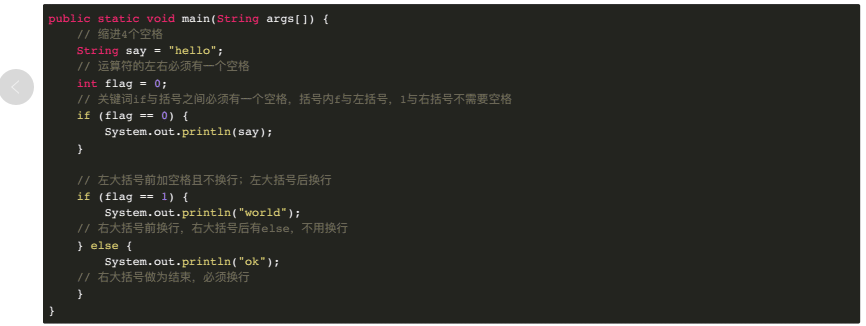
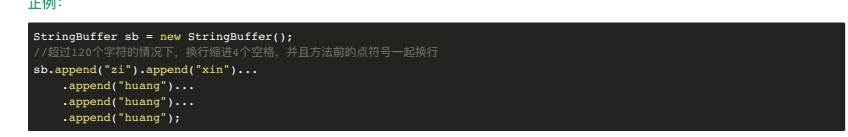
1. 编程规约
   1. 命名规约：
      1. 所有编程相关的命名均不能以下划线或美元符号开始，也不能以此结束
      2. 所有编程相关的命名禁止使用拼音与英文混合的方式，更不允许直接使用中文的方式
      3. 类名、方法名使用大驼峰
      4. 成员变量、局部变量、参数名、使用小驼峰
      5. 常量命名应该全部大写，单词间用下划线隔开，力求语义表达完整清楚，不要嫌名字长。
      6. 抽象类命名使用Abstract或Base开头；异常类命名使用Exception结尾；测试类命名以它要测试的类的名称开始，以Test结尾。
      7. 中括号是数组类型的一部分 应该这样string[] args不能这样string args[]
      8. 。。
      9. 。。
      10. 杜绝完全不规范的缩写
      11. 如果使用到了设计模式，建议在类名中体现出具体模式 OrderFactory
      12. 接口类中的方法和属性不要加任何的修饰符 public也不要加 保持代码的间接性 并且加上有效的注释
      13. 接口和实现类的两套规则：对于service和dao类 基于soa的理念，保罗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 CacheServiceImpl实现CacheService
      14. 枚举类名建议才上Enum后缀，枚举成员名称需要全大写，单词间用下划线隔开。枚举名字：DealStatusEnum成员名称：SUCCESS/UNKOWN\_REASON
      15. 各类命名规约:
          1. 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. 获取统计值的犯法用Count做前缀
             4. 插入的方法用save或insert做前缀
             5. 删除的方法用remove或delete做前缀
             6. 修改的方法用update做前缀
          2. 领域模型命名规约
             1. 数据对象：XXXDO XXX即为数据表明
             2. 数据传输对象：XXXDTO XXX为业务领域相关名称
             3. 展示对象：xxxVO，xxx一般为网页名称
   2. 常量定义
      1. 不允许出现任何魔法值（即未经定义的常量）直接出现在代码中。String key=“ID#taobao\_”+tradeId 提取时比较容易少些下划线 导致故障
      2. long或者Long初始赋值时，必须使用大写的L，不能是小写的l 小写容易和数字1混淆
      3. 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。
      4. 常量的服用层次有五层，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
      5. 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
   3. 格式规约
      1. 如果大括号内为空，则简洁的写成{}，不需要换行
      2. 
      3. 单行字符数限制不超过120个 超出需要换行
         1. 第二行相对第一行缩进4个空格，从第三行开始，不再进行缩进
         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
         6. 
