1. 覆盖和重载分别是什么?

重载：同一个类中，同名函数，不同参数个数或参数类型

覆盖：两个类，父类和子类，子类对父类中的同名函数给予不同的函数体实现

覆盖也叫重写，发生在子类与父类之间，表示子类中的方法可以与父类中的某个方法的名称和参数完全相同，通过子类创建的实例对象调用这个方法时，将调用子类中的定义方法，这相当于把父类中定义的那个完全相同的方法给覆盖了，这也是面向对象编程的多态性的一种表现。

重载是指在一个类中，可以有多个相同名称的方法，但是他们的参数列表的个数或类型不同，当调用该方法时，根据传递的参数类型调用对应参数列表的方法。当参数列表相同但返回值不同时，将会出现编译错误，这并不是重载，因为jvm无法根据返回值类型来判断应该调用哪个方法。

1. Java支持多继承么？如果不支持，如何实现?

Java中一个类智能继承一个父类，但能继承多个接口。通过继承接口实现多继承

java中实现多继承有两种方式,一是接口，而是内部类.

1. 什么是值传递和引用传递？java中是值传递还是引用传递，还是都有?

本质上都是值传递

值传递 就是在方法调用的时候，实参是将自己的一份拷贝赋给形参，在方法内，对该参数值的修改不影响原来实参，常见的例子就是刚开始学习c语言的时候那个交换方法的例子了。

引用传递 是在方法调用的时候，实参将自己的地址传递给形参，此时方法内对该参数值的改变，就是对该实参的实际操作。

在java中只有一种传递方式，那就是值传递.可能比较让人迷惑的就是java中的对象传递时，对形参的改变依然会影响到该对象的内容。

下面这个例子来说明java中是值传递.

public class Test {

public static void main(String[] args) {

StringBuffer sb = new StringBuffer("hello ");

getString(sb);

System.out.println(sb);

}

public static void getString(StringBuffer s) {

//s = new StringBuffer("ha");

s.append("world");

}

}

在上面这个例子中,当前输出结果为:hello world。这并没有什么问题，可能就是大家平常所理解的引用传递，那么当然会改变StringBuffer的内容。但是如果把上面的注释去掉，那么就会输出:hello.此时sb的值并没有变成ha hello. 假如说是引用传递的话，那么形参的s也就是sb的地址，此时在方法里new StringBuffer（），并将该对象赋给s，也就是说s现在指向了这个新创建的对象.按照引用传递的说法，此时对s的改变就是对sb的操作，也就是说sb应该也指向新创建的对象，那么输出的结果应该为ha world.但实际上输出的仅是hello.这说明sb指向的还是原来的对象，而形参s指向的才是创建的对象,这也就验证了java中的对象传递也是值传递。

1. 接口和抽象类的区别是什么?

区别1：抽象类单继承，接口多继承

区别2：抽象类中有抽象方法，也有非抽象方法；接口中都是抽象的

区别3：接口中的方法和属性默认都是public final的，抽象类中可以有private,public等不同类型

相同点:

1) 都是上层的抽象层。

2) 都不能被实例化

3) 都能包含抽象的方法

不同点在于：

1. 接口中所有的方法隐含的都是抽象的。而抽象类则可以同时包含抽象和非抽象的方法
2. 类可以实现很多个接口，但是只能继承一个抽象类
3. 类如果要实现一个接口，它必须要实现接口声明的所有方法。但是，类可以不实现抽象类声明的所有方法，当然，在这种情况下，类也必须得声明成是抽象的。
4. 抽象类可以在不提供接口方法实现的情况下实现接口。
5. Java 接口中声明的变量默认都是 final 的。抽象类可以包含非 final 的变量。
6. Java 接口中的成员函数默认是 public 的。抽象类的成员函数可以是 private，protected 或者是 public 。
7. 接口是绝对抽象的，不可以被实例化(java 8已支持在接口中实现默认的方法)。抽象类也不可以被实例化，但是，如果它包含 main 方法的话是可以被调用的。
8. 构造器（constructor）是否可被重写（override）?能被重载吗？

构造器可以被重载，不能被重写

1. String, StringBuffer StringBuilder的区别。

String:固定长度，不能修改。

StringBuffer:可变长度，线程安全

StringBuilder:可变长度，线程不安全

1. JVM内存分哪几个区，每个区的作用是什么?

方法区：线程共享，存储类信息和静态变量，常量等

本地方法栈：线程私有，为native方法服务

虚拟机栈：线程私有,为虚拟机执行java方法(字节码)服务

堆：线程共享，保存对象实例

程序计数器：线程私有，保存线程调用信息, 当前线程所执行的字节码的行号指示器

1. java中垃圾收集的方法有哪些?

