1、什么是接口？试比较接口与抽象类的异同

答：接口规定了一组功能的对外调用的规范和协议，但并没有实现这组功能；

与抽象类的区别是：接口中包含的一组方法全是抽象的，而抽象类中包含的方法可以使具体的，也可以是抽象的。

**2、JAVA语言如何进行异常处理，关键字：throws,throw分别代表什么意义？**

**答：**Java通过面向对象的方法进行异常处理，把各种不同的异常进行分类，并提供了良好的接口。在Java中，每个异常都是一个对象，它是Throwable类或其它子类的实例。当一个方法出现异常后便抛出一个异常对象，该对象中包含有异常信息，调用这个对象的方法可以捕获到这个异常并进行处理。用try来执行一段程序，如果出现异常，系统会抛出（throws）一个异常，这时候你可以通过它的类型来捕捉（catch）它，或最后（finally）由缺省处理器来处理。  
用try来指定一块预防所有"异常"的程序。紧跟在try程序后面，应包含一个catch子句来指定你想要捕捉的"异常"的类型。  
throw语句用来明确地抛出一个"异常"。  
throws用来标明一个成员函数可能抛出的各种"异常"。

3、什么叫线程，Java中有几种方法可以实现一个线程？

答：线程是程序中一个单一的顺序控制流程。有两种实现方法，分别是继承Thread类与实现Runnable接口

4、启动一个线程是用run()还是start()?  
答：启动一个线程是调用start()方法，使线程所代表的虚拟处理机处于可运行状态，这意味着它可以由JVM调度并执行。这并不意味着线程就会立即运行。run()方法可以产生必须退出的标志来停止一个线程。

 5请简述Java中关键字super的几种用法。

（1）作为构造方法使用。子类构造方法中调用父类构造方法时使用，且必须放在构造方法的第一行

（2）作为作用域使用。子类成员与父类成员同名并发生冲突时，使用super.成员表示父类成员。可放在构造方法种使用，也可放在成员方法中使用，可以放在任意行。

6构造方法重载的特点是什么？有什么作用？

答：构造方法的重载的特点：方法名与类名相同，公有，无返回值类型，作用用于完成对象的初始化；

重载是定义一组同名的构造方法，在形式参数的个数、类型上有区别，用于创建对象时传递给对象的参数可以多样化；

7简述Java线程的生命周期

答：Java中一个线程完整的生命周期通常包括五个状态：创建状态（Born）、就绪状态（Ready）、运行状态（Running）、阻塞状态（Blocked，Waiting，Sleeping）和死亡状态（Dead）。线程在各个状态间转换构成了线程的生命周期。

8.请写出Java GUI中三种常用容器，以及它们的默认布局。

（1）JFrame（窗口或框架） 默认布局：BorderLayout

（2）JPanel（面板） 默认布局：FlowLayout

（3）JDialog（对话框） 默认布局：Borderlayout

9．请简述Java中包（package）的作用。

包是类和接口的集合。具有以下几个方面作用：

（1）划分名字空间。包是一种名字空间机制，同一包中的类（包括接口）名不能重名，不同包中的类名可以重名。解决了类和接口命名冲突问题。

（2）控制类之间的访问。包是一个访问域，对包中的类有保护作用。例如，一个类的访问控制符是public，则该类不仅可供同一包中的类访问，而且也可以被其他包中的类访问。如果类声明无修饰符，则该类仅供同一包中的类访问。

（3）有助于划分和组织Java应用中的各个类。

书本中P60页，例 2-29，掌握有关数组的定义并且分配

1. int a[]=new int[10];

数组元素的输入

Scanner s=new Scanner(System.in);

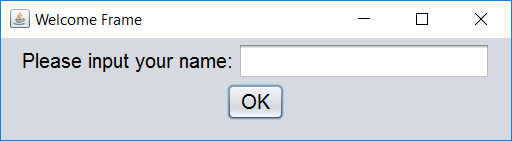
int i;

for (i=0;i<a.length;i++)

a[i]=s.nextInt();

选择排序中比较的趟数及每趟比较的次数，一定要学会；

2、下面的程序创建了一个窗体，如下图所示。阅读程序并补全关键语句。



import java.awt.\*;

import （6） javax.swing.\* ;//导入swing组件所在包中的所有类

public class WelcomeFrameDemo extends JFrame { //自定义一个窗体类

private JLabel lblname;

private JTextField tfname;

private JButton btnOK;

public WelcomeFrameDemo(){

Super ("Welcome Frame"); //访问父类的构造方法

setLayout(new FlowLayout());

lblname = new JLabel("Please input your name:");

lblname.setFont(new Font("Arial",Font.PLAIN,20));

getContentPane(). add (lblname); //添加组件lblname到内容面板中

tfname = new JTextField(15);

tfname.setFont(new Font("Arial",Font.PLAIN,20));

getContentPane(). add (tfname); //添加组件tfname到内容面板中

btnOK = new JButton("OK");

btnOK.setFont(new Font("Arial",Font.PLAIN,20));

getContentPane(). add (btnOK); //添加组件btnOK到内容面板中

}

}

public class WelcomeFrameTest {

public static void main(String[] args){

////创建WelcomeFrameDemo类的对象，名为demo

WelcomeFrameDemo demo=new WelcomeFrameDemo();

