面试总结

Scott

目　录

空白目录

自我介绍

自我介绍

非常感谢你们给我这个机会，能够参加这次面试。

我叫汪林，毕业于15年,在14年的时候曾协助并指导班上同学，共同完成了一套酒店管理系统(Android + Java web)，可能正是因为这个经历,我才能够提前进入武汉的夏宇信息实习。

15年3月份来到深圳,然后就职于深圳德杰美格斯,在这家公司的主要成有公司网页的制作, Android Launcher应用、Android普通App的开发,和固件工程师调试Firmware程序(比如,验证BLE设备的firmware程序问题)。

16年五月的时候进入了诺尔信息，在这家公司,是以Android开发进去的,进去之后学习iOS开发,然后负责的事情也很多,也比较杂,Android、iOS App的开发,和客户沟通App程序的需求,搭建简单的服务器提供给固件工程师(因为除了蓝牙BLE也还有Wi-Fi方面的设备,他们也需要联网),最终的话就是App的上架和管理，以及iOS证书管理，最终在离职整理的时候，整理了一下大概有二十多个,但是还在运行的就只有四五个项目，还在优化或者维护更新，这些大概就是我的一些经历,也比较适合你们公司

第一家公司,Wi-Fi方面的设备多一点, 在第二家公司基本上都是蓝牙方面的项目, 我看了下你们公司业务方向,比较适合

薪资问题:贵公司除了薪资还有什么其他福利吗?
第一: 上家公司的薪资是14k，
第二: 经过这两年的成长,而且目前正在学习跨平台方面的事情,在上一家公司也写过相关的蓝牙测试程序.
所以我期望能够在18k左右

其他问题:
1. 公司的晋升机制是什么样的?
2. 上班的时间
3. 社保公积金缴纳

为什么离职
其实那个公司挺好的,在工作上没有什么发展空间,不符合我的发展规划，在那个公司一直都是一个人开发,给他们写测试程序,在学习上和工作上也得不到发展。

Android面试题

Android多线程

Android中的多线程本质上也是Java的多线程，同时添加了一些不同的特性和使用的场景。其中，最主要的一个区别就是Android中主线程和子线程中的区分
Android中的主线程是UI线程，负责运行四大组件并与用户实现交互，需要保持较高的反应速度，所以主线程不允许进行耗时的操作（比如说网络请求和访问），否则容易出现ANR现象，子线程则负责处理 一些耗时的任务，而如果子线程中想要实现对UI的操作，则需要通过Android的handler消息机制。

面试题总结

Handler

Handler是什么

handler是android中提供用来更新UI的一套机制,也是一套消息处理机制,我们可以发送消息,也可以通过它处理消息

为什么要使用handler

更新UI，

handler怎么使用

Looper 死循环为什么不会导致应用卡死？

线程默认没有Looper的，如果需要使用Handler就必须为线程创建Looper。我们经常提到的主线程，也叫UI线程，它就是ActivityThread，ActivityThread被创建时就会初始化Looper，这也是在主线程中默认可以使用Handler的原因。

在子线程中，如果手动为其创建了Looper，那么在所有的事情完成以后应该调用quit方法来终止消息循环，否则这个子线程就会一直处于等待（阻塞）状态，而如果退出Looper以后，这个线程就会立刻（执行所有方法并）终止，因此建议不需要的时候终止Looper。

网络请求框架

OkHttp

是一款用于 Android 和 Java 的网络请求库，也是目前 Android 中最火的一个网络库。OkHttp 有很多的优点

1. 在 HTTP/2 上允许对同一个 host 的请求共同一个 socket
2. 连接池的使用减少请求延迟（如果 HTTP/2 不支持）
3. 透明的 GZIP 压缩减少数据量大小
4. 响应的缓存避免重复的网络请求

Retrofit 底层其实就是用的 OkHttp 去请求网络。本文分析 OKHttp 的源码，主要是针对一次网络请求的基本流程，

图片处理框架Picasso，Glide

加载大图

一种情况

1. 对于图片显示：根据需要显示图片控件的大小对图片进行压缩显示。
2. 果图片数量非常多：则会使用LruCache等缓存机制，将所有图片占据的内容维持在一个范围内

另一种情况

单个图片非常巨大，并且还不允许压缩
首先不压缩，按照原图尺寸加载，那么屏幕肯定是不够大的，并且考虑到内存的情况，不可能一次性整图加载到内存中，所以肯定是局部加载，那么就需要用到一个类：
BitmapRegionDecoder

其次，既然屏幕显示不完，那么最起码要添加一个上下左右拖动的手势，让用户可以拖动查看。
去自定义一个显示巨图的View，支持用户去拖动查看

Glide 和 Picasso 可以说是目前 Android 上最流行的图片加载库了。大部分安卓应用开发人员都有使用过这两个库在他们的开发工作中。这两个库也都确实提供了大量图片加载的功能，而且也都经过了很多应用的检验，是可靠可信的。表面看上去似乎两者工作原理很相似，但是实际上是有着很大差别的，主要体现在下面几个方面：

1. 下载图片的方式
2. 图片的缓存机制
3. 加载到内存的机制

缓存大小

两个库也都支持缓存图片，都通过下载图片后，缓存到本地。但是这里对于缓存本地的机制，两个库是完全不同的做法。

Picasso 是下载图片然后缓存完整的大小到本地，比如说图片的大小是1080p的，之后如果我需要同一张图片，就会返回这张 full size 的，如果我需要resize，也是对这种 full size 的做 resize。

Glide 则是完全不一样的做法。Glide 是会先下载图片，然后改变图片的大小，以适应 imageView 的要求，然后缓存到本地。 所以如果你是下载同一张图片，但是设定两个不一样大小的 imageView, 那么 Glide 实际上是会缓存两份。
换个角度来看，这里不仅仅是缓存的问题，比如一个 ImageView 要改变它的大小，PIcasso 就只需要下载一次 full size 的图片，但是 Glide 实际上就不仅仅是下载一次了，它需要去单独下载然后改变大小适配 imageView，因为对于 Glide 来讲，需要缓存不同大小的同一张图片。

内存的使用

Glide 默认是用的 RGB_555 的设定，PIcasso 则是用的 ARGB _8888的设定。

加载图片的时间

当尝试加载一个图片的时候，两个库都会采用先从缓存中读取，如果缓存中没有，再去下载的做法。
实际试验中，Picasso 会比 Glide 快一点。猜测可能的原因还是因为之前讲到的缓存机制导致，因为 Picasso 是直接把图加载到内存中，而 Glide 则需要改变图片大小再加载到内存中去。这个应该是会耗费一定的时间。

但是，当加载图片从内存中的时候，Glide 则比 Picasso 要快。其原理还是因为缓存机制的区别。因为 Picasso 从缓存中拿到的图片，还要先去 resize 后，然后设定给 imageView，但是 Glide 则不需要这样。

其他功能的对比

1. GIF 支持：Glide 支持 GIF。 对于加载 GIF 来说，Glide 只需要简单使用 Glide.with(...).load(...)。 但是 Picasso 是不支持的，因此如果你的应用中是需要加载 GIF 的话，那就只能用 Glide 了。
2. 灵活性：Glide 提供了非常多的配置，你可以非常灵活的根据你的需求来客制化，从而缩减 Glide 库的大小等。

Glide和Picasso他们的对比的优缺点

1.Picasso和Glide的withi后面的参数不同

Picasso.with(这里只能传入上下文) .
Glide.with,后面可以传入上下文,activity实例,FragmentActivity实例,Fragement.传入的对象要比前者多.

