目录

[一、 面试题\_Android 1](#_Toc374969441)

[1. Overload和Override的区别 1](#_Toc374969442)

[2. String 和StringBuffer的区别 1](#_Toc374969443)

[3. 实现一个字符串倒序 1](#_Toc374969444)

[4. 抽象类与接口的区别（abstract与interface的区别） 1](#_Toc374969445)

[5. 集合的实现类与区别 1](#_Toc374969446)

[6. 线程有几种状态,分别是哪些（调用run()和调用start()的区别） 2](#_Toc374969447)

[7. Final、finally、finanlize()的区别 3](#_Toc374969448)

[8. J2EE是什么？请对以下在J2EE中常用的名词进行解释(或简单描述) 3](#_Toc374969449)

[9. j2ee常用的设计模式？说明工厂模式 3](#_Toc374969450)

[10. 开发中都用到了那些设计模式?用在什么场合 3](#_Toc374969451)

[11. 常用设计模式及应用场景 3](#_Toc374969452)

[12. 广播如何调用，有什么方式，各自的区别，实现广播的意图是什么，哪里用到 4](#_Toc374969453)

[13. android系统架构 4](#_Toc374969454)

[14. Android的四大组件是什么？它们的作用是什么 5](#_Toc374969455)

[15. Activity生命周期？保存activity的一些信息在哪个生命周期方法中 5](#_Toc374969456)

[16. Activity的 onSaveInstanceState() 和 onRestoreInstanceState() 5](#_Toc374969457)

[17. activity在屏幕旋转时的生命周期 5](#_Toc374969458)

[18. Android中asset文件夹和raw文件夹区别 5](#_Toc374969459)

[19. Android中的五种存储方式及其应用场景 6](#_Toc374969460)

[20. 什么是ANR 如何避免它？（Android线程间的通信方式 ） 6](#_Toc374969461)

[21. listview优化策略 6](#_Toc374969462)

[22. ListView分页加载实现思路 6](#_Toc374969463)

[23. ListView图片异步加载实现思路 7](#_Toc374969464)

[24. Intent的原理、作用、可以传递哪些类型的参数 7](#_Toc374969465)

[25. 如何实现屏幕分辨率的自适应（布局、九线图、目录） 7](#_Toc374969466)

[26. 简述Android中的IPC机制 7](#_Toc374969467)

[27. Android程序入口如何判断 7](#_Toc374969468)

[28. android哪几种方式访问网络 7](#_Toc374969469)

[29. 说说HttpClient的通信过程 8](#_Toc374969470)

[30. 移动互联数据交互格式有哪些及其区别 8](#_Toc374969471)

[31. XML解析有哪几种？各自优缺点，官方推荐使用哪种 8](#_Toc374969472)

[32. sax解析代码 8](#_Toc374969473)

[33. 百度地图核心类，及实现的功能 8](#_Toc374969474)

[34. GC内存泄露在什么情况下会出现？怎么解决 8](#_Toc374969475)

[35. android内存的优化 9](#_Toc374969476)

[36. 加载大图片的时候如何防止内存溢出 9](#_Toc374969477)

[37. Android缓存机制 9](#_Toc374969478)

[38. 如何实现消息推送 10](#_Toc374969479)

[39. MVC在Android中的应用 10](#_Toc374969480)

[40. Android自定义组件实现思路 10](#_Toc374969481)

[41. 版本更新的实现思路 10](#_Toc374969482)

[42. 播放视频有哪些实现方式 10](#_Toc374969483)

[43. NDK开发流程？（JNI运行原理） 11](#_Toc374969484)

[44. 如何实现一键退出 11](#_Toc374969485)

[45. AndroidManifest.xml清单文件<Activity>标签中属性的含义 11](#_Toc374969486)

[46. 如何将一个Activity设置成窗口的样式 11](#_Toc374969487)

[47. 谈谈UI中， Padding和Margin有什么区别,gravity与layout\_gravity的区别 11](#_Toc374969488)

[48. 实现手风琴效果 11](#_Toc374969489)

[49. Android SDK 3.0(HoneyComb)及4.0(Ice Cream)新特性 11](#_Toc374969490)

[50. android客户端如何实现自动登录 12](#_Toc374969491)

[51. 请介绍下Android中常用的五种布局。 12](#_Toc374969492)

[52. android中的动画有哪几类，它们的特点和区别是什么 12](#_Toc374969493)

[53. activity的启动模式有哪些？是什么含义 12](#_Toc374969494)

[54. 跟activity和Task 有关的 Intent启动方式有哪些？其含义是 13](#_Toc374969495)

[55. 如何启用Service，如何停用Service。 15](#_Toc374969496)

[56. Handler 16](#_Toc374969497)

[57. 说明handler机制的原理 17](#_Toc374969498)

[58. 什么情况会导致Force Close ？如何避免？能否捕获导致其的异常 17](#_Toc374969499)

[59. 请介绍下ContentProvider是如何实现数据共享的。 18](#_Toc374969500)

[60. Service和Thread的区别 18](#_Toc374969501)

[61. Android异常 18](#_Toc374969502)

[62. IntentService有何优点 18](#_Toc374969503)

[63. Activity 19](#_Toc374969504)

[64. 如何将一个Activity设置成窗口的样式 19](#_Toc374969505)

[65. 如何退出Activity？如何安全退出已调用多个Activity的Application 19](#_Toc374969506)

[66. AIDL的全称是什么？如何工作？能处理哪些类型的数据 20](#_Toc374969507)

[67. 请解释下Android程序运行时权限与文件系统权限的区别 20](#_Toc374969508)

[68. 系统上安装了多种浏览器，能否指定某浏览器访问指定页面？请说明原由 20](#_Toc374969509)

[69. android系统的优势和不足 20](#_Toc374969510)

[70. Android dvm的进程和Linux的进程, 应用程序的进程是否为同一个概念 21](#_Toc374969511)

[72. 什么是嵌入式实时操作系统, Android 操作系统属于实时操作系统吗 21](#_Toc374969512)

[73. 一条最长的短信息约占多少byte 22](#_Toc374969513)

[74. 用算法调整图片的透明度 22](#_Toc374969514)

[75. 如何将SQLite数据库(dictionary.db文件)与apk文件一起发布 22](#_Toc374969515)

[76. sqlite的用处及用法, 哪里用过 22](#_Toc374969516)

[77. sqlite，修改，添加字段，版本更新，排序，默认排序 22](#_Toc374969517)

[78. 如何建一个“aaa.db”数据库，如何建一个user表；写出增删改查语句 22](#_Toc374969518)

[79. SQLite线程安全吗？sqlite3\_threadsafe()来确实是否线程安全版本 22](#_Toc374969519)

[80. 怎么提高sqlite的效率 22](#_Toc374969520)

[81. 如何将打开res aw目录中的数据库文件 22](#_Toc374969521)

[82. DDMS和TraceView的区别 22](#_Toc374969522)

[83. java中如何引用本地语言 22](#_Toc374969523)

[84. ListView切换奇偶行背景色 22](#_Toc374969524)

[85. getView()方法中判断position的奇偶性, 然后给ConvertView设置背景色 23](#_Toc374969525)

[86. EditText单行显示 23](#_Toc374969526)

[87. oom避免以及产生方式，遇到过几种OOM 23](#_Toc374969527)

[88. throw与throws区别 23](#_Toc374969528)

[89. LOST.DIR 23](#_Toc374969529)

[90. 设置界面怎么做的 24](#_Toc374969530)

[91. ui效果图介绍布局 24](#_Toc374969531)

[92. login模块怎么做的 24](#_Toc374969532)

[93. 点击登录按钮的时候，会从server端获得什么？传送什么 24](#_Toc374969533)

[94. android客户端如何实现自动登录 24](#_Toc374969534)

[95. 版本更新，从server端获得什么，通过什么进行更新 24](#_Toc374969535)

[96. Fragment和Activity之间的联系 24](#_Toc374969536)

[97. Fragment如何实现Activity跳转,你怎么处理Fragment之间的信息交互 24](#_Toc374969537)

[98. fragment在哪个方法里实现代码 24](#_Toc374969538)

[99. res和assets资源文件的作用分别是什么，怎么读取 24](#_Toc374969539)

[100. 了解pad开发吗 24](#_Toc374969540)

[101. 常用的适配有哪些，如何进行优化 24](#_Toc374969541)

[102. 应用换肤有什么方案可以实现 24](#_Toc374969542)

[103. 异步任务中允许有多个线程同时进行吗 24](#_Toc374969543)

[104. 图片是怎么获取的 24](#_Toc374969544)

[105. 文件的下载 24](#_Toc374969545)

[106. 图片的下载 24](#_Toc374969546)

[107. 缓存区的图片如何适当的释放 24](#_Toc374969547)

[108. 图片缓存到哪里 24](#_Toc374969548)

[109. 那你的图片是如何请求的 24](#_Toc374969549)

[110. 图文混排怎么实现 24](#_Toc374969550)

[111. 相册缩略图是怎么实现的 24](#_Toc374969551)

[112. 下拉刷新的实现思路，反弹的效果是怎么实现的 24](#_Toc374969552)

[113. 自定义的View 24](#_Toc374969553)

[114. 自定义控件用过哪些？对自定义控件是怎么看的 24](#_Toc374969554)

[115. 瀑布流的实现思路 24](#_Toc374969555)

[116. 聊天窗口的实现 24](#_Toc374969556)

[117. android的dvm和Linux进程是一样的吗 24](#_Toc374969557)

[118. 监听模式 24](#_Toc374969558)

[119. 如何刷新view 24](#_Toc374969559)

[120. Listview 24](#_Toc374969560)

[121. viewPager左右切屏 24](#_Toc374969561)

[122. listview加载图片错位，什么原因导致的 24](#_Toc374969562)

[123. ListView中条目中显示webView会怎么样 24](#_Toc374969563)

[124. 用listview进行快速翻页，如何解决内存溢出 24](#_Toc374969564)

[125. listview快速滑动时，加载几条数据 24](#_Toc374969565)

[126. 实现listview分页 24](#_Toc374969566)

[127. 动画分为哪几种，各自的特点和区别 25](#_Toc374969567)

[128. 写出Socket服务器与客户端交互的代码和实现思路 25](#_Toc374969568)

[129. 谈谈对Http的理解 25](#_Toc374969569)

[130. 谈谈递归 25](#_Toc374969570)

[131. 写出Activity里的方法都是在什么情况下执行 25](#_Toc374969571)

[132. dp与sp的区别 25](#_Toc374969572)

[133. java有那些排序，冒泡代码 25](#_Toc374969573)

[134. Java线程实现方式和区别 25](#_Toc374969574)

[135. Android中布局、区别、应用场景 25](#_Toc374969575)

[136. 线程间通讯 25](#_Toc374969576)

[137. 进程间通讯 25](#_Toc374969577)

[138. 判断网络 25](#_Toc374969578)

[139. 网络代理 25](#_Toc374969579)

[140. HTTP的post和get方法的区别 25](#_Toc374969580)

[141. 怎么判断服务端返回的数据类型 25](#_Toc374969581)

[142. 栈，队列代码 25](#_Toc374969582)

[143. 服务service 25](#_Toc374969583)

[144. 多线程下载，离线下载 25](#_Toc374969584)

[145. 支付方式 26](#_Toc374969585)

[146. 如果想改变分辨率但不动原来布局，实现思路是什么 26](#_Toc374969586)

[147. java常用的处理异常的机制，原理及应用 26](#_Toc374969587)

[148. 一段字符串“abdecffr”,请输出只出现一次的字母 26](#_Toc374969588)

[149. 输入一个数，求其二进数中有多少个1 26](#_Toc374969589)

[150. 有一个pizza生产线 26](#_Toc374969590)

[151. 推送了解吗？推送底层代码掌握吗 26](#_Toc374969591)

[152. asynctack原理 26](#_Toc374969592)

[153. NDK具体说一下 26](#_Toc374969593)

[154. Service结合代码说一下，原理 26](#_Toc374969594)

[155. 当内存不足时，如何对尚未关闭的应用程序进行回收 26](#_Toc374969595)

[156. service的运行级别 26](#_Toc374969596)

[157. service如何一直保持在后台运行 26](#_Toc374969597)

[158. service的启动方式 26](#_Toc374969598)

[159. 软引用怎么用？为什么要用软引用 26](#_Toc374969599)

[160. 还有一个Service是建立在哪个线程上面 27](#_Toc374969600)

[161. 接口都有哪些内容 27](#_Toc374969601)

[162. 接口是否可继承接口 27](#_Toc374969602)

[163. 如何通过程序进行对服务器的接口调用，写出具体过程 27](#_Toc374969603)

[164. 分页加载思路 27](#_Toc374969604)

[165. 分页加载需要传递哪些参数 27](#_Toc374969605)

[166. 讯飞语音在什么地方用到 27](#_Toc374969606)

[167. 消息推送 27](#_Toc374969607)

[168. LBS定位用到哪个地图？为什么要使用这个？实现了哪些功能 27](#_Toc374969608)

[169. 地图中怎样获得附近的信息 27](#_Toc374969609)

[170. 百度地图显示周边招聘信息 27](#_Toc374969610)

[171. 移动互联使用了哪种方式 27](#_Toc374969611)

[172. 集合方面的底层理解 27](#_Toc374969612)

[173. 每个item都是不同的内容，怎么弄 27](#_Toc374969613)

[174. 有没有自己写过NDK 27](#_Toc374969614)

[175. 一个版本的app不更新，只是给他赠加或着减少一个功能，怎么弄 27](#_Toc374969615)

[176. 架构的情况 27](#_Toc374969616)

[177. 多线程的讲解 27](#_Toc374969617)

[178. 线程的异常怎么处理 27](#_Toc374969618)

[179. 请求网络用什么，http的使用 27](#_Toc374969619)

[180. 解释intent和activity的区别 27](#_Toc374969620)

[181. 如何处理内存溢出 27](#_Toc374969621)

[182. 如何去除一个字符串首尾的空格 27](#_Toc374969622)

[183. 如何把一个字符串转换成asll码 27](#_Toc374969623)

[184. 得到一个数组中出现次数最多的数 27](#_Toc374969624)

[185. 手机QQ聊天框冒泡效果怎么做 27](#_Toc374969625)

[186. groupby是干什么的 27](#_Toc374969626)

[187. 你android应用假如你手机设置了其他语言你的应用语言会变吗 27](#_Toc374969627)

[188. 了解后台吗，对后台的认识 27](#_Toc374969628)

[189. '学java'内存中占多少个字节 27](#_Toc374969629)

[190. Intent可以传递那些数据 28](#_Toc374969630)

[191. DDMS与TraceView的区别 28](#_Toc374969631)

[192. 横竖屏切换不设置属性会怎么样 28](#_Toc374969632)

[193. 单例类在android中哪种更合适 28](#_Toc374969633)

[194. 一串字符串编写程序翻转split（） 28](#_Toc374969634)

[195. json写实体类 28](#_Toc374969635)

[196. 如果标题过长超出界面，请写出你的解决方法 28](#_Toc374969636)

[197. 开发android应用怎样减少耗电量 28](#_Toc374969637)

[198. 提供额外的两个int域和一个Object域 28](#_Toc374969638)

[199. android开发过程有几种缓存方式，并说明缺点 28](#_Toc374969639)

[200. Activity相关 28](#_Toc374969640)

[201. Bitemap内存溢出解决方法 28](#_Toc374969641)

[202. 推送 28](#_Toc374969642)

[203. 文件传输 28](#_Toc374969643)

[204. 缓存app缓存 28](#_Toc374969644)

[205. 边距测试 28](#_Toc374969645)

[206. Java数据结构 28](#_Toc374969646)

[207. 常用的分辨率 28](#_Toc374969647)

[208. Ui一般有几套，是什么格式的，.9格式的还是什么格式的 28](#_Toc374969648)

[209. onCreateView和onViewCreated之间的区别 28](#_Toc374969649)

[210. 继承与组合的关系 28](#_Toc374969650)

[211. IPC机制具体解释 28](#_Toc374969651)

[212. aidl具体解释 28](#_Toc374969652)

[213. 为什么要有NDK 28](#_Toc374969653)

[214. 为什么json与js交互方便 28](#_Toc374969654)

[215. 进程与线程的区别 28](#_Toc374969655)

[216. 当一个activity退出之后在进程中会不会杀死 28](#_Toc374969656)

[217. jvm的内存分配 28](#_Toc374969657)

[218. MVC模式的理解 28](#_Toc374969658)

[219. 排序有了解过吗？及其实现复杂度 28](#_Toc374969659)

[220. android手机的密度，你如何对屏幕密度进行自适应 28](#_Toc374969660)

[221. handler的理解 29](#_Toc374969661)

[222. 对于instancevariable和localvarial,JVM在少女情怀内存时有什么区别 29](#_Toc374969662)

[223. Java线程都有哪些状态？列举你所知道的线程同步方法 29](#_Toc374969663)

[224. 如何避免比不加载图片混乱的问题 29](#_Toc374969664)

[225. AysnTask最多可以开启几个线程 29](#_Toc374969665)

[226. 数据库如何升级（添加字段什么的） 29](#_Toc374969666)

[227. WebService 29](#_Toc374969667)

[228. IntentService的用法 29](#_Toc374969668)

[229. service的启动 33](#_Toc374969669)

[230. 异步任务 33](#_Toc374969670)

[231. 强引用 33](#_Toc374969671)

[232. 新浪微博头像设置 33](#_Toc374969672)

[233. 加密算法 33](#_Toc374969673)

[234. 高德地图 33](#_Toc374969674)

[235. asynctask和handler的区别 33](#_Toc374969675)

[236. 跳转图片释放 33](#_Toc374969676)

[237. shareprefercnce保存用户名密码时候怎么加密 33](#_Toc374969677)

[238. 离线下载怎么实现 33](#_Toc374969678)

[239. 用过viewstub吗？include呢 33](#_Toc374969679)

[240. android应用程序自启动 33](#_Toc374969680)

[241. service是在主线程中吗？service中可不可以耗时操作 33](#_Toc374969681)

[242. 数据库优化 33](#_Toc374969682)

[243. socket 33](#_Toc374969683)

[244. 屏幕适配 33](#_Toc374969684)

[245. 关于图片压缩，用算法实现 33](#_Toc374969685)

[246. 关于第三方分享 33](#_Toc374969686)

[247. MVC在安卓的体现 33](#_Toc374969687)

[248. surfaceview和VIEW和GLsurfaceview的区别 34](#_Toc374969688)

[249. 都用过什么模式，经常用到的服务器是什么 34](#_Toc374969689)

[250. 编写代码输入安卓abc汉字6字节输出安卓abc而不是安卓abc氵 34](#_Toc374969690)

[251. 看一个for循环循环了几次 34](#_Toc374969691)

[252. 怎杨跳出一个循环继续执行下一个循环 34](#_Toc374969692)

[253. 使一个字符串逆序输出Stringa=abcde 34](#_Toc374969693)

[254. 一个for循环，每次i+=2，截取字符串输出。 34](#_Toc374969694)

[255. Math的操作，其实用了好几个方法，选择正确的结果 34](#_Toc374969695)

[256. 线程的状态 34](#_Toc374969696)

[257. 消息推送,xmpp轮询 34](#_Toc374969697)

[258. udp/tcp 34](#_Toc374969698)

[259. 动画 34](#_Toc374969699)

[260. intent的几种模式 34](#_Toc374969700)

[261. 分页加载，每页15条数据，拉到底部自动加载数据。 34](#_Toc374969701)

[262. Android系统架构从小到大写 34](#_Toc374969702)

[263. 用代码手写堆栈 34](#_Toc374969703)

[264. 即时通讯 34](#_Toc374969704)

[265. 多人聊天，视频，语音 34](#_Toc374969705)

[266. 侧拉效果 34](#_Toc374969706)

[267. 侧拉多界面，组件焦点 34](#_Toc374969707)

[268. 别的公司封装好的播放器，从C中提取源码 34](#_Toc374969708)

[269. AsyTask的三个参数是什么 34](#_Toc374969709)

[270. finish()与System.exit(0)的区别，再谈谈你对退出的理解。 34](#_Toc374969710)

[271. 反编译和防止反编译 34](#_Toc374969711)

[272. tcp/ip,通讯思路 35](#_Toc374969712)

[273. UDP，通讯思路 35](#_Toc374969713)

[274. hTTP请求网络流程，HTTPclient 35](#_Toc374969714)

[275. httpUrlconnection 35](#_Toc374969715)

[276. Java对象池 35](#_Toc374969716)

[277. Int与intger的区别 35](#_Toc374969717)

[278. 做项目的时候遇到哪些印象深刻的问题 35](#_Toc374969718)

[279. 你在安卓开发中的心得 35](#_Toc374969719)

[280. 网络互联使用哪种方式 35](#_Toc374969720)

[281. 动画 35](#_Toc374969721)

[282. 书架跟随书滚动 35](#_Toc374969722)

[283. 3D效果如何实现的 35](#_Toc374969723)

[284. 读取详细的新闻信息里面的图片读取多张不固定显示 35](#_Toc374969724)

[285. 服务器端使用什么搭建的 35](#_Toc374969725)

[286. 怎么调用android自带的功能 35](#_Toc374969726)

[287. 图片的优化 35](#_Toc374969727)

[288. 内存的优化 35](#_Toc374969728)

[289. Android每个进程分配的内存是多少 35](#_Toc374969729)

