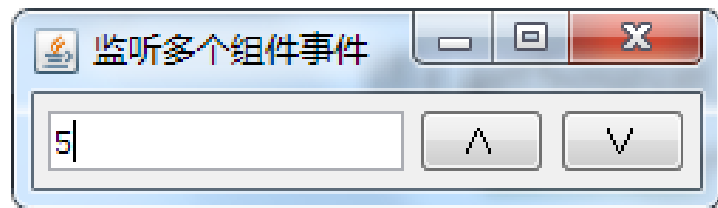
```
        new TestJButtonClick2();
```

```
    }
```

```
}
```

## 9.7.4、编写事件处理程序

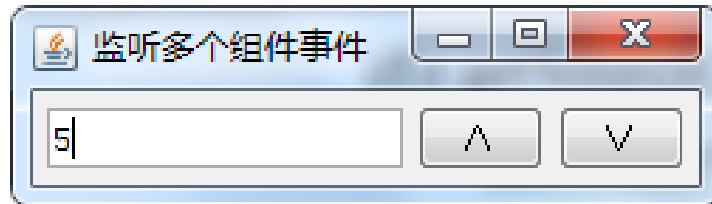
```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*; //监听多个组件
public class TestListenMulti extends JFrame implements ActionListener{
    JTextField txtNumber;
    JButton btnInc, btnDec;
    public void initComponents(){
        Container c=getContentPane();
        c.setLayout(new FlowLayout());
        //添加单行框
        txtNumber = new JTextField("0",20);
        c.add(txtNumber);
        //btnInc按钮
        btnInc = new JButton("<");
        c.add(btnInc);
        btnInc.addActionListener(this);
        //btnDec按钮
        btnDec = new JButton(">");
        c.add(btnDec);
        btnDec.addActionListener(this);
        catch (Exception e){}
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        pack();
    }
}
```



TestListenMulti.java

## 9.7.4、编写事件处理程序

```
TestListenMulti(String sTitle){  
    super(sTitle);  
    initComponents();  
}
```



```
public void actionPerformed(ActionEvent e){  
    int oldNum = Integer.parseInt(txtNumber.getText());  
    int newNum = oldNum;  
    if(e.getSource()==btnInc)//点击了btnInc按钮  
        newNum++;  
    else if (e.getSource()==btnDec)//点击了btnDec按钮  
        newNum--;  
  
    txtNumber.setText(String.valueOf(newNum));  
}  
public static void main(String args[]) {  
    TestListenMulti f = new TestListenMulti ("监听多个组件事件");  
    f.setVisible(true);  
}  
}
```

### 图形用户界面编程

9.1、AWT与Swing

9.2、容器：JFrame\JPanel\JScrollPane\JSplitPane

9.3、菜单和工具条

9.4、基本组件：JLabel\JButton\JComboBox\JTree等

9.5、组件常用方法

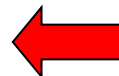
9.6、布局管理器

9.7、事件处理模型

9.8、鼠标事件处理

9.9、事件适配器类

9.10、键盘事件处理



鼠标事件的监听器接口有MouseListener(鼠标事件)、MouseMotionListener(鼠标移动事件)和MouseWheelListener(鼠标滚轮事件)，前两个接口中的鼠标事件对应的类为MouseEvent，鼠标滚轮事件则对应MouseWheelEvent。

MouseEvent常用的常量与方法

常量或方法定义	功能说明
MouseEvent.MOUSE_PRESSED	鼠标按下
MouseEvent.MOUSE_CLICKED	鼠标单击(按下并释放鼠标按键)
MouseEvent.MOUSE_RELEASED	鼠标松开
MouseEvent.MOUSE_ENTERED	鼠标进入
MouseEvent.MOUSE_EXITED	鼠标离开
int getX()	取得鼠标的X坐标
int getY()	取得鼠标的Y坐标
int getClickCount()	取得鼠标连续单击的次数



```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
class MousePanel extends JPanel{
    int x_pos,y_pos;
    MousePanel(){
        //注册鼠标事件监听器，并用匿名类来实现事件处理程序
        //注意，必须实现(覆盖)接口中的全部方法，哪怕实现代码一句也没有
        addMouseListener(new MouseListener() {
            public void mouseClicked(MouseEvent e) {}
            public void mouseClicked(MouseEvent e){}
            public void mouseEntered(MouseEvent e){}
            public void mouseExited(MouseEvent e){}
            public void mouseReleasedsePressed(MouseEvent e){
                x_pos=e.getX();
                y_pos=e.getY();
                repaint();//本方法会自动触发paintComponent方法的运行
            }
        });
    }
};
```

//注册鼠标移动事件监听器，并用匿名类来实现事件处理程序

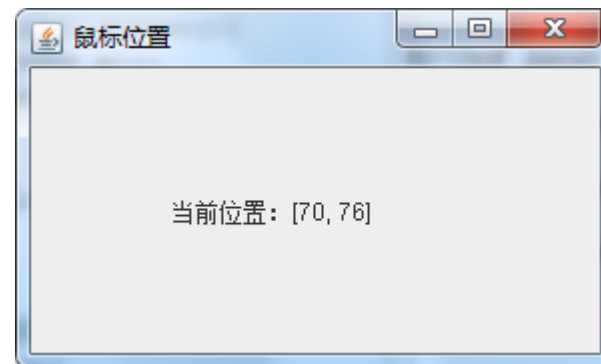
```
addMouseListener(new MouseMotionListener(){  
    public void mouseDragged(MouseEvent e){  
    public void mouseMoved(MouseEvent e){  
        x_pos=e.getX();  
        y_pos=e.getY();  
        repaint();//本方法会自动触发paintComponent方法的运行  
    }  
});
```

//覆盖父类的paintComponent方法以绘制当前鼠标的坐标

```
protected void paintComponent(Graphics g){  
    super.paintComponent(g);  
    g.drawString( "当前位置: [" + x_pos + ", " + y_pos + "]",x_pos, y_pos);  
}
```

```
}  
public class TestMouseListener extends JFrame{  
    TestMouseListener(){  
        super("鼠标位置");  
        setContentPane(new MousePanel());  
    }  
    public static void main(String args[ ]){  
        TestMouseListener f = new TestMouseListener();  
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
        f.setSize(300, 180);  
        f.setVisible(true);  
    }  
}
```

TestMouseListener.java



### 图形用户界面编程

9.1、AWT与Swing

9.2、容器：JFrame\JPanel\JScrollPane\JSplitPane

9.3、菜单和工具条

9.4、基本组件：JLabel\JButton\JComboBox\JTree等

9.5、组件常用方法

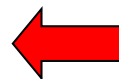
9.6、布局管理器

9.7、事件处理模型

9.8、鼠标事件处理

9.9、事件适配器类

9.10、键盘事件处理



- Java中为那些具有多个方法的监听者接口提供了事件适配器类，这个类通常命名为XXXAdapter，在该类中以空方法体实现了相应接口的所有方法；
- 可通过继承适配器类来编写监听者类，在类中只需给出关心的方法，从而减轻工作量。

Listener Interface	Adapter Class	Methods
ActionListener	无	<b>actionPerformed</b>
AdjustmentListener	无	<b>adjustmentValueChanged</b>
ComponentListener	<b>ComponentAdapter</b>	<b>componentHidden</b> <b>componentMoved</b> <b>componentResized</b> <b>componentShown</b>
ContainerListener	<b>ContainerAdapter</b>	<b>componentAdded</b> <b>componentRemoved</b>
FocusListener	<b>FocusAdapter</b>	<b>focusGained</b> <b>focusLost</b>
ItemListener	无	<b>itemStateChanged</b>