2.加载后图片质量不同

Picasso采用的ARGB-8888,Glide采用的是RGB-565
相对而言,Picasso加载的是全图,图片质量和清晰对要比Glide的要高,但是,因为加载的采样率过高,导致,出现OOM异常的概率要比Glide要大很多.

3.加载Gif图片(备注:Gif图片消耗太对内存,尽量谨慎使用):

Picasso不能加载git图片
Glide可以加载缓存图片

4.缓存策略和加载速度.

Picasso缓存的是全尺寸,而 Glide的缓存的更ImageView的尺寸相同.
讲ImageView调整为不同的大小,不管大小如何设置,Picasso只缓存一个全尺寸的,Glide则不同,他会为每种

大小不一致的ImageView都缓存一次.
Glide的这个特点,让加载显得特别的快,而Picasso则因为需要在显示之前重新调整大小而导致一些延迟,(即便是添加了noFade)

5.总结:

Glide比Picasso加载速度要快,其实他是在Picasso的基础上进行了第二次封装,但是Glide比Picasso需要更多的空间来缓存;Glide加载图像以及磁盘缓存的方式,都优于Picasso,且Glide更有利于减少OutOfMemoryError的发生;
Gif动画,是Glide的杀手锏.

Android最佳性能实践OOM

合理管理内存

节制地使用Service

当界面不可见时释放内存

onTrimMemory

当内存紧张时释放内存

避免在Bitmap上浪费内存

使用优化过的数据集合

知晓内存的开支情况

谨慎使用抽象编程

尽量避免使用依赖注入框架

使用ProGuard简化代码

ProGuard相信大家都不会陌生，很多人都会使用这个工具来混淆代码，但是除了混淆之外，它还具有压缩和优化代码的功能。ProGuard会对我们的代码进行检索，删除一些无用的代码，并且会对类、字段、方法等进行重命名，重命名之后的类、字段和方法名都会比原来简短很多，这样的话也就对内存的占用变得更少了。

使用多个进程

这个技巧其实并不是非常建议使用，但它确实是一种可以帮助我们节省和管理内存的高级技巧。如果你要使用它的话一定要谨慎使用，因为绝大多数的应用程序都不应该在多个进程当中运行的，一旦使用不当，它甚至会增加额外的内存而不是帮我们节省内存。这个技巧比较适用于那些需要在后台去完成一项独立的任务，和前台的功能是可以完全区分开的场景。

这里举一个比较适合去使用多进程技巧的场景，比如说我们正在做一个音乐播放器软件，其中播放音乐的功能应该是一个独立的功能，它不需要和UI方面有任何关系，即使软件已经关闭了也应该可以正常播放音乐。如果此时我们只使用一个进程，那么即使用户关闭了软件，已经完全由Service来控制音乐播放了，系统仍然会将许多UI方面的内存进行保留。在这种场景下就非常适合使用两个进程，一个用于UI展示，另一个则用于在后台持续地播放音乐。

异步：RxJava，AsyncTask

RxJava

原理简析

RxJava 的异步实现，是通过一种扩展的观察者模式来实现的。
观察者模式

先简述一下观察者模式，已经熟悉的可以跳过这一段。

观察者模式面向的需求是：A 对象（观察者）对 B 对象（被观察者）的某种变化高度敏感，需要在 B 变化的一瞬间做出反应。举个例子，新闻里喜闻乐见的警察抓小偷，警察需要在小偷伸手作案的时候实施抓捕。在这个例子里，警察是观察者，小偷是被观察者，警察需要时刻盯着小偷的一举一动，才能保证不会漏过任何瞬间。程序的观察者模式和这种真正的『观察』略有不同，观察者不需要时刻盯着被观察者（例如 A 不需要每过 2ms 就检查一次 B 的状态），而是采用注册(Register)或者称为订阅(Subscribe)的方式，告诉被观察者：我需要你的某某状态，你要在它变化的时候通知我。 Android 开发中一个比较典型的例子是点击监听器 OnClickListener 。对设置 OnClickListener 来说， View 是被观察者，OnClickListener 是观察者，二者通过 setOnClickListener() 方法达成订阅关系。订阅之后用户点击按钮的瞬间，Android Framework 就会将点击事件发送给已经注册的 OnClickListener 。采取这样被动的观察方式，既省去了反复检索状态的资源消耗，也能够得到最高的反馈速度。当然，这也得益于我们可以随意定制自己程序中的观察者和被观察者，而警察叔叔明显无法要求小偷『你在作案的时候务必通知我』。

OnClickListener 的模式大致如下图：

如图所示，通过 `setOnClickListener()` 方法，`Button` 持有 `OnClickListener` 的引用（这一过程没有在图上画出）；当用户点击时，`Button` 自动调用 `OnClickListener` 的 `onClick()` 方法。另外，如果把这张图中的概念抽象出来（`Button` -> 被观察者、`OnClickListener` -> 观察者、`setOnClickListener()` -> 订阅，`onClick()` -> 事件），就由专用的观察者模式（例如只用于监听控件点击）转变成了通用的观察者模式。如下图：

而 RxJava 作为一个工具库，使用的就是通用形式的观察者模式。

RxJava 的观察者模式

RxJava 有四个基本概念：Observable (可观察者，即被观察者)、 Observer (观察者)、 subscribe (订阅)、事件。Observable 和 Observer 通过 subscribe() 方法实现订阅关系，从而 Observable 可以在需要的时候发出事件来通知 Observer。

