1. JAVA 语言中有哪些基本数据类型？这些数据类型的数据在进行混

合计算时，应该遵循什么样的原则？为什么？

1. 八大基本数据类型：byte char short int long short float double
2. 原则：系统会根据各种类型的优先级进行操作，先将低优先级的数转换为较高优先级的数，再进行计算，基本数据类型优先级排列：byte->short-char->int->long->float->double.
3. 因为如：byte转换为int类型，int类型的取值范围大于byte类型的取值范围，编译器在赋值过程中不会造成数据丢失，相反，若将double强制转换为int，则会造成一定数据丢失，

2. JAVA 语言中是怎样处理字符串的？试说明 String 类和 StringBuffer

类的特点和区别，以及常用的一些方法。

(1)用变量String 或 StringBuffer 或StringBuilder进行新建和处理一般用new String(“字符串”),String 也可直接用赋值运算符来创建，引用在内存中为字符串开辟空间的位置，所以String用==比较的话是比较引用地址，需要equals才能比较相等否。String处理较少的字符串，StringBuffer和StringBuilder处理较多的字符串，而StringBuffer又在多线程下是安全的，StringBuilder只能处理单线程字符串操作，但StringBuilder处理字符串最快。

(2)

特点:

String类创建一个对象后，在内存块中那字符串是一个地址，String对象只是引用它，如果改变这个对象的值，其实是新开辟一个空间给这个新字符串，然后String对象再引用它，而原本的字符串还存在内存中。

StringBuffer类创建一个对象后，可以直接对字符串操作，一般比String快，并且内部有许多处理字符串的方法，增删改，支持多线程操作是线程安全，有synchronized关键字在内部实现

区别:

String：字符串是常量，创建了修改不了

StringBuffer:字符串是变量，可以增删改，一般情况下运行速度比String快。

(3)常用方法：

String:equals,length,substring,trim,replace,charAt,toChArarray

StringBuffer:append,toString,length,reverse,delete,equals,insert

3. JAVA 语言中怎样定义类？怎样实现类的继承与派生？怎样控制类

的访问权限？如何在子类中使用父类的私有成员？试举例说明。

1. 定义类：class 类名，权限符不写为默认权限
2. 类的继承与派生

Class Animal{…}

Class Tiger extends Animal{…}

1. 可以在定义类时在class前面加上权限修饰符，分别有public protected default private,四种，

使用：public class A{…}即将类A定义为public权限。

1. 可以通过在父类中定义方法调用自身的私有成员，将该方法定义子类可以访问范围，子类调用该方法即可以调用父类私有成员，私有变量与方法皆可

父类：

**public** **class** Animal {

**private** String name;

**private** **void** fly(){

System.***out***.println("I can fly");

}

**protected** **void** getfly() {

fly();

}

**protected** String getName() {

**return** name;

}

**protected** **void** setName(String name) {

**this**.name = name;

}

}

子类：

**public** **class** Triger **extends** Animal{

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

Animal tg = **new** Triger();

tg.getfly();

tg.setName("老虎");

System.***out***.println(tg.getName());

}

}



4. 什么是 JAVA 语言中的重载(OverLoad)和重写(Override)？它们有什

么区别，请举例说明。

1. 重载：发生在同一个类中，同一个方法名有不同的实现方式，通过不同的参数类型，个数来达到不同的实现方式

重写：发生在继承关系的类中，子类继承父类，实现与父类同名的方法时，若new 子类是以父类为基类时，如：父类 st= new 子类（）,用st.调用父类与子类同名的方法，会实现子类的方法，不会调用父类的，这就是重写

重载：

**public** **class** OverloadTest {

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

*book*(1,"一千零一夜");

*book*("活着",2);

*book*("Java从入门到精通");

}

**public** **static** **void** book(**int** id,String name){

System.***out***.println("我是"+name+",我的id是"+id);

}

**public** **static** **void** book(String name,**int** id){

System.***out***.println("我的id是"+id+",我是"+name);

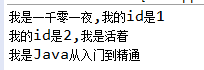
}

**public** **static** **void** book(String name){

System.***out***.println("我是"+name);

}

}



重写：

**public** **class** SuperClass {

**public** **void** print(){

System.***out***.println("Superclass：print()");