[290. 都使用过那些第三方平台 35](#_Toc374969730)

[291. 一个不同公司用一个APP的思路 35](#_Toc374969731)

[292. 平台兼容性 35](#_Toc374969732)

[293. 重复登录怎么解决比如两台设备同时登录一个账号 35](#_Toc374969733)

[294. 是每次获取图片都用AsyncTask获取吗 35](#_Toc374969734)

[295. 用户体验度：获取数据之前等待的时间是怎么做的 35](#_Toc374969735)

[296. 从网络获取图片的大概思路 35](#_Toc374969736)

[297. 是否做过视频类,关于视频播放使用的什么方式 35](#_Toc374969737)

[298. 大量耗时操作使用哪种方式,为什么 35](#_Toc374969738)

[299. 建立一个Android工程，实现一个简单的书籍管理应用 35](#_Toc374969739)

[300. 图片上传怎么做的 36](#_Toc374969740)

[301. 如果图片太大怎么处理 36](#_Toc374969741)

[302. 项目开发流程 36](#_Toc374969742)

[303. GSON的使用 36](#_Toc374969743)

[304. 讯飞语音在什么地方用到 36](#_Toc374969744)

[305. LBS定位用到哪个地图？为什么要使用这个？实现了哪些功能 36](#_Toc374969745)

[306. 分页加载需要传递哪些参数 36](#_Toc374969746)

[307. 移动互联使用了哪种方式 36](#_Toc374969747)

[308. java.exe指的是 36](#_Toc374969748)

[309. 掌握Android项目目录结构 36](#_Toc374969749)

[310. 掌握AndroidManifest.xml结构(应用程序入口) 37](#_Toc374969750)

[311. 掌握模拟器的使用以及常用adb命令 38](#_Toc374969751)

[二、 面试题\_Java 38](#_Toc374969752)

[(一) 概念 39](#_Toc374969753)

[1. 作用域public,private,protected,以及不写时的区别 39](#_Toc374969754)

[2. StaticNestedClass和InnerClass的不同 39](#_Toc374969755)

[3. &和&&的区别 39](#_Toc374969756)

[4. Collection和Collections的区别 39](#_Toc374969757)

[5. 什么时候用assert。 39](#_Toc374969758)

[6. Strings=newString("xyz");创建了几个StringObject 40](#_Toc374969759)

[7. Math.round(11.5)等於多少?Math.round(-11.5)等於多少 40](#_Toc374969760)

[8. shorts1=1;s1=s1+1;有什么错?shorts1=1;s1+=1;有什么错 40](#_Toc374969761)

[9. Java有没有goto 40](#_Toc374969762)

[10. 数组有没有length()这个方法?String有没有length()这个方法 40](#_Toc374969763)

[11. 给我一个你最常见到的runtimeexception 40](#_Toc374969764)

[12. error和exception有什么区别 40](#_Toc374969765)

[13. abstract的method 40](#_Toc374969766)

[14. 接口是否可继承接口 41](#_Toc374969767)

[15. 一个类实现接口，里面都变量Stringa; 41](#_Toc374969768)

[16. 构造器Constructor是否可被override 41](#_Toc374969769)

[17. 是否可以继承String类 41](#_Toc374969770)

[18. try{}里有一个return语句 41](#_Toc374969771)

[19. 用最有效率的方法算出2乘以8等於几 41](#_Toc374969772)

[20. 当一个对象被当作参数传递到一个方法后 41](#_Toc374969773)

[21. swtich是否能作用在byte上 41](#_Toc374969774)

[22. List,Set,Map是否继承自Collection接口 41](#_Toc374969775)

[23. ArrayList和Vector的区别,HashMap和Hashtable的区别 42](#_Toc374969776)

[24. 说出ArrayList,Vector, LinkedList的存储性能和特性 42](#_Toc374969777)

[25. char型变量中能不能存贮一个中文汉字?为什么 42](#_Toc374969778)

[26. 垃圾回收器的基本原理是什么？垃圾回收器可以马上回收内存吗 42](#_Toc374969779)

[27. 垃圾回收的优点和原理。并考虑2种回收机制。 42](#_Toc374969780)

[28. float型floatf=3.4是否正确 42](#_Toc374969781)

[29. 介绍JAVA中的CollectionFrameWork(包括如何写自己的数据结构) 42](#_Toc374969782)

[30. 实体Bean的三个状态： 43](#_Toc374969783)

[31. JDBC连接数据库6步 43](#_Toc374969784)

[32. 事务的4大特性 43](#_Toc374969785)

[33. 查询语句之间的区别 43](#_Toc374969786)

[34. 假设现在有表system.table1 43](#_Toc374969787)

[35. 创建CUSTOMERS表 44](#_Toc374969788)

[36. 使用左外连接查询，ORDERS和CUSTOMERS表， 44](#_Toc374969789)

[37. 简述数据库事务的生命周期？（可画流程图） 44](#_Toc374969790)

[38. deletefromtablea&truncatetabletablea的区别 44](#_Toc374969791)

[39. 什么是WebService? 44](#_Toc374969792)

[40. 运行时异常与一般异常有何异同？ 45](#_Toc374969793)

[41. 说出Servlet的生命周期，并说出Servlet和CGI的区别。 45](#_Toc374969794)

[42. 同步和异步有何异同，在什么情况下分别使用他们？举例说明。 45](#_Toc374969795)

[43. heap和stack有什么区别 45](#_Toc374969796)

[44. forward和redirect的区别 45](#_Toc374969797)

[45. 说出数据连接池的工作机制是什么 45](#_Toc374969798)

[46. Set里的元素是不能重复的，那么用什么方法来区分重复与否呢 45](#_Toc374969799)

[47. 两个对象值相同(x.equals(y) == true) 46](#_Toc374969800)

[48. 重写equals()方法的时候，我们还需要重写哪个方法？为什么 46](#_Toc374969801)

[49. 当一个对象被当作参数传递到一个方法后 46](#_Toc374969802)

[50. java中有几种方法可以实现一个线程 46](#_Toc374969803)

[51. 简述synchronized和java.util.concurrent.locks.Lock的异同 46](#_Toc374969804)

[52. 当一个线程进入一个对象的一个synchronized方法后 46](#_Toc374969805)

[53. Java的接口和C++的虚类的相同和不同处。 46](#_Toc374969806)

[54. Java中的异常处理机制的简单原理和应用。 47](#_Toc374969807)

[55. 描述一下JVM加载class文件的原理机制? 47](#_Toc374969808)

[56. 线程的基本概念、线程的基本状态以及状态之间的关系 47](#_Toc374969809)

[57. 什么情况下调用doGet()和doPost() 47](#_Toc374969810)

[58. 我们在web应用开发过程中经常遇到输出某种编码的字符 47](#_Toc374969811)

[59. 排序都有哪几种方法？请列举。用JAVA实现一个快速排序。 47](#_Toc374969812)

[60. JAVA语言如何进行异常处理 47](#_Toc374969813)

[61. 一个".java"源文件中是否可以包括多个类（不是内部类）？有什么限制 48](#_Toc374969814)

[62. java中有几种类型的流 48](#_Toc374969815)

[63. 字节流与字符流的区别 48](#_Toc374969816)

[64. java中会存在内存泄漏吗，请简单描述。 49](#_Toc374969817)

[65. 什么是java序列化，如何实现java序列化？ 49](#_Toc374969818)

[66. 是否可以从一个static方法内部发出对非static方法的调用 50](#_Toc374969819)

[67. 写clone()方法时，通常都有一行代码，是什么 50](#_Toc374969820)

[68. 在JAVA中，如何跳出当前的多重嵌套循环 50](#_Toc374969821)

[69. 说出一些常用的类，包，接口，请各举5个 50](#_Toc374969822)

[70. 内部类可以引用他包含类的成员吗？有没有什么限制 50](#_Toc374969823)

[71. 面向对象的特征有哪些方面 50](#_Toc374969824)

[72. javac命令和java命令做什么事情呢 50](#_Toc374969825)

[73. Class.forName的作用?为什么要用 51](#_Toc374969826)

[74. 为什么要用 ORM? 和 JDBC 有何不一样? 51](#_Toc374969827)

[75. JDBC中的PreparedStatement相比Statement的好处 51](#_Toc374969828)

[76. 静态变量和实例变量的区别 51](#_Toc374969829)

[77. 请说出作用域public，private，protected，以及不写时的区别 52](#_Toc374969830)

[78. 构造函数与方法的区别 52](#_Toc374969831)

[79. 对于instance varisble 和 local variable jvm 再申请内存时有什么区别 53](#_Toc374969832)

[80. 如何设置两个应用在一个进程中 54](#_Toc374969833)

[81. android中常用的权限，最少写三个 54](#_Toc374969834)

[82. IO问题 54](#_Toc374969835)

[83. java内存中占多少个字节 54](#_Toc374969836)

[84. Android开发心得:思想，进程间通信，四大组件 54](#_Toc374969837)

# 面试题\_Android

1. Overload和Override的区别

Overloaded的方法是否可以改变返回值的类型?

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。如果在子类中定义某方法与其父类有相同的名称和参数，我们说该方法被重写 (Overriding)。子类的对象使用这个方法时，将调用子类中的定义，对它而言，父类中的定义如同被"屏蔽"了。如果在一个类中定义了多个同名的方法，它们或有不同的参数个数或有不同的参数类型，则称为方法的重载(Overloading)。Overloaded的方法是可以改变返回值的类型。

1. String 和StringBuffer的区别

答: STRING的长度是不可变的，STRINGBUFFER的长度是可变的。如果你对字符串中的内容经常进行操作，特别是内容要修改时，那么使用StringBuffer，如果最后需要String，那么使用StringBuffer（缓存）的toString()方法。

StringBuilder：节约内存，创建字符串；

1. 实现一个字符串倒序

字符串“abcde”通过写一个函数不让调用第三方的字符串，实现一个字符串倒序，比如字符串“abcde” 变成“edcba”

答： String src = "ABCDEF ";  
String dst = new StringBuffer(src).reverse().toString();

1. 抽象类与接口的区别（abstract与interface的区别）

答：abstract可以修饰抽象方法，而一个类只要有一个抽象方法，就必须用abstract定义该类，即抽象类。抽象类，被继承，实现它里面的方法，可以只有普通方法

用interface修饰的类，里面的方法都是抽象方法（不能实例化），因此在定义接口的时候，可以直接不加那些修饰，系统会默认的添上去。接口里面的字段都是公有常量，即public static final修饰的字段。可以没有函数原型，里面可以是空的

1. 集合的实现类与区别

答:Collection接口，集合结构总的父接口，有两个子接口list和set

List接口 元素有序可重复.

实现类有：ArrayList 数组实现轻量级，运行快，线程不安全。JDK1.2 查询快

Vector 数组实现重量级，运行慢，线程安全。JDK1.0

LinkedList链表实现 常用语堆栈与队列的实现 增删操作快

Set 接口 元素无序不可重复

实现类有：HashSet，底层用hashCode()算法实现，保证元素的无序唯一，自定义对象存进HashSet为了保证元素内容不重复需要覆盖hashCode()与equals()方法。

SortedSet(不重要) 元素有序（Unicode升序）唯一

TreeSet要求元素有序，自定义的对象需要实现Comparable接口的 compareTo（object o）方法

Map(接口): 与Collection接口无关,有一个子接口SortedMap特点: 元素是key-value, key

唯一,无序; value可重复

实现类: HashMap 轻量级 线程不安全的,允许key或value为null JDK1.2

HashTable 重量级 线程安全的 不允许key或value为null JDK1.0

Properties是HashTable的子类,主键和值都是字符串

SortedMap:(不重要)

特点: key唯一,有序(Unicode升序)

实现类:TreeMap

1. 线程有几种状态,分别是哪些（调用run()和调用start()的区别）

答：1)新建状态(New)：新创建了一个线程对象。

　　2)就绪状态(Runnable)：线程对象创建后，其他线程调用了该对象的start()方法。该状态的线程位于可运行线程池中，变得可运行，等待获取[CPU](http://product.it168.com/list/b/0217_1.shtml)的使用权。

　　3)运行状态(Running)：就绪状态的线程获取了CPU，执行run()方法。

　　4)阻塞状态(Blocked)：阻塞状态是线程因为某种原因放弃CPU使用权，暂时停止运行。直到线程进入就绪状态，才有机会转到运行状态。阻塞的情况分三种：

①等待阻塞：运行的线程执行wait()方法，JVM会把该线程放入等待池中。

②同步阻塞：运行的线程在获取对象的同步锁时，若该同步锁被别的线程占用，则JVM会把该线程放入锁池中。

③其他阻塞：运行的线程执行sleep()或join()方法，或者发出了I/O请求时，JVM会把该线程置为阻塞状态。当sleep()状态超时、join()等待线程终止或者超时、或者I/O处理完毕时，线程重新转入就绪状态。

5)死亡状态(Dead)：线程执行完了或者因异常退出了run()方法，该线程结束生命周期。

当调用start方法的时候，该线程就进入就绪状态。等待CPU进行调度执行，此时还没有真正执行线程。

当调用run方法的时候，是已经被CP U进行调度，执行线程的主要任务。

1. 线程的实现方式

答：线程的实现有两种方式，一是继承Thread类，二是实现Runnable接口

1. sleep() 与 wait()的区别

答：①这两个方法来自不同的类分别是，sleep来自Thread类，和wait来自Object

类。

②最主要是sleep方法没有释放锁，而wait方法释放了锁，使得其他线程可

以使用同步控制块或者方法。sleep不出让系统资源；wait是进入线程等待池

等待，出让系统资源，其他线程可以占用CPU。一般wait不会加时间限制，

因为如果wait线程的运行资源不够，再出来也没用，要等待其他线程调用

notify/notifyAll唤醒等待池中的所有线程，才会进入就绪队列等待OS分配系

统资源。sleep(milliseconds)可以用时间指定使它自动唤醒过来，如果时间不到

只能调用interrupt()强行打断。

3.wait，notify和notifyAll只能在同步控制方法或者同步控制块里面使用，而

sleep可以在任何地方使用

4. Sleep需要捕获异常,而wait不需要

1. 线程中wait，join，sleep，yield, notify，notifyall，synchronized，区别及联系

答：1).sleep()方法

在指定时间内让当前正在执行的线程暂停执行，但不会释放“锁标志”。不推荐使用。sleep()使当前线程进入阻塞状态，在指定时间内不会执行。

2).wait()方法

在其他线程调用对象的notify或notifyAll方法前，导致当前线程等待。线程会释放掉它所占有的“锁标志”，从而使别的线程有机会抢占该锁。

唤醒当前对象锁的等待线程使用notify或notifyAll方法,waite() 和notify()必须在synchronized函数或synchronized　block中进行调用。3.yield方法

暂停当前正在执行的线程对象。yield()只是使当前线程重新回到可执行状态，所以执行3)yield()的线程有可能在进入到可执行状态后马上又被执行。yield()只能使同优先级或更高优先级的线程有执行的机会。

4).join方法

等待该线程终止。等待调用join方法的线程结束，再继续执行。如：t.join();//主要用于等待t线程运行结束，若无此句，main则会执行完毕，导致结果不可预测。

1. Final、finally、finanlize()的区别

答：final用于声明属性，方法和类，分别表示属性不可变，方法不可覆盖，类不可继承。

finally是异常处理语句结构的一部分，表示总是执行。

finalize是Object类的一个方法，在垃圾收集器执行的时候会调用被回收对象的此方法，可以覆盖此方法提供垃圾收集时的其他资源回收，例如关闭文件等。

1. J2EE是什么？请对以下在J2EE中常用的名词进行解释(或简单描述)

J2EE是Sun公司提出的多层(multi-diered),分布式(distributed),基于组件(component-base)的企业级应用模型(enterprieseapplicationmodel).在这样的一个应用系统中，可按照功能划分为不同的组件，这些组件又可在不同计算机上，并且处于相应的层次(tier)中。所属层次包括客户层(clietntier)组件,web层和组件,Business层和组件,企业信息系统(EIS)层。

1. j2ee常用的设计模式？说明工厂模式

Java中的23种设计模式：

Factory（工厂模式），Singleton（单例模式），Adapter（适配器模式），FactoryMethod（工厂方法模式），Prototype（原始模型模式），Facade（门面模式），

Bridge（桥梁模式），Composite（合成模式），Builder（建造模式），

Decorator（装饰模式），Flyweight（享元模式），Proxy（代理模式），

Command（命令模式），Interpreter（解释器模式），Visitor（访问者模式），

Iterator（迭代子模式），Mediator（调停者模式），Memento（备忘录模式），

Observer（观察者模式），State（状态模式），Strategy（策略模式），

TemplateMethod（模板方法模式），ChainOfResponsibleity（责任链模式）

工厂模式：工厂模式是一种经常被使用到的模式，根据工厂模式实现的类可以根据提供的数据生成一组类中某一个类的实例，通常这一组类有一个公共的抽象父类并且实现了相同的方法，但是这些方法针对不同的数据进行了不同的操作。首先需要定义一个基类，该类的子类通过不同的方法实现了基类中的方法。然后需要定义一个工厂类，工厂类可以根据条件生成不同的子类实例。当得到子类的实例后，开发人员可以调用基类中的方法而不必考虑到底返回的是哪一个子类的实例。

1. 开发中都用到了那些设计模式?用在什么场合

每个模式都描述了一个在我们的环境中不断出现的问题，然后描述了该问题的解决方案的核心。通过这种方式，你可以无数次地使用那些已有的解决方案，无需在重复相同的工作。主要用到了MVC的设计模式。用来开发JSP/Servlet或者J2EE的相关应用。简单工厂模式等。

1. 常用设计模式及应用场景

常用设计模式及应用场景，用两种方式实现单例模式，要求线程安全

答: 常用设计模式:

单例模式: Calendar实例的获取

适配器模式: Adapter 为ListView GridView等添加数据

工厂模式: Spring IOC 反转控制

代理模式: Spring AOP 面向切面编程

观察者模式: ContentObserver监听内容改变

（懒汉式）程序执行过程中需要这个类的对象时再实例化该类的对象

步骤

1)定义静态私有对象

2)构造方法私有化保证在类的外部无法实例化该类的对象

3)定义对外开放的静态方法在调用方法是判断对象是否为空，为空再创建对象返回

public class Singleton {

private static Singleton singleton;

// 构造方法私有化，保证在类的外部无法实例化该类的对象

private Singleton() {

}

public static synchronized Singleton getSingletonInstance() {

if (singleton == null) {

singleton = new Singleton();

}

return singleton;

}

}

（饿汉式）类加载的时候就实例化该类的对象

public class Singleton {

private static Singleton singleton = new Singleton();

// 构造方法私有化，保证在类的外部无法实例化该类的对象

private Singleton() {

}

public static Singleton getSingletonInstance() {

return singleton;

}

}

1. 广播如何调用，有什么方式，各自的区别，实现广播的意图是什么，哪里用到

答：程序中发送广播通过sendBroadcastReceiver（）实现

接收广播通过定义一个类继承BroadcastReceiver并重写onReceive（）方法实现

注册广播有两种方式：

第一种静态方式:在清单文件中通过<receive>标签声明

第二种代码动态方式:

IntentFilter filter = new IntentFilter("android.provider.Telephony.SMS\_RECEIVED");

IncomingSMSReceiver receiver = new IncomgSMSReceiver();

registerReceiver(receiver.filter);

1)第一种不是常驻型广播，也就是说广播跟随activity的生命周期。注意: 在activity结束前，移除广播接收器。

2)第二种是常驻型，也就是说当应用程序关闭后，如果有信息广播来，程序也会被系统调用自动运行。

1. android系统架构

答：

1）应用程序层 java语言 应用程序开发

1. 应用程序框架层 java语言 OS定制 framework层开发
2. 系统运行库层 C C++ 实现 so库
3. Linux内核层
4. Android的四大组件是什么？它们的作用是什么

答:

Android有四大组件：Activity、Service、Broadcast Receiver、Content Provider。

Activity :应用程序中，一个Activity通常就是一个单独的屏幕，它上面可以显示一些控件也可以监听并处理用户的事件做出响应。Activity之间通过Intent进行通信。

Service 服务:一个Service 是一段长生命周期的，没有用户界面的程序，可以用来开发如监控类程序。

BroadcastReceive广播接收器:你的应用可以使用它对外部事件进行过滤只对感兴趣的外部事件(如当电话呼入时，或者数据网络可用时)进行接收并做出响应。广播接收器没有用户界面。然而，它们可以启动一个activity或serice 来响应它们收到的信息。

Content Provider内容提供者 :主要用于多个应用间数据共享。这些数据可以存储在文件系统中或SQLite数据库。

1. Activity生命周期？保存activity的一些信息在哪个生命周期方法中

横竖屏切换activity的生命周期

答：共有七个周期函数：

void onCreate(Bundle savedInstanceState) 第一次创建时调用

void onStart() 被用户可见时调用

void onRestart() 当Activity处于stop状态又被重新启动时调用

void onResume() 当获得焦点即可与用户交互时调用

void onPause() 当失去焦点时调用

void onStop() 当不可见时调用

void onDestroy() 当销毁时调用

1. Activity的 onSaveInstanceState() 和 onRestoreInstanceState()

答: Activity的 onSaveInstanceState() 和 onRestoreInstanceState()并不是生命周期方法，它们不同于 onCreate()、onPause()等生命周期方法，它们并不一定会被触发。当应用遇到意外情况（如：内存不足、用户直接按Home键）由系统销毁一个Activity时，onSaveInstanceState() 会被调用。但是当用户主动去销毁一个Activity时，例如在应用中按返回键，onSaveInstanceState()就不会被调用。因为在这种情况下，用户的行为决定了不需要保存Activity的状态。通常onSaveInstanceState()只适合用于保存一些临时性的状态，而onPause()适合用于数据的持久化保存。

