(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】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】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

4. 【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

5. 【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。

6. 【强制】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。

7. 【推荐】为了达到代码自解释的目标，任何自定义编程元素在命名时，使用尽量完整的单词组合来表达其意。

   正例：在 JDK 中，表达原子更新的类名为：AtomicReferenceFieldUpdater。

   反例：变量 int a 的随意命名方式。

8. 【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;

9. 【参考】各层命名规约：

10. 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，复数形式结尾如：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) 代码及注释格式

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】在 if/else/for/while/do 语句中必须使用大括号。即使只有一行代码，避免采用单行的编码方式：if (condition) statements;

3. 【推荐】 类内方法定义的顺序依次是：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；因为承载的信息价值较低，所有 Service 和 DAO 的 getter/setter 方法放在类体 最后。

4. 【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。

5. 【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。

6. 【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。

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。

参考：阿里巴巴 Java 开发手册