Android 开发代码规范

1. 命名基本原则

在面向对象编程中，对于类，对象，方法，变量等方面的命名是非常有技巧的。比如，大小写的区分，使用不同字母开头等等。但究其本，追其源，在为一个资源其名称的时候，应该本着描述性以及唯一性这两大特征来命名，才能保证资源之间不冲突，并且每一个都便于记忆。

对于理解应用程序的逻辑流，命名方案是最有影响力的一种帮助。名称应该说明“什么”而不是“如何”。命名原则是：使名称足够长以便有一定的意义，并且足够短以避免冗长。唯一名称在编程上仅用于将各项区分开。以下几点是规范的命名方法。

2.命名基本规范

2.1. 编程基本命名规范

（1）避免难懂的名称，如属性名 xxK8，这样的名称会导致多义性。

（2） 在面向对象的语言中，在类属性的名称中包含类名是多余的，如 Book.BookTitle，而是应该使用 Book.Title。

（3）在允许函数重载的语言中，所有重载都应该执行相似的函数。

（4）使用动词-名词的方法来命名对给定对象执行特定操作的例程，如 CalculateInvoiceTotal()。（例程是某个系统对外提供的功能接口或服务的集合）

（5）只要合适，在变量名的末尾或开头加计算限定符（Avg、Sum、Min、Max、Index）。

（6）在变量名中使用互补对，如 min/max、begin/end 和 open/close。

（7）布尔变量名应该包含 Is，这意味着 Yes/No 或 True/False 值，如 fileIsFound。

（8）即使对于可能仅出现在几个代码行中的生存期很短的变量，仍然使用有意义的名 称。仅对于短循环索引使用单字母变量名，如 i 或 j。

（9）为了帮助区分变量和例程，对例程名称使用 Pascal 大小写处理 (CalculateInvoiceTotal)，其中每个单词的第一个字母都是大写的。对于变量名，使用 camel 大小写处理 (documentFormatType)，其中除了第一个单词外每个单词的第一个字母都是大写的。

（10）不要使用原义数字或原义字符串，而是使用命名常数，NUM_DAYS_IN_WEEK ，以便于维护和理解。

2. 2. 分类命名规范

（1）包的命名

Java 包的名字都是由小写单词组成。但是由于 Java 面向对象编程的特性，每一名 Java 程序员都可以编写属于自己的 Java 包，为了保障每个 Java 包命名的唯一性，在最新的 Java 编程规范中，要求程序员在自己定义的包的名称之前加上唯一的前缀。由于互联网上的域名称是不会重复的，所以程序员一般采用自己在互联网上的域名称作为自己程序包的唯一前缀。

例如： net.frontfree.javagroup

（2）类的命名

类的名字必须由大写字母开头而单词中的其他字母均为小写；如果类名称由多个单词组成，则每个单词的首字母均应为大写例如 TestPage；如果类名称中包含单词缩写，则这个所写词的每个字母均应大写，如：XMLExample,还有一点命名技巧就是由于类是设计用来代表对象的，所以在命名类时应尽量选择名词。

例如： Circle

（3）方法的命名

方法的名字的第一个单词应以小写字母作为开头，后面的单词则用大写字母开头。

例如： sendMessge

（4）.常量的命名

常量的名字应该都使用大写字母，并且指出该常量完整含义。如果一个常量名称由多个单词组成，则应该用下划线来分割这些单词。

例如： MAX_VALUE

（5）参数的命名

参数的命名规范和方法的命名规范相同，而且为了避免阅读程序时造成迷惑，请在尽量保证参数名称为一个单词的情况下使参数的命名尽可能明确。

（6）Javadoc 注释

Java 除了可以采用我们常见的注释方式之外，Java 语言规范还定义了一种特殊的注释，也就是我们所说的 Javadoc 注释，它是用来记录我们代码中的 API 的。Javadoc 注释是一种多行注释，以/**开头，而以*/结束，注释可以包含一些 HTML 标记符和专门的关键词。使用 Javadoc 注释的好处是编写的注释可以被自动转为在线文档，省去了单独编写程序文档的麻烦。

