图形界面目录

[1. Hello Swing 1](#_Toc6979)

[2. 事件监听 3](#_Toc11785)

[3. 容器 7](#_Toc22474)

[4. 布局器 8](#_Toc21820)

[5. 组件 11](#_Toc18934)

[6. 面板 17](#_Toc19722)

[7. 练习：显示文件夹复制进度条 18](#_Toc26294)

[8. 菜单 21](#_Toc7396)

[9. 工具栏 21](#_Toc27893)

[10. 表格 21](#_Toc28766)

[11. 表格综合练习 27](#_Toc26881)

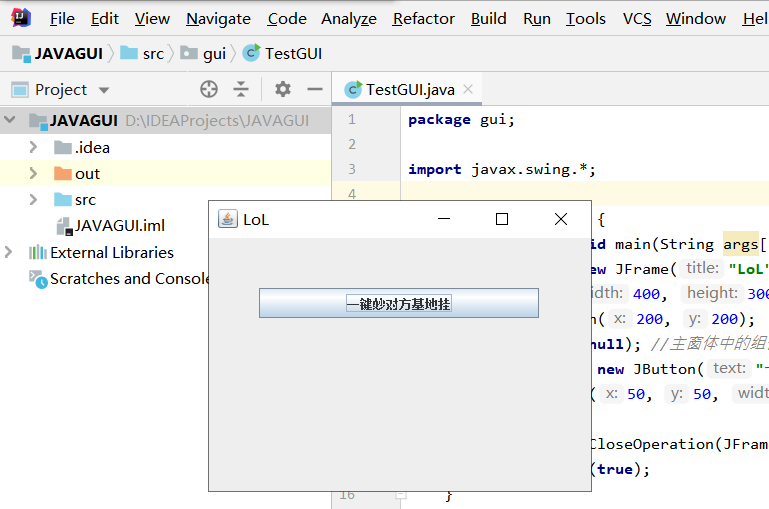
1. Hello Swing

JFrame是GUI中的容器

JButton是最常见的组件-按钮

注意：f.setVisible(true); 会对所有的组件进行渲染，所以一定要放在最后面。

**import** javax.swing.\*;  
  
**public class** TestGUI {  
 **public static void** main(String args[]){  
 JFrame f = **new** JFrame(**"LoL"**);  
 f.setSize(400, 300);  
 f.setLocation(200, 200);  
 f.setLayout(**null**); *//主窗体中的组件设置为绝对定位* JButton bt = **new** JButton(**"一键妙对方基地挂"**);  
 bt.setBounds(50, 50, 280, 30);  
 f.add(bt);  
 f.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);  
 f.setVisible(**true**);  
 }  
}



练习：在上次关闭位置启动窗口

比如这次使用这个窗口，导致窗口被移动到了右下角。关闭这个窗口，下一次再启动的时候，就会自动出现在右下角。  
****思路提示：****  
启动一个线程，每隔100毫秒读取当前的位置信息，保存在文件中，比如location.txt文件。  
启动的时候，从这个文件中读取位置信息，如果是空的，就使用默认位置，如果不是空的，就把位置信息设置在窗口上。   
读取位置信息的办法： f.getX() 读取横坐标信息，f.getY()读取纵坐标信息。  
**package** gui;  
  
**import** javax.swing.\*;  
**import** java.io.\*;  
*//SavingPositionThread.java***public class** SavingPositionThread **extends** Thread {  
 **private** JFrame **f**;  
 File **file** = **new** File(**"D:/IDEAProjects/JAVAGUI/location.txt"**);  
 SavingPositionThread(JFrame f){  
 **this**.**f** = f;  
 }  
 **public void** run(){  
 **while**(**true**){  
 **int** x = **f**.getX();  
 **int** y = **f**.getY();  
  
 **try**(FileOutputStream fos = **new** FileOutputStream(**file**);  
 DataOutputStream dos = **new** DataOutputStream(fos);  
 ){  
 dos.writeInt(x);  
 dos.writeInt(y);  
 }**catch** (Exception e){  
 e.printStackTrace();  
 }  
  
 **try**{  
 Thread.*sleep*(100);  
 }**catch** (InterruptedException e){  
 e.printStackTrace();  
 }  
 }  
 }  
}

**package** gui;  
  
**import** javax.swing.\*;  
**import** java.io.\*;  
*//TestGUI***public class** TestGUI {  
 **public static void** main(String args[]){  
 JFrame f = **new** JFrame(**"LoL"**);  
 f.setSize(400, 300);  
 Point p = *getPointFromLocationFile*();  
 **if**(p != **null**)  
 f.setLocation(p.**x**, p.**y**);  
 **else** f.setLocation(200, 200);  
 f.setLayout(**null**);  
 JButton b = **new** JButton(**"一键秒掉挂"**);  
 b.setBounds(50, 50, 280, 30);  
 f.add(b);  
 f.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);  
 f.setVisible(**true**);  
  
 **new** SavingPositionThread(f).start();  
 }  
 **static class** Point{  
 **int x**;  
 **int y**;  
 }  
 **public static** Point getPointFromLocationFile(){  
 File file = **new** File(**"D:/IDEAProjects/JAVAGUI/location.txt"**);  
 Point p = **null**;  
 **try**(FileInputStream fis = **new** FileInputStream(file); DataInputStream dis = **new** DataInputStream(fis);){  
 **int** x = dis.readInt();  
 **int** y = dis.readInt();  
 p = **new** Point();  
 p.**x** = x;  
 p.**y** = y;  
 }**catch** (FileNotFoundException e){  
 *//第一次运行,并没有生成位置文件,所以会出现FileNotFoundException* }**catch** (IOException e){  
 e.printStackTrace();  
 }  
 **return** p;  
 }  
}

1. 事件监听
2. 按钮监听 ActionListener

创建一个匿名类实现ActionListener接口，当按钮被点击时，actionPerformed方法就会被调用。

**public class** TestGUI {  
 **public static void** main(String[] args){  
 JFrame f = **new** JFrame(**"LoL"**);  
 f.setSize(400, 300);  
 f.setLocation(580, 200);  
 f.setLayout(**null**);  
 **final** JLabel l = **new** JLabel();  
 ImageIcon i = **new** ImageIcon(**"D:/IDEAProjects/JAVAGUI/shana.png"**);  
 l.setIcon(i);  
 l.setBounds(50, 50, i.getIconWidth(), i.getIconHeight());  
 JButton bt = **new** JButton(**"隐藏图片"**);  
 bt.setBounds(150, 200, 100, 30);  
 bt.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 l.setVisible(**false**);  
 }  
 });  
 f.add(l);  
 f.add(bt);  
 f.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);  
 f.setVisible(**true**);  
 }  
}

1. 键盘监听 KeyListener

keyPressed 代表 键被按下

keyReleased 代表 键被弹起

keyTyped 代表 一个按下弹起的组合动作

KeyEvent.getKeyCode() 可以获取当前点下了哪一个键

**package** gui;  
*//TestGUI***import** javax.swing.\*;  
**import** java.awt.event.KeyEvent;  
**import** java.awt.event.KeyListener;  
  
**public class** TestGUI {  
 **public static void** main(String[] args){  
 JFrame f = **new** JFrame(**"LoL"**);  
 f.setSize(400, 300);  
 f.setLocation(580, 200);  
 f.setLayout(**null**);  
  
