**Android代码规范**

**一个软件的生命周期中，80%的花费在于维护**

**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**

**编码规范可以改善软件的可读性，可以让程序员尽快而准地理解新的代码**

**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**

**参考链接：**

[**https://source.android.com/source/code-style**](https://source.android.com/source/code-style)**（面向贡献者的 AOSP 代码样式指南）**

[**http://www.hawstein.com/posts/google-java-style.html**](http://www.hawstein.com/posts/google-java-style.html)**（Google Java编程风格指南）**

[**http://www.wanandroid.com/blogimgs/a2609aed-1000-4039-93c3-7541aaa2013b.pdf**](http://www.wanandroid.com/blogimgs/a2609aed-1000-4039-93c3-7541aaa2013b.pdf)**（阿里巴巴Android开发手册）**

<https://github.com/alibaba/p3c>**（阿里巴巴Java开发手册）**

[**http://km.oa.com/group/ep/articles/show/332047**](http://km.oa.com/group/ep/articles/show/332047) **(Android Java编码规范 adroplv)**

[**http://km.oa.com/group/19976/articles/show/164164**](http://km.oa.com/group/19976/articles/show/164164) **(Android编码指南 sophiafu)**

**https://www.cnblogs.com/jesonjason/p/5577395.html(Internet)**

1. **命名规范**

命名规范主要针对编程过程中所有需要自定义名称的场景，使用同一规范方便维护和理解代码。

UpperCamelCase：每个词首字母大写，其他字母小写

LowerCamelCase：第一个词全部小写，其他词首字母大写，其他字母小写

**1.[通用][强制]**代码中命名均不能以下划线或美元符开始和结尾

**1.[通用][强制]**代码中的命名严禁使用拼音与英文混合的方式

**1.[通用][强制]**代码中的命名严禁使用不规范的缩写（望文不知义）

**1.[通用][强制]**代码中的命名拼写错误必须检查，避免拼写错误（专有名词/通用缩写除外）

IDE会把拼写不正确的单词标波浪线，并提示“typo: In word”。每出现这种情况时，必须检查是否需要修正。

**1.[包名][强制]**包名统一使用小写，点分隔之前有且仅有一个自然语义单词，不使用下滑线

一级包名为顶级域名：com/edu/gov/net/org

二级包名为公司/个人名

常见分包：db/base/util/activity/adapter/manager/test/common/model等等

**1.[包名][建议]**包名统一使用单数形式

**1.[类名][强制]** 类名使用 UpperCamelCase 风格

抽象类命名使用 Abstract 或 Base 开头；

异常类命名使用 Exception 结尾；

测试类命名以它要测试的类名开始，以 Test 结尾

解析类以Paser结尾

自定义共享基础类以Base开头

类名中通用缩写不需全部大写，使用UpperCamelCase 风格即可（Android特有，jdk中专有缩写全部大写）

**1.[方法名][强制]** 方法名统一使用LowerCamelCase 风格

方法名尽量能自解释方法功能

**1.[变量][强制]**局部变量名统一使用LowerCamelCase 风格

**1.[变量][建议]**临时变量取名使用单字母

i/j/k/m/n用于整型；c用于字符型

**1.[变量][强制]**非public的字段名统一使用LowerCamelCase 风格，**[建议]**以小写m开头

**1.[变量][强制]** 静态字段名统一使用LowerCamelCase 风格，**[建议]**以小写s(private)/g(public)开头

**1.[常量][强制]**常量命名统一使用大写字母加下划线格式

**1.[控件ID][建议]**xml文件内控件id的命名使用“view缩写\_描述”的格式

|  |  |
| --- | --- |
| LinearLayout | ll |
| RelativeLayout | rl |
| FrameLayout | fl |
| RecyclerView | rv |
| ListView | lv |
| GridView | gv |
| ImageView | iv |
| TextView | tv |
| Button | btn |