      4. 方法参数在定义和传入时 多个参数逗号后边必须加空格
      5. 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
      6. 方法内的执行语句组、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。
   4. OOP规约
   5. 集合处理
   6. 并发处理
   7. 控制语句
      1. 在一个switch块内，每个case要么通过break/return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，即使它什么代码也没有。
      2. 在if/else/for/while/do语句中必须使用大括号，即使只有一行代码，也要使用大括号
      3. 推荐尽量少用else 直接返回return 或者用状态设计模式（定义好状态类且用switch）
      4. 不要再条件语句中执行复杂的语句，以提高可读性
      5. 循环体内的语句要考量性能，没必要在循环中的语句要移出循环体外，如定义变量 声明 try-catch等
      6. 接口入参保护 一般对参数或者调用次数的限制
      7. 需要进行参数校验的场景：
         1. 调用频次低的方法
         2. 执行时间开销很大的方法
         3. 需要极高稳定性和可用性的方法
         4. 对外提供的开放接口
      8. 方法中不需要参数校验的场景
         1. 极有可能被循环调用的方法，不建议对参数进行校验
         2. 底层的方法调用拼读都比较高，一般不校验，参数错误不太可能到底层才会暴露问题，一般在前几层就处理掉了。
         3. 被声明的private只会被自己代码所调用的方法
   8. 注释规约
      1. 类、类属性、类方法的注释必须使用/\*\*内容\*/格式 不得使用//内容 格式
      2. 所有的抽象方法（包括接口中的方法）必须使用/\*\*内容\*/格式的注释，处了返回值，参数，异常说明外，还必须指出该方法做了什么事情，实现了什么功能
      3. 所有的类都必须添加创建者信息和创建日期
      4. 方法内部使用 //内容 注释格式 多行使用/\*\*/格式 注意与代码对齐
      5. 所有的枚举类型字段都必须要有注释
      6. 与其“半吊子”英文来注释，不如用中文注释把问题说清楚
      7. 代码修改的同时，注释也要进行响应的修改，尤其是参数，返回值，异常，核心逻辑等的修改
      8. 注释掉的代码尽量要配合说明，而不是简单地注释掉
      9. 注释的要求：第一、要能准确地反应设计思想和代码逻辑 第二、能够描述业务含义
      10. 好的命名、代码结构是自解释的，注释力求精简准确、表达到位
      11. 特殊注释标记，请注明标记人与标记时间，注意即使处理这些标记，通过标记扫描，及时清理此类标记。
   9. 其他
      1. 任何数据结构的使用都应该限制大小
2. 异常日志
   1. 异常处理
      1. 异常不要用来作流程控制，条件控制，因为异常的处理效率比条件分支低
      2. 对大段代码进行try-catch 这是不负责任的表现。非稳定代码加try-catch
      3. 捕获异常是为了处理它，不要捕获了什么都不处理。如果不想处理，请把该异常抛给它的调用者。最外层的业务使用者，必须处理异常，将其转化为用户可以理解的内容。
      4. 事务场景中，抛出异常被catch后 如果需要回滚，一定要手动回滚事务
      5. Finally块必须对资源对象、流对象进行关闭，有异常也要做try-catch
      6. 不能在finally块中使用return finally块中的return返回方法结束执行，不会在执行try块中的return语句
      7. 捕获与抛异常，必须是完全匹配，或者捕获异常是抛异常的父类
      8. 方法的返回值可以为null，不强制返回空集合，或者空对象等，必须添加说明什么情况下会返回null值。调用方需要进行null判断
      9. 非空判断是程序员的基本修养
      10. 在代码中使用“抛异常”还是返回“错误码”对于公司外的http/api开放接口必须使用错误码 而应用内部推荐异常抛出
      11. 避免出现重复的代码（随意的赋值粘贴 必然出现代码的重复）
   2. 日志规约
3. MYSQL规约
   1. 建表规约
   2. 索引规约
   3. SQL规约
   4. ORM规约
4. 工程规约
   1. 应用分层
   2. 二房库规约
   3. 中间件规约
   4. 服务器规约
5. 安全规约
6. 命名推荐列表