[常用控件 2](#_Toc13443)

[1.按钮---JButton 2](#_Toc18779)

[2.文本标签 2](#_Toc27038)

[3.单行文本框 2](#_Toc10597)

[4.多行文本框(文本域) 2](#_Toc4493)

[5.复选框 2](#_Toc1804)

[6.下拉列表 3](#_Toc28511)

[9.菜单MenuBar-->Menu-->MenuItem 3](#_Toc9185)

[10.选项卡 4](#_Toc9752)

[容器窗体JFrame 4](#_Toc18352)

[基本设置（窗体实现类构造方法） 4](#_Toc5086)

[面板的布局方式 5](#_Toc24424)

[事件处理 6](#_Toc25533)

[基于监听的事件处理模型 6](#_Toc31825)

[基于适配器的事件处理程序 7](#_Toc29321)

C/S:client/server 客户端/服务器(运行效率高)

B/S:browser/server 浏览器/服务器(利于维护)

AWT:(abstract window toolkit)---抽象窗口工具包

java.awt.\*包重量级(依赖于操作系统)

Swing:javax.swing.\*包轻量级(不依赖与操作系统)

java界面：1 界面元素/控件（所有的控件都是类）和布局2事件处理

基本所有组件都有getText和setText两个方法,分别用来设置属性值和得到属性值

setBounds(x,y,a,b);设置坐标及长宽 setBackground;设置背景颜色

setEnabled;是否可编辑 setIcon;设置背景图 setToolTipText;提示信息

常用控件

1.按钮---JButton（单选按钮JRadioButton/分组ButtonGroup）

rbMan = new JRadioButton("男");rbMan.setBounds(70, 80, 90, 20);

rbWoman = new JRadioButton("女"); rbWoman.setBounds(170, 80, 90, 20);

bg = new ButtonGroup();// 分组才能实现单选

bg.add(rbMan); bg.add(rbWoman); jp.add(rbMan);jp.add(rbWoman);

JButton btn1=new JButton("点我");

btn1.addActionListener(new ActionListener(){//按钮添加监听事件

public void actionPerformed(ActionEvent e) {

JOptionPane.showConfirmDialog(jf, "放假要写作业吗?");

}}); jp.add(btn1); jf.add(jp);

2.文本标签---JLabel只显示文字不能修改

设置文本内容:setText("内容");得到文本内容:getText(); //返回String内容

3.单行文本框---JTextField

JTextField(String text);---给文本框赋初始值

JTextField(int a);---限定文本框长度

JTextField(String text,int a);---先限定文本框长度,显示文本初始内容,如果长度不够显示,则隐藏在文本框长度之后,拖动鼠标才可见

4.多行文本框(文本域)---JTextArea

JTextArea(String text);---给文本域赋初始值

JTextArea(int a,int b);---设置文本域的行和列

JTextArea(String text,int a,int b);---设置文本域的行和列并赋初始值

文本域对象.setLineWrap(boolean b);---如果b为true则换行,否则不换行

jta = new JTextArea();jta.setBounds(70, 260, 100, 50);jp.add(jta);

5.复选框JCheckBox

h1= new JCheckBox("篮球"); h1.setBounds(70, 110, 60, 20);

h2= new JCheckBox("足球"); h2.setBounds(140, 110, 60, 20);

h3= new JCheckBox("电子竞技"); h3.setBounds(210, 110, 60, 20);

jp.add(h1); jp.add(h2); jp.add(h3);

6.下拉列表JComboBox

String[] options = {"重庆","海口","贵阳","上海"};

jcb= new JComboBox(options); jcb.setBounds(70, 140, 100, 20); jp.add(jcb);

7.列表JList显示所有的可选项

jlist = new JList(options); jlist.setBounds(70, 170, 100, 80); jp.add(jlist);

8.对话框（多种）

9.菜单MenuBar-->Menu-->MenuItem

JFramejf; JMenuBar bar;//菜单栏 JMenufile,edit,format;//三个菜单

JMenuItemcreate,open,save,cut,copy,paste,font;//菜单项

publicTestMenu(){

jf = new JFrame("菜单例子");jf.setSize(300, 300); jf.setVisible(true);