Listener Interface	Adapter Class	Methods
KeyListener	KeyAdapter	keyPressed keyReleased keyTyped
MouseListener	MouseAdapter	mouseClicked mouseEntered mouseExited mousePressed mouseReleased
MouseMotionListener	MouseMotionAdapter	mouseDragged mouseMoved
TextListener	无	textValueChanged
WindowListener	WindowAdapter	windowActivated windowClosed windowClosing windowDeactivated windowDeiconified windowIconified windowOpened

例如，将前述鼠标事件示例的源代码TestMouseListener.java中的注册监听器的代码改为如下即可实现同样的功能：

```
...
//匿名继承MouseAdapter类
addMouseListener(new MouseAdapter(){
    public void mousePressed(MouseEvent e){
        x_pos=e.getX();
        y_pos=e.getY();
        repaint();
    }
});
//匿名继承MouseMotionAdapter类
addMouseMotionListener(new MouseMotionAdapter(){
    public void mouseMoved(MouseEvent e){
        x_pos=e.getX();
        y_pos=e.getY();
        repaint();
    }
});
...
```

MousePanel.java

### 图形用户界面编程

9.1、AWT与Swing

9.2、容器：JFrame\JPanel\JScrollPane\JSplitPane

9.3、菜单和工具条

9.4、基本组件：JLabel\JButton\JComboBox\JTree等

9.5、组件常用方法

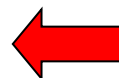
9.6、布局管理器

9.7、事件处理模型

9.8、鼠标事件处理

9.9、事件适配器类

9.10、键盘事件处理





键盘事件的监听器接口为KeyListener，适配器类为KeyAdapter，键盘事件对应的类是KeyEvent，位于java.awt.event包中。

KeyEvent常用的常量与方法

常量或方法定义	功能说明
KEY_PRESSED	"按下键"事件
KEY_RELEASED	"释放键"事件
KEY_TYPED	"键入键"事件，即按下并释放
VK_*	代表键盘上的某个键，如KeyEvent.VK_A到KeyEvent.VK_Z与ASCII码的'A'到'Z' (0x41 - 0x5A) 相同，KeyEvent.VK_F1表示F1功能键，依此类推。
char getKeyChar()	返回与此事件中的键相关联的字符。例如，shift + "a" 的KEY_TYPED 事件返回值"A"。本方法只适用于KEY_TYPED事件。
int getKeyCode()	返回与此事件中的键相关联的整数keyCode。与getKeyChar不同的地方在于，本方法返回的是getKeyCode得到的是键码常量，是在KeyEvent定义的常量（比如KeyEvent.VK_Z 代表Z），对应物理键；而getKeyChar返回的是实际输入的字符（区分大小写）
int getKeyLocation()	某些键在键盘上有多个，如数字键、shift键、ctrl键等，本方法就是提供了一种区分这些键的方式。
static String getKeyModifiersText(int modifiers)	返回描述组合键的 String，如"Shift"或"Ctrl+Shift"
static String getKeyText(int keyCode)	返回描述 keyCode 的 String，如"HOME"、"F1"或"A"

例：

//下面的代码用于判断是不是左边的shift键：

```
if(e.getKeyCode()==KeyEvent.VK_SHIFT &  
    e.getKeyLocation()==KeyEvent.KEY_LOCATION_LEFT){  
    ...  
}
```

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

public class TestKeyListener extends JFrame{
    TestKeyListener(String sTitle){
        super(sTitle);
        Container c = getContentPane();
        c.setLayout(new GridLayout(4,1,2,2));
        //选择角色
        JPanel panel1=new JPanel();
        panel1.setLayout(new FlowLayout(FlowLayout.LEFT));
        panel1.add(new JLabel("选择角色: "));
        JComboBox jcb=new JComboBox();
        jcb.addItem("教师");
        jcb.addItem("学生");
        jcb.setSelectedIndex(1);
        panel1.add(jcb);
        c.add(panel1);
```

TestKeyListener.java

//输入学号

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

```
panel2.setLayout(new FlowLayout(FlowLayout.LEFT));
```

```
panel2.add(new JLabel("输入学号： "));
```

```
FgTextField txtNum=new FgTextField("", 15, true);
```

```
panel2.add(txtNum);
```

```
c.add(panel2);
```

//输入密码

```
JPanel panel3=new JPanel();
```

```
panel3.setLayout(new FlowLayout(FlowLayout.LEFT));
```

```
panel3.add(new JLabel("输入密码： "));
```

```
panel3.add(new JPasswordField("", 15));
```

```
c.add(panel3);
```

//登录按钮

```
JPanel panel4=new JPanel();
```

```
panel4.setLayout(new FlowLayout(FlowLayout.RIGHT));
```

```
panel4.add(new JButton("登录"));
```

```
panel4.add(new JButton("取消"));
```

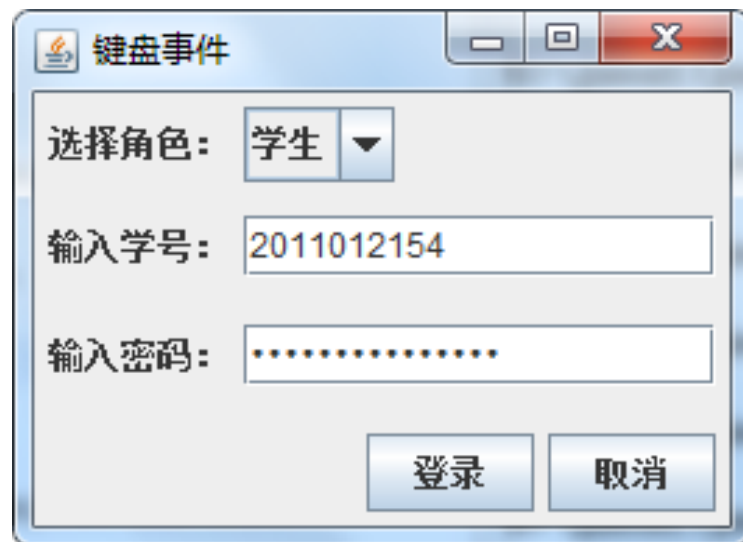
```
c.add(panel4);
```

```
}
```

```
public static void main(String args[ ]){
    TestKeyListener f = new TestKeyListener("键盘事件");
    f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    f.pack();
    f.setVisible(true);
}

