# 第一部分 Java

## Java基础

1. equals方法的作用是什么，它和==有什么区别。
2. hashCode方法的作用是什么，它和equals方法有什么联系。
3. 可以直接根据hashcode判断两个对象相等吗。
4. 抽象类和接口的区别。
5. 什么是面向接口编程，它有什么好处。
6. String、StringBuffer和StringBuilder的区别。
7. 什么是序列化，序列化前后对象有何区别。
8. try catch finally，try里有return，finally还执行么。
9. Exception与Error类结构，Exception与Error的区别。
10. Java面向对象的三个特征和含义。
11. 使用for循环、foreach循环和迭代器遍历集合时能否进行删除操作，为什么？
12. 直接对字符串赋值和使用new String()的区别。
13. 泛型的优缺点。
14. 解析XML的几种方式的原理与特点：DOM、SAX、PULL。
15. Java1.7、Java1.8和Java1.9的新特性。
16. 序列化前后对象有何区别。

## Java集合框架

1. Java集合框架的结构。
2. ArrayList实现原理。
3. LinkedList实现原理。
4. HashMap实现原理。
5. LinkedHashMap实现原理。
6. ConcurrentHashMap的原理。
7. Vector实现原理。
8. HashMap实现原理。
9. HashTable实现原理。
10. TreeMap实现原理。
11. TreeSet实现原理。
12. HashMap的原理，ConcurrentHashMap的原理。
13. SparseArray、ArrayMap和HashMap区别。

## IO/NIO

## Java并发（Concurrent）

1. 什么是线程，什么是进程，能不能只用进程。
2. 什么是线程安全，如何保证线程安全，Vector是线程安全的吗。
3. wait()和sleep()的区别。
4. 如何停止一个线程。
5. 什么是守护线程。
6. 一个线程包含了哪些状态。
7. 线程同步的方法。
8. Java中volatile变量是什么。
9. ThreadLocal是什么。
10. Java锁的分类。
11. Java中的线程池有几种。
12. 如果提交任务时，线程池队列已满，会发生什么。
13. 有三个线程T1、T2、T3，怎么保证它们顺序执行。
14. 一个线程发生异常会怎么样。
15. 锁的等级：方法锁、对象锁、类锁。
16. 生产者消费者模式的集中实现，阻塞队列，sync关键字，lock实现，ReentrantLock实现，写出生产者消费者模式简单代码实现。
17. ThreadLocal的设计理念与作用，ThreadPool用法与优势（AsyncTask底层有使用）。
18. 线程池的底层实现与工作原理。
19. 阻塞队列，ArrayBlockingQueue、CountDownLatch等。
20. 多线程场景遇见过没有。
21. synchronize原理。
22. lock原理。
23. Integer类是不是线程安全的，为什么。

## JVM

1. Java内存模型、堆区和栈区的区别，OOM、SOF。
2. Java虚拟机怎么判断那些对象是需要进行回收的。
3. Java的四种引用，强弱软虚，及其适用的场景。
4. 强引用置为null，会不会被回收
5. Java类加载机制，类加载分为那几个阶段。
6. 垃圾回收机制和算法。
7. 动态代理的原理，代理对象在那个阶段生成的。
8. Java值传递。
9. 垃圾回收原理以及回收策略。
10. 类的加载机制，为什么用双亲委派模型，Java是否支持多态。

## 设计模式

1. 手动画出设计模式的UML类图。
2. 工具类使用单例和静态内部类有什么区别。
3. 什么是IOC，他和DI有什么区别，为什么使用IOC而不使用工厂模式。
4. 动态代理模式的原理。
5. 适配器模式，装饰者模式，外观模式的异同。

## 七、其他

# 第二部分 Android

## Android基础

1. Serializable和Parcalable的区别。
2. dpi、ppi、px、pt、dp、sp的区别。
3. Android 5.0、6.0、7.0、8.0的新特性。
4. Android持久化方案。
5. 如何判断应用被强杀，应该如何解决。
6. 怎么退出终止App。
7. apk的目录结构。
8. AysncTask各个函数分别在什么线程回调的，以及底部实现原理。
9. 了解sp的apply和commit的区别吗，sp具体是怎么实现的，sp什么时候会将内存中的数据写入到文件中。