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

bar = new JMenuBar();// 1 创建菜单栏

file = new JMenu("文件(F)"); //2创建菜单

edit=new JMenu(“编辑(E)”);format =new JMenu(“格式化(O)”);

create = new JMenuItem("新建");// 3创建对应的菜单选项

open = new JMenuItem("打开"); save = new JMenuItem("保存");

file.add(create);// 4将菜单选项添加到菜单中

file.addSeparator();//分隔线

file.add(open); file.add(save);

bar.add(file);// 5将菜单添加到菜单栏

jf.setJMenuBar(bar);//6将菜单栏加到面板 }

public static void main(String[] args) {newTestMenu();}

10.选项卡

JFramejf;

JTabbedPanejtb;// 选项卡面板

publicTestTab() {

jf = new JFrame("选项卡例子"); jf.setSize(300, 300);

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

jtb = new JTabbedPane(JTabbedPane.TOP);//top bottom left right

jtb.add("国际新闻", new JLabel("奥巴马下台，trump上台"));

jtb.add("国内新闻", new JLabel("油价又涨了！"));

jf.add(jtb);

jf.setVisible(true);

}

public static void main(String[] args) {newTestTab();}

容器窗体JFramenew类且继承JFrame,构造方法中设置窗体的各种属性及功能

基本设置（窗体实现类构造方法）

setTitle("我的第一个窗体");//设置窗体标题setSize(400, 400);//设置窗体大小

setBackground(Color.blue);//设置背景颜色setResizable(false);//设置窗体是否可拉伸

setLocationRelativeTo(null);//设置窗体居中(null表示居中)

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);//设置窗体的关闭

setDefaultCloseOperation(3);//设置窗体的关闭setVisible(true);//设置窗体是否可视

ImageIcon ii = new ImageIcon("F:/lgxr28/20160331/img/1.jpg");//设置图标

Image image = ii.getImage();this.setIconImage(image);

\*\*\*窗体操作之前必须先在窗体上得到一个面板:Container/JPanel

A Container c=getContentPane();B JPanel jp = new JPanel();//得到主面板A或B

GridLayoutgl = new GridLayout(7, 1); jp.setLayout(gl); this.add(jp);//主面板布局

JPanel jp1 = new JPanel();jp1.setLayout(null); //子面板取消默认的布局方式，空布局JLabel jl1 = new JLabel("用户信息管理");jl1.setBounds(80, 0, 200, 50); jp1.add(jl1);

JPanel jp2 = new JPanel();jp2.setLayout(null);// 子面板2

JLabel jl2 = new JLabel("姓名:");jl2.setBounds(25, 0, 50, 30);jp2.add(jl2);

JTextFieldjf = new JTextField();jf.setBounds(65, 5, 150, 25);jp2.add(jf);

面板的布局方式:

1.流式布局(先按照从左到右，从上到下):FlowLayout

2.网格布局:GridLayout

3.边界布局(整个界面分为五块；东西南北中):BorderLayout

4.坐标布局(空布局):setLayout(null);

5.坐标式的网格布局gridbagLayout

GridBagConstraints gc=new GridBagConstraints();//参数(指定位置x和y,延伸fill及长宽等)

6.盒子布局boxLayout 垂直或者水平布局

setLayout(new BoxLayout(this.getContentPane(),BoxLayout.X\_AXIS));

setLayout(new BoxLayout(this.getContentPane(),BoxLayout.Y\_AXIS));

7.groupLayout

GroupLayout 是一个 LayoutManager，它将组件按层次分组，以决定它们在 Container 中的位置。GroupLayout 主要供生成器使用，但也可以手工编码。分组由 [Group](http://doc.java.sun.com/DocWeb/api/javax.swing.GroupLayout.Group) 类的实例来完成。GroupLayout 支持两种组。串行组 (sequential group) 按顺序一个接一个地放置其子元素。并行组 (parallel group) 能够以四种方式对齐其子元素。GroupLayout必须要设置它的GroupLayout.setHorizontalGroup和GroupLayout.setVerticalGroup。

JPanel jp; JButton b1,b2,b3,b4;

public Gp() {

setTitle("组布局");setSize(400, 400);setDefaultCloseOperation(EXIT\_ON\_CLOSE);

b1 = new JButton("按钮一"); b2 = new JButton("按钮二");

b3 = new JButton("按钮三"); b4 = new JButton("按钮四");

