**Google Java代码规范**

[1.简介 2](#_Toc377666256)

[2.源码基本要求 2](#_Toc377666257)

[3. 源码结构 2](#_Toc377666258)

[4. 格式化 3](#_Toc377666259)

[5. 命名 5](#_Toc377666260)

[6. 编程实践： 5](#_Toc377666261)

[7. Javadoc 5](#_Toc377666262)

Created by： 蔡观洋

Time：2014-01-16

1. 简介

Google的编码规范

2.源码基本要求

2.1 文件名：

以文件中第一个公共类的名字加.java后缀组成，名字是大小写敏感的。

2.2 文件编码：

UTF-8

**2.3 特殊字符**

2.3.1 空白符：

处理行结束符，ASCII中的水平空白符（0x20）是唯一的一个可以在文件中出现的空白符。

其他：

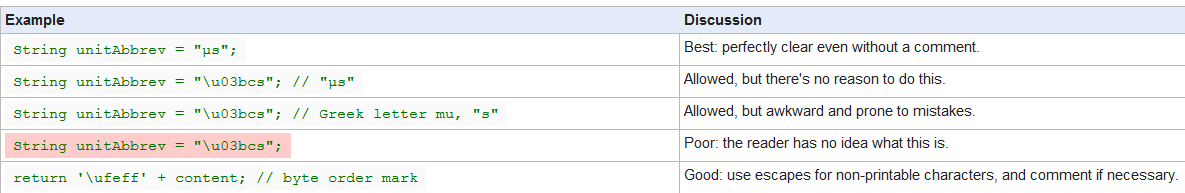
1. 在字符串和字符中的空白符不受此约束
2. 不能使用Tab符缩进代码

2.3.2 特殊序列：

注意转义字符的使用：[\\b](\\\\b), [\\t](file:///\\t), [\\r](file:///\\r), [\\n](file:///\\n).....

2.3.3 非ASCII字符的使用：

使用那些Unicode字符或者对应的字符编码，最好添加合适的注释



3. 源码结构

源码的构成：

1. 代码版权或证书信息
2. package声明
3. import声明
4. 顶层类

每一个模块间用一个空行来分隔；

**3.1 Package声明**

不能出现嵌套行，它不受一行建议字符数的限制

**3.2 Import 声明**

不能使用通配符导入多个包，也不能使用静态导入；

同包声明一样，也不能出现嵌套行；

导入的包分组放置：（顺序如下）

1. 静态导入
2. com.cattsoft…
3. 第三方的包，没一个第三方的包发放置在个小组中，且他们之间按字符顺序排列
4. java imports
5. javax imports

各分组间以空行分隔，分组内按名字排序（ASCII）

**3.3 类声明**

1. 每一个顶级类应该有自己对应的文件

2. 有关类成员如何排序，则没有唯一正确的标准，不同类别的类可能有不同的方式；但是重要的一点是，类成员应该按“逻辑顺序”来排列。开发人员可能大多时候都是习惯性的在文件的末尾添加新方法，但是这是按时间顺序排列的，而不是逻辑顺序。

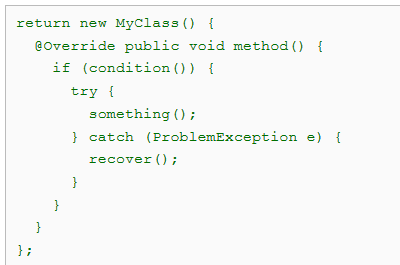
3. 当一个类有多个构造函数，或同名方法时，这些同名方法应按照参数的格式排序。

4. 格式化

**4.1 大括号**

1. if, else, for, do, while都需要使用大括号，即使这些语句体为空或者仅有一条语句；

2．当时一个类，方法，构造体的结束大括号时，后面应接一个新行，但是如果后面为else，或逗号等，则不需要接新行。如：



3. 针对空块：我们可以直接写一对大括号，中间没有字符或者空行分隔，但是含有多个语句块的声明除外，如if/else-if/else try/catch/finally

4.2 块缩进：

缩进宽度为2个空格；

4.3 每个声明占一行

4.4 行字符数限制为80或100：

工程可以任意选择这两个限制中的一个，如果一行的字符数超过了这个限制，需要分行书写。但也有例外：

1. 有些信息行是不可能遵循行字符数限制的，如注释中的URL，一个比较长的JSNI方法引用；
2. Package和import声明
3. 在注释中的一个命令行，可能被复制到shell中

**4.5 分隔行**

1. 解决分隔行的方法是抽取一个函数或者局部变量

2. 如何分隔：

1）当一个非赋值语句需要分隔时，需要在符号之前分隔。如. & |等符号，场景为<T extends Foo & Bar> catch (FooException | BarException e)

