# 阿里Java开发规范

1. **命名规范**
2. 所有编程相关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。
3. 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，即使纯拼音命名方式也要避免采用。
4. 类名使用 UpperCamelCase风格，必须遵从驼峰形式，但以下情形例外：（领域模型的相关命名）DO / DTO / VO / DAO等。
5. 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。
6. 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。
7. 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。
8. 中括号是数组类型的一部分，数组定义如下：String[] args; 反例：请勿使用 String args[]的方式来定义。
9. POJO类中的任何布尔类型的变量，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 isSuccess；的属性，它的方法也是 isSuccess()，RPC框架在反向解析的时候，“以为”对应的属性名称是 success，导致属性获取不到，进而抛出异常。
10. 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）
11. 杜绝完全不规范的缩写， 避免望文不知义。反例： AbstractClass“ 缩写” 命名成AbsClass；condition“ 缩写” 命名成condi，此类随意缩写严重降低了代码的可阅读性。
12. 【推荐】如果使用到了设计模式，建议在类名中体现出具体模式。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思想。正例： public class OrderFactory;public class LoginProxy;public class ResourceObserver;
13. 【推荐】接口类中的方法和属性不要加任何修饰符号（public也不要加），保持代码的简洁性，并加上有效的 Javadoc注释。尽量不要在接口里定义变量，如果一定要定义变量，肯定是与接口方法相关，并且是整个应用的基础常量。正例： 接口方法签名： void f();接口基础常量表示：String COMPANY= "alibaba";反例： 接口方法定义： public abstractvoid f();说明： JDK8中接口允许有默认实现，那么这个default方法，是对所有实现类都有价值的默认实现。
14. 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，取对应的形容词做接口名（ 通常是–able的形式）。正例： AbstractTranslator实现 Translatable。
15. 【参考】枚举类名建议带上Enum 后缀，枚举成员名称需要全大写，单词间用下划线隔开。说明： 枚举其实就是特殊的常量类，且构造方法被默认强制是私有。正例： 枚举名字： DealStatusEnum，成员名称：SUCCESS/ UNKOWN\_REASON。
16. 【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。

​ 3）获取统计值的方法用count做前缀。

​ 4）插入的方法用save（推荐）或 insert 做前缀。

​ 5）删除的方法用remove（推荐）或 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。

1. **常量定义**
2. 【强制】不允许出现任何魔鬼值（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);
3. 【强制】long 或者 Long初始赋值时，必须使用大写的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?
4. 推荐】不要使用一个常量类维护所有常量，应该按常量功能进行归类，分开维护。如：缓存相关的常量放在类： CacheConsts下；系统配置相关的常量放在类：ConfigConsts下。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。
5. 【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：放置在一方库的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，导致产生线上问题。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
6. 【推荐】如果变量值仅在一个范围内变化用Enum 类。如果还带有名称之外的延伸属性，必须使用 Enum类，下面正例中的数字就是延伸信息，表示星期几。正例： public Enum{MONDAY(1),TUESDAY(2),WEDNESDAY(3),THURSDAY(4),FRIDAY(5),SATURDAY(6),SUNDAY(7);}
7. **代码格式规范**
8. 【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；表示终止右大括号后必须换行。
9. 【强制】 左括号和后一个字符之间不出现空格；同样，右括号和前一个字符之间也不出现空格。详见第 5 条下方正例提示。
10. 【强制】if/for/while/switch/do等保留字与左右括号之间都必须加空格。
11. 【强制】任何运算符左右必须加一个空格。说明： 运算符包括赋值运算符=、逻辑运算符&&、加减乘除符号、三目运行符等。
12. 【强制】 缩进采用4 个空格，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}}

