目录

**[1变量的命名规则](#_Toc18014_WPSOffice_Level1)** **[4](#_Toc18014_WPSOffice_Level1)**

**[2方法的命名规则](#_Toc22385_WPSOffice_Level1)** **[6](#_Toc22385_WPSOffice_Level1)**

**[3类及接口的命名规则](#_Toc18847_WPSOffice_Level1)** **[7](#_Toc18847_WPSOffice_Level1)**

**[4作用域](#_Toc37_WPSOffice_Level1)** **[9](#_Toc37_WPSOffice_Level1)**

**[5注释的编写规则](#_Toc19903_WPSOffice_Level1)** **[10](#_Toc19903_WPSOffice_Level1)**

**[6格式](#_Toc24393_WPSOffice_Level1)** **[11](#_Toc24393_WPSOffice_Level1)**

**[7性能与安全](#_Toc20225_WPSOffice_Level1)** **[13](#_Toc20225_WPSOffice_Level1)**

# 1变量的命名规则

## 1.1常量(包含静态的)

一个或多个英文单词的组合，所有字母均大写，单词之间以“\_”分隔，如：

public static final String UPDATE\_FLAG = “T”；

final double PI = 3.14

一般情况下常量的public等修饰符不可少。

## 1.2类变量（静态变量）及实例变量

一个或多个英文单词的组合，第一个单词的首字母小写，其他单词首字母均大写，其余所有字母均小写。如:

private ProjectSes projectSes;

一般情况下类变量（静态变量）及实例变量的public等修饰符不可少。

## 1.3局部变量

一个或多个英文单词的组合，一般采用Hungaryn naming法（匈牙利定义法），如下：

以下是基本数据类型的前缀列表

|  |  |  |
| --- | --- | --- |
| **前缀** | **含义** | **实例** |
| ch | 表示char类型 | char chTemp; |
| i | 表示int类型 | int iNumber; |
| byte | 表示Byte类型 | byte byteGet; |
| s | 表示short类型 | short sNumber; |
| l | 表示long类型 | long lNumber; |
| f | 表示float类型 | float fCount; |
| d | 表示double类型 | double dPrise; |
| str | 表示String类型 | String strSend; |
| b | 表示boolean类型 | Boolean bFlag; |

其他数据类型的前缀列表

|  |  |  |
| --- | --- | --- |
| **前缀** | **含义** | **实例** |
| col | 表示Collection类型 | Collection colUser = new ArrayList(); |
| Collection colUser = new Vector(); |
| lst | 表示List类型 | List lstUser = new ArrayList(); |
| date | 表示Date类型 | Date dateStart = new Date(); |
| sb | 表示StringBuffer类型 | StringBuffer sbName= new StringBuffer(1024) |

除上述情况以外的数据类型，都以obj作为前缀，后面跟一个能说明变量功能或意义单词作为变量，如：UserVO objUserVO;

## 1.4参数

一个或多个英文单词的组合，第一个单词的首字母小写，其他单词首字母均大写，其余所有字母均小写。如:

public void setProjectVO(ProjectVO projectVO)

public void setUserId(String userId)

建议方法的参数不要超过5个，超过时可以将多个参数合并为一个对象进行传递。

## 1.5其它

一、常用数据类型的变量，采用固定的命名，包括以下几种:

Connection conn；

ResultSet rs；

PreparedStatement pstmt；

Statement stmt；

二、数组变量命名与普通变量命名规则一致，如：

int iProjectId[];

public String userName[];

三、每个变量的声明单独占一行。不能一个类型同时声明两个变量。如：int i, j; 这样的写法是不允许的。

四、不要在代码中出现不使用的变量，如果以后会用到或有其他用途要写上注释说明。

五、类名、变量名中含缩写词组：缩写词组全部大写，如:

String strSQL;

public class ProjectDAO

{

..........

