# Android面试试题

### 一般简单题

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？
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Dalivk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。
2. SIM卡的 EF 文件有何作用？
sim卡的文件系统有自己的规范，主要是为了和手机通讯，sim本身可以有自己的操作系统，EF就是做存储并和手机通讯用的。
3. 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，虚拟控件等技术。
4. 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。
5. 一条最长的短信息约占多少byte？
中文70(包括标点)，英文占160字节。
6. Android中的动画有哪几类，它们的特点和区别是什么？
两种，一中是Tween动画，一种是Frame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。
7. Handler机制的原理
Andriod提供了 Handler 和 Looper 来满足线程间的通信。Handler 先进先出原则。Looper类用来管理特定线程内对象之间的消息交换(Message Exchange)。
Looper: 一个线程可以产生一个Looper对象，由它来管理此线程里的Message Queue(消息队列)。
Handler: 你可以构造Handler对象来与Looper沟通，以便push新消息到Message Queue里；或者接收Looper从Message Queue取出)所送来的消息。
Message Queue(消息队列):用来存放线程放入的消息。
线程：UI thread 通常就是main thread，而Android启动程序时会替它建立一个Message Queue。
8. 说说MVC模式的原理，它在Android中的运用
MVC(Model\_view\_controller)” 模型-视图-控制器”。 MVC应用程序总是由这三个部分组成。Event(事件)导致Controller改变Model或View，或者同时改变两者。只要 Controller改变了Models的数据或者属性，所有依赖的View都会自动更新。类似的，只要Controller改变了View，View会从潜在的Model中获取数据来刷新自己。
--- ### View重绘和内存泄露
9. View的刷新
在需要刷新的地方，使用Handler.sendmessage发送消息，然后在Handler的getmessage里面执行invalidate或者invalidate
10. GC内存泄露
出现情况：
* 数据库的cursor没有关闭。
* 构造adapter时，没有使用缓存contentview。衍生listview的优化问题：减少创建view的对象，充分使用contentview，可以使用一静态类来优化处理getview的过程。
* Bitmap对象不使用时采用recycle()释放内存。

## activity中的对象的生命周期大于activity。

### Activity

1. Activity的生命周期
``` @Override protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.activity\_main); //在这里创建界面，做一些数据的初始化工作。 }

@Override protected void onStart() { super.onStart(); //到这一步变成用户可见不可交互的。 }

@Override protected void onRestart() { super.onRestart(); //從stop重新開始 }

@Override protected void onResume() { super.onResume(); //变成和用户可交互的 }

@Override protected void onPause() { super.onPause(); //到这一步是可见但不可交互的，系统会停止动画等消耗CPU的事情从上文的描述已经知道，应该在这里保存你的一些数据， // 因为这个时候你的程序的优先级降低，有可能被系统收回。在这里保存的数据，应该在 onResume里读出来， // 注意：这个方法里做的事情时间要短，因为下一个activity不会等到这个方法完成才启动。 }

@Override protected void onStop() { super.onStop(); //变得不可见，被下一个activity覆盖了。 }

@Override protected void onDestroy() { super.onDestroy(); //这是activity被干掉前最后一个被调用方法了，可能是外面类调用finish方法或者是系统为了 // 节省空间将它暂时性的干掉，可以用isFinishing()来判断它， // 如果你有一个Progress Dialog在线程中转动，请在onDestroy里把他cancel掉， // 不然等线程结束的时候，调用Dialog的cancel方法会抛异常的。 } 12. 让Activity变成一个窗口：Activity属性设定 可能有人希望做出来的应用程序是一个漂浮在手机主界面的东西，那么很简单你只需要设置一下Activity的主题就可以了在AndroidManifest.xml 中定义 Activity的地方一句话： android:theme="@android:style/Theme.Dialog" 这就使你的应用程序变成对话框的形式弹出来了，或者 android:theme="@android:style/Theme.Translucent" 就变成半透明的，类似的这种activity的属性可以在android.R.styleable 类的AndroidManifestActivity 方法中看到，AndroidManifest.xml中所有元素的属性的介绍都可以参考这个类android.R.styleable上面说的是属性名称，具体有什么值是在android.R.style中可以看到，比如这个"@android:style/Theme.Dialog"就对应于android.R.style.Theme\_Dialog ,（'\_'换成'.' ）就可以用在描述文件中了，找找类定义和描述文件中的对应关系就都明白了。 13. 你后台的Activity被系统回收怎么办：onSaveInstanceState 当你的程序中某一个Activity A 在运行时中，主动或被动地运行另一个新的Activity B 这个时候A会执行Java代码： @Override protected void onSaveInstanceState(Bundle outState) { super.onSaveInstanceState(outState); } 14. 面試流程[!http://www.bubuko.com/infodetail-992911.html] http://www.testtao.com/thread-32127-1-1.html 14. java.io包中定义了多个流类型来实现输入和输出功能，可以从不同的角度对其进行分 类,按功能分为：(节点流和处理流),如果为读取的内容进行处理后再输出，需要使用下列哪种流? Filter stream 15. 下列代码的执行结果是：(B) public class Test3{ public static void main(String args[]){ System.out.print(100%3); System.out.print(","); System.out.println(100%3.0); } } A、1,1 B、1,1.0 C、1.0,1 D、1.0,1.0 16. 在继承中，关于构造方法的说明，下列说法错误的是(D) A、子类无条件的继承父类的无参构造方法， 　　B、子类可以引用父类中的有参构造方法，使用super关键字， 　　C、如果子类没有构造方法，则父类无参构造方法作为自已的构造方法， 　　D、如果子类有无参构造方法，而父类的无参构造方法则被覆盖。 17. 以下程序的运行结果为(B) public class IfTest{

　　public static void main(String args[]){

　　int x=3;

　　int y=1;

　　if(x==y)

　　System.out.println("Not equal");

　　else

　　System.out.println("Equal");

　　} 　　} A、Not equal B、Equal C、无输出 D、编译出错 18. Java语言中字符串“学Java”所占的内存空间是(a) A. 6个字节 B. 7个字节 C. 10个字节 D. 11个字节 19. 关于下列程序段的输出结果，说法正确的是：( D) public class MyClass{

　　static int i; public static void main(Stringargv[]){ System.out.println(i); } } ``` A、有错误，变量i没有初始化。 B、null C、1 D、0
20. 下列哪些语句关于内存回收的说明是正确的? (B )
A、 程序员必须创建一个线程来释放内存

　　B、 内存回收程序负责释放无用内存

　　C、 内存回收程序允许程序员直接释放内存

　　D、 内存回收程序可以在指定的时间释放内存对象 21. 下面异常是属于Runtime Exception 的是(ABCD)(多选)
A、ArithmeticException

　　B、IllegalArgumentException

　　C、NullPointerException

　　D、BufferUnderflowException 22. Math.round(11.5)等于多少(). Math.round(-11.5)等于多少(C).
