Java 语言进阶与 Android 技术相关面试题

Java 类加载+与编程关键字

1.Java 中的==、equals 和 hashCode 的区别

（1）“==”运算符用来比较两个变量的值是否相等，即该运算符用于比较变量之间对应的内存中的地址是否相同，要比较两个基本类型的数据或两个引用变量是否相等，只能使用”==”运算符

（2）equals 是 Object 类提供的方法之一，每个 java 类都集成自 Object 类，即每个对象都有 equals 方法，equals 与==一样，比较的都是引用，相比==运算符，equals(Object)方法的特殊之处在于其可以被覆盖，所以可以通过覆盖的方法让他比较的不是引用而是数据内容，即堆中的内容是否相等。

（3）hashCode()方法是从 Object 类继承过来的，他也是用来比较两个对象是否相等，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，所以如果未重写 hashCode 方法，任何对象的 hashCode 方法返回的值都是不相等的。

综上而言，==和 equanl 判断的是（基本类型数据）引用是否相等，但 equals 可以通过覆盖方法使其比较的是数据内容，事实上，Java 中很多类库中的类已经覆盖了此方法，而 hashCode 判断的是对象在内存中地址是否相等。

2.int 和 integer 的区别

Integer 是 int 提供的封装类，而 int 是 java 的基本数据类型，Integer 的默认值是 null，而 int 的默认值是 0，声明 Integer 的变量需要实例化，而 int 不需要，Integer 是对象，是一个引用指向这个对象，而 int 是基本数据类型，直接存储数据。

3.String、StringBuffer 和 StringBuilder 的区别

他们的主要区别在于运行速度和线程安全两个方面，运行速度：StringBuilder>StringBuffer>String,String 最慢的原因在于 String 是字符串常量，一旦创建是不可以再更改的，但后两者的对象是变量，是可以更改的，Java 中对 String 对象的操作实际上是一个不断创建新的对象而将旧的对象回收的过程，而后两者因为是变量，所以可以直接进行更改，在线程安全上，StringBuilder 是线程不安全的，而 StringBuffer 是线程安全的，因为在 StringBuffer 对象在字符串缓冲区被多个线程使用时，StringBuffer 中很多方法可以带有 synchronized 关键字，所以可以保证线程是安全的，而 StringBuilder 不存在该关键字，所以在线程中并不安全。

4.final、finally、finalize 的区别

final 是用于修饰类、成员变量和成员方法，类不可被继承，成员变量不可变，成员方法不可被重写；finally 与 try…catch…共同使用，确保无论是否出现异常都能被调用到；finalize 是类的方法，垃圾回收前会调用此方法，子类可以重写 finalize 方法实现对资源的回收。

5.Serializable 和 Parcelable 的区别

Serlizable 是 java 序列化接口，在硬盘上读写，读写的过程中有大量临时变量产生，内部执行大量的 I/O 操作，效率很低；Parcelable 是 Android 序列化接口，效率高，在内存中读写，但使用麻烦，对象不能保存到磁盘中。

19.Object 类的 equal 方法和 hashcode 方法的重写

equal 和 hashCode 的关系是这样的：（1）如果两个对象相同（即用 equal 比较返回 true），那么他们的 hashcode 值一定要相同；（2）如果两个对象的

hashcode 相同，他们并不一定相同（即用 equal 比较返回 false），因为 hashcode 的方法是可以重载的，如果不重载，会用 Java.long.Object 的 hashcode 方法，只要是不同的对象，hashcode 必然不同。

由于为了提高程序的效率才实现了 hashcode 方法，先进行 hashcode 的比较，如果不同，就没有必要再进行 equal 的比较了，这样就大大减少了 equals 比较的次数，这对需要比较数量很大的运算效率特稿是很多的。

附加：一旦 new 对象就会在内存中开辟空间，==比较的是对象的地址值，返回的是 boolean 型，equal 方法默认比较的对象的地址值，但是 Integer 等基本类型包装类以及 String 类中已经重写了 equal（）方法，比较的是对象内存中的内容，返回值是 boolean 型。