}

# 2方法的命名规则

一、方法命名的基本原则：容易看懂

二、一般的方法名采用两个单词动宾结构形式的名称，两个单词之间不要带其它符号，第二个单词的首字母大写，其它的都小写。

如：readBudget(int budgetId)、deleteBudget(int budgetId)

三、只有一个动词形式的方法名不推荐使用。

四、不容易看明白的方法名或有歧义的方法名可采用多单词的形式，每两个单词之间不要带其它符号，从第二个单词开始，每个单词的首字母大写，其它的都小写。

如：readBudgetByProjectId(int projectId)、readBudgetByProjectIdAndYear(int projectId,int year)

五、方法命名不得采用缩写形式。

# 3类及接口的命名规则

## 3.1一般类名

一个或多个英文单词的组合，所有单词的首字母大写，其余所有字母均小写，

如：

public class ProjectUser

{ .......}

## 3.2值对象

数据库表的逻辑名+VO，如：

表PUB\_DICTIONARY的值对象名为DictionaryVO。

## 3.3 Action

表名/模块名+Action，如：

字典的Action类名为DictionaryAction。

## 3.4 MyBatis配置文件

表名/模块名+SQL.xml，如：

字典模块的MyBatis配置文件命名为DictionarySQL.xml。

## 3.5 DAO

DAO类名为模块名+DAO，如：

字典DAO类名为：DictionaryDAO

## 3.6 Application Service

Application Service类名为模块名+AppService，如：

字典Application Service类名为：DictionaryAppService

## 3.7工具类

工具类的类名为模块名+Util，如：

字典的工具类名为：DictionaryUtil

## 3.8门面类

门面类的类名为模块名+Facade，如：

字典的门面类名为：DictionaryFacade

## 3.9代理类

代理类的类名为模块名+Man，如：

字典的代理类名为：DictionaryMan

## 3.10异常类

异常类的类名为模块名+Exception，如：

字典的异常类名为：DictionaryException

## 3.11接口类

接口类的类名为模块名+Interface，如：

字典的接口类名为：DictionaryInterface

## 3.12接口实现类

接口实现类的类名为实现名+Imp，如：

字典的接口实现类名为：DictionaryImp

# 4作用域

## 4.1类的作用域

类的作用域保持最小范围。供包外其它类引用的类才添加public作用域修饰符。

## 4.2方法的作用域

只供对象或类内部调用的方法必须使用private作用域修饰符。

包外不会调用的方法严禁使用public作用域修饰符。

## 4.3属性的作用域

静态常量（类常量）属性可以使用各种作用域修饰符。

对象属性变量严禁使用public作用域修饰符。

VO对象属性变量必须使用private作用域修饰符。

## 4.4局部变量的作用域

方法内的变量定义应该遵循最小作用域规则。

如：

// iSize只在下面的for循环中使用

int iSize = alItems.size();

for(int i=0;i<iSize;i++){……………………………….}

建议写成：

for(int i=0,iSize = alItems.size();i<iSize;i++){……………………………….}

属性是对象的特征，不要把非对象的属性定义为实例变量。

# 5注释的编写规则

所有类及接口头要写类注释（包含作者、创建日期等），@author、@history中要书写中文名，如果注释的内容比较长则需要加“<br>”。如下:

/\*\*

\* 技改项目业务代表实现类.实现新增项目,删除项目等方法，<br>

\* 提供对表现层的接口.

\* @author 黑白团团队

\* @since JDK1.4

\* @history 2019-6-16 黑白团团队 新建

\*/

对代码块的注释，注释语句放在代码块之上，用”//”进行注释，如果注释语句本身超过一行，则用”/\* \*/”进行注释。

对某一行代码的注释，注释语句放在代码行的后面，用”//”进行注释。

建议代码块之间空一行。如下：

public void insertProject(ProjectVO projectVO) {

// 新增项目信息

iProjectId = Toolkit.getInstance().getNextKey("Project"); // 取项目ID

logger.debug("Generate Project ID : " + iProjectId);

Project objProject = projectHome.create(iProjectId);

objProject.setProjectVO(projectVO);

<空一行>

/\*

从项目VO中取得申请人ID及联系人ID，

并将申请人ID及联系人ID插入到PUB\_ACCESS\_CONTROL表

\*/

projectVO.setProjectId(iProjectId);

this.authProject(projectVO);

}

# 6格式

引入指定的格式化配置文件最新版本，使用Eclipse的格式化功能进行代码格式化。

## 6.1缩进

设定为4 个字符。

## 6.2每代码行的长度

代码行的长度不超过120个字符。

## 6.3大括号

大括号中的“{”与条件在同一行，“}”单独一行，即使大括号中只有一条执行语句也要使用大括号，如：

if (a ！= b) {

return 0 ;

}

## 6.4空行

类文件头注释、package语句、import语句、类头注释、类的属性、方法等之间都空一行。

类的属性与属性之间、方法与方法之间都空一行。

如下：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Copyright (C) 2019 黑白团团队

\* 本软件为 黑白团团队开发。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

<空一行>

package com.comtop.app.project.projectinfo.exception;

<空一行>

import com.comtop.util.BaseException;

<空一行>

/\*\*

\* @author 黑白团团队

\* @history 2019-06-16 张三 黑白团团队

\*/

public class ProjectException extends BaseException {

private String message;

<空一行>

/\*\*

... ...

