作者：肥肥鱼  
链接：https://www.zhihu.com/question/27227425/answer/35747000  
来源：知乎  
著作权归作者所有，转载请联系作者获得授权。

不请自来说点废话。肥肥做了四年 Android 开发，其中三年时间是在做各种中间件。  
我觉得首先是命名规范。命名规范这种东西每个人都有自己的风格，Google 也有自己的一套规范（多看看 Android 系统源码就明白了）。好的规范可以有效地提高代码的可读性，对于将来接手代码的小伙伴也是一件幸事。题主可以自行 Google 一下 Java （Android）命名规范，会由不少的博客介绍。  
  
其次是注释。严格来说这个应该属于命名规范的范畴。注释一方面是帮助自己记忆 ，另一方面是团队协作中的一个规范，特别是对于开发 API 的小伙伴来说，总不能天天被人跟在屁股后面问你这个接口是什么作用，你这参数是什么意思？好的注释配合好的命名规范，可以省去很多沟通上的成本。 注释至少要有如下几方面的内容：

1. 该接口（或类）的作用。注意写的应该是作用，而不是你做了什么；
2. 参数列表的各个参数说明；
3. 返回值的说明；
4. 如果有异常抛出，对抛出异常的说明；
5. 如果注释是在类上的，总得留个联系方式吧，免得以后出了问题都找不到原作者。当然了，如果类设计的过于让人蛋疼，我也可以联系到作者，约出来，打一顿的嘛。
6. 其他你认为有必要解释。

再者是版本控制。就算是自己一个人写代码，版本控制也是有必要的。Git 也好，SVN 也好，都是有帮助的。版本控制一方面是对自己代码的一个备份，另一方面，如果想回滚到历史版本也是极有帮助的。所以最好能够熟悉一下 Git 或者 SVN 的使用。  
  
第四是名词表。这个应该属于肥肥自创。Android 是围绕四大组件特别是 Activity 和 Service 进行开发的，但是如果项目庞大，有多个 Activity 存在，那么记住每一个 Activity 的类名是很难得，但是记住每一个 Activity 的功能却相对容易。故而肥肥在带项目的过程中自创了名词表这一东东。记录了每一个模块、组件、甚至是每一个 Activity 的官方统一名称（比如，功能是作品列表的 Activity，名称就叫作品列表页，对应的类是 WorksListActivity），在沟通过程中，大家（包括测试人员等项目相关人员）统一说“作品列表页”。当时的初衷是解决测试团队的Bug 描述过于模糊（如果有多个列表页，测试人员往往会写列表页\*\*\*\*问题）。  
  
第五是不要重复制造轮子。这个适用于代码层面以及业务层面。  
  
第六是千万不要饿着肚子写代码。先挖这个坑，吃了东西再找时间继续写。  
=========================刚才公司小伙伴在互换礼物，来晚了===================  
第七这一点应该不算是习惯，但是却需要每一个 Android 程序猿需要慎重再者慎重的地方---内存管理。Android 虽然延续了 Java 的垃圾回收机制，但是并不意味 Android 应用程序就不会出现内存问题。在 Android 中引起内存开销过大的往往是 BitMap 对象。BitMap.java实际上是 Skia 引擎中对图片处理部分的 Java 层代码而已（真正工作的是 C++层代码，通过 JNI 封装，最后提供 Java 层的接口），那么你创建 BitMap 对象实际上是创建了两部分内存，一部分是 Java 层的，就是 BitMap对象，Java 的垃圾回收会在合适的时机回收这一部分内存。另一部分是 C++层面的，也就是通过 JNI 调用 C++层的代码分配的那一部分内存。Java 的垃圾回收是不会回收这一部分内存的，所以如果不手动释放的话就容易引起内存问题。  
  
第八是千万不要阻塞用户主线程。用户主线程就是 UI 线程，主要负责 UI 的绘制（除 SurfaceView 外，其他 View 对象都是需要在 UI 线程中进程操作的）。为了保证 App 的交互尽可能的流程，请不要在 UI 线程中进行耗时操作（文件读写、Http 请求（4.0之前可以在主线程中发起）等）。否则会引起两种可能的问题：第一是造成用户交互极度不流畅，第二容易触发 ANR 的超时机制（UI 线程5秒，广播10秒）。  
  
第九是严格把控生命周期（Activity、Service、ContentProvider 等）。在每一个生命周期事件中，明确应该做什么不应该做什么是很有必要的，不然也会容易造成各种莫名其妙的问题（比如 onCreate 中使用了 onResume 中才初始化的对象）。  
  