A、11 ,-11 B、11 ,-12 C、12 ,-11 D、12 ,-12 23. 下列程序段的输出结果是：(B )
void complicatedexpression\_r(){ int x=20, y=30; boolean b; b=x>50&&y>60||x>50&&y<-60||x<-50&&y>60||x<-50&&y<-60; System.out.println(b); } A、true B、false C、1 D、0 24. activity对一些资源以及状态的操作保存，最好是保存在生命周期的哪个函数中进行(A) A、onPause() B、onCreate() C、 onResume() D、onStart() 25. Intent传递数据时，下列的数据类型哪些可以被传递(ABCD)(多选) A、Serializable B、charsequence C、Parcelable D、Bundle 26. android 中下列属于Intent的作用的是(C)
A、实现应用程序间的数据共享

　　B、是一段长的生命周期，没有用户界面的程序，可以保持应用在后台运行，而不会因为切换页面而消失

　　C、可以实现界面间的切换，可以包含动作和动作数据，连接四大组件的纽带

　　D、处理一个应用程序整体性的工作 27. 下列属于SAX解析xml文件的优点的是(B)
A、将整个文档树在内存中，便于操作，支持删除，修改，重新排列等多种功能

　　B、不用事先调入整个文档，占用资源少

　　C、整个文档调入内存，浪费时间和空间

　　D、不是长久驻留在内存，数据不是持久的，事件过后，若没有保存数据，数据就会

　　消失 28. 在android中使用Menu时可能需要重写的方法有(AC)。(多选) A、onCreateOptionsMenu(AC)

　　B、onCreateMenu()

　　C、onOptionsItemSelected()

　　D、onItemSelected() 29. 在android中使用SQLiteOpenHelper这个辅助类时，可以生成一个数据库，并可以对数 据库版本进行管理的方法可以是(AB)
A、getWriteableDatabase()

　　B、getReadableDatabase()

　　C、getDatabase()

　　D、getAbleDatabase()
30. android 关于service生命周期的onCreate()和onStart()说法正确的是(AD)(多选题)
A、当第一次启动的时候先后调用onCreate()和onStart()方法

　　B、当第一次启动的时候只会调用onCreate()方法

　　C、如果service已经启动，将先后调用onCreate()和onStart()方法

　　D、如果service已经启动，只会执行onStart()方法，不在执行onCreate()方法 31. 下面是属于GLSurFaceView特性的是(ABC)(多选)
A、管理一个surface，这个surface就是一块特殊的内存，能直接排版到android的视图

　　view上。

　　B、管理一个EGL display，它能让opengl把内容渲染到上述的surface上。

　　C、让渲染器在独立的线程里运作，和UI线程分离。

　　D、可以直接从内存或者DMA等硬件接口取得图像数据
32. 关于ContenValues类说法正确的是(A)
A、他和Hashtable比较类似，也是负责存储一些名值对，但是他存储的名值对当中的

　　名是String类型，而值都是基本类型

　　B、他和Hashtable比较类似，也是负责存储一些名值对，但是他存储的名值对当中的

　　名是任意类型，而值都是基本类型

　　C、他和Hashtable比较类似，也是负责存储一些名值对，但是他存储的名值对当中的

　　名，可以为空，而值都是String类型

　　D、他和Hashtable比较类似，也是负责存储一些名值对，但是他存储的名值对当中

　　的名是String类型，而值也是String类型 33. 我们都知道Hanlder是线程与Activity通信的桥梁,如果线程处理不当，你的机器

　　就会变得越慢，那么线程销毁的方法是(A)
A、onDestroy()

　　B、onClear()

　　C、onFinish()

　　D、onStop() 34. 下面退出Activity错误的方法是(C)
A、finish()

　　B、抛异常强制退出

　　C、System.exit()

　　D、onStop() 35. 下面属于android的动画分类的有(AB)(多项) A、Tween B、FrameC、Draw D、Animation 36. Android项目工程下面的assets目录的作用是什么 B A、放置应用到的图片资源。

　　B、主要放置多媒体等数据文件

　　C、放置字符串，颜色，数组等常量数据

　　D、放置一些与UI相应的布局文件，都是xml文件 37. 关于res/raw目录说法正确的是 A A、 这里的文件是原封不动的存储到设备上不会转换为二进制的格式

　　B、 这里的文件是原封不动的存储到设备上会转换为二进制的格式

　　C、 这里的文件最终以二进制的格式存储到指定的包中

　　D、 这里的文件最终不会以二进制的格式存储到指定的包中 38. 下列对android NDK的理解正确的是(ABCD ) A、 NDK是一系列工具的集合

　　B、 NDK 提供了一份稳定、功能有限的 API 头文件声明。

　　C、 使“Java+C” 的开发方式终于转正，成为官方支持的开发方式

　　D、 NDK 将是 Android 平台支持 C 开发的开端 39. 我们用\_***int****来定义一个整数，用****char来定义一个字符类型，称为原始数据类型。***
***40. android中常用的四个布局是，***\_\_\_\_\_\_\_，\_\_\_\_\_\_\_\_\_\_和\_\_\_\_\_\_\_\_\_\_。 FrameLayout,LilnearLayout,RelativeLayout,TableLayout 41. android 的四大组件是\_\_\_\_\_\_\_\_\_\_，Activity,Broadcast,Service,ContentProvide 42. java.io包中的\_\_\_\_\_\_\_\_\_\_\_\_和\_\_\_\_\_\_\_\_\_\_\_\_类主要用于对对象(Object)的读写。 ObjectInputStream ObjectOutputSteam 43. android 中service的实现方法是：\_\_\_\_\_\_\_\_\_\_\_\_\_和\_\_\_\_\_\_\_\_\_\_\_\_\_。startService ,bindService 44.activity一般会重载7个方法用来维护其生命周期，除了onCreate(),onStart(),onDestory()外还有\_\_\_\_\_\_\_\_\_\_\_\_\_,\_\_\_\_\_\_\_\_\_\_\_\_\_\_,\_\_\_\_\_\_\_\_\_\_\_\_,\_\_\_\_\_\_\_\_\_\_\_。 onRestart(),onResume(),onPause(),onStop() 45. android的数据存储的方式\_\_\_\_\_\_\_\_\_,\_\_\_\_\_\_\_\_\_\_\_,\_\_\_\_\_\_\_\_\_\_,\_\_\_\_\_\_\_\_\_,\_\_\_\_\_\_\_\_。SharedPreferences存储，文件存储，SQLite存储，ContentProvider，网络存储 46. 当启动一个Activity并且新的Activity执行完后需要返回到启动它的Activity来执行的回调函数是\_***startActivityResult()***\_\_\_\_\_\_\_\_\_\_。 47. 请使用命令行的方式创建一个名字为myAvd,sdk版本为2.2,sd卡是在d盘的根目录下，名字为scard.img，并指定屏幕大小HVGA.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_。adnroid create acd -n myAvd -t 8 -s HVDA - Cd:.img 48. 程序运行的结果是：\_\_\_\_\_\_\_good and gbc\_\_\_\_\_\_\_\_。
public classExample{ 　　Stringstr=new String("good"); 　　char[]ch={'a','b','c'}; 　　publicstatic void main(String args[]){ 　　Exampleex=new Example(); 　　ex.change(ex.str,ex.ch); 　　System.out.print(ex.str+"and "); 　　Sytem.out.print(ex.ch); 　　} 　　public voidchange(String str,char ch[]){ 　　str="testok"; 　　ch[0]='g'; 　　} 　　}
49. 横竖屏切换时候 activity 的生命周期
1. 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 . 2. 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 . 3. 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 . 50. android 中线程与线程，进程与进程之间如何通信
1 、一个 Android 程序开始运行时，会单独启动一个 Process 。
默认情况下，所有这个程序中的 Activity 或者 Service 都会跑在这个 Process 。
默认情况下，一个 Android 程序也只有一个 Process ，但一个 Process 下却可以有许多个 Thread 。
2 、一个 Android 程序开始运行时，就有一个主线程 Main Thread 被创建。该线程主要负责 UI 界面的显示、更新和控件交互，所以又叫 UI Thread 。
一个 Android 程序创建之初，一个 Process 呈现的是单线程模型 -- 即 Main Thread ，所有的任务都在一个线程中运行。所以， Main Thread 所调用的每一个函数，其耗时应该越短越好。而对于比较费时的工作，应该设法交给子线程去做，以避免阻塞主线程（主线程被阻塞，会导致程序假死 现象）。