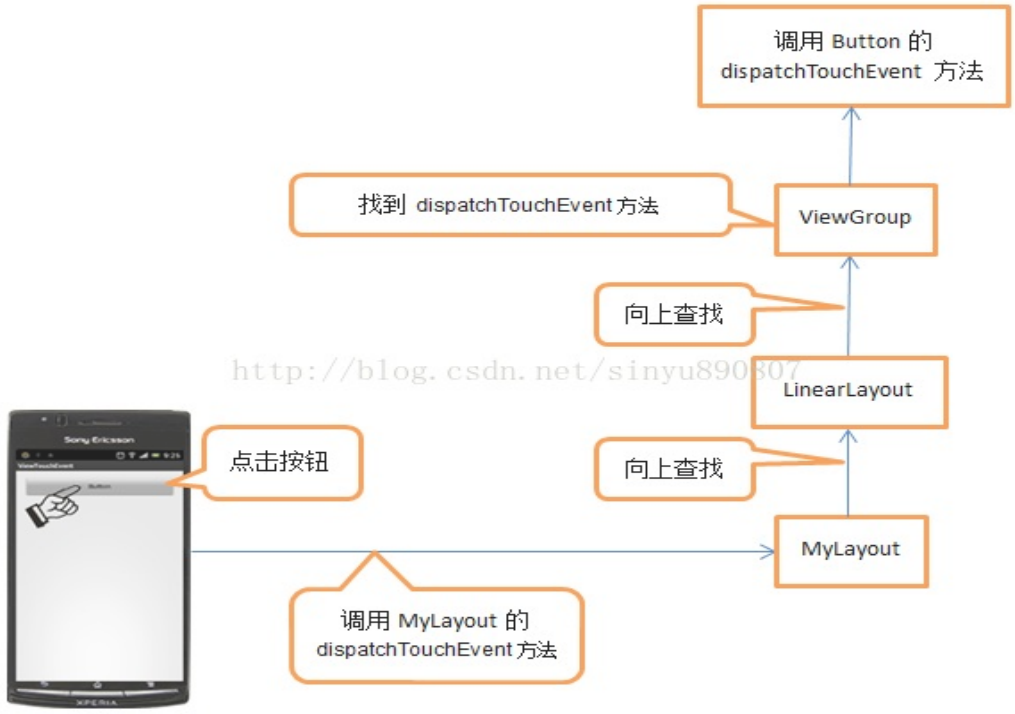
与传统观察者模式不同， RxJava 的事件回调方法除了普通事件 onNext() （相当于 onClick() / onEvent()）之外，还定义了两个特殊的事件：onCompleted() 和 onError()。

onCompleted(): 事件队列完结。RxJava 不仅把每个事件单独处理，还会把它们看做一个队列。RxJava 规定，当不会再有新的 onNext() 发出时，需要触发 onCompleted() 方法作为标志。
onError(): 事件队列异常。在事件处理过程中出异常时，onError() 会被触发，同时队列自动终止，不允许再有事件发出。
在一个正确运行的事件序列中, onCompleted() 和 onError() 有且只有一个，并且是事件序列中的最后一个。需要注意的是，onCompleted() 和 onError() 二者也是互斥的，即在队列中调用了其中一个，就不应该再调用另一个。

View，ViewGroup事件分发

事件分发

当你点击了某个控件，首先会去调用该控件所在布局的dispatchTouchEvent方法，然后在布局的dispatchTouchEvent方法中找到被点击的相应控件，再去调用该控件的dispatchTouchEvent方法。如果我们点击了MyLayout中的按钮，会先去调用MyLayout的dispatchTouchEvent方法，可是你会发现MyLayout中并没有这个方法。那就再到它的父类LinearLayout中找一找，发现也没有这个方法。那只好继续再找LinearLayout的父类ViewGroup，你终于在ViewGroup中看到了这个方法，按钮的dispatchTouchEvent方法就是在这里调用的



整个ViewGroup的事件分发流程

1. Android事件分发是先传递到ViewGroup，再由ViewGroup传递到View的。
2. 在ViewGroup中可以通过onInterceptTouchEvent方法对事件传递进行拦截，onInterceptTouchEvent方法返回true代表不允许事件继续向子View传递，返回false代表不对事件进行拦截，默认返回false。
3. 子View中如果将传递的事件消费掉，ViewGroup中将无法接收到任何事件。

http://blog.csdn.net/guolin_blog/article/details/9097463
http://blog.csdn.net/guolin_blog/article/details/9153747

消息传递：EventBus

EventBus

EventBus能够简化各组件间的通信，让我们的代码书写变得简单，能有效的分离事件发送方和接收方(也就是解耦的意思)，能避免复杂和容易出错的依赖性和生命周期问题。

关于EventBus的概述

三要素

1. Event 事件。它可以是任意类型。
2. Subscriber 事件订阅者。在EventBus3.0之前我们必须定义以onEvent开头的那几个方法，分别是onEvent、onEventMainThread、onEventBackgroundThread和onEventAsync，而在3.0之后事件处理的方法名可以随意取，不过需要加上注解@subscribe()，并且指定线程模型，默认是POSTING。
3. Publisher 事件的发布者。我们可以在任意线程里发布事件，一般情况下，使用EventBus.getDefault()就可以得到一个EventBus对象，然后再调用post(Object)方法即可。

四种线程模型

EventBus3.0有四种线程模型，分别是：

1. POSTING (默认) 表示事件处理函数的线程跟发布事件的线程在同一个线程。
2. MAIN 表示事件处理函数的线程在主线程(UI)线程，因此在这里不能进行耗时操作。
3. BACKGROUND 表示事件处理函数的线程在后台线程，因此不能进行UI操作。如果发布事件的线程是主线程(UI线程)，那么事件处理函数将会开启一个后台线程，如果果发布事件的线程是在后台线程，那么事件处理函数就使用该线程。
4. ASYNC 表示无论事件发布的线程是哪一个，事件处理函数始终会新建一个子线程运行，同样不能进行UI操作。

我们发现EventBus使用起来是如此的方便，当我们的代码量变得很多的时候，使用EventBus后你的逻辑非常的清晰，并且代码之间高度解耦，在进行组件、页面间通信的时候，EventBus是一个不错的选择。

HTTPS和HTTP的区别

什么是 HTTPS?