1. 【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）在多个参数超长，逗号后进行换行。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，并且方法前的点符号一起换行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);
2. 【强制】方法参数在定义和传入时，多个参数逗号后边必须加空格。正例： 下例中实参的"a",后边必须要有一个空格。method("a", "b", "c");
3. 【强制】IDE 的 text file encoding设置为 UTF-8; IDE中文件的换行符使用Unix 格式，不要使用 windows格式。
4. 【推荐】没有必要增加若干空格来使某一行的字符与上一行的相应字符对齐。正例：int a = 3;long b = 4L;float c = 5F;StringBuffer sb = new StringBuffer();说明： 增加 sb这个变量，如果需要对齐，则给a、b、c 都要增加几个空格，在变量比较多的情况下，是一种累赘的事情。
5. 【推荐】方法体内的执行语句组、变量的定义语句组、不同的业务逻辑之间或者不同的语义之间插入一个空行。相同业务逻辑和语义之间不需要插入空行。说明： 没有必要插入多行空格进行隔开。
6. **OOP规约**
7. 【强制】避免通过一个类的对象引用访问此类的静态变量或静态方法，无谓增加编译器解析成本，直接用类名来访问即可。
8. 【强制】所有的覆写方法，必须加@Override注解。反例： getObject()与get0bject()的问题。一个是字母的O，一个是数字的0，加@Override可以准确判断是否覆盖成功。另外，如果在抽象类中对方法签名进行修改，其实现类会马上编译报错。
9. 【强制】相同参数类型，相同业务含义，才可以使用Java 的可变参数，避免使用 Object。说明： 可变参数必须放置在参数列表的最后。（ 提倡同学们尽量不用可变参数编程）正例： public User getUsers(String type, Integer... ids)
10. 【强制】对外暴露的接口签名，原则上不允许修改方法签名，避免对接口调用方产生影响。接口过时必须加@Deprecated注解，并清晰地说明采用的新接口或者新服务是什么。
11. 【强制】不能使用过时的类或方法。说明： java.net.URLDecoder中的方法 decode(String encodeStr)这个方法已经过时，应该使用双参数 decode(String source, String encode)。接口提供方既然明确是过时接口，那么有义务同时提供新的接口；作为调用方来说，有义务去考证过时方法的新实现是什么。
12. 【强制】Object的 equals 方法容易抛空指针异常，应使用常量或确定有值的对象来调用equals。正例： "test".equals(object);反例： object.equals("test");说明： 推荐使用 java.util.Objects#equals（ JDK7引入的工具类）
13. 【强制】所有的相同类型的包装类对象之间值的比较，全部使用equals方法比较。说明： 对于 Integer var=?在-128至 127 之间的赋值，Integer对象是在IntegerCache.cache产生，会复用已有对象，这个区间内的Integer值可以直接使用==进行判断，但是这个区间之外的所有数据，都会在堆上产生，并不会复用已有对象，这是一个大坑，推荐使用 equals方法进行判断。
14. 【强制】关于基本数据类型与包装数据类型的使用标准如下：1）所有的 POJO 类属性必须使用包装数据类型。2）RPC 方法的返回值和参数必须使用包装数据类型。3）所有的局部变量【 推荐】 使用基本数据类型。说明： POJO 类属性没有初值是提醒使用者在需要使用时，必须自己显式地进行赋值，任何NPE 问题，或者入库检查，都由使用者来保证。正例： 数据库的查询结果可能是 null，因为自动拆箱，用基本数据类型接收有NPE 风险。反例： 比如显示成交总额涨跌情况，即正负x%，x 为基本数据类型，调用的 RPC 服务，调用不成功时，返回的是默认值，页面显示： 0%，这是不合理的，应该显示成中划线-。所以包装数据类型的 null值，能够表示额外的信息， 如：远程调用失败，异常退出。
15. 【强制】定义DO/DTO/VO等 POJO 类时，不要设定任何属性默认值。反例： POJO 类的 gmtCreate默认值为 new Date();但是这个属性在数据提取时并没有置入具体值，在更新其它字段时又附带更新了此字段，导致创建时间被修改成当前时间。
16. 【强制】序列化类新增属性时，请不要修改serialVersionUID字段，避免反序列失败；如果完全不兼容升级，避免反序列化混乱，那么请修改 serialVersionUID值。说明： 注意 serialVersionUID不一致会抛出序列化运行时异常。
17. 【强制】构造方法里面禁止加入任何业务逻辑，如果有初始化逻辑，请放在init 方法中。
18. 【强制】POJO 类必须写 toString 方法。使用 IDE 的中工具：source>generate toString时，如果继承了另一个POJO 类，注意在前面加一下 super.toString。说明： 在方法执行抛出异常时，可以直接调用POJO 的 toString()方法打印其属性值，便于排查问题。
