Android编码规范

命名规范

包命名规范

- 采用反域名命名规则，包名全部小写，连续的单词只是简单地连接起来，不使用下划线，一级包名为com，二级包名为xxx（可以是公司域名或者个人命名），三级包名根据应用进行命名，四级包名为模块名或层级名。如：`com.isa.crm.activity` `com.isa.crm.adapter`

Java类命名规范

- 采用大驼峰式命名法，尽量避免缩写，除非该缩写是众所周知的，比如HTML，URL,如果类名称包含单词缩写，则单词缩写的每个字母均应大写。如：`ProductListActivity` `JsonHTTPSRequest`

接口命名规范

- 命名规则与类一样采用大驼峰命名法，多以`able`或`ible`结尾。例如：`interface Runable` `interface Accessible`

成员变量命名规范

类成员变量名称

- 使用Google的m命名法，例如：`private String mUserName;`

方法中临时变量名称

- 如用户名: `String userName;`

常量命名

- 常量使用全大写字母加下划线的方式命名。例如：`public static final String TAG_ERROR = "error";`

控件实例命名

- 类中控件名称必须与xml布局id保持一致(可以去掉`{module_name}`)。例如: 在布局文件中Button 的id为:`android:id="@+id/btn_pay"` `private Button btn_pay;`

方法命名规范

- 动词或动名词，采用小驼峰命名法。例如：`run();` `onCreate()` `syncProducts()`;

布局文件(Layout)命名规范

*全部小写，采用下划线命名法。其中`{module_name}`为业务模块或是功能模块等模块化的名称或简称。

布局	**命名**	**例子**
activity layout	(module_name)_activity_(名称)	user_activity_info.xml
fragment layout	(module_name)_fragment_(名称)	user_fragment_info.xml
Dialog layout	(module_name)_dialog_(名称)	login_dialog_loading.xml
列表项布局	(module_name)_list_item_(名称)	main_list_item_devices
包含项布局命名	include_(名称)	include_head.xml
adapter的字布局	(module_name)_item_(名称)	qz_item_order.xml

资源id命名规范

- 命名模式为:(view缩写)_(view的逻辑名称),如：
  用户详情模块布局`LinearLayout`的布局id->`ll_content`;
  常见控件View

控件	缩写
LinearLayout	ll
RelativeLayout	rl
TextView	tv
Button	btn
ImageButton	imgBtn
ImageView	iv
CheckBox	cb
RadioButton	rb
DatePicker	dtPk
EditText	et
TimePicker	tmPk
ProgressBar	proBar
WebView	wv
Spinner	spn
ScrollView	sv
listView	lv

代码风格

大括号

```
if (hasMoney()) {

} else {

}
```

空格问题

`if else` `while` `运算符两端`等后面需要空格隔开

> 规范的编程方式:

```
if (hasMoney()) {

} else {

}

for (int i = 0; i &lt; 10; i++) {

}
```

> 不规范的编写方式：

```
if(hasMoney()){

}else{

}

for(int i=0;i&lt;10;i++){

}
```

文件组织

类注释

- 所有的源文件都应该在开头有一个C语言风格的注释,其中列出类名、版本信息、日期、版权声明、描述、作者:

```
/**
* @ClassName: ${class_name}
*
* @Description: ${todo}
*
* @author xxx
*
* @date ${date} ${time}
*
* @version V1.0
*
* Copyright (C)
*/
```

块注释

- 看块注释通常用于提供对文件,方法,数据结构和算法的描述。

```
/*
 * Here is a block comment.
 */
```

单行注释

- 短注释可以显示在一行内,并与其后的代码具有一样的缩进层级

```
if (condition) {

    /* Handle the condition. */
    ...
}
```

尾端注释

- 极短的注释可以与它们所要描述的代码位于同一行,但是应该有足够的空白来分开代码和注释。若有多个短注释出现于大段代码中,它们应该具有相同的缩进。

```
if (a == 2) {
    return TRUE;         /* special case */
} else {
    return isPrime(a);   /* works only for odd a */
}
```