 **final** JLabel l = **new** JLabel();  
 ImageIcon i = **new** ImageIcon(**"D:/IDEAProjects/JAVAGUI/shana.png"**);  
 l.setIcon(i);  
 l.setBounds(50, 50, i.getIconWidth(), i.getIconHeight());  
  
 f.addKeyListener(**new** KeyListener() {  
 @Override  
 **public void** keyTyped(KeyEvent e) {  
  
 }  
 @Override  
 **public void** keyPressed(KeyEvent e) {  
  
 }  
 @Override  
 **public void** keyReleased(KeyEvent e) {  
 System.***out***.println(e.getKeyCode()); *//获取到按键对应的KeyCode* **if**(e.getKeyCode() == 39){  
 l.setLocation(l.getX() + 10, l.getY());  
 }  
 }  
 });  
 f.add(l);  
 f.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);  
 f.setVisible(**true**);  
 }  
}

1. 鼠标监听 MouseListener

MouseListener 鼠标监听器

mouseReleased 鼠标释放

mousePressed 鼠标按下

mouseExited 鼠标退出

mouseEntered 鼠标进入

mouseClicked 鼠标点击

本例中，使用mouseEntered,当鼠标进入图片的时候，图片就移动位置。

**public class** TestGUI {  
 **public static void** main(String[] args) {  
 **final** JFrame f = **new** JFrame(**"LoL"**);  
 f.setSize(800, 600);  
 f.setLocationRelativeTo(**null**);  
 f.setLayout(**null**);  
  
 **final** JLabel l = **new** JLabel();  
 ImageIcon i = **new** ImageIcon(**"D:/IDEAProjects/JAVAGUI/shana\_heiheihei.png"**);  
 l.setIcon(i);  
 l.setBounds(375, 275, i.getIconWidth(), i.getIconHeight());  
  
 f.add(l);  
  
 l.addMouseListener(**new** MouseListener() {  
 *// 释放鼠标* **public void** mouseReleased(MouseEvent e) {  
 *//* ***TODO Auto-generated method stub*** }  
 *// 按下鼠标* **public void** mousePressed(MouseEvent e) {  
 *//* ***TODO Auto-generated method stub*** }  
 *// 鼠标退出* **public void** mouseExited(MouseEvent e) {  
 *//* ***TODO Auto-generated method stub*** }  
 *// 鼠标进入* **public void** mouseEntered(MouseEvent e) {  
 Random r = **new** Random();  
 **int** x = r.nextInt(f.getWidth() - l.getWidth());  
 **int** y = r.nextInt(f.getHeight() - l.getHeight());  
 l.setLocation(x, y);  
 }  
 *// 按下释放组合动作为点击鼠标* **public void** mouseClicked(MouseEvent e) {  
 *//* ***TODO Auto-generated method stub*** }  
 });  
 f.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);  
 f.setVisible(**true**);  
 }  
}

1. 适配器 Adapter

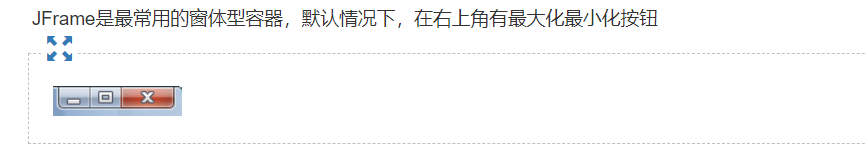
****MouseAdapter****鼠标监听适配器  
一般说来在写监听器的时候，会实现MouseListener。  
但是MouseListener里面有很多方法实际上都没有用到，比如mouseReleased ，mousePressed，mouseExited等等。  
这个时候就可以使用 鼠标监听适配器，MouseAdapter ****只需要重写必要的方法即可****。

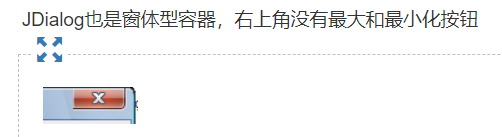
**public class** TestGUI {  
 **public static void** main(String[] args) {  
 **final** JFrame f = **new** JFrame(**"LoL"**);  
 f.setSize(800, 600);  
 f.setLocationRelativeTo(**null**);  
 f.setLayout(**null**);  
  
 **final** JLabel l = **new** JLabel();  
 ImageIcon i = **new** ImageIcon(**"D:/IDEAProjects/JAVAGUI/shana\_heiheihei.png"**);  
 l.setIcon(i);  
 l.setBounds(375, 275, i.getIconWidth(), i.getIconHeight());  
  
 f.add(l);  
  
 l.addMouseListener(**new** MouseAdapter() {  
 @Override  
 **public void** mouseEntered(MouseEvent e) {  
 Random r = **new** Random();  
 **int** x = r.nextInt(f.getWidth() - l.getWidth());  
 **int** y = r.nextInt(f.getHeight() - l.getHeight());  
 l.setLocation(x, y);  
 }  
 });  
 f.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);  
 f.setVisible(**true**);  
 }  
}

1. 容器

java的图形界面中，容器是用来存放按钮，输入框等组件的。

窗体型容器有两个，一个是JFrame,一个是JDialog





**模态JDialog：**当一个对话框被设置为模态的时候，其背后的父窗体，是不能被激活的，除非该对话框被关闭

**public class** TestGUI {  
 **public static void** main(String[] args) {  
 JFrame f = **new** JFrame(**"外部窗体"**);  
 f.setSize(800, 600);  
 f.setLocation(100, 100);  
 JDialog d = **new** JDialog(f); *//根据外部窗体实例化JDialog* d.setModal(**true**); *//设置为模态* d.setTitle(**"模态的对话框"**);  
 d.setSize(400, 300);  
 d.setLocation(200, 200);  
 d.setLayout(**null**);  
 JButton b = **new** JButton(**"Button"**);  
 b.setBounds(50, 50, 280, 30);  
 d.add(b);  
 f.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);  
 f.setVisible(**true**);  
 d.setVisible(**true**);  
 }  
}

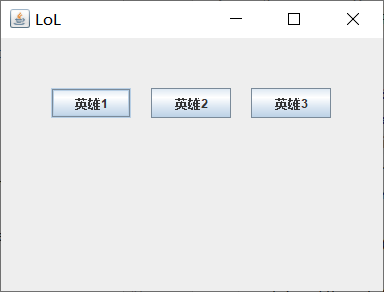
窗口大小不可变化：通过调用方法 setResizable(false); 做到窗体大小不可变化

1. 布局器

布局器是用在容器上的，用来决定容器上的组件摆放的位置和大小。

1. 绝对定位：指不使用布局器，组件的位置和大小需要单独指定

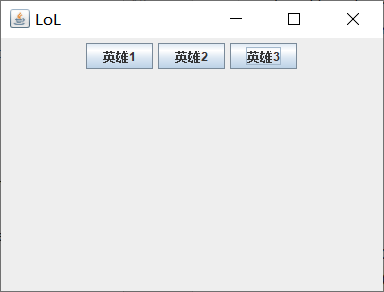
**public class** TestGUI {  
 **public static void** main(String[] args) {  
 JFrame f = **new** JFrame(**"LoL"**);  
 f.setSize(400, 300);  
 f.setLocation(200, 200);  
 f.setLayout(**null**); *//设置布局器为null,即进行绝对定位,容器上的组件都需要指定位置和大小.* JButton b1 = **new** JButton(**"英雄1"**);  
 b1.setBounds(50, 50, 80, 30);  
 JButton b2 = **new** JButton(**"英雄2"**);  
 b2.setBounds(150, 50, 80, 30);  
 JButton b3 = **new** JButton(**"英雄3"**);  
 b3.setBounds(250, 50, 80, 30);  
 JButton b4 = **new** JButton(**"英雄4"**); *//b4没有指定位置和大小,不会出现在容器上* f.add(b1);  
 f.add(b2);  
 f.add(b3);  
 f.add(b4);  
 f.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);  
 f.setVisible(**true**);  
 }  
}



