一、1 – 10 题

1. 下列对继承的说法，正确的是：子类能继承父类的所有成员。

解析：**在一个子类被创建的时候，首先会在内存中创建一个父类对象，然后在父类对象外部放上子类独有的属性，两者合起来形成一个子类的对象**。所以所谓的继承使子类拥有父类所有的属性和方法其实可以这样理解，**子类对象确实拥有父类对象中所有的属性和方法，但是父类对象中的私有属性和方法，子类是无法访问到的，只是拥有，但不能使用。**就像有些东西你可能拥有，但是你并不能使用。所以子类对象是绝对大于父类对象的，所谓的子类对象只能继承父类非私有的属性及方法的说法是错误的。可以继承，只是无法访问到而已。

2. 接口的定义正确的是：

interface B{ void print(); }

解析：关键字interface，接口中的方法默认为public abstract，属性默认为public static final。接口中的方法不能有实现，没有大括号方法体，Java中类只支持单继承，而接口可以支持多继承。

3. Obeject类中的方法：不包括finally()

解析：Object类中的全部方法

protected Object clone() //创建并返回此对象的副本。

boolean equals(Object obj) //判断是否相等

protected void finalize() //通常用于GC回收，手动调用

class<?> getClass() //得到该类的反射

int ashCode() //返回对象的hashCode()

void otify() //唤醒等待中的单个线程

void otifyAll() /唤醒等待中的所有线程

String toString() /返回去对象的字符串表示

void ait() /使线程进入等待状态

void wait(long timeout)

//导致当前线程等待，直到另一个线程调用 notify()方法或该对象的 notifyAll()方法，或者指定的时间已过。

void wait(long timeout, int nanos)

//导致当前线程等待，直到另一个线程调用该对象的 notify()方法或 notifyAll()方法，或者某些其他线程中断当前线程，或一定量的实时时间。

4. 运行下列代码

abstract class Animal{

abstract void say(); //抽象类有一个抽象方法

}

public class Cat extends Animal{

public Cat(){ //构造方法

System.out.printf("I am a cat");

} //原因：Cat继承抽象类，但是没有实现抽象方法所以本身也是抽象类。

public static void main(String[] args) {

Cat cat=new Cat(); //抽象类不能创建实例

}}

运行结果：Animal能编译，Cat不能编译

解析：

包含抽象方法的类称为抽象类，但并不意味着抽象类中只能有抽象方法，它和普通类一样，同样可以拥有成员变量和普通的成员方法。注意，抽象类和普通类的主要有三点区别：

(1) 抽象方法必须为public或者protected（因为如果为private，则不能被子类继承，子类便无法实现该方法），缺省情况下默认为public。

(2) 抽象类不能用来创建对象；

(3) **如果一个类继承于一个抽象类，则子类必须实现父类的抽象方法**。如果子类没有实现父类的抽象方法，则必须将子类也定义为为abstract类。

在其他方面，抽象类和普通的类并没有区别。

5. 重载和重写描述正确的是：方法重写的返回值类型必须相同或相容。（或是其子类）

解析：

(1) 方法重载（overload）：//返回值相同与否都可以

① 必须是同一个类

② 方法名（也可以叫函数）一样

③ 参数类型不一样或参数数量不一样

(2) 方法的重写（override）两同两小一大原则：

① 方法名相同，参数类型相同

② 子类返回类型小于等于父类方法返回类型，

③ 子类抛出异常小于等于父类方法抛出异常，

④ 子类访问权限大于等于父类方法访问权限。

6. 用命令行方式运行下列代码：

public class f{

public static void main(String[] args){

String foo1 = args[1];

String foo2 = args[2];

String foo3 = args[3];

}

}

命令行：java T11 a b c // 命令 类名 参数

该命令编译错误的原因：因为参数只有三个，而arg[3]数组越界了，所以编译会报错。另一个原因是，找不到T11这个类。

7. 运行下列代码：

int i = 5;

int j = 10;

System.out.println(i + ~j);