HTTPS (基于安全套接字层的超文本传输协议 或者是 HTTP over SSL) 是一个 Netscape 开发的 Web 协议。

你也可以说：HTTPS = HTTP + SSL

HTTPS 在 HTTP 应用层的基础上使用安全套接字层作为子层。

为什么需要 HTTPS ？

超文本传输协议 (HTTP) 是一个用来通过互联网传输和接收信息的协议。HTTP 使用请求/响应的过程，因此信息可在服务器间快速、轻松而且精确的进行传输。当你访问 Web 页面的时候你就是在使用 HTTP 协议，但 HTTP 是不安全的，可以轻松对窃听你跟 Web 服务器之间的数据传输。在很多情况下，客户和服务器之间传输的是敏感歇息，需要防止未经授权的访问。为了满足这个要求，网景公司(Netscape)推出了 HTTPS，也就是基于安全套接字层的 HTTP 协议。

HTTP 和 HTTPS 的相同点

大多数情况下，HTTP 和 HTTPS 是相同的，因为都是采用同一个基础的协议，作为 HTTP 或 HTTPS 客户端——浏览器，设立一个连接到 Web 服务器指定的端口。当服务器接收到请求，它会返回一个状态码以及消息，这个回应可能是请求信息、或者指示某个错误发送的错误信息。系统使用统一资源定位器 URI 模式，因此资源可以被唯一指定。而 HTTPS 和 HTTP 唯一不同的只是一个协议头(https)的说明，其他都是一样的。

HTTP 和 HTTPS 的不同之处

1. HTTP 的 URL 以 http:// 开头，而 HTTPS 的 URL 以 https:// 开头
2. HTTP 是不安全的，而 HTTPS 是安全的
3. HTTP 标准端口是 80 ，而 HTTPS 的标准端口是 443
   在 OSI 网络模型中，HTTP 工作于应用层，而 HTTPS 工作在传输层
4. HTTP 无需加密，而 HTTPS 对传输的数据进行加密
5. HTTP 无需证书，而 HTTPS 需要认证证书

HTTPS 如何工作?

使用 HTTPS 连接时，服务器要求有公钥和签名的证书。
当使用 https 连接，服务器响应初始连接，并提供它所支持的加密方法。作为回应，客户端选择一个连接

方法，并且客户端和服务器端交换证书验证彼此身份。完成之后，在确保使用相同密钥的情况下传输加密信息，然后关闭连接。为了提供 https 连接支持，服务器必须有一个公钥证书，该证书包含经过证书机构认证的密钥信息，大部分证书都是通过第三方机构授权的，以保证证书是安全的。

换句话说，HTTPS 跟 HTTP 一样，只不过增加了 SSL。

HTTP 包含如下动作：

浏览器打开一个 TCP 连接
浏览器发送 HTTP 请求到服务器端
服务器发送 HTTP 回应信息到浏览器
TCP 连接关闭

SSL 包含如下动作：

验证服务器端
允许客户端和服务器端选择加密算法和密码，确保双方都支持
验证客户端(可选)
使用公钥加密技术来生成共享加密数据
创建一个加密的 SSL 连接
基于该 SSL 连接传递 HTTP 请求

什么时候该使用 HTTPS?

银行网站、支付网关、购物网站、登录页、电子邮件以及一些企业部门的网站应该使用 HTTPS，例如：

PayPal: https://www.paypal.com
Google AdSense: https://www.google.com/adsense/
如果某个网站要求你填写信用卡信息，首先你要检查该网页是否使用 https 加密连接，如果没有，那么请不要输入任何敏感信息如信用卡号。

进程间通信的方式

广播

https://www.jianshu.com/p/0cca211df63c

Messenger

http://blog.csdn.net/lmj623565791/article/details/47017485

此处延伸：简述Binder ， http://blog.csdn.net/luoshengyang/article/details/6618363/

AIDL(Android Interface Definition Language)

为什么要设计AIDL

Android为了实现进程间的通信,尤其是在涉及到多进程并发情况下的进程通信。
Android中每一个进程都对应一个Dalvik VM实例,都有一块自己独立的内存,都在自己的内存上存储数据,执行自己的操作, 各个进程之间就像海上的小岛,在同一个世界，但又有自己的独立的世界。AIDL就相当于两座岛之间的桥梁,通过AIDL制定一些规则, 规定他们能进行哪些交流。
最终达到一个进程访问另一个进程的数据。甚至调用它们一些特定的方法。
如果仅仅是为了跨进程通信我们也还有其他的选择,比如BroadcastReceiver，Message等都可以达到跨进程通信，但是BroadcastRecever占用系统的资源比较多,如果频繁的跨进程通信的话显然就不可取;
Message进行跨进程通信时请求队列时同步进行的,无法并发执行,在有些要求多进程的情况下不实用,这个时候就需要使用AIDL。

它有哪些语法?

语法上基本和Java一致, 只是再一些细微处有些许的差别

文件类型

用AIDL书写的文件后缀是.aidl, 而不是java

数据类型

它只支持一些数据类型,这些数据类型不需要导包

其他知识:

https://www.jianshu.com/p/54bc60246e67

HttpClient与HttpUrlConnection的区别

总结

在Android 2.2版本之前，HttpClient拥有较少的bug，因此使用它是最好的选择。

而在Android 2.3版本及以后，HttpURLConnection则是最佳的选择。它的API简单，体积较小，因而非常适用于Android项目。压缩和缓存机制可以有效地减少网络访问的流量，在提升速度和省电方面也起到了较大的作用。对于新的应用程序应该更加偏向于使用HttpURLConnection，因为在以后的工作当中我们也会将更多的时间放在优化HttpURLConnection上面。

性能优化

内存泄漏

简单的说本该释放的资源和对象没有得到释放，一直被某个或者某些实例所持有却不能被使用导致GC不能够回收。
https://blog.csdn.net/north1989/article/details/51999920

GC(垃圾回收)

在堆中，找到已经无用的对象，并把这些对象占用的空间收回使其可以重新利用.大多数垃圾回收的 算法思路都是一致的：把所有对象组成一个集合，或可以理解为树状结构，从树根开始找，只要可以找到的都是活动对象，如果找不到，这个对象就是凋零的昨日黄 花，应该被回收了。
在sun 的文档说明中，对JVM堆的新域，是采用coping算法，该算法的提出是为了克服句柄的开销和解决堆碎片的垃圾回收。它开始时把堆分成一个对象面和多个 空闲面，程序从对象面为对象分配空间，当对象满了，基于 coping算法的垃圾收集就从根集中扫描活动对象，并将每个活动对象复制到空闲面(使得活动对象所占的内存之间没有空闲洞)，这样空闲面变成了对象面， 原来的对象面变成了空闲面，程序会在新的对象面中分配内存。

布局优化

内存优化

电量优化

Java多线程

Java 多线程

为了解决负载均衡问题,充分利用CPU资源.为了提高CPU的使用率,采用多线程的方式去同时完成几件事情而不互相干扰.为了处理大量的IO操作时或处理的情况需要花费大量的时间等等,比如:读写文件,视频图像的采集,处理,显示,保存等

多线程的好处:

1. 使用线程可以把占据时间长的程序中的任务放到后台去处理
2. 用户界面更加吸引人,这样比如用户点击了一个按钮去触发某件事件的处理,可以弹出一个进度条来显示处理的进度
3. 程序的运行效率可能会提高
4. 在一些等待的任务实现上如用户输入,文件读取和网络收发数据等,线程就比较有用了.

