目录

[Java代码规范 2](#_Toc501149841)

[1. 标识符命名规范 2](#_Toc501149842)

[1.1 概述 2](#_Toc501149843)

[1.2 包名 2](#_Toc501149844)

[1.3 类名 2](#_Toc501149845)

[1.4 方法名 3](#_Toc501149846)

[1.5 域（field）名 4](#_Toc501149847)

[1.6 局部变量名 4](#_Toc501149848)

[2. 代码格式 4](#_Toc501149849)

[2.1 源文件编码 4](#_Toc501149850)

[2.2 行宽 4](#_Toc501149851)

[2.3 包的导入 5](#_Toc501149852)

[2.4 域格式 5](#_Toc501149853)

[2.5 代码块格式 5](#_Toc501149854)

[3. 注释规范 5](#_Toc501149855)

[3.1 注释 vs 代码 5](#_Toc501149856)

[3.2 Java Doc 5](#_Toc501149857)

[3.3 块级别注释 6](#_Toc501149858)

[3.4 行内注释 6](#_Toc501149859)

[4 最佳实践和禁忌 6](#_Toc501149860)

[Python 7](#_Toc501149861)

[1.1缩进 7](#_Toc501149862)

[1.2行的最大长度 7](#_Toc501149863)

[1.3空行 7](#_Toc501149864)

[1.4注释 7](#_Toc501149865)

[1.5命名约定 7](#_Toc501149866)

# Java代码规范

## 1. 标识符命名规范

## 1.1 概述

识符的命名力求做到统一、达意和简洁。

1.1.1 统一

统一是指，对于同一个概念，在程序中用同一种表示方法，比如对于供应商，既可以用supplier，也可以用provider，但是我们只能选定一个使用，至少在一个Java项目中保持统一。统一是作为重要的，如果对同一概念有不同的表示方法，会使代码混乱难以理解。即使不能取得好的名称，但是只要统一，阅读起来也不会太困难，因为阅读者只要理解一次。

1.1.2 达意

达意是指，标识符能准确的表达出它所代表的意义，比如： newSupplier, OrderPaymentGatewayService等；而 supplier1, service2，idtts等则不是好的命名方式。准确有两成含义，一是正确，而是丰富。如果给一个代表供应商的变量起名是 order，显然没有正确表达。同样的，supplier1, 远没有targetSupplier意义丰富。

1.1.3 简洁

简洁是指，在统一和达意的前提下，用尽量少的标识符。如果不能达意，宁愿不要简洁

1.1.4 骆驼法则

Java中，除了包名，静态常量等特殊情况，大部分情况下标识符使用骆驼法则，即单词之间不使用特殊符号分割，而是通过首字母大写来分割。

1.1.5 英文 vs 拼音

尽量使用通俗易懂的英文单词，如果不会可以向队友求助，实在不行则使用汉语拼音，避免拼音与英文混用。

### 1.2 包名

使用小写字母如 com.xxx.settlment，不要 com.xxx.Settlement

单词间不要用字符隔开，比如 com.xxx.settlment.jsfutil，而不要com.xxx.settlement.jsf\_util

### 1.3 类名

1.3.1 首字母大写

类名要首字母大写，比如 SupplierService, PaymentOrderAction；不要 supplierService, paymentOrderAction.

1.3.2 后缀

类名往往用不同的后缀表达额外的意思，如下表：

|  |  |  |
| --- | --- | --- |
| 后缀名 | 意义 | 举例 |
| Inter | 这个类是一个接口 | LifeCycleInter |
| Dao | 这个类封装了数据访问方法 | PaymentOrderDao |
| Action | 直接处理页面请求，管理页面逻辑了类 | UpdateOrderListAction |
| Bean | 这是一个POJO | MenuStateBean |

### 1.4 方法名

首字母小写，如 addOrder() 不要 AddOrder()

动词在前，如 addOrder()，不要orderAdd()

动词前缀往往表达特定的含义，如下表：

|  |  |  |
| --- | --- | --- |
| 前缀名 | 意义 | 举例 |
| create | 创建 | createOrder() |
| delete | 删除 | deleteOrder() |
| add | 创建，暗示新创建的对象属于某个集合 | addPaidOrder() |
| Init或者initialize | 初始化，暗示会做些诸如获取资源等特殊动作 | initializeObjectPool |
| write | 写入 | writeUserName() |
| get | 获得 | getName() |
| modity | 修改 | modifyActualTotalAmount() |
| start | 开始 | startOrderProcessing() |
| find | 查找对象 | findNewSupplier() |
| update | 更新对象 | updateCommission() |

find方法在业务层尽量表达业务含义，比如 findUnsettledOrders()，查询未结算订单，而不要findOrdersByStatus()。 数据访问层，find,update等方法可以表达要执行的sql，比如findByStatusAndSupplierIdOrderByName(Status.PAID, 345)

