Topview Java 编程规范

一、工具及环境

1.开发工具

统一使用 Eclipse 或者 MyEclipse 作为开发工具，工作区间的编码格式使用 UTF-8。

2.环境

统一所使用的 JDK 版本和 JAVAEE 版本，避免产生冲突。

二、命名

包（package）：采用完整的英文描述符，由小写字母组成，对于全局包，将你的 Internet 域名反转接上包名：

```
package com.topview.www.model
```

类（class）采用完整的英文描述符，所有单词的第一个字母大写：

```
class SavingAccount
```

异常（exception）：通常采用字母 e 表示异常：

```
try {
  ...
}catch(Exception 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母大写：

```
String firstName;
String lastName;
```

静态常量字段：全部采用大写字母，单词之间用下划线分隔：

```
public static final int MIN_SIZE = 10;
```

循环计数器：通常采用字母 i，j，k 等：

```
for(int i = 0; i < 10; i++)
```

数组：数组采用下面的方式来命名：

```
byte[] buffer;
String[] array;
```

项目分层：简单示例：

```
com.topview.www.model
    User.java
com.topview.www.dao
    UserDao.java
com.topview.www.service
    UserService.java
com.topview.www.action
    UserAction.java
```

三、注释

编写注释的规则：

1.文件头范例：

```
/**
 * Copyright <year> topview Inc.
 */
```

2.类（接口）范例：

- 必须注明类（接口）的编写目的、用途

```
/**
 * 类或接口作用描述
 * @author xxx <email>
 */
```

3.方法头范例：

- 说明方法的作用
- 说明方法的参数

- 说明方法的返回值
- 说明方法抛出的异常

```
/**
 * 方法的描述
 * @param 参数的描述
 * @return 返回类型的描述
 * @exception 异常信息的描述
 */
```

4.方法体范例：

- 控制流结构说明
- 变量说明
- 复杂代码说明
- 处理顺序说明

```
public void example() {
    //使用双斜杠进行注释
}
```

四、代码格式

1.缩进

使用 Tab 键进行缩进代码

2.空格

- 逗号前面空一个空格
- 所有运算符（“.”、“++”、“--”除外）、赋值符前后必须空一格
- 行尾不能有空格

3.空行

- package 和 import 之间空一行
- import 组之间空一行
- import 语句与类、接口声明之间空一行
- 实例变量定义与方法定义之间空一行
- 方法与方法之间空一行
- 代码块之间空一行
- 文件结尾后空一行
- 超过 15 行代码考虑用空行分割，否则有阅读困难
- 除非有特殊考虑，不要有一行以上的空行

4.拆行

- 一对花括号之间必须折行，数组值列表未超过行规定长度的情况除外
- 每一行最多 120 个字符，如果超过则折行（在 java->code style->formatter->line wrapping 中设置）
- 如果表达式折行，则在低优先级操作符前折行，操作符在新行之首
- 声明方法时参数列折行，在逗号之后，折行后参数列起始与上行参数列平齐
- 方法调用时参数列折行，在逗号之后，折行后参数列缩进 4 格，多次折行时，每行都缩进 4 个空格
- if/for/while 的主体即使只有一行，也应该用花括号{}包括

五、类和接口

1.类和接口名不要和 Java 标准库冲突

2.类与接口的声明顺序：

a.静态成员变量
b.静态初始化块
c.成员变量
d.初始化块
e.构造方法
f.静态成员方法
g.成员方法
h.重载来自 Object 的方法如 hashCode 方法和 toString 方法
i.类型（内部类）
j.同一级别的按照 public、protected、private、defaul 排列

3.类的长度不超过 2000 行。

4.已经不需要再使用的 import 引用要清理掉。

5.类/接口命名参考

命名修饰	*前缀/后缀*	*功能描述*	*备注*
Abstract	前缀	抽象类	如：AbstractHandler
Action	后缀	控制逻辑层	如：UserAction
Service	后缀	逻辑层	如：UserService
Dao	后缀	数据层	如：UserDao
Factory	后缀	工厂类	如：SessionFactory
Impl	后缀	针对接口的实现	如：UserDaoImpl
Exception	后缀	自定义异常	如：MyException
Test	后缀	测试类	如：UserTest
Util	后缀	工具类	如：UUIDUtil
Constants	后缀	常量类	该类不可实例化，不含有方法，只有 public static final 修饰

			的变量

六、方法

1.方法体的大小通常不超过 70 行，最好控制在 50 以下。

2.方法参数控制在 6 个参数以内。

3.方法的命名参考

前缀名	*意义*	*示例*
create	创建	createOrder()
add	增加	addOrder()
delete	删除	deleteOrder()
update	更新	updateOrder()
find	查找	findOrder()
init	初始化	initOrder()
destroy	销毁	destroyOrder()
get	获取	getOrder()
set	设置	setOrder()
is	判断	isOrder()

七、局部变量

1.局部变量名要尽量短，推荐使用缩写。比如 StringBuilder sb。

如这个

```
public String abc(String str){
    AbcObjectSet abcObjectSet = new AbcObjectSet();
    abcObjectSet.setName(str);
    return abcObjectSet.getBrief()
}
```

就不如下面这个好

```
public String abc(String str){
    AbcObjectSet aos = new AbcObjectSet();
    aos.setName(str);
    return aos.getBrief();
}
```

2.尽量在代码块或者代码段的开始就声明变量。不要等到要用该变量

的时候才来声明变量。

```
void myMethod() {
    int int1 = 0;         // beginning of method block

    if (condition) {
        int int2 = 0;     // beginning of "if" block
        ...
    }
}
```

3.避免声明的局部变量覆盖上一级声明的变量。例如，不要在内部代码块中声明相同的变量名：

```
int count;
...
myMethod() {
    if (condition) {
        int count = 0;     // AVOID!
        ...
```

4.避免重复初始化变量。

5.尽量采用 lazy loading 的策略，即在需要的时候才开始创建对象

```
Connection conn=null;
...
try {
    conn = createConnection()
        ...
```

八、异常处理

1.使用 try-catch 时需在 finally 内释放关键资源,例如连接、流等，避免代码发生异常时关键资源泄露，例如：

```
try {
    statements;
} catch (Exception e) {
    statements;
} finally {
    if(conn != null) {
        try {
```

```
            conn.close();
        }catch(Exception e) {
        };
    };
}
```

2.记录异常时不能只记录异常的 message，需记录异常的整个 stack，例如：

```
try {
      statements;
} catch (Exception e) {
      e.printStackTrace();
}
```

3.在所有异常被捕获且没有重新抛出的地方必须写日志

4.如果属于正常异常的空异常处理块必须注释说明原因，否则不允许空的 catch 块