1. FlowLayout 顺序布局器

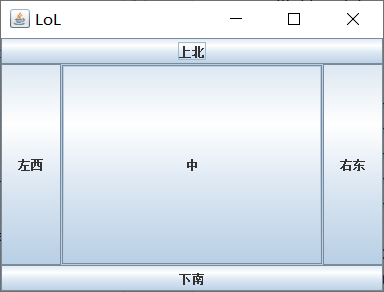
容器上的组件水平摆放，加入到容器即可，无需单独指定大小和位置。

**public class** TestGUI {  
 **public static void** main(String[] args) {  
 JFrame f = **new** JFrame(**"LoL"**);  
 f.setSize(400, 300);  
 f.setLocation(200, 200);  
 f.setLayout(**new** FlowLayout());  
  
 JButton b1 = **new** JButton(**"英雄1"**);  
 JButton b2 = **new** JButton(**"英雄2"**);  
 JButton b3 = **new** JButton(**"英雄3"**);  
  
 f.add(b1);  
 f.add(b2);  
 f.add(b3);  
 f.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);  
 f.setVisible(**true**);  
 }  
}



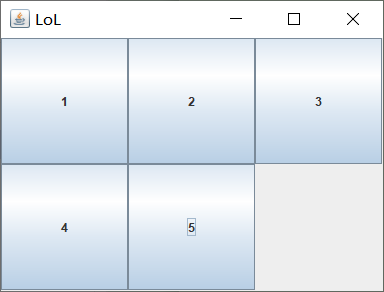
1. BorderLayout

**public class** TestGUI {  
 **public static void** main(String[] args) {  
 JFrame f = **new** JFrame(**"LoL"**);  
 f.setSize(400, 300);  
 f.setLocation(200, 200);  
 f.setLayout(**new** BorderLayout());  
  
 JButton b1 = **new** JButton(**"上北"**);  
 JButton b2 = **new** JButton(**"下南"**);  
 JButton b3 = **new** JButton(**"左西"**);  
 JButton b4 = **new** JButton(**"右东"**);  
 JButton b5 = **new** JButton(**"中"**);  
   
 f.add(b1, BorderLayout.***NORTH***);  
 f.add(b2, BorderLayout.***SOUTH***);  
 f.add(b3, BorderLayout.***WEST***);  
 f.add(b4, BorderLayout.***EAST***);  
 f.add(b5, BorderLayout.***CENTER***);  
   
 f.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);  
 f.setVisible(**true**);  
 }  
}



1. GridLayout 网格布局器

f.setLayout(**new** GridLayout(2, 3)); *//表示该网络是2行3列*JButton b1 = **new** JButton(**"1"**);  
JButton b2 = **new** JButton(**"2"**);  
JButton b3 = **new** JButton(**"3"**);  
JButton b4 = **new** JButton(**"4"**);  
JButton b5 = **new** JButton(**"5"**);  
  
f.add(b1);  
f.add(b2);  
f.add(b3);  
f.add(b4);  
f.add(b5);



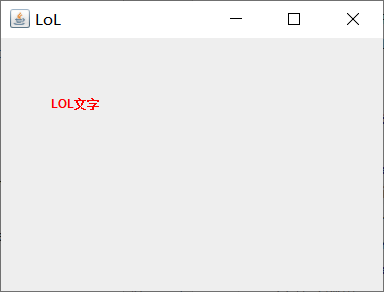
1. CardLayout
2. 组件

JAVA的图形界面下有两组控件，一组是awt，一组是swing。

1. 标签

Label用于显示文字

JLabel l = **new** JLabel(**"LOL文字"**);  
*//文字颜色*l.setForeground(Color.***red***);  
l.setBounds(50, 50, 280, 30);



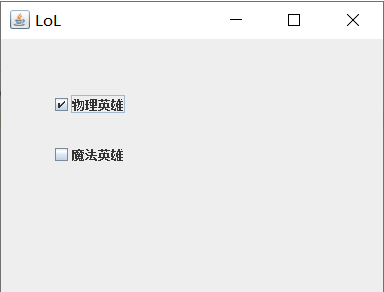
1. 使用JLabel显示图片

JLabel l = **new** JLabel();  
*//根据图片创建ImageIcon对象*ImageIcon i = **new** ImageIcon(**"D:/IDEAProjects/JAVAGUI/shana.png"**);  
*//设置ImageIcon*l.setIcon(i);  
*//label的大小设置为ImageIcon,否则显示不完整*l.setBounds(50, 50, i.getIconWidth(), i.getIconHeight());

1. 按钮 略
2. 复选框

JCheckBox 可以使用isSelected来获取是否选中了。

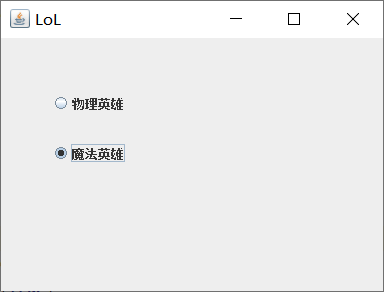
**public class** TestGUI {  
 **public static void** main(String[] args) {  
 JFrame f = **new** JFrame(**"LoL"**);  
 f.setSize(400, 300);  
 f.setLocation(580, 200);  
 f.setLayout(**null**);  
 JCheckBox cb = **new** JCheckBox(**"物理英雄"**);  
 cb.setSelected(**true**); *//设置为默认选中* cb.setBounds(50, 50, 130, 30);  
 JCheckBox cb2 = **new** JCheckBox(**"魔法英雄"**);  
 cb2.setBounds(50, 100, 130, 30);  
 System.***out***.println(cb2.isSelected());  
  
 f.add(cb);  
 f.add(cb2);  
 f.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);  
 f.setVisible(**true**);  
 }  
}



1. 单选框 按钮组

JRadioButton 单选框 可使用isSelected来获取是否选中了

ButtonGroup 对按钮进行分组，把不同的按钮，放在同一个分组里，同一时间，只有一个按钮会被选中。



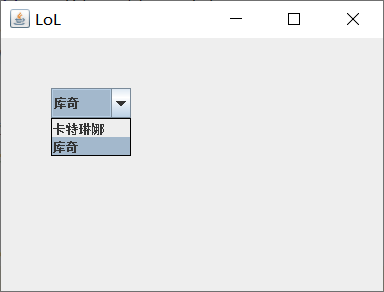
代码：

**public class** TestGUI {  
 **public static void** main(String[] args) {  
 JFrame f = **new** JFrame(**"LoL"**);  
 f.setSize(400, 300);  
 f.setLocation(580, 240);  
 f.setLayout(**null**);  
 JRadioButton b1 = **new** JRadioButton(**"物理英雄"**);  
 b1.setSelected(**true**);  
 b1.setBounds(50, 50, 130, 30);  
 JRadioButton b2 = **new** JRadioButton(**"魔法英雄"**);  
 b2.setBounds(50, 100, 130, 30);  
 *//按钮分组* ButtonGroup bg = **new** ButtonGroup();  
 bg.add(b1);  
 bg.add(b2);  
 f.add(b1);  
 f.add(b2);  
 f.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);  
 f.setVisible(**true**);  
 }  
}

