面试：

1. 介绍一下hashmap的原理
2. 初始大小16
3. 最大数量2的30次方
4. 负载因子0.75
5. 1.8jdk后使用数组 链表 红黑树 链表大于8就是用红黑树
6. 首先计算key的hash值，通过hash值定位数组中的位置i，判断i是否为空，若为空创建entry，加入链表；否则遍历entry，查找是否存在key的hash值相同并且key值一样的元素，若是替换value值；否则创建entry加入链表。结束
7. java IO流使用的设计模式

装饰者模式：动态给一个对象添加一些额外的职责,就象在墙上刷油漆.使用Decorator模式相比用生成子类方式达到功能的扩充显得更为灵活。

1. 多线程

1)Activity.runOnUiThread(Runnable)

2)View.post(Runnable) ;View.postDelay(Runnable , long)

3)Handler

4)AsyncTask：public abstract class AsyncTask<Params, Progress, Result> “启动任务执行的输入参数”、“后台任务执行的进度”、“后台计算结果的类型”

1.**execute(Params... params)**，执行一个异步任务，需要我们在代码中调用此方法，触发异步任务的执行。

2.**onPreExecute()**，在execute(Params... params)被调用后立即执行，一般用来在执行后台任务前对UI做一些标记。

3.**doInBackground(Params... params)**，在onPreExecute()完成后立即执行，用于执行较为费时的操作，此方法将接收输入参数和返回计算结果。在执行过程中可以调用publishProgress(Progress... values)来更新进度信息。

4.**onProgressUpdate(Progress... values)**，在调用publishProgress(Progress... values)时，此方法被执行，直接将进度信息更新到UI组件上。

5.**onPostExecute(Result result)**，当后台操作结束时，此方法将会被调用，计算结果将做为参数传递到此方法中，直接将结果显示到UI组件上。

1. Git常用命令：

Git help查看命令的help、

Git show 显示某次提交内容

1. git和svn的区别
2. svn是集中化版本控制系统 ；git是分布式版本控制系统
3. git使用元数据存储 svn使用文件存储
4. 最大特征：svn有全局版本号 git没有
5. Git使用sha-1算法存储 保障完整性
6. Git无网可以查看log svn没网就废了
7. 线程和进程的关系

进程是系统进行资源分配和调度的独立单位。

线程是进程的一个实体，是cpu调度和分派的基本单位。

**1) 简而言之,一个程序至少有一个进程,一个进程至少有一个线程.**

2) 线程的划分尺度小于进程，使得多线程程序的并发性高。

3) 另外，进程在执行过程中拥有独立的内存单元，而多个线程共享内存，从而极大地提高了程序的运行效率。

4) 线程在执行过程中与进程还是有区别的。每个独立的线程有一个程序运行的入口、顺序执行序列和程序的出口。**但是线程不能够独立执行，**必须依存在应用程序中，由应用程序提供多个线程执行控制。

5) 从逻辑角度来看，多线程的意义在于一个应用程序中，有多个执行部分可以同时执行。但操作系统并没有将多个线程看做多个独立的应用，来实现进程的调度和管理以及资源分配。**这就是进程和线程的重要区别。**

1. **java虚拟机和dalvik虚拟机**
2. **jvm基于栈 dalvik基于寄存器**
3. **jvn运行的是java字节码，dalvik运行的是。Dex文件**
4. **运行环境Dalvik经过优化，允许在有限的内存中同时运行多个虚拟机的实例，并且每一个 Dalvik应用作为一个独立的Linux进程执行。**
5. 安卓在4.4中发布了ART运行时（预编译）安装时dex转换成目标机械码
6. 面向对象的认识
7. 继承
8. 封装
9. 多态
10. 五大基本原则：单一原则，开闭原则，里氏替换原则，本木倒置原则，接口隔离原则
11. **wait和sleep的区别**
12. **wait来自thread类，sleep来自object类**
13. **wait释放锁，释放资源，sleep不释放资源**
14. wait，notify和notifyAll只能使用在同步块和同步方法中，sleep可以是在任何地方。
15. sleep需要捕捉异常，wait不需要。
16. 常见的内存泄漏memory leak
17. **Handler 引起的内存泄漏。**

**二、单例模式引起的内存泄漏**

**三、非静态内部类创建静态实例引起的内存泄漏**

**四、非静态匿名内部类引起的内存泄漏**

**五、注册/反注册未成对使用引起的内存泄漏**

**六、资源对象没有关闭引起的内存泄漏**

**七、集合对象没有及时清理引起的内存泄漏**

**11、android触发anr的原因和解决方法**

**1）主线程进行了耗时操作（网络访问 数据库操作等）**

**2）其他进程占用cpu导致本进程得不到时间片 。**

**解决**

**1.避免在主线程执行耗时操作，所有耗时操作应新开一个子线程完成，然后再在主线程更新UI。**

**2.BroadcastReceiver要执行耗时操作时应启动一个service，将耗时操作交给service来完成。**

**3.避免在Intent Receiver里启动一个Activity，因为它会创建一个新的画面，并从当前用户正在运行的程序上抢夺焦点。如果你的应用程序在响应Intent广 播时需要向用户展示什么，你应该使用Notification Manager来实现。**

**12、gc回收算法**

**1）引用计数器回收**

**2）标注清理**

**3）标注整理**

**4）拷贝算法**

**5）逐代回收**

**Android使用的是标注清理加上拷贝gc**

**14、http消息结构**

**1）请求**

**请求行 请求头部 空行 请求数据**

**2）响应**

**状态行 消息头部 空行 响应数据**

**15、java堆栈**

**栈主要存放基本类型的变量和对象的引用**

**堆主要存放数组和new的对象。**

**16、基本数据类型：int float double long byte shout bealoon char**

**17、tcp和udp的区别**

**TCP是面向连接的，可靠地，有序的**

**UDP是面向数据的，不可靠的，无序的**

**18、Linux跨进程通信机制**

**1.管道（Pipe）及有名管道（named pipe）**

**2.信号（Signal）**

**3.报文（Message）队列（消息队列）**

**4.共享内存**

**5.信号量（semaphore）**

**6.套接字（Socket）**

**Binder特性:驱动程序推动程序，共享内存提高性能，为进程请求分配每个进程的线程池。**

**针对系统中的对象引入了引用计数和跨进程的对象引用映射。**

**进程间同步调用。**

**19、两种序列化**

**Serializable，**[**Parcelable**](https://www.jianshu.com/p/82e3090e00e0)

**前者通过磁盘以及通过反射进行序列化和反序列化**

**后者是内存的序列化以及反序列化**