//自定义单行文本框，从JTextField继承过来
class FgTextFiled extends JTextField{
    boolean m_bOnlyInteger;//用于指示是否只允许输入整数
    FgTextFiled(String sText, int columns, boolean bOnlyInteger){
        super(sText,columns);

        m_bOnlyInteger=bOnlyInteger;
        addKeyListener(new KeyAdapter(){
            public void keyTyped(KeyEvent e){
                if(m_bOnlyInteger){
                    char c=e.getKeyChar();
                    if(c<'0' | c>'9')
                        e.consume(); //取消输入
                }
            }
        });
    }
}
```



### 图形用户界面编程

9.1、AWT与Swing

9.2、容器：JFrame\JPanel\JScrollPane\JSplitPane

9.3、菜单和工具条

9.4、基本组件：JLabel\JButton\JComboBox\JTree等

9.5、组件常用方法

9.6、布局管理器

9.7、事件处理模型

9.8、鼠标事件处理

9.9、事件适配器类

9.10、键盘事件处理

### 图形用户界面编程

9.1、AWT与Swing

9.2、容器：JFrame\JPanel\JScrollPane\JSplitPane

9.3、菜单和工具条

9.4、基本组件：JLabel\JButton\JComboBox\JTree等

9.5、组件常用方法

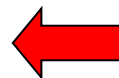
9.6、布局管理器

9.7、事件处理模型

9.8、鼠标事件处理

9.9、事件适配器类

9.10、键盘事件处理



- 流式布局管理器(FlowLayout)把容器看成一个行集，好象平时在一张纸上写字一样，一行写满就换下一行，行高由一行中最高的组件决定。FlowLayout是所有 JApplet或JPanel的默认布局。

FlowLayout的构造方法

方法定义	功能说明
FlowLayout()	生成一个默认的流式布局，组件在容器里在水平方向上居中显示，组件之间的间距、行与行的间距均为5个像素
FlowLayout(int alignment)	设定每行组件的对齐方式为alignment，组件之间的间距、行与行的间距均为5个像素。 alignment可取值：FlowLayout.LEFT、FlowLayout.CENTER或FlowLayout.RIGHT
FlowLayout(int alignment,int horGap,int verGap)	设定每行组件的对齐方式为alignment，并设定组件水平方向上的间距为horGap像素，行间距为verGap像素

例：

Eg01\_FlowLayout.java

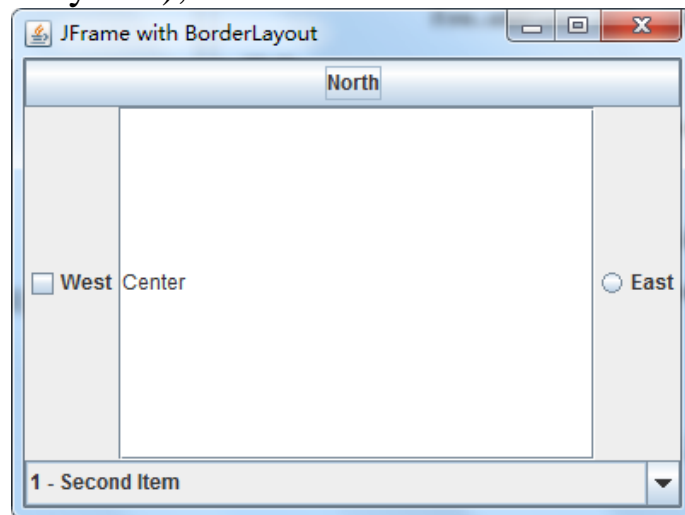
```
//使面板靠左对齐，组件从左上角开始，按从左至右的方式排列。  
JPanel panel= new JPanel(new FlowLayout(FlowLayout.LEFT));
```



- BorderLayout 是一种非常简单的布局策略，它把容器内的空间简单地划分为东(East)、西(West)、南(South)、北(North)、中(Center)**五个区域**，每加入一个组件都应该指明把这个组件加在哪个区域中，而**每个区域最多只能放一个组件**，默认的情况是加入到中间。
- 是顶层容器(**JFrame**、**JDialog**和 **JApplet**)的**默认布局管理器**。
- 位于**java.awt**包中
- 不一定所有的区域都有组件，如果四周的区域(West、East、North、South区域)没有组件，则由Center区域去填充(即填满整个容器空间)，但是如果 Center区域没有组件，则保持空白。
- 如果容器的大小发生变化，其变化规律为：组件的相对位置不变，大小发生变化，例如容器变高了，则North、South 区域不变，West、Center、East区域变高；如果容器变宽了，West、East区域不变，North、Center、South区域变宽；如果容器变高变宽，North、South变宽但高度不变，West、East变高但宽度不变，Center则变高变宽。

## 9.6.2、边界布局: BorderLayout

```
import ...
public class TestBorderLayout{
    public static void main(String args[]){
        JFrame frm= new JFrame("JFrame with BorderLayout");
        frm.setLayout(new BorderLayout());
        //North放置一个按钮
        frm.add("North", new JButton("North"));
        //South放置一个下拉框JComboBox
        JComboBox jcb = new JComboBox();
        jcb.addItem("0 - First Item");
        jcb.addItem("1 - Second Item");
        jcb.setSelectedIndex(1);
        frm.add(jcb, "South");
        //East放置一个单选框JRadioButton
        frm.add(BorderLayout.EAST, new JRadioButton("East"));
        //West放置一个复选框JCheckBox
        frm.add(new JCheckBox("West"), BorderLayout.WEST);
        //Center放置一个单行文本框JTextField
        frm.add("Center", new JTextField("Center"));
        //窗口属性
        frm.setSize(400,300);
        frm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frm.setVisible(true);
    }
}
```



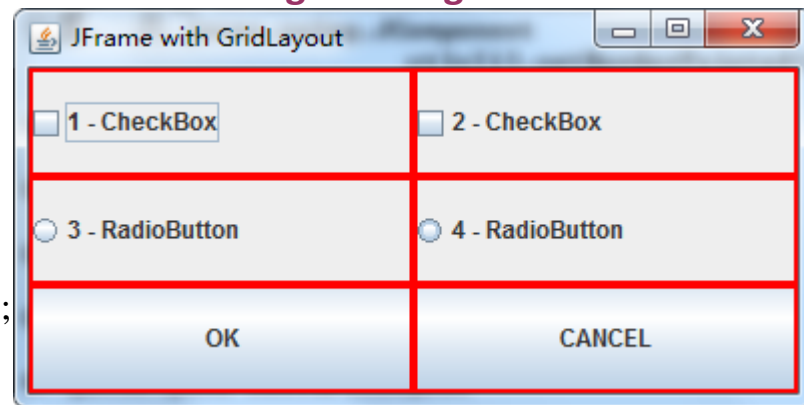
- 位于**java.awt**包中
- 把容器的空间划分成若干行与若干列的网格区域，组件就位于这些划分出来的小区域中，**每个网络区域只能放一个组件**，所有的区域大小一样，组件按从上到下、从左到右依次加入，每个小区域尽可能地占据网格的空间，每个网格也同样尽可能地占据空间，从而各个组件按一定的大小比例放置。
- 如果改变容器大小，GridLayout将相应地改变每个网格的大小，以使各个网格尽可能地大，占据Container容器全部的空间。

### GridLayout的构造方法

方法定义	功能说明
GridLayout()	生成一个默认的网络布局
GridLayout(int rows, int cols)	生成rows行与cols列的网络布局，间距均为5个像素
GridLayout(int rows,int cols,int horz,int vert)	生成rows行与cols列的网络布局，并设定网格列间距为horz像素，行间距为vert像素

## 9.6.4、网格布局：GridLayout