1. 下拉框 JComboBox

使用getSelectedItem来获取被选中项

使用setSelectedItem来指定要选中项



代码：

**public class** TestGUI {  
 **public static void** main(String[] args) {  
 JFrame f = **new** JFrame(**"LoL"**);  
 f.setSize(400, 300);  
 f.setLocation(580, 240);  
 f.setLayout(**null**);  
 *//下拉框出现的条目* String[] heros = **new** String[]{**"卡特琳娜"**, **"库奇"**};  
 JComboBox cb = **new** JComboBox(heros);  
 cb.setBounds(50, 50, 80, 30);  
 f.add(cb);  
 f.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);  
 f.setVisible(**true**);  
 }  
}

1. 对话框 JOptionPane

用于弹出对话框：

JOptionPane.showConfirmDialog(f, “是否 使用外挂?”);

表示询问，第一个参数是该对话框以哪个组件对齐。

JOptionPane.showInputDialog(f, "请输入yes,表明使用外挂后果自负");

接收用户的输入

JOptionPane.showMessageDialog(f, "你使用外挂被抓住!");

显示信息

**public class** TestGUI {  
 **public static void** main(String[] args) {  
 JFrame f = **new** JFrame(**"LoL"**);  
 f.setSize(400, 300);  
 f.setLocation(580, 240);  
 f.setLayout(**null**);  
 f.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);  
 f.setVisible(**true**);  
  
 **int** option = JOptionPane.*showConfirmDialog*(f, **"是否 使用外挂?"**);  
 **if**(JOptionPane.***OK\_OPTION*** == option) {  
 String answer = JOptionPane.*showInputDialog*(f, **"请输入yes,表明使用外挂后果自负"**);  
 **if**(answer.equals(**"yes"**))  
 JOptionPane.*showMessageDialog*(f, **"你使用外挂被抓住!"**);  
 }  
 }  
}

1. 文本框 JTextField

setText 设置文本 getText 获取文本

tfPassword.grabFocus(); 表示让密码输入框获取焦点。

**public class** TestGUI {  
 **public static void** main(String[] args) {  
 JFrame f = **new** JFrame(**"LoL"**);  
 f.setSize(400, 300);  
 f.setLocation(200, 200);  
 f.setLayout(**new** FlowLayout());  
 JLabel lName = **new** JLabel(**"账号："**);  
 *// 输入框* JTextField tfName = **new** JTextField(**"请输入账号"**);  
 tfName.setPreferredSize(**new** Dimension(80, 30));  
 JLabel lPassword = **new** JLabel(**"密码："**);  
 *// 输入框* JTextField tfPassword = **new** JTextField(**""**);  
 tfPassword.setText(**"请输入密码"**);  
 tfPassword.setPreferredSize(**new** Dimension(80, 30));  
  
 f.add(lName);  
 f.add(tfName);  
 f.add(lPassword);  
 f.add(tfPassword);  
 f.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);  
 f.setVisible(**true**);  
 tfPassword.grabFocus();  
 }  
}

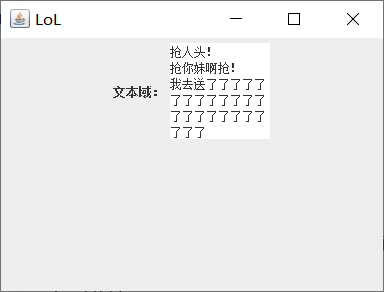
1. 密码框 JPasswordField

与文本框不同，获取密码框里的内容，推荐使用getPassword，该方法会返回一个字符数组，而非字符串

**public class** TestGUI {  
 **public static void** main(String[] args) {  
 JFrame f = **new** JFrame(**"LoL"**);  
 f.setSize(400, 300);  
 f.setLocation(200, 200);  
 f.setLayout(**new** FlowLayout());  
 JLabel l = **new** JLabel(**"密码："**);  
 JPasswordField pf = **new** JPasswordField(**""**);  
 pf.setText(**"&48kdh4@#"**);  
 pf.setPreferredSize(**new** Dimension(80, 30));  
 *//与文本框不同，获取密码框里的内容，推荐使用getPassword，该方法会返回一个字符数组，而非字符串* **char**[] password = pf.getPassword();  
 String p = String.*valueOf*(password);  
 System.***out***.println(p);  
 f.add(l);  
 f.add(pf);  
 f.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);  
 f.setVisible(**true**);  
 }  
}

1. 文本域 JTextArea

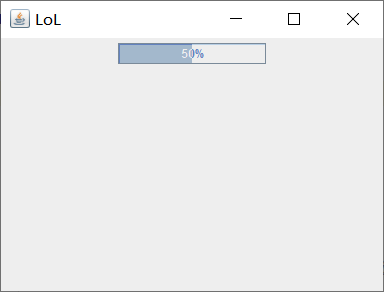
如果要给文本域初始文本，通过\n来实现换行效果  
JTextArea通常会用到append来进行数据追加  
如果文本太长，会跑出去，可以通过setLineWrap(true) 来做到自动换行



代码：

**public class** TestGUI {  
 **public static void** main(String[] args) {  
 JFrame f = **new** JFrame(**"LoL"**);  
 f.setSize(400, 300);  
 f.setLocation(200, 200);  
 f.setLayout(**new** FlowLayout());  
 JLabel l = **new** JLabel(**"文本域："**);  
 JTextArea ta = **new** JTextArea();  
 ta.setText(**"抢人头！\n抢你妹啊抢！\n"**);  
 ta.append(**"我去送了了了了了了了了了了了了了了了了了了了了了了了了"**);  
 ta.setLineWrap(**true**);  
 f.add(l);  
 f.add(ta);  
 f.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);  
 f.setVisible(**true**);  
 }  
}

1. 进度条 JProgressBar



JProgressBar pb = **new** JProgressBar();  
pb.setMaximum(100); *//进度条最大100*pb.setValue(50); *//当前进度是50*pb.setStringPainted(**true**); *//显示当前进度*

1. 文件选择器 JFileChooser

fc.showOpenDialog(); 用于打开文件  
fc.showSaveDialog(); 用于保存文件

**public class** TestGUI {  
 **public static void** main(String[] args) {  
 JFrame f = **new** JFrame(**"LOL"**);  
 f.setLayout(**new** FlowLayout());  
 JFileChooser fc = **new** JFileChooser();  
 fc.setFileFilter(**new** FileFilter() {  
 @Override  
 **public** String getDescription() {  
 *//* ***TODO Auto-generated method stub* return ".txt"**;  
 }  
 @Override  
 **public boolean** accept(File f) {  
 **return** f.getName().toLowerCase().endsWith(**".txt"**);  
 }  
 });  
 JButton bOpen = **new** JButton(**"打开文件"**);  
 JButton bSave = **new** JButton(**"保存文件"**);  
 f.add(bOpen);  
 f.add(bSave);  
 f.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);  
 f.setSize(250, 150);  
 f.setLocationRelativeTo(**null**);  
 f.setVisible(**true**);  
 bOpen.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **int** returnVal = fc.showOpenDialog(f);  
 File file = fc.getSelectedFile();  
 **if** (returnVal == JFileChooser.***APPROVE\_OPTION***) {  
 JOptionPane.*showMessageDialog*(f, **"计划打开文件:"** + file.getAbsolutePath());  
 }  
 }  
 });  
 bSave.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **int** returnVal = fc.showSaveDialog(f);  
 File file = fc.getSelectedFile();  
 **if** (returnVal == JFileChooser.***APPROVE\_OPTION***) {  
 JOptionPane.*showMessageDialog*(f, **"计划保存到文件:"** + file.getAbsolutePath());  
 }  
 }  
 });  
 }  
}