另外，当屏幕的方向发生了改变， Activity会被摧毁并且被重新创建，如果你想在Activity被摧毁前缓存一些数据，并且在Activity被重新创建后恢复缓存的数据。可以重写Activity的 onSaveInstanceState() 和 onRestoreInstanceState()方法。

1. activity在屏幕旋转时的生命周期

答：不设置Activity的android:configChanges时，切屏会重新调用各个生命周期，切横屏时会执行一次，切竖屏时会执行两次；设置Activity的android:configChanges="orientation"时，切屏还是会重新调用各个生命周期，切横、竖屏时只会执行一次；设置Activity的android:configChanges="orientation|keyboardHidden"时，切屏不会重新调用各个生命周期，只会执行onConfigurationChanged方法

1. Android中asset文件夹和raw文件夹区别

答：res/raw和assets的相同点：

两者目录下的文件在打包后会原封不动的保存在apk包中，不会被编译成二进制。

res/raw和assets的不同点：

1）res/raw中的文件会被映射到R.java文件中，访问的时候直接使用资源ID即 R.raw.filename；assets文件夹下的文件不会被映射到R.java中，访问的时候需要AssetManager类。

2）res/raw不可以有目录结构，而assets则可以有目录结构，也就是assets目录下可以再建立文件夹

3）读取文件资源举例：

读取res/raw下的文件资源，通过以下方式获取输入流来进行写操作

InputStream is = getResources().openRawResource(R.raw.filename);

读取assets下的文件资源，通过以下方式获取输入流来进行写操作

AssetManager am = null;

am = getAssets();

InputStream is = am.open("filename");

1. Android中的五种存储方式及其应用场景

答：1)SharedPreferences

存储路径:(data/data/packagename/shares\_prefs), 轻量级存储,以键值对的形式存储在xml中,一般用来保存应用中的设置属性

2)文件存储 SD卡存储多媒体文件, 文件缓存

3) Sqlite数据库 存储路径:(data/data/packagename/databases), 一种嵌入式数据库,支持sql语言,存储大量结构性数据

4)ContentProvider 进程(应用程序)间数据共享,数据源可以是sqlite,也可以是xml,相关类: ContentResolver(内容解析器), ContentObserver(数据 观察者)

5) 网络存储 天气数据的xml,json格式等等,通过HttpUrlConnection,HttpClient,或者SOAP协议获取数据

1. 什么是ANR 如何避免它？（Android线程间的通信方式 ）

答：ANR：Application Not Responding(应用程序无响应).当出现下列情况时，Android就会显示ANR对话框了： 对输入事件（如按键、触摸屏事件）的响应超过5秒 意向接受器（intentReceiver）超过10秒钟仍未执行完毕Android应用程序完全运行在一个独立的线程中（例如main）。这就意味着，任何在主线程中运行的，需要消耗大量时间的操作都会引发ANR。

解决方案有两种:

1)AsyncTask异步任务中，doInBackground()和onPostExecute(Result)两个方法非常重要

doInBackground() 这个方法运行在后台线程中，主要负责执行那些很耗时的操作，如移动护理系统中的网络连接、解析XML等操作。该方法必须重载。

onPostExecute(Result) 这个方法也运行于UI线程，在doInBackground(Params…)方法执行后调用，该方法用于处理后台任务执行后返回的结果。

2) 子thread + handler

1. listview优化策略

答：1)对convetView进行判空，是当convertView不为空的时候直接重新使用convertView

从而减少了很多不必要的View的创建

2)定义一个ViewHolder，将convetView的tag设置为ViewHolder,不为空时重新使用即可

3)当ListView加载数据量较大时可以采用分页加载和图片异步加载

1. ListView分页加载实现思路

实现OnScrollListener 接口重写onScrollStateChanged 和onScroll方法，使用onscroll方法实现”滑动“后处理检查是否还有新的记录，如果有，调用 addFooterView，添加记录到adapter, adapter调用 notifyDataSetChanged 更新数据;如果没有记录了，把自定义的mFooterView去掉。使用onScrollStateChanged可以检测是否滚到最后一行且停止滚动然后执行加载

1. ListView图片异步加载实现思路

1)先从内存缓存中获取图片显示（内存缓冲）

2)获取不到的话从SD卡里获取（SD卡缓冲，从SD卡获取图片是放在子线程里执行的，否则快速滑屏的话会不够流畅）

3)都获取不到的话从网络下载图片并保存到SD卡同时加入内存并显示（视情况看是否要显示）

1. Intent的原理、作用、可以传递哪些类型的参数

答：

intent是连接Activity, Service, BroadcastReceiver, ContentProvider四大组件的信使,，可以传递八种基本数据类型以及string, Bundle类型，以及实现了Serializable或者Parcelable的类型。

Intent可以划分成显式意图和隐式意图。

显式意图：调用Intent.setComponent()或Intent.setClass()方法明确指定了组件名的Intent为显式意图，显式意图明确指定了Intent应该传递给哪个组件。

隐式意图：没有明确指定组件名的Intent为隐式意图。 Android系统会根据隐式意图中设置的动作(action)、类别(category)、数据（URI和数据类型）找到最合适的组件来处理这个意图。

1. 如何实现屏幕分辨率的自适应（布局、九线图、目录）

答: 最好可以通过权重(layout\_weight)的方式来分配每个组件的大小，也可以通过具体的像素(dip)来确定大小。

尽量使用Relativelayout 。

已知应用支持平台设备的分辨率,可以提供多个layout\_320\*480 ...

drawable-hdpi,drawable-mdpi,drawable-ldpi分别代表分辨率为480\*800,360\*480,240\*360, 放置图片大小相差1.5倍

最后还需要在AndroidManifest.xml里添加下面一段，没有这一段自适应就不能实现：

<supports-screens

android:largeScreens="true"

android:normalScreens="true"

android:anyDensity = "true"/>

在</application>标签和</manifest> 标签之间添加上面那段代码。即可。

备注：三者的解析度不一样，就像你把电脑的分辨率调低，图片会变大一样，反之分辨率高，图片缩小

还可以通过.9.png实现图片的自适应

1. 简述Android中的IPC机制

IPC（Inter-Process Communication，进程间通信）,aidl是 Android Interface definition language的缩写，它是一种android内部进程通信接口的描述语言，通过它我们可以定义进程间的通信接口.编译器可以通过扩展名为aidl的文件生成一段代码，通过预先定义的接口达到两个进程内部通信进程的目的.

BroadcastReceiver也可以实现进程间通信

ContentProvider 提供进程间数据共享

1. Android程序入口如何判断

action节点中的android.intent.action.MAIN表明它所在的Activity是整个应用程序的入口点

1. android哪几种方式访问网络

http协议：超文本传输协议

(最底层的是Socket,接着是URLConnection，HttpClient)

HttpURLConnection

HttpClient方式（HttpGet和HttpPost类）

1. 说说HttpClient的通信过程

1)生成请求对象（HttpGet get，HttpPost post）

2)生成客户端对象 HttpClient client

3)执行请求接收相应 HttpResponse response = client.execute(post)

HttpEntity entity = response.getEntity()

4)得到数据流

InputStream inputStream = entity.getContent();

5)最后关闭过期连接

1. 移动互联数据交互格式有哪些及其区别

（Json（有数据类型）与xml（没有）的区别？）

移动互联数据交互格式有XML和JSON

1)JSON和XML的数据可读性基本相同

2)JSON和XML同样拥有丰富的解析手段

3)JSON相对于XML来讲，数据的体积小

4)JSON与JavaScript的交互更加方便

5)JSON对数据的描述性比XML较差

1. XML解析有哪几种？各自优缺点，官方推荐使用哪种

基本的解析方式有三种: DOM,SAX,Pull  
1)dom解析解析器读入整个文档，然后构建一个驻留内存的树结构，然后代码就可以使用 DOM 接口来操作这个树结构的优点是对文档增删改查比较方便，缺点占用内存比较大。  
2)sax解析基于事件驱动型,优点占用内存少，解析速度快，缺点是只适合做文档的读取，不适合做文档的增删改查。

3)pull解析同样基于事件驱动型,android 官方API提供,可随时终止,调用next() 方法提取它们（主动提取事件）

1. sax解析代码

首先SAXParserFactory来创建一个SAXParserFactory实例

SAXParserFactory factory = SAXParserFactory.newInstance();

根据SAXParserFactory实例来创建SAXParser

SAXParser产生SAXReader

XMLReader reader = factory.newSAXParser().getXMLReader();

XMLReader 加载XML，然后解析XML，在解析的过程中触发相对于接口中的事件处理程序

1. 百度地图核心类，及实现的功能

BMapManager:地图引擎管理类,负责初始化，开启地图API，终止百度地图API等工作

MKSearch：搜索服务.用于位置检索、周边检索、范围检索、公交检索、驾乘检索、步行检索

MKSearchListener搜索结果通知接口。该接口返回poi搜索,公交搜索,驾乘路线,步行路线结果

MapView:显示地图的View

MyLocationOverlay:一个负责显示用户当前位置的Overlay。

Overlay:Overlay是一个基类，它表示可以显示在地图上方的覆盖物。

1. GC内存泄露在什么情况下会出现？怎么解决
2. 查询数据库没有关闭游标
3. 构造Adapter时，没有使用缓存的 convertView
4. Bitmap对象不在使用时调用recycle()释放内存
5. 不用的对象没有及时释放对象的引用
6. android内存的优化

答：android内存泄露容易导致内存溢出，又称为OOM。

Android内存优化策略：

1）在循环内尽量不要使用局部变量  
2）不用的对象即时释放，即指向NULL

3）数据库的cursor即时关闭。  
4）构造adapter时使用缓存contentview  
5）调用registerReceiver()后在对应的生命周期方法中调用unregisterReceiver()  
6）即时关闭InputStream/OutputStream。  
7）android系统给图片分配的内存只有8M, 图片尽量使用软引用, 较大图片可通过BitmapFactory缩放后再使用,并及时recycle   
8）尽量避免static成员变量引用资源耗费过多的实例。

1. 加载大图片的时候如何防止内存溢出

答: android系统给图片分配的内存只有8M,当加载大量图片时往往会出现OOM。

Android加载大量图片内存溢出解决方案：

1）尽量不要使用setImageBitmap或setImageResource或BitmapFactory.decodeResource来设置一张大图，因为这些函数在完成decode后，最终都是通过java层的createBitmap来完成的，需要消耗更多内存，可以通过BitmapFactory.decodeStream方法，创建出一个bitmap，再将其设为ImageView的 source

2）使用BitmapFactory.Options对图片进行压缩

InputStream is = this.getResources().openRawResource(R.drawable.pic1);  
 BitmapFactory.Options options=new BitmapFactory.Options();  
 options.inJustDecodeBounds = false;  
 options.inSampleSize = 10; //width，hight设为原来的十分一  
 Bitmap btp =BitmapFactory.decodeStream(is,null,options);   
3）运用Java软引用，进行图片缓存，将需要经常加载的图片放进缓存里，避免反复加载

及时销毁不再使用的Bitmap对象

if(!bmp.isRecycle() ){  
 bmp.recycle() //回收图片所占的内存  
 system.gc() //提醒系统及时回收  
 }

1. Android缓存机制

答: 客户端缓存机制是android应用开发中非常重要的一项工作,使用缓存机制不仅仅可以为用户节省3G流量,同时在用户体验方面也是非常好的选择，比如有些新闻客户端支持离线模式，也是通过缓存机制实现的.缓存机制分为两部分,一部分是文字缓存,另一部分是多媒体文件缓存.

文字缓存有两种实现：

1）可以将与服务器交互得到的json数据或者xml数据存入sd卡中，并在数据库添加该数据的记录.添加数据库记录时,提供两个关键字段,一个是请求的URL,另一个则是本地保存后的文件地址，每次加载数据之前都会根据URL在数据库中检索

2）将JSON数据解析后装入List<Map>对象中,然后遍历List,将数据统统写入相应的数据库表结构中,以后每次向服务器发起请求之前可以先在数据库中检索,如果有直接返回.

多媒体文件缓存：主要指图片缓存

图片的缓存可以根据当前日期,时间为名字缓存到SD卡中的指定图片缓存目录,同时数据库中做相应记录,记录办法可以采用两个关键字段控制,一个字段是该图片的URL地址,另一个字段是该图片的本机地址.取图片时根据URL在数据中检索,如果没有则连接服务器下载,下载之后再服务器中作出相应记录

缓存文件删除策略:

1) 每一个模块在每次客户端自动或者用户手动更新的时候删除相应模块的缓存文件,并重新下载新的缓存文件.

2)在设置界面中提供删除缓存的功能,点击后删除本机所有缓存.

1. 如何实现消息推送

有哪些方式，各自优缺点，最常使用哪种

答：实现消息推送的方式有五种，分别是轮询，SMS,C2DM,MQTT,XMPP最常使用的是XMPP, 我们做项目时采用的是XMPP协议

1)XMPP协议，它是一种基于XML的传递协议，具有很强的灵活性和可扩展性。它的特点是将复杂性从客户端转移到了服务器端。GTalk、QQ、IM等都用这个协议。

2)轮询:客户端定时去服务端取或者保持一个长Socket，从本质讲这个不叫推送，而是去服务端拽数据。但是实现简单，主要缺点：耗电,浪费用户流量等  
3)Google的C2DM，具体不细说，缺点，服务器在国外，不是很稳定。

4)通过短信方式, 但是很难找到免费短信平台

5.)MQTT协议, IBM提供的一种推送服务,不太灵活

1. MVC在Android中的应用

答: Android中界面部分也采用了当前比较流行的MVC框架，在Android中：

1) 视图层（View）：一般采用XML文件进行界面的描述，使用的时候可以非常方便的引入。也可以使用JavaScript+HTML等的方式作为View层，通过WebView组件加载,同时可以实现Java和JavaScript之间的通信。

2) 控制层（Controller）：这句话也就暗含了不要在Acitivity中写代码，要通过Activity交割Model业务逻辑层处理，这样做的另外一个原因是Android中的Acitivity的响应时间是5s，如果耗时的操作放在这里，Android的控制层的重任通常落在了众多的Acitvity的肩上，程序就很容易被回收掉。

3) 模型层（Model）：对数据库的操作、对网络等的操作都应该在Model里面处理，当然对业务计算等操作也是必须放在的该层的。  
 在Android SDK中的数据绑定，也都是采用了与MVC框架类似的方法来显示数据。在控制层上将数据按照视图模型的要求（也就是Android SDK中的Adapter）封装就可以直接在视图模型上显示了，从而实现了数据绑定。比如显示Cursor中所有数据的ListActivity，其视图层就是一个ListView，将数据封装为ListAdapter，并传递给ListView，数据就在ListView中显示。

1. Android自定义组件实现思路

答: Android自定义组件有三种实现思路:

1. 继承某个现有组件，在其基础上添加额外功能,如继承Gallery实现CoverFlow效果
2. 继承某个Layout，实现复合组件自定义，如TextView和EditText组合实现登录注册组件
3. 继承View，实现onDraw()方法，实现自己绘制组件，如翻页效果组件
4. 版本更新的实现思路

答：在服务器相应URL上有版本文件, 客户端同时存储该应用当前版本号 (SharedPreferences/Sqlite), 每次打开应用,去检测服务器版本号与本地版本号是否一致,如果不一 致,则自定义对话框提示是否下载更新

1. 播放视频有哪些实现方式

答：

1)使用系统自带的播放器来播放，指定Action为ACTION\_VIEW,Data为Uri，Type为其MIME类型。

//调用系统自带的播放器

Intent intent = new Intent(Intent.ACTION\_VIEW);

intent.setDataAndType(uri, "video/mp4");

startActivity(intent);

2)使用VideoView组件来播放, 可以结合MediaController来实现播控, 只是不能随意更改视频的大小及位置。

3)使用MediaPlayer和SurfaceView来实现，这种方式很灵活,可以自定义视频播放的大小和位置。

1. NDK开发流程？（JNI运行原理）

答: NDK应用的开发流程(在应用中定义本地接口(native), 编译成.h头文件,交由C程序员实现,将.c实现通过NDK编译成.so动态链接库,导入项目中libs/armeabi,代码中调用该本地接口)

应用场景: 音频,视频解码,拍摄车牌号,识别车牌号

1. 如何实现一键退出

答：定义一个类继承Application，定义一个集合存放所有的activity，

定义一个添加的方法，再写一个退出的方法，使用for循环全部调用finish方法，然

后在每个Activity的onCreate方法中调用自定义类里的添加方法，然后在需要使用一

键退出的地方调用类中的退出方法即可。

1. AndroidManifest.xml清单文件<Activity>标签中属性的含义

AndroidManifest.xml清单文件<Activity>标签中属性android:excludeFromRecents="true" android:screenOrientation="portrait" android:configChanges="orientation|locale"的含义

答: android:excludeFromRecents表示是否可被显示在最近打开的activity列表里,true表示否,false表示是

android:screenOrientation表示activity显示的模式, 一般用来设置activity横屏显示(horizontal)或竖屏显示(portrait)

android:configChanges=[oneormoreof:"mcc""mnc""locale""touchscreen""keyboard""keyboardHidden""navigation""orientation""fontScale"]  
是当所指定属性(Configuration Changes)发生改变时，通知程序调用 onConfigurationChanged()函数,比如orientation屏幕方向发生改变,locale语言环境发生改变时

1. 如何将一个Activity设置成窗口的样式

答：在清单文件AndroidManifest.xml中相应的<activity>标签内设置属性android:theme=”@android:style/Theme.Dialog”

1. 谈谈UI中， Padding和Margin有什么区别,gravity与layout\_gravity的区别

答：Padding 用来指定组件内的内容距离组件边界的距离;

Margin用来指定控件与控件之间的距离

Gravity用来指定组件内的内容相对于组件本身的位置

Layout\_gravity用来指定组件相对于其父组件的位置

1. 实现手风琴效果

哪个组件可以实现手风琴效果，用来实现设置界面的类，实现抽屉效果, 悬浮窗口？

答：实现手风琴效果（ExpandableListView）

设置界面的类（preferenceActivity）保存到sharedpreference中

抽屉效果（slidingDrawer）组件

悬浮窗口: PopWindow,可以实现类似Dialog和菜单的效果

1. Android SDK 3.0(HoneyComb)及4.0(Ice Cream)新特性

答：新版SDK发布的同时也发布了一个扩展包android-support-v4，把部分特性单独的抽出来，使低版本的SDK也可以使用这些特性，主要支持以下特性：

Fragment: 3.0引入,碎片管理,可以局部刷新UI,它设计的功能和Activity一样强大，包括生命周期、导航等，Fragment的每次导航都可以记录下来用于返回。

ViewPager: 提供了多界面切换的新效果

GridLayout: 4.0引入, 网格布局, android第六大布局

Loader: 装载器从android3.0开始引进。它使得在activity或fragment中异步加载数据变得简单

1. android客户端如何实现自动登录

答: 通过SharedPreferences存储用户名,密码,当存储不为空时实现自动登录功能

1. 请介绍下Android中常用的五种布局。

常用五种布局方式，分别是：FrameLayout（框架布局），LinearLayout （线性布局），AbsoluteLayout（绝对布局），RelativeLayout（相对布局），TableLayout（表格布局）。

1)FrameLayout：所有东西依次都放在左上角，会重叠，这个布局比较简单，也只能放一点比较简单的东西。2)LinearLayout：线性布局，每一个LinearLayout里面又可分为垂直布局（android:orientation="vertical"）和水平布局（android:orientation="horizontal" ）。当垂直布局时，每一行就只有一个元素，多个元素依次垂直往下；水平布局时，只有一行，每一个元素依次向右排列。

3)AbsoluteLayout：绝对布局用X,Y坐标来指定元素的位置，这种布局方式也比较简单，但是在屏幕旋转时，往往会出问题，而且多个元素的时候，计算比较麻烦。

4)RelativeLayout：相对布局可以理解为某一个元素为参照物，来定位的布局方式。主要属性有：相对于某一个元素android:layout\_below、 android:layout\_toLeftOf相对于父元素的地方android:layout\_alignParentLeft、android:layout\_alignParentRigh;

5)TableLayout：表格布局，每一个TableLayout里面有表格行TableRow，TableRow里面可以具体定义每一个元素。每一个布局都有自己适合的方式，这五个布局元素可以相互嵌套应用，做出美观的界面。

1. android中的动画有哪几类，它们的特点和区别是什么

答：两种，一种是Tween动画、还有一种是Frame动画。Tween动画，这种实现方式可以使视图组件移动、放大、缩小以及产生透明度的变化;另一种Frame动画，传统的动画方法，通过顺序的播放排列好的图片来实现，类似电影。

1. activity的启动模式有哪些？是什么含义

答：在android里，有4种activity的启动模式，分别为：

“standard” (默认)

“singleTop”

“singleTask”

“singleInstance”

它们主要有如下不同：

1)如何决定所属task

“standard”和”singleTop”的activity的目标task，和收到的Intent的发送者在同一个task内，除非intent包括参数FLAG\_ACTIVITY\_NEW\_TASK。