3 、 Android 单线程模型： Android UI 操作并不是线程安全的并且这些操作必须在 UI 线程中执行。如果在子线程中直接修改 UI ，会导致异常。
51. 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
52. 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。
53. Android中五种数据存储方式分别是什么？他们的特点？
(1)SharedPreference，存放较少的五种类型的数据，只能在同一个包内使 用，生成XML的格式存放在设备中 (2) SQLite数据库，存放各种数据，是一个轻量级的嵌入式数据库 (3) File文件，通过读取写入方式生成文件存放数据 (4) ContentProvider，主要用于让其他应用程序使用保存的数据 (5) 通过网络获取数据和写入数据到网络存储空间

 答：Android提供了五种存取数据的方式

1. 说说 android 中 mvc 的具体体现
mvc是model,view,controller的缩写，mvc包含三个部分： 模型（model）对象：是应用程序的主体部分，所有的业务逻辑都应该写在该层。 视图（view）对象：是应用程序中负责生成用户界面的部分。也是在整个mvc架构中用户唯一可以看到的一层，接收用户的输入，显示处理结果。 控制器（control）对象：是根据用户的输入，控制用户界面数据显示及更新model对象状态的部分，控制器更重要的一种导航功能，响应用户出发的相关事件，交给m层处理。 android鼓励弱耦合和组件的重用，在android中mvc的具体体现如下：
1)视图（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。 2)控制层（controller）：android的控制层的重任通常落在了众多的acitvity的肩上，这句话也就暗含了不要在acitivity中写过多的代码，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，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。
2. 简述SharedPreferences存储方式以及SharedPreferences与SQLite数据库的区别
SharedPreferences也是一种轻型的数据存储方式，它的本质是基于XML文件存储key-value键值对数据，通常用来存储一些简单的配置信息。其存储位置在/data/data//shared\_prefs目录下。SharedPreferences对象本身只能读取数据而不支持写入数据，存储修改是通过Editor对象实现。SharedPreferences对象与SQLite数据库相比，免去了创建数据库，创建表，写SQL语句等诸多操作，相对而言更加方便，简洁。但是SharedPreferences也有其自身缺陷，比如其职能存储boolean，int，float，long和String五种简单的数据类型，比如其无法进行条件查询等。所以不论SharedPreferences的数据存储操作是如何简单，它也只能是存储方式的一种补充，而无法完全替代如SQLite数据库这样的其他数据存储方式。
3. 描述handler 机制的原理
andriod提供了 Handler 和 Looper 来满足线程间的通信。 Handler 先进先出原则。 Looper类用来管理特定线程内对象之间的消息交换(Message Exchange)。 1)Looper: 一个线程可以产生一个Looper对象，由它来管理此线程里的Message Queue(消息队列)。 2)Handler: 你可以构造Handler对象来与Looper沟通，以便push新消息到Message Queue里;或者接收Looper从Message Queue取出)所送来的消息。
4. Message Queue(消息队列):用来存放线程放入的消息。 4)线程：UI thread 通常就是main thread，而Android启动程序时会替它建立一个Message Queue。
5. 显式intent和隐式intent的区别是什么（android）
答：Intent定义：Intent是一种在不同组件之间传递的请求消息，是应用程序发出的请求和意图。作为一个完整的消息传递机制，Intent不仅需要发送端，还需要接收端。 显式Intent定义：对于明确指出了目标组件名称的Intent，我们称之为显式Intent。 隐式Intent定义：对于没有明确指出目标组件名称的Intent，则称之为隐式Intent。 说明：Android系统使用IntentFilter 来寻找与隐式Intent相关的对象。
6. sqlite升级步骤：
1.自己写一个类继承自SqliteOpenHelper

2.会实现SqliteOpenHelper的两个方法 onCreate与onUpgrade，google文档对两个回调方法的解释是创建数据库的时候调用与更新数据库的版本的时候调用

3.Sqlite数据库主要是用来缓存应用的数据,而应用却是一直在更新版本，相应的数据的表的字段也会一直增加会改变或减少

4.这个时候就需要控制数据库的版本,因为Sqlite数据库中的字段假设新版的应用里面设计的表是10个字段，而缓存却是之前缓存的只有9个字段的话，查询数据库之后的列

然后取的值会出现空指针异常或报错

5.所以android中引入了Sqlite数据库的版本，让应用的旧版数据库能够与新版的数据库的字段兼容

6.为了兼容之前的数据库的版本,只需要在应用的版本更新的时候,添加字段或者删除字段即可

7.你开发程序当前是1.0.0的版本，该程序用到了数据库，但是版本迭代之后到1.0.1的时候，数据库的某个表添加了某个字段在软件1.0.1的版本就需要升级

8.数据库升级可以为了能够让旧的数据不能丢，所以不能删除掉之前数据库中的所有数据，那么就需要有地方能够检测到版本的变化，这个跟Android的APP升级是一个道理

当然这个检测就是在SqliteOpenHelper的onUpgrade方法中 59. 数据库升级应该注意什么？
软件的1.0版本升级到1.1版本时，老的数据不能丢。那么在1.1版本的程序中就要有地方能够检测出来新的软件版本与老的 数据库不兼容，并且能够有办法把1.0软件的数据库升级到1.1软件能够使用的数据库。换句话说，要在1.0软件的数据库的那个表中增加那个字段，并赋予这个字段默认值。 60. 程序如何知道数据库需要升级？
SQLiteOpenHelper类的构造函数有一个参数是int version，它的意思就是指数据库版本号。比如在软件1.0版本中，我们使用SQLiteOpenHelper访问数据库时，

该参数为1，那么数据库版本号1就会写在我们的数据库中。

到了1.1版本，我们的数据库需要发生变化，那么我们1.1版本的程序中就要使用一个大于1的整数来构造SQLiteOpenHelper类，用于访问新的数据库，比如2。

当我们的1.1新程序读取1.0版本的老数据库时，就发现老数据库里存储的数据库版本是1，而我们新程序访问它时填的版本号为2，系统就知道数据库需要升级。 61. android版本适配(如何兼容4.3-2.3版本)
比如产品设计中想要一些4.3以上的新特效，但是如何去兼容4.3-2.3的用户群体呢， 前提是我们apk在友盟数据上显示4.3-2.3占有25%的用户群体。 居于这个的考虑，我们目前的做法就是新设计的页面使用新特效的话需要根据手机版本号判断， 如果是低版本的手机并且大部分新特效是无法兼容我们展示老页面. 62. 一个apk如何快速方便的打多个不同包名的产品（多渠道多产品推广）
我们市场在推广apk的时候有时候需要根据渠道打不同包名的apk并且这些打出来的apk风格和内容展示以及文字展现略有不同。 我们现在的做法是，把主工程项目当做libs形式关联到想要打包的工程，这样打不同包名的时候就方便，直接创建一个工程， 把主工程关联，然后可以在新创建的工程里面略修改一些比如title风格，首页面进入风格 （因为首页我们做了好几套可以根据类型来判断你走的是哪一个风格），就是一个新的apk出现了。 63. android 适配
1、不要使用绝对布局

2、尽量使用match\_parent 而不是fill\_parent 。

3、能够使用权重的地方尽量使用权重（android:layout\_weight）

4、如果是纯色背景，尽量使用android的shape 自定义。

5、如果需要在特定分辨率下适配，可以在res目录上新建layout-HxW.xml的文件夹。比如要适配1080\*1800的屏幕

（魅族MX3采用此分辨率）则新建layout-1800x1080.xml的文件夹，然后在下面定义布局。Android系统会优先查

找分辨率相同的布局，如果不存在则换使用默认的layout下的布局。 64. ArrayList,Vector,LinkedList的区别
ArrayList Vector LinkedList
实现原理 数组 数组 双向链表
线程安全 否 是 否 优点 1.数组实现优于遍历 1.数组实现优于遍历 1.节点的增删无需对象的重建 2.非线程安全，效率较高 2.线程安全 2.空间利用毫无浪费 缺点 1.非线程安全 1.数组中未使用的元素造成空间的浪费 1.遍历效率较低 2.数组中未使用元素照成了空间的浪费 2.扩容可能引起对象的重建 2.非线程安全 3.扩容可能引起对象的重建 3.线程安全，效率相对低 4.增删有可能引起数组元素的移动 4.增删有可能引起数组元素的移动 扩容 0.5倍增量 1倍增量 按需增删 使用场景 1.无线程的要求 1.有线程安全的要求 增删场景较多的时候 2.遍历较多，增删较少 2.遍历场景较多，增删场景较少
65. int与Integer的区别
int Integer 类型 基本类型 复合类型 默认值 0 null 存储 栈（局部变量）堆（成员变量，有待进一步确认） 堆上（只能通过new创建） 方法 基本类型无方法 有 速度 快（栈上 的操作相对快） 慢 泛型支持 否（java中的泛型不支持，C++中的模板支持） 支持 容器类支持 否（直接使用通常会进行装箱操作） 支持 存在意义 1.历史原因（顺延C/C++中存在）2.方便快速（无需new） 基本类型int的包装类,提供了对泛型，容器类的支持
66. RuntimeException与普通异常，error的区别。
Checked Exception：在编译时就能够被Java编译器所检测到的。
UncheckedException：则是编译时，java编译器不能检查到。
RuntimeException 普通Exception Error 受控异常 否 是 否 产生原因 开发者的编程错误 由于外界环境所限，本身潜在的一些问题 Java运行时的系统错误，资源耗尽，是一种严重的，程序无法修复的问题