GroupLayout layout = new GroupLayout(this.getContentPane());

setLayout(layout); layout.setAutoCreateContainerGaps(true);

layout.setAutoCreateGaps(true);//组间距

//创建连续的组，摆放按钮b1,b2,b3

GroupLayout.ParallelGroup pg=layout.createParallelGroup(Alignment.BASELINE);

pg.addComponent(b1);pg.addComponent(b2);pg.addComponent(b3);

//创建并行的组，组件放在其他组件上面

GroupLayout.SequentialGroup sg=layout.createSequentialGroup();

sg.addGroup(pg); sg.addComponent(b4);

//分组布局既要横向实现，也要纵向实 最终的纵向的分组

layout.setVerticalGroup(pg);

//方案二 最终的横向分组

GroupLayout.ParallelGroup pg1=layout.createParallelGroup(Alignment.LEADING);

pg1.addComponent(b3); pg1.addComponent(b4);

GroupLayout.SequentialGroup sg1=layout.createSequentialGroup();

sg1.addComponent(b1); sg1.addComponent(b2);

sg1.addGroup(pg1); layout.setHorizontalGroup(sg1);setVisible(true);

}

事件处理

基于监听的事件处理模型

1 什么是事件 用户在界面上的操作

2 事件源 用户操作的界面上的控件

3 事件监听程序（接口） 默认是静默，一旦发生事件 触发事件监听程序

4 事件处理程序 对用户操作做出响应代码

实现事件处理

1 为控件绑定/添加事件监听程序 调用 addActionListener方法

需要一个参数（接口类型），需要传入接口实现类

a.匿名内部类

b.当前类 implements 接口

c.创建一个其他类 implements 接口

2 实现事件处理程序

public class EventDemo extends JFrame implements ActionListener {

JPanel jp; JButton btn1; JTextField tf;

public EventDemo() {

setTitle("事件处理例子！");setDefaultCloseOperation(EXIT\_ON\_CLOSE);

setSize(250, 300); setResizable(false);

tf = new JTextField("初始文字"); btn1 = new JButton("点点");

// 接口实现类之创建一个其他类实现接口

EventHandeler hand=new EventHandeler(tf);btn1.addActionListener(hand);

// 接口实现类之当前类实现接口

//btn1.addActionListener(this);

// 接口实现类之匿名内部类

// btn1.addActionListener(new ActionListener() {

// public void actionPerformed(ActionEvent e) {

// tf.setText("点了就点了吧！");// } // });

jp = new JPanel(); jp.add(btn1); jp.add(tf); add(jp);

}

public static void main(String[] args) {

EventDemo e = new EventDemo(); e.setVisible(true);

}

public void actionPerformed(ActionEvent e) {// 接口实现类之当前类实现接口

tf.setText("点了就点了吧！");}}

public class EventHandeler implements ActionListener {

JTextField tf;// 接口实现类之创建一个其他类实现接口

public EventHandeler(JTextField tf) { this.tf=tf;}

public void actionPerformed(ActionEvent e) {tf.setText("abcd");}

}

基于适配器的事件处理程序

适配器:java自带的实现了监听的类

常用的适配器：

keyAdapter 键盘适配器 keyPressed keyReleased keyTyped

mouseAdapter 鼠标适配器

windowAdapter 窗体

实现步骤：

1 继承某个适配器

2 重写适配器中的方法

public class AdapterDemo extends MouseAdapter {

JPanel jp; JFrame jf; JTextField tf;

public AdapterDemo() {

jf = new JFrame(); jf.setTitle("基于适配器事件处理例子！");

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

jf.setSize(250, 300); jf.setResizable(false);

tf = new JTextField("显示鼠标坐标"); jp = new JPanel();

jp.add(tf);jf.add(jp);jf.addMouseListener(this);jf.setVisible(true); }

public static void main(String[] args) {

AdapterDemo am = new AdapterDemo(); }

public void mouseClicked(MouseEvent e) { //重写适配器的方法 实现事件处理

super.mouseClicked(e);tf.setText("x:" + e.getX() + "y:" + e.getY());}}