一.命名规范

1.【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。
反例： `DaZhePromotion [打折] / getPingfenByName() [评分] / int 某变量 = 3`
正例： `alibaba / taobao / youku / hangzhou` 等国际通用的名称，可视同英文。

2. 类名使用 UpperCamelCase 风格（首字母大写），必须遵从驼峰形式。

正例： `MarcoPolo / UserDO / XmlService / TcpUdpDeal / TaPromotion`
反例： `macroPolo / UserDo / XMLService / TCPUDPDeal / TAPromotion`

3. 方法名、参数名、成员变量、局部变量都统一使用 lowerCamelCase 风格（首字母小写），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。

正例： 应用工具类包名为 com.java.open.util、类名为 MessageUtils（此规则参考spring 的框架结构）

6. 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】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;
```

8. 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`。

常量定义

1.