示例：btn\_title\_back、iv\_donwload\_manager、fl\_content

**1.[控件名][建议]**代码中控件变量的命名参照[]，变量以控件缩写结尾，大块背景控件以RootView或Layout作为变量名称结尾（ll,fl,rl不太好理解）

示例：mNameTv、mNumberEdt、mPhotoIv、mRootView、mCtrlLayout

**1.[ layout文件名][强制]** 命名方式：小写字母加下划线

Activity/Fragement/Dialog以activity\_/fragment\_/dialog\_开头

ListView/RecyclerView/GridView的子项以list\_item/recycle\_item/grid\_item开头

自定义卡片/导航栏以view\_开头

**1.[** **xml drawable文件名][强制]**命名方式：小写字母加下划线

以selector\_/shape\_/layer\_list\_开头

**1.[图片资源][强制]**命名方式：小写字母加下划线

图标统一以icon\_开头（icon\_back.png）

背景图以bg\_开头（bg\_default\_banner.png）

普通通用图片pic\_开头（pic\_red\_dot）

**1.[anim文件名][强制]**命名方式：小写字母加下划线

以动画名称命名：fade\_in.xml、slip \_out.xml

**1.[** **color/strings/dimens][强制]**命名方式：小写字母加下划线

使用模块名+描述的方式：

<color name=”home\_navigation\_btn\_tx\_color”>#112233</color>

<string name=”home\_navigation”>导航</ string>

<dimens name=”home\_navigation\_btn\_height”>56dp</ dimens>

**1.[** **declare-styleable/style][强制]**命名方式：UpperCamelCase 风格

<style name="BigRatingBar" parent="@android:style/Widget.RatingBar">

<item name="android:progressDrawable">@drawable/layer\_big\_ratingbar</item>

<item name="android:minHeight">27dip</item>

</style>

<declare-styleable name="ExpandableTextView">

<attr name="maxCollapsedLines" format="integer"/>

<attr name="animDuration" format="integer"/>

<attr name="animAlphaStart" format="float"/>

<attr name="expandDrawable" format="reference"/>

<attr name="collapseDrawable" format="reference"/>

</declare-styleable>

**每当不确定某种类型的命名方式时，参照Android源码中相应内容的命名方式，不会错**。

1. **编码/代码风格**

**1.[注释][强制]**类、类成员、类方法的注释必须使用Javadoc规范，即“/\*\*内容\*/”格式，不使用“//内容”方式

**1.[注释][强制]**所有的类都必须添加创建者和创建日期

**1.[注释][强制]**不使用行尾注释

**1.[注释][建议]**注释掉的代码，要么说明保留的原因，要么删除掉。

**1.[注释][建议]**抽象方法、接口中的方法必须添加注释说明（参数、返回值、完成的功能等）

**1.[大括号][强制]**大括号内容为空简写为“{ }”

**1.[大括号][强制]**大括号内容非空时：

- 左大括号前不换行

- 左大括号后换行

- 右大括号前换行

- 右大括号后还有else/while等代码时不换行，表示块终止时必须换行

**1.[接口][强制]**接口内的变量、常量、方法不添加任何修饰符

**1.[方法摆放][建议]**类内方法摆放顺序，从上至下依次按照：public/protected > private>getter/setter的顺序摆放

**1.[方法摆放][强制]**类内同名方法放在一起（优先级高于1.）

**1.[文件编码][强制]**工程内文本文件编码方式统一设置为UTF-8

**1.[换行][建议]** 每一行文本的长度都应该不超过 100 个字符（超过IDE的行宽提示线，就可以考虑换行）

**1.[换行][建议]**换行时的断开规则

- 在非赋值运算符处换行，那么在该符号前断开(比如+，它将位于下一行)

- 在赋值运算符处断开，建议在该符号后断开(比如=，它与前面的内容留在同一行)

- foreach语句中的分号同赋值运算符

- 方法名或构造函数名与左括号留在同一行

- 逗号“,”与其前面的内容留在同一行

**1.[Long][强制]** long 或者 Long 初始赋值时， 使用大写的 L，不能是小写的 l

**1.[数组类型][强制]** 定义数组类型时中括号是类型的一部分，与类名放在一起：String[] args 而非 String args[]

**1.[日志打印][建议]** 类打印日志定义private final static String TAG，并使用类名的字符串常量命名

**1.[代码格式化][建议]** 新代码提交前先格式化代码（IDE统一代码格式/Ctrl+Shift+F）

1. **编码规约**

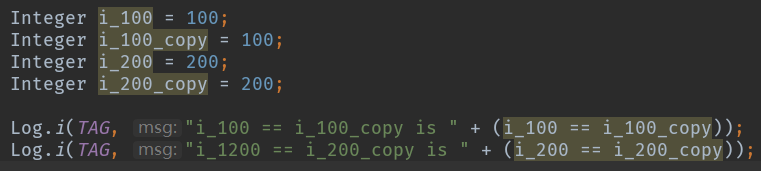
**Java 篇**

**1.[finalize][强制]** 不要重载finalize方法

**1.[导包][强制]**import导包不使用通配符“\*”

**1.[复写][强制]** 复写父类方法一定要使用@Override标注

**1.[值比较][强制]** 值比较统一使用equals



**1.[值比较][强制]** equals尽量用确认有值的对象调用

**1.[字符串拼接][强制]** 循环体内字符串拼接使用StringBuilder的append

**1.[Map遍历][建议]** 使用entrySet遍历Map，而不是 keySet 方式进行遍历

**1.[线程创建][强制]** 线程创建时必须指定有意义的线程名

**1.[线程创建][建议]** 尽量避免直接创建线程，建议使用线程池操作线程

**1.[加锁][强制]** 对多个资源、数据库表、对象同时加锁时，需要保持一致的加锁顺序

线程一获取ABC锁之后才能更新数据，线程二加锁顺序也必须是ABC，否则可能出现死锁

**1.[编码习惯][建议]** 表达异常的分支时，少用 if-else ，这种方式可以改写成卫语句

if (condition) {

…

retunrn obj;