## 四大组件相关

1. 手动画出Activity、Fragment的生命周期，他们是怎么关联的。
2. Activity常见情形下的生命周期，如按下home键、锁屏。
3. 异常情况下Activity的生命周期，如横竖屏切换、系统资源不足。
4. 什么时Activity任务栈。
5. Activity启动模式以及使用的场景，对应的Intent Flag。
6. IntentFilter匹配规则。
7. Fragment commit和commitAllowingStateLossing的区别。
8. Service生命周期，与IntentService的区别。
9. startService和bindService混合使用的场景，这时候Service的生命周期是怎样的。
10. 广播的两种注册方式有什么区别。
11. 怎么增加广播的安全性。
12. fragment实现懒加载，与ViewPager搭配时。
13. AppCompatActivity、FragmentActivity和Activity区别。

## View剖析

1. Touch事件的分发流程。
2. 点击事件被拦截，但是想传到下面的view，如何操作。
3. View的位置参数有哪些，left、x、translationX的含义以及三者的关系。
4. 什么是MeasureSpec。
5. View绘制过程。
6. 怎么获取View的宽高。
7. 自定义View的流程，自定义View需要注意的问题，例如自定义View是否需要重写onLayout，onMeasure。
8. Invalidate、postInvalidate、requestLayout的区别。
9. 动画的原理，底层如何给上层信号。
10. 插值器和估值器。
11. Android多点触控。
12. FragmentPager、FragmentStatePagerAdapter区别。
13. ViewPager、RecyclerView和SrcollView嵌套滑动冲突，显示不完整，滑动卡顿问题。
14. RecyclerView和ListView复用机制。
15. 处理滑动的几种方式，Scroller滑动的原理。
16. 如何优化WebView，WebView的泄露。
17. Native和js交互流程。
18. 简述工作线程更新UI的方法。
19. 滑动卡顿如何解决（不同原因及对应处理方式）。

## 三、文件系统

1. Android有多个资源文件夹，应用在不同分辨率下是如何查找对应文件夹下的资源的，描述整个过程。
2. 应用最多占多少内存。

## 四、源码解析

1. 怎么进行IPC，详细解释Binder机制。
2. Android系统的启动过程。
3. Launcher的启动过程。
4. Apk的安装过程。
5. Activity的启动过程。
6. Service的启动过程。
7. Apk编译打包过程。
8. Android PendingIntent内部机制。
9. 一个apk从开始安装启动，系统都做了哪些事情？请从AMS，WMS，PMS的角度考虑下，一级进程是如何启动的？
10. AMS和WMS相关的数据结构和沟通的桥梁是什么？AMS的堆栈是如何管理的？WMS的堆栈是如何管理的？
11. Android内存管理机制以及LMK相关的机制，以及AMS中进程管理机制，请分别介绍下。
12. PMS相关的开机流程，apk安装的流程，adb install和pms scan的却别有哪些？
13. Broadcast的机制，分发的流程是什么？如何传递到每个app进程的？动态广播和静态广播的处理流程在哪里不一样？
14. 多用户最主要的机制以及创建一个新的用户系统需要做哪些事情。
15. Runtime permission，如何把一个预置的app默认给它权限，不要授权。
16. 如何实现预装的apk在卸载后，通过恢复出厂设置恢复过来，请介绍下方案。
17. Android资源加载和打包的机制介绍，一个图片在app中调用R.id.调用后是如何找到的？
18. Android Overlay的机制是什么？
19. Android权限管理的机制是什么。
20. 为何android.uid.system相关的进程不能访问sdcard。
21. 开机流程和关机流程请描述下。
22. Bootanimation是如何启动和退出的。
23. Binder相关的机制以及在Android平台的使用，Android还有什么IPC通信方式，各有什么优缺点。
24. 死机，重启等stability问题分析流程？watchdog reset如何分析？
25. Native Crash问题如何分析，以及crash在art相关的oat，odex文件如何分析。
26. Android ART机制，与dalvik的区别，JIT与AOT的区别是什么？ART GC有什么改善，还有什么缺点。
27. ANR，OOM等问题的分析流程介绍。
28. Android++智能指针相关的使用介绍。
29. Android编译，优化，ART相关编译优化。
30. Input相关事件的分发机制，tp相关问题解决。
31. 按键事件和tp事件的处理有什么不同点和相同点。
32. 功耗相关问题的分析。
33. 性能相关问题的分析。
34. Android N与M的一些典型的改变有哪些？Multi-window机制介绍。
35. PowerManagerService主要做了哪些相关的操作？系统亮灭屏都有哪些流程。
36. Wakelock机制，Android如何和linux管理这些wakelock。
37. Alarm相关机制，doze相关的机制以及运行方式。
38. 性能优化如何分析systrace。
39. ActivityThread相关。

