# Java主要知识点（部分）

## 填空题

1. java程序用关键字\_\_throw\_\_\_\_\_抛出异常，用关键字\_\_catch\_\_\_\_\_捕获异常。
2. java语言中提供的GUI功能主要由\_java.awt\_\_\_\_\_\_和\_\_java.swing\_\_\_\_\_包中的类和接口实现。
3. 子类必须通过关键字\_\_\_\_super\_\_\_调用父类有参数的构造函数；java中所有类都是\_object\_\_\_\_\_\_类的子类。
4. java中类不支持多重继承，但允许一个类同时实现多个\_\_接口\_\_\_\_\_，来实现多重继承；子类中可以直接访问从超类继承下来的三种访问权限设定的成员，包括public，\_\_protected\_\_\_\_\_和缺省。
5. 在子类的创建中，如果出现与超类中有相同名、同参数及同返回类型的成员方法，则超类中的成员方法被子类中的成员方法所\_\_\_过载\_\_\_\_；在同一个类中创建的具有相同方法名，但是参数不同的方法称为方法的\_\_\_覆盖\_\_\_\_。
6. 一个新线程启动使用的方法是\_\_start( )\_\_\_\_\_；方法\_\_notifyAll()\_\_\_\_\_唤醒所有正在等待对象锁的线程。
7. 字节流由两个类层次结构定义。在顶层有两个抽象类：\_\_InputStream\_\_\_\_\_ 和\_\_\_\_ OutputStream \_\_\_ 。
8. 类\_\_\_InetAddress\_\_\_\_提供从域名地址查询IP地址的方法；方法\_\_\_getHostAddress\_\_\_\_返回IP地址，它是用字符串表示的IP地址。
9. Java支持在一个类中声明另一个类，这样的类称作\_\_内部类\_\_\_\_\_；接口实现中，接口变量引用子类对象的机制称为\_\_\_多态\_\_\_\_。
10. 用来创建一个类的实例的方法称为\_\_\_实例化\_\_\_\_；关键字\_\_\_static\_\_\_\_修饰的成员称为类成员。
11. Java语言中，表示一个类A继承自父类B，并实现接口C的语句是\_\_\_\_class A extends B implements C\_\_\_\_\_\_\_\_。如果子类中的方法compute()覆盖了父类中的方法compute()，假设父类的compute()方法头部有可见性修饰符public，则子类中的compute()的可见性修饰符必须是\_\_\_\_public\_\_\_\_\_\_。
12. \_\_PipedInputStream\_\_\_\_\_\_类和\_\_\_ PipedOutputStream \_\_\_\_\_类用于实现管道的输入和输出字节流操作。
13. Socket\_\_\_类和\_\_\_ServerSocket\_\_\_\_\_类用于TCP/IP中实现面向连接的通信。
14. \_\_\_DatagramPacket\_\_\_\_\_类和\_\_\_DatagramSocket\_\_\_\_\_类用于TCP/IP中实现无连接服务的通信。

声明方法可能抛出异常的关键字是\_\_throws\_\_\_\_\_\_；打印堆栈轨迹的方法是\_\_ printStackTrace()

1. 内部类可以对同一个包中的其它类隐藏起来，有四种类型的内部类，包括常规内部类(内部成员类)，\_\_\_局部内部类\_\_\_\_\_， \_\_\_\_匿名内部类\_\_\_\_和静态内部类。
2. 检查一个对象是否是某个类的实例的关键字是\_\_\_instanceof\_\_\_\_\_；给一个对象赋值为空的关键字是\_\_\_null\_\_\_\_\_。
3. java中方法的参数传递永远是\_\_按值传递\_\_\_\_\_\_；同一个类中构造函数A(String s)中调用另一个构造函数A(int i)的方法是\_\_\_this(int i)\_\_\_\_\_。
4. \_\_\_\_class\_\_\_\_\_\_\_\_是Java的关键字，用来定义类。
5. 在一个Java应用程序中main方法必须被声明为\_\_public\_\_\_\_\_、\_\_static\_\_\_\_\_\_\_、\_\_\_void\_\_\_\_\_。
6. 面向对象编程序主要有\_\_\_\_\_封装性\_\_\_\_\_\_\_\_、\_\_\_继承性\_\_\_\_\_\_\_、多态性的特点。
7. java中\_\_\_\_\_\_构造\_\_\_\_\_\_方法与类名相同，没有返回值，在创建对象实例时由new运算符自动调用。
8. 访问权限是\_\_\_\_private\_\_\_\_\_\_\_\_变量，只能在本类中使用，不能在类外使用。
9. 如果一个类的声明中没有使用extends关键字，这个类被系统默认为是\_\_\_\_Object\_\_\_\_\_\_类的子类。
10. Object类有一个public方法是\_\_\_\_toString()\_\_\_\_\_\_\_\_，一个对象通过调用该方法可以获得该对象的字符串表示。
11. 对于一个已经不被任何引用变量引用的对象，当垃圾回收器准备回收该对象所占用的内存时，将自动调用该对象的\_\_\_\_finalize()\_\_\_\_\_\_\_\_方法。