多线程的缺点:

1. 如果有大量的线程,会影响性能,因为操作系统需要在它们之间切换.
2. 更多的线程需要更多的内存空间
3. 线程中止需要考虑对程序运行的影响.
4. 通常块模型数据是在多个线程间共享的,需要防止线程死锁情况的发生

多线程的实现

1. 通过实现 Runnable 接口
2. 通过继承 Thread 类本身

线程的几个主要概念

在多线程编程时，你需要了解以下几个概念：

1. 线程同步
2. 线程间通信
3. 线程死锁
4. 线程控制：挂起、停止和恢复

多线程的使用

1. 有效利用多线程的关键是理解程序是并发执行而不是串行执行的。例如：程序中有两个子系统需要并发执行，这时候就需要利用多线程编程。

2. 通过对多线程的使用，可以编写出非常高效的程序。不过请注意，如果你创建太多的线程，程序执行的效率实际上是降低了，而不是提升了。
3. 请记住，上下文的切换开销也很重要，如果你创建了太多的线程，CPU 花费在上下文的切换的时间将多于执行程序的时间！

如何共享线程之间的数据

https://blog.csdn.net/hejingyuan6/article/details/47053409

https://blog.csdn.net/zhh1072773034/article/details/74240897?locationNum=8&fps=1

Fragment状态保持和恢复

https://segmentfault.com/a/1190000006691830

讲解一下Context

https://blog.csdn.net/lmj623565791/article/details/40481055

JNI

http://www.jianshu.com/p/aba734d5b5cd

此处延伸：项目中使用JNI的地方，如：核心逻辑，密钥，加密逻辑

java虚拟机和Dalvik虚拟机的区别

https://www.jianshu.com/p/923aebd31b65

线程sleep和wait有什么区别

http://blog.csdn.net/liuzhenwen/article/details/4202967

保存Activity状态

onSaveInstanceState()

http://blog.csdn.net/yuzhiboyi/article/details/7677026

WebView与js交互（调用哪些API）

http://blog.csdn.net/cappuccinolau/article/details/8262821/

内存泄露检测，内存性能优化

http://blog.csdn.net/guolin_blog/article/details/42238627

这篇文章有四篇，很详细。

此处延伸：

（1）内存溢出（OOM）和内存泄露（对象无法被回收）的区别。

（2）引起内存泄露的原因

布局优化

http://blog.csdn.net/guolin_blog/article/details/43376527

自定义view和动画

以下两个讲解都讲得很透彻，这部分面试官多数不会问很深，要么就给你一个效果让你讲原理。

（1）http://www.gcssloop.com/customview/CustomViewIndex

（2）http://blog.csdn.net/yanbober/article/details/50577855

设计模式（单例，工厂，观察者。作用，使用场景）

一般说自己会的就ok，不要只记得名字就一轮嘴说出来，不然有你好受。
http://blog.csdn.net/jason0539/article/details/23297037/
此处延伸：Double Check的写法被要求写出来。

String，Stringbuffer，Stringbuilder 区别

开源框架，为什么使用，与别的有什么区别

这个问题基本必问。在自己简历上写什么框架，他就会问什么。
如：Volley，面试官会问我Volley的实现原理，与okhttp和retrofit的区别。
开源框架很多，我就选几个多数公司都会用的出来（框架都是针对业务和性能，所以不一定出名的框架就有人用）
网络请求：Volley，okhttp，retrofit
异步：RxJava，AsyncTask
图片处理：Picasso，Glide
消息传递：EventBus
以上框架请自行查找，太多了就不贴出来了。

Android大厂面试题

爱奇艺

Android消息机制

Android View绘制流程，当一个TextView的实例调用setText()方法后执行了什么

Android dalvik虚拟机和Art虚拟机的优化升级点

Android屏幕渲染机制

热修复的原理，你都了解过哪几种热修复框架

目前较为成熟的热修复框架主要有AndFix、Nuwa以及微信的热更新思想。现在将其主要思想总结如下：

Android 主要的热修复方案原理分析

https://www.jianshu.com/p/d10aa991ca76

OkHttp的原理

Android 线程池的实现原理

Java GC机制

HashMap的实现机制，怎么样HashMap线程安全

可重入锁的实现，公平锁非公平锁都是什么定义？

都用过那些常用的数据结构，说说对树的了解？

Activity启动模式，allowReparent的特点和栈亲和性

WebView优化

有没有Jni使用经验

有使用过RxJava吗？

说说你对设计模式的理解，开发过程中主要用到了哪些设计模式？

快排写一下，动态规划了解吗？

小米

冒泡排序的链表实现

写个快排

写个单例模式

Activity启动模式

异常生命周期

从点击应用图标到进入应用，Android系统都做了哪些工作，期间涉及到的进程切换有哪些？

说说你了解的IPC方法

说说Binder的大体实现

怎么控制另外一个进程的View显示

广播中怎么进行网络请求

说说Android中线程池的实现

HashMap如果Hash冲突了怎么解决？

双线程通过线程同步的方式打印12121212.....

腾讯

HTTPS是怎么实现的？

Android事件传递流程和OnTouchListener的关系

Activity启动模式

Android线程池实现原理

双指缩放拖动大图

客户端网络安全实现

Webview优化

Android应用保活

RemoteViews实现和使用场景

RecyclerView绘制步骤和复用机制

Binder的大体设计模式

Activity遵循什么设计模式

EventBus源码解析，遵循什么设计模式

Handler为什么会发生内存泄漏

Java内存模块分区和GC机制，GC算法有哪些

Finalize机制

强引用 弱引用 软引用 虚引用的区别和使用场景

LeakCanary的使用和实现原理

介绍一下你在开发过程中使用到的设计模式

快排

对服务器众多错误码的处理（错误码有好几万个）

阿里

Android消息机制

Android事件传递流程

Android View绘制流程

Activity启动模式

Android IPC机制

Android线程池设计原理

EventBus源码和设计模式

Android应用保活

Android UI优化

Android启动优化

快排 堆排 单例

进程线程在操作系统中的实现

HTTPS的组成是什么？

ClassLoader的双亲委托

Android中的ClassLoader

有过Jni使用经验吗？

有过跨平台开发经验吗？

今日头条

视频加密

Android Native Crash

视频组成

播放器原理

共同问到的

为什么离职

开发过程中遇到最深刻的BUG是什么

当你的开发任务很紧张，你怎么去做代码优化的？

怎么和你的同事沟通

你对某某某互联网发生事情的看法？（直播答题等等）

其他问题

最新面试题

https://blog.csdn.net/huangqili1314/article/details/72792682

框架MVC、MVP、MVVM

为什么需要架构

通过设计使程序模块化，做到模块内部的高聚合和模块之间的低耦合。这样做的好处是使得程序在开发的过程中，开发人员只需要专注于一点，提高程序开发的效率，并且更容易进行后续的测试以及定位问题。但设计不能违背目的，对于不同量级的工程，具体架构的实现方式必然是不同的，切忌犯为了设计而设计，为了架构而架构的毛病。