例子 NullPointerException ClassNotFoundException VirtualMachineError ArrayOutOfIndexException IOException StackOverflowError ClassCastException FileNotFoundException OutOfMemoryError ArithmeticException UnsupportedOperationException 67. final,finalize,finally的区别
final:关键字，表不变 方法：方法不可Override 类：不可被继承 基本类型量：常量，值不可变 符合类型量：引用不可变，即引用的值不可变
finally:关键字，Java异常处理机制的一部分，在异常发生时，用来提供一个必要的清理的机会。
finalize：Object类的方法 意义：Java技术允许使用finalize()方法在垃圾回收器将对象回收之前，做一些必要的清理操作。 调用前提：这个对象确定没有被引用到。 工作原理： 垃圾收集器准备好释放对象占用的空间。 首先调用其finalize方法。 下一次垃圾收集过程中，真正回收内存。 不确定性： finalize的执行时间是不缺定的。 一个对象引用另一个对象，并不能保证finalize的方法按照特定的执行顺序。 68. Override,Overload
Override Overload 签名+返回值 相同 方法名相同，签名不同 关系 父子类继承关系 通常是同一类层次中 识别 运行时多态 编译时多态 根据具体的对象 由对象的外观类型（即声明类型）决定 查询对象的虚方法表，确定调用关系 修饰符限制 非private 无特别 非static 非final 异常关系 子类方法不能抛出被父类方法更多的异常 无特别 可见性关系 子类不能比父类访问权限更窄（里氏替换原则决定） 无特别 59. Collection Collections
Collection:接口，集合类的接口，一个契约，提供了集合基本的大小，添加，清除，遍历方法等。 Collections:工具类，提供了很多静态方法，给集合提供一些查询，比较，排序，交换，线程安全化等方法。 60. sleep方法和wait方法的区别
wait sleep 所属类 Object Thread 意义 让线程挂起 让线程休眠指定的时间 释放锁 是 否（这个跟锁本来就没有关系） 恢复 1.有参：wait指定时间2.无参：等待其他线程notify 1.根据参数长度自动恢复。2.异常打断 使用限制 wait，notify必须持有当前对象锁的情况下调用 无特别 抛出异常 否 是 静态方法 否 是 61. HashMap和Hashtable的区别。
HashMap是Hashtable的轻量级实现（非线程安全的实现），他们都完成了Map接口，主要区别在于HashMap允许空（null）键值（key）,由于非线程安全，效率上可能高于Hashtable。
HashMap允许将null作为一个entry的key或者value，而Hashtable不允许。
HashMap把Hashtable的contains方法去掉了，改成containsvalue和containsKey。因为contains方法容易让人引起误解。 62. 算法 1.冒泡 2.选择 3.插入 4.并归 5.快速 63. 深入探索Java工作原理：JVM内存回收及其他
1．Java虚拟机：
Java源程序通过编译器编译成.Class文件,然后java虚拟机中的java 解释器负责将字节码文件解释成为特定的机器码进行运行。 java是一种半编译半解释型语言。半编译是指：java源代码，会经过javac命令变成 .class文件。半解释是指： .class文件被jvm解释的过程。也就是因为jvm的半解释才有了java的动态语言特性：反射和annotation。 2.和android区别
alvik有自己的libdex库负责对.class进行处理。libdex主要对.class进行处理生成自己的dex文件。主要做的工作是，对虚拟机指令进行转换(dalvik是基于寄存器的，sun虚拟机是基于栈的)，对类的静态数据进行归类、压缩。dalvik基于寄存器，而JVM基于stack，Dalvik执行的是特有的DEX文件格式，而JVM运行的是*.class文件格式。 3.优势*
*1、在编译时提前优化代码而不是等到运行时 2、 虚拟机很小，使用的空间也小；被设计来满足可高效运行多种虚拟机实例。 Java虚拟机的建立需要针对不同的软硬件平台来实现，既要考虑处理器的型号，也要考虑操作系统的种类。由此在SPARC结构、X86结构、MIPS和PPC等嵌入式处理芯片上，在UNIX、Linux、Windows和部分实时操作系统上都可实现Java虚拟机。 4.无用内存自动回收机制 而在Java运行环境中，始终存在着一个系统级的线程，专门跟踪内存的使用情况， 定期检测出不再使用的内存，并自动进行回收，避免了内存的泄露，也减轻了程序员的工作量。 5.JVM JVM是Java平台的核心，为了让编译产生的字节码能更好地解释与执行，因此把JVM分成了6个部分：JVM解释器、指令系统、寄存器、栈、存储区和碎片回收区。 64. Android框架 http://www.cnblogs.com/forlina/archive/2011/06/29/2093332.html 65. 基于android的Socket通信 一、Socket通信简介 Android 与服务器的通信方式主要有两种，一是Http通信，一是Socket通信。两者的最大差异在于，http连接使用的是“请求—响应方式”，即在请求时建立 连接通道，当客户端向服务器发送请求后，服务器端才能向客户端返回数据。而Socket通信则是在双方建立起连接后就可以直接进行数据的传输，在连接时可 实现信息的主动推送，而不需要每次由客户端想服务器发送请求。 那么，什么是socket？Socket又称套接字，在程序内部提供了与外界通信的端口，即端口通信。通过建立socket连接，可为通信双方的数据传输 传提供通道。socket的主要特点有数据丢失率低，使用简单且易于移植。 1.2Socket的分类 根据不同的的底层协议，Socket的实现是多样化的。本指南中只介绍TCP/IP协议族的内容，在这个协议族当中主要的Socket类型为流套接字 （streamsocket）和数据报套接字(datagramsocket)。流套接字将TCP作为其端对端协议，提供了一个可信赖的字节流服务。数据 报套接字使用UDP协议，提供数据打包发送服务。 二、Socket 基本通信模型 【http://www.itlanbao.com/ns/news.aspx?s=600031】 三、Socket基本实现原理 3.1基于TCP协议的Socket 服务器端首先声明一个ServerSocket对象并且指定端口号，然后调 用Serversocket的accept（）方法接收客户端的数据。accept（）方法在没有数据进行接收的处于堵塞状态。 （Socketsocket=serversocket.accept()）,一旦接收到数据，通过inputstream读取接收的数据。 客户端创建一个Socket对象，指定服务器端的ip地址和端口号 （Socketsocket=newSocket("172.168.10.108",8080);）,通过inputstream读取数据，获取服务器 发出的数据（OutputStreamoutputstream=socket.getOutputStream()），最后将要发送的数据写入到 outputstream即可进行TCP协议的socket数据传输。 3.2基于UDP协议的数据传输 服务器端首先创建一个DatagramSocket对象，并且指点监听的端 口。接下来创建一个空的DatagramSocket对象用于接收数据 （bytedata[]=newbyte[1024;]DatagramSocketpacket=newDatagramSocket（data，data.length））, 使用DatagramSocket的receive方法接收客户端发送的数据，receive（）与serversocket的accepet（）类似， 在没有数据进行接收的处于堵塞状态。 客户端也创建个DatagramSocket对象，并且指点监听的端口。接 下来创建一个InetAddress对象，这个对象类似与一个网络的发送地址 （InetAddressserveraddress=InetAddress.getByName（"172.168.1.120"）） .定义要发送的 一个字符串，创建一个DatagramPacket对象，并制定要讲这个数据报包发送到网络的那个地址以及端口号， 最后使用DatagramSocket 的对象的send（）发送数据。* （Stringstr="hello";bytedata[]=str.getByte(); DatagramPacketpacket=new DatagramPacket(data,data.length,serveraddress,4567);socket.send(packet);） 四、android 实现socket简单通信 4.1使用TCP协议通信 android端实现： protected void connectServerWithTCPSocket() {
Socket socket;
try {// 创建一个Socket对象，并指定服务端的IP及端口号
socket = new Socket("192.168.1.32", 1989);
// 创建一个InputStream用户读取要发送的文件。
InputStream inputStream = new FileInputStream("e://a.txt");
// 获取Socket的OutputStream对象用于发送数据。
OutputStream outputStream = socket.getOutputStream();
// 创建一个byte类型的buffer字节数组，用于存放读取的本地文件
byte buffer[] = new byte[4 \* 1024];
int temp = 0;
// 循环读取文件
while ((temp = inputStream.read(buffer)) != -1) {
// 把数据写入到OuputStream对象中
outputStream.write(buffer, 0, temp);
}
// 发送读取的数据到服务端
outputStream.flush();

 /\*\* 或创建一个报文，使用BufferedWriter写入,看你的需求 \*\*/
// String socketData = "[2143213;21343fjks;213]";
// BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(
// socket.getOutputStream()));
// writer.write(socketData.replace("\n", " ") + "\n");
// writer.flush();
 /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
 } catch (UnknownHostException e) {
 e.printStackTrace();
 } catch (IOException e) {
 e.printStackTrace();
 }
 }
服务器端简单实现：
public void ServerReceviedByTcp() {
 // 声明一个ServerSocket对象
 ServerSocket serverSocket = null;
 try {
 // 创建一个ServerSocket对象，并让这个Socket在1989端口监听
 serverSocket = new ServerSocket(1989);
 // 调用ServerSocket的accept()方法，接受客户端所发送的请求，
 // 如果客户端没有发送数据，那么该线程就停滞不继续
 Socket socket = serverSocket.accept();
 // 从Socket当中得到InputStream对象
 InputStream inputStream = socket.getInputStream();
 byte buffer[] = new byte[1024 \* 4];
 int temp = 0;
 // 从InputStream当中读取客户端所发送的数据
 while ((temp = inputStream.read(buffer)) != -1) {
 System.out.println(new String(buffer, 0, temp));
 }
 serverSocket.close();
 } catch (IOException e) {
 e.printStackTrace();
 }
}
4.2使用UDP协议通信
 客户端发送数据实现：
 protected void connectServerWithUDPSocket() {
 DatagramSocket socket;
 try {
 //创建DatagramSocket对象并指定一个端口号，注意，如果客户端需要接收服务器的返回数据,
 //还需要使用这个端口号来receive，所以一定要记住
 socket = new DatagramSocket(1985);
 //使用InetAddress(Inet4Address).getByName把IP地址转换为网络地址
 InetAddress serverAddress = InetAddress.getByName("192.168.1.32");
 //Inet4Address serverAddress = (Inet4Address) Inet4Address.getByName("192.168.1.32");
 String str = "[2143213;21343fjks;213]";//设置要发送的报文
 byte data[] = str.getBytes();//把字符串str字符串转换为字节数组
 //创建一个DatagramPacket对象，用于发送数据。
 //参数一：要发送的数据 参数二：数据的长度 参数三：服务端的网络地址 参数四：服务器端端口号
 DatagramPacket packet = new DatagramPacket(data, data.length ,serverAddress ,10025);
 socket.send(packet);//把数据发送到服务端。
 } catch (SocketException e) {
 e.printStackTrace();
 } catch (UnknownHostException e) {
 e.printStackTrace();
 } catch (IOException e) {
 e.printStackTrace();
 }
}

客户端接收服务器返回的数据：
public void ReceiveServerSocketData() {
 DatagramSocket socket;
 try {
 //实例化的端口号要和发送时的socket一致，否则收不到data
 socket = new DatagramSocket(1985);
 byte data[] = new byte[4 \* 1024];
 //参数一:要接受的data 参数二：data的长度
 DatagramPacket packet = new DatagramPacket(data, data.length);
 socket.receive(packet);
 //把接收到的data转换为String字符串
 String result = new String(packet.getData(), packet.getOffset(),
 packet.getLength());
 socket.close();//不使用了记得要关闭
 System.out.println("the number of reveived Socket is :" + flag
 + "udpData:" + result);
 } catch (SocketException e) {
 e.printStackTrace();
 } catch (IOException e) {
 e.printStackTrace();
 }
}
服务器接收客户端实现：
public void ServerReceviedByUdp(){
 //创建一个DatagramSocket对象，并指定监听端口。（UDP使用DatagramSocket）
 DatagramSocket socket;
 try {
 socket = new DatagramSocket(10025);
 //创建一个byte类型的数组，用于存放接收到得数据
 byte data[] = new byte[4\*1024];
 //创建一个DatagramPacket对象，并指定DatagramPacket对象的大小
 DatagramPacket packet = new DatagramPacket(data,data.length);
 //读取接收到得数据
 socket.receive(packet);
 //把客户端发送的数据转换为字符串。
 //使用三个参数的String方法。参数一：数据包 参数二：起始位置 参数三：数据包长
 String result = new String(packet.getData(),packet.getOffset() ,packet.getLength());
 } catch (SocketException e) {
 e.printStackTrace();
 } catch (IOException e) {
 e.printStackTrace();
 }
}
五、总结：
使用UDP方式android端和服务器端接收可以看出，其实android端和服务器端的发送和接收大庭相径，只要端口号正确了，相互通信就没有问题，TCP使用的是流的方式发送，UDP是以包的形式发送。
Android操作HTTP实现与服务器通信 【http://www.cnblogs.com/hanyonglu/archive/2012/02/19/2357842.html】

# Android四大组件

http://www.cnblogs.com/pepcod/archive/2013/02/11/2937403.html # 三级缓存 http://blog.saymagic.cn/2015/01/30/android-pic-three-cache.html # 图片的处理和优化 http://www.cnblogs.com/elliotta/p/3633752.html http://blog.csdn.net/yudajun/article/details/9323941 # Android5.0新特性 技术方面说明 http://blog.csdn.net/lwyygydx/article/details/41870377 功能改进方面说明 http://digi.tech.qq.com/a/20150121/012030.htm # 图文混排 http://blog.csdn.net/fancylovejava/article/details/39927539 # 第三方框架:xUtils,Gson 极光推送 第三方登录 友盟第三方登录 http://blog.umeng.com/uncategorized/4160.html 第三方登录案例 http://blog.csdn.net/yueqinglkong/article/details/15028041 # 线程池 http://blog.csdn.net/lyf\_007217/article/details/8542238 http://www.cnblogs.com/devinzhang/p/3856200.html # lru算法底层 http://www.360doc.com/content/14/0402/09/10504424\_365635496.shtml http://blog.csdn.net/androidzhaoxiaogang/article/details/7910364 # ListView的局部刷新 http://www.2cto.com/kf/201409/335964.html http://blog.csdn.net/u200814499/article/details/40391443 # 及时通讯 http://blog.csdn.net/jiangliloveyou/article/details/9849775 http://blog.csdn.net/lnb333666/article/details/7471292 http://skywen.iteye.com/blog/1811310 # GC原理 http://blog.csdn.net/wuqiong\_524itcast/article/details/25378685 http://blog.csdn.net/wangshione/article/details/8490245 http://blog.csdn.net/lnb333666/article/details/8031770 1.