}

}

**public** **class** SubClass **extends** SuperClass{

**public** **void** print(){

System.***out***.println("Subclass：print()");

}

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

SuperClass sb = **new** SubClass();

sb.print();

}

}

SubClass类继承父类SuperClass，子类与父类同有print()方法，对象sb的基类是SuperClass，sb调用print方法本该调用父类的print方法，但是因为子类也写了同名的方法，导致重写了父类的方法，其实父类的方法被隐藏了，只需要把子类的print方法给注释掉，即会调用父类的print方法



综上可得，重写隐藏了其它同名函数的实现方式，调用时只实现自己，重载保留同名函数各个的实现方式，通过传入的参数不同调用不同实现方式

5. JAVA 语言中的异常类的作用是什么？它们的主要结构是什么？应

如何使用？抛出（throw）错误是什么意思？它有什么功能？

1. 在程序中可能出现各种错误，如：超出可控范围的因素像执行的文件不存在，在程序运行时可能出现这些错误造成程序非正常退出或关闭，使系统不再执行下去，提前结束，这样会达不到预期效果，所以有异常类来捕获处理程序中可能出现的异常，保证程序有效的运行。
2. 结构：

Try{

//程序代码块

}catch(Exceptiontype1 e){

//对Exceptiontype1的处理

}catch(Exceptiontype2 e){

//对Exceptiontype2的处理

}finally{

//程序块

}

1. 使用：

在代码中，对可能出现错误异常的地方进行try包围，在认为正常代码处结束，用catch捕获可能出现的错误进行处理。如：

**public** **class** ExcptionTest {

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

**try** {

**int** get = 10 / 0;

} **catch** (ArithmeticException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

System.***out***.println("continue");

}

}

此程序中 进行除法，但0是不能作为除数的，所以对其try包围，用catch检测可能出现的错误，若捕获是会这个异常ArithmeticException则会打印错误信息，一般用Exception异常父类来捕获，假使没有这段try-catch异常操作，运行到那行程序会直接报错并停止运行不会运行continue的输出，但加上后可自己处理是否打印错误或者其它操作，并且程序正常运行执行输出continue。

1. throw是主动抛出一个异常，当有这条语句时必须捕获异常，不然throw后面的代码都不会执行，通常在方法内部使用，若不想在本方法内捕获可使用throws在上一级代码捕获并处理。
2. 可以抛出用户自定义的异常，通过新建一个类继承Exception异常父类，在构造方法中传入参数，可以实现自己想输出的信息这是其中一个用法。

6. Socket编程是JAVA语言的一个应用，根据给出的客户端示例代码，

按以下要求实现相应的服务器端的功能并给出主要的实现代码:

完成如下功能（已给出客户端示例代码）：

（1） 客户端用控制台录入3个小于100的正整数（中间用空格分开），

发送到服务器端；

（2） 服务器端如果收到 3 个正整数，判断其是否可组成一个三角形，

如果可以则返回三角形的面积；

（3） 服务器端收到其它输入，返回出错信息；

要求：

（1） 操作可以无限重复，完成一轮输入-计算-返回结果后，客户端和

服务端继续工作；

（2） 完成正确输入时的计算功能（输入 3 4 5，可以正确计算出其面

积）；

1. 建议完成识别非法输入功能（输入 2 2 5，可以报错）；

提示：

1. 服务器端入

PrintWriter out = new PrintWriter(socket.getOutputStream(),true);

2. 客户端读入

BufferedReader line = new BufferedReader(new InputStreamReader(System.in));



附：客户端程序：AreaSpcket.java

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.io.PrintWriter;

import java.net.Socket;

public class AreaSocket {

public static void main(String[] args) {

try {

// 创建客户端 Socket，连接指定服务器(127.0.0.1)和端口(3344)

Socket socket = new Socket("127.0.0.1", 3344);

System.out.println("Connected to server " + socket.getRemoteSocketAddress() +

"\n 请输入三个正整数，用空格隔开");

// 由系统标准输入设备构造 BufferedReader 对象，从命令行读取输入内容

BufferedReader keyboardInput = new BufferedReader(new InputStreamReader(System.in));

// 通过 Socket 对象建立与服务器发送数据的通道，构造 PrintWriter 对象

PrintWriter out = new PrintWriter(socket.getOutputStream(), true);