第十是在使用 XML 文件进行 UI 布局时，应该尽量减少 Layout 的嵌套层级。Layout 的过度嵌套会造成渲染时资源开销过大的问题。  
  
写到这里突然不想写了。由于右手受伤的缘故，今天高强度的键盘敲击已经造成右手非常疲惫。等有时间的时候再写吧。  
  
  
==============2014年12月26日=============  
第一次在知乎上的得到这么多赞（竟然有24了），受宠若惊，决定继续补充前面的各种废话。  
  
第十一应该是资源文件的使用，资源文件包括图片、字符串、尺寸、颜色等等。在使用尺寸资源的时候应该尽量使用像素无关的单位，比如 dp 和 sp。而字符串资源（比如 Button 上显示的名称）也应该尽可能的抽离出来，使用 res/value 下的xml 文件进行维护。一方面方便日后管理，另一方面方便国际化。  
  
第十二多线程以及线程池的使用。前面说过应该尽量避免在主线程中执行耗时操作，那么多线程就变得很有必要。对于 Java 来说，线程的创建与销毁是非常占用资源的，线程的滥用（随手 new Thread 等）会造成 App 整体性能的下降。Java 提供了Executors的线程池方案，而 Android 自身也提供了AsyncTask 这样的异步任务方案（实际上也是线程池）。  
  
第十三 Java 的权限控制机制。Java提供了public, private, protected 三个访问权限修饰词，提供了以下四种访问权限控制机制：

1. 包访问权限；
2. Public访问权限；
3. Private访问权限；
4. Protected访问权限；

访问权限的合理使用可以有效地隐藏实现，避免将不必要的数据或接口暴露出来。  
  
第十四 final 和 static 关键字的合理使用。很多人觉得这是很基础的东西，但是 final 和 static 关键字的合理使用能够有效提升代码的执行效率，而不合理使用则后患无穷。  
  
第十五 Android 设备的内存资源是极度珍贵的，合理的使用、回收内存也是一种好的编程习惯。Java 对象的引用类型会影响到垃圾回收对象的时机。Java 有强引用、 软引用、 弱引用、虚引用，以及 Android 增加的 Lru 内存管理。建议题主了解一下这四种引用类型的特点以及 Lru 内存管理的具体实现。  
  
第十六 接口和抽象类。这是老生常谈的话题了，但却是永恒的话题。接口和抽象类的合理使用，可以增加代码的可维护性和扩展性。接口和抽象类也是各种设计模式的基石。  
  
第十七 软件设计的六大设计原则，即

1. 针对接口编程，不针对实现编程
2. 单一职责原则
3. 开放封闭原则
4. 里氏代换原则
5. 迪米特法则
6. 合成聚合复用原则

第十八 统一项目的编码格式（推荐使用 UTF-8）。如果多人协作，这种举措显得尤为重要。由此引申出来的另外一个规范就是，规范统一命名风格，即团队中使用相同的命名风格。（感谢@[徐强](/people/xu-qiang-45-74) 的提醒。更新于2014年12月30日17:22:20）  
  
  
第十九 TextView（往往 TextView 派生子类同样适用）调用 setText 方法设置一个 int 型的数据，千万要将该值转为 String，否则在某些设备中它会默认去查询 R 文件中定义的资源  
  
第二十 使用友盟分享 SDK，需要执行分享的 Activity 请不要为该 Activity 设置android:process属性。比如你的 App 运行在 com.codingfish.test 进程，需要产生分享动作的Activity 设置 android:proces=":com.codingfish.hello" ，那么新浪微博就会出现你设置的分享内容没有显示的问题。该 Bug 已经提交给友盟的技术人员，但是 N 久没有得到修复。  
  
第二十一 上线之前一定要使用正式签名打包。某朋友公司 Android 的应用上架之前，负责打包上线的童鞋（新人，老人已离职，只有这一个Android）没有签名的概念，直接将 Debug 签名的 Apk 投放到渠道了，到现在还有一批设备没有替换回来。  
  
第二十二 在 Activity 中尽可能少的创建 Handler 对象，创建一个主线程 Handler，一个后台 HandlerThread 就可以了。  
  
第二十三 使用线程的地方尽量不要 new Thread，而是使用 AsyncThread 。  
  
第二十四 onCreate(Bundle savedInstanceState) 切记将super.onCreate(savedInstanceState);放在一切业务的前面。  
  