```
import java.awt.*;
import javax.swing.*;
import javax.swing.border.*;
import java.awt.Color;
public class TestGridLayout{
    public static void main(String args[]){
        JFrame f=new JFrame("JFrame with GridLayout");
        Container c=f.getContentPane();
        c.setLayout(new GridLayout(3,2));//3行2列
        JComponent[] ctls={new JCheckBox("1 - CheckBox"), new JCheckBox("2 - CheckBox"),
                            new JRadioButton("3 - RadioButton"), new JRadioButton("4 - RadioButton"),
                            new JButton("OK"), new JButton("CANCEL")};
        for(int i=0;i<ctls.length;i++){
            c.add(ctls[i]);//从上到下， 从左到右添加组件
            //设置边框
            if(i>=0 & i<2)
                ((JCheckBox)ctls[i]).setBorderPainted(true);
            if(i>=2 & i<4)
                ((JRadioButton)ctls[i]).setBorderPainted(true);
            ctls[i].setBorder(BorderFactory.createLineBorder (Color.RED, 2));
        }
        f.setSize(400,200);
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setVisible(true);
    }
}
```



- 位于**java.awt**包中
- 它把容器分成许多层，每层的显示空间占据整个容器的大小，但是每层只允许放置一个组件，但每层都可以利用JPanel来实现复杂的用户界面。
- CardLayout就象一副叠得整整齐齐的**扑克牌**一样，但是你能只能看见最上面的一张牌，每一张牌就相当于布局管理器中的每一层。

## 9.6.5、卡片式布局: CardLayout

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class TestCardLayout implements MouseListener{
    CardLayout layout = new CardLayout();
    JFrame f = new JFrame("JFrame with CardLayout");
    Container c;
    JButton page1Button;
    JLabel page2Label; //标签,一行文本
    JTextArea page3Text; // 多行多列的文本区域
    JButton page3Top;
    JButton page3Bottom;

    public static void main(String args[]) {
        TestCardLayout frm=new TestCardLayout();
        frm.show();
    }
    public void mouseClicked(MouseEvent arg0) {
        layout.next(c);
    }
    public void mouseEntered(MouseEvent arg0){}
    public void mouseExited(MouseEvent arg0){}
    public void mousePressed(MouseEvent arg0){}
    public void mouseReleased(MouseEvent arg0){}
```

## 9.6.5、卡片式布局：CardLayout

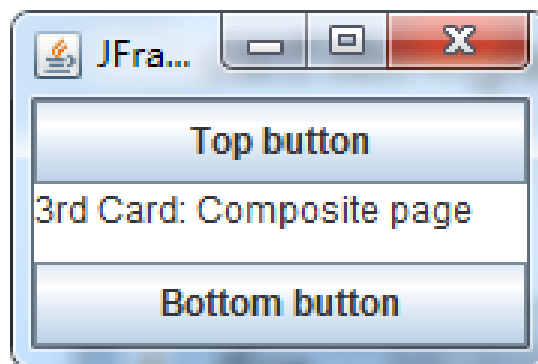
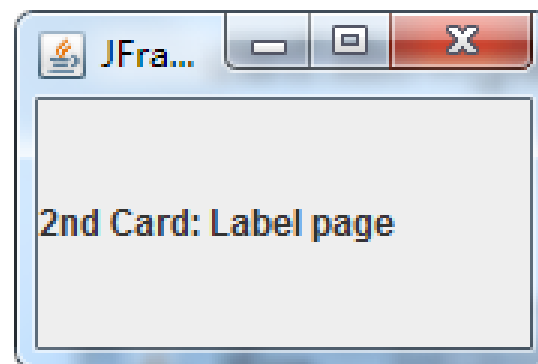
```
public void show() {
    c=f.getContentPane();
    c.setLayout(layout); // 设置为CardLayout
    //第一张Card放了一个JButton
    c.add(page1Button = new JButton("1st Card: Button page"), "page1Button");
    page1Button.addMouseListener(this); // 注册监听器

    //第二张Card放了一个JLabel
    c.add(page2Label = new JLabel("2nd Card: Label page"), "page2Label");
    page2Label.addMouseListener(this); // 注册监听器

    //第三张Card放一个JPanel，在这个JPanel上又放了一个JTextArea、两个JButton
    JPanel pan= new JPanel();
    pan.setLayout(new BorderLayout());
    pan.add(page3Text = new JTextArea("3rd Card: Composite page"), "Center");
    page3Text.addMouseListener(this);
    pan.add(page3Top = new JButton("Top button"), "North");
    page3Top.addMouseListener(this);
    pan.add(page3Bottom = new JButton("Bottom button"), "South");
    page3Bottom.addMouseListener(this);
    c.add(pan, "panel");

    f.setSize(180, 120);
    f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    f.setVisible(true);
}
}
```

## 9.6.5、卡片式布局: CardLayout





### ❖ 相关注意事项

- JFrame是一个顶级窗口。**JFrame的缺省**布局管理器为**BorderLayout**。
- JPanel无法单独显示，必须添加到某个容器中。**JPanel的缺省布局管理器为FlowLayout**。
- 当把JPanel作为一个组件添加到某个容器中后，该JPanel仍然可以有自己的布局管理器。因此，可以利用JPanel使得BorderLayout中某个区域显示多个组件，达到设计复杂用户界面的目的。
- 如果采用无布局管理器 setLayout(null)，则必须使用 setLocation(); setSize(); setBounds()等方法手工设置组件的大小和位置，但是此方法会导致平台相关，不鼓励使用。
- 除了**BoxLayout**位于**javax.swing包**中，其余**5种布局管理器均位于java.awt包**中。

### ❖ 参考准则

- 若组件尽量充满容器空间，可以考虑使用BorderLayout
- 若用户需要在紧凑的一行中以组件的自然尺寸显示较少的组件，用户可以考虑用面板容纳组件，并使用面板的默认布局管理器FlowLayout。
- 若用户需要在多行或多列中显示一些同样尺寸的组件，GridLayout最适合此情况。
- 若界面较为复杂，可先使用面板来容纳组件，然后选用适当的布局管理器。

### ❖ 容器变化时，布局的变化规律

- **FlowLayout**: 容器大小发生变化，组件的大小不变，但是相对位置会发生变化。
- **BorderLayout, GridLayout**: 容器的大小发生变化，组件的相对位置不变，大小发生变化。