## 性能优化

1. 如何进行屏幕适配。
2. ANR原理（回答主线程阻塞5秒不算，要分析源码）。
3. Java的四种引用，强弱软虚，及其适用的场景。
4. 内存泄漏如何产生，常见的Android内存泄漏的场景，怎么解决。
5. 用IDE如何分析内存泄漏。
6. 怎么减小APK大小。
7. 如何解决卡顿问题。
8. Android UI卡顿怎么进行监测。
9. 窗口泄露原因，及解决办法。
10. 怎么实现页面渲染完成之后才加载数据，页面加载卡顿问题优化。
11. Android启动冷白屏优化。
12. ListView的卡顿优化、资讯流懒加载如何实现。
13. ListView如何优化，ListView和RecyclerView的区别，二者的缓存逻辑。
14. 提高app安全性的方法。
15. Bitmap使用的时候注意什么。
16. bitmap recycler相关
17. 如何处理App启动流程优化。
18. OOM能不能用try catch捕获。
19. OOM遇到过哪些情况，如何解决的。
20. 如何对Android应用进行性能分析。

## 热门技术

1. 插件化怎么加载资源和绑定Activity生命周期。
2. 插件化技术原理及常用的插件化框架。
3. 热修复技术原理及常用的热修复框架。
4. 在线更新。
5. 增量更新。
6. 多渠道打包。
7. 插件化开发流程，插件化优势，插件化开发中遇到的问题以及如何解决的。
8. 插件化Activity生命周期的管理，资源的访问原理。
9. 组件化开发流程，ARouter路由协议的原理分析。
10. 插件化首次启动耗时长，如何优化。
11. 什么是gradle task。

## 常用框架

1. Glide传入不同Context的区别，如何绑定生命周期的。
2. Glide使用什么缓存，如何设置缓存大小。
3. RxJava切换线程的原理。
4. OkHttp采用的是那种线程池，这样设计的好处是什么。

## 架构设计

1. MVP模式的优缺点，怎么改进。
2. 怎么实现MVP Presenter的复用，怎么实现在不固定位置进行网络请求。
3. MVC到MVP的架构重构流程。

## 经典案例实现

1. 加载1000张图片，怎么进行优化。
2. 怎么管理线程任务，暂停和提前中断任务。
3. 怎么加载大图，怎么进行优化。
4. 实现一个库，完成日志的实时上报和延时上报两种功能，该从哪些方面考虑。
5. 如果想要频繁的读写一个大文件，提供实现方案。
6. 给你一个list数量长达1000，如何实现高效的插入数据库，使用事务，事务的效率为什么高，多线程并发插入。
7. 如何实现一个监控ANR的库，有哪几种方案。

## 其他

1. 进程保活的实现方案和优缺点。
2. 检测前后台切换。
3. Android长连接心跳机制。
4. 自定义gradle插件，gradle加载的几个步骤，gradle语法。
5. json、xml数据格式有什么优势和劣势。
6. JNI用过么。
7. splite升级，增加字段的语句。
8. Sqlite 怎么增加一个字段。
9. Sqlite 怎么删除一个字段。

# 第三部分 计算机网络

1. TCP三次握手过程，为什么需要三次握手。
2. TCP四次挥手过程，为什么需要四次挥手。
3. TCP四次挥手为什么需要等待2MSL。
4. 什么是DNS。
5. Http 的session和cookie的区别。
6. Http 和 Https的区别。
7. Https工作原理。
8. HTTP协议请求响应过程和HTTPS工作原理。
9. TCP、UDP区别，拥塞控制原理，视频传输为什么用UDP，UDP丢包会产生什么问题，如何处理数据包的发送速度和带宽的关系。

# 第四部分 操作系统

1. 对象和类之间的相互关系。
2. 单例模式。

# 第五部分 操作系统