如果提供了FLAG\_ACTIVITY\_NEW\_TASK参数，会启动到别的task里。

“singleTask”和”singleInstance”总是把activity作为一个task的根元素，他们不会被启动到一个其他task里。

2)是否允许多个实例

“standard”和”singleTop”可以被实例化多次，并且存在于不同的task中，且一个task可以包括一个activity的多个实例；

“singleTask”和”singleInstance”则限制只生成一个实例，并且是task的根元素。 singleTop要求如果创建intent的时候栈顶已经有要创建 的Activity的实例，则将intent发送给该实例，而不发送给新的实例。

3)是否允许其它activity存在于本task内

“singleInstance”独占一个task，其它activity不能存在那个task里；如果它启动了一个新的activity，不管新的activity的launch mode 如何，新的activity都将会到别的task里运行（如同加了FLAG\_ACTIVITY\_NEW\_TASK参数）。

而另外三种模式，则可以和其它activity共存。

4)是否每次都生成新实例

“standard”对于没一个启动Intent都会生成一个activity的新实例；

“singleTop”的activity如果在task的栈顶的话，则不生成新的该activity的实例，直接使用栈顶的实例，否则，生成该activity的实例。

比如现在task栈元素为A-B-C-D（D在栈顶），这时候给D发一个启动intent，如果D是 “standard”的，则生成D的一个新实例，栈变为A－B－C－D－D。

如果D是singleTop的话，则不会生产D的新实例，栈状态仍为A-B-C-D

如果这时候给B发Intent的话，不管B的launchmode是”standard” 还是 “singleTop” ，都会生成B的新实例，栈状态变为A-B-C-D-B。

“singleInstance”是其所在栈的唯一activity，它会每次都被重用。

“singleTask”如果在栈顶，则接受intent，否则，该intent会被丢弃，但是该task仍会回到前台。

当已经存在的activity实例处理新的intent时候，会调用onNewIntent()方法 如果收到intent生成一个activity实例，那么用户可以通过back键回到上一个状态；如果是已经存在的一个activity来处理这个intent的话，用户不能通过按back键返回到这之前的状态。

1. 跟activity和Task 有关的 Intent启动方式有哪些？其含义是

核心的Intent Flag有：

FLAG\_ACTIVITY\_NEW\_TASK

FLAG\_ACTIVITY\_CLEAR\_TOP

FLAG\_ACTIVITY\_RESET\_TASK\_IF\_NEEDED

FLAG\_ACTIVITY\_SINGLE\_TOP

FLAG\_ACTIVITY\_NEW\_TASK

如果设置，这个Activity会成为历史stack中一个新Task的开始。一个Task（从启动它的Activity到下一个Task中的 Activity）定义了用户可以迁移的Activity原子组。Task可以移动到前台和后台；在某个特定Task中的所有Activity总是保持相同的次序。

这个标志一般用于呈现“启动”类型的行为：它们提供用户一系列可以单独完成的事情，与启动它们的Activity完全无关。

使用这个标志，如果正在启动的Activity的Task已经在运行的话，那么，新的Activity将不会启动；代替的，当前Task会简单的移入前台。参考FLAG\_ACTIVITY\_MULTIPLE\_TASK标志，可以禁用这一行为。

这个标志不能用于调用方对已经启动的Activity请求结果。

FLAG\_ACTIVITY\_CLEAR\_TOP   
 如果设置，并且这个Activity已经在当前的Task中运行，因此，不再是重新启动一个这个Activity的实例，而是在这个Activity上方的所有Activity都将关闭，然后这个Intent会作为一个新的Intent投递到老的Activity（现在位于顶端）中。   
 例如，假设一个Task中包含这些Activity：A，B，C，D。如果D调用了startActivity()，并且包含一个指向Activity B的Intent，那么，C和D都将结束，然后B接收到这个Intent，因此，目前stack的状况是：A，B。   
 上例中正在运行的Activity B既可以在onNewIntent()中接收到这个新的Intent，也可以把自己关闭然后重新启动来接收这个Intent。如果它的启动模式声明为 “multiple”(默认值)，并且你没有在这个Intent中设置FLAG\_ACTIVITY\_SINGLE\_TOP标志，那么它将关闭然后重新创建；对于其它的启动模式，或者在这个Intent中设置FLAG\_ACTIVITY\_SINGLE\_TOP标志，都将把这个Intent投递到当前这个实例的onNewIntent()中。   
 这个启动模式还可以与FLAG\_ACTIVITY\_NEW\_TASK结合起来使用：用于启动一个Task中的根Activity，它会把那个Task中任何运行的实例带入前台，然后清除它直到根Activity。这非常有用，例如，当从Notification Manager处启动一个Activity。

FLAG\_ACTIVITY\_RESET\_TASK\_IF\_NEEDED

如果设置这个标志，这个activity不管是从一个新的栈启动还是从已有栈推到栈顶，它都将以the front door of the task的方式启动。这就讲导致任何与应用相关的栈都讲重置到正常状态（不管是正在讲activity移入还是移除），如果需要，或者直接重置该栈为初始状态。

FLAG\_ACTIVITY\_SINGLE\_TOP   
 如果设置，当这个Activity位于历史stack的顶端运行时，不再启动一个新的

FLAG\_ACTIVITY\_BROUGHT\_TO\_FRONT   
 这个标志一般不是由程序代码设置的，如在launchMode中设置singleTask模式时系统帮你设定。

FLAG\_ACTIVITY\_CLEAR\_WHEN\_TASK\_RESET   
 如果设置，这将在Task的Activity stack中设置一个还原点，当Task恢复时，需要清理Activity。也就是说，下一次Task带着 FLAG\_ACTIVITY\_RESET\_TASK\_IF\_NEEDED标记进入前台时（典型的操作是用户在主画面重启它），这个Activity和它之上的都将关闭，以至于用户不能再返回到它们，但是可以回到之前的Activity。   
 这在你的程序有分割点的时候很有用。例如，一个e-mail应用程序可能有一个操作是查看一个附件，需要启动图片浏览Activity来显示。这个 Activity应该作为e-mail应用程序Task的一部分，因为这是用户在这个Task中触发的操作。然而，当用户离开这个Task，然后从主画面选择e-mail app，我们可能希望回到查看的会话中，但不是查看图片附件，因为这让人困惑。通过在启动图片浏览时设定这个标志，浏览及其它启动的Activity在下次用户返回到mail程序时都将全部清除。

FLAG\_ACTIVITY\_EXCLUDE\_FROM\_RECENTS   
 如果设置，新的Activity不会在最近启动的Activity的列表中保存。

FLAG\_ACTIVITY\_FORWARD\_RESULT   
 如果设置，并且这个Intent用于从一个存在的Activity启动一个新的Activity，那么，这个作为答复目标的Activity将会传到这个新的Activity中。这种方式下，新的Activity可以调用setResult(int)，并且这个结果值将发送给那个作为答复目标的 Activity。

FLAG\_ACTIVITY\_LAUNCHED\_FROM\_HISTORY   
 这个标志一般不由应用程序代码设置，如果这个Activity是从历史记录里启动的（常按HOME键），那么，系统会帮你设定。

FLAG\_ACTIVITY\_MULTIPLE\_TASK   
 不要使用这个标志，除非你自己实现了应用程序启动器。与FLAG\_ACTIVITY\_NEW\_TASK结合起来使用，可以禁用把已存的Task送入前台的行为。当设置时，新的Task总是会启动来处理Intent，而不管这是是否已经有一个Task可以处理相同的事情。   
 由于默认的系统不包含图形Task管理功能，因此，你不应该使用这个标志，除非你提供给用户一种方式可以返回到已经启动的Task。   
 如果FLAG\_ACTIVITY\_NEW\_TASK标志没有设置，这个标志被忽略。

FLAG\_ACTIVITY\_NO\_ANIMATION   
 如果在Intent中设置，并传递给Context.startActivity()的话，这个标志将阻止系统进入下一个Activity时应用 Acitivity迁移动画。这并不意味着动画将永不运行——如果另一个Activity在启动显示之前，没有指定这个标志，那么，动画将被应用。这个标志可以很好的用于执行一连串的操作，而动画被看作是更高一级的事件的驱动。

FLAG\_ACTIVITY\_NO\_HISTORY   
 如果设置，新的Activity将不再历史stack中保留。用户一离开它，这个Activity就关闭了。这也可以通过设置noHistory特性。

FLAG\_ACTIVITY\_NO\_USER\_ACTION   
 如果设置，作为新启动的Activity进入前台时，这个标志将在Activity暂停之前阻止从最前方的Activity回调的onUserLeaveHint()。   
 典型的，一个Activity可以依赖这个回调指明显式的用户动作引起的Activity移出后台。这个回调在Activity的生命周期中标记一个合适的点，并关闭一些Notification。   
 如果一个Activity通过非用户驱动的事件，如来电或闹钟，启动的，这个标志也应该传递给Context.startActivity，保证暂停的Activity不认为用户已经知晓其Notification。

FLAG\_ACTIVITY\_PREVIOUS\_IS\_TOP   
 If set and this intent is being used to launch a new activity from an existing one, the current activity will not be counted as the top activity for deciding whether the new intent should be delivered to the top instead of starting a new one. The previous activity will be used as the top, with the assumption being that the current activity will finish itself immediately.

FLAG\_ACTIVITY\_REORDER\_TO\_FRONT   
 如果在Intent中设置，并传递给Context.startActivity()，这个标志将引发已经运行的Activity移动到历史stack的顶端。   
 例如，假设一个Task由四个Activity组成：A,B,C,D。如果D调用startActivity()来启动Activity B，那么，B会移动到历史stack的顶端，现在的次序变成A,C,D,B。如果FLAG\_ACTIVITY\_CLEAR\_TOP标志也设置的话，那么这个标志将被忽略。

1. 如何启用Service，如何停用Service。

服务的开发比较简单，如下：

第一步：继承Service类

public class SMSService extends Service {}

第二步：在AndroidManifest.xml文件中的<application>节点里对服务进行配置:<service android:name=".SMSService" />

服务不能自己运行，需要通过调用Context.startService()或Context.bindService()方法启动服务。这两个方法都可以启动Service，但是它们的使用场合有所不同。使用startService()方法启用服务，调用者与服务之间没有关连，即使调用者退出了，服务仍然运行。使用bindService()方法启用服务，调用者与服务绑定在了一起，调用者一旦退出，服务也就终止，大有“不求同时生，必须同时死”的特点。

如果打算采用Context.startService()方法启动服务，在服务未被创建时，系统会先调用服务的onCreate()方法，接着调用onStart()方法。如果调用startService()方法前服务已经被创建，多次调用startService()方法并不会导致多次创建服务，但会导致多次调用onStart()方法。采用startService()方法启动的服务，只能调用Context.stopService()方法结束服务，服务结束时会调用onDestroy()方法。

如果打算采用Context.bindService()方法启动服务，在服务未被创建时，系统会先调用服务的onCreate()方法，接着调用onBind()方法。这个时候调用者和服务绑定在一起，调用者退出了，系统就会先调用服务的onUnbind()方法，接着调用onDestroy()方法。如果调用bindService()方法前服务已经被绑定，多次调用bindService()方法并不会导致多次创建服务及绑定(也就是说onCreate()和onBind()方法并不会被多次调用)。如果调用者希望与正在绑定的服务解除绑定，可以调用unbindService()方法，调用该方法也会导致系统调用服务的onUnbind()-->onDestroy()方法。

服务常用生命周期回调方法如下：

onCreate() 该方法在服务被创建时调用，该方法只会被调用一次，无论调用多少次startService()或bindService()方法，服务也只被创建一次。

onDestroy()该方法在服务被终止时调用。

与采用Context.startService()方法启动服务有关的生命周期方法

onStart() 只有采用Context.startService()方法启动服务时才会回调该方法。该方法在服务开始运行时被调用。多次调用startService()方法尽管不会多次创建服务，但onStart() 方法会被多次调用。

与采用Context.bindService()方法启动服务有关的生命周期方法

onBind()只有采用Context.bindService()方法启动服务时才会回调该方法。该方法在调用者与服务绑定时被调用，当调用者与服务已经绑定，多次调用Context.bindService()方法并不会导致该方法被多次调用。

onUnbind()只有采用Context.bindService()方法启动服务时才会回调该方法。该方法在调用者与服务解除绑定时被调用

1. Handler

请解释下在单线程模型中Message、Handler、Message Queue、Looper之间的关系。

答：简单的说，Handler获取当前线程中的looper对象，looper用来从存放Message的MessageQueue中取出Message，再有Handler进行Message的分发和处理.

Message Queue(消息队列)：用来存放通过Handler发布的消息，通常附属于某一个创建它的线程，可以通过Looper.myQueue()得到当前线程的消息队列

Handler：可以发布或者处理一个消息或者操作一个Runnable，通过Handler发布消息，消息将只会发送到与它关联的消息队列，然也只能处理该消息队列中的消息

Looper：是Handler和消息队列之间通讯桥梁，程序组件首先通过Handler把消息传递给Looper，Looper把消息放入队列。Looper也把消息队列里的消息广播给所有的

Handler：Handler接受到消息后调用handleMessage进行处理

Message：消息的类型，在Handler类中的handleMessage方法中得到单个的消息进行处理

在单线程模型下，为了线程通信问题，Android设计了一个Message Queue(消息队列)， 线程间可以通过该Message Queue并结合Handler和Looper组件进行信息交换。下面将对它们进行分别介绍：

1） Message

Message消息，理解为线程间交流的信息，处理数据后台线程需要更新UI，则发送Message内含一些数据给UI线程。

2） Handler

Handler处理者，是Message的主要处理者，负责Message的发送，Message内容的执行处理。后台线程就是通过传进来的 Handler对象引用来sendMessage(Message)。而使用Handler，需要implement 该类的 handleMessage(Message)方法，它是处理这些Message的操作内容，例如Update UI。通常需要子类化Handler来实现handleMessage方法。

3） Message Queue

Message Queue消息队列，用来存放通过Handler发布的消息，按照先进先出执行。

每个message queue都会有一个对应的Handler。Handler会向message queue通过两种方法发送消息：sendMessage或post。这两种消息都会插在message queue队尾并按先进先出执行。但通过这两种方法发送的消息执行的方式略有不同：通过sendMessage发送的是一个message对象,会被 Handler的handleMessage()函数处理；而通过post方法发送的是一个runnable对象，则会自己执行。

4） Looper

Looper是每条线程里的Message Queue的管家。Android没有Global的Message Queue，而Android会自动替主线程(UI线程)建立Message Queue，但在子线程里并没有建立Message Queue。所以调用Looper.getMainLooper()得到的主线程的Looper不为NULL，但调用Looper.myLooper() 得到当前线程的Looper就有可能为NULL。对于子线程使用Looper，API Doc提供了正确的使用方法：这个Message机制的大概流程：

①在Looper.loop()方法运行开始后，循环地按照接收顺序取出Message Queue里面的非NULL的Message。

② 一开始Message Queue里面的Message都是NULL的。当Handler.sendMessage(Message)到Message Queue，该函数里面设置了那个Message对象的target属性是当前的Handler对象。随后Looper取出了那个Message，则调用 该Message的target指向的Hander的dispatchMessage函数对Message进行处理。在dispatchMessage方法里，如何处理Message则由用户指定，三个判断，优先级从高到低：

1. Message里面的Callback，一个实现了Runnable接口的对象，其中run函数做处理工作；
2. Handler里面的mCallback指向的一个实现了Callback接口的对象，由其handleMessage进行处理；
3. 处理消息Handler对象对应的类继承并实现了其中handleMessage函数，通过这个实现的handleMessage函数处理消息。

由此可见，我们实现的handleMessage方法是优先级最低的！

③ Handler处理完该Message (update UI) 后，Looper则设置该Message为NULL，以便回收！

在网上有很多文章讲述主线程和其他子线程如何交互，传送信息，最终谁来执行处理信息之类的，个人理解是最简单的方法——判断Handler对象里面的Looper对象是属于哪条线程的，则由该线程来执行！

①当Handler对象的构造函数的参数为空，则为当前所在线程的Looper；

②Looper.getMainLooper()得到的是主线程的Looper对象，Looper.myLooper()得到的是当前线程的Looper对象。

1. 说明handler机制的原理

一个Handler允许你发送和处理Message和Runable对象，每个线程都有自己的Looper，每个Looper中封装着MessageQueue。Looper负责不断的从自己的消息队列里取出队头的任务或消息执行。每个handler也和线程关联，Handler负责把Message和Runable

对象传递给MessageQueue（用到post，sendMessage等方法），而且在这些对象离开MessageQueue时，Handler负责执行他们（用到handleMessage方法）。

其中Message类就是定义了一个信息，这个信息中包含一个描述符和任意的数据对象，这个信息被用来传递给Handler.Message对象

1. 什么情况会导致Force Close ？如何避免？能否捕获导致其的异常

答：程序出现异常，比如nullpointer。

避免：编写程序时逻辑连贯，思维缜密。能捕获异常，在logcat中能看到异常信息

1. 请介绍下ContentProvider是如何实现数据共享的。

一个程序可以通过实现一个Content provider的抽象接口将自己的数据完全暴露出去，而且Content providers是以类似数据库中表的方式将数据暴露。Content providers存储和检索数据，通过它可以让所有的应用程序访问到，这也是应用程序之间唯一共享数据的方法。

要想使应用程序的数据公开化，可通过2种方法：创建一个属于你自己的Content provider或者将你的数据添加到一个已经存在的Content provider中，前提是有相同数据类型并且有写入Content provider的权限。

如何通过一套标准及统一的接口获取其他应用程序暴露的数据

Android提供了ContentResolver，外界的程序可以通过ContentResolver接口访问ContentProvider提供的数据。

1. Service和Thread的区别

答：servie是系统的组件，它由系统进程托管（servicemanager）；它们之间的通信类似于client和server，是一种轻量级的ipc通信，这种通信的载体是binder，它是在linux层交换信息的一种ipc。而thread是由本应用程序托管。1). Thread：Thread 是程序执行的最小单元，它是分配CPU的基本单位。可以用 Thread 来执行一些异步的操作。

2). Service：Service 是android的一种机制，当它运行的时候如果是Local Service，那么对应的 Service 是运行在主进程的 main 线程上的。如：onCreate，onStart 这些函数在被系统调用的时候都是在主进程的 main 线程上运行的。如果是Remote Service，那么对应的 Service 则是运行在独立进程的 main 线程上。

既然这样，那么我们为什么要用 Service 呢？其实这跟 android 的系统机制有关，我们先拿 Thread 来说。Thread 的运行是独立于 Activity 的，也就是说当一个 Activity 被 finish 之后，如果你没有主动停止 Thread 或者 Thread 里的 run 方法没有执行完毕的话，Thread 也会一直执行。因此这里会出现一个问题：当 Activity 被 finish 之后，你不再持有该 Thread 的引用。另一方面，你没有办法在不同的 Activity 中对同一 Thread 进行控制。

举个例子：如果你的 Thread 需要不停地隔一段时间就要连接服务器做某种同步的话，该 Thread 需要在 Activity 没有start的时候也在运行。这个时候当你 start 一个 Activity 就没有办法在该 Activity 里面控制之前创建的 Thread。因此你便需要创建并启动一个 Service ，在 Service 里面创建、运行并控制该 Thread，这样便解决了该问题（因为任何 Activity 都可以控制同一 Service，而系统也只会创建一个对应 Service 的实例）。

因此你可以把 Service 想象成一种消息服务，而你可以在任何有 Context 的地方调用 Context.startService、Context.stopService、Context.bindService，Context.unbindService，来控制它，你也可以在 Service 里注册 BroadcastReceiver，在其他地方通过发送 broadcast 来控制它，当然这些都是 Thread 做不到的。

1. Android异常

Android本身的api并未声明会抛出异常，则其在运行时有无可能抛出runtime异常，你遇到过吗？诺有的话会导致什么问题？如何解决？

答：会，比如nullpointerException。我遇到过，比如textview.setText()时，textview没有初始化。会导致程序无法正常运行出现forceclose。打开控制台查看logcat信息找出异常信息并修改程序。

1. IntentService有何优点

答：Acitivity的进程，当处理Intent的时候，会产生一个对应的Service； Android的进程处理器现在会尽可能的不kill掉你；非常容易使用

1. Activity

如果后台的Activity由于某原因被系统回收了，如何在被系统回收之前保存当前状态？

答：重写onSaveInstanceState()方法，在此方法中保存需要保存的数据，该方法将会在activity被回收之前调用。通过重写onRestoreInstanceState()方法可以从中提取保存好的数据

1. 如何将一个Activity设置成窗口的样式

答：<activity>中配置：android :theme="@android:style/Theme.Dialog"

另外android:theme="@android:style/Theme.Translucent" 是设置透明

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

答：对于单一Activity的应用来说，退出很简单，直接finish()即可。当然，也可以用killProcess()和System.exit()这样的方法。

对于多个activity:

1)记录打开的Activity：每打开一个Activity，就记录下来。在需要退出时，关闭每一个Activity即可

2)发送特定广播：在需要结束应用时，发送一个特定的广播，每个Activity收到广播后，关闭即可。

3)递归退出：在打开新的Activity时使用startActivityForResult，然后自己加标志，在onActivityResult中处理，递归关闭。为了编程方便，最好定义一个Activity基类，处理这些共通问题。

在2.1之前，可以使用ActivityManager的restartPackage方法。

它可以直接结束整个应用。在使用时需要权限android.permission.RESTART\_PACKAGES。

注意不要被它的名字迷惑。

可是，在2.2，这个方法失效了。在2.2添加了一个新的方法，killBackground Processes()，需要权限 android.permission.KILL\_BACKGROUND\_PROCESSES。可惜的是，它和2.2的restartPackage一样，根本起不到应有的效果。

另外还有一个方法，就是系统自带的应用程序管理里，强制结束程序的方法，forceStopPackage()。它需要权限android.permission.FORCE\_STOP\_PACKAGES。并且需要添加android:sharedUserId="android.uid.system"属性。同样可惜的是，该方法是非公开的，他只能运行在系统进程，第三方程序无法调用。

因为需要在Android.mk中添加LOCAL\_CERTIFICATE := platform。

而Android.mk是用于在Android源码下编译程序用的。

从以上可以看出，在2.2，没有办法直接结束一个应用，而只能用自己的办法间接办到。

现提供几个方法，供参考：

1)抛异常强制退出：

该方法通过抛异常，使程序Force Close。

验证可以，但是，需要解决的问题是，如何使程序结束掉，而不弹出Force Close的窗口。

2)记录打开的Activity：

每打开一个Activity，就记录下来。在需要退出时，关闭每一个Activity即可。

3)发送特定广播：

在需要结束应用时，发送一个特定的广播，每个Activity收到广播后，关闭即可。

4)递归退出

在打开新的Activity时使用startActivityForResult，然后自己加标志，在onActivityResult中处理，递归关闭。

除了第一个，都是想办法把每一个Activity都结束掉，间接达到目的。但是这样做同样不完美。你会发现，如果自己的应用程序对每一个Activity都设置了nosensor，在两个Activity结束的间隙，sensor可能有效了。但至少，我们的目的达到了，而且没有影响用户使用。为了编程方便，最好定义一个Activity基类，处理这些共通问题。

1. AIDL的全称是什么？如何工作？能处理哪些类型的数据

答：全称是：Android Interface Define Language

在Android中, 每个应用程序都可以有自己的进程. 在写UI应用的时候, 经常要用到Service. 在不同的进程中, 怎样传递对象呢?显然, Java中不允许跨进程内存共享. 因此传递对象, 只能把对象拆分成操作系统能理解的简单形式, 以达到跨界对象访问的目的. 在J2EE中,采用RMI的方式, 可以通过序列化传递对象. 在Android中, 则采用AIDL的方式. 理论上AIDL可以传递Bundle,实际上做起来却比较麻烦。

AIDL(AndRoid接口描述语言)是一种借口描述语言; 编译器可以通过aidl文件生成一段代码，通过预先定义的接口达到两个进程内部通信进程的目的. 如果需要在一个Activity中, 访问另一个Service中的某个对象, 需要先将对象转化成AIDL可识别的参数(可能是多个参数), 然后使用AIDL来传递这些参数, 在消息的接收端, 使用这些参数组装成自己需要的对象.

