**Android编程规范**

为什么需要编码规范?

编码规范对于程序员而言尤为重要，有以下几个原因：

1. 一个软件的生命周期中，80%的花费在于维护

2. 几乎没有任何一个软件，在其整个生命周期中，均由最初的开发人员来维护

3. 编码规范可以改善软件的可读性，可以让程序员尽快而彻底地理解新的代码

4. 如果你将源码作为产品发布，就需要确任它是否被很好的打包并且清晰无误，一如你已构建的其它任何产品

以下为编码规范详则：

1. **命名方法要求：**

**包名 ：以公司域名开头，后面跟功能紧密相关的名称:com.xxx.func**

**命名规则：一个唯一包名的前缀总是全部小写的ASCII 字母并且是一个顶级域名，通常是com，edu，gov，mil，net，org。包名的后续部分根据不同机构各自内部的命名规范而不尽相同。这类命名规范可能以特定目录名的组成来区分部门 (department) ，项目(project)，机器(machine)，或注册名(login names)。**

**类名：要求类名能够体现类的功能，使用名词或者动宾结构的短语**

**命名规则：类名采用大小写混合的方式，每个单词的首字母大写。尽量使你的类名简洁而富于描述。使用完整单词，避免缩写词(除非该缩写词被更广泛使用，像 URL，HTML)**

**规约：类名必须使用驼峰规则，即首字母必须大写，如果为词组，则每个单词的首字母也必须要大写，类名必须使用名词，或名词词组。要求类名简单，不允许出现无意义的单词（如 class XXXActivity）。**

**如：class BookMarkAdd 正确**

**如：class AddBookReadPlanActivity 错误！**

**应为 class BookReadPlanAddActivity**

**方法名：要求方法名能够体现方法的功能，其中to简写为2，for简写为4**

**命名规则：方法名是一个动词，采用大小写混合的方式，第一个单词的首字母小写，其后单词的首字母大写。**

**例如： public void run(); public String getBookName();**

**类中常用方法的命名：**

**类的获取方法（一般具有返回值）一般要求在被访问的字段名前加上get，如getFirstName()，getLastName()。一般来说，get前缀方法返回的是单个值，find前缀的方法返回的是列表值。**

**类的设置方法（一般返回类型为void）：被访问字段名的前面加上前缀 set，如setFirstName(),setLastName().**

**类的布尔型的判断方法一般要求方法名使用单词 is或has 做前缀，如isPersistent()，isString()。或者使用具有逻辑意义的单词，例如equal 或equals。**

**类的普通方法一般采用完整的英文描述说明成员方法功能，第一个单词尽可能采用动词，首字母小写，如openFile（），addCount（）。**

**构造方法应该用递增的方式写。（参数多的写在后面）。**

**toString()方法：一般情况下，每个类都应该定义toString(),其格式为：**

**变量命名**

**命名规则：第一个单词的首字母小写，其后单词的首字母大写。变量名不应以下划线或美元符号开头，尽管这在语法上是允许的。变量名应简短且富于描述。变量名的选用应该易于记忆，即，能够指出其用途。尽量避免单个字符的变量名，除非是一次性的临时变量。临时变量通常被取名为 i，j，k，m 和 n，它们一般用于整型；c，d，e，它们一般用于字符型。**

