啊对对队项目开发规范

前言

规约依次分为【强制】、【推荐】、【参考】三大类。在延伸信息中，“说明”对规约做了适当扩展和解释；

“正例”提倡什么样的编码和实现方式；

“反例”说明需要提防的雷区，以及真实的错误案例。

本开发规范强烈建议配合阿里发布的配套[Java开发规约IDE插件](https://github.com/alibaba/p3c)使用，以达到更好的使用效果。

本版本为第一版本v1.0.0

编辑于2021/10/29

目录

**(一) 命名风格** ·················································2

**(二) 常量定义** ·················································4

**(三) 代码格式** ·················································4

**(四) 日期时间** ·················································6

**(五) 控制语句** ·················································6

**(六) 注释规约** ·················································6

**(一) 命名风格**

1.【强制】代码中的命名均不能以下划线或美元符号开始，也不能以下划线或美元符号结束。

反例：\_name / \_\_name / $name / name\_ / name$ / name\_\_

2.【强制】所有编程相关的命名严禁使用拼音与英文混合的方式，更不允许直接使用中文的方式。

说明：正确的英文拼写和语法可以让阅读者易于理解，避免歧义。注意，纯拼音命名方式更要避免采用。

正例：ali / alibaba / taobao / cainiao/ aliyun/ youku / hangzhou 等国际通用的名称，可视同英文。

反例：DaZhePromotion [打折] / getPingfenByName() [评分] / String fw[福娃] / int 某变量 = 3

3.【强制】类名使用 UpperCamelCase 风格，但以下情形例外：DO / BO / DTO / VO / AO / PO / UID 等。

正例：ForceCode / UserDO / HtmlDTO / XmlService / TcpUdpDeal / TaPromotion

反例：forcecode / UserDo / HTMLDto / XMLService / TCPUDPDeal / TAPromotion

4.【强制】方法名、参数名、成员变量、局部变量都统一使用 lowerCamelCase 风格。

正例： localValue / getHttpMessage() / inputUserId

5.【强制】常量命名全部大写，单词间用下划线隔开，力求语义表达完整清楚，不要嫌名字长。

正例：MAX\_STOCK\_COUNT / CACHE\_EXPIRED\_TIME

反例：MAX\_COUNT / EXPIRED\_TIME

6.【强制】类型与中括号紧挨相连来表示数组。

正例：定义整形数组 int[] arrayDemo。

反例：在 main 参数中，使用 String args[]来定义

7.【强制】避免在子父类的成员变量之间、或者不同代码块的局部变量之间采用完全相同的命名，使可理解性降低。

说明：子类、父类成员变量名相同，即使是 public 类型的变量也能够通过编译，另外，局部变量在同一方法内的不同代码块中同名也是合法的，这些情况都要避免。对于非 setter/getter 的参数名称也要避免与成员变量名称相同。

反例：

public class ConfusingName {

public int stock;

// 非 setter/getter 的参数名称，不允许与本类成员变量同名

public void get(String alibaba) {

if (condition) {

final int money = 666;

// ...

}

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

// 在同一方法体中，不允许与其它代码块中的 money 命名相同

final int money = 15978;

// ...

}

}

}

class Son extends ConfusingName {

// 不允许与父类的成员变量名称相同

public int stock;

}

8.【强制】杜绝完全不规范的缩写，避免望文不知义。

反例：AbstractClass“缩写”成 AbsClass；condition“缩写”成 condi；Function 缩写”成 Fu，此类随意缩写严重降低了代码的可阅读性。

9.【推荐】在常量与变量的命名时，表示类型的名词放在词尾，以提升辨识度。

正例：startTime / workQueue / nameList / TERMINATED\_THREAD\_COUNT

反例：startedAt / QueueOfWork / listName / COUNT\_TERMINATED\_THREAD

10.【推荐】如果模块、接口、类、方法使用了设计模式，在命名时需体现出具体模式。 说明：将设计模式体现在名字中，有利于阅读者快速理解架构设计理念。

正例： public class OrderFactory; public class LoginProxy; public class ResourceObserver;