AIDL的IPC的机制和COM或CORBA类似, 是基于接口的，但它是轻量级的。它使用代理类在客户端和实现层间传递值. 如果要使用AIDL, 需要完成2件事情: 1. 引入AIDL的相关类.; 2. 调用aidl产生的class.

AIDL的创建方法:

AIDL语法很简单,可以用来声明一个带一个或多个方法的接口，也可以传递参数和返回值。 由于远程调用的需要, 这些参数和返回值并不是任何类型.下面是些AIDL支持的数据类型:

1)不需要import声明的简单Java编程语言类型(int,boolean等)

2) String, CharSequence不需要特殊声明

3) List, Map和Parcelables类型, 这些类型内所包含的数据成员也只能是简单数据类型, String等其他比支持的类型.

(另外: 我没尝试Parcelables, 在Eclipse+ADT下编译不过, 或许以后会有所支持)

1. 请解释下Android程序运行时权限与文件系统权限的区别

答：运行时权限Dalvik( android授权)

文件系统 linux 内核授权

1. 系统上安装了多种浏览器，能否指定某浏览器访问指定页面？请说明原由

通过直接发送Uri把参数带过去，或者通过manifest里的intentfilter里的data属性

1. android系统的优势和不足

答：Android平台手机 5大优势：

⑴开放性

在优势方面，Android平台首先就是其开发性，开发的平台允许任何移动终端厂商加入到Android联盟中来。显著的开放性可以使其拥有更多的开发者，随着用户和应用的日益丰富，一个崭新的平台也将很快走向成熟。开放性对于Android的发展而言，有利于积累人气，这里的人气包括消费者和厂商，而对于消费者来讲，随大的受益正是丰富的软件资源。开放的平台也会带来更大竞争，如此一来，消费者将可以用更低的价位购得心仪的手机。

⑵挣脱运营商的束缚

在过去很长的一段时间，特别是在欧美地区，手机应用往往受到运营商制约，使用什么功能接入什么网络，几乎都受到运营商的控制。从去年iPhone 上市 ，用户可以更加方便地连接网络，运营商的制约减少。随着EDGE、HSDPA这些2G至3G移动网络的逐步过渡和提升，手机随意接入网络已不是运营商口中的笑谈，当你可以通过手机IM软件方便地进行即时聊天时，再回想不久前天价的彩信和图铃下载业务，是不是像噩梦一样？互联网巨头Google推动的Android终端天生就有网络特色，将让用户离互联网更近。

⑶丰富的硬件选择

这一点还是与Android平台的开放性相关，由于Android的开放性，众多的厂商会推出千奇百怪，功能特色各具的多种产品。功能上的差异和特色，却不会影响到数据同步、甚至软件的兼容，好比你从诺基亚 Symbian风格手机 一下改用苹果 iPhone ，同时还可将Symbian中优秀的软件带到iPhone上使用、联系人等资料更是可以方便地转移，是不是非常方便呢？

⑷不受任何限制的开发商

Android平台提供给第三方开发商一个十分宽泛、自由的环境，不会受到各种条条框框的阻扰，可想而知，会有多少新颖别致的软件会诞生。但也有其两面性，血腥、暴力、情色方面的程序和游戏如可控制正是留给Android难题之一。

⑸无缝结合的Google应用

如今叱诧互联网的Google已经走过10年度历史，从搜索巨人到全面的互联网渗透，Google服务如地图、邮件、搜索等已经成为连接用户和互联网的重要纽带，而Android平台手机将无缝结合这些优秀的Google服务。

再说Android的5大不足：

⑴安全和隐私

由于手机 与互联网的紧密联系，个人隐私很难得到保守。除了上网过程中经意或不经意留下的个人足迹，Google这个巨人也时时站在你的身后，洞穿一切，因此，互联网的深入将会带来新一轮的隐私危机。

⑵首先开卖Android手机的不是最大运营商

众所周知，T-Mobile在23日，于美国纽约发布 了Android首款手机G1。但是在北美市场，最大的两家运营商乃AT&T和Verizon，而目前所知取得Android手机销售权的仅有 T-Mobile和Sprint，其中T-Mobile的3G网络相对于其他三家也要逊色不少，因此，用户可以买账购买G1，能否体验到最佳的3G网络服务则要另当别论了！

⑶运营商仍然能够影响到Android手机

在国内市场，不少用户对购得移动定制机不满，感觉所购的手机被人涂画了广告一般。这样的情况在国外市场同样出现。Android手机的另一发售运营商Sprint就将在其机型中内置其手机商店程序。

⑷同类机型用户减少

在不少手机论坛都会有针对某一型号的子论坛，对一款手机的使用心得交流，并分享软件资源。而对于Android平台手机，由于厂商丰富，产品类型多样，这样使用同一款机型的用户越来越少，缺少统一机型的程序强化。举个稍显不当的例子，现在山寨机泛滥，品种各异，就很少有专门针对某个型号山寨机的讨论和群组，除了哪些功能异常抢眼、颇受追捧的机型以外。

⑸过分依赖开发商缺少标准配置

在使用PC端的Windows Xp系统的时候，都会内置微软Windows Media Player这样一个浏览器程序，用户可以选择更多样的播放器，如Realplay或暴风影音等。但入手开始使用默认的程序同样可以应付多样的需要。在 Android平台中，由于其开放性，软件更多依赖第三方厂商，比如Android系统的SDK中就没有内置音乐 播放器，全部依赖第三方开发，缺少了产品的统一性。

1. Android dvm的进程和Linux的进程, 应用程序的进程是否为同一个概念

答：DVM指dalivk的虚拟机。每一个Android应用程序都在它自己的进程中运行，都拥有一个独立的Dalvik虚拟机实例。而每一个DVM都是在Linux 中的一个进程，所以说可以认为是同一个概念。

1. 嵌入式操作系统内存管理有哪几种， 各有何特性

　　页式，段式，段页，用到了MMU,虚拟空间等技术

1. 什么是嵌入式实时操作系统, Android 操作系统属于实时操作系统吗

嵌入式实时操作系统是指当外界事件或数据产生时，能够接受并以足够快的速度予以处理，其处理的结果又能在规定的时间之内来控制生产过程或对处理系统作出快速响应，并控制所有实时任务协调一致运行的嵌入式操作系统。主要用于工业控制、 军事设备、 航空航天等领域对系统的响应时间有苛刻的要求，这就需要使用实时系统。又可分为软实时和硬实时两种，而android是基于linux内核的，因此属于软实时。

1. 一条最长的短信息约占多少byte

　　中文70(包括标点)，英文160，160个字节。

1. 用算法调整图片的透明度

有一个一维整型数组int[]data保存的是一张宽为width，高为height的图片像素值信息。请写一个算法，将该图片所有的白色不透明(0xffffffff)像素点的透明度调整为50%。

1. 如何将SQLite数据库(dictionary.db文件)与apk文件一起发布

解答：可以将dictionary.db文件复制到Eclipse Android工程中的res aw目录中。所有在res aw目录中的文件不会被压缩，这样可以直接提取该目录中的文件。可以将dictionary.db文件复制到res aw目录中

1. sqlite的用处及用法, 哪里用过
2. sqlite，修改，添加字段，版本更新，排序，默认排序
3. 如何建一个“aaa.db”数据库，如何建一个user表；写出增删改查语句
4. SQLite线程安全吗？sqlite3\_threadsafe()来确实是否线程安全版本

答案：SQLite是线程安全的。

由于很多用户会忽略我们在上一段中给出的建议，我们做出了这种让步。但是，为了达到线程安全，SQLite在编译时必须将SQLITE\_THREADSAFE预处理宏置为1。在Windows和Linux上，已编译的好的二进制发行版中都是这样设置的。如果不确定你所使用的库是否是线程安全的，可以调用sqlite3\_threadsafe()接口找出。

1. 怎么提高sqlite的效率

事物

1. 如何将打开res aw目录中的数据库文件

解答：在Android中不能直接打开res aw目录中的数据库文件，而需要在程序第一次启动时将该文件复制到手机内存或SD卡的某个目录中，然后再打开该数据库文件。

复制的基本方法是使用getResources().openRawResource方法获得res aw目录中资源的 InputStream对象，然后将该InputStream对象中的数据写入其他的目录中相应文件中。在Android SDK中可以使用SQLiteDatabase.openOrCreateDatabase方法来打开任意目录中的SQLite数据库文件。

1. DDMS和TraceView的区别

DDMS是一个程序执行查看器，在里面可以看见线程和堆栈等信息，TraceView是程序性能分析器 。

1. java中如何引用本地语言

可以用JNI（java native interface java 本地接口）接口 。

1. ListView切换奇偶行背景色

ListView中有时候又这样的需求，当选中某项listView的时候，需求是改变listiew的背景颜色，如果用resource文件会有缓存问题，背景显示错误，那么直接用16进制表示颜色即可。如0xFF7AACC6. 注意表示透明度的需要写上，否则显示不正确

ListView源码中两个设置背景方法的不同点：

1. @RemotableViewMethod
2. public void setBackgroundResource(int resid) {
3. if (resid != 0 && resid == mBackgroundResource) {
4. return;
5. }
6. Drawable d= null;
7. if (resid != 0) {
8. d = mResources.getDrawable(resid);
9. }
10. setBackground(d);
11. mBackgroundResource = resid;
12. }
13. 上面使用资源文件设置背景颜色
14. @RemotableViewMethod
15. public void setBackgroundColor(int color) {
16. if (mBackground instanceof ColorDrawable) {
17. ((ColorDrawable) mBackground).setColor(color);
18. } else {
19. setBackground(new ColorDrawable(color));
20. }
21. }
22. 上面是使用16进制数设置背景颜色，
23. getView()方法中判断position的奇偶性, 然后给ConvertView设置背景色
24. EditText单行显示
    1. 有属性 android:singleLine

多个输入框,点击Enter,焦点切换到下一个: EditText有属性android:imeOptions

输入法输入文本都有下划线

* 1. 自定义组件,实现onDraw方法

1. oom避免以及产生方式，遇到过几种OOM

图片和数据库，不要过多加载图片，记得回收

五大存储哪个容易报OOM，如何处理

oom情况遇到过么，怎么处理的

图片压缩后会出现什么问题？图片混乱

1. throw与throws区别

throws是用来声明一个方法可能抛出的所有异常信息  
throw则是指抛出的一个具体的异常类型。  
通常在一个方法（类）的声明处通过throws声明方法（类）可能抛出的异常信息，而在方法（类）内部通过throw声明一个具体的异常信息。  
throws通常不用显示的捕获异常，可由系统自动将所有捕获的异常信息抛给上级方法；  
throw则需要用户自己捕获相关的异常，而后在对其进行相关包装，最后在将包装后的异常信息抛

1. LOST.DIR

LOST.DIR是专门收集Android系统运行时意外丢失的文件而设置的文件夹，其收集的内容包括系统因为意外而没能保存的各类内存、交换、暂存等数据，软件的缓存数据、以及其他各类文件。只要系统发生意外（如软件突然无法正常运行），便会将发生意外的文件移到此目录下，以数字序列名称代替该文件的扩展名。注意：由于LOST.DIR文件夹的特殊作用，里面的文件都是“意外而生”，因此可以删除里面的文件，但请不要删除LOST.DIR文件夹。

1. 设置界面怎么做的
2. ui效果图介绍布局
3. login模块怎么做的
4. 点击登录按钮的时候，会从server端获得什么？传送什么
5. android客户端如何实现自动登录
6. 版本更新，从server端获得什么，通过什么进行更新
7. Fragment和Activity之间的联系
8. Fragment如何实现Activity跳转,你怎么处理Fragment之间的信息交互
9. fragment在哪个方法里实现代码
10. res和assets资源文件的作用分别是什么，怎么读取
11. 了解pad开发吗
12. 常用的适配有哪些，如何进行优化
13. 应用换肤有什么方案可以实现
14. 异步任务中允许有多个线程同时进行吗
15. 图片是怎么获取的
16. 文件的下载
17. 图片的下载
18. 缓存区的图片如何适当的释放
19. 图片缓存到哪里
20. 那你的图片是如何请求的
21. 图文混排怎么实现
22. 相册缩略图是怎么实现的
23. 下拉刷新的实现思路，反弹的效果是怎么实现的
24. 自定义的View
25. 自定义控件用过哪些？对自定义控件是怎么看的
26. 瀑布流的实现思路
27. 聊天窗口的实现
28. android的dvm和Linux进程是一样的吗
29. 监听模式
30. 如何刷新view
31. Listview

如果一个屏幕可以显示20条数据，但服务器只有4条数据，他创建了几个item

1. viewPager左右切屏

viewPager左右切屏的时候和listview上下拖动会不会发生冲突，如果会是怎么解决的

ViewPager有几个子View

1. listview加载图片错位，什么原因导致的
2. ListView中条目中显示webView会怎么样
3. 用listview进行快速翻页，如何解决内存溢出
4. listview快速滑动时，加载几条数据
5. 实现listview分页

页面下端有左右两个button(分别是上页,下页);

要求：实现两页分页效果就ok，在第一页时，button上页不能点击，

在第二页时，button下页不能点击；

1. 动画分为哪几种，各自的特点和区别
2. 写出Socket服务器与客户端交互的代码和实现思路
3. 谈谈对Http的理解

它的优点，缺点都有哪些；访问网络的方式，请求头，服务器是如何响应的；

1. 谈谈递归
2. 写出Activity里的方法都是在什么情况下执行
3. dp与sp的区别

dp也就是dip。这个和sp基本类似。如果设置表示长度、高度等属性时可以使用dp 或sp。但如果设置字体，需要使用sp。dp是与密度无关，sp除了与密度无关外，还与scale无关。如果屏幕密度为160，这时dp和sp和px是一样的。1dp=1sp=1px，但如果使用px作单位，如果屏幕大小不变（假设还是3.2寸），而屏幕密度变成了320。那么原来TextView的宽度设成160px，在密度为320的3.2寸屏幕里看要比在密度为160的3.2寸屏幕上看短了一半。但如果设置成160dp或160sp的话。系统会自动将width属性值设置成320px的。也就是160 \* 320 / 160。其中320 / 160可称为密度比例因子。也就是说，如果使用dp和sp，系统会根据屏幕密度的变化自动进行转换。

1. java有那些排序，冒泡代码

for(inti=0;i<num.length;i++){

//内循环控制比较后移位

for(intj=num.length-1;j>i;j--){

if(num[j-1]>num[j]){

temp=num[j-1];

num[j-1]=num[j];

num[j]=temp;

}

}

1. Java线程实现方式和区别

继承Thread，实现runable接口。

区别：避免点继承的局限，一个类可以继承多个接口。

?适合于资源的共享

详细出处参考：http://www.jb51.net/article/37465.htm

1. Android中布局、区别、应用场景

FrameLayout，LinearLayout，TableLayout，RelativeLayout，AbsoluteLayout

1. 线程间通讯

Handler

1. 进程间通讯

Intent 等

1. 判断网络
2. 网络代理
3. HTTP的post和get方法的区别
4. 怎么判断服务端返回的数据类型
5. 栈，队列代码
6. 服务service
7. 多线程下载，离线下载
8. 支付方式
9. 如果想改变分辨率但不动原来布局，实现思路是什么

使用dimen进行更改数据，进行配置

1. java常用的处理异常的机制，原理及应用
2. 一段字符串“abdecffr”,请输出只出现一次的字母
3. 输入一个数，求其二进数中有多少个1
4. 有一个pizza生产线

有一个pizza生产线：开了两家pizza店，其中一家的pizza为奶酪口味，另一家三文鱼口味，现在又要添一家店，是黑椒香肠味，请用简单的类图描述以上情形，满足不同地区，不同店的pizza供应，保证好的扩展性和低耦合性，添第三家时UML何如修改？

1. 推送了解吗？推送底层代码掌握吗
2. asynctack原理
3. NDK具体说一下
4. Service结合代码说一下，原理
5. 当内存不足时，如何对尚未关闭的应用程序进行回收
6. service的运行级别
7. service如何一直保持在后台运行
8. service的启动方式
9. 软引用怎么用？为什么要用软引用

http://www.cnblogs.com/blogoflee/archive/2012/03/22/2411124.html

1)强引用  
平时我们编程的时候例如：Object object=new Object（）；那object就是一个强引用了。如果一个对象具有强引用，那就类似于必不可少的生活用品，垃圾回收器绝不会回收它。当内存空 间不足，Java虚拟机宁愿抛出OutOfMemoryError错误，使程序异常终止，也不会靠随意回收具有强引用的对象来解决内存不足问题。

2)软引用（SoftReference）  
如果一个对象只具有软引用，那就类似于可有可物的生活用品。如果内存空间足够，垃圾回收器就不会回收它，如果内存空间不足了，就会回收这些对象的内存。只 要垃圾回收器没有回收它，该对象就可以被程序使用。软引用可用来实现内存敏感的高速缓存。 软引用可以和一个引用队列（ReferenceQueue）联 合使用，如果软引用所引用的对象被垃圾回收，Java虚拟机就会把这个软引用加入到与之关联的引用队列中。  
3)弱引用（WeakReference）

如果一个对象只具有弱引用，那就类似于可有可物的生活用品。弱引用与软引用的区别在于：只具有弱引用的对象拥有更短暂的生命周期。在垃圾回收器线程扫描它 所管辖的内存区域的过程中，一旦发现了只具有弱引用的对象，不管当前内存空间足够与否，都会回收它的内存。不过，由于垃圾回收器是一个优先级很低的线程， 因此不一定会很快发现那些只具有弱引用的对象。 弱引用可以和一个引用队列（ReferenceQueue）联合使用，如果弱引用所引用的对象被垃圾回 收，Java虚拟机就会把这个弱引用加入到与之关联的引用队列中。

4)虚引用（PhantomReference）

"虚引用"顾名思义，就是形同虚设，与其他几种引用都不同，虚引用并不会决定对象的生命周期。如果一个对象仅持有虚引用，那么它就和没有任何引用一样，在 任何时候都可能被垃圾回收。 虚引用主要用来跟踪对象被垃圾回收的活动。虚引用与软引用和弱引用的一个区别在于：虚引用必须和引用队列 （ReferenceQueue）联合使用。当垃圾回收器准备回收一个对象时，如果发现它还有虚引用，就会在回收对象的内存之前，把这个虚引用加入到与之 关联的引用队列中。程序可以通过判断引用队列中是否已经加入了虚引用，来了解被引用的对象是否将要被垃圾回收。程序如果发现某个虚引用已经被加入到引用队 列，那么就可以在所引用的对象的内存被回收之前采取必要的行动。

