Android 面试常见 58 题

1、java 中==和 equals 和 hashCode 的区别

基本数据类型的==比较的值相等.

类的==比较的内存的地址，即是否是同一个对象，在不覆盖 equals 的情况下，同比较内存地址，原实现也为 == ，如 String 等重写了 equals 方法.

hashCode 也是 Object 类的一个方法。返回一个离散的 int 型整数。在集合类操作中使用，为了提高查询速度。（HashMap，HashSet 等比较是否为同一个）

如果两个对象 equals，Java 运行时环境会认为他们的 hashcode 一定相等。

如果两个对象不 equals，他们的 hashcode 有可能相等。

如果两个对象 hashcode 相等，他们不一定 equals。

如果两个对象 hashcode 不相等，他们一定不 equals。

2、int 与 integer 的区别

int 基本类型

integer 对象 int 的封装类

3、String、StringBuffer、StringBuilder 区别

String:字符串常量 不适用于经常要改变值得情况，每次改变相当于生成一个新的对象

StringBuffer:字符串变量 （线程安全）

StringBuilder:字符串变量（线程不安全） 确保单线程下可用，效率略高于 StringBuffer

4、什么是内部类？内部类的作用

内部类可直接访问外部类的属性

Java 中内部类主要分为**成员内部类**、**局部内部类**(嵌套在方法和作用域内)、**匿名内部类**（没构造方法）、**静态内部类**（static 修饰的类，不能使用任何外围类的非 static 成员变量和方法， 不依赖外围类）

5、进程和线程的区别

进程是 cpu 资源分配的最小单位，线程是 cpu 调度的最小单位。

进程之间不能共享资源，而线程共享所在进程的地址空间和其它资源。

一个进程内可拥有多个线程，进程可开启进程，也可开启线程。

一个线程只能属于一个进程，线程可直接使用同进程的资源,线程依赖于进程而存在。

6、final，finally，finalize 的区别

final: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:类的方法,垃圾回收之前会调用此方法,子类可以重写 finalize()方法实现对资源的回收

7、Serializable 和 Parcelable 的区别

Serializable Java 序列化接口 在硬盘上读写 读写过程中有大量临时变量的生成，内部执行大量的 i/o 操作，效率很低。

Parcelable Android 序列化接口 效率高 使用麻烦 在内存中读写（AS 有相关插件一键生成所需方法），对象不能保存到磁盘中

8、静态属性和静态方法是否可以被继承？是否可以被重写？以及原因？

可继承 不可重写 而是被隐藏

如果子类里面定义了静态方法和属性，那么这时候父类的静态方法或属性称之为"隐藏"。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。

9、成员内部类、静态内部类、局部内部类和匿名内部类的理解，以及项目中的应用

ava 中内部类主要分为**成员内部类**、**局部内部类**(嵌套在方法和作用域内)、**匿名内部类**（没构造方法）、**静态内部类**（static 修饰的类，不能使用任何外围类的非 static 成员变量和方法， 不依赖外围类）

使用内部类最吸引人的原因是：每个内部类都能独立地继承一个（接口的）实现，所以无论外围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。

因为 Java 不支持多继承，支持实现多个接口。但有时候会存在一些使用接口很难解决的问题，这个时候我们可以利用内部类提供的、可以继承多个具体的或者抽象的类的能力来解决这些程序设计问题。可以这样说，接口只是解决了部分问题，而内部类使得多重继承的解决方案变得更加完整。

10、string 转换成 integer 的方式及原理

String 到 integer Intrger.parseInt(string);

Integer 到 string Integer.toString();

11、哪些情况下的对象会被垃圾回收机制处理掉？

1.所有实例都没有活动线程访问。

2.没有被其他任何实例访问的循环引用实例。

3.Java 中有不同的引用类型。判断实例是否符合垃圾收集的条件都依赖于它的引用类型。

要判断怎样的对象是没用的对象。这里有 2 种方法：

1.采用标记计数的方法：

给内存中的对象给打上标记，对象被引用一次，计数就加 1，引用被释放了，计数就减一，当这个计数为 0 的时候，这个对象就可以被回收了。当然，这也就引发了一个问题：循环引用的对象是无法被识别出来并且被回收的。所以就有了第二种方法：

2.采用根搜索算法：

从一个根出发，搜索所有的可达对象，这样剩下的那些对象就是需要被回收的

12、静态代理和动态代理的区别，什么场景使用？

静态代理类：

由程序员创建或由特定工具自动生成源代码，再对其编译。在程序运行前，代理类的.class 文件就已经存在了。动态代理类：在程序运行时，运用反射机制动态创建而成。

14、Java 中实现多态的机制是什么？

答：方法的重写 Overriding 和重载 Overloading 是 Java 多态性的不同表现

重写 Overriding 是父类与子类之间多态性的一种表现

重载 Overloading 是一个类中多态性的一种表现.