1. 面板

JPanel 基本面板

ContentPane JFrame上的一层面板

SplitPanel

JScrollPanel 带有滚动条的面板

TabbedPanel

CardLayerout

1. 练习：显示文件夹复制进度条

**package** gui;  
**public class** TestGUI {  
  
 **static long** *allFileSize* = 0; *// 所有需要复制的文件大小* **static long** *currentFileSizeCopied* = 0;*// 已复制的文件总大小  
  
 /\*\*  
 \* 遍历文件夹获取文件夹内容总大小  
 \*  
 \** ***@param file*** *\*/* **public static void** calclateAllFilesize(File file) {  
  
 **if** (file.isFile()) {  
 *allFileSize* += file.length();  
 }  
 **if** (file.isDirectory()) {  
 File[] fs = file.listFiles();  
 **for** (File f : fs) {  
 *calclateAllFilesize*(f);  
 }  
 }  
  
 }  
  
 **public static void** main(String[] args) {  
  
 JFrame f = **new** JFrame(**"带进度条的文件夹复制"**);  
 f.setSize(450, 140);  
 f.setLocation(200, 200);  
 f.setLayout(**new** FlowLayout());  
  
 *// 文件地址* JLabel lStr = **new** JLabel(**"源文件地址："**);  
 JTextField strTf = **new** JTextField(**""**);  
 strTf.setText(**"e:/JDK"**);  
 strTf.setPreferredSize(**new** Dimension(100, 30));  
 JLabel lDest = **new** JLabel(**"复制到："**);  
 JTextField destTf = **new** JTextField(**""**);  
 destTf.setText(**"e:/JDK2"**);  
 destTf.setPreferredSize(**new** Dimension(100, 30));  
  
 f.add(lStr);  
 f.add(strTf);  
 f.add(lDest);  
 f.add(destTf);  
  
 JButton bStartCopy = **new** JButton(**"开始复制"**);  
 bStartCopy.setPreferredSize(**new** Dimension(100, 30));  
  
 JLabel l = **new** JLabel(**"文件复制进度："**);  
 JProgressBar pb = **new** JProgressBar();  
 pb.setMaximum(100);  
 pb.setStringPainted(**true**);  
  
 f.add(bStartCopy);  
 f.add(l);  
 f.add(pb);  
 f.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);  
  
 f.setVisible(**true**);  
  
 *// 计算需要复制的文件的总大小* String srcPath = strTf.getText();  
 File folder = **new** File(srcPath);  
 *calclateAllFilesize*(folder);  
  
 *// 点击开始复制* bStartCopy.addActionListener(**new** ActionListener() {  
 **public void** actionPerformed(ActionEvent e) {  
 *currentFileSizeCopied* = 0;  
 String srcPath = strTf.getText();  
 String destPath = destTf.getText();  
 **new** Thread(() -> copyFolder(srcPath, destPath)).start();  
 bStartCopy.setEnabled(**false**);  
 }  
  
 **public void** copyFile(String srcPath, String destPath) {  
  
 File srcFile = **new** File(srcPath);  
 File destFile = **new** File(destPath);  
  
 *// 缓存区，一次性读取1024字节* **byte**[] buffer = **new byte**[1024];  
  
 **try** (FileInputStream fis = **new** FileInputStream(srcFile);  
 FileOutputStream fos = **new** FileOutputStream(destFile);) {  
 **while** (**true**) {  
 *// 实际读取的长度是 actuallyReaded,有可能小于1024* **int** actuallyReaded = fis.read(buffer);  
 *// -1表示没有可读的内容了* **if** (-1 == actuallyReaded)  
 **break**;  
 fos.write(buffer, 0, actuallyReaded);  
 fos.flush();  
 }  
 } **catch** (FileNotFoundException e) {  
 e.printStackTrace();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 **public void** copyFolder(String srcPath, String destPath) {  
 File srcFolder = **new** File(srcPath);  
 File destFolder = **new** File(destPath);  
  
 **if** (!srcFolder.exists())  
 **return**;  
  
 **if** (!srcFolder.isDirectory())  
 **return**;  
  
 **if** (destFolder.isFile())  
 **return**;  
  
 **if** (!destFolder.exists())  
 destFolder.mkdirs();  
  
 File[] files = srcFolder.listFiles();  
 **for** (File srcFile : files) {  
  
 **if** (!(srcFile.isDirectory())) {  
 File newDestFile = **new** File(destFolder, srcFile.getName());  
 copyFile(srcFile.getAbsolutePath(), newDestFile.getAbsolutePath());  
 *currentFileSizeCopied* += srcFile.length();  
  
 **double** current = (**double**) *currentFileSizeCopied* / (**double**) *allFileSize*;  
 **int** progress = (**int**) (current \* 100);  
 pb.setValue(progress);  
 **if** (progress == 100) {  
 JOptionPane.*showMessageDialog*(f, **"复制完毕"**);  
 bStartCopy.setEnabled(**true**);  
 }  
  
 }  
  
 **if** (srcFile.isDirectory()) {  
 File newDestFolder = **new** File(destFolder, srcFile.getName());  
 copyFolder(srcFile.getAbsolutePath(), newDestFolder.getAbsolutePath());  
 }  
  
 }  
 }  
 });  
 }  
}

1. 菜单

菜单栏 JMenuBar

菜单 JMenu

菜单项 JMenuItem

1. 工具栏

工具栏 JToolBar

设置提示信息 setToolTipText(“提示信息”);

1. 表格

**表格 JTable**

显示Table需要两组数据：

1. String[] columnNames 表示表格的标题
2. String[][] heros 表格中的内容

**TableModel**

数据和显示分离开，数据部分，专门做一个类TableModel，就用于存放要显示的数据。

HeroTableModel 继承AbstractTableModel ，进而实现了接口TableModel  
在HeroTableModel 中提供一个table显示需要的所有信息  
1. getRowCount 返回一共有多少行  
2. getColumnCount 返回一共有多少列  
3. getColumnName 每一列的名字  
4. isCellEditable 单元格是否可以修改  
5. getValueAt 每一个单元格里的值

代码：

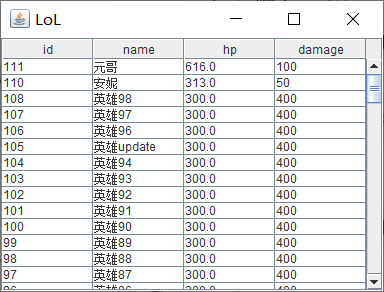
**public class** HeroTableModel **extends** AbstractTableModel {  
 String[] **columnNames** = **new** String[] {**"id"**, **"name"**, **"hp"**, **"damage"**};  
 String[][] **heros** = **new** String[][] { {**"1"**, **"盖伦"**, **"616"**, **"100"**},  
 {**"2"**, **"提莫"**, **"512"**, **"102"**}, {**"3"**, **"奎因"**, **"832"**, **"200"**}};  
  
 @Override  
 **public int** getRowCount() {  
 **return heros**.**length**;  
 }  
  
 @Override  
 **public int** getColumnCount() {  
 **return columnNames**.**length**;  
 }  
  