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

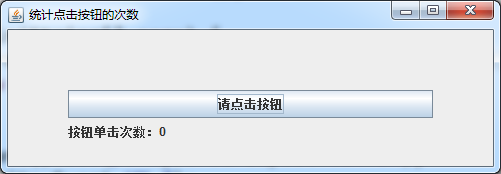
demo.setSize(600,300);

demo.setVisible(true);

}

}

3、下列程序实现的功能是：在JFrame中设置按钮并命名为“请点击按钮”，按钮的下方，由label输出点击按钮的次数。运行界面如下图所示，请完成有关填空。



import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

public class Main {

private static int numclicks = 0; //记录次数

private static JFrame frame = new JFrame("统计点击按钮的次数");

private static \_\_JLabel \_label\_\_\_ =new JLabel();成显示点击次数的标签；

private static JButton button =new JButton(“请点击按钮”);//生成按钮

private static JPanel pane = new JPanel(); //容器

public static void main(String[] args) {

pane.setBorder(BorderFactory.createEmptyBorder(60,60,20,60));

pane.setLayout(new GridLayout(0,1)); //设置布局

pane.add(button);

pane.add(label);

frame.getContentPane().add(pane,BorderLayout.CENTER);

button.setMnemonic(KeyEvent.VK\_I);

button.addActionListener(new ActionListener(){ //鼠标单击事件

public void \_actionPerformed\_\_(ActionEvent e){//实现ActionListener中的抽象方法；

\_\_numclicks++;\_\_//实现点击次数的自加

label.setText("按钮单击次数："+numclicks);

}

});

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

frame.pack();

frame. setVisible(true)\_\_\_\_;//调用函数使JFrame可视；

}

}

例如：

窗口中有三个组件，依次是：标签(JLabel)、文本框(JTextField)、按钮(JButton)。

现在在文本框中输入字符，单击“确定”按钮后，标签就显示为文本框中的内容。

如图1所示，点击“确定”按钮后，如图2所示

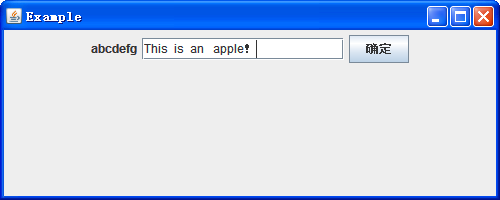


图1 初始状态

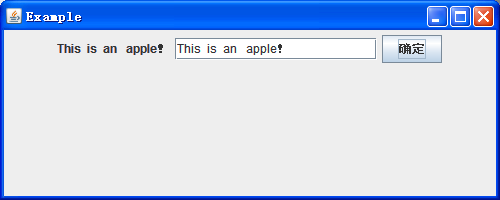


图2 点击按钮后

方法一：在主类中创建JFrame类对象引用，再创建各组件并添加到窗口

**import** javax.swing.\*; //JFrame\JLabel\JTextField\JButton 类所在包

**import** java.awt.\*; // 常用布局类所在包

**import** java.awt.event.\*; //事件处理

**public** **class** TestJFrame {

**private** **static** JFrame *myframe*=**new** JFrame(); //创建窗口类对象，作为该类成员变量，注意是static

**private** **static** JLabel *mylabel*=**new** JLabel("abcdefg"); //创建标签类对象，作为该类成员变量，注意是static

**private** **static** JTextField *mytxt*=**new** JTextField(18); //创建文本框类对象，作为该类成员变量，注意是static

**private** **static** JButton *mybtn*=**new** JButton("确定"); //创建按钮类对象，作为该类成员变量，注意是static

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

*myframe*.setTitle("Example"); //为窗口设置标题

*myframe*.setLayout(**new** FlowLayout()); //为窗口设置布局，这里采用的是流式布局

*myframe*.add(*mylabel*); // 为窗口添加标签组件

*myframe*.add(*mytxt*); // 为窗口添加文本框组件

*myframe*.add(*mybtn*); // 为窗口添加按钮组件

*mybtn*.addActionListener(**new** ActionListener(){ // 按钮组件作为事件源，添加动作事件监听器，并用匿名内部类作为事件监听器对象

**public** **void** actionPerformed(ActionEvent e) // 动作事件处理方法

{

*mylabel*.setText(*mytxt*.getText());

}

});

*myframe*.setSize(500,200); //为窗口设置大小

*myframe*.setVisible(**true**); //窗口设置为可见

*myframe*.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);

}

}

方法二：创建JFrame类的子类

**import** javax.swing.\*; //JFrame\JLabel\JTextField\JButton 类所在包

**import** java.awt.\*; // 常用布局类所在包

**import** java.awt.event.\*; //事件处理

**class** MyFrame **extends** JFrame **implements** ActionListener //继承了JFrame类，同时实现了ActionListener接口

{

JLabel l;

JTextField tf;

JButton btn;

MyFrame()

