**软件编码规范-JAVA**

昆山华恒工程技术中心有限公司

**2017年3月**

**文件修订记录**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 变更版本 | 修订日期 | 原因与修改情况描述 | 位置（页/段落/章节号 | 修订人 | 审核人 |
| V1.0 | 2017.3 | 新建 | 全文 | 郑博 |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

# 命名规范

除了以下几个特例之外，命名时应始终采用完整的英文描述符。此外，一般应采用小写字母，但类名、接口名以及任何非初始单词的第一个字母要大写。

## 1.1 一般概念

* 尽量使用完整的英文描述符
* 采用适用于相关领域的术语
* 采用大小写混合使名字可读
* 尽量少用缩写，但如果用了，要明智地使用，且在整个工程中统一
* 避免使用长的名字（小于 15 个字母是个好主意）
* 避免使用类似的名字，或者仅仅是大小写不同的名字
* 避免使用下划线（除静态常量等）

## 1.2 示范

**包**（Package）采用完整的英文描述符，应该都是由小写字母组成。在部门内部应该规划好包名的范围，防止产生冲突。部门内部产品使用部门的名称加上模块名称。产品线的产品使用产品的名称加上模块的名称。

格式：

com.cvicse.产品名.模块名称

com.cvicse.部门名称.项目名称

**类**（Class）采用完整的英文描述符，所有单词的第一个字母大写，其余字母使用小写的大小写混合法。

示例：OrderInformation, CustomerList, LogManager, LogConfig, SmpTransaction

**接口**（Interface）采用完整的英文描述符说明接口封装，所有单词的第一个字母大写，其余字母使用小写的大小写混合法。

**方法**（Method）采用完整的英文描述符说明接口封装，所有单词的第一个字母大写，其余字母使用小写的大小写混合法。

示例：

private void calculateRate();

public void addNewOrder();

**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。如：

okButton

customerList

fileMenu

**异常**（Exception）通常采用字母 e 表示异常。 e

**类变量字段**采用完整的英文描述，第一个字母小写，任何中间单词的首字大写，如： firstName

lastName

**实参/参数**同字段/属性的命名规则

public void setFirstName(String firstName){

this.firstName = firstName;

}

**局部变量**同字段/属性的命名规则

**获取成员函数**被访问字段名的前面加上前缀 get。 getFirstName(), getLastName()

**布尔型**的获取成员函数所有的布尔型获取函数必须用单词 is 做前缀。 isPersistent(), isString()

**设置成员函数**被访问字段名的前面加上前缀 set。 setFirstName(), setLastName(),setWarpSpeed()

**普通成员函数**采用完整的英文描述说明成员函数功能，第一个单词尽可能采用一个生动的动词，第一个字母小写。 openFile(), addAccount()

**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

**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

**数组**应该总是用下面的方式来命名： byte[] buffer;

# 2.注释规范

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。

## 2.1.一般概念

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

注释出为什么做了一些事，而不仅仅是做了什么

## 2.2. 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。

/\*\*

\*

\* 客户：客户是我们将服务和产品卖给的人或机构。

\*/

* **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。如：

// 遵照 Sarek 的规定，给所有

// 超过 $1000 的发货单

// 打 5% 的折扣。让利活

// 动于 1995年 2 月开始.

## 2.3. 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

# 3.排版规范

## 3.1.一般概念

程序块要采用缩进风格编写，缩进的空格数为4个。

## 3.2.示范

* 分界符（如大括号‘{’和‘}’）应各独占一行并且位于同一列，同时与引用它们的语句左对齐。在函数体的开始、类和接口的定义、以及if、for、do、while、switch、case语句中的程序都要采用如上的缩进方式。

示例：如下例子不符合规范。

for (...) {

... // program code

}

if (...)

{

... // program code

}

void example\_fun( void )

{

... // program code

}

应如下书写：

for (...)

{

... // program code

}

if (...)

