**创新事业部JAVA开发手册**

2019年03月

**修改记录**

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 修订日期 | 修订人员 | 修订纪要 |
| 1.0 | 2019-03-23 | 徐强辉 | 初始版本 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**文档目录**

[1. 一级Title1 3](#_Toc17809)

[1.1. 二级Title1 3](#_Toc1549)

[1.2. 二级Title2 3](#_Toc25507)

[1.2.1. 三级Title1 3](#_Toc27634)

[1.2.2. 三级Title2 3](#_Toc13629)

[1.2.2.1. 四级Title 4](#_Toc8803)

[2. 一级Title2 4](#_Toc28876)

[2.1. 二级Title 4](#_Toc12598)

[2.2. 二级Title 4](#_Toc25982)

[2.2.1. 三级Title 5](#_Toc16068)

[2.2.2. 三级Title 5](#_Toc3207)

[2.2.2.1. 四级Title 5](#_Toc8877)

[2.2.2.2. 四级Title 5](#_Toc32092)

# 编程规约

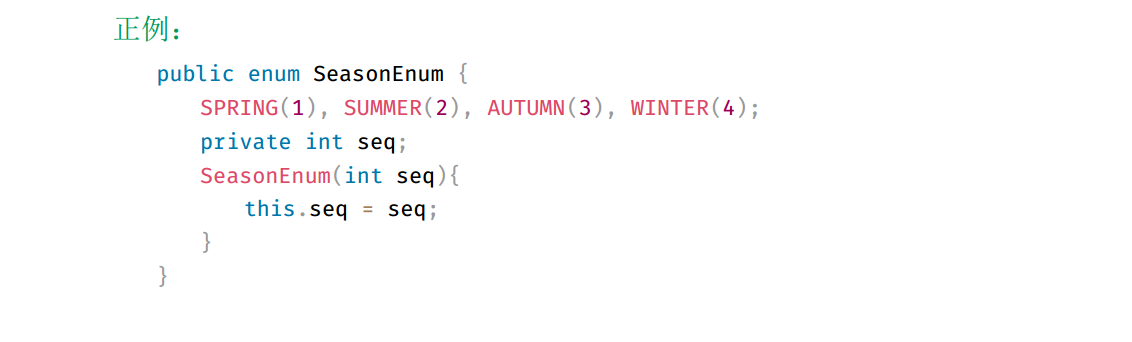
## 命名风格

1. 【强制】 代码中的命名均不能以下划线或美元符号开始，也不能以下划线或美元符号结束。  
   反例： \_name / \_\_name / $name / name\_ / name$ / name\_\_
2. 【强制】 代码中的命名严禁使用拼音与英文混合的方式，更不允许直接使用中文的方式。  
   说明： 正确的英文拼写和语法可以让阅读者易于理解，避免歧义。注意，即使纯拼音命名方式也要避免采用。  
   反例： DaZhePromotion [打折] / getPingfenByName() [评分] / int 某变量 = 3
3. 【强制】类名使用 UpperCamelCase 风格，但以下情形例外： DO / BO / DTO / VO / AO /PO / UID 等。  
   正例： MarcoPolo / UserDO / XmlService / TcpUdpDeal / TaPromotion  
   反例： macroPolo / UserDo / XMLService / TCPUDPDeal / TAPromotion
4. 【强制】方法名、参数名、成员变量、局部变量都统一使用 lowerCamelCase 风格，必须遵从驼峰形式。  
   正例： localValue / getHttpMessage() / inputUserId
5. 【强制】常量命名全部大写，单词间用下划线隔开，力求语义表达完整清楚，不要嫌名字长。  
   正例： MAX\_STOCK\_COUNT  
   反例： MAX\_COUNT
6. 【强制】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的名称开始，以 Test 结尾。
7. 【强制】类型与中括号紧挨相连来表示数组。  
   正例： 定义整形数组 int[] arrayDemo;  
   反例： 在 main 参数中，使用 String args[]来定义。
8. 【强制】POJO 类中布尔类型的变量，都不要加 is 前缀，否则部分框架解析会引起序列化错误。  
   反例： 定义为基本数据类型 Boolean isDeleted 的属性，它的方法也是 isDeleted()， RPC框架在反向解析的时候， “误以为” 对应的属性名称是 deleted，导致属性获取不到，进而抛出异常。
9. 【强制】包名统一使用小写，点分隔符之间有且仅有一个自然语义的英语单词。包名统一使用单数形式，但是类名如果有复数含义，类名可以使用复数形式。  
   正例： 应用工具类包名为 com.unisound.ai.util、类名为 MessageUtils（此规则参考 spring的框架结构）
10. 【强制】杜绝完全不规范的缩写， 避免望文不知义。  
    反例： AbstractClass“缩写” 命名成 AbsClass； condition“缩写” 命名成 condi，此类随意缩写严重降低了代码的可阅读性。
11. 【推荐】为了达到代码自解释的目标，任何自定义编程元素在命名时，使用尽量完整的单词组合来表达其意。  
    正例：在JDK中， 表达原子更新的类名为： AtomicReferenceFieldUpdater。  
    反例： 变量 int a 的随意命名方式。
12. 【推荐】如果模块、 接口、类、方法使用了设计模式，在命名时需体现出具体模式。  
    说明： 将设计模式体现在名字中，有利于阅读者快速理解架构设计理念。  