得到的结果是：-6

解析：~表示按位非，就是表示按位取反，比如”1100”取值为”0011”。公式：-n = ~n + 1，所以~n = -11，所以结果等于-6。

扩展（原码，补码，反码） -- 已整理

(1) 在java虚拟机中整数有byte、short、int、long四种 分别表示 8位、16位、32位、64位有符号整数。整数使用补码表示。

原码：符号位加上数字的二进制表示（假设为byte字节）

+7的原码为： 00000111

-7的原码为： 10000111

反码：一个数如果为正，则它的反码与原码相同；一个数如果为负，则符号位为1

+7的反码为：00000111

-7的反码为： 11111000

补码：一个数如果为正，则它的原码反码补码相同；一个数如果为负，取到反码然后加1。

+7的补码为： 00000111

-7的补码为： 11111001

注意：在计算机当中，使用补码存储。

(2) 上面的那个例子

10补码：0000 1010（原码，反码，补码一样）

~ 10补码：1111 0101（这是存储在机器里的补码）

~10反码：1111 0100

~10 原码：1000 1011 值为-11

8. 方法通常存储在进程中的哪一区（）：方法区

解析：

一条进程的栈区、堆区、数据区和代码区在内存中的映射

(1) 栈区：主要用来存放局部变量, 传递参数, 存放函数的返回地址。.esp 始终指向栈顶, 栈中的数据越多, esp的值越小。

(2) 堆区：用于存放动态分配的对象, 当你使用 malloc和new 等进行分配时,所得到的空间就在堆中。动态分配得到的内存区域附带有分配信息, 所以你　 能够 free和delete它们。

(3) 数据区：全局，静态和常量是分配在数据区中的，数据区包括bss（未初始化数据区）和初始化数据区。

(4) 方法一般放在代码区里，也可以称为方法区。

(5) 注意：

1)堆向高内存地址生长；

2)栈向低内存地址生长；

3)堆和栈相向而生，堆和栈之间有个临界点，称为stkbrk。

9. 下列关于java并发的说法中正确的是：readwritelock适用于读多写少的并发场景

解析：

(1) CopyOnWriteArrayList适合使用在读操作远远大于写操作的场景里，比如缓存。\

(2) ReadWriteLock 当写操作时，其他线程无法读取或写入数据，而当读操作时，其它线程无法写入数据，但却可以读取数据 。适用于 读取远远大于写入的操作。

(3) ConcurrentHashMap是一个线程安全的Hash Table，它的主要功能是提供了一组和HashTable功能相同但是线程安全的方法。

(4) ConcurrentHashMap可以做到读取数据不加锁，并且其内部的结构可以让其在进行写操作的时候能够将锁的粒度保持地尽量地小，不用对整个ConcurrentHashMap加锁。

(5) Volatile：与锁相比，Volatile 变量是一种非常简单但同时又非常脆弱的同步机制，它在某些情况下将提供优于锁的性能和伸缩性。如果严格遵循 volatile 的使用条件 —— 即变量真正独立于其他变量和自己以前的值 —— 在某些情况下可以使用 volatile 代替 synchronized 来简化代码。然而，使用 volatile 的代码往往比使用锁的代码更加容易出错。

要使 volatile 变量提供理想的线程安全，必须同时满足下面两个条件：

① 对变量的写操作不依赖于当前值。

② 该变量没有包含在具有其他变量的不变式中。

10. 重载和重写，见5题。

二、11-20题

1. 运行下列程序：

public class Person{

static int arr[] = new int[5]; //已经分配了内存空间，但是没有初始化

public static void main(String a[]){

System.out.println(arr[0]); //输出默认值

}

}  
 编译正确，输出0。

(1) 解析：

① 类的实例变量会被默认初始化，方法里的局部变量不会被默认初始化。

② char类型默认为空格，int类型默认为0，String类型默认为null。

3. 执行new Child("John", 10); 要使数据域data得到10，则子类空白处应该填写

class Parent {

private int data;

public Parent(int d){

data = d;

}

}

class Child extends Parent{

String name;

public Child(String s, int d){

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ //填写super(d);

name = s;

}

}

(1) 解析：

① 创建子类对象时，会默认调用父类的无参构造函数。如果父类没有无参构造函数，需显式用。

4. Java 的源码文件.java在计算机中的执行流程

(1) 源文件由编译器编译成字节码（ByteCode）

(2) 字节码由java虚拟机解释运行。因为java程序既要编译同时也要经过JVM的解释运行，所以说Java被称为半解释语言（ "semi-interpreted" language）。

(3) 最后由JVM解释成机器码（Machine Code）在计算机上执行。

6. 选项中哪一行代码可以添加 到题目中而不产生编译错误？

public abstract class MyClass {

public int constInt = 5;

//add code here

public void method() {

}

}、

(1) public abstract void method(int a); //抽象方法可以包含重载的抽象方法

(2) constInt = constInt + 5; //类由实例变量和方法体组成，应放到方法里

(3) public int method(); //参数一样返回值不同，不构成重载条件

(4) public abstract void anotherMethod() {} //抽象方法没有方法体，所以没有大括号

7. 文件名称：forward,jsp //请求重定向和请求转发的区别

<html>

<head><title> 跳转 </title> </head>

<body>

<jsp:forward page="index.htm"/>

</body>

</html>

(1) redirect：请求重定向：客户端行为，本质上为2次请求，地址栏改变，前一次请求对象消失。

(2) forward：请求转发:服务器行为，地址栏不变。

8. Maven和Ant的区别【都是Java构建工具】

(1) Ant和Maven都是基于Java的构建(build)工具。Ant是软件构建工具，Maven的定位是软件项目管理和理解工具。]