\*/

public ProjectException(String message) {

super(message);

this.message = message;

}

}

## 6.5空格

“=”、“+”、“==”等二元操作符两边分别空一格。如：

int i = 1;

“，”的后面空一格。如：implements Runnable, cloneable

# 7性能与安全

## 7.1空指针引用(null pointer dereference)

在使用或引用对象前，要先对其进行检查，判断其是否为空。

## 7.2数组引用问题(RETURN ARRAY）

不要直接返回指向包含敏感数据的内部数组的引用。不要传回一个数组，而是数组的拷贝。

## 7.3整数溢出

应该对所有整数计算的结果进行检查，防止整数溢出。

## 7.4对象初始化

对变量的使用不要依赖于初始化。在使用对象之前，应该检查对象的初始化过程。可以采用下面的方法实现：在每个类中都应该有一个在构造器中设置的私有布尔标志，在每个非 static 方法中，代码在任何进一步执行之前都应该检查该标志的值。如果该标志的值为 true ，那么控制应该进一步继续；否则，控制应该抛出一个例外并停止执行。

## 7.5集合对象初始化容量

在初始化StringBuffer、集合类（List、Set、Map）等时，一般要指定对象的初始化容量，避免不必要的空间浪费，提升性能。

StringBuffer的初始化容量比PMD插件检查后给出的建议值大一些的值；

List、Set初始化容量可设置为估计值；

Map初始化容量可设置为估计值的2倍到3倍。

## 7.6简单参数

方法调用时，传入的参数应该是一个具体对象，而不是一个嵌套另一个方法。比如类似“this.a(b(c()))”，要将方法的返回值赋给变量再作为参数传递。

## 7.7 Final类和方法

应该将不允许扩展的类和方法应该声明为 final，这样可以防止系统外的代码扩展类并修改类的行为，避免使用非final的公共静态变量。

## 7.8无用的代码

应该将（除启动应用程序的 main() 方法之外的） main() 方法、未使用的方法以及死代码从应用程序代码中除去。

调试使用的控制台输出语句(System.out.println)在调试结束后要删除。

## 7.9日志级别

使用正确的日志级别输出，注意info、warning、debug、error等级别的使用场合。

禁止使用debug级别输出错误信息。

禁止使用errror级别输出调试信息。

## 7.10 类克隆

除非有明确的需求，否则要保证你定义的类是不可克隆的。要使类不可被复制，只要在每个类里定义如下方法：

public final Object clone()‏

throws java.lang.CloneNotSupportedException{

throw new java.lang.CloneNotSupportedException();

}

如果想让类可克隆并且已经考虑了这一选择的后果，需要在类中定义一个为 final 的克隆方法

public final Object clone()‏

super.clone();

## 7.11 JDBC使用

SQL语句变量使用“？”绑定变量。在DAO中，所有SQL语句中的变量都要使用“？”来绑定变量，禁止直接拼凑变量到SQL语句中；

查询分页数据方法中，避免取所有记录的方式。在DAO中，查询分页数据的方法，不得使用取所有记录出来后再分页的方式，而要使用分页SQL语句包装方法。

SQL语句不得在DAO外书写。不得在DAO类外写好SQL语句后传入DAO执行。

查询数据集合时要使用DAO方式，避免使用EntityBean查询数据集合。

## 7.12 SQL语句

类中包含的SQL语句必须全部大写(包括关键字、表名、字段名等)，如：

SELECT PROJECT\_NAME FROM IMP\_PROJECT WHERE PROJECT\_ID = 1

EJB的EJB-QL中关键字(如WHERE等)必须全部大写, 实体类的别名为英文单词(组合)且单词首字母大写，其余字母全部小写。如：

SELECT OBJECT(User) FROM User AS User WHERE User.Id= ?1

## 7.13使用存储过程

对于复杂的数据库操作应该使用存储过程。

存储过程具有更短运行时间和更短的事务锁，具有更高的运行效率。