    正例： public class OrderFactory;  
    public class LoginProxy;  
    public class ResourceObserver;
13. 【推荐】接口类中的方法和属性不要加任何修饰符号（public 也不要加） ，保持代码的简洁性，并加上有效的 Javadoc 注释。尽量不要在接口里定义变量，如果一定要定义变量，肯定是与接口方法相关，并且是整个应用的基础常量。  
    正例： 接口方法签名 void commit();  
    接口基础常量 String COMPANY = "unisound";  
    反例： 接口方法定义 public abstract void f();  
    说明： JDK8 中接口允许有默认实现，那么这个 default 方法，是对所有实现类都有价值的默认实现。
14. 接口和实现类的命名有两套规则：  
    1） 【强制】对于 Service 和 DAO 类，基于 SOA 的理念，暴露出来的服务一定是接口，内部的实现类用 Impl 的后缀与接口区别。  
    正例： CacheServiceImpl 实现 CacheService 接口。  
    2）【推荐】 如果是形容能力的接口名称，取对应的形容词为接口名（通常是–able 的形式）。  
    正例： AbstractTranslator 实现 Translatable 接口。
15. 【参考】枚举类名建议带上 Enum 后缀，枚举成员名称需要全大写，单词间用下划线隔开。  
    说明： 枚举其实就是特殊的类， 域成员均为常量， 且构造方法被默认强制是私有。  
    正例： 枚举名字为 ProcessStatusEnum 的成员名称： SUCCESS / UNKNOWN\_REASON。
16. 【参考】各层命名规约：  
    A) Service/DAO 层方法命名规约  
     1） 获取单个对象的方法用 get 做前缀。  
     2） 获取多个对象的方法用 list 做前缀，复数形式结尾如： listObjects。  
     3） 获取统计值的方法用 count 做前缀。  
     4） 插入的方法用 save/insert 做前缀。  
     5） 删除的方法用 remove/delete 做前缀。  
     6） 修改的方法用 update 做前缀。  
    B) 领域模型命名规约  
     1） 数据对象： xxxDO， xxx 即为数据表名。  
     2） 数据传输对象： xxxDTO， xxx 为业务领域相关的名称。  
     3） 展示对象： xxxVO， xxx 一般为网页名称。  
     4） POJO 是 DO/DTO/BO/VO 的统称，禁止命名成 xxxPOJO。

## 常量定义

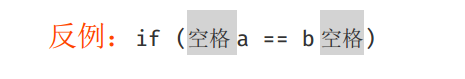
1. 【强制】不允许任何魔法值（即未经预先定义的常量） 直接出现在代码中。  
   反例： String key = "Id#unisound\_" + tradeId;  
   cache.put(key, value);
2. 【强制】 在 long 或者 Long 赋值时， 数值后使用大写的 L，不能是小写的 l，小写容易跟数字  
   1 混淆，造成误解。  
   说明： Long a = 2l; 写的是数字的 21，还是 Long 型的 2?
3. 【推荐】不要使用一个常量类维护所有常量， 要按常量功能进行归类，分开维护。  
   说明： 大而全的常量类， 杂乱无章， 使用查找功能才能定位到修改的常量，不利于理解和维护。  
   正例： 缓存相关常量放在类 CacheConsts 下； 系统配置相关常量放在类 ConfigConsts 下。
4. 【推荐】常量的复用层次有五层：跨应用共享常量、应用内共享常量、子工程内共享常量、包内共享常量、类内共享常量。  
   1） 跨应用共享常量：放置在二方库中，通常是 client.jar 中的 constant 目录下。  
   2） 应用内共享常量：放置在一方库中， 通常是子模块中的 constant 目录下。  
   反例： 易懂变量也要统一定义成应用内共享常量，两位攻城师在两个类中分别定义了表示  
   “是”的变量：  
   类 A 中： public static final String YES = "yes";

类 B 中： public static final String YES = "y";  
A.YES.equals(B.YES)，预期是 true，但实际返回为 false，导致线上问题。  
3） 子工程内部共享常量：即在当前子工程的 constant 目录下。  
4） 包内共享常量：即在当前包下单独的 constant 目录下。  
5） 类内共享常量：直接在类内部 private static final 定义。

