**JAVA编码规范**

**一、命名规范**

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。任何类名,变量名最大长度25个字符以内。

**1.1. Package包的命名**

包(Package) 采用完整的英文描述符,应该都是由小写字母组成。对于全局包,将你的 Internet 域名反转并接上包名。

如: com.yourdomain.project(yourdomain:你的域名,yourproject:你的项目名称)

**1.2. Class类的命名**

采用完整的英文描述符,所有单词的第一个字母大写。 如:Customer,UserAccount

**1.3. Interface的命名**

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。类的最前面一个字符是I,如:IUser.java。

**1.4. Abstract类的命名**

由于abstract类相对比较少,但是他是非常重要,有别于别的类,采用前面加Abstract方式。如: AbstractUser。

**1.5. Class类变量的命名**

变量的名字必须用一个小写字母开头。后面的单词用大写字母开头。 当超过2个以上英文单词组成,使用前面几个单词的第一个字母小写,后面单词保留,第一个字符采用大写。

如: CustomerInfocustomerInfo,CustomerInfoManager ciManager。

**1.6. static final常量的命名**

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。

如: MIN\_BALANCE, DEFAULT\_DATE。

**1.7. Method方法的命名**

成员函数的命名应采用完整的英文描述符,大小写混合使用:所有中间单词的第一个字母大写。如果牵涉到动作的方法,动作写在前面

如: save(); addAccount()。

**1.8.参数的命名(方法内的参数**)

参数的名字必须和变量的命名规范一致。 使用有意义的参数命名,如果可能的话,使用和要赋值的字段一样的名字。

如:

setCount(int count) {

this.count= count;

}

如果不是set/get的方式的时候如果参数传入跟现有变量的全局参数名称一样, 当该方法不是用来赋值操作的时候,要求开发人员以全局变量为准,尽量采用相同意思的另外的变量名。

**1.9.数组的命名**

数组应该总是用下面的方式来命名:

byte[] bufferArray

而不是:

byte bufferArray[]

**1.10.获取和设置变量的方法**

类中的字段不应被直接访问,应该采用set /get方式获取。要求采用getXXX()方式。当参数是boolean 类型的时候采用isXXX(),设置则采用setXXX()。

**1.11.命名集合**

一个集合,例如数组和矢量,应采用复数命名来表示队列中存放的对象类型。命名应采用完整的英文描述符,名字中所有非开头的单词的第一个字母应大写。名称的最后要说明他是什么类型的集合,如: customerArray ,costomerSet等,后面的采用基础数据类型。

**二、Checkstype验证规范**

采用动态验证工具checkstyle工具验证

1.每个类的最后是一个空白行.

2. 每个类内部必须有@author和@version.

3. 对类中的所有的类变量注释.

4. 验证是否合法的html标签.

5. 要求所有的静态常量要求static final 顺序.

6. 每一个method 必须写名 @param@return @throw 或@exception只检查public方法.

7. 验证static final的变量是大写字符,内部可以用\_和数字.

8. 对于内部变量要求是字符和数字组成,第一个必须是小写字符.

9. 对于所有的方法名要求约束第一个是小写字母,内部是字符和数字组成.

10. 对参数命名也要求标准变量名的约束.

11. 对类名,接口,抽象检查是否是开头是大写字母.

12. 检查import不允许.\*方式.

13. 检查不允许在某个项目里出现的包 如java.util.Vector等.

14. 检查是否倒入了不用的包,还有如java.lang.包内的内容.

15. 约束每个文件的总体长度1500行 .根据不同的项目可以配置

16. 约束一行最大长度,建议开发者80个字符,验证的时候是120个字符

17. 方法的行数一般建议80行内,检查120行左右,不包括注释和空格

18. 检查一个方法不可以出现超过5个参数的方法.

19. 检查for循环里默认要求有初始内容和step内容

20. 验证在‘,’后面必须后空格.

21. 检查不允许出现空的block.

22.检查不该用的{}.

23. 检查long类型申明必须后面跟L.

24. 验证出现System.out.println的地方.console端调用除外.

25. switch必须有default.

26. 不允许出现if( val == true).