{

... // program code

}

void example\_fun( void )

{

... // program code

}

* 不允许把多个短语句写在一行中，即一行只写一条语句

示例：如下例子不符合规范。

LogFilename now = null; LogFilename that = null;

应如下书写：

LogFilename now = null;

LogFilename that = null;

* 相对独立的程序块之间、变量说明之后必须加空行。

示例：如下例子不符合规范。

if(log.getLevel() < LogConfig.getRecordLevel())

{

return;

}

LogWriter writer;

应如下书写：

if(log.getLevel() < LogConfig.getRecordLevel())

{

return;

}

LogWriter writer;

int index;

* 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); // 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。

* 比较操作符, 赋值操作符"="、 "+="，算术操作符"+"、"%"，逻辑操作符"&&"、"&"，位域操作符"<<"、"^"等双目操作符的前后加空格。

if (current\_time >= MAX\_TIME\_VALUE)

a = b + c;

a \*= 2;

a = b ^ 2;

# 4.文件样式规范

所有的 Java(\*.java) 文件都必须遵守如下的样式规则：

* **版权信息**

版权信息必须在 java 文件的开头，比如：

/\*\*

\* @description：描述

\* @author: zhang\_shan

\* @version: 1.0

\* @Create date: 2008-12-22

\* @Copyright (c) CVIC SE.

\*/

其他不需要出现在 javadoc 的信息也可以包含在这里。

* **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。

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。

* **Class**

接下来的是类的注释，一般是用来解释类的。

/\*\*

\* A class representing a set of packet and byte counters

\* It is observable to allow it to be watched, but only

\* reports changes when the current set is complete

\*/

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{

……

……

}

* **Class Fields**

接下来是类的成员变量：

/\*\*

\* Packet counters

\*/ protected int[] packets;

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。

* **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。它只是简单的用来将类的变量赋值获取值的话，可以简单的写在一行上，如类的成员变量已经有注释，类变量的存取方法可以没有注释。

public int[] getPackets() { return this.packets; }   
public void setPackets(int[] packets) { this.packets = packets; }   
……

　　要求说明的是，对于集合，加入成员函数来插入和删除项；另其它的方法不要写在一行上。

* **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;}

* **克隆方法**

如果这个类是可以被克隆的，那么下一步就是 clone 方法：

public Object clone() { try { …… }catch(CloneNotSupportedException e) { …… }}

* **类方法（类的普通成员函数）**

下面开始写类的方法：

/\*\*

\* Set the packet counters

\* param r1 - ……

\* param r2 - ……

\* ……

\*/

protected final void setArray(int[] r1, int[] r2, int[] r3, int[] r4) throws IllegalArgumentException{   
// Ensure the arrays are of equal size

……

}

# 5.编码规范

* 明确方法功能，精确（而不是近似）地实现方法设计。一个函数仅完成一件功能，即使简单功能也应该编写方法实现。

说明：虽然为仅用一两行就可完成的功能去编方法好象没有必要，但用方法可使功能明确化，增加程序可读性，亦可方便维护、测试。

* 应明确规定对接口方法参数的合法性检查应由方法的调用者负责还是由接口方法本身负责，缺省是由方法调用者负责。

说明：对于模块间接口方法的参数的合法性检查这一问题，往往有两个极端现象，即：要么是调用者和被调用者对参数均不作合法性检查，结果就遗漏了合法性检查这一必要的处理过程，造成问题隐患；要么就是调用者和被调用者均对参数进行合法性检查，这种情况虽不会造成问题，但产生了冗余代码，降低了效率。

* 明确类的功能，精确（而不是近似）地实现类的设计。一个类仅实现一组相近的功能。

说明：划分类的时候，应该尽量把逻辑处理、数据和显示分离，实现类功能的单一性。

比如：

数据类不能包含数据处理的逻辑。

通信类不能包含显示处理的逻辑。

* 数据库操作、IO操作等需要使用结束close()的对象必须在try -catch-finally 的finally中close()。

示例：

try

{

// ... ...

}

catch(IOException ioe)

{

//... ...

}

finally

{

try

{

If（null!=out）

{

out.close();

}

}

catch (IOException ioe)

{

//... ...