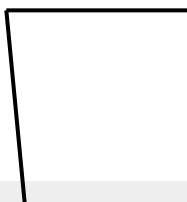
## 9.6.7、布局基本原则及复杂布局举例

Java Programming



## 9.6.7、布局基本原则及复杂布局举例

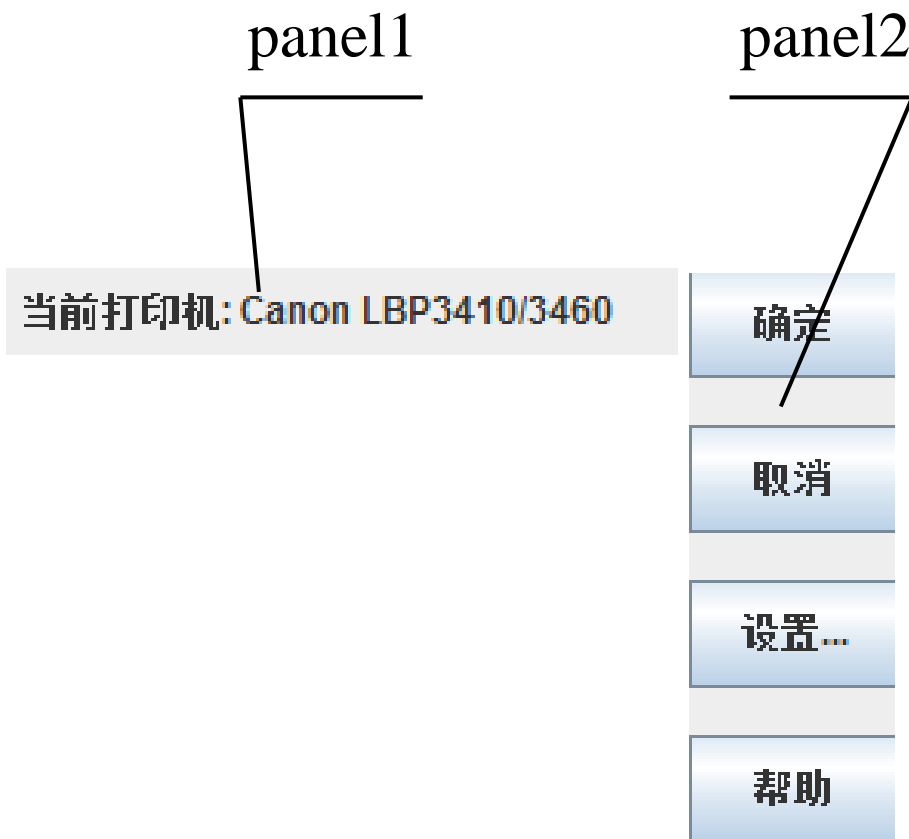
panel1



当前打印机: Canon LBP3410/3460

```
//创建panel1
JPanel panel1=new JPanel();
panel1.setLayout(new
FlowLayout(FlowLayout.LEFT));
panel1.add(new JLabel("当前打印机: Canon
LBP3410/3460"));
```

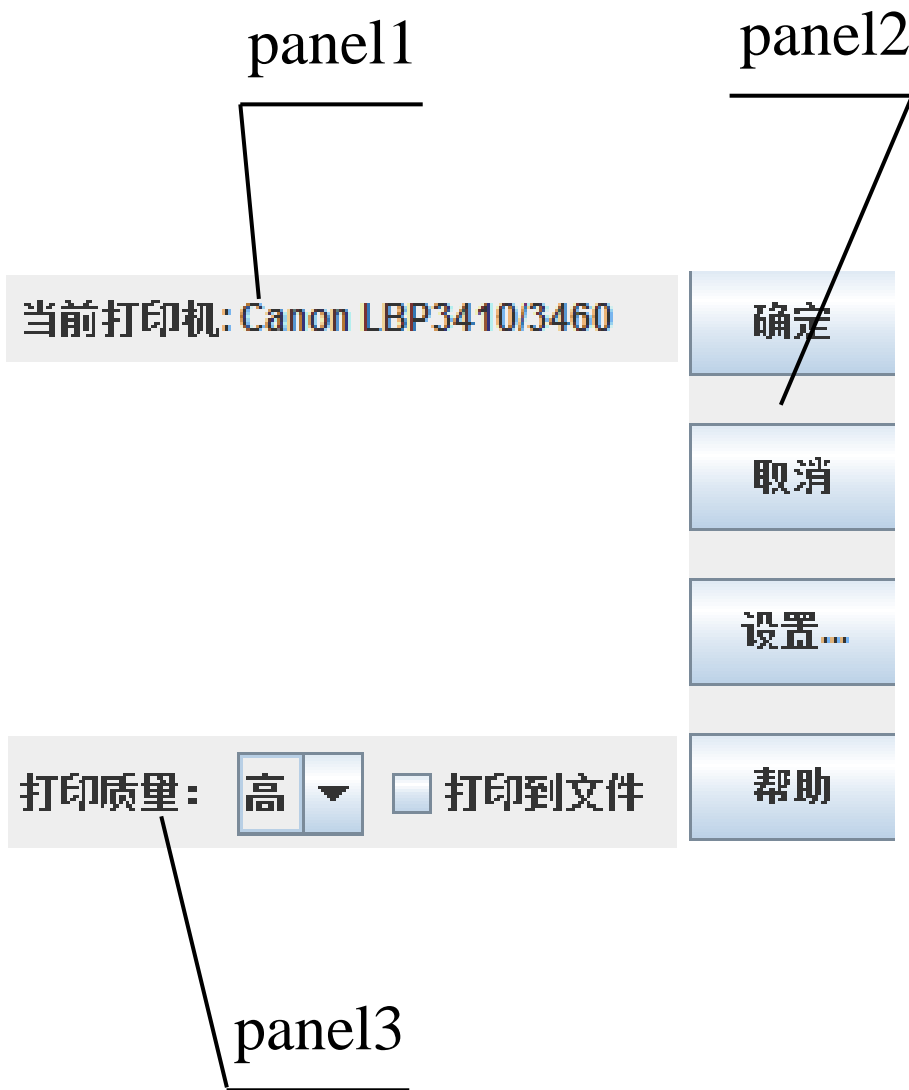
## 9.6.7、布局基本原则及复杂布局举例



```
//创建panel2
JPanel panel2=new JPanel();
panel2.setLayout(new GridLayout(4,1,15,15));
JButton[] btn={new JButton("确定"),
               new JButton("取消"),
               new JButton("设置..."),
               new JButton("帮助")};

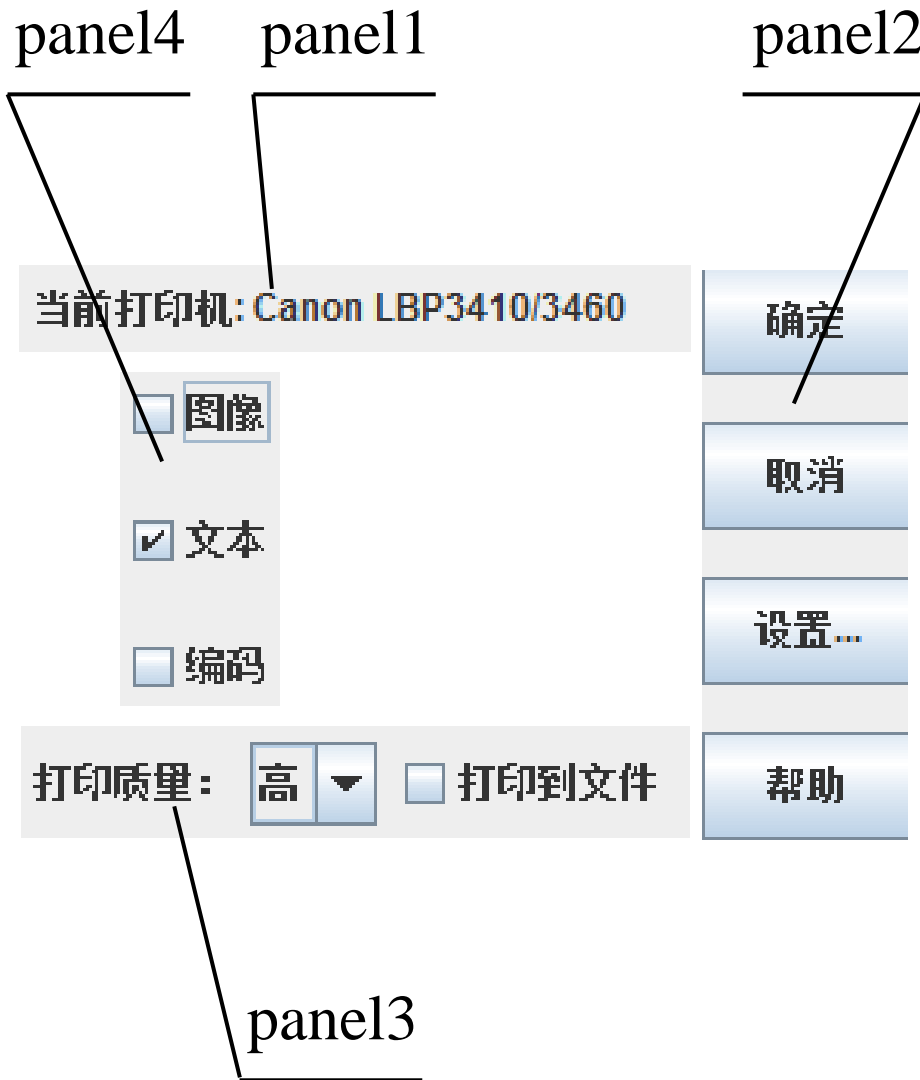
int maxWidth=0;
int i;
for(i=0;i<btn.length;i++)
    panel2.add(btn[i]);
```