例如：

```
/**
* This is an example of
* Javadoc
```

*

* @author darchon

* @version 0.1, 10/11/2002

*/

在每个程序的最开始部分，一般都用 Javadoc 注释对程序的总体描述以及版权信息，之后在主程序中可以为每个类、接口、方法、字段添加 Javadoc 注释，每个注释的开头部分先用一句话概括该类、接口、方法、字段所完成的功能，这句话应单独占据一行以突出其概括作用，在这句话后面可以跟随更加详细的描述段落。在描述性段落之后还可以跟随一些以 Javadoc 注释标签开头的特殊段落，例如上面例子中的@auther 和@version，这些段落将在生成文档中以特定方式显示。

虽然为一个设计低劣的程序添加注释不会使其变成好的程序，但是如果按照编程规范编写程序并且为程序添加良好的注释却可以帮助你编写出设计完美，运行效率高且易于理解的程序，尤其是在多人合作完成同一项目时编程规范就变得更加重要。俗话说“磨刀不误砍柴工”，花费一点时间去适应一下 Java 编程规范是有好处的。

3. 分类命名规范

3. 1. 基本数据类型命名规范

Integer：int+描述	Char：chr+描述	Boolean：bln+描述
Long：lng+描述	Short：shr +描述	Double：dbl+描述
String：str+描述	Float：flt+描述	Single：sng+描述
DataTime：dt+描述	Array：arr+描述	Object：obj+描述

如：String srtName;

3. 2. 控件命名规范

TextView ： txt_+描述	Button ： btn_+描述
ImageButton ： imgBtn_+描述	ImageView ： imgView_+描述
CheckBox ： chk_+描述	RadioButton ： rdoBtn_+描述
AnalogClock ： anaClk_+描述	DigitalClock ： DgtClk_+描述
DatePicker ： dtPk_+描述	TimePicker ： tmPk _+描述
ToggleButton ： tglBtn_+描述	EditText：edtTxt_+描述
ProgressBar：lcb_+描述	SeekBar：skBar _+描述
AutoCompleteTextView：autoTxt_+描述	MultiAutoCompleteTextView：mlAutoTxt_+描述

ZoomControls：zmCtrl_+描述	Include：ind_+描述
VideoView：vdoVi_+描述	WebView：webVi_+描述
RatingBar：ratBar_+描述	Tab：tab__+描述
Spinner：spn_+描述	Chronometer：Cmt_+描述
ScrollView：sclVi_+描述	TextSwitcher：txtSwt_+描述
Gallery：gal_+描述	ImageSwitcher：imgSwt_+描述
GridView：gV_+描述	ListView：lVi_+描述
ExpandableList: epdLt_+描述	MapView: mapVi_+描述

控件说明如下：

- TextView - 文本显示控件
- Button - 按钮控件
- ImageButton - 图片按钮控件
- ImageView - 图片显示控件
- CheckBox - 复选框控件
- RadioButton - 单选框控件
- AnalogClock - 钟表（带表盘的那种）控件
- DigitalClock - 电子表控件
- DatePicker - 日期选择控件
- TimePicker - 时间选择控件
- ToggleButton - 双状态按钮控件
- EditText - 可编辑文本控件
- ProgressBar - 进度条控件
- SeekBar - 可拖动的进度条控件
- AutoCompleteTextView - 支持自动完成功能的可编辑文本控件
- MultiAutoCompleteTextView - 支持自动完成功能的可编辑文本控件，允许输入多值（多值之间会自动地用指

定的分隔符　　分开）

- ZoomControls - 放大/缩小按钮控件
- Include - 整合控件
- VideoView - 视频播放控件
- WebView - 浏览器控件
- RatingBar - 评分控件
- Tab - 选项卡控件
- Spinner - 下拉框控件
- Chronometer - 计时器控件