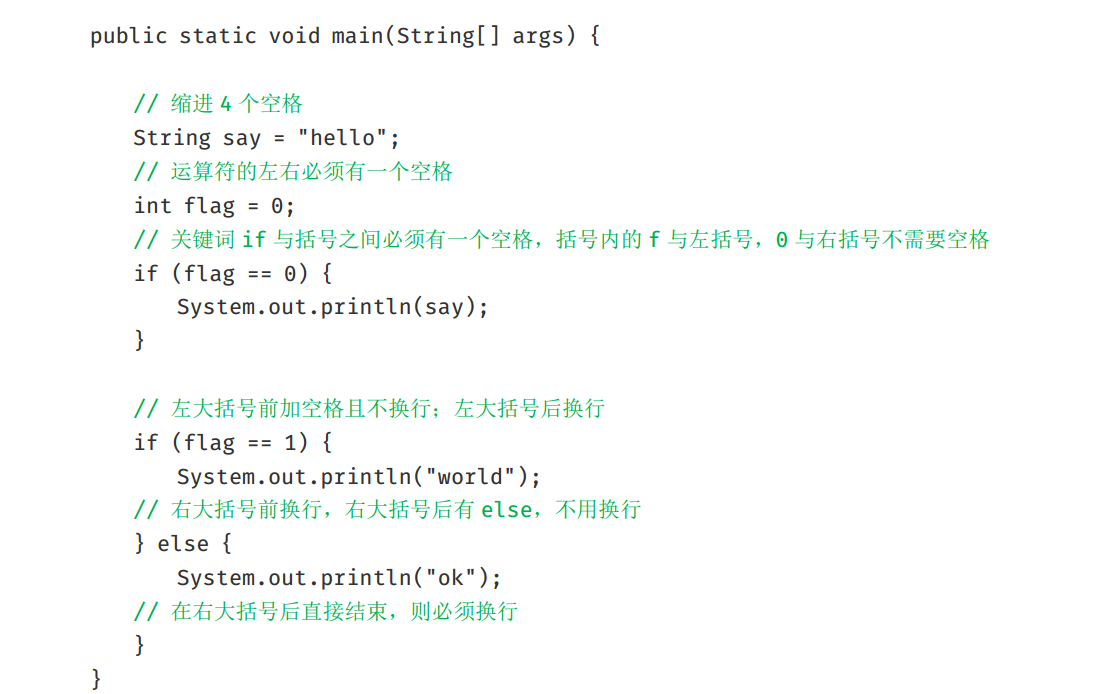
1. 【推荐】 如果变量值仅在一个固定范围内变化用 enum 类型来定义。  
   说明： 如果存在名称之外的延伸属性应使用 enum 类型，下面正例中的数字就是延伸信息，表示一年中的第几个季节.  
   

## 代码格式

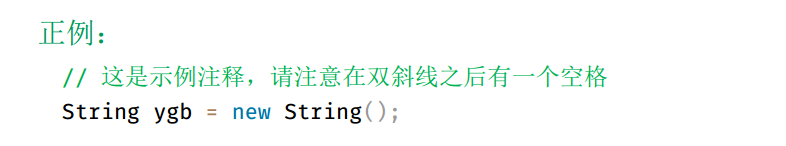
1. 【强制】大括号的使用约定。如果是大括号内为空，则简洁地写成{}即可，不需要换行； 如果  
   是非空代码块则：  
   1） 左大括号前不换行。  
   2） 左大括号后换行。  
   3） 右大括号前换行。  
   4） 右大括号后还有 else 等代码则不换行； 表示终止的右大括号后必须换行。
2. 【强制】左小括号和字符之间不出现空格； 同样，右小括号和字符之间也不出现空格；而左大  
   括号前需要空格。详见第 5 条下方正例提示。