11.接口和实现类的命名有两套规则：

1）【强制】对于 Service 和 DAO 类，基于 SOA 的理念，暴露出来的服务一定是接口，内部的实现类用 Impl 的后缀与接口区别。

正例：CacheServiceImpl 实现 CacheService 接口。

2）【推荐】如果是形容能力的接口名称，取对应的形容词为接口名（通常是–able 的形容词）。

正例：AbstractTranslator 实现 Translatable 接口。

12.【参考】各层命名规约：

A) Service/DAO 层方法命名规约

1） 获取单个对象的方法用 get 做前缀。

2） 获取多个对象的方法用 list 做前缀，复数结尾，如：listObjects。

3） 获取统计值的方法用 count 做前缀。

4） 插入的方法用 save/insert 做前缀。

5） 删除的方法用 remove/delete 做前缀。

6） 修改的方法用 update 做前缀。

B) 领域模型命名规约

1） 数据对象：xxxDO，xxx 即为数据表名。

2） 数据传输对象：xxxDTO，xxx 为业务领域相关的名称。

3） 展示对象：xxxVO，xxx 一般为网页名称。

4） POJO 是 DO/DTO/BO/VO 的统称，禁止命名成 xxxPOJO。

**(二) 常量定义**

1.【强制】不允许任何魔法值（即未经预先定义的常量）直接出现在代码中。

反例：

// 本例中，开发者 A 定义了缓存的 key，然后开发者 B 使用缓存时少了下划线，即 key 是"Id#taobao"+tradeId，导致 出现故障

String key = "Id#taobao\_" + tradeId;

cache.put(key, value);

2.【强制】在 long 或者 Long 赋值时，数值后使用大写字母 L，不能是小写字母 l，小写容易跟 数字混淆，造成误解。

说明：Long a = 2l; 写的是数字的 21，还是 Long 型的 2？

3.【推荐】如果变量值仅在一个固定范围内变化用 enum 类型来定义。

说明：如果存在名称之外的延伸属性应使用 enum 类型，下面正例中的数字就是延伸信息，表示一年中的 第几个季节。

正例：

public enum SeasonEnum {

SPRING(1), SUMMER(2), AUTUMN(3), WINTER(4);

private int seq;

SeasonEnum(int seq) {

this.seq = seq;

}

public int getSeq() {

return seq;

}

}

**(三) 代码格式**

1.【强制】如果是大括号内为空，则简洁地写成{}即可，大括号中间无需换行和空格；如果是非空代码块则：

1） 左大括号前不换行。

2） 左大括号后换行。

3） 右大括号前换行。

4） 右大括号后还有 else 等代码则不换行；表示终止的右大括号后必须换行。

2.【强制】左小括号和右边相邻字符之间不出现空格；右小括号和左边相邻字符之间也不出现空格；而左大括号前需要加空格。详见第 5 条下方正例提示。

反例：if (空格 a == b 空格)

3.【强制】if/for/while/switch/do 等保留字与括号之间都必须加空格。

4.【强制】任何二目、三目运算符的左右两边都需要加一个空格。

说明：包括赋值运算符=、逻辑运算符&&、加减乘除符号等。

5.【强制】采用 4 个空格缩进，禁止使用 Tab 字符。

说明：如果使用 Tab 缩进，必须设置 1 个 Tab 为 4 个空格。IDEA 设置 Tab 为 4 个空格时，请勿勾选 Use tab character；而在 Eclipse 中，必须勾选 insert spaces for tabs。

正例： （涉及 1-5 点）

public static void main(String[] args) {

// 缩进 4 个空格

String say = "hello";

// 运算符的左右必须有一个空格

int flag = 0;

// 关键词 if 与括号之间必须有一个空格，括号内的 f 与左括号，0 与右括号不需要空格

if (flag == 0) {

System.out.println(say);

}

// 左大括号前加空格且不换行；左大括号后换行

if (flag == 1) {

System.out.println("world");

// 右大括号前换行，右大括号后有 else，不用换行

} else {

System.out.println("ok");

// 在右大括号后直接结束，则必须换行

}

}

6. 【强制】注释的双斜线与注释内容之间有且仅有一个空格。

正例：

// 这是示例注释，请注意在双斜线之后有一个空格

String commentString = new String();

7. 【强制】方法参数在定义和传入时，多个参数逗号后面必须加空格。

正例：下例中实参的 args1，后边必须要有一个空格。

method(args1, args2, args3);