**三、Java文件样式**

对于java的注释方式,原则上建议采用/\*\*\*//\*\*/。关于//不推荐放在行末说明,放在上行注释,原则是行采用中文注释

**1.package命名**

必须符合命名规范。

**2.import模块**

跟上一模块保持一个空行

在内部原则上要求所有的倒入class必须采用全名称方式,不允许出现.\*方式，以下是建议:

第一块是sun jdk导入的包,

第二块是第三方的包,如apache的log4j等

第三块是自己公司内部的包

每个块之间建议空一行,每个倒入文件必须占用一行

**3.类局部变量的定义**

原则上要求所有的变量都是private变量.要求对每个变量都有说明信息,采用/\*\*\*/放在变量命名的上面,包括私有变量。对于不改变的采用static final方式。

对于有些变量是不固定的,但是可能需要配置修改的,都采用到公共类常量实现方式.

**4.类的详细说明**

4.1.首先必须说明该类的功能描述,注意问题,现在有限解决的问题,以后将要解决的问题,尽可能的详细,

4.2.必须添加版本信息,最后修改的作者,什么时候修改,修改的原因。

4.3.当这个类是公共类的时候,任何人的改变原则上要求不改变原来程序的结构

采用多态方式实现.

/\*\*

\*该类主要用于数据导出的基础类,每个导出文件的类必须继承他

\* @author Arthur.Xie

\* @version 1.0

\*/

**5.构造方法**

跟方法的说明一样,必须有标准的注释,如果有参数,要求在类说明里说明调用顺序。如

/\*\*

\* 构造方法,单例模式

\* @param size传入实列对象cache的大小,必须是大于0的整数

\*/

**6.方法**

要求按照标准的javadoc方式书写,并且,写清楚内部实现的原理。以及注意的地方,当这个是公共类的时候必须写名这个方法是谁修改的,增加作者描述。

/\*\*

\*详细的描述

\* CREATE BY Arthur.Xie

\* @param key desc

\* @return

\* @throw

\*/

**7.对方法内部的规范尽可能的考虑以代码清晰的方式为主。**

7.1.内部注释采用/\*\*/和//都在要说明的前面一行,独立占用一行。

7.2对于内部的局部变量要求有一定的说明.

7.3对于复杂的block嵌套建议在成对的()和{}的最后增加//end if ;//end while等说明.

7.4缩进缩进应该是每行4个空格.

7.5{}成对出现{}中的语句应该单独作为一行.并在一行的最前面.要准守缩进原则.

7.6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.

7.7赋值在=前后必须空格,

7.8.方法参数的时候要求‘,’后面空一格。

7.9.有任何if的地方要求有{}出现,在任何if转移的地方尽可能多的注释。

7.10.在一个程序块的前后建议都要求空一行。如for循环等。

7.11.尽可能的给代码分段.要求一个方法内的语句不要超过80行(不包括注释),尽可能方法抽取重组。代码检查的时候是120行左右约束.

**四、一些约束原则**

1.类中的字段不应被直接访问

2.注释每一个内部类变量

3.所有的变量要求都初始化,特别是基础数据类型.对象类型申明为null ;

4.对于long类型采用大写L代替l如:long i= 10000L;

5.类中方法中除了log可以采用默认的字符输出外,其余的所有常量的东西必须采用final方式申明,需要配置的改变的申明在公共的常量类里.如果是是不配置的,但是不是这个类私有的必须写到公共方法里.

6.所有代码里不可以出现System.out.println这样的方式,一致采用统一log方式.

7.尽可能的写接口方式实现。

8.exception原则上不要采用通过Exception方式捕获,要定位要具体的异常.

9.关于try-catch-final使用,对于资源类的使用必须加入finally,最后关闭约束.

**五、代码修改规范**

1.不同版本的代码间发生的修改,必须对修改部分增加注释。注释内容包括修改目的、修改人、修改时间。

2.不同版本的代码间发生的修改,必须在文件头体现修改历史;修订历史包括,修改人,修改日期……

3.新版本升级时,旧代码不允许删除,只可以注释;一定版本之后,或一定时间后集中整理代码,评估后决定删除确认的废代码。

4.用多态方式构造新方法.不可以删除原有的方法。

5.提供修改的文件可能涉及到的别的逻辑,提供出来,和整体改动的文件,在版本发布后建议采用.主要考虑测试方面等原因.

新增加的方法注释如下:

/\*\*

\*方法的描述:

\*增加方法的原因

\* @author Arthur.Xie

\* @date 2015-3-1

\*/

在类的上面也要增加说明

当要修改对方的代码最多改动20%左右的时候可以在原有方法内修改,要求保留所有以前的代码,写清楚谁注释掉,几行,然后增加的行数是谁增加的,增加了几行,都要写清楚.

建议在修改的时候考虑重构造新的方法,或者增加多态处理,尽量保证原有代码运行正常.