# Android面试题

## Java基础面试题

1. java中==、equals和hashCode的区别

“==”运算符用来比较两个变量的值是否相同，比较变量对应的内存中所存储的数值是否相同，要比较两个基本类型的数据或者两个引用变量是否相等，要使用“==”。要比较两个引用变量的内容是否相同，那么用“==”是无法实现的。

String s1=new String("hello");  
String s2=new String("hello");

例如要比较上面s1和s2 使用“==”就不行了。

equals是Object类提供的一个方法。每一个java类都是继承自Object类，在Object类中的equals方法是和“==”一样的，比较的是引用。但是继承Object类的子类是可以覆盖该方法的，例如String类中的equals是比较的两个对象的内容是否相同。

HashCode方法是从Object类中继承过来的。。Object类中的hashCode()方法返回对象在内存中地址转换成的一个int值，所以如果没有重写hashCode()方法，任何对象的hashCode()方法都是不相等的。一般情况下两个对象equals方法true，hashCode方法返回相同，equals方法false，hashCode方法返回不一定。

1. int 、char、long各占多少字节

整 int 4字节（32位） -231~ 231-1 0 Integer

数 short 2字节（16位） -215~215-1 0 Short

类 long 8字节（64位） -263~263-1 0 Long

型 byte 1字节（8位） -27~27-1 0 Byte

浮点 float 4字节（32位） -3.4e+38 ~ 3.4e+38 0.0f Float

类型 double 8字节（64位） -1.7e+308 ~ 1.7e+308 0 Double

字符 char 2字节（16位） u0000~uFFFF（‘’~‘？’） ‘0’ Character

（0~216-1（65535））

布尔 boolean 1/8字节（1位） true, false FALSE Boolean

1. int 和 Integer的区别

int是基本数据类型，直接存数值，初始化的时候int变量为0。

Integer是对象，是int的封装类，默认值是null。

1. 对java多态的理解

多态就是指程序中定义的引用变量所指向的具体类型和通过该引用变量发出的方法调用在编程时并不确定，而是在程序运行期间才确定，即一个引用变量倒底会指向哪个类的实例对象，该引用变量发出的方法调用到底是哪个类中实现的方法，必须在由程序运行期间才能决定。因为在程序运行时才确定具体的类，这样，不用修改源程序代码，就可以让引用变量绑定到各种不同的类实现上，从而导致该引用调用的具体方法随之改变，即不修改程序代码就可以改变程序运行时所绑定的具体代码，让程序可以选择多个运行状态，这就是多态性。

讲述多态的一片美文<https://www.cnblogs.com/chenssy/p/3372798.html>

1. String、StringBuffer、StringBuilder的区别

String是字符串**常量**

StringBuffer是字符串变量（线程安全）

StringBuilder是字符串变量（非线程安全）

每次对String进行改变都会创建一个新的String对象，然后指向新的对象，所以说在需要频繁改变String内容的时候最好别用String。用多了会触发GC。

针对后面两种就不一样了，每次的修改都是针对一个对象上操作的，所以一般情况下速度是比较快的。

但是在字符串缓冲被多线程使用的时候，StringBuilder是无法保证安全的，但是StringBuffer是可以保证的。

因此：

1.如果要操作少量的数据用 = String

2.单线程操作字符串缓冲区下操作大量数据 StringBuilder

3.多线程操作字符串缓冲区下操作大量数据 StringBuffer

1. 什么是内部类？内部类的作用

内部类是将一个类定义在另一个类里面或者一个方法里面。一般有四种内部类：

1. 成员内部类
2. 局部内部类
3. 匿名内部类
4. 静态内部类

作用

1.每个内部类都能独立的继承一个接口的实现，所以无论外部类是否已经继承了某个(接口的)实现，对于内部类都没有影响。内部类使得多继承的解决方案变得完整，

2.方便将存在一定逻辑关系的类组织在一起，又可以对外界隐藏。

3.方便编写事件驱动程序

4.方便编写线程代码

1. 抽象类和接口的区别

1、抽象类和接口都不能直接实例化，如果要实例化，抽象类变量必须指向实现所有抽象方法的子类对象，接口变量必须指向实现所有接口方法的类对象。

2、抽象类要被子类继承，接口要被类实现。

3、接口只能做方法申明，抽象类中可以做方法申明，也可以做方法实现

4、接口里定义的变量只能是公共的静态的常量，抽象类中的变量是普通变量。

5、抽象类里的抽象方法必须全部被子类所实现，如果子类不能全部实现父类抽象方法，那么该子类只能是抽象类。同样，一个实现接口的时候，如不能全部实现接口方法，那么该类也只能为抽象类。

6、抽象方法只能申明，不能实现，接口是设计的结果 ，抽象类是重构的结果

7、抽象类里可以没有抽象方法

8、如果一个类里有抽象方法，那么这个类只能是抽象类

9、抽象方法要被实现，所以不能是静态的，也不能是私有的。

10、接口可继承接口，并可多继承接口，但类只能单根继承。

1. 抽象类的意义

抽象类的主要作用是将有关的操作作为结果接口组织在一个继承层次结构中，由它来为派生类提供一个公共的根，派生类将具体实现在其基类中作为接口的操作。所以派生类实际上刻画了一组子类的操作接口的通用语义，这些语义也传给子类，子类可以具体实现这些语义，也可以再将这些语义传给自己的子类。

简单的意思：为子类提供一个公共的类型，封装子类中重复内容（成员变量和方法），定义抽象方法

1. 抽象类和接口应用场景

接口interface：

1. 类与类之前需要特定的接口进行协调，而不在乎其如何实现。
2. 作为能够实现特定功能的标识存在，也可以是什么接口方法都没有的纯粹标识。
3. 需要将一组类视为单一的类，而调用者只通过接口来与这组类发生联系。
4. 需要实现特定的多项功能，而这些功能之间可能完全没有任何联系。

