**Java代码规范**

**1.标识符命名规范**

**1.1 概述**

标识符的命名力求做到统一、达意和简洁。尽量做到每个人按照规范来，多人开发如一人开发一样。

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。比如：theOrderNameOfTheTargetSupplierWhichIsTransfered 太长，transferedTargetSupplierOrderName则较好，但是transTgtSplOrdNm就不好了。省略元音的缩写方式不要使用，我们的英语往往还没有好到看得懂奇怪的缩写。

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。比如: supplierName, addNewContract，而不是 supplier\_name, add\_new\_contract。

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。比如表示归档，用archive比较好, 用pigeonhole则不好，用guiDang尚可接受。

**1.2 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，比如 com.amerisia.ebills，而不要com.amerisia.ebills \_util

**1.3 类名**

1.3.1 首字母大写

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

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** |
| EJB | 表示这个类为EJB类 | LCIssueInfoManagerEB |
| 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 |
| 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 |
| 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 | 生成某种对象工厂的类 | 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 |
| Wrapper | 这是一个包装类，为了给某个类提供没有的能力 | SelectableOrderListWrapper |
| 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() 。

查询方法要查询的内容在前，条件在后。 如getXxByXx（） 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() |
| remove | 删除 | removeOrder() |
| init或则initialize | 初始化，暗示会做些诸如获取资源等特殊动作 | initializeObjectPool |
| destroy | 销毁，暗示会做些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() |
| calculate | 数值计算 | calculateCommission() |
| do | 执行某个过程或流程 | doOrderCancelJob() |
| dispatch | 判断程序流程转向 | dispatchUserRequest() |
| start | 开始 | startOrderProcessing() |
| stop | 结束 | stopOrderProcessing() |
| send | 发送某个消息或事件 | sendOrderPaidMessage() |
| receive | 接受消息或时间 | receiveOrderPaidMessgae() |
| respond | 响应用户动作 | responseOrderListItemClicked() |
| 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，如 public static find String ORDER\_PAID\_EVENT = “ORDER\_PAID\_EVENT”;

1.5.2 枚举

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

1.5.3 其他

首字母小写，骆驼法则，如： public String orderName;

**1.6 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。

**2.接口（Interface）**

接口的名字取决于接口的主要功能和用途。如果接口是使对象具有某种特定的功能，则接口的名字建议使用可以描述这种功能的形容词（加上-able或者-ible后缀）

例如：Searchable，Sortable，NetworkAccessible等

如果接口不是使对象具有某种特定的功能则建议使用名词或者名词短语（同类的命名规则）。

3.**组件/部件（Component）**

使用完整的英文描述来说明组件的用途，末端应接上组件类型。 okButton, customerList,fileMenu

4.**异常（Exception）**

通常采用字母 e 表示异常。

5.**静态常量字段（static final）**

全部采用大写字母，单词之间用下划线分隔。 MIN\_BALANCE, DEFAULT\_DATE

6.**空白的使用相关**

**11.1空行规范**

空行的使用有益于将代码按照逻辑分段，提高代码的可读性。在下列情况下建议使用一个空行：

* 在版权声明块、包声明块、引用声明块之后；
* 在类的声明之间；
* 在方法的声明之间；
* 在类中声明最后一个属性之后，声明第一个方法之前。

**11.2空格规范**

下列情况建议使用单个空格（不是Tab）:

在一个关键字和它所修饰的块之间。可用于如下的关键字：catch，for，if，switch，synchronized，while 。下列关键字的后面不要跟随空格：super ，this 。

在一个参数列表中逗号之后。

空格不应该置于方法名与其左括号之间

左括号和后一个字符之间不应该出现空格

右括号和前一个字符之间也不应该出现空格

例如：

public setMethod(int i, String j)

**11.3缩进规范**

行的缩进要求是四个空格。由于在使用不同的源代码管理工具时Tab字符将因为用户设置的不同而扩展为不同的宽度，所以请不要使用Tab键。建议修改各管理工具的设置将Tab字符扩展成4个空格。

**7.循环计数器**

通常采用字母 i，j，k 或者 counter 都可以接受。 i, j, k, counter

**12.1Java 注释约定** *一个很好的可遵循的有关注释的经验法则是：问问你自己，你如果从未见过这段代码， 要在合理的时间内有效地明白这段代码，你需要哪些信息。*  
 **·一般概念**

注释应该增加代码的清晰度  
保持注释的简洁  
在写代码之前写注释（建议）  
对于复杂点的逻辑，注释出为什么做了一些事，而不仅仅是做了什么

**· 示范**

**文档注释** 在紧靠接口、类、成员函数和字段声明的前面注释它们。

**单行** 在成员函数内采用单行注释，来说明业务逻辑、代码段和暂时变量的声明。注释符"//"后必须紧跟一个空格，然后才是注释信息。

**·注释哪些部分  
类** 类的目的、即类所完成的功能，注释出采用的不变量。  
**接口** 设置接口的目的、它应如何被使用以及如何不被使用。  
**成员函数注释** 对于设置与获取成员函数，在成员变量已有说明的情况下，可以不加注释；普通成员函数要求说明完成什么功能，参数含义是什么返回什么；   
普通成员函数内部注释 控制结构，代码做了些什么以及为什么这样做，处理顺序等。  
实参/参数 参数含义、及其它任何约束或前提条件  
字段/属性 字段描述  
局部变量 无特别意义的情况下不加注释