## 9.6.7、布局基本原则及复杂布局举例



```
//创建panel3
JPanel panel3=new JPanel();
panel3.add(new JLabel("打印质量: "));
panel3.setLayout(new
FlowLayout(FlowLayout.LEFT));
JComboBox jcb=new JComboBox();
jcb.addItem("高");
jcb.addItem("中");
jcb.addItem("低");
jcb.setSelectedIndex(0);
panel3.add(jcb);
panel3.add(new JCheckBox("打印到文件"));
```

## 9.6.7、布局基本原则及复杂布局举例

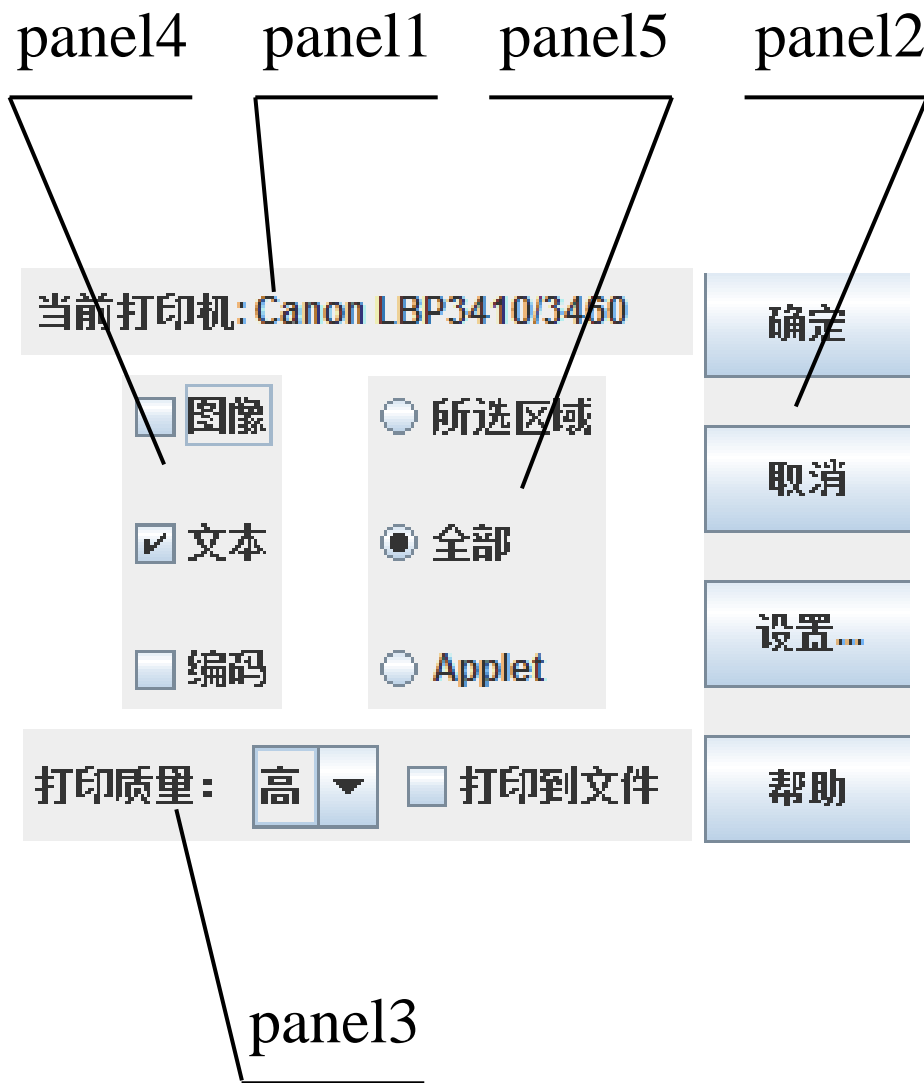


//创建panel4

```
JPanel panel4=new JPanel();  
panel4.setLayout(new GridLayout(3,1,15,15));  
panel4.add(new JCheckBox("图像"));  
panel4.add(new JCheckBox("文本",true));  
panel4.add(new JCheckBox("编码"));
```



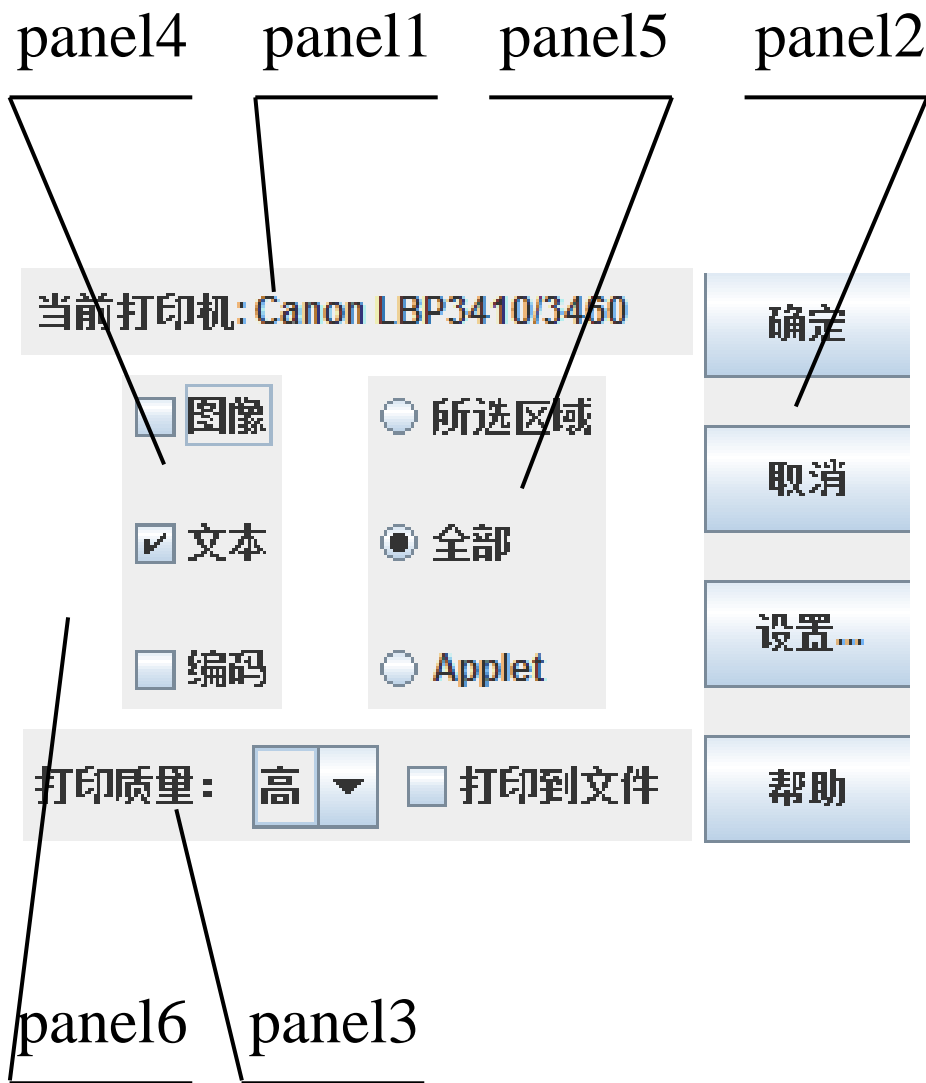
## 9.6.7、布局基本原则及复杂布局举例



```
//创建panel5
JPanel panel5=new JPanel();
panel5.setLayout(new GridLayout(3,1,15,15));
ButtonGroup bg=new ButtonGroup();
JRadioButton[] rb={new JRadioButton("所选区域"),
    new JRadioButton("全部",true),
    new JRadioButton("Applet")};
for(i=0;i<rb.length;i++){
    bg.add(rb[i]);
    panel5.add(rb[i]);
}
```