垃圾收集算法的核心思想 Java语言建立了垃圾收集机制，用以跟踪正在使用的对象和发现并回收不再使用(引用)的对象。该机制可以有效防范动态内存分配中因内存垃圾过多而引发的内存耗尽，以及不恰当的内存释放所造成的内存非法引用。 　垃圾收集算法的核心思想是：对虚拟机可用内存空间，即堆空间中的对象进行识别，如果对象正在被引用，那么称其为存活对象，反之，如果对象不再被引用，则 为垃圾对象，可以回收其占据的空间，用于再分配。垃圾收集算法的选择和垃圾收集系统参数的合理调节直接影响着系统性能，因此需要开发人员做比较深入的了解。 触发主GC(Garbage Collector)的条件　JVM进行次GC的频率很高,但因为这种GC占用时间极短,所以对系统产生的影响不大。更值得关注的是主GC的触发条件,因为它对系统影响很明显。总的来说,有两个条件会触发主GC: 　　①当应用程序空闲时,即没有应用线程在运行时,GC会被调用。因为GC在优先级最低的线程中进行,所以当应用忙时,GC线程就不会被调用,但以下条件除外。 　　②Java堆内存不足时,GC会被调用。当应用线程在运行,并在运行过程中创建新对象,若这时内存空间不足,JVM就会强制地调用GC线程,以 便回收内存用于新的分配。若GC一次之后仍不能满足内存分配的要求,JVM会再进行两次GC作进一步的尝试,若仍无法满足要求,则 JVM将报“out of memory”的错误,Java应用将停止。 3.减少GC开销的措施 根据上述GC的机制,程序的运行会直接影响系统环境的变化,从而影响GC的触发。若不针对GC的特点进行设计和编码,就会出现内存驻留等一系列负面影响。为了避免这些影响,基本的原则就是尽可能地减少垃圾和减少GC过程中的开销。具体措施包括以下几个方面: 　　(1)不要显式调用System.gc() 　　此函数建议JVM进行主GC,虽然只是建议而非一定,但很多情况下它会触发主GC,从而增加主GC的频率,也即增加了间歇性停顿的次数。 　　(2)尽量减少临时对象的使用 　　临时对象在跳出函数调用后,会成为垃圾,少用临时变量就相当于减少了垃圾的产生,从而延长了出现上述第二个触发条件出现的时间,减少了主GC的机会。 　　(3)对象不用时最好显式置为Null 　　一般而言,为Null的对象都会被作为垃圾处理,所以将不用的对象显式地设为Null,有利于GC收集器判定垃圾,从而提高了GC的效率。 　　(4)尽量使用StringBuffer,而不用String来累加字符串(详见blog另一篇文章JAVA中String与StringBuffer) 　　由于String是固定长的字符串对象,累加String对象时,并非在一个String对象中扩增,而是重新创建新的String对象,如 Str5=Str1+Str2+Str3+Str4,这条语句执行过程中会产生多个垃圾对象,因为对次作“+”操作时都必须创建新的String对象,但 这些过渡对象对系统来说是没有实际意义的,只会增加更多的垃圾。避免这种情况可以改用StringBuffer来累加字符串,因StringBuffer 是可变长的,它在原有基础上进行扩增,不会产生中间对象。 　　(5)能用基本类型如Int,Long,就不用Integer,Long对象 　　基本类型变量占用的内存资源比相应对象占用的少得多,如果没有必要,最好使用基本变量。 　　(6)尽量少用静态对象变量 　　静态变量属于全局变量,不会被GC回收,它们会一直占用内存。 　　(7)分散对象创建或删除的时间 　　集中在短时间内大量创建新对象,特别是大对象,会导致突然需要大量内存,JVM在面临这种情况时,只能进行主GC,以回收内存或整合内存碎片, 从而增加主GC的频率。集中删除对象,道理也是一样的。它使得突然出现了大量的垃圾对象,空闲空间必然减少,从而大大增加了下一次创建新对象时强制主GC 的机会。
gc()函数的作用只是提醒虚拟机：程序员希望进行一次垃圾回收。但是它不能保证垃圾回收一定会进行，而且具体什么时候进行是取决于具体的虚拟机的，不同的虚拟机有不同的对策。在Davilk中，给程序分配的内存是根据机型厂商的不同而不同(现在大部分为32MB),在VM内部会将内存分为：java使用的内存，Native使用的内存，他们之间不能共享，当某一方面不足的时候必须向VM申请，而不能直接使用另外一个的内存。 # 出现内存泄漏的可能性： 出现情况: 1. 数据库的cursor没有关闭 2.构造adapter时,没有使用缓存contentview 衍生listview的优化问题-----减少创建view的对象,充分使用contentview,可以使用一静态类来优化处理getview的过程 3.Bitmap对象不使用时采用recycle()释放内存 4.activity中的对象的生命周期大于activity 调试方法: DDMS==> HEAPSZIE==>dataobject==>[Total Size] # Android 内存浅析【管理、机制、分析】 一、 Android的内存机制 Android的程序由Java语言编写，所以Android的内存管理与Java的内存管理相似。程序员通过new为对象分配内存，所有对象在java 堆内分配空间；然而对象的释放是由垃圾回收器来完成的。C／C++中的内存机制是“谁污染，谁治理”，java的就比较人性化了，给我们请了一个专门的清 洁工（GC） 二、GC是什么? 为什么要有GC? 　　 GC是垃圾收集的意思（Gabage Collection）,内存处理是编程人员容易出现问题的地方，忘记或者错误的内存回收会导致程序或系统的不稳定甚至崩溃，Java提供的GC功能可以 自动监测对象是否超过作用域从而达到自动回收内存的目的，Java语言没有提供释放已分配内存的显示操作方法。 三、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。间而忘记了释放。如果程序中存在对无用对象的引用，那么这些对象就会驻留内存，消耗内存，因为无法让垃圾回收器GC验证这些对象是否不再需要。如果存在对象的引用，这个对象就被定义为"有效的活动"，同时不会被释放。要确定对象所占内存将被回收，我们就要务必确认该对象不再会被使用。典型的做法就是把对象数据成员设为null或者从集合中移除该对象。但当局部变量不需要时，不需明显的设为null，因为一个方法执行完毕时，这些引用会自动被清理。

Vector v = new Vector(10);
 for (int i = 1; i < 100; i++) {
 Object o = new Object();
 v.add(o);
 o = null;
 }// 此时，所有的Object对象都没有被释放，因为变量v引用这些对象。
Java 内存泄露的根本原因就是 保存了不可能再被访问的变量类型的引用

六、Android的内存溢出 Android的内存溢出是如何发生的? Android的虚拟机是基于寄存器的Dalvik，它的最大堆大小一般是16M，有的机器为24M。也就是说我们所能利用的内存空间是有限的。如果我们的内存占用超过了一定的水平就会出现OutOfMemory的错误。 为什么会出现内存不够用的情况呢？我想原因主要有两个： 由于我们程序的失误，长期保持某些资源（如Context）的引用，造成内存泄露，资源造成得不到释放。保存了多个耗用内存过大的对象（如Bitmap），造成内存超出限制。 # 在Android适配方案小结 600dp的含义是：代表这个设备的最短的那一边。 获取设备的最短边的代码是：Configuration config = getResources().getConfiguration(); int smallestScreenWidth = config.smallestScreenWidthDp; 这个时候拿smallestScreenWidth 与600想比较就可以知道该设备能否读取里面的资源了。 除此之外，为了方便适配，在编码时我们还应该注意什么呢，主要有以下几点： 除此之外，为了方便适配，在编码时我们还应该注意什么呢，主要有以下几点：

（1）多使用权重(android:layout\_weight)
尤其是在tab切换布局，listview title及Item布局等情况下；
（2）设置宽度和高度时，尽量使用match\_parent和wrap\_content，避免把控件宽高设死；
（3）父容器布局选用
多使用RelativeLayout，FrameLayout，GridLayout等，减少布局层次。当然，在使用
权重时，得采用LinearLayout；
(4) 在xml里，设置高度、宽度采用dp(dip)，设置字体采用sp。
（应该注意，在代码里面，我们写的setHeight(...)单位是px）
那么在具体开发中，我们应该注意什么呢。
首先，我们必须要知道，其实适配的关键在于两点：
（1）不同分辨率设备的适配，这点在单位的使用上用dp、sp以及图片资源存放于不同的drawable文件夹就可以解决问题；
（2）不同尺寸的适配，这点主要靠将相关值以及布局文件放置于不同的文件夹中来解决。
2.1 values文件夹
可以在工程下创建不同的values文件夹：values-sw480dp， values-sw600dp，
values-sw720dp-land等。比如一个控件的宽度，在10寸pad上是10dp，在8寸pad
上是5dp。这时，你可以定义一个变量，button\_width，然后在values-sw600dp
下写5dp，在values-sw720-land下写
10dp。这样就达到了在不同尺寸pad上，
相应控件大小不一样的效果。
2.1 layout文件夹
如果在不同尺寸设备上展示的布局有明显差别，仅仅用values不同已经难以控制，
那么就可以考虑写不同的布局文件置于不同的layout文件夹下，android会根据设备
尺寸去加载相应文件夹下的布局文件。如:layout-sw480dp，layout-sw600dp，
layout-sw700dp等。
值得注意的是，如果不是很有必要，尽量采用2.1方案，方便维护。如果尺寸和分辨率都不同，
那么就要结合（1）、（2）考虑了。
（补充：其实values文件夹和layout文件夹不仅仅是根据尺寸判断，也和分辨率有关，不过在通常情况下，
综合计算考虑，仅根据尺寸判断就可以了：

# Java 基础

1：
int a = 1;
int m1 = ++a +3;
结果 ：m1 = 5；a=2;
++a表示先赋值
2:
int a = 1;
int m = a+++3;
结果 m = 4; a= 2;
a++表示后赋值
3：
 m<<2 表示 m\*2\*2
 m<< 3 表示 m\*2\*2\*2
 int result =5<<2 ;//a 20 5\*2\*2 ;
 int result1 =6<<3 ;// 48 6\*2\*2\*2
 int result2 =7<<4 ;//112 7\*2\*2\*2\*2