1. 还有一个Service是建立在哪个线程上面
2. 接口都有哪些内容
3. 接口是否可继承接口

抽象类是否可实现(implements)接口? 抽象类是否可继承具体类(concrete class)? 抽象类中是否可以有静态的main方法

接口可以继承接口。抽象类可以实现(implements)接口，抽象类是否可继承具体类。抽象类中可以有静态的main方法。

备注：只要明白了接口和抽象类的本质和作用，这些问题都很好回答，你想想，如果你是java语言的设计者，你是否会提供这样的支持，如果不提供的话，有什么理由吗？如果你没有道理不提供，那答案就是肯定的了。

只有记住抽象类与普通类的唯一区别就是不能创建实例对象和允许有abstract方法。

1. 如何通过程序进行对服务器的接口调用，写出具体过程
2. 分页加载思路
3. 分页加载需要传递哪些参数
4. 讯飞语音在什么地方用到
5. 消息推送
6. LBS定位用到哪个地图？为什么要使用这个？实现了哪些功能
7. 地图中怎样获得附近的信息
8. 百度地图显示周边招聘信息
9. 移动互联使用了哪种方式
10. 集合方面的底层理解
11. 每个item都是不同的内容，怎么弄
12. 有没有自己写过NDK
13. 一个版本的app不更新，只是给他赠加或着减少一个功能，怎么弄
14. 架构的情况
15. 多线程的讲解
16. 线程的异常怎么处理
17. 请求网络用什么，http的使用
18. 解释intent和activity的区别
19. 如何处理内存溢出
20. 如何去除一个字符串首尾的空格
21. 如何把一个字符串转换成asll码
22. 得到一个数组中出现次数最多的数
23. 手机QQ聊天框冒泡效果怎么做

怎么使聊天时对方发的信息在左边，自己发的在右边

1. groupby是干什么的
2. 你android应用假如你手机设置了其他语言你的应用语言会变吗
3. 了解后台吗，对后台的认识
4. '学java'内存中占多少个字节

10个字节，12

1. Intent可以传递那些数据

基本数据类型，以及序列化的对象。

Serializable  、charsequence  、Parcelable  、Bundle

1. DDMS与TraceView的区别

DDMS是一个程序执行查看器，在里面可以看见线程和堆栈等信息，TraceView是程序性能分析器。

1. 横竖屏切换不设置属性会怎么样

重新执行Activity生命周期

1. 单例类在android中哪种更合适

例如工具类可以封装为单例，用户登录后保存用户信息也可以封装为单例。

1. 一串字符串编写程序翻转split（）
2. json写实体类
3. 如果标题过长超出界面，请写出你的解决方法

跑马灯的效果和HorizontalScrollView

1. 开发android应用怎样减少耗电量

建立连接前按Wifi-3G-2G顺序检查网络不要保持2G/3G连接wifi可用时，尽量用push，不要用定时查询优化应用程序的设计和算法使用简单高效的数据使用gzip压缩数据流>树-节点层次结构Ex,protobuf>json>xml

1. 提供额外的两个int域和一个Object域
2. android开发过程有几种缓存方式，并说明缺点
3. Activity相关

ActivityA点击启动ActivityB，b里有一个geradviewgeradview里有一个textview倒计时，到0的时候关闭两个antivity

1. Bitemap内存溢出解决方法
2. 推送
3. 文件传输
4. 缓存app缓存
5. 边距测试
6. Java数据结构
7. 常用的分辨率
8. Ui一般有几套，是什么格式的，.9格式的还是什么格式的
9. onCreateView和onViewCreated之间的区别
10. 继承与组合的关系
11. IPC机制具体解释
12. aidl具体解释
13. 为什么要有NDK
14. 为什么json与js交互方便
15. 进程与线程的区别
16. 当一个activity退出之后在进程中会不会杀死
17. jvm的内存分配
18. MVC模式的理解
19. 排序有了解过吗？及其实现复杂度
20. android手机的密度，你如何对屏幕密度进行自适应
21. handler的理解
22. 对于instancevariable和localvarial,JVM在少女情怀内存时有什么区别

实例变量与本地变量在内存分配上的区别:实例变量在堆内存中,本地变量在栈中

1. Java线程都有哪些状态？列举你所知道的线程同步方法

新建状态(New)

就绪状态(Runnable)

运行状态(Running)

阻塞状态(Blocked)

死亡状态(Dead)

1. 如何避免比不加载图片混乱的问题
2. AysnTask最多可以开启几个线程
3. 数据库如何升级（添加字段什么的）
4. WebService
5. IntentService的用法

Android开发中IntentService的使用

分类：android笔记2011-11-0910:03960人阅读评论(0)收藏举报

androidservicethreadstringclassnullIntentService实现原理及内部代码

Android中除了Service还有一个IntentService，他们之间到底有哪些区别呢?在继承关系上而言IntentService是Service的子类，内部实现的代码中涉及到一些Android入门开发者不了解的Looper，Android123在早期的文章中已经说明他们的用法，这里不再赘述，有关原理大家可以看源码实现如下:

publicabstractclassIntentServiceextendsService{

privatevolatileLoopermServiceLooper;

privatevolatileServiceHandlermServiceHandler;//一个Handler封装了Looper对象

privateStringmName;

privatebooleanmRedelivery;

privatefinalclassServiceHandlerextendsHandler{

publicServiceHandler(Looperlooper){

super(looper);

}

@Override

publicvoidhandleMessage(Messagemsg){

onHandleIntent((Intent)msg.obj);

stopSelf(msg.arg1);

}

}

publicIntentService(Stringname){//构造方法，需要提供一个name作为标识

super();

mName=name;

}

对于下面的setIntentRedelivery的参数，如果为真时

onStartCommand(Intent,int,int)}willreturn

Service#START\_REDELIVER\_INTENT},soifthisprocessdiesbefore

onHandleIntent(Intent)}returns,theprocesswillberestarted

如果为假时

onStartCommand(Intent,int,int)}willreturn

Service#START\_NOT\_STICKY},andiftheprocessdies

publicvoidsetIntentRedelivery(booleanenabled){

mRedelivery=enabled;

}

@Override

publicvoidonCreate(){//这里重写父类Service的创建，主要是构造一个线程

super.onCreate();

HandlerThreadthread=newHandlerThread("IntentService["+mName+"]");

thread.start();

mServiceLooper=thread.getLooper();

mServiceHandler=newServiceHandler(mServiceLooper);

}

@Override

publicvoidonStart(Intentintent,intstartId){//Android2.0以前的Service启动参数控制

Messagemsg=mServiceHandler.obtainMessage();

msg.arg1=startId;

msg.obj=intent;

mServiceHandler.sendMessage(msg);

}

@Override

publicintonStartCommand(Intentintent,intflags,intstartId){//Android2.0以后的服务启动参数

onStart(intent,startId);

returnmRedelivery?START\_REDELIVER\_INTENT:START\_NOT\_STICKY;

}

@Override

publicvoidonDestroy(){//服务摧毁的时候Looper一定要释放掉，这点很重要。

mServiceLooper.quit();

}

@Override

publicIBinderonBind(Intentintent){

returnnull;

}

protectedabstractvoidonHandleIntent(Intentintent);

}

从上面的代码相信大家可以看出IntentService和Service的不同了，通过Looper和Thread来解决标准Service中处理逻辑的阻塞问题，毕竟Android的Service也是会阻塞的。

关于使用

IntentService是Service类的子类，用来处理异步请求。客户端可以通过startService(Intent)方法传递请求给IntentService，IntentService通过workerthread处理每个Intent对象，执行完所有的工作之后自动停止Service。

说明：workerthread处理所有通过传递过来的请求，创建一个workerqueue，一次只传递一个intent到onHandleIntent中，从而不必担心多线程带来的问题。处理完毕之后自动调用stopSelf()方法；默认实现了Onbind（）方法，返回值为null；

模式实现了哦你StartCommand()方法，这个方法会放到workerqueue中，然后在onHandleIntent()中执行0

使用IntentService需要两个步骤：

1、写构造函数

2、复写onHandleIntent()方法

好处：处理异步请求的时候可以减少写代码的工作量，比较轻松地实现项目的需求

用Service来处理后台耗时操作，却很少注意到还有个IntentService

首先IntentService是继承自Service的，那我们先看看Service的官方介绍，这里列出两点比较重要的地方：

1.AServiceisnotaseparateprocess.TheServiceobjectitselfdoesnotimplyitisrunninginitsownprocess;unlessotherwisespecified,itrunsinthesameprocessastheapplicationitispartof.

2.AServiceisnotathread.Itisnotameansitselftodoworkoffofthemainthread(toavoidApplicationNotRespondingerrors).

稍微翻一下(英文水平一般)

1.Service不是一个单独的进程，它和应用程序在同一个进程中。

2.Service不是一个线程，所以我们应该避免在Service里面进行耗时的操作

关于第二点我想说下，不知道很多网上的文章都把耗时的操作直接放在Service的onStart方法中，而且没有强调这样会出现ApplicationNotResponding！希望我的文章能帮大家认清这个误区(Service不是一个线程，不能直接处理耗时的操作)。

有人肯定会问，那么为什么我不直接用Thread而要用Service呢？关于这个，大家可以网上搜搜，这里不过多解释。有一点需要强调，如果有耗时操作在Service里，就必须开启一个单独的线程来处理！！！这点一定要铭记在心。

IntentService相对于Service来说，有几个非常有用的优点，首先我们看看官方文档的说明：

IntentServiceisabaseclassforServicesthathandleasynchronousrequests(expressedasIntents)ondemand.ClientssendrequeststhroughstartService(Intent)calls;theserviceisstartedasneeded,handleseachIntentinturnusingaworkerthread,andstopsitselfwhenitrunsoutofwork.

This"workqueueprocessor"patterniscommonlyusedtooffloadtasksfromanapplication'smainthread.TheIntentServiceclassexiststosimplifythispatternandtakecareofthemechanics.Touseit,extendIntentServiceandimplementonHandleIntent(Intent).IntentServicewillreceivetheIntents,launchaworkerthread,andstoptheserviceasappropriate.

Allrequestsarehandledonasingleworkerthread--theymaytakeaslongasnecessary(andwillnotblocktheapplication'smainloop),butonlyonerequestwillbeprocessedatatime.

稍微翻译理一理，这里主要是说IntentService使用队列的方式将请求的Intent加入队列，然后开启一个workerthread(线程)来处理队列中的Intent，对于异步的startService请求，IntentService会处理完成一个之后再处理第二个，每一个请求都会在一个单独的workerthread中处理，不会阻塞应用程序的主线程，这里就给我们提供了一个思路，如果有耗时的操作与其在Service里面开启新线程还不如使用IntentService来处理耗时操作。下面给一个小例子：

1Service：

packagecom.zhf.service;

importAndroid.app.Service;

importAndroid.content.Intent;

importAndroid.os.IBinder;

publicclassMyServiceextendsService{

@Override

publicvoidonCreate(){

super.onCreate();

}

@Override

publicvoidonStart(Intentintent,intstartId){

super.onStart(intent,startId);

//经测试，Service里面是不能进行耗时的操作的，必须要手动开启一个工作线程来处理耗时操作

System.out.println("onStart");

try{

Thread.sleep(20000);

}catch(InterruptedExceptione){

e.printStackTrace();

}

System.out.println("睡眠结束");

}

@Override

publicIBinderonBind(Intentintent){

returnnull;

}

}

2.IntentService：

packagecom.zhf.service;

importAndroid.app.IntentService;

importAndroid.content.Intent;

publicclassMyIntentServiceextendsIntentService{

publicMyIntentService(){

super("yyyyyyyyyyy");

}

@Override

protectedvoidonHandleIntent(Intentintent){

//经测试，IntentService里面是可以进行耗时的操作的

//IntentService使用队列的方式将请求的Intent加入队列，然后开启一个workerthread(线程)来处理队列中的Intent

//对于异步的startService请求，IntentService会处理完成一个之后再处理第二个

System.out.println("onStart");

try{

Thread.sleep(20000);

}catch(InterruptedExceptione){

e.printStackTrace();

}

System.out.println("睡眠结束");

}

}

测试主程序：

packagecom.zhf.service;

importAndroid.app.Activity;

importAndroid.content.Intent;

importAndroid.os.Bundle;

publicclassServiceDemoActivityextendsActivity{

/\*\*Calledwhentheactivityisfirstcreated.\*/

@Override

publicvoidonCreate(BundlesavedInstanceState){

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

startService(newIntent(this,MyService.class));//主界面阻塞，最终会出现Applicationnotresponding

//连续两次启动IntentService，会发现应用程序不会阻塞，而且最重的是第二次的请求会再第一个请求结束之后运行(这个证实了IntentService采用单独的线程每次只从队列中拿出一个请求进行处理)

startService(newIntent(this,MyIntentService.class));

startService(newIntent(this,MyIntentService.class));

}

}

1. service的启动
2. 异步任务
3. 强引用
4. 新浪微博头像设置
5. 加密算法

MD5;SH1;双面解密

1. 高德地图
2. asynctask和handler的区别
3. 跳转图片释放

当前Activity包括多张图片当点击某一张图片跳转Activity时候之前Activity是否释放怎么释放的？

1. shareprefercnce保存用户名密码时候怎么加密
2. 离线下载怎么实现
3. 用过viewstub吗？include呢
4. android应用程序自启动
5. service是在主线程中吗？service中可不可以耗时操作
6. 数据库优化
7. socket
8. 屏幕适配
9. 关于图片压缩，用算法实现
10. 关于第三方分享
11. MVC在安卓的体现
12. surfaceview和VIEW和GLsurfaceview的区别
13. 都用过什么模式，经常用到的服务器是什么
14. 编写代码输入安卓abc汉字6字节输出安卓abc而不是安卓abc氵
15. 看一个for循环循环了几次
16. 怎杨跳出一个循环继续执行下一个循环
17. 使一个字符串逆序输出Stringa=abcde
18. 一个for循环，每次i+=2，截取字符串输出。
19. Math的操作，其实用了好几个方法，选择正确的结果
20. 线程的状态
21. 消息推送,xmpp轮询
22. udp/tcp
23. 动画
24. intent的几种模式
25. 分页加载，每页15条数据，拉到底部自动加载数据。
26. Android系统架构从小到大写

应用层:Java应用开发工程师开发的所有应用程序比如地图,浏览器,QQ等属于该层,手机中的短信,拨号,浏览器等这些应用程序都是可以被开发人员开发的其他应用程序所替换，这点不同于其他手机操作系统固化在系统内部的系统软件，更加灵活和个性化

应用框架层:Javaframework层源码OS定制开发为应用层开发人员提供API

系统运行库层:C语言包括C语言标准库,多媒体库,OpenGLES,SQLite,Webkit,Dalvik虚拟机等,该层是对应用框架层提供支持的层,Java访问硬件需通过NDK实现

Linux内核层:Android是基于Linux2.6内核，其核心系统服务如安全性、内存管理、进程管理、网路协议以及驱动模型都依赖于Linux内核

1. 用代码手写堆栈
2. 即时通讯
3. 多人聊天，视频，语音
4. 侧拉效果
5. 侧拉多界面，组件焦点
6. 别的公司封装好的播放器，从C中提取源码
7. AsyTask的三个参数是什么
8. finish()与System.exit(0)的区别，再谈谈你对退出的理解。

首先一个Activity是有生命周期onCreate,onStart,onResume,onPause,onStop,onDestroy...

finish是Activity的类，仅仅针对Activity，当调用finish()时，只是将活动推向后台，并没有立即释放内存，活动的资源并没有被清理；当调用System.exit(0)时，杀死了整个进程，这时候活动所占的资源也会被释放。

其次，android的机制决定了用户无法完全退出应用，当你的application最长时间没有被用过的时候，android自身会决定将application关闭了。

finish方法可以控制像Activity一样的组件，使其结束生命周期，但应用程序全局的资源并不会立刻释放，像static变量的值仍然会保留，保以再次使用。

1. 反编译和防止反编译

android基于java的，而java反编译工具很强悍，所以对正常apk应用程序基本上可以做到100%反编译还原。

因此开发人员如果不准备开源自己的项目就需要知道怎样防止反编译和反编译他人的项目来学习。

2.3版本以上在eclipse自动生成的default.properties文件中加上一句“proguard.config=proguard.cfg”可以对代码进行混淆，反编译后是很难看懂的。

2.3之前的SDK版本也没关系，把上面的proguard.cfg文件复制一份放到项目中，然后进行相同的操作即可。

Android-proguard混淆器使用，有引用第三方jar包的时候报错：

Youmayneedtospecifyadditionallibraryjars(using'-libraryjars')

处理方法：在proguard.cfg文件顶部加入

-dontwarncom.motorola.\*\*

-keepclasscom.motorola.\*\*{\*;}

1. tcp/ip,通讯思路
2. UDP，通讯思路
3. hTTP请求网络流程，HTTPclient
4. httpUrlconnection
5. Java对象池

Java内存泄漏

1. Int与intger的区别
2. 做项目的时候遇到哪些印象深刻的问题
3. 你在安卓开发中的心得
4. 网络互联使用哪种方式
5. 动画
6. 书架跟随书滚动
7. 3D效果如何实现的
8. 读取详细的新闻信息里面的图片读取多张不固定显示
9. 服务器端使用什么搭建的
10. 怎么调用android自带的功能
11. 图片的优化
12. Android每个进程分配的内存是多少
13. 都使用过那些第三方平台
14. 一个不同公司用一个APP的思路

有同一行业的100家公司业务流程全都不同要做一个app供他们办公使用请你给出思路(变态)

1. 平台兼容性

各个厂家生产的手机之中都有定制的模块怎么做到它们的兼容请提供思路

1. 重复登录怎么解决比如两台设备同时登录一个账号
2. 是每次获取图片都用AsyncTask获取吗
3. 用户体验度：获取数据之前等待的时间是怎么做的
4. 从网络获取图片的大概思路
5. 是否做过视频类,关于视频播放使用的什么方式
6. 大量耗时操作使用哪种方式,为什么
7. 建立一个Android工程，实现一个简单的书籍管理应用

具体功能如下：

1)启动程序时显示蓝色背景的欢迎（Splash）画面，界面上有一个进度圆环，旋转5秒后进入主界面；

2)主界面显示一个书籍列表，列表中至少显示8本书，每本书有封面图片、标题、简介、售价等信息。其中封面图片要使用远程地址，例

如：http://images.china-pub.com/ebook60001-65000/62279/shupi.jpg

3)在书籍列表中长按一本书，弹出“删除书籍”菜单项，用户点击“删除书籍”后从列表中删除此书籍；

4)屏幕底部设置“添加书籍”按钮：点击后弹出一个对话框，用户输入书籍信息（封面图片地址可不输入），然后点击“确定”按钮，此书

籍加入到书籍列表底部；

5)屏幕底部设置“偏好设置”按钮，点击后进入偏好设置界面，这个界面里提供一个“清除缓存”选项，用户点击后清除下载到本地的封面

图片缓存；

注：请在1.5小时以内完成，时间允许情况下按以上列表中的顺序尽量实现。

1. 图片上传怎么做的
2. 如果图片太大怎么处理
3. 项目开发流程
4. GSON的使用
5. 讯飞语音在什么地方用到
6. LBS定位用到哪个地图？为什么要使用这个？实现了哪些功能
7. 分页加载需要传递哪些参数
8. 移动互联使用了哪种方式
9. java.exe指的是
10. 掌握Android项目目录结构

1)src目录：src（即SourceCode)，src目录用来存放应用程序中所有的源代码，代码的源文件一般存放在该目录下的相

应的包下。

2)gen目录：gen目录，该目录下一般只有一个文件，即R文件。该文件夹下面有个R.java文件，R.java是在建立项目时自

动生成的，定义该项目所有资源的索引。

3)Android4.1目录：该文件夹下包含android.jar文件，这是一个Java归档文件，其中包含构建应用程序所需的所有的

AndroidSDK库（如Views、Controls）和APIs。

通过android.jar将自己的应用程序绑定到AndroidSDK和AndroidEmulator，这允许你使用所有Android的库和包，且使

你的应用程序在适当的环境中调试。

4)assets目录：该目录存放应用程序中使用的外部资源文件，如声音视频等，在Android程序中可以通过输入/输出流对

该目录中的文件进行读写。

5)res目录：该目录用来存放程序这用到的图片、界面布局文件及XML格式的描述文件。该目录下有多个目录。新建一

Android项目，在res目录下一般

会有drawable,layout,valuse三个目录。drawable用来存放图片资源,目录下一般会有三个目录分别存放高中低三种不同

分辨率的图片，layout目录用来存放应用程序的

界面布局文件，values目录下存放字符串资源，颜色资源，数组资源等XML文件。

6)AndroidManifest.xml

1. 掌握AndroidManifest.xml结构(应用程序入口)