16、说说你对 Java 反射的理解

JAVA 反射机制是在运行状态中，对于任意一个类，都能够知道这个类的所有属性和方法；对于任意一个对象，都能够调用它的任意一个方法和属性。从对象出发，通过反射（Class 类）可以取得取得类的完整信息（类名 Class 类型，所在包、具有的所有方法 Method[]类型、某个方法的完整信息（包括修饰符、返回值类型、异常、参数类型）、所有属性 Field[]、某个属性的完整信息、构造器 Constructors），调用类的属性或方法自己的总结： 在运行过程中获得类、对象、方法的所有信息。

17、说说你对 Java 注解的理解

元注解

元注解的作用就是负责注解其他注解。java5.0 的时候，定义了 4 个标准的 meta-annotation 类型，它们用来提供对其他注解的类型作说明。

1.@Target

2.@Retention

3.@Documented

4.@Inherited

18、Java 中 String 的了解

在源码中 string 是用 final 进行修饰，它是不可更改，不可继承的常量。

19、String 为什么要设计成不可变的？

1、字符串池的需求

字符串池是方法区（Method Area）中的一块特殊的存储区域。当一个字符串已经被创建并且该字符串在 池 中，该字符串的引用会立即返回给变量，而不是重新创建一个字符串再将引用返回给变量。如果字符串不是不可变的，那么改变一个引用（如: string2）的字符串将会导致另一个引用（如: string1）出现脏数据。

2、允许字符串缓存哈希码

在 java 中常常会用到字符串的哈希码，例如： HashMap 。String 的不变性保证哈希码始终一，因此，他可以不用担心变化的出现。 这种方法意味着不必每次使用时都重新计算一次哈希码——这样，效率会高很多。

3、安全

String 广泛的用于 java 类中的参数，如：网络连接（Network connetion），打开文件（opening files ）等等。如果 String 不是不可变的，网络连接、文件将会被改变——这将会导致一系列的安全威胁。操作的方法本以为连接上了一台机器，但实际上却不是。由于反射中的参数都是字符串，同样，也会引起一系列的安全问题。

20、Object 类的 equal 和 hashCode 方法重写，为什么？

首先 equals 与 hashcode 间的关系是这样的：

1、如果两个对象相同（即用 equals 比较返回 true），那么它们的 hashCode 值一定要相同；

2、如果两个对象的 hashCode 相同，它们并不一定相同(即用 equals 比较返回 false)

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，那没就不必在进行 equals 的比较了，这样就大大减少了 equals 比较的次数，这对比需要比较的数量很大的效率提高是很明显的

21、List,Set,Map 的区别

Set 是最简单的一种集合。集合中的对象不按特定的方式排序，并且没有重复对象。 Set 接口主要实现了两个实现类：HashSet： HashSet 类按照哈希算法来存取集合中的对象，存取速度比较快

TreeSet ：TreeSet 类实现了 SortedSet 接口，能够对集合中的对象进行排序。

List 的特征是其元素以线性方式存储，集合中可以存放重复对象。

ArrayList() : 代表长度可以改变得数组。可以对元素进行随机的访问，向 ArrayList() 中插入与删除元素的速度慢。

LinkedList(): 在实现中采用链表数据结构。插入和删除速度快，访问速度慢。

Map 是一种把键对象和值对象映射的集合，它的每一个元素都包含一对键对象和值对象。 Map 没有继承于 Collection 接口 从 Map 集合中检索元素时，只要给出键对象，就会返回对应的值对象。

HashMap：Map 基于散列表的实现。插入和查询“键值对”的开销是固定的。可以通过构造器设置容量 capacity 和负载因子 load factor，以调整容器的性能。