MVC

View：对应于布局文件
Model：业务逻辑和实体模型
Controllor：对应于Activity

在Android开发中，Activity并不是一个标准的MVC模式中的Controller，它的首要职责是加载应用的布局和初始化用户界面，并接受并处理来自用户的操作请求，进而作出响应。随着界面及其逻辑的复杂度不断提升，Activity类的职责不断增加，以致变得庞大臃肿。

MVP

View 对应于Activity，负责View的绘制以及与用户交互
Model 依然是业务逻辑和实体模型
Presenter 负责完成View于Model间的交互

在MVP里，Presenter完全把Model和View进行了分离，主要的程序逻辑在Presenter里实现。而且，Presenter与具体的View是没有直接关联的，而是通过定义好的接口进行交互，从而使得在变更View时候可以保持Presenter的不变。 View只应该有简单的Set/Get的方法，用户输入和设置界面显示的内容，除此就不应该有更多的内容，绝不容许直接访问Model，这就是与MVC很大的不同之处。

MVP的优缺点

优点

降低耦合度，实现了Model和View真正的完全分离。
模块职责划分明显，层次清晰。
Presenter可以复用，一个Presenter可以用于多个View，而不需要更改Presenter的逻辑（当然是在View的改动不影响业务逻辑的前提下）。
如果我们把逻辑放在Presenter中，那么我们就可以脱离用户接口来测试这些逻辑（单元测试）。

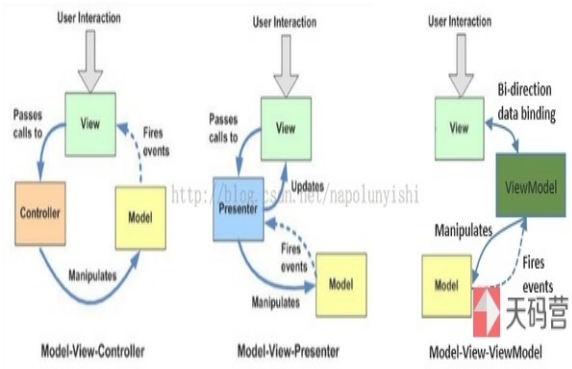
缺点

额外的代码复杂度及学习成本。
如果Presenter过多地与特定的视图的联系过于紧密，一旦视图需要变更，那么Presenter也需要变更了。

MVVM

MVVM可以算是MVP的升级版，
其中的VM是ViewModel的缩写，ViewModel可以理解成是View的数据模型和Presenter的合体，
ViewModel和View之间的交互通过Data Binding完成，

而Data Binding可以实现双向的交互，这就使得视图和控制层之间的耦合程度进一步降低，关注点分离更为彻底，同时减轻了Activity的压力。



MVC->MVP->MVVM演进过程

MVC -> MVP -> MVVM 这几个软件设计模式是一步步演化发展的，MVVM 是从 MVP 的进一步发展与规范，MVP 隔离了MVC中的 M 与 V 的直接联系后，靠 Presenter 来中转，所以使用 MVP 时 P 是直接调用 View 的接口来实现对视图的操作的，这个 View 接口的东西一般来说是 showData、showLoading 等等。M 与 V已经隔离了，方便测试了，但代码还不够优雅简洁，所以 MVVM 就弥补了这些缺陷。在 MVVM 中就出现的 Data Binding 这个概念，意思就是 View 接口的 showData 这些实现方法可以不写了，通过 Binding 来实现。

同

三者的差异在于如何粘合View和Model，实现用户的交互操作以及变更通知

如果把这三者放在一起比较，先说一下三者的共同点，也就是Model和View：

Model：

数据对象，同时，提供本应用外部对应用程序数据的操作的接口，也可能在数据变化时发出变更通知。
Model不依赖于View的实现，只要外部程序调用Model的接口就能够实现对数据的增删改查。

View：

UI层，提供对最终用户的交互操作功能，包括UI展现代码及一些相关的界面逻辑代码。

异

Controller

Controller接收View的操作事件，根据事件不同，或者调用Model的接口进行数据操作，或者进行View的跳转，从而也意味着一个Controller可以对应多个View。Controller对View的实现不太关心，只会被动地接收，Model的数据变更不通过Controller直接通知View，通常View采用观察者模式监听Model的变化。

Presenter

Presenter与Controller一样，接收View的命令，对Model进行操作；与Controller不同的是Presenter会反作用于View，Model的变更通知首先被Presenter获得，然后Presenter再去更新View。一个Presenter只对应于一个View。根据Presenter和View对逻辑代码分担的程度不同，这种模式又有两种情况：Passive View和Supervisor Controller。

ViewModel

注意这里的“Model”指的是View的Model，跟MVVM中的一个Model不是一回事。所谓View的Model就是包含View的一些数据属性和操作的这么一个东东，这种模式的关键技术就是数据绑定（data binding），View的变化会直接影响ViewModel，ViewModel的变化或者内容也会直接体现在View上。这种模式实际上是框架替应用开发者做了一些工作，开发者只需要较少的代码就能实现比较复杂的交互。

sleep和wait有什么区别

第一种解释：

功能差不多,都用来进行线程控制,他们最大本质的区别是:sleep()不释放同步锁,wait()释放同步缩.