// 通过 Socket 对象建立从服务器接收数据的通道，构造相应的 BufferedReader 对象

BufferedReader input = new BufferedReader(new InputStreamReader(socket.getInputStream()));

String keyInput;

while (true) {

keyInput = keyboardInput.readLine(); // 从系统标准输入用户输入

// 将用户输入发送到服务器端

out.println(keyInput);

out.flush();

System.out.println("Client input: " + keyInput);

// 接收服务端返回的信息

System.out.println("Server return: " + input.readLine());

if(keyInput.equals("bye")){ // 若从标准输入读入的字符串为 "end"则停止循环

break;

}

}

out.close();

input.close();

socket.close();

} catch (Exception e) {

}

}

}

附：服务器端代码框架,文件：AreaServerSocket.java

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.io.PrintWriter;

import java.net.ServerSocket;

import java.net.Socket;

public class AreaServerSocket {

public static void main(String[] args) {

try {

ServerSocket s\_socket = new ServerSocket(3344); // 建立服务器，监听 3344 端口

while (true) {

System.out.println("等待客户端连接...");

Socket socket = s\_socket.accept();

System.out.println("Client " + socket.getRemoteSocketAddress() + " connected.");

}

System.out.println("Client 输入 bye， 断开连接");

out.println("结束对话，断开连接");

out.flush();

socket.close();

} catch (Exception e) {

System.out.println(e);

}

}

}

代码

客户端：

**public** **class** AreaSocket {

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

**try** {

// 创建客户端 Socket，连接指定服务器(127.0.0.1)和端口(3344)

Socket socket = **new** Socket("127.0.0.1", 3344);

System.***out***.println("Connected to server " + socket.getRemoteSocketAddress() + "\n 请输入三个正整数，用空格隔开");

// 由系统标准输入设备构造 BufferedReader 对象，从命令行读取输入内容

BufferedReader keyboardInput = **new** BufferedReader(**new** InputStreamReader(System.***in***));

// 通过 Socket 对象建立与服务器发送数据的通道，构造 PrintWriter 对象

PrintWriter out = **new** PrintWriter(socket.getOutputStream(), **true**);

// 通过 Socket 对象建立从服务器接收数据的通道，构造相应的 BufferedReader 对象

BufferedReader input = **new** BufferedReader(**new** InputStreamReader(socket.getInputStream()));

String keyInput;

**while** (**true**) {

keyInput = keyboardInput.readLine(); // 从系统标准输入用户输入

// 将用户输入发送到服务器端

out.println(keyInput);

out.flush();//刷新缓存

System.***out***.println("Client input: " + keyInput);

// 接收服务端返回的信息，即 是面积 还是 提示输入数值有误

System.***out***.println("Server return: " + input.readLine());

// 当客户端这边自己说bye则表示结束 断开链接

**if** (keyInput.equals("bye")) { // 若从标准输入读入的字符串为 "end"则停止循环

**break**;

}

}

out.close();

input.close();

socket.close();

} **catch** (Exception e) {

}

}

}

服务端：

**public** **class** AreaServerSocket {

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

**try** {

ServerSocket s\_socket = **new** ServerSocket(3344); // 建立服务器，监听 3344 端口

// 先将实例设为空，以免放在while中造成重复不美观

Socket sockets = **null**;

PrintWriter out = **null**;

BufferedReader line = **null**;

System.***out***.println("等待客户端连接...");

**while** (**true**) {

// 当socket为空时才建立连接，以及实例printfwriter 和 bufferreader对象，以免重复new

**if** (sockets == **null**) {

sockets = s\_socket.accept();

System.***out***.println("Client " + sockets.getRemoteSocketAddress() + " connected.");

// 通过 Socket 对象建立与服务器发送数据的通道，构造 PrintWriter 对象

out = **new** PrintWriter(sockets.getOutputStream(), **true**);

// 通过 Socket 对象建立从服务器接收数据的通道，构造相应的 BufferedReader 对象

line = **new** BufferedReader(**new** InputStreamReader(sockets.getInputStream()));

}

// 这里开始每次读取客户端的输入才会进行 下面的代码，这里相当于一个断点 ，会等待客户端的输入才会继续运行

String s\_data = line.readLine();