抽象类abstract class

1. 定义了一组接口，但又不想强迫每个实现类都必须实现所有的接口。可以用abstract class定义一组方法体，甚至可以是空方法体，然后由子类选择自己所感兴趣的方法来覆盖。
2. 某些场合下，只靠纯粹的接口不能满足类与类之间的协调，还必需类中表示状态的变量来区别不同的关系。abstract的中介作用可以很好地满足这一点。
3. 规范了一组相互协调的方法，其中一些方法是共同的，与状态无关的，可以共享的，无需子类分别实现；而另一些方法却需要各个子类根据自己特定的状态来实现特定的功能
4. 抽象类是否可以没有方法和属性

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。 所以，java中 抽象类里面可以没有抽象方法。比如HttpServlet类。 抽象类和普通类的区别就在于，抽象类不能被实例化，就是不能被new出来，即使抽象类里面没有抽象方法。 抽象类的作用在于子类对其的继承和实现，也就是多态； 而没有抽象方法的抽象类的存在价值在于：实例化了没有意义，因为类已经定义好了，不能改变其中的方法体，但是实例化出来的对象却满足不了要求，只有继承并重写了他的子类才能满足要求。所以才把它定义为没有抽象方法的抽象类

1. 接口的意义

1）类描述了一个实体，包括实体的状态，也包括实体可能发出的动作。

2）接口定义了一个实体可能发出的动作。但是只是定义了这些动作的原型，没有实现，也没有任何状态信息。

3）所以接口有点象一个规范、一个协议，是一个抽象的概念；而类则是实现了这个协议，满足了这个规范的具体实体，是一个具体的概念。

4）从程序角度，简单理解，接口就是函数声明，类就是函数实现。需要注意的是同一个声明可能有很多种实现。

12、泛型中extends和super的区别

这个讲的蛮不错的简单易懂<https://www.zhihu.com/question/20400700>

* 13、父类的静态方法能否被子类重写

答案：不能

重写：子类继承父类后，定义了一个和父类中的一模一样方法，这个一模一样是值方法名和参数的定义一模一样。这时候子类要实现这个方法，就称为对父类方法的重写。

重载：子类继承父类后，定义了一个和父类中相同名字的方法，但是参数不一样（必须），实现也不同（可选），这就是重载。

静态方法：java中，static修饰符修饰的方法就是静态方法。所谓静态就是指：在编译之后所分配的内存会一直存在（不会被回收），直到程序退出内存才会释放这个空间。

不能说重写，没啥意思。

14、进程和线程的区别

（1）调度：线程作为调度和分配的基本单位，进程作为拥有资源的基本单位。

（2）并发性：不仅进程之间可以并发执行，同一个进程的多个线程之间也可并发执行。

（3）拥有资源：进程是拥有资源的一个独立单位，线程不拥有系统资源，但可以访问隶属于进程的资源。

（4）系统开销：在创建或撤消进程时，由于系统都要为之分配和回收资源，导致系统的开销明显大于创建或撤消线程时的开销。

15、final、finally、finalize的区别

final—修饰符（关键字）如果一个类被声明为final，意味着它不能再派生出新的子类，不能作为父类被继承。因此一个类不能既被声明为 abstract的，又被声明为final的。将变量或方法声明为final，可以保证它们在使用中不被改变。被声明为final的变量必须在声明时给定初值，而在以后的引用中只能读取，不可修改。被声明为final的方法也同样只能使用，不能重载。

finally—再异常处理时提供 finally 块来执行任何清除操作。如果抛出一个异常，那么相匹配的 catch 子句就会执行，然后控制就会进入 finally 块（如果有的话）。 finalize—方法名。Java 技术允许使用 finalize() 方法在垃圾收集器将对象从内存中清除出去之前做必要的清理工作。这个方法是由垃圾收集器在确定这个对象没有被引用时对这个对象调用的。它是在 Object 类中定义的，因此所有的类都继承了它。子类覆盖 finalize() 方法以整理系统资源或者执行其他清理工作。finalize() 方法是在垃圾收集器删除对象之前对这个对象调用的。

16、序列化方式

我写的Android序列化<https://www.jianshu.com/p/d89e13627245>

17、静态属性和静态方法是否可以被继承，是否可以重写 原因？

java中静态属性和静态方法可以被继承，但是没有被重写(overwrite)而是被隐藏。

原因：

1， 静态方法和属性是属于类的，调用的时候直接通过类名；方法名完成对，不需要继承机制即可以调用；

（1）如果子类里面定义了静态方法和属性，则这时候父类的静态方法或属性称之为"隐藏"；

（2）如果你想要调用父类的静态方法和属性，直接通过父类名.方法或变量名完成，至于是否继承，子类是有继承静态方法和属性，但是跟实例方法和属性不太一样，存在"隐藏"的这种情况。；

2， 多态之所以能够实现依赖于继承、接口和重写、重载（继承和重写最为关键），有了继承和重写就可以实现父类的引用指向不同子类的对象；

3，重写的功能是："重写"后子类的优先级要高于父类的优先级，但是“隐藏”是没有这个优先级之分的。

4， 静态属性、静态方法和非静态的属性都可以被继承和隐藏而不能被重写，因此不能实现多态，不能实现父类的引用可以指向不同子类的对象。

5，非静态方法可以被继承和重写，因此可以实现多态。

18、静态内部类的设计意图

JAVA 内部静态类--解析静态内部类的使用目的与限制

<http://blog.csdn.net/vange/article/details/5407625>