## 9.6.7、布局基本原则及复杂布局举例

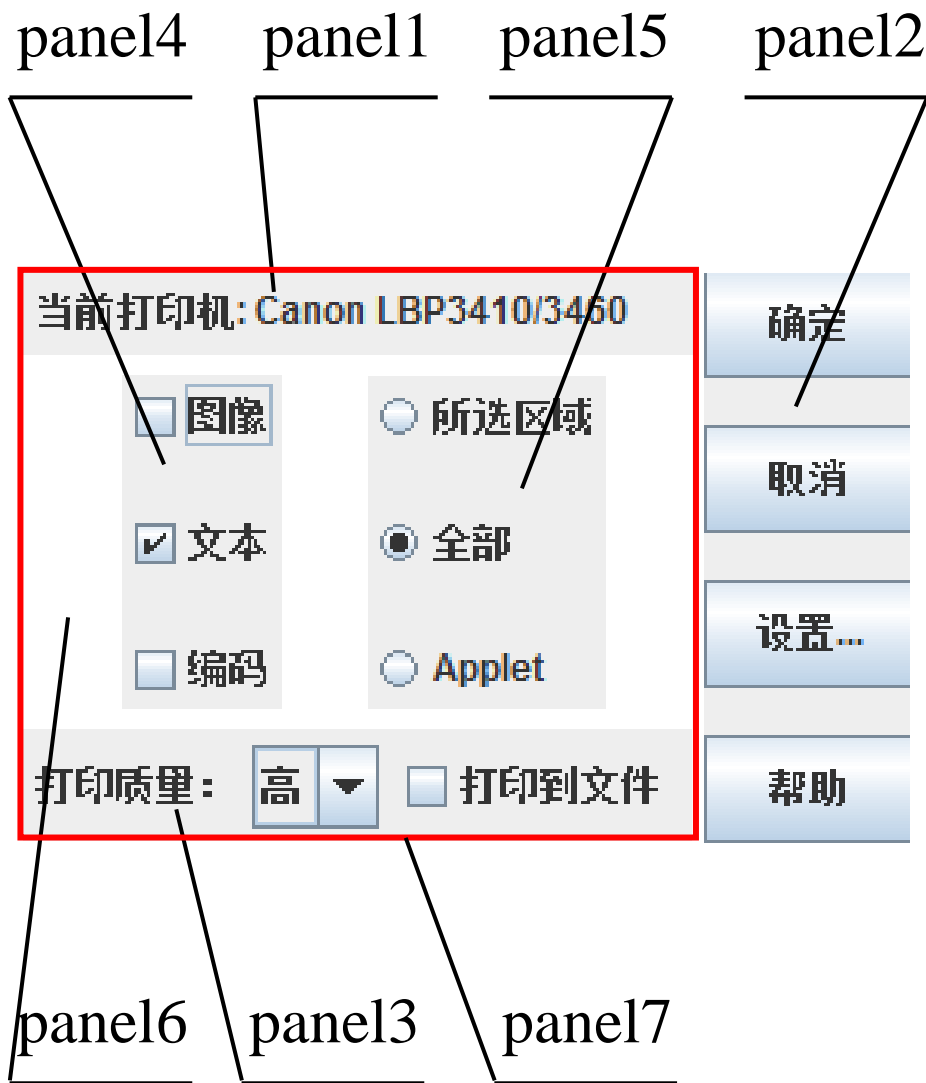
Java Programming



```
//创建panel6,  
//并将panel4、panel5添加到panel6上面  
JPanel panel6=new JPanel();  
panel6.setBackground(Color.WHITE);  
panel6.setLayout(new  
FlowLayout(FlowLayout.LEFT,30,5));  
panel6.add(panel4);  
panel6.add(panel5);
```

## 9.6.7、布局基本原则及复杂布局举例

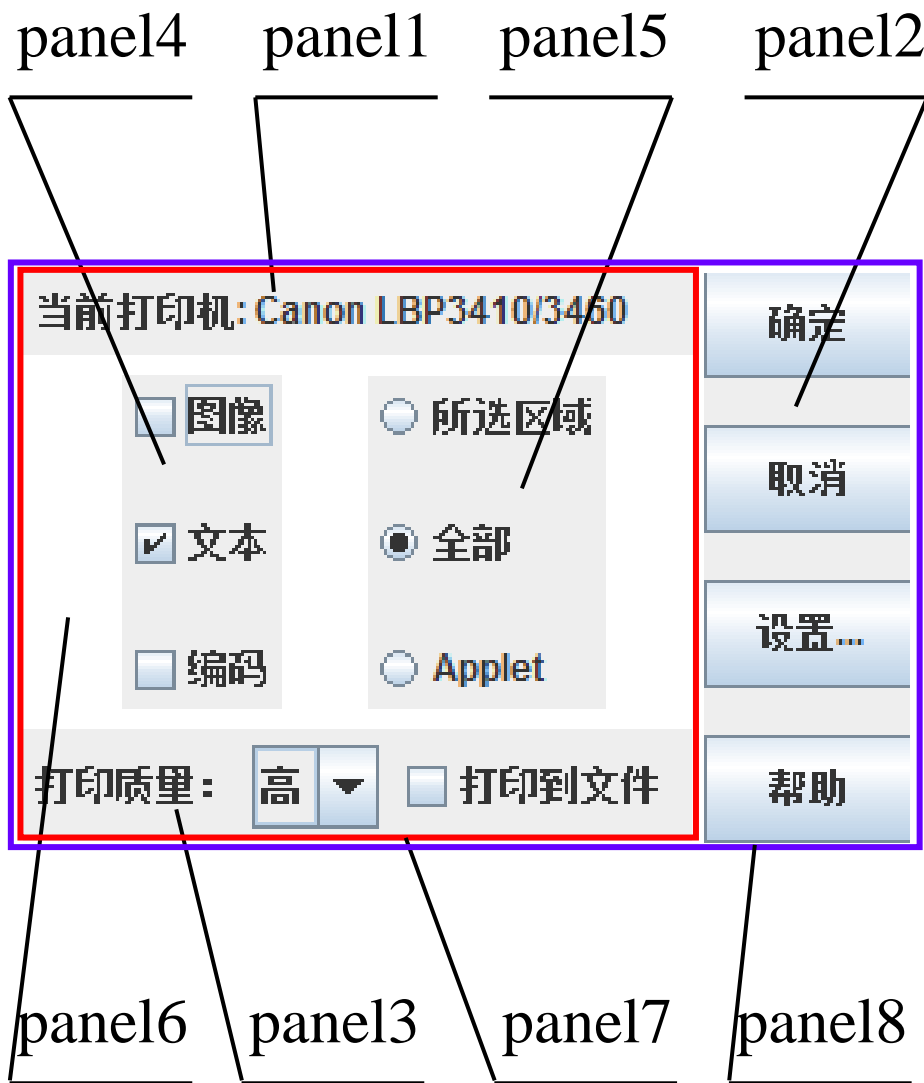
Java Programming



```
//创建panel7,  
//并将panel1、panel3、panel6添加到  
panel7中  
JPanel panel7=new JPanel();  
panel7.setLayout(new BorderLayout());  
panel7.add(panel1, "North");  
panel7.add(panel6, "Center");  
panel7.add(panel3, "South");  
panel7.setBorder(BorderFactory.createLi  
neBorder(Color.RED, 2));//红色边框
```