**8.Java 文件样式约定** *所有的 Java(\*.java) 文件都必须遵守如下的样式规则：***1) 版权信息**版权信息必须在 java 文件的开头，比如：   
/\*\* \* Copyright ? 2000 Shanghai XXX Co. Ltd. \* All right reserved. \*/  
其他不需要出现在 javadoc 的信息也可以包含在这里。  
**2) Package/Imports**package 行要在 import 行之前，import 中标准的包名要在本地的包名之前，而且按照字母顺序排列。如果 import 行中包含了同一个包中的不同子目录，则应该用 \* 来处理。(如果有特殊要求，可以不采用\*来处理，有时候那样更加明确使用了那些类)   
package hotlava.net.stats;import java.io.\*;import java.util.Observable;import hotlava.util.Application;  
这里 java.io.\* 是用来代替InputStream and OutputStream 的。  
**3) Class**接下来的是类的注释，一般是用来解释类的。   
接下来是类定义，包含了在不同的行的 extends 和 implements   
public class CounterSet extends Observable implements Cloneable{……}  
**4) Class Fields**接下来是类的成员变量：   
public 的成员变量必须生成文档（JavaDoc）。  
proceted、private和 package 定义的成员变量如果名字含义明确的话，可以没有注释。（建议都加注释，以便更明确，一目了然）  
**5) 存取方法（类的设置与获取成员函数）**  
接下来是类变量的存取的方法。  
**6) 构造函数**接下来是构造函数，它应该用递增的方式写（比如：参数多的写在后面）。   
public CounterSet(int size){  
 this.size = size;  
}  
**7) 类方法 （类的普通成员函数）**  
下面开始写类的方法：  
**8) toString 方法**   
一般情况下，每一个类都应该定义 toString 方法：   
public String toString() { ……}  
**9) main 方法**   
普通类，考虑置入一个main()方法，其中包含用于测试那个类的代码，如果包含了main() 方法, 那么它应该写在类的底部。

**9. Java编码其它约定  
  
文档化** 必须用 javadoc 来为类生成文档。不仅因为它是标准，这也是被各种 java 编译器都认可的方法。  
**缩进**   
缩进应该是每行2个（或4个）空格。 尽量不要在源文件中保存Tab字符， 在使用不同的源代码管理工具时Tab字符将因为用户设置的不同而扩展为不同的宽度。   
如果你使用 UltrEdit 作为你的 Java 源代码编辑器的话，你可以通过如下操作来禁止保存Tab字符， 方法是通过 UltrEdit中先设定 Tab 使用的长度室2个空格，然后用 Format|Tabs to Spaces 菜单将 Tab 转换为空格。   
**页宽 换行**  
页宽应该设置为80字符。 源代码一般不会超过这个宽度， 并导致无法完整显示， 但这一设置也可以灵活调整。 在任何情况下， 一条语句折行后， 应该比原来的语句再缩进2个字符。

可以依据如下一般规则断开：   
在一个逗号后面断开   
在一个操作符前面断开   
宁可选择较高级别(higher-level)的断开，而非较低级别(lower-level)的断开   
新的一行应该与上一行同一级别表达式的开头处对齐   
如果以上规则导致你的代码混乱或者使你的代码都堆挤在右边，那就代之以缩进8个空格。  
**{} 对**   
{} 中的语句应该单独作为一行。   
例如， 下面的第1行是错误的， 第2行是正确的:   
if (i>0) { i ++ }; // 错误, { 和 } 在同一行   
// 下面写法正确, 单独作为一行  
if (i>0) {   
 i ++   
};   
**括号**   
左括号和后一个字符之间不应该出现空格； 同样， 右括号和前一个字符之间也不应该出现空格。  
下面的例子说明括号和空格的错误及正确使用:   
CallProc( AParameter ); // 错误   
CallProc(AParameter); // 正确   
不要在语句中使用无意义的括号，括号只应该为达到某种目的而出现在源代码中。  
**JSP文件命名**  
采用完整的英文描述说明JSP所完成的功能，尽可能包括一个生动的动词，第一个字母小写，如：viewMessage.jsp、editUser.jsp或者forumChooser.jsp等。  
**Servlet类命名**  
一般对应于所服务的对象加后缀Service来命名，如：UserService，TradeService等。

**10.编程建议  
使用 StringBuffer 对象**   
 在处理 String 的时候要尽量使用 StringBuffer 类，StringBuffer 类是构成 String 类的基础。String 类将 StringBuffer 类封装了起来，（以花费更多时间为代价）为开发人员提供了一个安全的接口。当我们在构造字符串的时候，我们应该用 StringBuffer 来实现大部分的工作，当工作完成后将 StringBuffer 对象再转换为需要的 String 对象。比如：如果有一个字符串必须不断地在其后添加许多字符来完成构造，那么我们应该使用 StringBuffer 对象和它的 append() 方法。如果我们用 String 对象代替 StringBuffer 对象的话，会花费许多不必要的创建和释放对象的 CPU 时间。  
**避免太多的使用 synchronized 关键字**   
 避免不必要的使用关键字 synchronized，应该在必要的时候再使用它，这是一个避免死锁的好方法。 必须使用时，也尽量控制范围，最好在块级控制。  
**避免使用 java.util.Vector 类**  
 因为"Unlike the new collection implementations, Vector is synchronized."，所以使用java.util.Vector类在性能上会有所减低。  
尽量使用接口而不是一个具体的类。(即面向接口编程)  
**每行声明变量的数量(Number Per Line)**

推荐一行一个声明，因为这样以利于写注释

**语句相关规范**

**·简单语句**

赋值和表达式规范 **：** 每行只包含一条语句。

本地变量：如果本地变量声明后不再变动应当声明为final，使编译器生成更有效率的代码。

数组的声明：数组的[]应该紧跟类型名，而不是数组名。在“[”之前不能有空格。

return语句规范：return语句后面的返回值不可以用括号包起来，除非是后面是一个表达式。