**例如：String bookName;**

**规约：变量命名也必须使用驼峰规则，但是首字母必须小写，变量名尽可能的使用名词或名词词组。同样要求简单易懂，不允许出现无意义的单词。如：String bookName; 正确 如：String bookNameString; 错误！**

**成员变量命名**

**同变量命名，但不要在私有变量前添加m字样！**

**常量命名**

**命名规则：类常量的声明，应该全部大写，单词间用下划线隔开。**

**例如：static final int MIN\_WIDTH = 4;**

**例如：static final int MAX\_WIDTH = 999;**

**例如：static final int GET\_THE\_CPU = 1;**

**异常命名**

**自定义异常的命名必须以Exception为结尾。已明确标示为一个异常。**

**layout 命名**

**规约：layout xml 的命名必须以 全部单词小写，单词间以下划线分割，并且使用名词或名词词组，即使用 模块名\_功能名称 来命名。**

**如：knowledge\_gained\_main.xml正确**

**如：list\_book.xml错误！**

**id 命名**

**规约：layout 中所使用的id必须以全部单词小写，单词间以下划线分割，并且使用名词或名词词组，并且要求能够通过id直接理解当前组件要实现的功能。**

**如：某TextView @+id/textbookname 错误 !应为 @+id/book\_name\_show**

**如：某EditText @+id/textbookname 错误 !应为 @+id/book\_name\_edit**

**资源命名**

**规约：layout中所使用的所有资源（如drawable,style等）命名必须以全部单词小写，单词间以下划线分割，并且尽可能的使用名词或名词组，即使用 模块名\_用途 来命名。如果为公共资源，如分割线等，则直接用用途来命名**

**如：menu\_icon\_navigate.png 正确**

**如：某分割线：line.png 或 separator.png 正确**

1. **注释内容要有意义，并且要求能够通过javadoc导出形成api**

**注释**

**Java 程序有两类注释：实现注释(implementation comments)和文档注释(document comments)。实现注释是使用/\*...\*/和//界定的注释。文档注释(被称为"doc comments")由/\*\*...\*/界定。文档注释可以通过javadoc 工具转换成HTML 文件。**

**方法注释**

**每一个方法都要包含 如下格式的注释 包括当前方法的用途，当前方法参数的含义，当前方法返回值的内容和抛出异常的列表。**

**/\*\***

**\***

**\* 方法的一句话概述**

**\* <p>方法详述（简单方法可不必详述）</p>**

**\* @param s 说明参数含义**

**\* @return 说明返回值含义**

**\* @throws IOException 说明发生此异常的条件**

**\* @throws NullPointerException 说明发生此异常的条件**

**\*/**

**类成员变量和常量注释**

**成员变量和常量需要使用java doc形式的注释，以说明当前变量或常量的含义**

**/\*\***

**\* XXXX含义**

**\*/**

**其他注释**

**方法内部的注释 如果需要多行 使用/\*⋯⋯ \*/形式，如果为单行是用//⋯⋯形式的注释。不要再方法内部使用 java doc 形式的注释“/\*\*⋯⋯\*\*/”，简单的区分方法是，java doc形式的注释在 eclipse中为蓝色，普通注释为绿色。**

**XML注释**

**规约：如果当前layout 或资源需要被多处调用，或为公共使用的layout（若list\_item），则需要在xml写明注释。要求注释清晰易懂。**

3. **方法功能要求单一，减少if和switch使用，方法代码长度不应超过200行。方法要求体现访问权限，即对外提供的方法声明为public，如果有两个以上类共用的工具方法定义在工具类，其他方法声明为私有方法。工具类定义为常量类，工具类方法定义为静态方法。方法的参数尽量使用简单数据类型。**

1. **while和if语句的嵌套层数不能超过三层。**
2. **除线程池外，使用线程要求显示定义，并有销毁方法。不能使用while(true)。**
3. **文件保存格式统一为UTF-8。**
4. **对象释放时使用obj=null赋值，方便垃圾回收。**
5. **使用多线程时共享资源需要使用synchronized声明。**
6. **判断语句中，如果有常量比较要求常量在左**
7. **方法/匿名类或if、switch语句块之间使用空行分隔。**
8. **系统模块之间调用使用接口,接口和实现由功能提供方定义，调用方调用**
9. **日志分为运行日志和系统日志。包括错误日志、告警日志、info日志、debug四种级别。系统日志指系统启动过程中记录的日志，运行日志指系统启动后运行过程中记录的日志。系统日志和运行日志写文件。要求少用级联调用的方法，如getStaff().getDept().getSuper()类似的方法。getInstance()类型的方法要求只能两级，如XXX.getInstance().getStaffManager()。**
10. **在方法内对传入的复杂类型参数进行判断，防止参数错误引起的异常。如数组长度判断，对象类型判断等。**
11. **异常处理中禁止实现业务逻辑，如有必要需要通过项目经理评审。在catch和finally中要求记录日志并将定义的方法变量置为null。**
12. **在代码中原则上应该消除告警，不可避免的代码告警应有记录.**
13. **GC内存泄露：**
14. **数据库的cursor使用完要关闭**
15. **构造adapter时，要使用缓存contentView，衍生listView的优化问题---减少创建view的对象，充分使用contentView，可以使用最终（final）类库优化处理getView的过程。**
16. **Bitmap对象不使用时采用recycle()释放内存。**