(2) Ant特点：

没有一个约定的目录结构，必须明确让ant做什么，什么时候做，然后编译，打包 没有生命周期，必须定义目标及其实现的任务序列 没有集成依赖管理。

(3) Maven特点

拥有约定，知道你的代码在哪里，放到哪里去。拥有一个生命周期，例如执行 mvn install 就可以自动执行编译，测试，打包等构建过程。只需要定义一个pom.xml,然后把源码放到默认的目录，Maven帮你处理其他事情。拥有依赖管理，仓库管理。

9. jsp中代码 //JSP九大内置对象

<%@ page language="java" import="java.util.\*" errorPage="error.jsp"

isErrorPage="false" %>

Exception是JSP九大内置对象之一，其实例代表其他页面的异常和错误。只有当页面是错误处理页面时，即isErroePage为 true时，该对象才可以使用。、

10. Java类加载器（JVM相关内容）

JVM classLoader architecture ：

(1) Bootstrap ClassLoader/启动类加载器 //加载 Java 的核心库，用原生代码来实现的

主要负责jdk\_home/lib目录下的核心 api 或 -Xbootclasspath 选项指定的jar包装入工作.。

(2) Extension ClassLoader/扩展类加载器 //加载Java的扩展库

主要负责jdk\_home/lib/ext目录下的jar包或 -Djava.ext.dirs 指定目录下的jar包装入工作。

(3) System ClassLoader/系统类加载器

//根据 Java 应用的类路径（CLASSPATH）来加载 Java 类

主要负责java -classpath/-Djava.class.path所指的目录下的类与jar包装入工作.。

(4) User Custom ClassLoader/用户自定义类加载器(java.lang.ClassLoader的子类)

在程序运行期间, 通过java.lang.ClassLoader的子类动态加载class文件, 体现java动态实时类装入特性.。

三、21-30题

1. 设int x=1,float y=2,则表达式x/y的值是：0.5

(1) 解析：int类型会向float高精度类型转换，所以结果就是float类型0.5。

2. 在 Java 中，存放**字符串常量**的对象属于String类对象

(1) 解析：

① 生成字符串的两个类String和StringBuffer中，前者生成的字符串是不变字符串，不能直接对它的内容进行修改，而后者生成的字符串是可变的，可以对其内容进行修改。

② 而Character类是对字符类型的类封装。

3. 访问权限关键字



5. 语句：char foo='中'，是否正确？（假设源文件以GB2312编码存储，并且以javac – encoding GB2312命令编译）//考察char类型

java的char类型占两个字节，默认使用GBK编码存储。所以可以使用char存储中文字符。

6. 考察接口中默认类型，见一.2题

8. 除去扩展名.java的部分，包含在java文件里的Java里的公共类必须与文件的名字相同。

关键：但是公共的内部类是不需要和文件的名字相同的。

9. 对象序列化正确的是：

(1) 使用ObjectOutputStream和ObjectInputStream可以将对象进行传输.

(2) transient 修饰的变量在对象串化的时侯并不会将所赋值的值保存到传中，串化的对象从磁盘读取出来仍然是null。

(3) 对象序列化的所属类需要实现Serializable接口

10. 构造方法正确的特点：

(1) 构造方法的方法名必须与类名（public 类）相同。

(2) 构造方法没有返回类型，也不能定义为void，在方法名前面不声明方法类型。

(3) 构造方法的主要作用是完成对象的初始化工作，它能够把定义对象时的参数传给对象的域。

(4) 一个类可以定义多个构造方法，如果在定义类时没有定义构造方法，则编译系统会自动插入一个无参数的默认构造器，这个构造器不执行任何代码。

(5) 构造方法可以重载，以参数的个数，类型，顺序。

(6) 不能被static、final、synchronized、abstract和native修饰；访问权限修饰符是可以的。

四、31-40题

1. 考察构造方法 ，见三.10

2. static关键字，修饰实例变量为静态变量，修饰方法为静态方法。 -- 已整理

(1) 静态变量和静态方法，可以直接通过类名进行调用；其他的成员和方法则需要进行实例化成对象之后，通过对象来调用。

(2) 静态变量在第一次使用的时候就会初始化，有且只有一个成员对象，可以通过对象调用或者类名.变量名的方式来调用。

4. JDBC的statement类

(1) 标准的statement类有：Statement、PreparedStatement、CallableStatement

① Statement是最基本的用法，采用字符串拼接的方式，存在注入漏洞。

② PreparedStatement对Statement中的SQL语句进行预编译，同时检查合法性，效率高。

③ CallableStatement接口扩展 PreparedStatement，用来调用存储过程,它提供了对输出和输入/输出参数的支持。CallableStatement 接口还具有对 PreparedStatement 接口提供的输入参数的支持。

