# 代码规范

-----个人Java代码规范

|  |  |  |
| --- | --- | --- |
| **分类** | **规范** | **具体规范及解释** |
| 标识符  命名规范 | 统一 | 对于同一个概念，在程序中用同一种表示方法 |
| 达意 | 标识符能准确的表达出它所代表的意义 |
| 简洁 | 在统一和达意的前提下，用尽量少的标识符 |
| 骆驼法则（单词之间通过首字母大写来分割） | 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。比如: SupplierName |
| 尽量使用通俗易懂的英文单词， | 尽量使用通俗易懂的英文单词，实在不行使用汉语拼音，避免拼音与英文混用。 |
| 包名命名 | * 使用小写字母如com.xxx.settlment; * 单词间不要用字符隔开，比如  com.xxx.settlment.jsfutil |
| 类名命名 | * 首字母大写 * 类名往往用不同的后缀表达额外的意思首字母大写 |
| 方法名命名 | * 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： |
| 域（field）名命名 | * 静态变量，全大写用下划线分割 * 枚举，全大写，用下划线分割 * 其他，首字母小写，骆驼法则 |
| 局部变量名命名 | 参数和局部变量名首字母小写，骆驼法则。尽量不要和域冲突，尽量表达这个变量在方法中的意义 |
| 代码格式 | 用空格字符缩进源代码 | 不要用tab，每个缩进4个空格 |
| 源文件编码 | 源文件使用utf-8编码，结尾用unix n 分格 |
| 行宽 | 行宽度不要超过130。 |
| 包的导入 | 删除不用的导入，尽量不要使用整个包的导入。在eclipse下经常使用快捷键 ctrl+shift+o 修正导入 |
| 域格式 | 每行只能声明一个域。 域的声明用空行隔开 |
| 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 |
| 空格的使用 | * 表示分割时用一个空格 * 二元三元运算符两边用一个空格隔开，如a + b = c; * 逗号语句后如不还行，紧跟一个空格，如call(a, b, c); |
| 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。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 |
| 注释规范 | 注释宜少而精 | 不宜多而滥，更不能误导 |
| 先在代码本身下功夫 | 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。 |
| 正确表达代码意义 | 不能正确表达代码意义的注释，只会损害代码的可读性 |
| 解释要适当 | 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 |
| 不要的代码要删除，不要注释 | 注释掉后别人不知道知道那段注释掉的代码该不该删除，如果有代码不要了，直接删除，svn会有记录。 |
| 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。 |
| 块级别注释 | 单行时用 //, 多行时用 /\* .. \*/  * 较短的代码块用空行表示注释作用域较长的代码块要用/\*------ start: ------\*/和/\*-------- end: -------\*/包围 * 可以考虑使用大括号来表示注释范围 |
| 行内注释 | 行内注释用 // 写在行尾 |
| 编程规范 | 定义类和方法尽可能短小精悍 | 尽量简洁 |
| 将所有成员变量声明为private类型 | 可保证成员数据的一致性。访问所有成员变量都需要通过它的成员函数来访问。 |
| 语句和表达式 | * 用等价的方法替换重复的表达式 * 在控制结构中尽量使用代码块来代替表达 * 用圆括号清晰地表明运算得顺序 * 在switch语句的最后一个case语句块结束的地方加上一个break语句。 * 判两个对象的值是否相等，使用equals()而不是使用== |
| 构造函数 | 在构造函数中不要饮用非常量方法 |
| 异常处理 | * 使用未检测到的、运行时异常来报告程序中未预测到的逻辑错误 * 使用可检验的异常处理来报告可能发生的程序错误，但是，在正常的程序运行中，这种错误是很少发生的。 * 用return代码报告可预测性的状态变化 * 不要对运行时异常或者错误不做任何处理 * 用一个finally语句块释放资源 |