第二种解释：

sleep是Thread类的静态方法。sleep的作用是让线程休眠制定的时间，在时间到达时恢复，也就是说sleep将在接到时间到达事件事恢复线程执行

React Native原理

React Native

原理

周期

React Native面试题

React Native相对于原生的ios和Android有哪些优势？

1.性能媲美原生APP
2.使用JavaScript编码，只要学习这一种语言
3.绝大部分代码安卓和IOS都能共用
4.组件式开发，代码重用性很高
5.跟编写网页一般，修改代码后即可自动刷新，不需要慢慢编译，节省很多编译等待时间
6.支持APP热更新，更新无需重新安装APP

React Native组件的生命周期

getDefaultProps、getInitialState、componentWillMount、componentDidMount、componentWillReceiveProps、shouldComponentUpdate、componentWillUpdate、componentDidUpdate、componentWillUnmount

React Native的优点和缺点在哪里？

需要转换成原生，占用的内存比较高

父传子，子传父数据传递方式？

props state refs 方面回答
`父组件与子组件之间的数据传递的实现方式大致可以分为2种情况：`

1. 子组件用自己的flux环传递数据，最后调用父组件的onChange事件将数据传给父组件。
2. 子组件调用父组件的onChange事件，在父组件中的onChange事件中调用flux环传递数据到付组件的View层。

如何实现底部TabBar的高度不一样呢？（类似新浪微博底部加号）

`flex布局绝对定位问题`

你的项目有没有使用redux或者是mobx来管理数据呢？

请您简单介绍一下redux？dva? mobx？

简答：redux ==> action/reducer/store
mobx ==>数据双向绑定

当你调用setState的时候，发生了什么事？

数据结构

数据结构

数据结构是计算机存储、组织数据的方式。数据结构是指相互之间存在一种或多种特定关系的数据元素的集合。通常情况下，精心选择的数据结构可以带来更高的运行或者存储效率。数据结构往往同高效的检索算法和索引技术有关。

Android开发

基础知识

Java基础

GC是什么？为什么要有GC？

GC是垃圾收集的意思，内存处理是编程人员容易出现问题的地方，忘记或者错误的内存回收会导致程序或系统的不稳定甚至崩溃，Java提供的GC功能可以自动监测对象是否超过作用域从而达到自动回收内存的目的，Java语言没有提供释放已分配内存的显示操作方法。Java程序员不用担心内存管理，因为垃圾收集器会自动进行管理。要请求垃圾收集，可以调用下面的方法之一：System.gc() 或 Runtime.getRuntime().gc() ，但JVM可以屏蔽掉显示的垃圾回收调用。

垃圾回收可以有效的防止内存泄露，有效的使用可以使用的内存。垃圾回收器通常是作为一个单独的低优先级的线程运行，不可预知的情况下对内存堆中已经死亡的或者长时间没有使用的对象进行清除和回收，程序员不能实时的调用垃圾回收器对某个对象或所有对象进行垃圾回收。在Java诞生初期，垃圾回收是Java最大的亮点之一，因为服务器端的编程需要有效的防止内存泄露问题，然而时过境迁，如今Java的垃圾回收机制已经成为被诟病的东西。移动智能终端用户通常觉得iOS的系统比Android系统有更好的用户体验，其中一个深层次的原因就在于Android系统中垃圾回收的不可预知性。

面向对象的特征有哪些方面?

1)抽象：抽象是将一类对象的共同特征总结出来构造类的过程，包括数据抽象和行为抽象两方面。抽象只关注对象有哪些属性和行为，并不关注这些行为的细节是什么。

2)继承：继承是从已有类得到继承信息创建新类的过程。提供继承信息的类被称为父类（超类、基类）；得到继承信息的类被称为子类（派生类）。继承让变化中的软件系统有了一定的延续性，同时继承也是封装程序中可变因素的重要手段（如果不能理解请阅读阎宏博士的《Java与模式》或《设计模式精解》中关于桥梁模式的部分）。

3)封装：通常认为封装是把数据和操作数据的方法绑定起来，对数据的访问只能通过已定义的接口。面向对象的本质就是将现实世界描绘成一系列完全自治、封闭的对象。我们在类中编写的方法就是对实现细节的一种封装；我们编写一个类就是对数据和数据操作的封装。可以说，封装就是隐藏一切可隐藏的东西，只向外界提供最简单的编程接口（可以想想普通洗衣机和全自动洗衣机的差别，明显全自动洗衣机封装更好因此操作起来更简单；我们现在使用的智能手机也是封装得足够好的，因为几个按键就搞定了所有的事情）。

4)多态性：多态性是指允许不同子类型的对象对同一消息作出不同的响应。简单的说就是用同样的对象引用调用同样的方法但是做了不同的事情。多态性分为编译时的多态性和运行时的多态性。如果将对象的方法视为对象向外界提供的服务，那么运行时的多态性可以解释为：当A系统访问B系统提供的服务时，B系统有多种提供服务的方式，但一切对A系统来说都是透明的（就像电动剃须刀是A系统，它的供电系统是B系统，B系统可以使用电池供电或者用交流电，甚至还有可能是太阳能，A系统只会通过B类对象调用供电的方法，但并不知道供电系统的底层实现是什么，究竟通过何种方式获得了动力）。方法重载

（overload）实现的是编译时的多态性（也称为前绑定），而方法重写（override）实现的是运行时的多态性（也称为后绑定）。运行时的多态是面向对象最精髓的东西，要实现多态需要做两件事：1. 方法重写（子类继承父类并重写父类中已有的或抽象的方法）；2. 对象造型（用父类型引用引用子类型对象，这样同样的引用调用同样的方法就会根据子类对象的不同而表现出不同的行为）。

String 是最基本的数据类型吗?

答：不是。Java中的基本数据类型只有8个：byte、short、int、long、float、double、char、boolean；除了基本类型（primitive type）和枚举类型（enumeration type），剩下的都是引用类型（reference type）。

&和&&的区别？

&运算符有两种用法：(1)按位与；(2)逻辑与。&&运算符是短路与运算。逻辑与跟短路与的差别是非常巨大的，虽然二者都要求运算符左右两端的布尔值都是true整个表达式的值才是true。&&之所以称为短路运算是因为，如果&&左边的表达式的值是false，右边的表达式会被直接短路掉，不会进行运算。很多时候我们可能都需要用&&而不是&，例如在验证用户登录时判定用户名不是null而且不是空字符串，应当写为：username != null &&!username.equals(“”)，二者的顺序不能交换，更不能用&运算符，因为第一个条件如果不成立，根本不能进行字符串的equals比较，否则会产生NullPointerException异常。注意：逻辑或运算符（|）和短路或运算符（||）的差别也是如此。

补充：如果你熟悉JavaScript，那你可能更能感受到短路运算的强大，想成为JavaScript的高手就先从玩转短路运算开始吧。

数据结构

面向对象思想

设计模式

开发环境

Eclipse

Android Studio

Android SDK

Activity

什么是Activity?