}

}

* 自己抛出的异常必须要填写详细的描述信息。

说明：便于问题定位。

示例：

throw new IOException("Writing data error! Data: " + data.toString());

* 避免使用不易理解的数字，用有意义的标识来替代。涉及物理状态或者含有物理意义的常量，不应直接使用数字，必须用有意义的静态变量来代替。

示例：如下的程序可读性差。

if (state == 0)

{

state = 1;

... // program code

}

应改为如下形式：

private final static int TRUNK\_IDLE = 0;

private final static int TRUNK\_BUSY = 1;

private final static int TRUNK\_UNKNOWN = -1;

if (state == TRUNK\_IDLE)

{

state = TRUNK\_BUSY;

... // program code

}

* 数组声明的时候使用 int[] index ，而不要使用 int index[] 。

说明：使用int index[] 格式使程序的可读性较差

示例：

如下程序可读性差：

public int getIndex()[]

{

....

}

如下程序可读性好：

public int[] getIndex()

{

....

}

# 6.JTEST规范

* **规则（1-2级）**

1. 在switch 中每个 case 语句都应该包含 break 或者 return 。
2. 资源型的关闭必须放在finally中。
3. 不要使用空的for 、if 、while 语句。
4. 在运算中不要减小数据的精度。
5. switch 语句中的 case 关键字要和后面的常量保持一个空格，switch 语句中不要定义case 之外的无用标签。
6. 不要在if 语句中使用等号= 进行赋值操作。
7. 静态成员或者方法使用类名访问，不使用句柄访问。
8. 方法重载的时候，一定要注意方法名相同，避免类中使用两个非常相似的方法名。
9. 不要在ComponentListener.componentResized() 方法中调用 serResize() 方法。
10. 不要覆盖父类的静态方法和私有方法。
11. 不要覆盖父类的属性。
12. 不要使用两级以上的内部类。
13. 把内部类定义成私有类。
14. 掉接口中多余的定义（不使用 public, abstract, static, final 等，这是接口中默认的）。
15. 不要定义不会被用到的局部变量、类私有属性、类私有方法和方法参数。
16. 显式初始化所有的静态属性。
17. 不要使用 System.getenv() 方法。
18. 不要硬编码 ‘\n’和‘\r’作为换行符号。
19. 在处理 String 的时候要尽量使用 StringBuffer 类
20. 不要直接使用 java.awt.peer.\* 里面的接口。
21. 使用 System.arraycopy() ，不使用循环来复制数组。
22. 避免不必要的 instanceof 比较运算和类造型运算。
23. 不要在 finalize() 方法中删除监听器（Listeners）。
24. 在 finalize() 方法中一定要调用 super.finalize() 方法。
25. 在 finalize() 方法中的 finally 中调用 super.finalize() 方法。
26. 进行字符转换的时候应该尽可能的较少临时变量。
27. 使用ObjectStream 的方法后，调用reset() ，释放对象。
28. 线程同步中，在循环里面使用条件测试（使用 while(isWait) wait() 代替 if(isWait) wait()）。
29. 不调用 Thread 类的 resume(), suspend(), stop() 方法。
30. 减小单个方法的复杂度，使用的 if, while, for, switch 语句要在10个以内。
31. 在Servlets中，重用JDBC连接的数据源。
32. 减少在Sevlets中使用的同步方法。
33. 不定义在包中没有被用到的友好属性、方法和类。
34. 没有子类的友好类应该定义成 final 。
35. 没有被覆盖的友好方法应该定义成 final 。