 **public** String getColumnName(**int** columnIndex){  
 **return columnNames**[columnIndex];  
 }  
 **public boolean** isCellEditable(**int** rowIndex, **int** columnIndex){  
 **return false**;  
 }  
 @Override  
 **public** Object getValueAt(**int** rowIndex, **int** columnIndex) {  
 **return heros**[rowIndex][columnIndex];  
 }  
}

显示：

HeroTableModel htm = **new** HeroTableModel();  
JTable t = **new** JTable(htm);  
JScrollPane sp = **new** JScrollPane(t);  
f.add(sp, BorderLayout.***CENTER***);

**TableModel 与DAO结合**

**package** gui;  
**import** charactor.Hero;  
**import** jdbc.HeroDAO;  
**import** javax.swing.table.AbstractTableModel;  
**import** java.util.List;  
  
**public class** HeroTableModel **extends** AbstractTableModel {  
 String[] **columnNames** = **new** String[] {**"id"**, **"name"**, **"hp"**, **"damage"**};  
 **public** List<Hero> **heros** = **new** HeroDAO().list();  
 @Override  
 **public int** getRowCount() {  
 **return heros**.size();  
 }  
  
 @Override  
 **public int** getColumnCount() {  
 **return columnNames**.**length**;  
 }  
 **public** String getColumnName(**int** columnIndex){  
 **return columnNames**[columnIndex];  
 }  
 **public boolean** isCellEditable(**int** rowIndex, **int** columnIndex){  
 **return false**;  
 }  
 @Override  
 **public** Object getValueAt(**int** rowIndex, **int** columnIndex) {  
 Hero h = **heros**.get(rowIndex);  
 **if** (0 == columnIndex)  
 **return** h.**id**;  
 **if** (1 == columnIndex)  
 **return** h.**name**;  
 **if** (2 == columnIndex)  
 **return** h.**hp**;  
 **if** (3 == columnIndex)  
 **return** h.**damage**;  
 **return null**;  
 }  
}



**TableSelectionModel**

通过table可以获取一个 TableSelectionModel，专门用于监听jtable选中项的变化

代码：

**package** gui;  
  
**import** charactor.Hero;  
  
**import** javax.swing.\*;  
**import** javax.swing.border.Border;  
**import** javax.swing.event.ListSelectionEvent;  
**import** javax.swing.event.ListSelectionListener;  
**import** java.awt.\*;  
  
**public class** TestGUI {  
 **public static void** main(String args[]){  
 JFrame f = **new** JFrame(**"LoL"**);  
 f.setSize(400, 300);  
 f.setLocation(200, 200);  
 f.setLayout(**new** BorderLayout());  
  
 **final** HeroTableModel htm = **new** HeroTableModel();  
 **final** JTable t = **new** JTable(htm);  
  
 JPanel p = **new** JPanel();  
 **final** JLabel l = **new** JLabel(**"暂时未选择条目"**);  
 p.add(l);  
 JScrollPane sp = **new** JScrollPane(t);  
 t.getSelectionModel().addListSelectionListener(  
 **new** ListSelectionListener() {  
 @Override  
 **public void** valueChanged(ListSelectionEvent e) {  
 **int** row = t.getSelectedRow();  
 Hero h = htm.**heros**.get(row);  
 l.setText(**"当前选中的英雄是："** + h.**name**);  
 }  
 });  
 f.add(p, BorderLayout.***NORTH***);  
 f.add(sp, BorderLayout.***CENTER***);  
 f.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);  
 f.setVisible(**true**);  
 }  
}



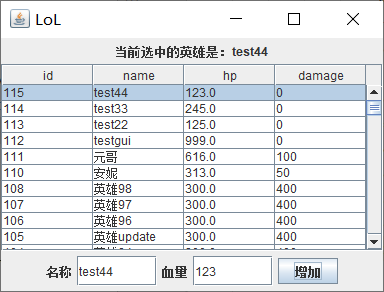
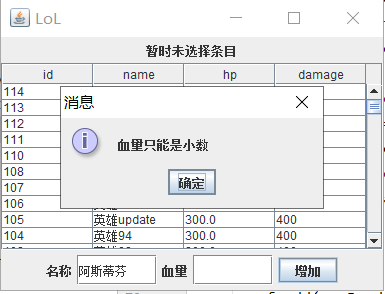
**更新Tabel + 错误提示**

1.以新增数据到数据库中，然后更新Table

2. table初始化后，应该默认选中第一行  
 3. 增加数据后，也应该选中新增的这一条

4. 错误提示

**package** gui;  
  
**import** charactor.Hero;  
**import** jdbc.HeroDAO;  
  
**import** javax.swing.\*;  
**import** javax.swing.border.Border;  
**import** javax.swing.event.ListSelectionEvent;  
**import** javax.swing.event.ListSelectionListener;  
**import** java.awt.\*;  
**import** java.awt.event.ActionEvent;  
**import** java.awt.event.ActionListener;  
  
**public class** TestGUI {  
 **public static void** main(String args[]){  
 JFrame f = **new** JFrame(**"LoL"**);  
 f.setSize(400, 300);  
 f.setLocation(200, 200);  
 f.setLayout(**new** BorderLayout());  
  
 **final** HeroTableModel htm = **new** HeroTableModel();  
 **final** JTable t = **new** JTable(htm);  
 *//准备一个Panel上面放一个Label用于显示哪条被选中了* JPanel p = **new** JPanel();  
 **final** JLabel l = **new** JLabel(**"暂时未选择条目"**);  
 p.add(l);  
 JScrollPane sp = **new** JScrollPane(t);  
 t.getSelectionModel().addListSelectionListener(  
 **new** ListSelectionListener() {  
 @Override  
 **public void** valueChanged(ListSelectionEvent e) {  
 **int** row = t.getSelectedRow();  
 Hero h = htm.**heros**.get(row);  
 l.setText(**"当前选中的英雄是："** + h.**name**);  
 }  
 });  
 *// 增加一个 panel用于放置名称，血量输入框和增加 按钮* JPanel p2 = **new** JPanel();  
 **final** JLabel lName = **new** JLabel(**"名称"**);  
 **final** JTextField tfName = **new** JTextField(**""**);  
 **final** JLabel lHp = **new** JLabel(**"血量"**);  
 **final** JTextField tfHp = **new** JTextField(**""**);  
 JButton bAdd = **new** JButton(**"增加"**);  
 tfName.setPreferredSize(**new** Dimension(80, 30));  
 tfHp.setPreferredSize(**new** Dimension(80,30));  
 p2.add(lName);  
 p2.add(tfName);  
 p2.add(lHp);  
 p2.add(tfHp);  
 p2.add(bAdd);  
 bAdd.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 HeroDAO dao = **new** HeroDAO();  
  
 Hero h = **new** Hero();  
 String name = tfName.getText();  
 **if**(name.length() == 0){  
 JOptionPane.*showMessageDialog*(f, **"名称不能为空"**);  
 tfName.grabFocus();  
 **return**;  
 }  
 String hp = tfHp.getText().trim();  
 **try**{  
 Float.*parseFloat*(hp);  
 }**catch** (NumberFormatException e1){  
 JOptionPane.*showMessageDialog*(f, **"血量只能是小数"**);  
 tfHp.grabFocus();  
 **return**;  
 }  
 h.**name** = name;  
 h.**hp** = Float.*parseFloat*(hp);  
 dao.add(h);  
 htm.**heros** = dao.list();  
 t.updateUI();  
 }  
 });  
  
 f.add(p, BorderLayout.***NORTH***);  
 f.add(sp, BorderLayout.***CENTER***);  
 f.add(p2, BorderLayout.***SOUTH***);  
 f.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);  
 f.setVisible(**true**);  
 }  
}



