**Java代码规范**

# 命名规范

## 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。例如: seedMessage

## 包名

使用小写字母如 com.amerisia.ebills ，不要 com.amerisia.Ebills单词间不要用字符隔开。

## 类名

首字母大写类名要首字母大写，比如LCIssueInfoManagerEJB, LCIssueAction，不要lcIssueInfoManagerEJB, lcIssueAction.

## 方法名

首字母小写，如addOrder()不要AddOrder()，动词在前，如addOrder() ， 不 要 orderAdd()；返回布尔型的判断方法一般要求方法名使用单词 is 做前缀，如isPersistent(),isString()。或者使用具有逻辑意义的单词，例如equal 或equals

## 枚举

全大写，用下划线分割，如public enum Events { ORDER\_PAID, ORDER\_CREATED}

## 静态常量

全大写用下划线分割，如public static find String ORDER\_PAID\_EVENT =“ ORDER\_PAID\_EVENT”

## 局部变量名

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

## 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80。Eclipse标准

## 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。

## 代码块格式

* 代码块缩进4个空格，如果使用tab缩进，请设置成1个tab为4个空格。
* 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） 换行时相对上一行缩进4个空格。

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） 在多个参数超长，逗号后进行换行。

5） 在括号前不要换行，见反例。正例：  
  
StringBuffer sb = new StringBuffer();  
  
//超过120个字符的情况下，换行缩进4个空格，并且方法前的点符号一起换行 sb.append("zi").append("xin")….append("huang");  
  
反例：  
StringBuffer sb = new StringBuffer();

# 注释规范

* 1. 注释vs代码 注释宜少而精，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
  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 JavaDoc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. 块级别注释块级别注释，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

# 实践应用

# **使用 log而不是System.out.println()**log 可以设定级别，可以控制输出到哪里，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。

# **每个if while for等语句，都不要省略大括号 {}**看下面的代码：if ( a > b ) a++;如果在以后维护的时候，需要在a > b 时，把 b++，一步小心就会写成：if ( a > b ) a++;b++;这样就错了，因为无论a 和 b 是什么关系，b++ 都会执行。如果一开始就这样写：if ( a > b ){ a++;}

# 不**要再对 boolean 值 做 true false 判断**比如：if ( order.isPaid() == true) {// Do something here}不如写成：if ( order.isPaid()) {//Do something here}后者读起来就很是 if order is paid, 要比if order’s isPaid method returns true,易理解

# 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