* **建议（3级及以上）**

1. 为 switch 语句提供一个 default 选项。
2. 不要在 for 循环体中对计数器的赋值。
3. 不要给非公有类定义 public 构建器。
4. 不要对浮点数进行比较运算，尤其是不要进行 ==, !=运算，减少 >, < 运算。
5. 实现 equals() 方法时，先用 getClass() 或者 instanceof 进行类型比较，通过后才能继续比较。
6. 不要重载 main() 方法用作除入口以外的其他用途。
7. 方法的参数名不要和类中的方法名相同。
8. 除了构建器外，不要使用和类名相同的方法名。
9. 不要定义 Error 和 RuntimeException 的子类，可以定义 Exception 的子类。
10. 线程中需要实现 run() 方法。
11. 使用 equals() 比较两个类的值是否相同。
12. 字符串和数字运算结果相连接的时候，应该把数字运算部分用小括号括起来。
13. 类中不要使用非私有（公有、保护和友好）的非静态属性。
14. 在类中对于没有实现的接口，应该定义成抽象方法，类应该定义成抽象类。
15. 不要显式导入 java.lang.\* 包；
16. 初始化时不要使用类的非静态属性。
17. 显式初始化所有的局部变量。
18. 按照方法名把方法排序放置，同名合同类型的方法应该放在一起。
19. 不要使用嵌套赋值，即在一个表达式中使用多个 = 。
20. 重载 equals() 方法的同时，也应该重载 hashCode() 方法。
21. 工具类（Utility）不要定义构建器，包括私有构建器。
22. 不要在 switch 中使用10个以上的 case 语句。
23. 把 main() 方法放在类的最后。
24. 声明方法违例的时候不要使用 Exception ，应该使用它的子类。
25. 不要直接扔出一个Error ，应该扔出它的子类。
26. 在进行比较的时候，总是把常量放在同一边（都放在左边或者都放在右边）。
27. 在可能的情况下，总是为类定义一个缺省的构建器。
28. 在捕获违例的时候，不使用 Exception, RuntimeException, Throwable，尽可能使用它们的子类。
29. 在接口或者工具类中中定义常量。
30. 使用大写‘L’表示 long 常量。
31. 对返回类型为 boolean 的方法使用 is 开头，其它类型的不能使用。
32. 方法需要有同样数量参数的注释 @param。
33. 不要在注释中使用不支持的标记，如：@unsupported 。
34. 不要使用 Runtime.exec() 方法。
35. 不要自定义本地方法（native method）。
36. 使用尽量简洁的的运算符号。
37. 使用集合时设置初始容量。
38. 单个首字符的比较使用 charAt() 而不用 startsWith() 。
39. 对于被除数或者被乘数为2的n次方的乘除运算使用移位运算符 >>, << 。
40. 一个字符的连接使用‘ ’而不使用 “ ”，如：String a = b + 'c'。
41. 不要使用不必要的布尔值比较，如：if (a.equals(b)), 而不是 if (a.equals(b)==true)。
42. 常量字符串使用 String, 非常量字符串使用 StringBuffer 。
43. 在循环条件判断的时候不要使用复杂的表达式。
44. 对于“if (condition) do1; else do2;”语句使用条件操作符“if (condition)?do1:do2;”。
45. 使用StringBuffer的时候设置初始容量。
46. 尽可能的使用局部变量进行运算。
47. 不要使用 Date[] 而要使用 long[] 替代。
48. 不要显式调用 finalize() 。
49. 不要使用静态集合，其内存占用增长没有边界。
50. 不要重复调用一个方法获取对象，使用局部变量重用对象。
51. 线程同步中，使用 notifyAll() 代替 notify()。
52. 避免在同步方法中调用另一个同步方法造成的死锁。
53. 非同步方法中不能调用 wait() , notify() 方法。
54. 使用 wait(), notify() 代替 while(), sleep() 。
55. 不要使用同步方法，使用同步块。（5级）
56. 把所有的公有方法定义为同步方法。（5级）
57. 实现的 Runnable.run() 方法必须是同步方法。 （5级）
58. 一个只有abstract方法、final static 属性的类应该定义成接口。
59. 在 clone() 方法中应该而且必须使用 super.clone() 而不是 new 。
60. 常量必须定义为 final 。
61. 在 for 循环中提供终止条件。
62. 在 for, while 循环中使用增量计数。
63. 使用 StringTokenizer 代替 indexOf() 和 substring() 。