## 几个概念

一、java体系

1、javaSE，标准版，各应用平台的基础，桌面开发和低端商务应用的解决方案。（eclipse，finalshell）

2、javaEE，企业版，以企业为环境而开发应用程序的解决方案。（spring框架支持）

3、javaME，微型版，致力于消费产品和嵌入式设备的最佳方案。

二、java可以做什么

1、开发桌面应用程序。

2、开发面向Internet的web应用程序。

3、开发android手机应用程序。

三、java的特性

1、一种面向对象的编程语言。

2、一种与平台无关的语言（根据JVM实现的）。

3、一种健壮性语言、具有较高的安全性。

四、java程序开发的三个步骤

1、编写源程序，java源代码文件。

2、编译源程序，编译器编译成java字节码文件。

3、运行，java虚拟机（JVM）。

五、JDK和JRE

JDK：java开发工具包

先编译（编译器javec），后运行（解释器java）

JRE：java运行环境

加载代码（加载器），校验代码（校验器），执行代码（解释器）

六、java虚拟机

java虚拟机实际上只是一层接口，一层Java程序和操作系统通讯的接口。java文件编译生成class文件，而java虚拟机就是这些class文件能够在上面运行的一个平台，你把class文件看成一个软件，java虚拟机就是这个软件可以运行的操作系统。

七、开发java应用的要点

1、一个源文件中只能有一个public修饰的类，其他类个数不限。

2、一个源文件有n个类时，编译结果的class文件就有n个。

3、源文件的名字必须和public修饰的类名相同

4、java语言中单词拼写大小写严格区分。

5、main方法入口

## 编程基础

一、命名规范

1、命名规则：由字母、下划线、数字组成，不能以数字开头，区分大小写，不能是关键字和保留字，长度一般不超过15个字符。

2、驼峰式命名：

类名、接口名：单个单词，首字母大写，多个单词，首字母都大写。

方法名、参数名、变量名：单个单词，首字母小写，多个单词，第一单词首字母小写，其他单词首字母大写。

包名：全部小写。

二、java数据类型划分

1、基本数据类型：

数值型：byte 1字节

short 2字节

int 4字节

long 8字节

浮点类型：

float 4字节

double 8字节

字符型：char 2字节

布尔型：boolean true false

2、引用类型：

字符串 String、 类 class 、枚举 enum、接口interface

三、基本数据类型之间的转换

1、自动类型转换：范围小→范围大

byte→short→int→long→float→double；

char→int→long→float→double

2、强制类型转换：范围大→范围小

需要加强制转换符

四、流程控制语句

If else while for switct …..

## 面向对象

1. 封装

作用：提高代码的安全性

1、将属性私有化，并提供对外界的接口（get/set方法）。

2、用private修饰的属性和方法，只能在本类中使用。

1. 继承

作用：提高代码的复用性，减少重复代码

1、子类可以继承父类非私有的属性和方法，不能继承构造方法。

2、子类在继承父类的各种属性和方法时，也可以有自己的属性和方法。

3、一个子类只能有一个父类，java只能单继承，不能多继承

4、一个类继承最顶端叫做基类或者超类，所有的超类叫做object。

1. 多态

1、分类

编译时多态：在编译过程中察觉的多态，重载，向上转型。

运行时多态：在运行过程中察觉的多态，向下转型。

2、向上转型、向下转型是在继承关系中，向下转型必须在向上转型的基之上。

3、在继承关系中，父类的引用可以指向子类的实例，父类引用实际方法的时候，是调用子类重写以后的方法。

4、向上转型

父类的引用指向子类的对象

父类类名 变量名=new 子类类();

优点：减少重复代码，提高代码的复用性

缺点:父类的引用无法调用子类特有的属性和方法

解决方法：向下转型

5、向下转型： B🡪A A a = new B(); a=====>show(a) B b = (B)a; b.test()

父类引用 赋给 子类

子类类名 变量名=（子类类名）父类引用;

6、 instanceof 判断左边的对象是否属于右边的类的实例 变量名 instanceof 类名（子类类名）

1. 抽象类abstract

作用：节省代码，提高代码的复用性

1、抽象类格式：访问权限修饰符 abstract class 类名{}

2、抽象方法格式：访问权限修饰符 abstract 返回值类型 方法名(形式参数列表);