(2) BatchedStatement用于批量操作数据库，BatchedStatement不是标准的Statement类

5. 见3题、

6. 终止线程的方法（sleep和wait并不会终止线程 ）

抛出一个异常、stop方法、interrupt、线程正常终止

8. 继承—构造方法

(1) 在java中，**子类构造器会默认调用super()(无论构造器中是否写有super（）),用于初始化父类成员**，同时当父类中存在有参构造器时，必须提供无参构造器，子类构造器中并不会自动继承有参构造器，仍然默认调用super()，使用无参构造器。因此，一个类想要被继承必须提供无参构造器。

10. 四个线程安全的集合类

(1) Vector：就比arraylist多了个同步化机制（线程安全），因为效率较低，现在已经不太建议使用。在web应用中，特别是前台页面，往往效率（页面响应速度）是优先考虑的。

(2) Statck：堆栈类，先进后出

(3) Hashtable：就比hashmap多了个线程安全

(4) Enumeration：枚举，相当于迭代器

五、41-50题

1. 静态变量 – 已整理

(1) 如果是本类使用，可以直接就用静态变量名。

(2) 如果是其他类使用，可以使用类名来调用，也可以创建一个实例对象来调用。

(3) 如果静态变量所在的类是静态类，那么不管在本类里或者在其他外部类，都可以直接使用静态变量名。

2. 异常的处理

(1) 在实际应用中，当一个异常出现时，单靠某个方法无法完全处理该异常，必须由几个方法协作才可以处理，这个时候同时使用catch和throw语句，**捕获异常后，只对异常进行部分处理，还有些处理需要在该方法的调用者中才能完成**，所以应该再次抛出异常，让该方法的调用者也能捕获到异常。

(2) 程序运行时异常由Java虚拟机自动进行处理

3. java文件中的类

(1) .java文件可以包含多个类，唯一的限制就是：一个文件中只能有一个public类， 并且此public类必须与文件名相同。而且这些类和写在多个文件中没有区别。

(2) 一个public类中也可以包含其他的内部类。

7. Java中的类型

(1) java语言是强类型语言，支持的类型分为两类：基本类型和引用类型。

(2) 基本类型包括boolean类型和数值类型，数值类型有整数类型和浮点类型。整数类型包括：byte、short、int、long和char；浮点类型包括：float和double

(3) 引用类型包括类、接口和数组类型以及特殊的null类型。

8. Java Web中的会话跟踪概念

(1) 会话跟踪是一种灵活、轻便的机制，它使Web上的状态编程变为可能。

HTTP是一种无状态协议，每当用户发出请求时，服务器就会做出响应，客户端与服务器之间的联系是离散的、非连续的。**当用户在同一网站的多个页面之间转换时，根本无法确定是否是同一个客户，会话跟踪技术就可以解决这个问题。**当一个客户在多个页面间切换时，服务器会保存该用户的信息。

(2) 有四种方法可以实现会话跟踪技术：URL重写、隐藏表单域、Cookie、Session。

① 隐藏表单域：<input type="hidden">，非常适合步需要大量数据存储的会话应用。

② URL 重写:URL 可以在后面附加参数，和服务器的请求一起发送，这些参数为名字/值对。

③ Cookie:一个 Cookie 是一个小的，已命名数据元素。服务器使用 SET-Cookie 头标将它作为 HTTP响应的一部分传送到客户端，客户端被请求保存 Cookie 值，在对同一服务器的后续请求使用一个Cookie 头标将之返回到服务器。与其它技术比较，Cookie 的一个优点是在浏览器会话结束后，甚至在客户端计算机重启后它仍可以保留其值

④ Session：使用 setAttribute(String str,Object obj)方法将对象捆绑到一个会话，存储在服务器端。

9. JVM中的垃圾回收（还未了解）

10. B,C继承A，D继承B

(1) 泛型的原理

泛型只是在编译期保证对象类型相同的技术。真正在代码的运行期，jvm会擦出泛型的存在。所以我们可以利用反射技术为一个已指定泛型的集合添加一个不符合泛型要求的元素，因为反射的生效期在运行期，泛型无法进行拦截。因此，**泛型指定的元素不具有继承的特性。不能将泛型中的派生类类型复制给基类类型。**

错误的例子：

The type List<B>is assignable to List<A>.

The type List<?extends A>is assignable to List<A>.

The type List<Object>is assignable to any List reference.

(2) 为了解决在泛型中不能像正常JAVA类中的继承关系。，出现了通配符写法。

通配符的继承就是为了弥补泛型在继承关系上面的缺陷而诞生的。

① <?> 可以表示所有类型

<? extends E> 表示E的所有子类

<? super E> 表示E的所有父类

② 正确的例子：

The type List<Object>is assignable to List<?>.

The type List<D>is assignable to List<?extends B>.

The type List<?extends B>is assignable to List<?extends A>.