}

//else 业务逻辑

**1.[编码习惯][建议]** 异常不要用来做流程/条件控制

**1.[finally** **][建议]** finally 块必须对资源对象、流对象进行关闭，有异常也要做 try-catch

**1.[finally** **][建议]** 不要在 finally 块中使用 return

**1.[编码习惯][建议]** 避免通过一个类的对象引用访问此类的静态变量或静态方法

**1.[switch][建议]** switch语句最好包含一个default，即使什么也不做

**1.[编码习惯][建议]** 不要使用组合声明，比如int a, b。

**1.[编码习惯][强制]** 判断语句的条件如果逻辑很复杂，需要适当分组拆分，并注释说明

If(a != null || a == b && c != null || c == d && (e != null && e.f == f )){

}

**Android 篇**

**1.[组件生命周期][强制]** 避免在UI线程执行耗时操作（Activity#onCreate/ BroadcastReceiver#onReceive()）

**1.[组件生命周期][建议]** 不要在 Activity#onDestroy()内执行释放资源的工作，例如一些工作线程的销毁和停止，因为 onDestroy()执行的时机可能较晚。可根据实际需要，在 Activity#onPause()/onStop()中结合 isFinishing()的判断来执行

**1.[组件生命周期][强制]** Activity或者Fragment中动态注册BroadCastReceiver时，registerReceiver() 和 unregisterReceiver()要成对出现

**1.[UI布局][建议]** 文本大小使用单位 dp，view 大小使用单位 dp。

**1.[UI布局][建议]** 页面布局内层级尽可能的少，可以使用merge/viewstub优化布局，减少视图树种的节点个数、延迟加载某些非必须控件。

**1.[UI布局][强制]** 不能使用 ScrollView 包裹 ListView/GridView/ExpandableListVIew;因为这 样会把 ListView 的所有 Item 都加载到内存中，要消耗巨大的内存和cpu去绘制界面。

**1.[通信][强制]** 不要通过 Intent 在 Android 基础组件之间传递大数据（binder transaction 缓存为 1MB），可能导致 OOM。

**1.[通信][强制]** 子线程中不能更新界面，更新界面必须在主线程中进行，网络操作不能在主线程中调用。

**1.[通信][建议]** 禁止在多进程之间用SharedPreferences共享数据， (MODE\_MULTI\_PROCESS)模式已过时，并不能保证多进程情况下的数据一致，但官方已不推荐。

**1.[文件操作][强制]**禁止硬编码文件路径，使用 Android 文件系统 API 访问文件系统

**1.[文件操作][建议]** 当使用外部存储时，需要检查外部存储的可用性

// 读/写检查

public boolean isExternalStorageWritable() {

String state = Environment.getExternalStorageState();

if (Environment.MEDIA\_MOUNTED.equals(state)) {

return true;

}

return false;

}

// 只读检查

public boolean isExternalStorageReadable() {

String state = Environment.getExternalStorageState();

if (Environment.MEDIA\_MOUNTED.equals(state) ||

Environment.MEDIA\_MOUNTED\_READ\_ONLY.equals(state)) {

return true;

}

return false;

}

**1.[文件操作][建议]** 应用间共享文件时应使用 FileProvider

Call起安装传递file://的case

**1.[文件操作][建议]** SharedPreference 提交数据时，如果仅是为了保存这次修改，并没有立即读取操作时，建议使用apply而非commit。（apply提交内存，commit写入磁盘）

**1.[文件操作][强制]** zip 中不允许../../file 这样的路径，可能被篡改目录结构，造成攻击。

当 zip 压缩包中允许存在"../"的字符串，攻击者可以利用多个"../"在解压时改变 zip 文件存放的位置，当文件已经存在时就会进行覆盖，如果覆盖掉的文件是 so、 dex或者 odex文件，就有可能造成严重的安全问题

**1.[数据库][强制]**多线程操作写入数据库时，需要使用事务，以免出现同步问题。 （事务操作原理）

**1.[Bitmap][建议]** bitmap使用结束后，在 2.3.3 及以下需要调用 recycle()函数，在 2.3.3 以上 GC 会自动管理，把bitmap的引用置空即可，除非明确不需要再用。

四、android