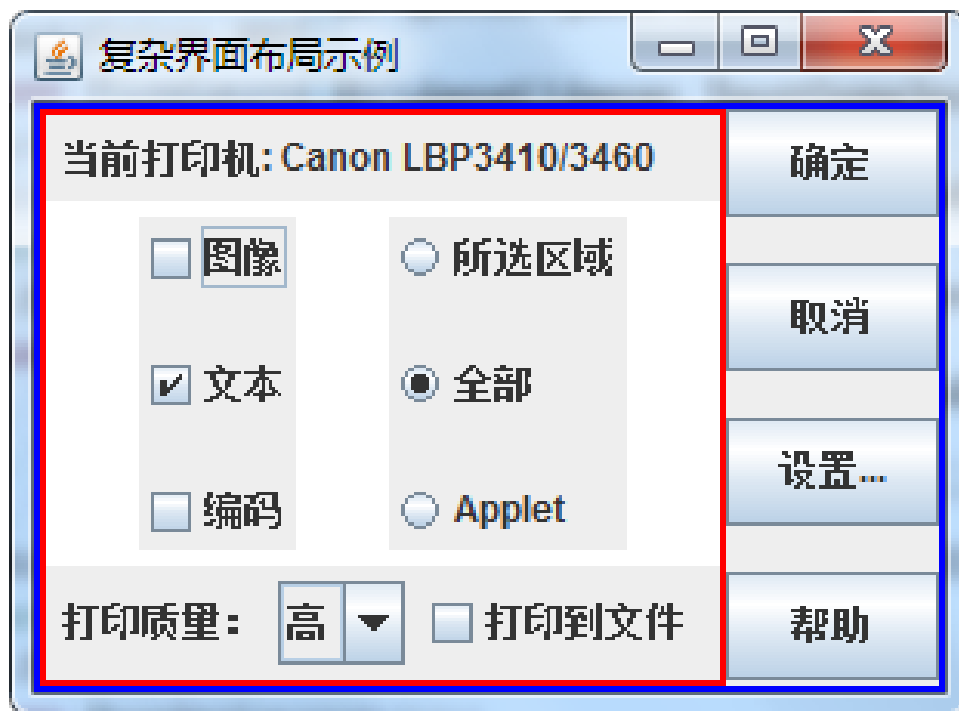
## 9.6.7、布局基本原则及复杂布局举例

Java Programming



```
//创建panel8,  
//并将panel7、panel2添加到panel8中  
JPanel panel8=new JPanel();  
panel8.setLayout(new BorderLayout());  
panel8.add(panel7, "West");  
panel8.add(panel2, "Center");  
panel8.setBorder(BorderFactory.createLineBorder(Color.BLUE, 2));//蓝色边框
```

## 9.6.7、布局基本原则及复杂布局举例



//将默认的内容面板替换成panel8  
setContentPane(panel8);

### ❖ 获取当前系统支持的界面风格

```
UIManager.LookAndFeelInfo[] ui=UIManager.getInstalledLookAndFeels();  
for(i=0;i<ui.length;i++)  
    System.out.println((i+1)+" - "+ui[i].getName()+": "+ui[i].getClassName());
```

系统(Windows 7简体中文版)环境下，上述代码输出如下：

- 1 - Metal: javax.swing.plaf.metal.MetalLookAndFeel
- 2 - Nimbus: com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel
- 3 - CDE/Motif: com.sun.java.swing.plaf.motif.MotifLookAndFeel
- 4 - Windows: com.sun.java.swing.plaf.windows.WindowsLookAndFeel
- 5 - Windows Classic: com.sun.java.swing.plaf.windows.WindowsClassicLookAndFeel

### ❖ 设置风格

例：

```
//将如下代码片段插入到TestComplexLayout.java例程中 initComponents()方法  
//中 setContentPane(panel8)的后面  
try{  
    //以下风格简称 Nimbus  
    String lfClassName="com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel";  
    UIManager.setLookAndFeel(lfClassName);//设置外观风格}  
catch (Exception e) { }
```

Eg04\_GridLayout.java

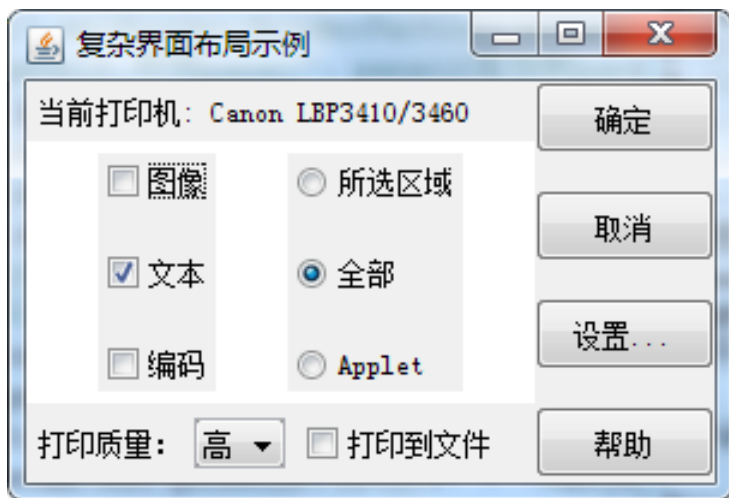
## 9.6.8、界面风格选择



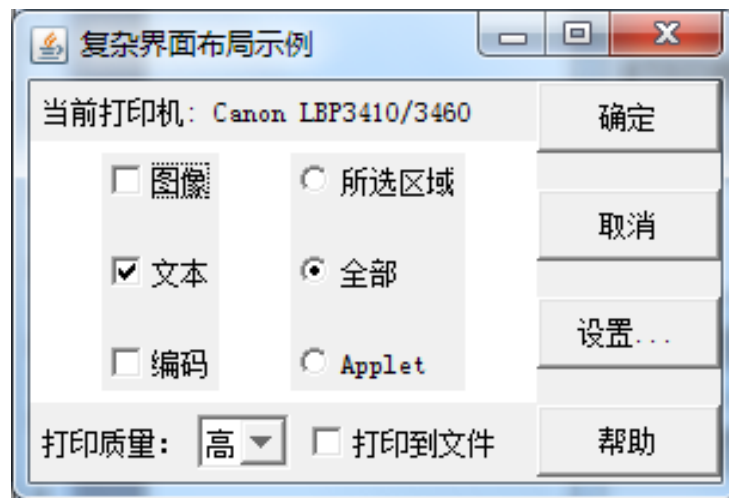
(a) Nimbus风格



(b) CDE/Motif风格



(c) Windows风格



(d) Windows Classic风格