8.【推荐】setter 方法中，参数名称与类成员变量名称一致，this.成员名 = 参数名。在 getter/setter 方法中，不要增加业务逻辑，增加排查问题的难度。

反例：

public Integer getData () {

if (condition) {

return this.data + 100;

} else {

return this.data - 100;

}

}

9.【推荐】类成员与方法访问控制从严：

1） 如果不允许外部直接通过 new 来创建对象，那么构造方法必须是 private。

2） 工具类不允许有 public 或 default 构造方法。

3） 类非 static 成员变量并且与子类共享，必须是 protected。

4） 类非 static 成员变量并且仅在本类使用，必须是 private。

5） 类 static 成员变量如果仅在本类使用，必须是 private。

6） 若是 static 成员变量，考虑是否为 final。

7） 类成员方法只供类内部调用，必须是 private。

8） 类成员方法只对继承类公开，那么限制为 protected。

说明：任何类、方法、参数、变量，严控访问范围。过于宽泛的访问范围，不利于模块解耦。思考：如果 是一个 private 的方法，想删除就删除，可是一个 public 的 service 成员方法或成员变量，删除一下，不 得手心冒点汗吗？变量像自己的小孩，尽量在自己的视线内，变量作用域太大，无限制的到处跑，那么你 会担心的。

**(四) 日期时间**

1.【强制】日期格式化时，传入 pattern 中表示年份统一使用小写的 y。

说明：日期格式化时，yyyy 表示当天所在的年，而大写的 YYYY 代表是 week in which year（JDK7 之后 引入的概念），意思是当天所在的周属于的年份，一周从周日开始，周六结束，只要本周跨年，返回的 YYYY 就是下一年。

正例：表示日期和时间的格式如下所示：

new SimpleDateFormat("yyyy-MM-dd HH:mm:ss")

**(五) 控制语句**

1.【强制】在一个 switch 块内，每个 case 要么通过 continue/break/return 等来终止，要么注释说明程序将继续执行到哪一个 case 为止；在一个 switch 块内，都必须包含一个 default语句并且放在最后，即使它什么代码也没有。

说明：注意 break 是退出 switch 语句块，而 return 是退出方法体。

2.【强制】在 if/else/for/while/do 语句中必须使用大括号。

说明：即使只有一行代码，也禁止不采用大括号的编码方式：if (condition) statements;

**(六) 注释规约**

1.【强制】类、类属性、类方法的注释必须使用 Javadoc 规范，使用/\*\*内容\*/格式，不得使用 // xxx 方式。

说明：在 IDE 编辑窗口中，Javadoc 方式会提示相关注释，生成 Javadoc可以正确输出相应注释；在 IDE 中，工程调用方法时，不进入方法即可悬浮提示方法、参数、返回值的意义，提高阅读效率。

2.【强制】所有的抽象方法（包括接口中的方法）必须要用 Javadoc 注释、除了返回值、参数、异常说明外，还必须指出该方法做什么事情，实现什么功能。

说明：对子类的实现要求，或者调用注意事项，请一并说明。

3. 【强制】所有的类都必须添加创建者和创建日期。

说明：在设置模板时，注意 IDEA 的@author 为`${USER}`，而 eclipse 的@author 为`${user}`，大小写有 区别，而日期的设置统一为 yyyy/MM/dd 的格式。

正例：

/\*\*

\* @author yuhaoyang

\* @date 2021/10/29

\*/

4. 【强制】方法内部单行注释，在被注释语句上方另起一行，使用//注释。方法内部多行注释使用/\* \*/注释，注意与代码对齐。

5. 【强制】所有的枚举类型字段必须要有注释，说明每个数据项的用途。

6. 【推荐】与其“半吊子”英文来注释，不如用中文注释把问题说清楚。专有名词与关键字保持 英文原文即可。

反例：“TCP 连接超时”解释成“传输控制协议连接超时”，理解反而费脑筋。

7. 【推荐】代码修改的同时，注释也要进行相应的修改，尤其是参数、返回值、异常、核心逻辑 等的修改。

说明：代码与注释更新不同步，就像路网与导航软件更新不同步一样，如果导航软件严重滞后，就失去了导航的意义。