AndroidManifest.xml是每个android程序中必须的文件。它位于整个项目的根目录，描述了package中暴露的组件（activities,services,等等），他们各自的实现类，各种能被处理的数据和启动位置。除了能声明程序中的Activities,ContentProviders,Services,和IntentReceivers,还能指定

permissions和instrumentation（安全控制和测试）

AndroidManifest.xml结构

<?xmlversion="1.0"encoding="utf-8"?>

<manifest>

<application>

<activity>

<intent-filter>

<action/>

<category/>

</intent-filter>

</activity>

<activity-alias>

<intent-filter></intent-filter>

<meta-data/>

</activity-alias>

<service>

<intent-filter></intent-filter>

<meta-data/>

</service>

<receiver>

<intent-filter></intent-filter>

<meta-data/>

</receiver>

<provider>

<grant-uri-permission/>

<meta-data/>

</provider>

<uses-library/>

</application>

<uses-permission/>

<permission/>

<permission-tree/>

<permission-group/>

<instrumentation/>

<uses-sdk/>

<uses-configuration/>

<uses-feature/>

<supports-screens/>

</manifest>

Android程序入口:

action节点中的android.intent.action.MAIN表明它所在的Activity是整个应用程序的入口点

1. 掌握模拟器的使用以及常用adb命令
2. adbdevices

查看当前连接设备

1. adbkill-server

终止adb进程

1. adbstart-server

启动adb进程

1. adbshell

进入shell模式(Linux模式),当连接有多个设备时,需执行命令

adb-s设备名shell

注:进入shell模式后序执行Linux命令

1. adbinstallapk路径

安装apk至设备,当连接有多个设备时,需执行命令

adb-s设备名installapk路径

当设备之前装有次apk时,替换安装序执行命令

adbinstall-rapk路径

1. adbuninstall要卸载的应用包名

卸载已安装的应用

1. adbpush要拷贝的文件路径设备上存放次文件的路径

将文件拷贝至设备sdcard,如adbpushF:\Android授课资料\项目实训\1205A\apk\com.moluo.android\_100111.apk/sdcard

1. adbpull设备上文件路径电脑存放文件路径

将文件从设备拷贝到电脑,如adbpull/sdcard/com.moluo.android\_100111.apkE:/

1. adblogcat

命令行的方式查看日志

若希望将日志保存至某文件,adblogcat->E:/test.txt

1. androidlistavd

查看当前创建的所有模拟器列表

1. emulator@模拟器名字

启动某个模拟器

1. 命令行操作数据库

adbshell

cddata/data

cdpackage

cddatabases

sqlite3test.db

>.tables

即可执行响应sql语句

* + 1. 面试题\_Java

## 概念

1. 作用域public,private,protected,以及不写时的区别

AnonymousInnerClass(匿名内部类)是否可以extends(继承)其它类，是否可以implements(实现)interface(接口)

匿名的内部类是没有名字的内部类。不能extends(继承)其它类，但一个内部类可以作为一个接口，由另一个内部类实现。

1. StaticNestedClass和InnerClass的不同

"NestedClass（一般是C++的说法），InnerClass(一般是JAVA的说法)。Java内部类与C++嵌套类最大的不同就在于是否有指向外部的引用上。

静态内部类（InnerClass）意味着1创建一个static内部类的对象，不需要一个外部类对象，2不能从一个static内部类的一个对象访问一个外部类对象

1. &和&&的区别

&和&&都可以用作逻辑与的运算符，表示逻辑与（and），当运算符两边的表达式的结果都为true时，整个运算结果才为true，否则，只要有一方为false，则结果为false。

&&还具有短路的功能，即如果第一个表达式为false，则不再计算第二个表达式，例如，对于if(str!=null&&!str.equals(“”))表达式，当str为null时，后面的表达式不会执行，所以不会出现NullPointerException如果将&&改为&，则会抛出NullPointerException异常。If(x==33&++y>0)y会增长，If(x==33&&++y>0)不会增长

&还可以用作位运算符，当&操作符两边的表达式不是boolean类型时，&表示按位与操作，我们通常使用0x0f来与一个整数进行&运算，来获取该整数的最低4个bit位，例如，0x31&0x0f的结果为0x01

1. Collection和Collections的区别

Collection是集合类的上级接口，继承与他的接口主要有Set和List.

Collections是针对集合类的一个帮助类，他提供一系列静态方法实现对各种集合的搜索、排序、线程安全化等操作。

1. 什么时候用assert。

assertion(断言)在软件开发中是一种常用的调试方式，很多开发语言中都支持这种机制。在实现中，assertion就是在程序中的一条语句，它对一个boolean表达式进行检查，一个正确程序必须保证这个boolean表达式的值为true；如果该值为false，说明程序已经处于不正确的状态下，assert将给出警告或退出。一般来说，assertion用于保证程序最基本、关键的正确性。assertion检查通常在开发和测试时开启。为了提高性能，在软件发布后，assertion检查通常是关闭的。

package com.huawei.interview;

public class AssertTest {

/\*\*

\* @param args

\*/

public static void main(String[] args) {

// TODO Auto-generated method stub

int i = 0;

for(i=0;i<5;i++)

{

System.out.println(i);

}

//假设程序不小心多了一句--i;

--i;

assert i==5;

}

}

1. Strings=newString("xyz");创建了几个StringObject

是2个不假。一个是编译时决定的，最后放在常量池中。一个是运行时放在堆里面的。两个都是"xyz"。还有的，s确实只是一个引用，本身不是对象。java不是纯粹的面向对象，也就是说并不是所有的都是对象，如基本类型等

1. Math.round(11.5)等於多少?Math.round(-11.5)等於多少

Math.round(11.5)==12

Math.round(-11.5)==-11

round方法返回与参数最接近的长整数，

1. shorts1=1;s1=s1+1;有什么错?shorts1=1;s1+=1;有什么错

当然有错误，short占2个字节，int占4个字节，short型变量+上int型值计算结果是int型，占4个字节，然后付给一个short型变量s1，就要丢失2个字节的精度，这是不容许的。你非要这么做的话，我给你提供一个办法：

s1=s1+1；改成s1+=1；虽然是一个意思，但是在jvm里面是不同的方法做的，是运算符重载，所以是不一样的。

1. Java有没有goto

保留关键字

1. 数组有没有length()这个方法?String有没有length()这个方法

数组没有length()这个方法，有length的属性。String有length()这个方法

1. 给我一个你最常见到的runtimeexception

常见的运行时异常有如下这些ArithmeticException,ArrayStoreException,BufferOverflowException,BufferUnderflowException,CannotRedoException,CannotUndoException,ClassCastException,CMMException,ConcurrentModificationException,DOMException,EmptyStackException,IllegalArgumentException,IllegalMonitorStateException,IllegalPathStateException,IllegalStateException,ImagingOpException,IndexOutOfBoundsException,MissingResourceException,NegativeArraySizeException,NoSuchElementException,NullPointerException,ProfileDataException,ProviderException,RasterFormatException,SecurityException,SystemException,UndeclaredThrowableException,UnmodifiableSetException,UnsupportedOperationException

1. error和exception有什么区别

error表示恢复不是不可能但很困难的情况下的一种严重问题。比如说内存溢出。不可能指望程序能处理这样的情况。exception表示一种设计或实现问题。也就是说，它表示如果程序运行正常，从不会发生的情况

1. abstract的method

是否可同时是static,是否可同时是native，是否可同时是synchronized

都不可以，因为abstract申明的方法是要求子类去实现的，abstract只是告诉你有这样一个接口，你要去实现，至于你的具体实现可以是native和synchronized，也可以不是，抽象方法是不关心这些事的，所以写这两个是没有意义的。然后，static方法是不会被覆盖的，而abstract方法正是要子类去覆盖它，所以也是没有意义的。所以，总的来说，就是java语法不允许你这样做，事实上，也没有意义这样做。

1. 接口是否可继承接口

?抽象类是否可实现(implements)接口?抽象类是否可继承实体类

(concreteclass)

1)接口可以继承接口..但是要使用extends~而不是用implements如:interfacea{}interfacebextendsa{}

2)抽象类可以实现接口..比如java.util中的AbstractCollection类就是实现的Collection接口

3)抽象类可以继承实体类下面这段执行无误的代码说明的所有的问题

1. 一个类实现接口，里面都变量Stringa;

接口中a=xxxx子类中是a=yyyy在子类中输出a=多少

1. 构造器Constructor是否可被override

构造器Constructor不能被继承，因此不能重写Override，但可以被重载Overload

1. 是否可以继承String类

String类是final类故不可以继承。

1. try{}里有一个return语句

try{}里有一个return语句，那么紧跟在这个try后的finally{}里的code会不会被执行，什么时候被执行，在return前还是后

根据java规范：在try-catch-finally中，如果try-finally或者catch-finally中都有return，则两个return语句都执行并且最终

返回到调用者那里的是finally中return的值；而如果finally中没有return，则理所当然的返回的是try或者catch中return的值，但是

finally中的代码是必须要执行的,而且是在return前执行,除非碰到exit()。

1. 用最有效率的方法算出2乘以8等於几

用移位运算inta=2<<3;

a就是2乘以8最后结果是16这是最省内存最有效率的方法

这个方法确实高效率的。我来解释一下：

2的二进制是10在32位存储器里面是0000000000000010

左移三位后变成0000000000010000也就是16

1. 当一个对象被当作参数传递到一个方法后

当一个对象被当作参数传递到一个方法后，此方法可改变这个对象的属性，并可返回变化后的结果，那么这里到底是值传递还是引用传递

是值传递。Java编程语言只有值传递参数。当一个对象实例作为一个参数被传递到方法中时，参数的值就是对该对象的引用。对象的内容可以在被调用的方法中改变，但对象的引用是永远不会改变的。

1. swtich是否能作用在byte上

，是否能作用在long上，是否能作用在String上

传递给switch和case语句的参数应该是int、short、char或者byte。long,string都不能作用于swtich

1. List,Set,Map是否继承自Collection接口

List和Set是继承自Collection接口的接口，Set不允许重复的项目，List允许重复项目，Set接口派生的类有TreeSet，HashSet，LinkedHashSet。List接口派生的类有ArrayList，Vector等。Map是独立的接口，不继承Collection接口

1. ArrayList和Vector的区别,HashMap和Hashtable的区别

答：就ArrayList与Vector主要从二方面来说.

1）同步性:Vector是线程安全的，也就是说是同步的，而ArrayList是线程序不安全的，不是同步的

2）数据增长:当需要增长时,Vector默认增长为原来一培，而ArrayList却是原来的一半

就HashMap与HashTable主要从三方面来说。

3）历史原因:Hashtable是基于陈旧的Dictionary类的，HashMap是Java1.2引进的Map接口的一个实现

1）同步性:Hashtable是线程安全的，也就是说是同步的，而HashMap是线程序不安全的，不是同步的

2）值：只有HashMap可以让你将空值作为一个表的条目的key或value

1. 说出ArrayList,Vector, LinkedList的存储性能和特性

ArrayList和Vector都是使用数组方式存储数据，此数组元素数大于实际存储的数据以便增加和插入元素，

它们都允许直接按序号索引元素，但是插入元素要涉及数组元素移动等内存操作，所以索引数据快而插入数据慢，

Vector由于使用了synchronized方法（线程安全），通常性能上较ArrayList差，而LinkedList使用双向链表实现存储，

按序号索引数据需要进行前向或后向遍历，但是插入数据时只需要记录本项的前后项即可，所以插入速度较快。

1. char型变量中能不能存贮一个中文汉字?为什么

char型变量是用来存储Unicode编码的字符的，unicode编码字符集中包含了汉字，所以，char型变量中当然可以存储汉字啦。不过，如果某个特殊的汉字没有被包含在unicode编码字符集中，那么，这个char型变量中就不能存储这个特殊汉字。补充说明：unicode编码占用两个字节，所以，char类型的变量也是占用两个字节。

1. 垃圾回收器的基本原理是什么？垃圾回收器可以马上回收内存吗

有什么办法主动通知虚拟机进行垃圾回收

对于GC来说，当程序员创建对象时，GC就开始监控这个对象的地址、大小以及使用情况。通常，GC采用有向图的方式记录和管理堆(heap)中的所有对象。通过这种方式确定哪些对象是"可达的"，哪些对象是"不可达的"。当GC确定一些对象为"不可达"时，GC就有责任回收这些内存空间。可以。程序员可以手动执行System.gc()，通知GC运行，但是Java语言规范并不保证GC一定会执行。

1. 垃圾回收的优点和原理。并考虑2种回收机制。

Java语言中一个显著的特点就是引入了垃圾回收机制，使c++程序员最头疼的内存管理的问题迎刃而解，它使得Java程序员在编写程序的时候不再需要考虑内存管理。由于有个垃圾回收机制，Java中的对象不再有"作用域"的概念，只有对象的引用才有"作用域"。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级别的线程运行，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楚和回收，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。回收机制有分代复制垃圾回收和标记垃圾回收，增量垃圾回收。

1. float型floatf=3.4是否正确

不正确。精度不准确,应该用强制类型转换，如下所示：floatf=(float)3.4或floatf=3.4f

在java里面，没小数点的默认是int,有小数点的默认是double;

1. 介绍JAVA中的CollectionFrameWork(包括如何写自己的数据结构)

Collection

├List

│├LinkedList

│├ArrayList

│└Vector

│　└Stack

└Set

Map

├Hashtable

├HashMap

└WeakHashMap

Collection是最基本的集合接口，一个Collection代表一组Object，即Collection的元素（Elements）

Map提供key到value的映射

事务属性ACID

答：原子性——一个事务要么完全执行，要么根本不执行。

一致性——一个事务在它完成后不能留下系统的不一致性。

隔离性——所有的事务不许允许在执行的时候不受其他进程或者事务的干涉。

持久性——在事务期间，所有提交的变更数据必须写入持久的数据存储器，并且应该能在硬件和软件失效中保存下来。

1. 实体Bean的三个状态：

答：no-state——Bean实例还没有创建。

pooled——Bean实例被创建，但还没有和一个EJBObject关联。

ready——与EJBObject相关联。若断开关联则回到pooled。

1. JDBC连接数据库6步

LoadtheJDBCDriver

EstablishtheDatabaseConnection

CreateaStatementObject

ExecuteaQuery

ProcesstheResults

ClosetheConnection

1. 事务的4大特性

答：原子性A，一致性C，隔离性I，永久性D

1. 查询语句之间的区别

selectcount（\*）fromstudent和selectcount（id）fromstudent之间的区别。

答案：

selectcount(\*)统计所有学生的记录个数，包括空记录。

Selectcount(Id)统计所有学生的记录个数，不包括null记录。

1. 假设现在有表system.table1

假设现在有表system.table1，表中有三个字段：id(数值型)、name（字符型）、age（数值型）写出SQL语句完成如下功能：在表中查出年龄大于20，且名字以“王”开头的记录，并且按照年龄的倒叙排列出来（年龄大的在前面）。

答案：

Select\*fromsystem.table1whereage>20andnamelike‘王%’orderbyageDESC;

1. 创建CUSTOMERS表

创建CUSTOMERS表，字段为：ID：（非空，主键）bigint，NAME：（非空）varchar，AGE：int类型；创建ORDERS表，字段为：ID：（非空，主键，）bigint，ORDER\_NUMBER：（非空）varchar，PRICE：double，CUSTOMER\_ID：（外键）bigint，设置级连删除；

答案：createtableCUSTOMBERS(

IDbigintnotnull,

NAMEvarchar(15),

AGEint,

primarykey(ID)

);

createtableORDERS(

IDbigintnotnull,

ORDER\_NUMBERvarchar(15)notnulll,

PRICEdoubleprecision,

CUSTOMER\_IDbigint,

primarykey(ID),

);

altertableORDERSaddconstraintFK\_CUSTOMERforeignkey(CUSTOMER\_ID)referencesCUSTOMERS(ID)ondeletecascade;

1. 使用左外连接查询，ORDERS和CUSTOMERS表，

答案：selectc.ID,o.CUSTOMER\_ID,c.NAME,o.IDORDER\_ID,ORDER\_NUMBERfromCUSTOMERScleftouterjoinORDERSonoc.ID=o.CUSTOMER\_ID;

1. 简述数据库事务的生命周期？（可画流程图）

答案：

1. deletefromtablea&truncatetabletablea的区别

　　truncate语句执行速度快,占资源少,并且只记录页删除的日志；

delete对每条记录的删除均需要记录日志

1. 什么是WebService?

　　WebService就是为了使原来各孤立的站点之间的信息能够相互通信、共享而提出的一种接口。

WebService所使用的是Internet上统一、开放的标准，如HTTP、XML、SOAP（简单对象访问协议）、WSDL等，所以WebService可以在任何支持这些标准的环境（Windows,Linux）中使用。

　　注：SOAP协议（SimpleObjectAccessProtocal,简单对象访问协议）,它是一个用于分散和分布式环境下网络信息交换的基于XML的通讯协议。在此协议下，软件组件或应用程序能够通过标准的HTTP协议进行通讯。它的设计目标就是简单性和扩展性，这有助于大量异构程序和平台之间的互操作性，从而使存在的应用程序能够被广泛的用户访问。

　　优势：

　　(1).跨平台。

　　(2).SOAP协议是基于XML和HTTP这些业界的标准的，得到了所有的重要公司的支持。

　　(3).由于使用了SOAP，数据是以ASCII文本的方式而非二进制传输，调试很方便；并且由于这样，它的数据容易通过防火墙，不需要防火墙为了程序而单独开一个“漏洞”。

　　(4).此外，WebService实现的技术难度要比CORBA和DCOM小得多。

　　(5).要实现B2B集成，EDI比较完善与比较复杂；而用WebService则可以低成本的实现，小公司也可以用上。

　　(6).在C/S的程序中，WebService可以实现网页无整体刷新的与服务器打交道并取数。

　　缺点：

　　(1).WebService使用了XML对数据封装，会造成大量的数据要在网络中传输。

　　(2).WebService规范没有规定任何与实现相关的细节，包括对象模型、编程语言，这一点，它不如CORBA。

1. 运行时异常与一般异常有何异同？

异常表示程序运行过程中可能出现的非正常状态，运行时异常表示虚拟机的通常操作中可能遇到的异常，是一种常见运行错误。java编译器要求方法必须声明抛出可能发生的非运行时异常，但是并不要求必须声明抛出未被捕获的运行时异常。

1. 说出Servlet的生命周期，并说出Servlet和CGI的区别。

Servlet被服务器实例化后，容器运行其init方法，请求到达时运行其service方法，service方法自动派遣运行与请求对应的doXXX方法（doGet，doPost）等，当服务器决定将实例销毁的时候调用其destroy方法。

与cgi的区别在于servlet处于服务器进程中，它通过多线程方式运行其service方法，一个实例可以服务于多个请求，并且其实例一般不会销毁，而CGI对每个请求都产生新的进程，服务完成后就销毁，所以效率上低于servlet。

1. 同步和异步有何异同，在什么情况下分别使用他们？举例说明。

如果数据将在线程间共享。例如正在写的数据以后可能被另一个线程读到，或者正在读的数据可能已经被另一个线程写过了，那么这些数据就是共享数据，必须进行同步存取。

当应用程序在对象上调用了一个需要花费很长时间来执行的方法，并且不希望让程序等待方法的返回时，就应该使用异步编程，在很多情况下采用异步途径往往更有效率。

1. heap和stack有什么区别

栈是一种线形集合，其添加和删除元素的操作应在同一段完成。栈按照后进先出的方式进行处理。

堆是栈的一个组成元素

1. forward和redirect的区别

forward是服务器请求资源，服务器直接访问目标地址的URL，把那个URL的响应内容读取过来，然后把这些内容再发给浏览器，浏览器根本不知道服务器发送的内容是从哪儿来的，所以它的地址栏中还是原来的地址。

redirect就是服务端根据逻辑,发送一个状态码,告诉浏览器重新去请求那个地址，一般来说浏览器会用刚才请求的所有参数重新请求，所以session,request参数都可以获取。

1. 说出数据连接池的工作机制是什么

J2EE服务器启动时会建立一定数量的池连接，并一直维持不少于此数目的池连接。客户端程序需要连接时，池驱动程序会返回一个未使用的池连接并将其表记为忙。如果当前没有空闲连接，池驱动程序就新建一定数量的连接，新建连接的数量有配置参数决定。当使用的池连接调用完成后，池驱动程序将此连接表记为空闲，其他调用就可以使用这个连接。

1. Set里的元素是不能重复的，那么用什么方法来区分重复与否呢

是用==还是equals()?它们有何区别

1)什么是Set?(what)

Set是Collection容器的一个子接口，它不允许出现重复元素，当然也只允许有一个null对象。

2)如何来区分重复与否呢？(how)

“用iterator()方法来区分重复与否”，这是在网上流传的答案，个人认为这是个错误的答案。JPI中写的很明白：“set不包含满足

e1.equals(e2)的元素对e1和e2”，由此可见回答使用equals()区分更合适。

3)为什么用equals()而不用==来区分？(why)

应该从它俩的区别谈起，==是用来判断两者是否是同一对象（同一事物），而equals是用来判断是否引用同一个对象。再看一下Set里面存的是

对象，还是对象的引用。根据java的存储机制可知，set里面存放的是对象的引用，所以当两个元素只要满足了equals()时就已经指向同一个对象，

也就出现了重复元素。所以应该用equals()来判断。

1. 两个对象值相同(x.equals(y) == true)

，但却可有不同的hash code，这句话对不对

对。

如果对象要保存在HashSet或HashMap中，它们的equals相等，那么，它们的hashcode值就必须相等。

如果不是要保存在HashSet或HashMap，则与hashcode没有什么关系了，这时候hashcode不等是可以的，例如arrayList存储的对象就不用实现hashcode，当然，我们没有理由不实现，通常都会去实现的。

1. 重写equals()方法的时候，我们还需要重写哪个方法？为什么

每个覆盖了equals方法的类中，也必须覆盖hashCode方法。  
如果不这样的话，就会违反Object.hashCode的通用约定，从而导致该类无法结合所有基于散列的集合一起正常运作，这样的集合包括HashMap、HashSet和Hashtable。

1. 当一个对象被当作参数传递到一个方法后

此方法可改变这个对象的属性，并可返回变化后的结果，那么这里到底是值传递还是引用传递?