{

**super**("Example"); //调用父类构造方法，传递参数，实现设置标题

l=**new** JLabel("abcdefgh");

tf=**new** JTextField(18);

btn=**new** JButton("确定");

add(l); add(tf); add(btn);

btn.addActionListener(**this**); //由于本类实现了ActionListener接口，所以本类对象可以作为监听器对象

setLayout(**new** FlowLayout());

setVisible(**true**);

setBounds(50,50,500,200);

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

}

**public** **void** actionPerformed(ActionEvent e) //重写ActionListener接口中actionPerformed方法，实现事件处理

{

l.setText(tf.getText());

}

}

**public** **class** TestJFrame2 { //测试类

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

**new** MyFrame();

}

}

综合（实验报告二（4））

学生类的创建和使用

①创建一个学Student类，成员变量包括：学号、姓名、性别、年龄等，且都是private类型。

②声明一个构造方法，初始化所有成员变量。

③分别声明获得各属性的public成员方法，方法名要求以get开头。

④分别声明修改各属性的public成员方法，方法名要求以set开头。

⑤声明一个public类型的toString（）方法，把该类中的所有域信息组合成一个字符串。

⑥声明统计创建Student对象的个数的私有域count和得到Student对象的个数的public方法。

主方法测试类功能（2种，任取一种）

编写测试类Test，在主类中，创建2个Student对象，输出对象的所有域信息；修改对象的姓名和年龄，输出修改后的姓名和年龄；比较两个Student对象的年龄的大小，输出年龄较大的Student对象。

在Student类中编写主方法，创建2个Student对象，输出对象的所有域信息；修改对象的姓名和年龄，输出修改后的姓名和年龄；比较两个Student对象的年龄的大小，输出年龄较大的Student对象。

**package** Student;

**public** **class** Student {

**private** String id;

**private** String name;

**private** **int** age;

**private** String gender;

**private** **static** **int** *count*=0;

**public** Student(String id, String name,String gender,**int** age){

**this**.id=id;

**this**.name=name;

**this**.age=age;

**this**.gender=gender;

*count*++;

}

**public** **static** **int** getCount(){

**return** *count*;

}

**public** String getId(){

**return** id;

}

**public** String getName(){

**return** name;

}

**public** String getGender(){

**return** gender;

}

**public** **int** getAge(){

**return** age;

}

**public** **void** setId(String a){

id=a;

}

**public** **void** setName(String c){

name=c;

}

**public** **void** setGender(String d){

gender=d;

}

**public** **void** setAge(**int** e){

age=e;

}

**public** String toString(){

**return** "学号： "+ **this**.id +"\t姓名："

+ **this**.name +"\t性别："+ **this**.gender +"\t年龄："+ **this**.age;

}

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

Student s1=**new** Student("0001","张三","男",16);

Student s2=**new** Student("0002","李四","女",18);

System.*out*.println(s1.toString());

System.*out*.println(s2.toString());

System.*out*.println("创建的学生总数为："+Student.*getCount*());

s1.setName("王五");

s1.setAge(17);

s2.setName("陆陆");

s2.setAge(19);

System.*out*.println(s1.toString());

System.*out*.println(s2.toString());

**if**(s1.getAge()>s2.getAge())

System.*out*.println("年龄较大的人的信息："+s1.toString());

**else**

System.*out*.println("年龄较大的人的信息： "+s2.toString());

}

}

**public** **class** Test {

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

Student s1=**new** Student("0001","张三","男",16);

Student s2=**new** Student("0002","李四","女",18);

System.*out*.println(s1.toString());

System.*out*.println(s2.toString());

System.*out*.println("创建的学生总数为："+Student.*getCount*());

s1.setName("王五");

s1.setAge(17);

s2.setName("陆陆");

s2.setAge(19);

System.*out*.println(s1.toString());

System.*out*.println(s2.toString());

**if**(s1.getAge()>s2.getAge())

System.*out*.println("年龄较大的人的信息："+s1.toString());

**else**

System.*out*.println("年龄较大的人的信息： "+s2.toString());

}

}

有关接口编程：

Interface　接口名{

抽象方法声明;

抽象方法声明；

　　若有常量，常量声明；

}

class 类名一 implements 接口名{

类型名　成员变量名;　　//类的数据声明；

　　类的构造方法定义及实现；

类名(参数表){

成员变量名=参数;

成员变量名=参数;

　　　　......

}

接口中的所有抽象方法的实现

}

class 类名二 implements 接口名{

类型名　成员变量名;　　//类的数据声明；

　　类的构造方法定义及实现；

类名(参数表){

成员变量名=参数;

成员变量名=参数;

　　　　......

}

接口中的所有抽象方法的实现

……

}

public class 主类{ // 主类定义

public static void main(String[] args){　//主方法定义

接口名　 接口引用名;　　　　//利用接口实现多态性，可定义接口引用，当引用指向不同的实现接口的类时，相同的调用形式，则实际的结果呈现多态

接口引用名=new 类一的构造方法(实参);

System.out.println("结果为："+接口引用名.方法名(……));

接口引用名=new 类二的构造方法(实参);

System.out.println("结果为："+接口引用名.方法名(……));

}

}