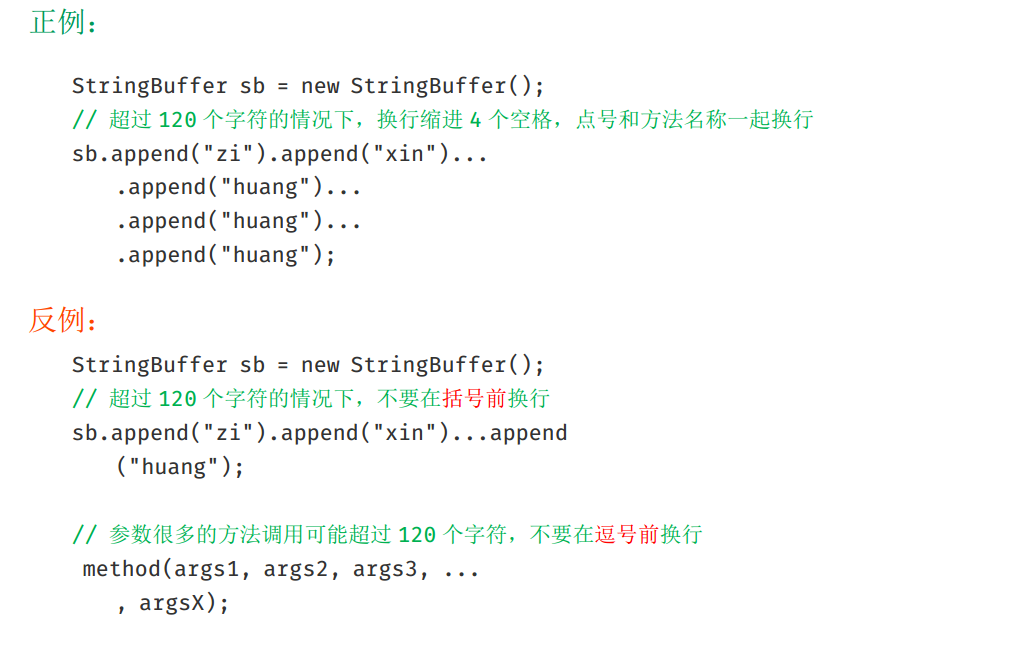
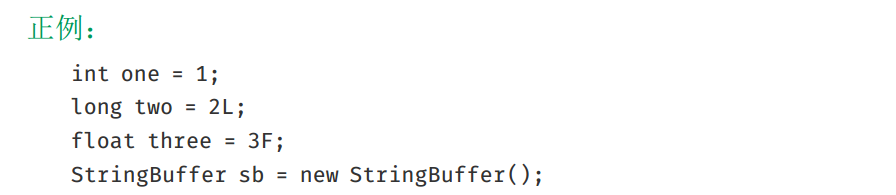
1. 【强制】 if/for/while/switch/do 等保留字与括号之间都必须加空格。
2. 【强制】任何二目、 三目运算符的左右两边都需要加一个空格。  
   说明： 运算符包括赋值运算符=、逻辑运算符&&、加减乘除符号等。
3. 【强制】 采用 4 个空格缩进，禁止使用 tab 字符。  
   说明： 如果使用 tab 缩进，必须设置 1 个 tab 为 4 个空格。 IDEA 设置 tab 为 4 个空格时，  
   请勿勾选 Use tab character；而在 eclipse 中，必须勾选 insert spaces for tabs。  
   正例： （涉及 1-5 点）



1. 【强制】注释的双斜线与注释内容之间有且仅有一个空格。



1. 【强制】单行字符数限制不超过 120 个，超出需要换行，换行时遵循如下原则：  
   1） 第二行相对第一行缩进 4 个空格，从第三行开始，不再继续缩进，参考示例。  
   2） 运算符与下文一起换行。  
   3） 方法调用的点符号与下文一起换行。  
   4） 方法调用中的多个参数需要换行时， 在逗号后进行。  
   5） 在括号前不要换行，见反例。

  
8. 【强制】方法参数在定义和传入时，多个参数逗号后边必须加空格。  
正例： 下例中实参的 args1， 后边必须要有一个空格。  
  
9. 【强制】 IDE 的 text file encoding 设置为 UTF-8; IDE 中文件的换行符使用 Unix 格式，不要使用 Windows 格式。  
10. 【推荐】单个方法的总行数不超过 80 行。  
说明： 包括方法签名、结束右大括号、方法内代码、注释、空行、回车及任何不可见字符的总行数不超过 80 行。  
正例： 代码逻辑分清红花和绿叶， 个性和共性，绿叶逻辑单独出来成为额外方法，使主干代码更加清晰； 共性逻辑抽取成为共性方法，便于复用和维护。  
11. 【推荐】没有必要增加若干空格来使某一行的字符与上一行对应位置的字符对齐。  
  
说明： 增加 sb 这个变量，如果需要对齐，则给 a、 b、 c 都要增加几个空格，在变量比较多的  
情况下，是非常累赘的事情。  
12.【推荐】 不同逻辑、不同语义、不同业务的代码之间插入一个空行分隔开来以提升可读性。  
说明： 任何情形， 没有必要插入多个空行进行隔开。

## OOP规约

## 集合处理

## 并发处理

## 控制语句

## 注释规约

## 其他

# 一级Title2

## 二级Title

## 二级Title

### 三级Title

### 三级Title

#### 四级Title

#### 四级Title

##### 五级Title

##### 五级Title

###### 六级title

###### 六级title