1. 表格综合练习

**package** gui;  
  
**import** java.awt.BorderLayout;  
**import** java.awt.GridLayout;  
**import** java.awt.event.ActionEvent;  
**import** java.awt.event.ActionListener;  
  
**import** javax.swing.JButton;  
**import** javax.swing.JComboBox;  
**import** javax.swing.JDialog;  
**import** javax.swing.JFrame;  
**import** javax.swing.JLabel;  
**import** javax.swing.JOptionPane;  
**import** javax.swing.JPanel;  
**import** javax.swing.JScrollPane;  
**import** javax.swing.JTable;  
**import** javax.swing.JTextField;  
**import** javax.swing.ListSelectionModel;  
  
**import** charactor.Hero;  
**import** jdbc.HeroDAO;  
  
**public class** TestGUI {  
 **private static void** setLookAndFeel() { *//皮肤* **try** {  
 javax.swing.UIManager.*setLookAndFeel*(**"com.birosoft.liquid.LiquidLookAndFeel"**);  
*// javax.swing.UIManager.setLookAndFeel("com.jtattoo.plaf.smart.SmartLookAndFeel");  
// javax.swing.UIManager.setLookAndFeel("com.jtattoo.plaf.mcwin.McWinLookAndFeel");  
// javax.swing.UIManager.setLookAndFeel("com.jtattoo.plaf.luna.LunaLookAndFeel");  
// javax.swing.UIManager.setLookAndFeel("com.jtattoo.plaf.aluminium.AluminiumLookAndFeel");  
// javax.swing.UIManager.setLookAndFeel("com.jtattoo.plaf.bernstein.BernsteinLookAndFeel");  
// javax.swing.UIManager.setLookAndFeel("com.jtattoo.plaf.hifi.HiFiLookAndFeel");  
// javax.swing.UIManager.setLookAndFeel("com.jtattoo.plaf.mint.MintLookAndFeel");  
// javax.swing.UIManager.setLookAndFeel("com.jtattoo.plaf.aero.AeroLookAndFeel");  
// javax.swing.UIManager.setLookAndFeel("com.jtattoo.plaf.fast.FastLookAndFeel");  
// javax.swing.UIManager.setLookAndFeel("com.jtattoo.plaf.acryl.AcrylLookAndFeel");  
// javax.swing.UIManager.setLookAndFeel("com.jtattoo.plaf.noire.NoireLookAndFeel");* } **catch** (Exception e) {  
 e.printStackTrace();  
 *// handle exception* }  
 }  
  
 **static** HeroTableModel *htm* = **new** HeroTableModel();  
 **static** JTable *t* = **new** JTable(*htm*);  
  
 *// 把分页按钮放在这里，后面监听器好访问* **static** JButton *bFirst* = **new** JButton(**"首页"**);  
 **static** JButton *bPre* = **new** JButton(**"上一页"**);  
 **static** JButton *bNext* = **new** JButton(**"下一页"**);  
 **static** JButton *bLast* = **new** JButton(**"末页"**);  
 **static** JComboBox<Integer> *cb* = **new** JComboBox<>();  
  
 **static int** *number* = 10;*// 每页显示10个* **static int** *start* = 0;*// 开始的页码* **private static boolean** *cbListenerEnabled* = **true**;  
  
 **public static void** main(String[] args) {  
 *//设置皮肤  
 //setLookAndFeel();* **final** JFrame f = **new** JFrame(**"LoL"**);  
 f.setSize(400, 340);  
 f.setLocation(200, 200);  
  
 *t*.setSelectionMode(ListSelectionModel.***SINGLE\_SELECTION***);  
  
 *t*.getSelectionModel().setSelectionInterval(0, 0);  
  
 JPanel pOperation = **new** JPanel();  
  
 JButton bAdd = **new** JButton(**"增加"**);  
 JButton bDelete = **new** JButton(**"删除"**);  
 JButton bEdit = **new** JButton(**"编辑"**);  
 pOperation.add(bAdd);  
 pOperation.add(bDelete);  
 pOperation.add(bEdit);  
  
 JPanel pPage = **new** JPanel();  
  
 pPage.add(*bFirst*);  
 pPage.add(*bPre*);  
 pPage.add(*cb*);  
  
 pPage.add(*bNext*);  
 pPage.add(*bLast*);  
  
 bEdit.addActionListener(**new** ActionListener() {  
  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 *// 判断是否选中* **int** index = *t*.getSelectedRow();  
 **if** (-1 == index) {  
 JOptionPane.*showMessageDialog*(f, **"编辑前需要先选中一行"**);  
 **return**;  
 }  
  
 *// 获取选中的对象* Hero hero = *htm*.**heros**.get(index);  
  
 *// 显示编辑Dialog* EditDialog ed = **new** EditDialog(f);  
 ed.**tfName**.setText(hero.**name**);  
 ed.**tfHp**.setText(String.*valueOf*((**int**) hero.**hp**));  
  
 ed.setVisible(**true**);  
  
 }  
 });  
  
 bAdd.addActionListener(**new** ActionListener() {  
  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **new** AddDialog(f).setVisible(**true**);  
  
 *updateButtonStatus*();  
 }  
 });  
 bDelete.addActionListener(**new** ActionListener() {  
  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
  
 *// 判断是否选中* **int** index = *t*.getSelectedRow();  
 **if** (-1 == index) {  
 JOptionPane.*showMessageDialog*(f, **"删除前需要先选中一行"**);  
 **return**;  
 }  
  
 *// 进行确认是否要删除* **if** (JOptionPane.***OK\_OPTION*** != JOptionPane.*showConfirmDialog*(f, **"确认要删除？"**))  
 **return**;  
  
 *// 获取id* Hero hero = *htm*.**heros**.get(index);  
 **int** id = hero.**id**;  
  
 *// 删除* **new** HeroDAO().delete(id);  
  
 *// 更新table  
 start* = 0;  
 *updateTable*();  
 *updateButtonStatus*();  
  
 }  
 });  
  
 *addPageListener*();  
  
 *cb*.addActionListener(**new** ActionListener() {  
  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **if**(!*cbListenerEnabled*)  
 **return**;  
  
 **int** currentPage = (**int**) *cb*.getSelectedItem();  
 *start* = (currentPage-1)\**number*;  
 *updateTable*();  
 *updateButtonStatus*();  
  
 }  
 });  
  
 JScrollPane sp = **new** JScrollPane(*t*);  
  
 f.setLayout(**null**);  
 sp.setBounds(0, 0, 394, 200);  
 pOperation.setBounds(0, 200, 394, 50);  
 pPage.setBounds(0, 250, 394, 200);  
 f.add(sp);  
 f.add(pOperation);  
 f.add(pPage);  
 *updateButtonStatus*();  
  
 f.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);  
  
 f.setVisible(**true**);  
 }  
  
 **private static void** addPageListener() {  
  
 *bFirst*.addActionListener(**new** ActionListener() {  
  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 *start* = 0;  
 *updateTable*();  
 *updateButtonStatus*();  
 }  
 });  
 *bPre*.addActionListener(**new** ActionListener() {  
  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 *start* -= *number*;  
 *updateTable*();  
 *updateButtonStatus*();  
 }  
 });  
 *bNext*.addActionListener(**new** ActionListener() {  
  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 *start* += *number*;  
 *updateTable*();  
 *updateButtonStatus*();  
 }  
 });  
 *bLast*.addActionListener(**new** ActionListener() {  
  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
  