2）赋值类型的语句，则需要在符号之后分隔

3）方法或构造函数，应在名字和小括号之后分隔

4）在语句中的逗号之前分隔

3. 注意分隔的行，一般要比原始行缩进至少4个空格，当有多个分隔行时，一般这些行之间的缩进也不一样，除非这两个分隔行是属于平行的语句元素，才会使用相同的缩进。

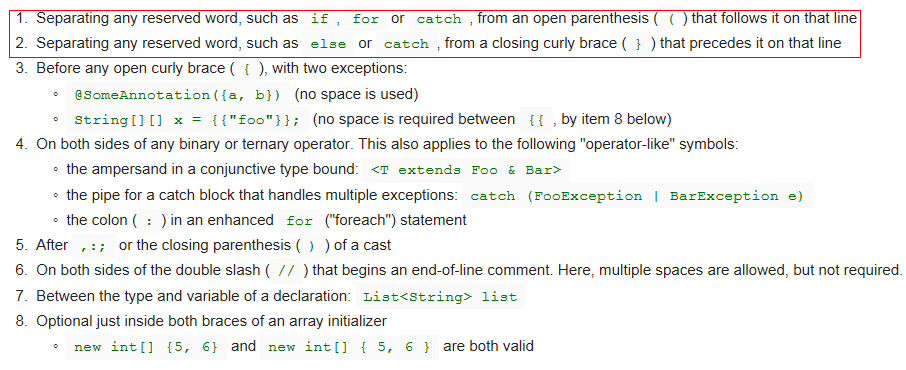
**4.6 空格**

1. 垂直方向：

一个空白行出现的场景：

1. 一个类的成员之间：域，构造函数，方法，静态初始化。以及域中的逻辑分组之间也可以添加空白行
2. 方法体内，逻辑声明之间
3. 文档其他部分说明

2. 水平方向：



4.7 构造规则：

1. 一个声明语句中只有一个变量： 不能使用”int a, b”

2. 局部变量在需要的时候声明并初始化即可，不需要在快的起始部分完成声明。应尽量缩小局部变量的作用域；

3．Switch：

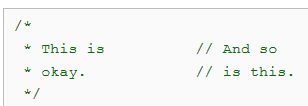
跟块的缩进一致，都为2个空格；

即使没有条件，也应有一个”default”标签；

4. 注解：

每个注解单独占一行，不需要缩进，与注解的内容保持一致即可。

1. 注释



Java源码中用于分隔代码中的各个域

/\* ---------------- Constants -------------- \*/

1. 修饰符书序：



5. 命名

5.1 包名：

包名字都是小写，不能有下划线

5.2 类名：

1. 采用大写驼峰规则

2. 类名为名称或者名词词组，接口则有时可以为形容词或者形容词词组；

3. 测试类则用要测试的类命名，以“Test”结尾，如HashTest

5.3 方法名：

1. 采用小写驼峰规则

2. 方法名一般为动词或动词词组；

3. 在Junit测试中，方法名可以使用下划线来分隔逻辑组件的名字。典型的模式为：

test<MethodUnderTest>\_<state>

5.4 常量：

采用全大写形式。由static final 修饰的，但是由他们修饰的并非都是常量；

5.5 非常量的域名字采用小写驼峰规则，

5.6 参数名字，采用小写驼峰规则，且尽量避免使用一个字符作为参数名字；

5.7 局部变量：

采用小写驼峰规则，且可以尽量采用缩写形式；

但是一个字符的命名方式也要尽量避免，除了临时变量和循环变量；

即使是final修饰的，不可变的变量也不能用常量的规则来命名。

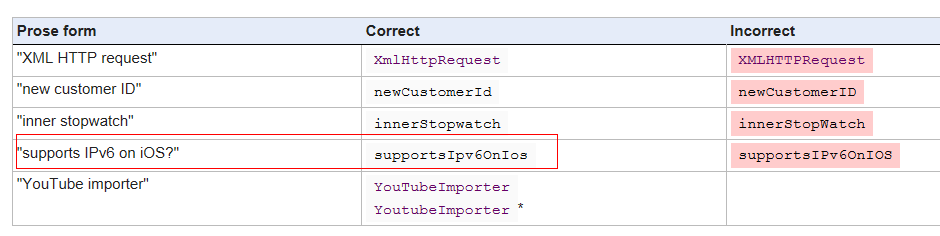
5.8 驼峰命名规则：

1. 将词组全部转换为ASCII字符，并删除所有的省略符号；

2. 将结果分隔为一系列的单词，以空格或标点符号来分。如果一些字符串已经符号驼峰规则，则按规则转换为单词组；

3. 将所有的单词都转为小写。然后仅仅将所有单词的首字符转换为大写（大写驼峰），或者除第一个单词外剩余单词的首字符大写（小写驼峰）；

4. 最后将这些单词连起来作为一个标示符



6. 编程实践：

6.1 @Override 需要的地方一定要用

6.2 捕获异常不要忽略，即使真的不需要对异常做任何处理，也要添加注释说明一下

6.3 一定要通过类名来调用静态成员

6.4 不要使用finalize()

7. Javadoc

7.1 格式：

1. 参考前面介绍的块注释

2. 段落：

段落之间要有以‘\*’开头的空行来分隔，且除了第一行外，没行的首字符由<P>开始；

3. At-clauses:

指@param @return @throws @deprecated，且如果出现此标识，后面内容不能为空；

7.2 总结描述：

每个类或方法的注释总是以一个简短的总结性描述开始，这段描述很重要，他会出现在类或方法的索引提示内容中供用户查看。他是一个名词或动词词组，不是一个完整的句子。也不是命令式的句子。

7.3 什么地方使用Javadoc

在public类，和每一个public和protected修饰的数据成员前，当然也有一些例外：

1. 那些简单的，明显的，自己的名字可以说明它作用的方法不需要注释，如getFoo()；
2. 一些单元测试的方法，从命名上已经说明了用途的，不需要注释；
3. Overrie修饰的方法不需要注释；

当然对于那些私有的成员，也应该有注释说明一下他的作用，实现上的要点等。