五. 编码规范：

5.1 格式规范

5.1.1 源文件结构

一个源文件包含(按顺序地)：

1>. package 语句

2>. import 语句

3>.一个顶级类(只有一个)

1>. package 语句不换行(即 package 语句写在一行里)

2>. import 不要使用通配符

即，不要出现类似这样的 import 语句：import java.util.*;

不要换行

import 语句不换行，列限制(4.4 节)并不适用于 import 语句。(每个 import 语句独立成行)

顺序和间距

import 语句可分为以下几组，按照这个顺序，每组由一个空行分隔：

所有的静态导入独立成组

com.google imports(仅当这个源文件是在 com.google 包下)

第三方的包。每个顶级包为一组，字典序。例如：android, com, junit, org, sun

java imports

javax imports

3>. 只有一个顶级类声明

每个顶级类都在一个与它同名的源文件中(当然，还包含.java 后缀)。

例外：package-info.java，该文件中可没有 package-info 类。

5.1.2 缩进：我们不采用 Tab 键，而是手动输入 4 个空格。

虽然 Tab 键一般为 4 个空格键，但在很多的编辑器中都可以拓展 Tab 键为多个空格键。不采用 Tab 键理由是不同情况可能显示不同的长度，严重影响阅读体验。

5.1.3 行宽：限定为 100 个字符

5.1.4 括号：在复杂的表达式中，用括号清楚地表示逻辑优先级

使读者能够快速、清楚看出表达式的运算顺序

5.1.5 断行与空白行{}:所有的‘{’ ‘}’各占一行

5.1.6 分行：多条语句不要放在同一行，

5.2 命名规范

首要原则--见名知意。普通变量采用 Camel 法，并采用名词或者组合名词来命名。而类型、类、函数名采用 Pascal 法，并采用动词或者动宾的方式命名。宏则全部采用大写字母，采用名词或者组合名词来命名，多个词之间用下划线连接。

5.2.1 变量（variables）:采用 Camel 命名法。类中控件名称必须与 xml 布局 id 保持一致。用统一的量词通过在结尾处放置一个量词，就可创建更加统一的变量，它们更容易理解，也更容易搜索。例如，请使用 strCustomerFirst 和 strCustomerLast，而不要使用 strFirstCustomer 和 strLastCustomer。

5.2.2 包（packages）：采用反域名命名规则，全部使用小写字母。一级包名为 com，二级包名为 xx（可以是公司或则个人的随便），三级包名根据应用进行命名，四级包名为模块名或层级名

5.2.3 类（classes）：名词，采用 Pascal 命名法，尽量避免缩写，除非该缩写是众所周知的， 比如 HTML,URL，如果类名称中包含单词缩写，则单词缩写的每个字母均应大写。

5.2.4 layout 中的 id 命名 命名模式为：view 缩写_模块名称_view 的逻辑名称

5.3 函数使用

5.3.1 非调度函数应减少或防止控制参数，尽量只使用数据参数。

5.3.2 除非必要，最好不要把与函数返回值类型不同的变量，以编译系统默认的转换方式或强制的转换方式作为返回值返回。

5.3.3 在调用函数填写参数时，应尽量减少没有必要的默认数据类型转换或强制数据类型转换。

5.3.4 设计高扇入、合理扇出（小于 7）的函数。

说明：扇出是指一个函数直接调用（控制）其它函数的数目，而扇入是指有多少上级函数调用它。

5.3.5 避免使用 BOOL 参数。

5.3.6 对于提供了返回值的函数，在引用时最好使用其返回值。

5.3.7 当一个过程（函数）中对较长变量（一般是结构的成员）有较多引用时，可以用一个意义相当的宏代替。

5.4 注释规范

1)、函数头的注释 对于函数，应该从“功能”,“参数”,“返回值”、“主要思路”、“调用方法”、“日期”六个方面用如下格式注释：

```
//程序开始
//================================//
//功能：
//参数：
//入口
//出口
//返回；对返回值有错误编码的要求列出错误编码
//================================================//  函数名(……)
//程序说明结束
```

①对于某些函数，其部分参数为传入值，而部分参数为传出值，所以对参数要详细说明该参数是入口参数，还是出口参数，对于某些意义不明确的参数还要做详细说明（例如：以角度作为参数时，要说明该角度参数是以弧度（PI）,还是以度为单位）,对既是入口又是出口的变量应该在入口和出口处同时标明。等等。

②函数的注释应该放置在函数的头文件中,在实现文件中的该函数的实现部分应该同时放置该注释。

③在注释中应该详细说明函数的主要实现思路、特别要注明自己的一些想法，如果有必要则应该写明对想法产生的来由。对一些模仿的函数应该注释上函数的出处。

④在注释中详细注明函数的适当调用方法，对于返回值的处理方法等。在注释中要强调调用时的危险方面，可能出错的地方。