第二十五 创建了四大组件一定记得要在 AndroidManifest 文件中声明（当然 BroadcastReceiver 可以动态注册）。  
  
=============2015年01月09日补充==============  
  
刚才 Google 了一下，有一篇文章介绍的不错，肥肥做了一次伸手党，直接复制过来了。  
原文链接： [Android生存指南之:开发中的注意事项\_Android\_脚本之家](//link.zhihu.com/?target=http%3A//www.jb51.net/article/37127.htm)  
  
**第二十六 为Activity声明系统配置变更事件** 系统配置变更事件是指转屏，区域语言发生变化，屏幕尺寸发生变化等等，如果Activity没有声明处理这些事件，发生事件时，系统会把Activity杀掉然后重启，并尝试恢复状态，Activity有机会通过onSaveInstanceState()保存一些基本数据到Bundle中，然后此Bundle会在Activity的onCreate()中传递过去。虽然这貌似正常，但是这会引发问题，因为很多其他的东西比如Dialog等是要依赖于具体Activity实例的。所以这种系统默认行为通常都不是我们想要的。  
为了避免这些系统默认行为，就需要为Activity声明这些配置，如下二个是每个Activity必须声明的：  
<activity android:configChanges="orientation|keyboardHidden">  
几乎所有的Activity都要声明如上，为什么Android不把它们变成Default的呢?  
  
**第二十七 尽量使用Android的API** 这好像是废话，在Android上面开发不用Android API用什么？因为Android几乎支持Java SE所有的API，所以有很多地方Android API与Java SE的API会有重复的地方，比如说对于文件的操作最好使用Android里面Context封装的API，而不要直接使用File对象：  
Context.openFileOutput(String); // no File file = new File(String)  
原因就是API里面会考虑到Android平台本身的特性；再如，少用Thread，而多使用AsyncTask等。  
  
  
**第二十八 要考虑到Activity和进程被杀掉的情况** 如了通常情况退出Activity外，还有Activity因其他原因被杀的情况，比如系统内存过低，系统配置变更，有异常等等，要考虑和测试这种情况，特别是Activity处理重要的数据时，做好的数据的保存。  
  
  
**第二十九 小心多语言** 有些语言真的很啰嗦，中文或英文很简短就能表达的事情到了其他语言就变的死长死长的，所以如果是wrap\_content就可能把其他控制挤出可视范围； 如果是指定长度就可能显示不全。也要注意特殊语言比如那些从右向左读的语言。  
  
**第三十 不要用四大组件去实现接口** 一是组件的对象都比较大，实现接口比较浪费，而且让代码更不易读和理解； 另外更重要的是导致多方引用，可能会引发内存泄露。  
  
  
**第三十二 用getApplication()来取Context当参数** 对于需要使用Context对象作为参数的函数，要使用getApplication()获取Context对象当参数，而不要使用this，除非你需要特定的组件实例！getApplication()返回的Context是属于Application的，它会在整个应用的生命周期内存在，远大于某个组件的生命周期，所以即使某个引用长期持有Context对象也不会引发内存泄露。  
  
  
**第三十三 主线程只做UI控制和Frameworks回调相关的事。附属线程只做费时的后台操作。交互只通过Handler。这样就可以避免大量的线程问题。**  
  
  
 **第三十四 Frameworks的回调不要做太多事情仅做必要的初始化，其他不是很重要的事情可以放到其他线程中去做，或者用Handler Schedule到稍后再做。**  
  
  
 **第三十五 要考虑多分辨率** 至少为hdpi, mdpi, ldpi准备图片和布局。元素的单位也尽可能的使用dip而不要用px。  
  
**第三十六 利用Android手机的硬键** 几乎所有的Android手机都有BACK和MENU，它们的作用是返回和弹出菜单，所以就不要再在UI中设计返回按扭和菜单按扭。很多优秀的应用如随手记和微信都有返回键，他们之所以有是因为他们都是从iOS上移植过来的，为了保存体验的一致，所以也有了返回和菜单。但这不够Android化，一个纯正的Android是没有必须重复硬键的功能的。  
PS：多看看官方的 APIDEMO，多读一下[Android](//link.zhihu.com/?target=http%3A//www.android.com) 上的内容。肥肥上面的各种废话，题主基本都能在开发者官网找到。  
  
第三十七 最好的一个习惯，放到最后压轴吧。善用 Google 和知乎。