是值传递。Java编程语言只有值传递参数。当一个对象实例作为一个参数被传递到方法中时，参数的值就是对该对象的引用。对象的内容可以在被调用的方法中改变，但对象的引用是永远不会改变的。

1. java中有几种方法可以实现一个线程

用什么关键字修饰同步方法?stop()和suspend()方法为何不推荐使用

有两种实现方法，分别是继承Thread类与实现Runnable接口

用synchronized关键字修饰同步方法

反对使用stop()，是因为它不安全。它会解除由线程获取的所有锁定，而且如果对象处于一种不连贯状态，那么其他线程能在那种状态下检查和修改它们。结果很难检查出真正的问题所在。suspend()方法容易发生死锁。调用suspend()的时候，目标线程会停下来，但却仍然持有在这之前获得的锁定。此时，其他任何线程都不能访问锁定的资源，除非被"挂起"的线程恢复运行。对任何线程来说，如果它们想恢复目标线程，同时又试图使用任何一个锁定的资源，就会造成死锁。所以不应该使用suspend()，而应在自己的Thread类中置入一个标志，指出线程应该活动还是挂起。若标志指出线程应该挂起，便用wait()命其进入等待状态。若标志指出线程应当恢复，则用一个notify()重新启动线程。

1. 简述synchronized和java.util.concurrent.locks.Lock的异同

主要相同点：Lock能完成synchronized所实现的所有功能

主要不同点：Lock有比synchronized更精确的线程语义和更好的性能。synchronized会自动释放锁，而Lock一定要求程序员手工释放，并且必须在finally从句中释放。

1. 当一个线程进入一个对象的一个synchronized方法后

，其它线程是否可进入此对象的其它方法?

不能，一个对象的一个synchronized方法只能由一个线程访问。

1. Java的接口和C++的虚类的相同和不同处。

由于Java不支持多继承，而有可能某个类或对象要使用分别在几个类或对象里面的方法或属性，现有的单继承机制就不能满足要求。与继承相比，接口有更高的灵活性，因为接口中没有任何实现代码。当一个类实现了接口以后，该类要实现接口里面所有的方法和属性，并且接口里面的属性在默认状态下面都是publicstatic,所有方法默认情况下是public.一个类可以实现多个接口。

1. Java中的异常处理机制的简单原理和应用。

当JAVA程序违反了JAVA的语义规则时，JAVA虚拟机就会将发生的错误表示为一个异常。违反语义规则包括2种情况。一种是JAVA类库内置的语义检查。例如数组下标越界,会引发IndexOutOfBoundsException;访问null的对象时会引发NullPointerException。另一种情况就是JAVA允许程序员扩展这种语义检查，程序员可以创建自己的异常，并自由选择在何时用throw关键字引发异常。所有的异常都是java.lang.Thowable的子类。

1. 描述一下JVM加载class文件的原理机制?

JVM中类的装载是由ClassLoader和它的子类来实现的,JavaClassLoader是一个重要的Java运行时系统组件。它负责在运行时查找和装入类文件的类。

1. 线程的基本概念、线程的基本状态以及状态之间的关系

线程指在程序执行过程中，能够执行程序代码的一个执行单位，每个程序至少都有一个线程，也就是程序本身。

Java中的线程有四种状态分别是：运行、就绪、挂起、结束。

1. 什么情况下调用doGet()和doPost()

Jsp页面中的form标签里的method属性为get时调用doGet()，为post时调用doPost()。

请求中的一个属性

1. 我们在web应用开发过程中经常遇到输出某种编码的字符

，如iso8859-1等，如何输出一个某种编码的字符串

PublicStringtranslate(Stringstr){

StringtempStr="";

try{

tempStr=newString(str.getBytes("ISO-8859-1"),"GBK");

tempStr=tempStr.trim();

}

catch(Exceptione){

System.err.println(e.getMessage());

}

returntempStr;

}

1. 排序都有哪几种方法？请列举。用JAVA实现一个快速排序。

排序的方法有：插入排序（直接插入排序、希尔排序），交换排序（冒泡排序、快速排序），选择排序（直接选择排序、堆排序），归并排序，分配排序（箱排序、基数排序）

快速排序的伪代码。

//使用快速排序方法对a[0:n-1]排序

从a[0:n-1]中选择一个元素作为middle，该元素为支点

把余下的元素分割为两段left和right，使得left中的元素都小于等于支点，而right中的元素都大于等于支点

递归地使用快速排序方法对left进行排序

递归地使用快速排序方法对right进行排序

所得结果为left+middle+right

1. JAVA语言如何进行异常处理

关键字：throws,throw,try,catch,finally分别代表什么意义？在try块中可以抛出异常吗？

Java通过面向对象的方法进行异常处理，把各种不同的异常进行分类，并提供了良好的接口。在Java中，每个异常都是一个对象，它是Throwable类或其它子类的实例。当一个方法出现异常后便抛出一个异常对象，该对象中包含有异常信息，调用这个对象的方法可以捕获到这个异常并进行处理。Java的异常处理是通过5个关键词来实现的：try、catch、throw、throws和finally。一般情况下是用try来执行一段程序，如果出现异常，系统会抛出（throws）一个异常，这时候你可以通过它的类型来捕捉（catch）它，或最后（finally）由缺省处理器来处理。

用try来指定一块预防所有"异常"的程序。紧跟在try程序后面，应包含一个catch子句来指定你想要捕捉的"异常"的类型。

throw语句用来明确地抛出一个"异常"。

throws用来标明一个成员函数可能抛出的各种"异常"。

Finally为确保一段代码不管发生什么"异常"都被执行一段代码。

可以在一个成员函数调用的外面写一个try语句，在这个成员函数内部写另一个try语句保护其他代码。每当遇到一个try语句，"异常"的框架就放到堆栈上面，直到所有的try语句都完成。如果下一级的try语句没有对某种"异常"进行处理，堆栈就会展开，直到遇到有处理这种"异常"的try语句。

1. 一个".java"源文件中是否可以包括多个类（不是内部类）？有什么限制

可以。必须只有一个类名与文件名相同。

1. java中有几种类型的流

JDK为每种类型的流提供了一些抽象类以供继承，请说出他们分别是哪些类？

字节流，字符流。字节流继承于InputStreamOutputStream，字符流继承于InputStreamReaderOutputStreamWriter。在java.io包中还有许多其他的流，主要是为了提高性能和使用方便。

1. 字节流与字符流的区别

要把一片二进制数据数据逐一输出到某个设备中，或者从某个设备中逐一读取一片二进制数据，不管输入输出设备是什么，我们要用统一的方式来完成这些操作，用一种抽象的方式进行描述，这个抽象描述方式起名为IO流，对应的抽象类为OutputStream和InputStream ，不同的实现类就代表不同的输入和输出设备，它们都是针对字节进行操作的。

在应用中，经常要完全是字符的一段文本输出去或读进来，用字节流可以吗？计算机中的一切最终都是二进制的字节形式存在。对于“中国”这些字符，首先要得到其对应的字节，然后将字节写入到输出流。读取时，首先读到的是字节，可是我们要把它显示为字符，我们需要将字节转换成字符。由于这样的需求很广泛，人家专门提供了字符流的包装类。

底层设备永远只接受字节数据，有时候要写字符串到底层设备，需要将字符串转成字节再进行写入。字符流是字节流的包装，字符流则是直接接受字符串，它内部将串转成字节，再写入底层设备，这为我们向IO设别写入或读取字符串提供了一点点方便。

字符向字节转换时，要注意编码的问题，因为字符串转成字节数组，

其实是转成该字符的某种编码的字节形式，读取也是反之的道理。

讲解字节流与字符流关系的代码案例：

import java.io.BufferedReader;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.InputStreamReader;

import java.io.PrintWriter;

public class IOTest {

public static void main(String[] args) throws Exception {

String str = "中国人";

/\*FileOutputStream fos = new FileOutputStream("1.txt");

fos.write(str.getBytes("UTF-8"));

fos.close();\*/

/\*FileWriter fw = new FileWriter("1.txt");

fw.write(str);

fw.close();\*/

PrintWriter pw = new PrintWriter("1.txt","utf-8");

pw.write(str);

pw.close();

/\*FileReader fr = new FileReader("1.txt");

char[] buf = new char[1024];

int len = fr.read(buf);

String myStr = new String(buf,0,len);

System.out.println(myStr);\*/

/\*FileInputStream fr = new FileInputStream("1.txt");

byte[] buf = new byte[1024];

int len = fr.read(buf);

String myStr = new String(buf,0,len,"UTF-8");

System.out.println(myStr);\*/

BufferedReader br = new BufferedReader(

new InputStreamReader(

new FileInputStream("1.txt"),"UTF-8"

)

);

String myStr = br.readLine();

br.close();

System.out.println(myStr);

}

}

1. java中会存在内存泄漏吗，请简单描述。

会。如：inti,i2;return(i-i2);//wheni为足够大的正数,i2为足够大的负数。结果会造成溢位，导致错误。

1. 什么是java序列化，如何实现java序列化？

或者请解释Serializable接口的作用。

我们有时候将一个java对象变成字节流的形式传出去或者从一个字节流中恢复成一个java对象，例如，要将java对象存储到硬盘或者传送给网络上的其他计算机，这个过程我们可以自己写代码去把一个java对象变成某个格式的字节流再传输，但是，jre本身就提供了这种支持，我们可以调用OutputStream的writeObject方法来做，如果要让java 帮我们做，要被传输的对象必须实现serializable接口，这样，javac编译时就会进行特殊处理，编译的类才可以被writeObject方法操作，这就是所谓的序列化。需要被序列化的类必须实现Serializable接口，该接口是一个mini接口，其中没有需要实现的方法，implements Serializable只是为了标注该对象是可被序列化的。

例如，在web开发中，如果对象被保存在了Session中，tomcat在重启时要把Session对象序列化到硬盘，这个对象就必须实现Serializable接口。如果对象要经过分布式系统进行网络传输或通过rmi等远程调用，这就需要在网络上传输对象，被传输的对象就必须实现Serializable接口。

1. 是否可以从一个static方法内部发出对非static方法的调用

不可以,如果其中包含对象的method()；不能保证对象初始化.

1. 写clone()方法时，通常都有一行代码，是什么

clone 有缺省行为，super.clone();因为首先要把父类中的成员复制到位，然后才是复制自己的成员。

1. 在JAVA中，如何跳出当前的多重嵌套循环

用break;return方法。

1. 说出一些常用的类，包，接口，请各举5个

常用的类：BufferedReaderBufferedWriterFileReaderFileWirterStringInteger

常用的包：java.langjava.awtjava.iojava.utiljava.sql

常用的接口：RemoteListMapDocumentNodeList

1. 内部类可以引用他包含类的成员吗？有没有什么限制

一个内部类对象可以访问创建它的外部类对象的内容

1. 面向对象的特征有哪些方面

**答：**主要有以下四方面：

1）抽象：

抽象就是忽略一个主题中与当前目标无关的那些方面，以便更充分地注意与当前目标有关的方面。抽象并不打算了解全部问题，而只是选择其中的一部分，暂时不用部分细节。抽象包括两个方面，一是过程抽象，二是数据抽象。

2）继承：

继承是一种联结类的层次模型，并且允许和鼓励类的重用，它提供了一种明确表述共性的方法。对象的一个新类可以从现有的类中派生，这个过程称为类继承。新类继承了原始类的特性，新类称为原始类的派生类（子类），而原始类称为新类的基类（父类）。派生类可以从它的基类那里继承方法和实例变量，并且类可以修改或增加新的方法使之更适合特殊的需要。

3）封装：

封装是把过程和数据包围起来，对数据的访问只能通过已定义的界面。面向对象计算始于这个基本概念，即现实世界可以被描绘成一系列完全自治、封装的对象，这些对象通过一个受保护的接口访问其他对象。

4）多态性：

多态性是指允许不同类的对象对同一消息作出响应。多态性包括参数化多态性和包含多态性。多态性语言具有灵活、抽象、行为共享、代码共享的优势，很好的解决了应用程序函数同名问题。

1. javac命令和java命令做什么事情呢

要知道java是分两部分的：一个是编译，一个是运行。

javac：负责的是编译的部分，当执行javac时，会启动java的编译器程序。对指定扩展名的.java文件进行编译。 生成了jvm可以识别的字节码文件。也就是class文件，也就是java的运行程序。

java：负责运行的部分.会启动jvm.加载运行时所需的类库,并对class文件进行执行.

一个文件要被执行,必须要有一个执行的起始点,这个起始点就是main函数.

1. Class.forName的作用?为什么要用

答：按参数中指定的字符串形式的类名去搜索并加载相应的类，如果该类字节码已经被加载过，则返回代表该字节码的Class实例对象，否则，按类加载器的委托机制去搜索和加载该类，如果所有的类加载器都无法加载到该类，则抛出ClassNotFoundException。加载完这个Class字节码后，接着就可以使用Class字节码的newInstance方法去创建该类的实例对象了。

有时候，我们程序中所有使用的具体类名在设计时（即开发时）无法确定，只有程序运行时才能确定，这时候就需要使用Class.forName去动态加载该类，这个类名通常是在配置文件中配置的，例如，spring的ioc中每次依赖注入的具体类就是这样配置的，jdbc的驱动类名通常也是通过配置文件来配置的，以便在产品交付使用后不用修改源程序就可以更换驱动类名。

1. 为什么要用 ORM? 和 JDBC 有何不一样?

orm是一种思想，就是把object转变成数据库中的记录，或者把数据库中的记录转变成objecdt，我们可以用jdbc来实现这种思想，其实，如果我们的项目是严格按照oop方式编写的话，我们的jdbc程序不管是有意还是无意，就已经在实现orm的工作了。

现在有许多orm工具，它们底层调用jdbc来实现了orm工作，我们直接使用这些工具，就省去了直接使用jdbc的繁琐细节，提高了开发效率，现在用的较多的orm工具是hibernate。也听说一些其他orm工具，如toplink,ojb等。

1. JDBC中的PreparedStatement相比Statement的好处

答：一个sql命令发给服务器去执行的步骤为：语法检查，语义分析，编译成内部指令，缓存指令，执行指令等过程。

select \* from student where id =3----缓存--xxxxx二进制命令

select \* from student where id =3----直接取-xxxxx二进制命令

select \* from student where id =4--- -会怎么干？

如果当初是select \* from student where id =?--- -又会怎么干？

上面说的是性能提高

可以防止sql注入。

1. 静态变量和实例变量的区别

在语法定义上的区别：静态变量前要加static关键字，而实例变量前则不加。

在程序运行时的区别：实例变量属于某个对象的属性，必须创建了实例对象，其中的实例变量才会被分配空间，才能使用这个实例变量。静态变量不属于某个实例对象，而是属于类，所以也称为类变量，只要程序加载了类的字节码，不用创建任何实例对象，静态变量就会被分配空间，静态变量就可以被使用了。总之，实例变量必须创建对象后才可以通过这个对象来使用，静态变量则可以直接使用类名来引用。

例如，对于下面的程序，无论创建多少个实例对象，永远都只分配了一个staticVar变量，并且每创建一个实例对象，这个staticVar就会加1；但是，每创建一个实例对象，就会分配一个instanceVar，即可能分配多个instanceVar，并且每个instanceVar的值都只自加了1次。

public class VariantTest{

public static int staticVar = 0;

public int instanceVar = 0;

public VariantTest(){

staticVar++;

instanceVar++;

System.out.println(“staticVar=” + staticVar + ”,instanceVar=” + instanceVar);

}

}

备注：这个解答除了说清楚两者的区别外，最后还用一个具体的应用例子来说明两者的差异，体现了自己有很好的解说问题和设计案例的能力，思维敏捷，超过一般程序员，有写作能力！

1. 请说出作用域public，private，protected，以及不写时的区别

这四个作用域的可见范围如下表所示。

说明：如果在修饰的元素上面没有写任何访问修饰符，则表示friendly。

作用域 当前类 同一package 子孙类 其他package

public √ √ √ √

protected √ √ √ ×

friendly √ √ × ×

private √ × × ×

备注：只要记住了有4种访问权限，4个访问范围，然后将全选和范围在水平和垂直方向上分别按排从小到大或从大到小的顺序排列，就很容易画出上面的图了。

1. 构造函数与方法的区别

1）构造方法是初始化一个类的对象时调用的，它没有返回值，而且名字必须与类的名字一样，  
public class wan  
{  
public static void main(String args[])  
{  
String str=new String("HelloWorld");//String类，String（）为构造方法，初始化一个对象str  
System.out.println(str);  
}  
  
}  
2）而成员函数是由类对象主动调用的，使用点操作符，它有返回值 。  
举个例子：  
public class wan   
{  
public static void main(String[] args) {  
String str=new String("Hello!");  
System.out.println(str);  
System.out.println("将Hello转化成全部大写："+str.toUpperCase());  
System.out.println("将Hello转化成全部小写"+str.toLowerCase());  
System.out.println("成员方法String（）内的字符长度是："+str.length());  
}  
}  
输出结果是：  
Hello!  
将Hello转化成全部大写：HELLO!  
将Hello转化成全部小写hello!  
成员方法String（）内的字符长度是：6总之，构造函数是赋予对象"生命特征"的,成员方法则是对象所具有的"行为,能力" 构造函数只有在建立对象时由系统调用的,其他任何时候你都别指望用他

1. 对于instance varisble 和 local variable jvm 再申请内存时有什么区别

根据声明方式详细来区分，Java的变量有七种,下面的程序代码展示了这七种变量的声明方式：

Class Myclass {

static int a;

int b;

public static void myMethod(int c) {

try {

int d;

} catch(Exception e) {

}

}

MyClass(int f) {

int[]g = new int[100];

}

}

class variable:声明在class内，method之外，且使用static修饰的变量,例如上面程序代码的a.

instance variable:声明在class内，method之外，且未使用static修饰的变量，例如上面程序的b.

method parameter:声明在method小括号内的变量，例如上面程序代码的c.

狭义的局部变量(local variable):声明在method内的变量，例如上面程序代码的d和g.

exception-handler parameter:声明在catch小括号内的变量，例如上面程序代码的e.

constructor parameter:声明在constructor小括号内的变量，例如上面程序代码的f.

数组元素(array element):数组的元素值没有识别名称，必须透过数组和索引值(index)来识别.例如上面

程序代码的g[0].

根据变量内存来分类

根据变量内存位置来区分，Java的变量有两种，包括了:

heap variable:占用的内存在heap中，这类变量包括了class variable,instance variable,array component,

即前面程序的a,b,g[0].这类变量会自动被JVM初始化默认值.

stack variable:通常广义的局部变量(pan-local variable),其占的内存在stack中，这类变量包括了狭义的

局部变量，method parameter,exception-handler parameter,constructor parameter,即前面程序 的c,d,e,f.狭义 的局部变量不会被JVM初始化成默认值，使用者必须自行初始化该变量,但是parameter类(包括method parameter,exception-handler parameter,constructor parameter)会被JVM初始化成传入值.

根据使用方式来为变量分类

根据使用方式，只要分三类即可，分别是:

class variable:即上例的a.

instance variable:即上例的b.

广义的局部变量:包含上例的c,d,e,f.这四者的差别很小，直接归为一类.

至于"数组元素"(array component)则不在此三类中，但是"数组元素"并不常被注意到，为它多分出一类的用处不大.我通常将数组视为对象,将array component视为对象的instance variable.

1. 如何设置两个应用在一个进程中
2. 在<manifest>里面添加： android:sharedUserId，注意这个属性的取值必须包含点（dot），也就是诸如java packae的形式。比如com.aaa.bbb。没有dot的话，将来adb install xxx.apk就会出错：Failure [INSTALL\_PARSE\_FAILED\_BAD\_SHARED\_USER\_ID]。最晕死的就是这一点在Android文档中没有提到，感谢万能的google赐予了我答案。所有application都要填写的一样。
3. 在<manifest>里面添加：android:sharedUserLabel="@string/shared\_user\_label"，这个label必须是一个string资源，不能是raw string。所有application都要填写的一样。
4. 在<application>里面添加：android:process="xxx.xxx.xxx"，这里所有的application都要填写的一样，内容就是process的名字，一般来说Android中process的名字就是manifest中的package的取值。
5. 所有的application用同样的一个key来sign。如果用Eclipse ADT plugin开发，由于所有的application都使用同一个debug key来sign，所以这一步没有什么额外的工作。更具体的有关sign apk的细节，参考Android文档：http://androidappdocs.appspot.com/guide/publishing/app-signing.html
6. android中常用的权限，最少写三个

读写权限、调用联系人数据的权限、网络权限、调用照相机硬件的权限

1. IO问题

有一个文件夹E://555下面有文件a.txt,b.txt,c.txt通过Java.io.file判断是文件夹还是文件，并获取文件夹以及文件的名字

1. java内存中占多少个字节

6

1. Android开发心得:思想，进程间通信，四大组件

Square开源交流，Dagger、Retrofit、Rxjava、Okhttp、Picasso、SQLBrite、Otto

QQ群：166354503