Activity是android的四大组件中最常见的部分，是Android用于交互的部分。Activity有四种状态：Active/Runing、Paused、Stoped、Killed。主要有七个生命周期方法。通过Intent与其他Activity或者组件通信。

请描述一下Activity生命周期。

Activitiy的生命周期方法主要有七个：
onCreate()、onRestart()、onStart()、onResume()、onPause()、onStop()、onDestory()。

两个Activity之间跳转时必然会执行的是哪几个方法。

这个问题与 从A跳转到B需要执行那些生命周期方法 的意思是相同的。分为两种情况：

1. B不透明时，A执行除onDestory之外的全部方法，B执行到onResume方法；
2. B透明时，A执行到onPause方法，B执行到onResume方法。

横竖屏切换时候Activity的生命周期。

如何将一个Activity设置成窗口的样式。

在AndroidManifest.xml中在你需要显示为窗口的activity中添加如果属性：
android:theme="@style/Theme.FloatActivity" 即可

你后台的Activity被系统回收怎么办？如果后台的Activity由于某原因被系统回收可了，如何在被系统回收之前保存当前状态？

onSaveInstanceState在这个方法中保存信息。在onCreate中判断savedInstanceState是否为空，不为空就去出来。

如何退出Activity？如何安全退出已调用多个Activity的Application？

退出单个Activity调用finish()方法。

两个Activity之间怎么传递数据？

可以通过Intent，Bundle或者SharedPreferences都可以传递数据。

怎么在启动一个Activity时就启动一个service？

在activity的onCreate里写
startService(xxx);
然后
this.finish();结束自己..
这是最简单的方法 可能会有屏幕一闪的现象，如果UI要求严格的话用AIDL把
根据service与activity的生命周期，选择在onCreate或onResume中startService;当然要记得stopService.

同一个程序，但不同的Activity是否可以放在不同的Task任务栈中？

可以放在不同的Task中。需要为不同的activity设置不同的taskaffinity属性，启动activity的Intent需要包含FLAG_ACTIVITY_NEW_TASK标记

Activity怎么和service绑定，怎么在activity中启动自己对应的service？

startService()一旦被创建 调用着无关 没法使用service里面的方法
bindService () 把service 与调用者绑定,如果调用者被销毁, service会销毁
bindService() 我们可以使用service 里面的方法
bindService(). 让activity能够访问到service里面的方法

Service

Broadcastreceiver

Contentprovider

ActionBar

Fragment

UI

通信

数据持久化

性能

调试

适配

测试

安全

NDK

手机功能

第三方扩展

其他

2018 Java面试题

https://www.jianshu.com/p/7de76a9646fc

Android(2017-2018)BAT面试题整理

https://www.jianshu.com/p/4115bcf9f92e

2017下半年，一二线互联网公司Android面试题汇总

https://zhuanlan.zhihu.com/p/30016683

2018阿里Android面试题

一面

android中的dp、px、dip相关概念

handler机制，四个组成部分及源码解析

布局相关的`<merge>`、`<viewstub>`控件作用及实现原理

android中的布局优化

relativelayout和LinearLayout在实现效果同等情况下选择使用哪个？为什么？

view的工作原理及measure、layout、draw流程，要求了解源码

怎样自定义一个弹幕控件？

如果控件内部卡顿你如何去解决并优化？

listview的缓存机制

Invalidate、postInvalidate、requestLayout应用场景

多线程，5个线程内部打印hello和word，hello在前，要求提供一种方法使得5个线程先全部打印出hello后再打印5个word。

实现一个自定义view，其中含有若干textview，textview文字可换行且自定义- - view的高度可自适应拓展

编程题：将元素均为0、1、2的数组排序。在手打了一种直接遍历三种数目并打印的方法后让手写实现，手写实现后让再说一种稳定的方法，说了一种通过三个下标遍历一遍实现的方法，读者可自行百度，在此不赘述。

二面

JVM方面

java内存模型，五个部分，程序计数器、栈、本地栈、堆、方法区。

每个部分的概念、特点、作用。

类加载的过程，加载、验证、准备、解析、初始化。每个部分详细描述。

加载阶段读入.class文件，class文件时二进制吗，为什么需要使用二进制的方式？

验证过程是防止什么问题？验证过程是怎样的？加载和验证的执行顺序？符号引用的含义？

准备过程的静态成员变量分配空间和设置初始值问题。

解析过程符号引用替代为直接引用细节相关。

初始化过程jvm的显式初始化相关。

类卸载的过程及触发条件。

三种类加载器，如何自定义一个类加载器？

双亲委派机制。

JVM内存分配策略，优先放于eden区、动态对象年龄判断、分配担保策略等。

JVM垃圾回收策略，怎样判对象、类需要被回收？

四种垃圾回收算法标记-清除、复制、标记-整理、分代收集。

JVM中的垃圾回收器，新生代回收器、老年代回收器、stop-the-world概念及解决方法。

四类引用及使用场景？

集合类

hashmap实现的数据结构，数组、桶等。

hashmap的哈希冲突解决方法：拉链法等。拉链法的优缺点。

hashmap的参数及影响性能的关键参数：加载因子和初始容量。

Resize操作的过程。

hashmap容量为2次幂的原因。

hashtable线程安全、synchronized加锁。

hashtable和hashmap异同。

为什么hashtable被弃用？

果断将话题扯到concurrenthashmap，讲了concurrenthashmap相比于hashtable做的优化、segment的概念、concurrenthashmap高效的原因。中间面试官问的问题：

容器类中fastfail的概念。

concurrenthashmap的插入操作是直接操作数组中的链表吗？

集合类相关over，由于都是自己主动在说，把握了主动权，相谈甚欢。

三面

为什么要使用多线程？多线程需要注意的问题。上下文开销、死锁等。

java内存模型、导致线程不安全的原因。

volatile关键字，缓存一致性、指令重排序概念。

synchronize关键字，java对象头、Markword概念、synchronize底层monitorenter和moniterexit指令。

lock语句和synchronize对比。

原子操作，CAS概念、相关参数。

乐观锁、悲观锁概念及使用场景。

线程池概念、实现原理等。

JVM锁的优化，偏向锁、轻量级锁概念及原理。

通信协议

TCP三次握手、四次挥手。

TCP保证可靠传输的实现：停止等待协议、滑动窗口协议、流量控制、拥塞控制等。

http请求报文结构、响应报文，状态码。

http2.0相比于http1.0的新特性，推送、多路复用、消息头压缩等。

Android

handler机制组成，handler机制每一部分的源码包括looper中的loop方法、threadlocal概念、dispatchmessage方法源码，runnable封装message等。

listview缓存机制、recycleview缓存机制。

bitmap高效加载，三级缓存等。

binder机制原理。

view的工作原理及measure、layout、draw流程。哪一个流程可以放在子线程中去执行？

draw方法中需要注意的问题？

view的事件分发机制。

android性能优化：布局优化、绘制优化、内存泄露优化、bitmap、内存泄露等。

内存泄露的概念？android中发生的场景？怎么解决？讲了handler、动画等