- ScrollView - 滚动条控件
- TextSwitcher - 文字转换器控件（改变文字时增加一些动画效果）
- Gallery –画廊控件
- ImageSwitcher - 图片转换器控件（改变图片时增加一些动画效果）
- GridView - 网格控件
- ListView - 列表控件
- ExpandableList - 支持展开/收缩功能的列表控件

3. 3. 变量命名规范

变量命名：前缀+类型描述+意义描述

前缀：

成员变量：m_***　　局部变量：l_***　　形参：a_***

常量：大写_***　　枚举值：em_***

3. 4. 程序规范

工程的命名为：描述

应用程序名的命名为：描述+App

4. 代码书写规范

（1）建立标准的缩进大小（如四个空格），并一致地使用此标准。用规定的缩进对齐代码节。

（2）在发布源代码的硬拷贝版本时使用特定的字体以及字号（新宋体、小五号）。

（3）在括号对对齐的位置垂直对齐左括号和右括号，如：

```
for   (i=0; i<100; i++)
{
    ;
}
```

（4）沿逻辑结构行缩进代码使代码更易于阅读和理解，如：

```
if(expression)
    {
```

```
if(expression )
 {
  //
  //此处填写你的代码块;
  //
 }
else
 {
  //
  //此处填写你的代码块;
  //
 }
}
```

（5）为注释和代码建立最大的行长度，以避免不得不滚动源代码编辑器，并且可以提供整齐的硬拷贝表示形式。

（6）当一行内容太长而必须换行时，在后面换行代码中要使用缩进格式，如下：

```
string inserString ="Insert Into TableName(username,password,email,sex,address) "
+"Values( 'Soholife ', 'chenyp ', 'soholife@sina.com ', 'male ', '深圳福田 ') ";
```

（7）每一行上放置的语句避免超过一条。特殊循环如 for(i =0;i<100;i++)等除外。

（8）编写 SQL 语句时，对于关键字使用全部大写，对于数据库元素（如表、列和视图）使用大小写混合。例如 SELECT * FROM Table1；

（9）将每个主要的 SQL 子句放在不同的行上，这样更容易阅读和编辑语句，例如：

```
SELECT FirstName, LastName
FROM Customers
WHERE State = 'WA '
```

（10）在物理文件之间在逻辑上划分源代码。

（11）使用空白为源代码提供结构线索。这样做会创建代码“段”，有助于读者理解软件的逻辑分段

（12）将大的复杂代码段分为较小的、易于理解的模块。

5. 注释

软件文档以两种形式存在：外部的和内部的。外部文档（如规范、帮助文件和设计文档）在源代码的外部维护。内部文档由开发人员在开发时在源代码中编写的注释组成。

不考虑外部文档的可用性，由于硬拷贝文档可能会放错地方，源代码清单应该能够独立存在。外部文档应该由规范、设计文档、更改请求、错误历史记录和使用的编码标准组成。　　以下几点是规范的注释方法：

（1）一个工程应有一个统一的头文件注释，以说明整个工程的信息、创建日期、版本等等

（2）对重要的程序加注释进行说明

（3）修改代码或删除时，将原代码用注释的方法屏蔽，同时要加开发者自身对修改操作的注释。格式为：

```
//原代码
//Added/（Modified/ Deleted） by 开发者姓名 年-月-日;
//因为业务原因修改的，要注明修改或删除原因）
新代码
```

（4）使用 XML 文档格式，如下面方法的注释：

```
/// <summary>
/// 得到某人的年龄
/// </summary>
/// <param name= "userName "> 用户名 </param>
/// <returns> 用户年龄 </returns>
public int GetUserAge(string userName)
{
  //
  //此处写你的程序代码
  //
}
```

（5）避免杂乱的注释，而是应该使用空白将注释同代码分开。

（6）移除所有临时或无关的注释，以避免在日后的维护工作中产生混乱。

（7）注释应对代码进行准确的说明，不应存在歧义。

（8）在整个应用程序中，使用具有一致的标点和结构的统一样式来构造注释。