LinkedHashMap： 类似于 HashMap，但是迭代遍历它时，取得“键值对”的顺序是其插入次序，或者是最近最少使用(LRU)的次序。只比 HashMap 慢一点。而在迭代访问时发而更快，因为它使用链表维护内部次序。

TreeMap ： 基于红黑树数据结构的实现。查看“键”或“键值对”时，它们会被排序(次序由 Comparabel 或 Comparator 决定)。TreeMap 的特点在 于，你得到的结果是经过排序的。TreeMap 是唯一的带有 subMap()方法的 Map，它可以返回一个子树。

WeakHashMao ：弱键(weak key)Map，Map 中使用的对象也被允许释放: 这是为解决特殊问题设计的。如果没有 map 之外的引用指向某个“键”，则此“键”可以被垃圾收集器回收。

26、ArrayMap 和 HashMap 的对比

1、存储方式不同

HashMap 内部有一个 HashMapEntry<K, V>[]对象，每一个键值对都存储在这个对象里，当使用 put 方法添加键值对时，就会 new 一个 HashMapEntry 对象，

2、添加数据时扩容时的处理不一样，进行了 new 操作，重新创建对象，开销很大。ArrayMap 用的是 copy 数据，所以效率相对要高。

3、ArrayMap 提供了数组收缩的功能，在 clear 或 remove 后，会重新收缩数组，是否空间

4、ArrayMap 采用二分法查找；

29、HashMap 和 HashTable 的区别

1 HashMap 不是线程安全的，效率高一点、方法不是 Synchronize 的要提供外同步，有 containsvalue 和 containsKey 方法。

hashtable 是，线程安全，不允许有 null 的键和值，效率稍低，方法是是 Synchronize 的。有 contains 方法方法。Hashtable 继承于 Dictionary 类

30、HashMap 与 HashSet 的区别

hashMap:HashMap 实现了 Map 接口,HashMap 储存键值对,使用 put()方法将元素放入 map 中,HashMap 中使用键对象来计算 hashcode 值,HashMap 比较快，因为是使用唯一的键来获取对象。

HashSet 实现了 Set 接口，HashSet 仅仅存储对象，使用 add()方法将元素放入 set 中，HashSet 使用成员对象来计算 hashcode 值，对于两个对象来说 hashcode 可能相同，所以 equals()方法用来判断对象的相等性，如果两个对象不同的话，那么返回 false。HashSet 较 HashMap 来说比较慢。

31、HashSet 与 HashMap 怎么判断集合元素重复？

HashSet 不能添加重复的元素，当调用 add（Object）方法时候，

首先会调用 Object 的 hashCode 方法判 hashCode 是否已经存在，如不存在则直接插入元素；如果已存在则调用 Object 对象的 equals 方法判断是否返回 true，如果为 true 则说明元素已经存在，如为 false 则插入元素。

33、ArrayList 和 LinkedList 的区别，以及应用场景

ArrayList 是基于数组实现的，ArrayList 线程不安全。

LinkedList 是基于双链表实现的：

使用场景：

（1）如果应用程序对各个索引位置的元素进行大量的存取或删除操作，ArrayList 对象要远优于 LinkedList 对象；

(2) 如果应用程序主要是对列表进行循环，并且循环时候进行插入或者删除操作，LinkedList 对象要远优于 ArrayList 对象；

34、数组和链表的区别

数组：是将元素在内存中连续存储的；它的优点：因为数据是连续存储的，内存地址连续，所以在查找数据的时候效率比较高；它的缺点：在存储之前，我们需要申请一块连续的内存空间，并且在编译的时候就必须确定好它的空间的大小。在运行的时候空间的大小是无法随着你的需要进行增加和减少而改变的，当数据两比较大的时候，有可能会出现越界的情况，数据比较小的时候，又有可能会浪费掉内存空间。在改变数据个数时，增加、插入、删除数据效率比较低。

链表：是动态申请内存空间，不需要像数组需要提前申请好内存的大小，链表只需在用的时候申请就可以，根据需要来动态申请或者删除内存空间，对于数据增加和删除以及插入比数组灵活。还有就是链表中数据在内存中可以在任意的位置，通过应用来关联数据（就是通过存在元素的指针来联系）