## 程序题

1、编程完成接口、类及其继承关系：接口为 (Shape)；矩形(Rectangle)实现接口；正方形(Square)继承了矩形的特征，并重写矩形实现的接口方法；圆形(Circle)实现了接口。

1）接口Shape具有抽象方法为：getPerimeter()返回几何图形的周长；getArea()返回几何图形的面积；disp()显示几何图形的属性、周长及面积。

2）矩形Rectangle具有属性长length与宽width；构造方法Retangle(**double** l, **double** w)；

3）正方形 Square具有属性边长length；构造方法Square (**double** side)；

4）圆形Circle 具有属性半径radius，构造方法Circle (**double** r)。

5）完成测试程序，分别定义矩形、正方形、圆的对象，并由接口引用，由接口调用disp()。

**package** test;

**public** **class** Pic {

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

Shape a=**new** Rectangle(2,4);

Shape b=**new** Square(3,3);

Shape c=**new** Circle(5);

a.disp();

b.disp();

c.disp();

}

}

**interface** Shape {

**public** **abstract** **double** getPerimeter();

**public** **abstract** **double** getArea();

**public** **abstract** **void** disp();

}

**class** Rectangle **implements** Shape{

**private** **double** lenght;

**private** **double** width;

**public** Rectangle(**double** l,**double** w){

**this**.lenght=l;

**this**.width=w;

}

**public** **double** getPerimeter(){

**return** 2\*(lenght+width);

}

**public** **double** getArea(){

**return** lenght\*width;

}

**public** **void** disp(){

System.***out***.println("【矩形】\n长："+lenght+"\n宽："+width+"\n周长："+getPerimeter()+"\n面积："+getArea());

}

}

**class** Square **extends** Rectangle{

**private** **double** lenght;

**public** Square(**double** l,**double** w){

**super**(l,w);

**this**.lenght=l;

}

**public** **double** getPerimeter(){

**return** 4\*lenght;

}

**public** **double** getArea(){

**return** lenght\*lenght;

}

**public** **void** disp(){

System.***out***.println("【正方形】\n边长："+lenght+"\n周长："+getPerimeter()+"\n面积："+getArea());

}

}

**class** Circle **implements** Shape{

**private** **double** radius;

**public** Circle(**double** r){

**this**.radius=r;

}

**public** **double** getPerimeter(){

**return** Math.***PI***\*2\*radius;

}

**public** **double** getArea(){

**return** Math.***PI***\*radius\*radius;

}

**public** **void** disp(){

System.***out***.println("【圆】\n半径:"+radius+"\n周长："+getPerimeter()+"\n面积："+getArea());

}

}

2、定义一个表示学生信息的类Student，要求如下：

　（1）类Student的成员变量：

sNO　表示学号；sName表示姓名；sSex表示性别；sAge表示年龄；

sJava：表示Java课程成绩。

　（2）类Student带参数的构造方法： 在构造方法中通过形参完成对成员变量的初始化。

（3）类Student的方法成员：

getNo（）：获得学号； getName（）：获得姓名；

getSex（）：获得性别；getAge（）获得年龄；getJava（）：获得Java 课程成绩

（4）根据类Student的定义，创建五个该类的对象，输出每个学生的信息，计算并输出这五个学生Java语言成绩的平均值，以及计算并输出他们Java语言成绩的最大值和最小值。