// 判断客户端是不是输入bye字符串，如果是则退出整个循环 不用计算三角形面积了，并且结束链接

**if** (s\_data.equals("bye")) {

**break**;// break跳出while

} **else** {

// 用字符串内置方法通过空格分割客户端输入的一行数据的每个数值到字符串数组中，因为输入三个数是用空格隔开的

String s\_dtarry[] = s\_data.split(" ");

// 当这个字符串数组长度没有3个 三个数 自然不能构成三角形即客户端没有输入三个用空格隔开的数，则小提示输入有误并且不执行下面代码 用else分离

**if** (s\_dtarry.length != 3) {

out.println("Client 输入有误，请重新输入");

} **else** {

// 客户端输入3个数后，接下来判断这3个数是否符合要求

// 1.先建立存储客户端输入3个数的数组，因为读取是string类型，需要转换为float才能计算

**float** f\_num[] = **new** **float**[s\_dtarry.length + 1];// 开辟4个空间，最后一个放面积，更简洁

**int** flag = 1;// 控制变量，这3个数是否符合要求，默认为符合

String s\_tr;// 临时变量，用来获取数组中每一个String字符串

// 2.将这3个数从数组中依次取出来依次判断是否为数值，即有没有输入什么 字符 a，b·什么符号

**for** (**int** i = 0; i < s\_dtarry.length; i++) {

s\_tr = s\_dtarry[i];

// 自定义的方法处理是否数值，是就返回true ，不是就false，提示客户端输入有误，并且flag = 0标识不符合要求

**if** (*isNumeric*(s\_tr)) {

f\_num[i] = Integer.*valueOf*(s\_tr);// 如果符合要求，则把数组从string转为float类型 才能计算

} **else** {

out.println("Client 输入有误，请重新输入");

flag = 0;// 有一个数不符合就标识为0

**break**;// 一旦有一个数不符合要求就直接退出当前循环

}

}

// 3.通过上面的判断flag是否符合要求，如果是则进行计算面积，如果不符合要求不执行

**if** (flag == 1) {

// 4.计算三角形面积封装在方法中，传入数值数组，返回flag 标识，是1则代表这3个数可以构成三角形并且将面积返回在数组的第三个下标中

flag = *calcuTrigArea*(f\_num);

**if**(flag == 1){// 通过返回的值判断是否构成三角形 若构成得到值

out.println("三角形面积为：" + f\_num[3]);// 向客户端发送消息

}**else**{

out.println("Client 输入有误，请重新输入");// 表示有误，不能构成三角形

}

}

}

}

}

System.***out***.println("Client 输入 bye， 断开连接");

out.println("结束对话，断开连接");

out.flush();

sockets.close();

s\_socket.close();

} **catch** (Exception e) {

System.***out***.println(e);

}

}

// 计算面积方法

**public** **static** **int** calcuTrigArea(**float** f\_num[]) {

// 1.判断是否构成3角形，只需三角形的定义，两个数加起来是否大于第三个数，需要处理三次，不同数加起来

**float** sum1 = f\_num[0] + f\_num[1];

**float** sum2 = f\_num[0] + f\_num[2];

**float** sum3 = f\_num[1] + f\_num[2];

// 每个数都需要判断一次，才能决定是否能构成三角形

**if** (sum1 > f\_num[2] && sum2 > f\_num[1] && sum3 > f\_num[0]) {

// 通过海伦公式，三角形面积可以通过，三角形一半周长再进行公式 area = sqrt(s\*(s-a)\*(s-b)\*(s-c)); sqrt是java内置的开平方方法

**float** sf = (f\_num[0] + f\_num[1] + f\_num[2]) / 2;// 一半周长

f\_num[3] = (**float**) Math.*sqrt*(sf \* (sf - f\_num[0]) \* (sf - f\_num[1]) \* (sf - f\_num[2]));// 保存在数组下标为3中

**return** 1;// 表示可以计算面积 能构成是三角形

}

**return** 0;//表示不能构成三角形

}

// 判断字符串是否为数值，不能有 其它字母和符号 只能0~9的数字

**public** **static** **boolean** isNumeric(String str) {

// 依次循环每个字符 用java自带的函数判断

**for** (**int** i = 0; i < str.length(); i++) {

**if** (!Character.*isDigit*(str.charAt(i))) {

**return** **false**;

}

}

**return** **true**;

}

}

运行结果：

