**Java代码规范**

## 1. 标识符命名规范

### 1.1 简洁

简洁是指，在清晰表达含义的前提下，用尽量少的标识符。如果不能清晰表达，宁愿不要简洁。

### 1.2 骆驼法则

Java中，除了包名，静态属性等特殊情况，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。

### 1.3 英文 vs 拼音

尽量使用通俗易懂的英文单词，其次则使用汉语拼音，避免拼音与英文混用。

### 1.4 类名首字母大写

类名使用首字母大写。

### 1.5 后缀

类名可以使用不同的后缀表达指定的含义，如下表：

|  |  |  |
| --- | --- | --- |
| 后缀名 | 含义 | 举例 |
| Service | 表明这个类是个服务类，里面包含了给其他类提同业务服务的方法 | PaymentOrderService |
| Impl | 这个类是一个实现类，而不是接口 | PaymentOrderServiceImpl |
| Dao | 这个类封装了数据访问方法 | PaymentOrderDao |
| Action | 直接处理页面请求，管理页面逻辑的类 | UpdateOrderListAction |
| Listener | 响应某种事件的类 | PaymentSuccessListener |
| Event | 这个类代表了某种事件 | PaymentSuccessEvent |
| Servlet | 一个Servlet | PaymentCallbackServlet |
| Factory | 生成某种对象的工厂类 | PaymentOrderFactory |
| Adapter | 用来连接某种以前不被支持的对象的类 | DatabaseLogAdapter |
| Job | 某种按时间运行的任务 | PaymentOrderCancelJob |
| Bean | 这是一个POJO | MenuStateBean |

### 1.6 方法名

方法名命名方式为首字母小写，动词在前，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，新创建的对象属于某个集合 | addPaidOrder() |
| remove | 删除 | removeOrder() |
| init或则initialize | 初始化，会做诸如获取资源等特殊动作 | initializeObjectPool |
| destroy | 销毁，会做诸如释放资源的特殊动作 | destroyObjectPool |
| open | 打开 | openConnection() |
| close | 关闭 | closeConnection()< |
| read | 读取 | readUserName() |
| write | 写入 | writeUserName() |
| get | 获得 | getName() |
| set | 设置 | setName() |
| prepare | 准备 | prepareOrderList() |
| copy | 复制 | copyCustomerList() |
| modity | 修改 | modifyActualTotalAmount() |
| start | 开始 | startOrderProcessing() |
| stop | 结束 | stopOrderProcessing() |
| send | 发送某个消息或事件 | sendOrderPaidMessage() |
| receive | 接受消息或事件 | receiveOrderPaidMessgae() |
| respond | 响应用户动作 | responseOrderListItemClicked() |
| find | 查找对象 | findNewSupplier() |
| update | 更新对象 | updateCommission() |

### 1.7 静态属性

静态属性使用全大写并用下划线分割。

### 1.8 局部属性名

参数和局部属性名使用首字母小写方式，尽量表达这个属性在方法中的意义。

## 2. 代码格式

### 2.1 代码缩进

代码块采用缩进风格编写，缩进的空格数为4个。

### 2.2不允许把多个短语句写在一行中，即一行只写一条语句。

### 2.3 包的导入

删除闲置未用的包导入，尽量不要使用整个包的导入。

### 2.4 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
* 超过十行的代码如果还不用空行分割，就会增加阅读困难

### 2.5 注释 vs 代码

* 注释语言必须准确、易懂、简洁。防止注释二义性，不能误导其他阅读者。
* 命名达意，结构清晰。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。
* 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，不要注释掉，否则以后没人知道那段注释掉的代码该不该删除。

### 2.6 行内注释

行内注释用 // 写在行尾。

## 3. 最佳实践和禁忌

### 3.1 每次保存的时候，都让你的代码是最简洁的

不要想着，等我完成了功能，再来优化代码的格式和结构，等把功能完成，很少有人会再愿意回头调整代码格式。

### 3.2 使用log而不是System.out.println

log可以设定级别，可以控制输出位置，容易区分是在代码的什么地方打印的，而System.out.print则不行。而且，System.out.print的速度很慢。所以，除非是有意的，否则，都要用log。至少在提交代码时把System.out.print换成log。

### 3.3 每个if、while、for等语句，都不要省略大括号。

大括号 {} 可以使作用范围更明显。

### 3.4 在需要留空的地方放一个空语句或注释，告知其它开发人员。

### 3.5 减少代码嵌套层次。

代码嵌套层次达3层以上时，阅读人理解起来都会比较困难。

**3.6 程序职责单一。**

关注点分离是软件开发的重点。让程序单元的职责单一，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。可以使你在编写程序时关注更少的东西，从而降低难度，减少出错。

### 3.7 缩小属性的作用域。

能用局部属性的，不要使用实例属性，能用实例属性的，不要使用类属性。属性的生存期越短，以后它被误用的机会越小，同一时刻程序员要关注的属性的状态越少。