35、开启线程的三种方式？

ava 有三种创建线程的方式，分别是继承 Thread 类、实现 Runable 接口和使用线程池

36、线程和进程的区别？

线程是进程的子集，一个进程可以有很多线程，每条线程并行执行不同的任务。不同的进程使用不同的内存空间，而所有的线程共享一片相同的内存空间。别把它和栈内存搞混，每个线程都拥有单独的栈内存用来存储本地数据。

38、run()和 start()方法区别

这个问题经常被问到，但还是能从此区分出面试者对 Java 线程模型的理解程度。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()方法的效果不一样。当你调用 run()方法的时候，只会是在原来的线程中调用，没有新的线程启动，start()方法才会启动新线程。

39、如何控制某个方法允许并发访问线程的个数？

semaphore.acquire() 请求一个信号量，这时候的信号量个数-1（一旦没有可使用的信号量，也即信号量个数变为负数时，再次请求的时候就会阻塞，直到其他线程释放了信号量）

semaphore.release() 释放一个信号量，此时信号量个数+1

40、在 Java 中 wait 和 seelp 方法的不同；

Java 程序中 wait 和 sleep 都会造成某种形式的暂停，它们可以满足不同的需要。wait()方法用于线程间通信，如果等待条件为真且其它线程被唤醒时它会释放锁，而 sleep()方法仅仅释放 CPU 资源或者让当前线程停止执行一段时间，但不会释放锁。

41、谈谈 wait/notify 关键字的理解

等待对象的同步锁,需要获得该对象的同步锁才可以调用这个方法,否则编译可以通过，但运行时会收到一个异常：IllegalMonitorStateException。

调用任意对象的 wait() 方法导致该线程阻塞，该线程不可继续执行，并且该对象上的锁被释放。

唤醒在等待该对象同步锁的线程(只唤醒一个,如果有多个在等待),注意的是在调用此方法的时候，并不能确切的唤醒某一个等待状态的线程，而是由 JVM 确定唤醒哪个线程，而且不是按优先级。

调用任意对象的 notify()方法则导致因调用该对象的 wait()方法而阻塞的线程中随机选择的一个解除阻塞（但要等到获得锁后才真正可执行）。

42、什么导致线程阻塞？线程如何关闭？

阻塞式方法是指程序会一直等待该方法完成期间不做其他事情，ServerSocket 的 accept()方法就是一直等待客户端连接。这里的阻塞是指调用结果返回之前，当前线程会被挂起，直到得到结果之后才会返回。此外，还有异步和非阻塞式方法在任务完成前就返回。

一种是调用它里面的 stop()方法

另一种就是你自己设置一个停止线程的标记 （推荐这种）

43、如何保证线程安全？

1.synchronized；

2.Object 方法中的 wait,notify；

3.ThreadLocal 机制 来实现的。

44、如何实现线程同步？

1、synchronized 关键字修改的方法。2、synchronized 关键字修饰的语句块 3、使用特殊域变量（volatile）实现线程同步

45、线程间操作 List

List list = Collections.synchronizedList(new ArrayList());

46、谈谈对 Synchronized 关键字，类锁，方法锁，重入锁的理解

java 的对象锁和类锁：java 的对象锁和类锁在锁的概念上基本上和内置锁是一致的，但是，两个锁实际是有很大的区别的，对象锁是用于对象实例方法，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。我们知道，类的对象实例可以有很多个，但是每个类只有一个 class 对象，所以不同对象实例的对象锁是互不干扰的，但是每个类只有一个类锁。但是有一点必须注意的是，其实类锁只是一个概念上的东西，并不是真实存在的，它只是用来帮助我们理解锁定实例方法和静态方法的区别的

49、synchronized 和 volatile 关键字的区别

1.volatile 本质是在告诉 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，其他线程被阻塞住。

2.volatile 仅能使用在变量级别；synchronized 则可以使用在变量、方法、和类级别的

3.volatile 仅能实现变量的修改可见性，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

4.volatile 不会造成线程的阻塞；synchronized 可能会造成线程的阻塞。

5.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

51、ReentrantLock 、synchronized 和 volatile 比较

ava 在过去很长一段时间只能通过 synchronized 关键字来实现互斥，它有一些缺点。比如你不能扩展锁之外的方法或者块边界，尝试获取锁时不能中途取消等。Java 5 通过 Lock 接口提供了更复杂的控制来解决这些问题。 ReentrantLock 类实现了 Lock，它拥有与 synchronized 相同的并发性和内存语义且它还具有可扩展性。