### 1.5 域（field）名

1.5.1 静态常量：全大写用下划线分割

1.5.2 枚举：全大写，用下划线分割

1.5.3 其他：首字母小写，小驼峰法则

### 1.6 局部变量名

参数和局部变量名首字母小写，驼峰法则。尽量不要和域冲突，尽量表达这个变量在方法中的意义。

## 2. 代码格式

用tab缩进源代码，每个缩进4个空格。

### 2.1 源文件编码

源文件使用utf-8编码

### 2.2 行宽

行宽度不要超过130。

### 2.3 包的导入

删除不用的导入，尽量不要使用整个包的导入。

### 2.4 域格式

每行只能声明一个域。域的声明用空行隔开。

### 2.5 代码块格式

2.5.1 缩进风格

大括号的开始在代码块开始的行尾，闭合在和代码块同一缩进的行首，例如：

2.5.2 空格的使用

2.5.2.1 表示分割时用一个空格

2.5.2.2 二元三元运算符两边用一个空格隔开

2.5.2.3 逗号语句后如不还行，紧跟一个空格

2.5.3 空行的使用

空行可以表达代码在语义上的分割，注释的作用范围，等等。将类似操作，或一组操作放在一起不用空行隔开，而用空行隔开不同组的代码

连续两行的空行代表更大的语义分割。

方法之间用空行分割

域之间用空行分割

超过十行的代码如果还不用空行分割，就会增加阅读困难

## 3. 注释规范

### 3.1 注释 vs 代码

注释宜少二精，不宜多而滥，更不能误导

命名达意，结构清晰， 类和方法等责任明确，往往不需要，或者只需要很少注释，就可以让人读懂；相反，代码混乱，再多的注释都不能弥补。所以，应当先在代码本身下功夫。

不能正确表达代码意义的注释，只会损害代码的可读性。

过于详细的注释，对显而易见的代码添加的注释，罗嗦的注释，还不如不写。

注释要和代码同步，过多的注释会成为开发的负担

注释不是用来管理代码版本的，如果有代码不要了，直接删除

### 3.2 Java Doc

表明类、域和方法等的意义和用法等的注释，要以javadoc的方式来写。Java Doc是个类的使用者来看的，主要介绍 是什么，怎么用等信息。凡是类的使用者需要知道，都要用Java Doc 来写。非Java Doc的注释，往往是个代码的维护者看的，着重告述读者为什么这样写，如何修改，注意什么问题等。

### 3.3 块级别注释

3.3.1 块级别注释，单行时用 //, 多行时用 /\* .. \*/。

3.3.2 较短的代码块用空行表示注释作用域

3.3.3 较长的代码块要用

/\*------ start: ------\*/

和

/\*-------- end: -------\*/

### 3.4 行内注释

行内注释用 // 写在行尾

## 4 最佳实践和禁忌

4.1 每次保存的时候，都让你的代码是最美的

程序员都是懒惰的，不要想着等我完成了功能，再来优化代码的格式和结构，等真的把功能完成，很少有人会再愿意回头调整代码。

4.2 每个if while for等语句，都不要省略大括号{}

4.3 善用TODO:

在代码中加入 //TODO: ，大部分的ide都会帮你提示，让你知道你还有什么事没有做。

4.4 在需要留空的地方放一个空语句或注释，告述读者，你是故意的

4.5 不要再对boolean值做true false判断

4.6 减少代码嵌套层次

代码嵌套层次达3层以上时，一般人理解起来都会困难。

4.7 程序职责单一

关注点分离是软件开发的真理。人类自所以能够完成复杂的工作，就是因为人类能够将工作分解到较小级别的任务上，在做每个任务时关注更少的东西。让程序单元的职责单一，可以使你在编写这段程序时关注更少的东西，从而降低难度，减少出错。

4.8 变量的声明，初始化和被使用尽量放到一起

4.9 缩小变量的作用域

能用局部变量的，不要使用实例变量，能用实例变量的，不要使用类变量。

4.10 尽量不要用参数来带回方法运算结果

# Python

1.1缩进

四格缩进，全用tab进行缩进

1.2行的最大长度：80字符

折叠长行的首选方法是使用Pyhon支持的圆括号，方括号和花括号内的行延续。

### 1.3空行

用两行空行分割顶层函数和类的定义，类内方法的定义用单个空行分割，额外的空行可被用于(保守的)分割相关函数组成的群，在一组相关的单句中间可以省略空行。

### 1.4注释

当代码修改时，始终优先更新注释!注释应该是完整的句子，如果注释是一个短语或句子，首字母应该大写，除非他是一个以小写字母开头的标识符(永远不要修改标识符的大小写)。

### 1.5命名约定

类名：类名使用CapWords约定。内部使用的类外加一个前导下划线。

函数名：函数名应该为小写，可能用下划线风格单词以增加可读性。

方法名和实例变量：通常使用小写单词，必要时用下划线分隔增加可读性。

view函数：

返回有对应模板的可是网页命名为

def\_[view]

返回表单提交功能的或接受ajax的功能的视图

def\_[view]\_fun

变量命名：

lower\_case\_with\_underscores（有下划线的小写）

函数内部临时变量

Ref\_var

model中的应用变量

ref

渲染变量

Var\_type