6.Java 中的线程的 run( )方法和 start（）方法的区别

start()方法被用来启动新创建的线程，而且 start( )内部调用了 run ( )方法，这和直接调用 run（ ）方法的效果不同，当调用 run（ ）方法时，只会是在原来的线程中调用，没有新的线程启动，只有 start( )方法才会启动新线程。

31.Java 中 wait 和 sleep 方法的不同

Java 程序中 wait 和 sleep 都会造成某种形式的暂停，sleep（）方法属于 Thread 类中，而 wait( )方法属于 Object 类中，sleep( )方法是让程序暂停执行指定的时间，释放 CPU 资源，但不会释放锁，线程的监控状态依然保持着，当指定的时间到了又会自动恢复运行状态，而当调用 wait( )方法的时候，线程会放弃对象锁，进入等待此对象的等待锁定池，只有针对此对象调用 notify（）方法后线程才进入对象锁定池准备，获取对象锁进入运行状态。

7.对 Java 中 wait/notify 关键字的理解

wait()、notify（）、notifyAll(　　)都不属于 Thread 类，而是属于 Object 基础类，也就是每个对象都有 wait(　)、notify（）、notifyAll(　)的功能，因为每个对象都有锁，锁是每个对象的基础。

wait（）:会把持有该对象线程的对象控制权交出去，然后处于等待状态。

notify（）:会通知某个正在等待这个对象的控制权的线程可以运行。

notifyAll(　)：会通知所有等待这个对象的控制权的线程继续运行，如果有多个正在等待该对象控制权时，具体唤醒哪个线程，就由操作系统进行调度。

8.Java 中 Synchronized 关键字的用法，以及对象锁、方法锁、类锁的理解

Java 的内置锁：每个 Java 对象都可以用作一个实现同步的锁，这些锁称为内置锁，线程进入同步代码块或方法时，会自动获得该锁，在退出同步代码块或方法时会释放该锁，获得内置锁的唯一途径就是进入这个锁的保护的同步代码块或方法。

Java 的内置锁是一个互斥锁，这即意味着最多只有一个线程获得该锁，当线程 A 尝试去获得线程 B 持有的内置锁时，线程 A 必须等待或阻塞，直到线程 B 释放该锁，如果 B 线程不释放该锁，那么 A 线程将一直等待下去。

Java 的对象锁与类锁，对象锁是用于实例方法的，或者一个对象实例上的，类锁是用于类的静态方法或者一个类的 class 对象上的，类的对象实例可以有很多个，但每个类只有一个 class 对象，所以不同对象的实例的对象锁是互不干扰，但是每个类只有一个类锁。

Synchronized 的用法：Synchronized 修饰方法和修饰代码块。

9.Synchronized 和 volatile 关键字的区别

Volatile 在本质上是告诉 JVM 当前变量在寄存器（工作内存）中的值是不确

定的，需要从主存中读取；Synchronized 则是锁定当前变量，只有当前线程可以访问该变量，其他线程会被阻塞。

Volatile 仅能使用在变量级别，synchronized 则可以使用在变量、方法和类级别。

Volatile 仅能修改变量的可见性，不能保证原子性，而 synchronized 则可以保证变量的修改可见性和原子性。

Volatile 不会造成线程的阻塞，synchronized 可能会造成线程的阻塞。

Volatile 标记的变量不会被编译器优化，synchronized 标记的变量可以被编译器优化。

10.ReentrantLock、Synchronized、Volatile 关键字

Synchronized:即互斥锁，即操作互斥，并发线程，串行得到锁，串行执行代码，就像一个房间一把钥匙，一个人进去后，下一个人必须等到第一个人出来得到钥匙才能进去；

ReetrantLock:可重入锁，和同步锁功能类似，不过需要显性的创建和销毁，其特点在于①ReentrantLock 有 try　Lock 方法，如果锁被其他线程持有，返回 false，可以避免形成死锁，②创建时可自定义是可抢占的，③ReentrantReadWriteLock,用于读多写少，且不需要互斥的场景大大提高性能；

Volatile：只保证同意变量在多线程中的可见，他会强制将对缓存的修改操作立即写入主存，如果是写操作，会导致其他 CPU 对应的缓存无效；