 *start* = *last*();  
  
 *updateTable*();  
  
 *updateButtonStatus*();  
  
 }  
  
 });  
  
 }  
  
 **private static void** updateButtonStatus() {  
 **int** last = *last*();  
  
 *// 是否有上一页* **if** (0 != *start*) {  
 *bFirst*.setEnabled(**true**);  
 *bPre*.setEnabled(**true**);  
 }  
  
 *// 是否是第一页* **if** (0 == *start*) {  
 *bFirst*.setEnabled(**false**);  
 *bPre*.setEnabled(**false**);  
 }  
  
 *// 是否是最后一页* **if** (*start* == last) {  
 *bLast*.setEnabled(**false**);  
 *bNext*.setEnabled(**false**);  
 }  
 *// 是否有下一页* **if** (*start* < last) {  
 *bLast*.setEnabled(**true**);  
 *bNext*.setEnabled(**true**);  
 }  
  
 *//总共的页数* **int** pageNumber =last/*number*+1;  
 *cbListenerEnabled* = **false**;  
 *cb*.removeAllItems();  
  
 **for** (**int** i = 0; i < pageNumber; i++) {  
  
 *cb*.addItem(i+1);  
 }  
 *cbListenerEnabled* = **true**;  
  
 **int** currentPage = *start*/*number* +1;  
 *cb*.setSelectedItem(currentPage);  
  
 }  
  
 **static class** AddDialog **extends** JDialog {  
 JLabel **lName** = **new** JLabel(**"名称"**);  
 JLabel **lHp** = **new** JLabel(**"血量"**);  
  
 JTextField **tfName** = **new** JTextField();  
 JTextField **tfHp** = **new** JTextField();  
  
 JButton **bSubmit** = **new** JButton(**"提交"**);  
  
 AddDialog(JFrame f) {  
 **super**(f);  
 **this**.setModal(**true**);  
 **int** gap = 50;  
 **this**.setLayout(**null**);  
  
 JPanel pInput = **new** JPanel();  
 JPanel pSubmit = **new** JPanel();  
  
 pInput.setLayout(**new** GridLayout(2, 2, gap, gap));  
 pInput.add(**lName**);  
 pInput.add(**tfName**);  
 pInput.add(**lHp**);  
 pInput.add(**tfHp**);  
  
 pSubmit.add(**bSubmit**);  
  
 pInput.setBounds(50, 20, 200, 100);  
 pSubmit.setBounds(0, 130, 300, 150);  
  
 **this**.add(pInput);  
 **this**.add(pSubmit);  
  
 **this**.setSize(300, 200);  
 **this**.setLocationRelativeTo(f);  
 **bSubmit**.addActionListener(**new** ActionListener() {  
  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
  
 **if** (*checkEmpty*(**tfName**, **"名称"**)) {  
 **if** (*checkNumber*(**tfHp**, **"hp"**)) {  
  
 String name = **tfName**.getText();  
 **int** hp = Integer.*parseInt*(**tfHp**.getText());  
  
 Hero h = **new** Hero();  
 h.**name** = name;  
 h.**hp** = hp;  
  
 **new** HeroDAO().add(h);  
  
 JOptionPane.*showMessageDialog*(f, **"提交成功 "**);  
  
 AddDialog.**this**.setVisible(**false**);  
 *start* = 0;  
 *updateTable*();  
 }  
 }  
  
 }  
 });  
  
 }  
 }  
  
 **public static void** updateTable() {  
 *htm*.**heros** = **new** HeroDAO().list(*start*, *number*);  
 *t*.updateUI();  
 **if** (!*htm*.**heros**.isEmpty())  
 *t*.getSelectionModel().setSelectionInterval(0, 0);  
 }  
  
 **private static boolean** checkEmpty(JTextField tf, String msg) {  
 String value = tf.getText();  
 **if** (0 == value.length()) {  
 JOptionPane.*showMessageDialog*(**null**, msg + **" 不能为空"**);  
 tf.grabFocus();  
 **return false**;  
 }  
 **return true**;  
 }  
  
 **private static boolean** checkNumber(JTextField tf, String msg) {  
 String value = tf.getText();  
 **if** (0 == value.length()) {  
 JOptionPane.*showMessageDialog*(**null**, msg + **" 不能为空"**);  
 tf.grabFocus();  
 **return false**;  
 }  
 **try** {  
 Integer.*parseInt*(value);  
 } **catch** (NumberFormatException e) {  
 JOptionPane.*showMessageDialog*(**null**, msg + **" 只能是整数"**);  
 tf.grabFocus();  
 **return false**;  
 }  
  
 **return true**;  
 }  
  
 **static class** EditDialog **extends** JDialog {  
 JLabel **lName** = **new** JLabel(**"名称"**);  
 JLabel **lHp** = **new** JLabel(**"血量"**);  
  
 JTextField **tfName** = **new** JTextField();  
 JTextField **tfHp** = **new** JTextField();  
  
 JButton **bSubmit** = **new** JButton(**"提交"**);  
  
 EditDialog(JFrame f) {  
 **super**(f);  
 **this**.setModal(**true**);  
 **int** gap = 50;  
 **this**.setLayout(**null**);  
  
 JPanel pInput = **new** JPanel();  
 JPanel pSubmit = **new** JPanel();  
  
 pInput.setLayout(**new** GridLayout(2, 2, gap, gap));  
 pInput.add(**lName**);  
 pInput.add(**tfName**);  
 pInput.add(**lHp**);  
 pInput.add(**tfHp**);  
  
 pSubmit.add(**bSubmit**);  
  
 pInput.setBounds(50, 20, 200, 100);  
 pSubmit.setBounds(0, 130, 300, 150);  
  
 **this**.add(pInput);  
 **this**.add(pSubmit);  
  
 **this**.setSize(300, 200);  
 **this**.setLocationRelativeTo(f);  
  
 **bSubmit**.addActionListener(**new** ActionListener() {  
  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 **if** (*checkEmpty*(**tfName**, **"名称"**)) {  
 **if** (*checkNumber*(**tfHp**, **"hp"**)) {  
  
 *// 获取id* **int** index = *t*.getSelectedRow();  
 **int** id = *htm*.**heros**.get(index).**id**;  
  
 String name = **tfName**.getText();  
 **int** hp = Integer.*parseInt*(**tfHp**.getText());  
  
 Hero h = **new** Hero();  
 h.**name** = name;  
 h.**hp** = hp;  
 h.**id** = id;  
  
 **new** HeroDAO().update(h);  
  
 JOptionPane.*showMessageDialog*(f, **"提交成功 "**);  
  
 EditDialog.**this**.setVisible(**false**);  
 *updateTable*();  
 }  
 }  
  
 }  
 });  
 }  
 }  
  
 **private static int** last() {  
 *// 最后一页开始的位置* **int** last;  
  
 **int** total = **new** HeroDAO().getTotal();  
  
 *// 最后一页要看总是是否能够整除每页显示的数量number* **if** (0 == total % *number*) {  
 *// 假设总数是20，那么最后一页开始的位置就是10* last = total - *number*;  
 } **else** {  
 *// 假设总数是21，那么最后一页开始的位置就是20* last = total - total % *number*;  
 }  
 **return** last;  
  
 }  
}



使用皮肤后：将第60行的注释去掉 setLookAndFeel();

