一、编程规约

(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 / $Object / name\_ / name$ / Object$

2. 【强制】代码中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，即使纯拼音命名方式 也要避免采用。 正例：alibaba / taobao / youku / hangzhou 等国际通用的名称，可视同英文。 反例：DaZhePromotion [打折] / getPingfenByName() [评分] / int 某变量 = 3

3. 【强制】类名使用 UpperCamelCase 风格，必须遵从驼峰形式，但以下情形例外：DO / BO / DTO / VO / AO 正例：MarcoPolo / UserDO / XmlService / TcpUdpDeal / TaPromotion 反例：macroPolo / UserDo / 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，必须遵从 驼峰形式。 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 反例：MAX\_COUNT

6. 【强制】抽象类命名使用 Abstract 或 Base 开头；异常类命名使用 Exception 结尾；测试类 命名以它要测试的类的名称开始，以 Test 结尾。

7. 【强制】中括号是数组类型的一部分，数组定义如下：String[] args; 反例：使用 String args[]的方式来定义。

8. 【强制】POJO 类中布尔类型的变量，都不要加 is，否则部分框架解析会引起序列化错误。 反例：定义为基本数据类型 Boolean isDeleted；的属性，它的方法也是 isDeleted()，RPC框架在反向解析的时候，“以为”对应的属性名称是 deleted，导致属性获取不到，进而抛出异 常。

9. 【强制】包名统一使用小写，点分隔符之间有且仅有一个自然语义的英语单词。包名统一使用 单数形式，但是类名如果有复数含义，类名可以使用复数形式。 正例： 应用工具类包名为 com.alibaba.open.util、类名为 MessageUtils（此规则参考 spring 的框架结构）

(二)常量定义

1. 【强制】不允许任何魔法值（即未经定义的常量）直接出现在代码中。 反例：String key = "Id#taobao\_" + tradeId; cache.put(key, value);

2. 【强制】long 或者 Long 初始赋值时，使用大写的 L，不能是小写的 l，小写容易跟数字 1 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】不要使用一个常量类维护所有常量，按常量功能进行归类，分开维护。 说明：大而全的常量类，非得使用查找功能才能定位到修改的常量，不利于理解和维护。 正例：缓存相关常量放在类 CacheConsts 下；系统配置相关常量放在类 ConfigConsts 下。

4. 【推荐】常量的复用层次有五层：跨应用共享常量、应用内共享常量、子工程内共享常量、包 内共享常量、类内共享常量。

1） 跨应用共享常量：放置在二方库中，通常是 client.jar 中的 constant 目录下。

2） 应用内共享常量：放置在一方库中，通常是 modules 中的 constant 目录下。 反例：易懂变量也要统一定义成应用内共享常量，两位攻城师在两个类中分别定义了表示 “是”的变量：类 A 中：public static final String YES = "yes"; 类 B 中：public static final String YES = "y"; A.YES.equals(B.YES)，预期是 true，但实际返回为 false，导致线上问题。

3） 子工程内部共享常量：即在当前子工程的 constant 目录下。

4） 包内共享常量：即在当前包下单独的 constant 目录下。 5） 类内共享常量：直接在类内部 private static final 定义

(三)代码格式

1. 【强制】大括号的使用约定。如果是大括号内为空，则简洁地写成{}即可，不需要换行；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】 左小括号和字符之间不出现空格；同样，右小括号和字符之间也不出现空格。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：运算符包括赋值运算符=、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：// 注释内容，注意在//和注释内容之间有一个空格。

7. 【强制】单行字符数限制不超过 120 个，超出需要换行，换行时遵循如下原则： 1） 第二行相对第一行缩进 4 个空格，从第三行开始，不再继续缩进，参考示例。 2） 运算符与下文一起换行。 3） 方法调用的点符号与下文一起换行。 4） 方法调用时，多个参数，需要换行时，在逗号后进行。 5） 在括号前不要换行，见反例。 正例： StringBuffer sb = new StringBuffer(); // 超过 120 个字符的情况下，换行缩进 4 个空格，点号和方法名称一起换行 sb.append("zi").append("xin")... .append("huang")... .append("huang")... .append("huang"); 反例： StringBuffer sb = new StringBuffer(); // 超过 120 个字符的情况下，不要在括号前换行 sb.append("zi").append("xin")...append ("huang"); // 参数很多的方法调用可能超过 120 个字符，不要在逗号前换行 method(args1, args2, args3, ... , argsX);