回收算法（1）标记清除

(2) 标记整理

(3) 复制算法

一般用的是分代回收算法:

新生代用复制算法

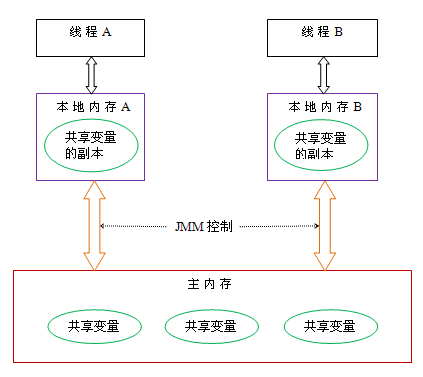
老生代用标记清除或者标记整理算法

1. 如何判断一个对象是否存活?(或者GC对象的判定方法)

判断对象是否需要回收：(1)引用计数法(2)可达性分析

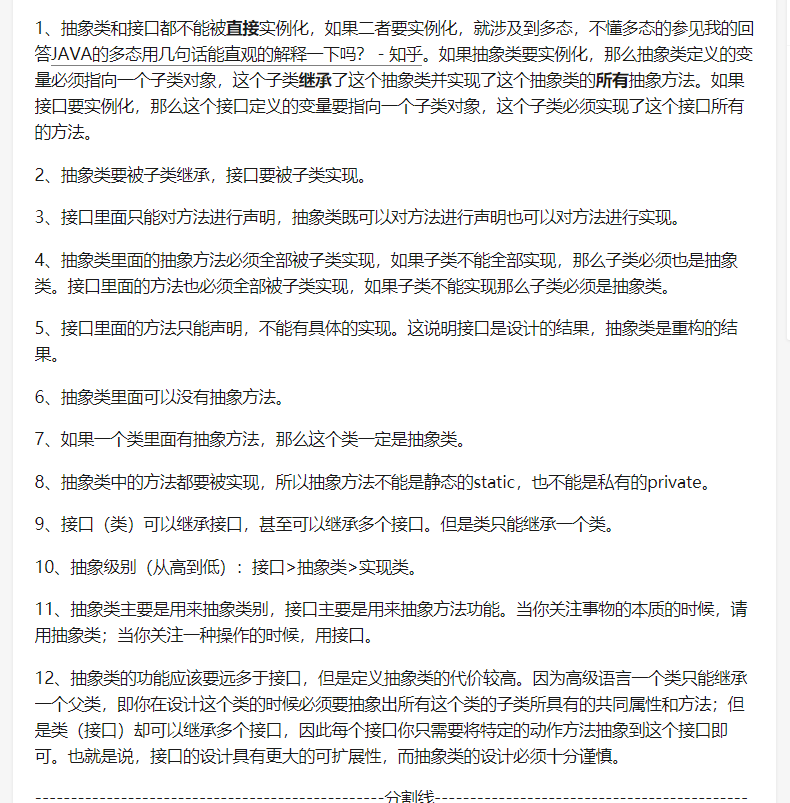
1. 简述java内存模型（画出示意图）

Java内存模型即Java Memory Model，简称JMM。JMM定义了Java 虚拟机(JVM)在计算机内存(RAM)中的工作方式。程序中的变量存储在主内存中，每个线程拥有自己的工作内存并存放变量的拷贝，线程读写自己的工作内存，通过主内存进行变量的交互。JMM就是规定了工作内存和主内存之间变量访问的细节，通过保障原子性、有序性、可见性来实现线程的有效协同和数据的安全。(主存，工作内存)



1. root对象一般是什么?

在Java语言中，可作为GC Roots的对象包括下面几种：   
1. 虚拟机栈（栈帧中的本地变量表）中引用的对象。   
2. 方法区中类静态属性引用的对象。   
3. 方法区中常量引用的对象。   
4. 本地方法栈中JNI（即一般说的Native方法）引用的对象。



抽象类中可以没有抽象方法，但有抽象方法的一定是抽象类，比如常见的有HttpServlet类。但是抽象类是不能被实例化的，即使它没有抽象方法。没有抽象方法的抽象类的价值在于:实例化了没有意义，因为类已经把方法都实现了，而且它不需要通过不同的对象来保存不同的状态。

https://www.zhihu.com/question/20149818