 4:
a++ 表示a+1
int a = 2;
int result = (a++ > 2)?(++a):(a+=3);
结果是//6
5. 下面程序的运行结果是（）
String str1 = "hello";String str2 = "he" + new String("llo");System.err.println(str1 == str2);
答案：false
解析：因为str2中的llo是新申请的内存块，而==判断的是对象的地址而非值，所以不一样。如果是String str2 .equals(str1)，那么就是true了。
4. 下列说法正确的有（）
A． class中的constructor不可省略
B． constructor必须与class同名，但方法不能与class同名
C． constructor在一个对象被new时执行
D．一个class只能定义一个constructor
答案：C
解析：这里可能会有误区，其实普通的类方法是可以和类名同名的，和构造方法唯一的区分就是，构造方法没有返回值。

下面程序的运行结果：（）
public static void main(String args[]) { Thread t = new Thread() { public void run() { pong(); } };
 t.run(); System.out.print("ping"); } static void pong() { System.out.print("pong"); }
A pingpong B pongping C pingpong和pongping都有可能 D 都不输出
答案：B
解 析：这里考的是Thread类中start()和run()方法的区别了。start()用来启动一个线程，当调用start方法后，系统才会开启一个新 的线程，进而调用run()方法来执行任务，
而单独的调用run()就跟调用普通方法是一样的，已经失去线程的特性了。因此在启动一个线程的时候一定要使 用start()而不是run()。
7. 下列属于关系型数据库的是（）
A. Oracle B MySql C IMS D MongoDB
答案：AB
解答：IMS（Information Management System ）数据库是IBM公司开发的两种数据库类型之一;
一种是关系数据库，典型代表产品：DB2；
另一种则是层次数据库，代表产品：IMS层次数据库。
非关系型数据库有MongoDB、memcachedb、Redis等。
8. GC线程是否为守护线程？（）
答案：是
解析：线程分为守护线程和非守护线程（即用户线程）。
只要当前JVM实例中尚存在任何一个非守护线程没有结束，守护线程就全部工作；只有当最后一个非守护线程结束时，守护线程随着JVM一同结束工作。
守护线程最典型的应用就是 GC (垃圾回收器)
9. volatile关键字是否能保证线程安全？（）
答案：不能
解析：volatile关键字用在多线程同步中，可保证读取的可见性，JVM只是保证从主内存加载到线程工作内存的值是最新的读取值，而非cache中。但多个线程对
volatile的写操作，无法保证线程安全。例 如假如线程1，线程2 在进行read,load 操作中，发现主内存中count的值都是5，那么都会加载这个最新的值，在线程1堆count进行修改之后，
会write到主内存中，主内存中的 count变量就会变为6；线程2由于已经进行read,load操作，在进行运算之后，也会更新主内存count的变量值为6；导致两个线程及时用 volatile关键字修改之后，还是会存在并发的情况。
10. 下列说法正确的是（AC）
A LinkedList继承自List
B AbstractSet继承自Set
C HashSet继承自AbstractSet
D WeakMap继承自HashMap
解析：下面是一张下载的Java中的集合类型的继承关系图，一目了然。 http://www.itlanbao.com/ns/news.aspx?s=600034
11. 存在使i + 1 < i的数吗（）
 答案：存在
 解析：如果i为int型，那么当i为int能表示的最大整数时，i+1就溢出变成负数了，此时不就<i了吗。
 扩展：存在使i > j || i <= j不成立的数吗（）
 答案：存在
 解析：比如Double.NaN或Float.NaN，
 12. 0.6332的数据类型是（）
 A float B double C Float D Double
 答案：B
 解析：默认为double型，如果为float型需要加上f显示说明，即0.6332f、
 13. 下面哪个流类属于面向字符的输入流( )
 A BufferedWriter B FileInputStream C ObjectInputStream D InputStreamReader
 答案：D
 解析：Java的IO操作中有面向字节(Byte)和面向字符(Character)两种方式。
 面向字节的操作为以8位为单位对二进制的数据进行操作，对数据不进行转换，这些类都是InputStream和OutputStream的子类。
 面向字符的操作为以字符为单位对数据进行操作，在读的时候将二进制数据转为字符，在写的时候将字符转为二进制数据，这些类都是Reader和Writer的子类。
 总结：以InputStream（输入）/OutputStream（输出）为后缀的是字节流；
 以Reader（输入）/Writer（输出）为后缀的是字符流。

下面程序能正常运行吗（）
public class NULL { public static void haha(){ System.out.println("haha"); } public static void main(String[] args) { ((NULL)null).haha(); } }
答案：能正常运行
解析：输出为haha，因为null值可以强制转 换为任何java类类型,(String)null也是合法的。但null强制转换后是无效对象，其返回值还是为null，而static方法的调用是和 类名绑定的，
不借助对象进行访问所以能正确输出。反过来，没有static修饰就只能用对象进行访问，使用null调用对象肯定会报空指针错了。这里和 C++很类似。

下面程序的运行结果是什么（）
 class HelloA { public HelloA() { System.out.println("HelloA"); } { System.out.println("I'm A class"); }
static { System.out.println("static A"); } }public class HelloB extends HelloA { public HelloB() { System.out.println("HelloB"); }
{ System.out.println("I'm B class"); }
static { System.out.println("static B"); } public static void main(String[] args) { 　　　　 new HelloB(); 　　 } }
答案：
static Astatic BI'm A classHelloAI'm B classHelloB
解析：说实话我觉得这题很好，考查静态语句块、构造语句块（就是只有大括号的那块）以及构造函数的执行顺序。
对象的初始化顺序：（1）类加载之后，按从上到下（从父类到子类）执行被static修饰的语句；（2）当static语句执行完之后,再执行main方法；
（3）如果有语句new了自身的对象，将从上到下执行构造代码块、构造器（两者可以说绑定在一起）。
下面稍微修改下上面的代码，以便更清晰的说明情况：
class HelloA { public HelloA() { System.out.println("HelloA"); } { System.out.println("I'm A class"); }
static { System.out.println("static A"); } }public class HelloB extends HelloA { public HelloB() { System.out.println("HelloB"); }
 { System.out.println("I'm B class"); } static { System.out.println("static B"); } public static void main(String[] args) { System.out.println("-------main start-------");
 new HelloB(); new HelloB(); System.out.println("-------main end-------"); }}
static Astatic B-------main start-------I'm A classHelloAI'm B classHelloBI'm A classHelloAI'm B classHelloB-------main end-------

Java7中的switch支持String的实现细节
在Java7之前，switch只能支持 byte、short、char、int或者其对应的封装类以及Enum类型。在Java7中，呼吁很久的String支持也终于被加上了。
public class Test { public void test(String str) { switch(str) { case "abc": System.out.println("abc"); break; case "def":
System.out.println("def"); break; default: System.out.println("default"); } } }

5：
16进制数必须以 0x开头
&是位操作符，“按位与”
1转成二进制 01
2转成二进制 10
与运算符
与运算符用符号“&”表示，其使用规律如下：
两个操作数中位都为1，结果才为1，否则结果为0，例如下面的程序段。
public class data13
{
public static void main(String[] args)
{
int a=129;
int b=128;
System.out.println("a 和b 与的结果是："+(a&b));
}
}
运行结果
a 和b 与的结果是：128
下面分析这个程序：
“a”的值是129，转换成二进制就是10000001，而“b”的值是128，转换成二进制就是10000000。根据与运算符的运算规律，只有两个位都是1，结果才是1，可以知道结果就是10000000，即128。 2\*2\*2\*2\*2\*2\*2