**package** test;

**import** java.util.Arrays;

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

**private** **int** sNO;

**private** String sName;

**private** **char** sSex;

**private** **int** sAge;

**private** **double** sJava;

**public** Student(**int** sNO,String sName,**char** sSex,**int** age,**double** sJava){

**this**.sNO=sNO;

**this**.sName=sName;

**this**.sSex=sSex;

**this**.sAge=age;

**this**.sJava=sJava;

}

**public** **int** getNo(){

**return** sNO;

}

**public** String getName(){

**return** sName;

}

**public** **char** getSex(){

**return** sSex;

}

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

**return** sAge;

}

**public** **double** getJava(){

**return** sJava;

}

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

Student a=**new** Student(1,"小明",'男',19,90);

Student b=**new** Student(2,"小红",'女',19,92);

Student c=**new** Student(3,"小军",'男',20,62);

Student d=**new** Student(4,"小刘",'男',19,98);

Student e=**new** Student(5,"小芳",'女',20,95);

System.***out***.println("【学生】\n学号："+a.getNo()+"\n姓名："+a.getName()+"\n性别："+a.getSex()+"\n年龄："+a.getAge()+"\nJava成绩："+a.getJava());

System.***out***.println("【学生】\n学号："+b.getNo()+"\n姓名："+b.getName()+"\n性别："+b.getSex()+"\n年龄："+b.getAge()+"\nJava成绩："+b.getJava());

System.***out***.println("【学生】\n学号："+c.getNo()+"\n姓名："+c.getName()+"\n性别："+c.getSex()+"\n年龄："+c.getAge()+"\nJava成绩："+c.getJava());

System.***out***.println("【学生】\n学号："+d.getNo()+"\n姓名："+d.getName()+"\n性别："+d.getSex()+"\n年龄："+d.getAge()+"\nJava成绩："+d.getJava());

System.***out***.println("【学生】\n学号："+e.getNo()+"\n姓名："+e.getName()+"\n性别："+e.getSex()+"\n年龄："+e.getAge()+"\nJava成绩："+e.getJava());

**double**[] grade=**new** **double**[5];

**double** average=0;

**double** sum=0;

grade[0]=a.getJava();

grade[1]=b.getJava();

grade[2]=c.getJava();

grade[3]=d.getJava();

grade[4]=e.getJava();

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

sum+=grade[i];

}

average=sum/5;

Arrays.*sort*(grade);

System.***out***.println("\n平均成绩："+average+"\n最高分："+grade[4]+"\n最低分"+grade[0]);

}

}

3、 （1）编写圆类Circle，该类拥有：

① 成员变量：Radius（私有，浮点型）存放圆的半径；

②两个构造方法：Circle( ) 将半径设为0

Circle(double r ) 创建Circle对象时将半径初始化为r；

③ 三个成员方法: double getArea( ) 获取圆的面积；

double getPerimeter( ) 获取圆的周长；

void show( ) 将圆的半径、周长、面积输出到屏幕；

（2）编写一个圆柱体类Cylinder，它继承于上面的Circle类。还拥有：

①一个成员变量：double hight（私有，浮点型）圆柱体的高；

②构造方法： Cylinder (double r, double h ) 创建Circle对象时将半径初始化为r；

③ 成员方法: double getVolume( ) 获取圆柱体的体积；

void showVolume( ) 将圆柱体的体积输出到屏幕；

编写应用程序，创建类的对象，分别设置圆的半径、圆柱体的高，计算并分别显示圆半径、圆面积、圆周长，圆柱体的体积。

**package** test;

**public** **class** Circle {

**private** **double** Radius;

**public** Circle(){

**this**.Radius=0;

}

**public** Circle(**double** r){

**this**.Radius=r;

}

**public** **double** getArea(){

**return** Math.***PI***\*Radius\*Radius;

}

**public** **double** getPerimeter(){

**return** Math.***PI***\*2\*Radius;

}

**public** **void** show(){

System.***out***.println("【圆】\n半径："+Radius+"\n周长："+getPerimeter()+"\n面积"+getArea());

}

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

Cylinder a=**new** Cylinder(2,3);

a.show();

a.showVolume();

}

}

**class** Cylinder **extends** Circle{

**private** **double** height;

**public** Cylinder(**double** r,**double** h){

**super**(r);

**this**.height=h;

}

**public** **double** getVolume(){

**return** getArea()\*height;

}

**public** **void** showVolume(){

System.***out***.println("圆柱体积："+getVolume());

}

}