注意：

1、如果一个类里有抽象方法，那么这个类必须声明成抽象类。

2、类继承抽象类，子类需重写父类的抽象的方法

3、抽象类不能创建对象。

4、抽象类可以有构造方法，在创建子类的时候，super隐式调用父类的构造方法。

5、在继承关系中，子类能够继承抽象类的各种属性和方法，除了私有的和构造方法。

1. 接口interface

作用：规范了代码，提高代码的拓展性

1、格式：访问权限修饰符 interface 接口名称{}

2、实现类的格式：访问权限修饰符 class 实现类名 implements 接口名{必须重写接口中的所有的抽象方法}

3、接口中只有全局常量和抽象方法。

4、书写的时候可以省略部分修饰符，系统会默认给添上。（变量默认是public static final，方法默认是public）

5、实现类可以在实现接口的同时继承其他类型，extends在implement前面。

6、接口可以多实现。

六、抽象类和接口的区别

1、关键字：抽象类 abstract 接口interface

2、抽象类继承 extends 接口实现 implements

3、接口中只有全局变量和抽象方法 抽象类中有各种属性和方法

4、抽象类只能单一继承，子类只能继承一个父类，但可以实现多个接口

5、抽象类的作用：提高代码的复用性 接口的作用：1、规范代码2、提高代码的拓展性

## 专题知识

一、Java异常处理机制

1、throwable是错误和异常的超类

2、错误Error：程序员无法处理的严重性问题，资源耗尽，jvm系统内部的错误

异常Exception:程序员可以处理的一般性问题，偶然的外界因素，编程的逻辑性错误

3、异常分类：

受检异常：编译时期就能发现的异常，必须要去处理的异常，继承自Exception（SQLException、IOException）

非受检异常：运行时期发现的异常，不是必须要去处理的，继承自RuntimeException（NullPointerException，IndexOutOfBoundsException）

4、异常的处理机制：

程序运行时产生异常对象，抛出给jvm，jvm会携带异常信息，去找代码能够处理或者捕获异常代码，

如果找到了，则交给这个代码去处理，没有找到，则程序停止运行。

5、异常处理的两种方式

1、捕获异常

try{可能会产生异常的代码}catch(异常类 异常对象){处理异常的代码}

try{}catch(){}catch(){}... catch中子类异常放在父类异常的上面

try{]catch(){}finally{} finally中的代码一定会被执行到

try{}finally{}

2、抛出异常

产生异常的位置不去处理异常,由调用此方法的调用者去处理

Throws 方法的声明 异常类名（可以跟多个类名之间用逗号隔开），可以抛出受检异常和非受检异常

throw 方法的内部 异常的对象名（只能抛出一个对象）可以抛出受检异常和非受检异常

6、自定义异常

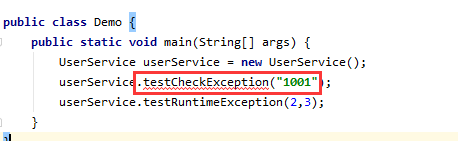
受检异常继承Exception

非受检异常 RuntimeException

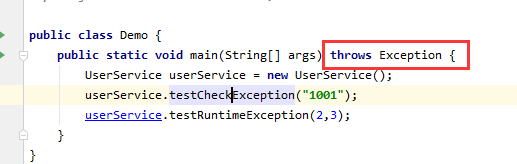
**public void** testCheckException(String str) **throws** Exception {  
 **try** {  
 **int** number = Integer.*valueOf*(str);  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 **throw new** IllegalArgumentException(**"参数不合法"**);  
 }  
}

**public void** testRuntimeException(**int** num1, **int** num2) **throws** RuntimeException {  
 **try** {  
 **int** num = num1 / num2;  
 } **catch** (Exception ex) {  
 **throw new** RuntimeException(**"数据不合法"**);  
 }  
}

调用方需要处理检查异常



方法一（继续向上层抛出）



方法二（调用方直接处理掉）