int a = 1234566;
查询a的二进制Integer.toBinaryString(a);

# 学生面试被问到的问题总结

## 1. 网络传输数据如何加密，比如账户密码，视频？ 可以这么回答： 进行安全保证的方式有很多种，如果进行简单的加密可以使用MD5或者DES，但是这些都是相对的， 如果在开发安全性较高的应用时，可以考虑模仿HTTP协议那样，自定义一个协议， 然后封装一下，在协议里使用时间戳+算法加密技术提高安全系数. Android网络传输中必用的两个加密算法:MD5 和 RSA 答案参考：http://blog.csdn.net/yanzi1225627/article/details/26508035 ## 2. 支付功能如何实现？ 回答： 目前主流的支付有三大，微信支付，支付宝支付，第三方银联支付。 如果是我，我的回答是，我做android的目前只是用到第三方开放平台来实现接入这些支付功能， 但是我不知道这些第三方支付功能具体怎么实现的，这个我真不知道，没研究。 如果是接入第三方支付功能的话就比较简单了：参考文章http://blog.163.com/benben\_long/blog/static/19945824320142279427395/ 支付宝集成： 注意事项 1.添加android.permission.INTERNET权限和android.permission.ACCESS\_NETWORK\_STATE权限 2.代码中出现注释的地方重点看，没注释的地方可以不看 3.想获取支付宝合作商户ID，及支付宝公钥请点击支付宝链接，生成密钥及PKCS8转码工具在文档中 微信支付集成注意：参考：http://blog.csdn.net/jdsjlzx/article/details/47422279 1.在你的项目测试微信的组件（分享、支付等）的时候，一定要用你自己的keystore签名出来测试， 如果用debug.keystore肯定是不成功的， 2.支付成功通知：在WXPayEntryActivity的OnResp中处理，不能以微信返回的通知界面为准 （我遇到的情况，网络不稳定的时候，微信返回界面提示支付失败，但是收到微信通知其实已经支付成功了）， 必须要去自己的服务器查询支付状态，这里微信建议用轮循机制去查询 @Override public void onResp(BaseResp resp) { Log.d(TAG, "onPayFinish, errCode = " + resp.errCode); if (resp.getType() == ConstantsAPI.COMMAND\_PAY\_BY\_WX) { AlertDialog.Builder builder = new AlertDialog.Builder(this); builder.setTitle(R.string.app\_tip); builder.setMessage(getString(R.string.pay\_result\_callback\_msg, resp.errStr +";code=" + String.valueOf(resp.errCode))); builder.show(); } } ## 3. 在开发中你都遇到了哪些难题，如何解决的？ 如何降低apk包的大小， apk大小，跟你工程文件大小有直接关系，其中关系最为密切的还是你的图片多少，图片上能用.9处理的图片尽量用。 如果你一个apk需要适配多种手机，那么你最好只搞一套图片，另外图片的大小也应该做适当压缩， 保证图片显示质量的前提下，尽量优化图片的存储空间 。 1 删除项目中没有到的文件，包括资源文件，java代码等等 2 避免jar包的重复引用 3 可以考虑压缩一下图片，前提是尽量保持图片别失真。

 ## 4.android 适配
 适配也是一个头疼的事，去年年底我们就开始正对720为主流做适配了，详细适配 百度

还有有的面试官会问你，你们设计师出的图时根据ios的出还是android的出图 有几种回答， （1）直接根据android的出图直接做一套720*1280 （2）根据ios出图 众所周知iOS设计的像素尺寸是640*960/1136，Android主流的hdpi模式下的像素尺寸是480\*800。如图， 他们的换算关系是，iOS像素尺寸的75%是Android的像素尺寸

大概算法，大家可以看看：

其实经过以上整个过程之后，我们已经得出了一个更简单的换算关系：iOS像素尺寸*75%=Android像素尺寸， Android像素尺寸*2/3=Android的dp尺寸。进而得出：iOS像素尺寸*75%*2/3=Android的dp尺寸。 所以，iOS里一个宽600px的东西，在Android的hdpi模式下，正好300dp，正好是50%，很容易算是吧？

 简单说一下：
一、关于布局适配
1、不要使用绝对布局
2、尽量使用match\_parent 而不是fill\_parent 。
3、能够使用权重的地方尽量使用权重（android:layout\_weight）
4、如果是纯色背景，尽量使用android的shape 自定义。
5、如果需要在特定分辨率下适配，可以在res目录上新建layout-HxW.xml的文件夹。比如要适配1080\*1800的屏幕
（魅族MX3采用此分辨率）则新建layout-1800x1080.xml的文件夹，然后在下面定义布局。Android系统会优先查找分
辨率相同的布局，如果不存在则换使用默认的layout下的布局。

## 5.一个apk如何快速方便的打多个不同包名的产品（多渠道多产品推广）
我们市场在推广apk的时候有时候需要根据渠道打不同包名的apk并且这些打出来的apk风格和内容展示以及文字展现略有不同。

我们现在的做法是，把主工程项目当做libs形式关联到想要打包的工程，这样打不同包名的时候就方便，直接创建一个工程， 把主工程关联，然后可以在新创建的工程里面略修改一些比如title风格，首页面进入风格 （因为首页我们做了好几套可以根据类型来判断你走的是哪一个风格），就是一个新的apk出现了。 ## 6.如何在webview中实现点击事件的监听处理？ http://blog.csdn.net/zzf112/article/details/19618101 ## 1、联网请求的时候HTTP协议的哪个部分耗时比较多，导致APP运行缓慢，该怎么优化 这个问题问得应该有问题，个人觉得你需要把http协议原理给他理清楚，这个问题都是与网络快慢有关的，在与服务器交互的时候尽量减少数据量， 这篇文章不错：blog.csdn.net/lmh12506/article/details/7794512 http://www.cnblogs.com/jdsjlzx/archive/2011/07/25/2116351.html ## 2、集成环信的及时通讯SDK如果遇到消息遗漏或者消息重复该怎么解决 ## 3、如何实现上传和离线上传 Android离线数据同步方案 参考文章： http://wenku.baidu.com/link?url=3SvxuKV03wXR6LbjJYmXtrtiX7jPehmDTQRklcf\_oXRX2FKoP2RzZVFp0Obl8cjZQED3en8orizKI9wFrYkdx3-izxjN8H2gjcpsiUXa98G 略熟悉第三方sdk: Android 版 SugarSync 加入更多离线功能 WebView实现离线缓存阅读 参考文章：http://blog.csdn.net/wwj\_748/article/details/44835865 ArcGIS for Android离线数据编辑实现原理 http://blog.csdn.net/arcgis\_mobile/article/details/7565877 ## 4、文件的加密 文件加密AES加密算法 AES加密算法是目前比较流行加密方式，目前还没有针对AES有效的破解方式，比较靠谱。 AES加密数据块和密钥长度可以是128比特、192比特、256比特中的任意一个。 AES加密有很多轮的重复和变换。大致步骤如下： 1、密钥扩展（KeyExpansion）， 2、初始轮（Initial Round）， 3、重复轮（Rounds），每一轮又包括：SubBytes、ShiftRows、MixColumns、AddRoundKey， 4、最终轮（Final Round），最终轮没有MixColumns。 我以前对文件加密的时候就是参考如下文章 请参考：http://blog.csdn.net/yudajun/article/details/40481135 http://blog.csdn.net/dalancon/article/details/20924823 ## （一）：Android卸载程序之后如何跳转到指定的反馈页面 比如：360被卸载之后会跳转到指定的反馈页面如何实现？ 本题解析：本题目的回答需要从C层出发，不过java层也需要接受一些android BroadcastReceiver机制， 以及BroadcastReceiver无法实现原因， 回答： 参考文章http://blog.csdn.net/jiangwei0910410003/article/details/42177117

## （二）：FragmentManager内部如何维护fragment队列，以及fragment事务的回退栈实现原理

本题解析：回答本题需要从以下几点去出发
1，fragment的生命周期
2，FragmentManager的作用，以及如何维护fragment队列
3，如何管理Fragment回退栈和回退栈实现原理
回答：
fragment的生命周期 参考：http://blog.csdn.net/t12x3456/article/details/8104531
FragmentManager的作用，以及如何维护fragment队列
参考：http://longshuai2007.blog.163.com/blog/static/142094414201362631129902/
http://www.cnblogs.com/mybkn/articles/2455138.html
http://www.mamicode.com/info-detail-612467.html

## （三）：如何保证后台Service不被杀掉
 本题解析：先大体介绍一下android的Service以及他的生命周期，其二 介绍出现哪些手机出现service被杀掉的问题，
比如红米手机，service运行一段时间后很容易就被杀掉问题，然后你如何解决Service不被杀掉的方法。
回答：参考 http://blog.csdn.net/mad1989/article/details/22492519