53、死锁的四个必要条件？

死锁产生的原因

1. 系统资源的竞争

系统资源的竞争导致系统资源不足，以及资源分配不当，导致死锁。

2. 进程运行推进顺序不合适

互斥条件：一个资源每次只能被一个进程使用，即在一段时间内某 资源仅为一个进程所占有。此时若有其他进程请求该资源，则请求进程只能等待。

请求与保持条件：进程已经保持了至少一个资源，但又提出了新的资源请求，而该资源 已被其他进程占有，此时请求进程被阻塞，但对自己已获得的资源保持不放。

不可剥夺条件:进程所获得的资源在未使用完毕之前，不能被其他进程强行夺走，即只能 由获得该资源的进程自己来释放（只能是主动释放)。

循环等待条件: 若干进程间形成首尾相接循环等待资源的关系

这四个条件是死锁的必要条件，只要系统发生死锁，这些条件必然成立，而只要上述条件之一不满足，就不会发生死锁。

死锁的避免与预防：

死锁避免的基本思想:

系统对进程发出每一个系统能够满足的资源申请进行动态检查,并根据检查结果决定是否分配资源,如果分配后系统可能发生死锁,则不予分配,否则予以分配。这是一种保证系统不进入死锁状态的动态策略。

理解了死锁的原因，尤其是产生死锁的四个必要条件，就可以最大可能地避免、预防和解除死锁。所以，在系统设计、进程调度等方面注意如何让这四个必要条件不成立，如何确定资源的合理分配算法，避免进程永久占据系统资源。此外，也要防止进程在处于等待状态的情况下占用资源。因此，对资源的分配要给予合理的规划。

死锁避免和死锁预防的区别：

死锁预防是设法至少破坏产生死锁的四个必要条件之一,严格的防止死锁的出现,而死锁避免则不那么严格的限制产生死锁的必要条件的存在,因为即使死锁的必要条件存在,也不一定发生死锁。死锁避免是在系统运行过程中注意避免死锁的最终发生。

56、什么是线程池，如何使用?

创建线程要花费昂贵的资源和时间，如果任务来了才创建线程那么响应时间会变长，而且一个进程能创建的线程数有限。为了避免这些问题，在程序启动的时候就创建若干线程来响应处理，它们被称为线程池，里面的线程叫工作线程。从 JDK1.5 开始，Java API 提供了 Executor 框架让你可以创建不同的线程池。比如单线程池，每次处理一个任务；数目固定的线程池或者是缓存线程池（一个适合很多生存期短的任务的程序的可扩展线程池）。

57、Java 中堆和栈有什么不同？

为什么把这个问题归类在多线程和并发面试题里？因为栈是一块和线程紧密相关的内存区域。每个线程都有自己的栈内存，用于存储本地变量，方法参数和栈调用，一个线程中存储的变量对其它线程是不可见的。而堆是所有线程共享的一片公用内存区域。对象都在堆里创建，为了提升效率线程会从堆中弄一个缓存到自己的栈，如果多个线程使用该变量就可能引发问题，这时 volatile 变量就可以发挥作用了，它要求线程从主存中读取变量的值。

58、有三个线程 T1，T2，T3，怎么确保它们按顺序执行？

在多线程中有多种方法让线程按特定顺序执行，你可以用线程类的 join()方法在一个线程中启动另一个线程，另外一个线程完成该线程继续执行。为了确保三个线程的顺序你应该先启动最后一个(T3 调用 T2，T2 调用 T1)，这样 T1 就会先完成而 T3 最后完成。

线程间通信

我们知道线程是 CPU 调度的最小单位。在 Android 中主线程是不能够做耗时操作的，子线程是不能够更新 UI 的。而线程间通信的方式有很多，比如广播，Eventbus，接口回掉，在 Android 中主要是使用 handler。handler 通过调用 sendmessage 方法，将保存消息的 Message 发送到 Messagequeue 中，而 looper 对象不断的调用 loop 方法，从 messageueue 中取出 message，交给 handler 处理，从而完成线程间通信。