**源文件规范**

1. 文件名  
   源文件名必须和它包含的顶层类名保持一致，包括大小写，并以.java作为后缀名。
2. 文件编码  
   所有源文件编码必须是UTF-8

**命名**

1. 包名  
   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).  
   例如： com.domain.xx

包命名必须以com.domain开始,后面跟有项目名称（或者缩写）,再后面为模块名或层级名称。  
如：com.domain.项目缩写.模块名 com.domain.xx.bookmark  
如：com.domain.项目缩写.层级名 com.domain.xx.activity

1. 类和接口命名  
   命名规则：类名是个一名词，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)

接口一般要使用able,ible,er等后缀

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。

1. 方法的命名  
   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.

1. 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,n它们一般用于整型；  
   c,d,e 它们一般用于字符型。  
   在Android中成员变量  
   非public非static的变量可以使用m开头  
   非常量的static变量可以使用s开头

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。  
例如：private String mBookName;

1. 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：private static final int MIN\_WIDTH = 4;
2. 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。
3. layout命名  
   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*所属页面类型 来命名。  
   如：video\_controller\_player\_activity 视频模块下的-控制栏-属于播放器的-Activity页
4. id命名  
   ·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/tv\_book\_name\_show如：某EditText @+id/et\_book\_name\_edit
5. 资源命名

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

**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。

1. 类注释  
   每一个类都要包含如下格式的注释，以说明当前类的功能等。  
   /\*\*
   * 类名
   * @author 作者
   * 实现的主要功能。
   * 创建日期
   * 修改者，修改日期，修改内容。  
     \*/
2. 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。  
   /\*\*
3. 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

\*/

1. 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“/\*\*……\*/”

**代码风格**

1. 缩进  
   除了换行符之外，ASCII空格（0x20）是唯一合法的空格字符。这意味着  
   不允许使用Tab进行缩进，应该使用空格进行缩进，推荐缩进为4个空格  
   Eclipse中将Tab替换为4个空格的设置方法(很多人都习惯直接按4次空格，感觉不设置习惯了也挺好)  
   – 代码设置  
   Window->Preferences->General->Editors->Text Editors->勾选Insert spaces for tabs``  
   - XML文件的Tab配置Window->Preferences->XML->XML Files->Editor>选择右侧区域的Indent using spaces`
2. 空行  
   空行将逻辑相关的代码段分隔开，以提高可读性。

下列情况应该总是使用空行：  
 – 一个源文件的两个片段之间  
 – 类声明和接口声明之间  
 – 两个方法之间  
 – 方法内的局部变量和方法的第一条语句之间  
 – 一个方法内的两个逻辑段之间，用以提高可读性

通常在 变量声明区域之后要用空行分隔，常量声明区域之后要有空行分隔，方法声明之前要有空行分隔。

1. 方法  
    – 一个方法尽量不要超过15行(可能会有难度，但是尽量不要太多，弄个方法几千行这是绝对不允许的)，如果方法太长，说明当前方法业务逻辑已经非常复杂，  
   那么就需要进行方法拆分，保证每个方法只作一件事。  
    – 不要使用try catch处理业务逻辑！！！！
2. 参数和返回值
   * 一个方法的参数尽可能的不要超过4个(根据情况可能也会有些难度)
   * 如果一个方法返回的是一个错误码，请使用异常！！
   * 尽可能不要使用null替代为异常
3. 神秘数字  
   代码中不允许出现单独的数字，字符！如果需要使用数字或字符，则将它们按照含义封装为静态常量!(for语句中除外)
4. 控制语句  
   判断中如有常量，则应将常量置于判断式的右侧。如：  
   if (true == isAdmin())...  
   尽量不要使用三目条件的嵌套。

在if、else、for、do和while语句中，即使没有语句或者只有一行，也不得省略花括号：

1. 访问控制  
   若没有足够理由，不要把实例或类变量声明为公有。
2. 变量赋值  
   不要使用内嵌(embedded)赋值运算符试图提高运行时的效率，这是编译器的工作。例如：  
   d = (a = b + c) + r;  
   应该写成
3. 圆括号的试用  
   一般而言，在含有多种运算符的表达式中使用圆括号来避免运算符优先级问题，是个好方法。  
   即使运算符的优先级对你而言可能很清楚，但对其他人未必如此。你不能假设别的程序员和你一样清楚运算符的优先级。  
   不要这样写：  
   if (a == b && c == d)  
   正确的方式为：  
   if ((a == b) && (c == d))

**开发格式统一**

1. Eclipse  
   Windows -> Preferences -> Java -> Code Style  
   然后选择Import导入相应的Clean Up、Code Templates、Formatter等XML文件。  
   如果不需要Copyright信息，想要自定义的，可以不导入Code Templates。
2. IDEA  
   File -> Import Settings选择下载链接中的IDEA\_Style.jar文件，  
   可以看到两个选项，只需代码风格的，可以仅选择Code style schemes,  
   如果需要默认的Copyright信息，选择Default Project settings。

**代码严谨性要求**

1. ArrayList通过get方法使用下标获取元素，如果使用的下标不在ArrayList大小范围内，将产生java.lang.IndexOutOfBoundsException的异常，导致app出现Crash。
2. 方法中存在return null返回对象直接进行方法调用隐患, 在使用时需要先判断是否为null，一般尽量不要在方法中直接return null，最好用异常代替。
3. 销毁Dialog前是否isShowing未判断隐患  
   调用Android.app.Dialog.cancel()方法前，如果这个dialog不处于showing状态时，会抛出java.lang.IllegalArgumentException的异常，导致app出现Crash。
4. 使用String.split结果未判断长度隐患  
   在使用String.split得到的结果数组前，未对数组进行长度检查，取字段的时候可能发生越界而导致Crash。