19. 【推荐】使用索引访问用String的 split 方法得到的数组时，需做最后一个分隔符后有无内容的检查，否则会有抛 IndexOutOfBoundsException的风险。说明：String str = "a,b,c,,";String[] ary = str.split(",");//预期大于3，结果是3System.out.println(ary.length);
20. 【推荐】当一个类有多个构造方法，或者多个同名方法，这些方法应该按顺序放置在一起，便于阅读。
21. 【推荐】 类内方法定义顺序依次是：公有方法或保护方法> 私有方法> getter/setter方法。说明： 公有方法是类的调用者和维护者最关心的方法，首屏展示最好；保护方法虽然只是子类关心，也可能是“模板设计模式”下的核心方法；而私有方法外部一般不需要特别关心，是一个黑盒实现；因为方法信息价值较低，所有Service和 DAO 的getter/setter方法放在类体最后。
22. 【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，尽量不要增加业务逻辑，增加排查问题的难度。反例：public Integer getData(){if(true) {return data + 100;} else {return data - 100;}}
23. 【推荐】循环体内，字符串的联接方式，使用StringBuilder的 append 方法进行扩展。反例：String str = "start";for(int i=0; i<100; i++){str = str + "hello";}说明： 反编译出的字节码文件显示每次循环都会new 出一个 StringBuilder对象，然后进行append操作，最后通过toString方法返回 String 对象，造成内存资源浪费。
24. 【推荐】final可提高程序响应效率，声明成final的情况：1）不需要重新赋值的变量，包括类属性、局部变量。2）对象参数前加final，表示不允许修改引用的指向。3）类方法确定不允许被重写。
25. 【推荐】慎用Object的 clone 方法来拷贝对象。说明： 对象的 clone方法默认是浅拷贝，若想实现深拷贝需要重写clone方法实现属性对象的拷贝。
26. 【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，必须考虑是否为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 方法，或者一个 public的成员变量，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，如果无限制的到处跑，那么你会担心的。​
27. **集合处理规范**
28. 【强制】 关于hashCode和 equals 的处理，遵循如下规则：1） 只要重写equals，就必须重写hashCode。2） 因为Set 存储的是不重复的对象，依据hashCode和 equals 进行判断，所以 Set 存储的对象必须重写这两个方法。3） 如果自定义对象做为Map 的键，那么必须重写 hashCode 和 equals。正例： String 重写了 hashCode和 equals 方法，所以我们可以非常愉快地使用String对象作为 key来使用。
29. 【强制】ArrayList的subList结果不可强转成ArrayList，否则会抛出ClassCastException异常： java.util.RandomAccessSubList cannot be cast to java.util.ArrayList ;说明： subList 返回的是 ArrayList 的内部类 SubList，并不是ArrayList，而是ArrayList的一个视图，对于SubList子列表的所有操作最终会反映到原列表上。
30. 【强制】 在subList场景中， 高度注意对原集合元素个数的修改，会导致子列表的遍历、增加、删除均产生 ConcurrentModificationException异常。
31. 【强制】使用集合转数组的方法，必须使用集合的toArray(T[] array)，传入的是类型完全一样的数组，大小就是 list.size()。反例： 直接使用 toArray 无参方法存在问题，此方法返回值只能是Object[]类，若强转其它类型数组将出现 ClassCastException错误。正例：List list = new ArrayList(2);list.add("guan");list.add("bao");String[] array = new String[list.size()];array = list.toArray(array);说明： 使用 toArray带参方法，入参分配的数组空间不够大时，toArray方法内部将重新分配内存空间，并返回新数组地址；如果数组元素大于实际所需，下标为[ list.size() ]的数组元素将被置为 null，其它数组元素保持原值，因此最好将方法入参数组大小定义与集合元素个数一致。
32. 【强制】使用工具类Arrays.asList()把数组转换成集合时，不能使用其修改集合相关的方法，它的 add/remove/clear方法会抛出UnsupportedOperationException异常。说明： asList 的返回对象是一个 Arrays 内部类，并没有实现集合的修改方法。Arrays.asList体现的是适配器模式，只是转换接口，后台的数据仍是数组。String[] str = new String[] { "a", "b" };List list = Arrays.asList(str);第一种情况：list.add("c");运行时异常。第二种情况： str[0]= "gujin";那么 list.get(0)也会随之修改。