行末注释

- 注释界定符"//",可以注释掉整行或者一行中的一部分。它一般不用于连续多行的注释文本;然而,它可以用来注释掉连续多行的代码段。

类和接口的声明

- 描述了类和接口声明的各个部分以及它们出现的先后次序

类/接口声明的各部分	**注解**
类/接口文档注释(/**.....*/)	生成文档给第三方调用者的内容
类/接口的声明	
类/接口实现的注释(/...../)	该注释应包含任何有关整个类/接口的信息,而这些信息又不适合作为类/接口的文档注释
类的(静态)变量	首先是类的公共变量,随后是保护变量,再后是包级别的变量(没有访问修饰符,access modifier),最后是私有变量。
实例变量	首先是公共级别的,随后是保护级别的,再后是包一级别的(没有访问修饰符),最后是私有级别的。
构造器	

类/接口声明的各部分	注解
方法	这些方法应该按功能,而非作用域或访问权限,分组。例如,一个私有的类方法可以置于两个公有的实例方法之间。其目的是为了更便于阅读和理解代码。

声明

- 变量的声明
  推荐一行一个声明,有利于写注释。

```
int level;   // indentation level
int size;    // size of table
```

- 初始化
  尽量在声明局部变量的同时初始化
- 布局
  只在代码块的开始处声明变量。(一个块是指任何被包含在大括号"{"和"}"中间的代码。)不要在首次用到该变量时才声明之。这会把注意力不集中的程序员搞糊涂,同时会妨碍代码在该作用域内的可移植性。

```
void myMethod() {
    int int1 = 0;
    // beginning of method block
    if (condition) {
        int int2 = 0;
        // beginning of "if" block
        ...
    }
}

该规则的一个例外是for循环的索引变量
for (int i = 0; i < maxLoops; i++) {
    ...
}
```

- 类和接口的声明
  当编写类和接口是,应该遵守以下格式规则:

- 在方法名与其参数列表之前的左括号"("间不要有空格
- 左大括号"{"位于声明语句同行的末尾
- 右大括号"}"另起一行,与相应的声明语句对齐,除非是一个空语句，"}"应紧跟在"{"之后
- 方法与方法之见以空行分隔

```
class Sample extends Object {
int ivar1;
int ivar2;

Sample(int i, int j) {
    ivar1 = i;
    ivar2 = j;
}

int emptyMethod() {}

...
}
```

排版

- 行长度
  尽量避免一行的长度超过80个字符,因为很多终端和工具不能很好处理之。
  注意:用于文档中的例子应该使用更短的行长,长度一般不超过70个字符。
- 换行
  当一个表达式无法容纳在一行内时,可以依据如下一般规则断开之:
  - 在一个逗号后面断开
  - 在一个操作符前面断开
  - 选择较高级别的操作符断开
  - 新的一行应该与上一行同一级别表达式的开头对齐
  - 如果以上规则导致你的代码混乱或者使你的代码都堆积在右边,那就缩进8个空格

推荐换行实例：

```
someMethod(longExpression1,longExpression2,longExpression3,
        longExpression4, longExpression5);

```

```
longName1 = longName2 * (longName3 + longName4 - longName5)
        + 4 * longname6;//更高级别的断开


if ((condition1 && condition2)
        || (condition3 && condition4)
        ||!(condition5 && condition6)) {
    doSomethingAboutIt();
}
```

空白

下面情况应该使用空格：

- 一个紧跟着括号的关键字应该被空格分开,例如:

```
while (true) {
    ...
}
```

- 空白应该位于参数列表逗号的后面
- 所有的二元运算符,除了".",应该使用空格将之与操作书分开。一元操作符和操作书之间不应该加空格,比如:符号("-"),自增("++")和自减("--")。如:

```
a += c + d;

a = (a + b) / (c * d);

while (d++ = s++) {
    n++;
}

printSize("size is " + foo + "\n");
```

- for语句中的表达式应该被空格分开,例如:

```
for(expr1; expr2; expr3)
```

- 强制转型后应该跟一个空格,例如:

```
1.  myMethod((byte) aNum, (Object) x);
2.
3.  myMethod((int) (cp + 5), ((int) (i + 3)) + 1);
```