京东秒送售后系统退款业务重构心得| 京东零售技术团队

一、重构背景

1.1、退款

京东秒送秒送退款有**2套**结构，代码逻辑混乱；

其中秒送、天选部分售后单是和平生pop交互退款，部分是和售后中台交互退款；并且**兼容3套**逻辑；

痛点：代码繁重，缺乏合理性的设计，后续迭代**开发**以及**维护成本高**，同时增加了系统的**风险和不稳定性**

1.2、金额计算

京东秒送两套独立的逻辑结构计算，在此基础上针对退差和非退差又实现了**2套**逻辑；

针对商品件维度、商品行维度、售后单维度计算金额混乱，**缺乏领域边界**和**分层设计**；

痛点：售后单维度、商品行维度、拆分件维度金额**计算混乱**，代码**缺乏层次结构**；代码**易读性**、**维护成本**、后续**扩展性**存在问题

1.3、售后逆向账

售后单详情接口、申诉单详情接口，针对京东秒送秒送做了**两套**逻辑；

其中售后单详情接口针对秒送黑名单、秒送白名单、天选、秒送退差、秒送非退差做了**5套**逻辑处理；

并且这两个接口都是实时从拆分获取金额进行售后逆向拆分计算，可以直接从数据库中进行取值赋值，不需要进行售后单维度的拆分计算；

痛点：代码大量**冗余**、改动**成本高** **、** 增加了系统的**风险**和**不稳定性**

二、重构思路和方案

2.1、重构思路

什么是重构呢？

名词：对软件内部结构的一种调整，目的是在不改变软件观察行为的的前提下提高其可理解性、降低其修改成本；

动词：使用一系列手法，在不改变软件可观察行为的前提下，调整其结构

重构的目的是使系统或代码更容易被理解、修改、迭代

重构秘诀：胆大心细

胆大（意味着有勇气和决心去改变和改进现有的代码。重构可能涉及对复杂的代码结构进行修改，甚至可能需要重写部分代码。胆大的开发者愿意面对这些挑战，相信通过改变可以带来更好的结果）

心细（指的是在进行重构时保持细致入微的思考和行动。这包括仔细分析代码的结构和逻辑，理解代码的功能和依赖关系，以及考虑每个重构步骤可能带来的潜在影响。心细的开发者会在重构过程中小心翼翼地处理每个细节，以确保代码的正确性和可维护性）

1.把握好重构时机：当我发现售后退款、金额计算等业务模块代码存在质量问题、可读性差、可维护性差或存在坏味道时，并且在项目需求排期并不紧张的情况下，是进行重构的好时机；

2.前期梳理很重要，先找到痛点 ；不宜长线作战，不宜和业务并行

3.明确出目标和价值：售后退款、金额计算重构后能提高开发效率、降低维护、开发成本等

4.确定重构的目标：首先要明确需要进行重构的代码块或功能，并明确重构的目标是什么。例如，可能需要提高代码的可读性、可维护性或性能。

5.分析代码坏味道：使用代码静态分析工具或手动检查代码，识别出可能存在的[代码坏味道](http://cxyroad.com/”https://www.qinglite.cn/doc/87036476d565d55f9”)；例如退款业务中存在1000多行的类、600多行的方法，过多的变量参数、诸多重复代码等[代码坏味道](http://cxyroad.com/”https://www.qinglite.cn/doc/87036476d565d55f9”)。

6.选择适当的重构技术：根据售后[代码坏味道](http://cxyroad.com/”https://www.qinglite.cn/doc/87036476d565d55f9”)的种类和重构的目标，选择适当的重构技术。我采用的重构手法是：**小规模重构-->大规模重构-->顶层设计模式；** 采用**先小后大，从大到全**的思路进行重构设计 **。小规模重构**：提取方法、消除超大类或函数方法、提取类、重命名、合并重复代码等方法；**大规模重构：** 采用的是分层、模块化、解耦、抽象可复用性等手法；**设计模式**：退款业务采用策略模式+抽象工厂；金额计算业务采用策略模式+抽象工厂+责任链模式

7.编写测试用例：在进行重构之前，编写适当的测试用例来验证重构后的代码的正确性。测试用例应该覆盖重构的代码块或功能的各种情况。

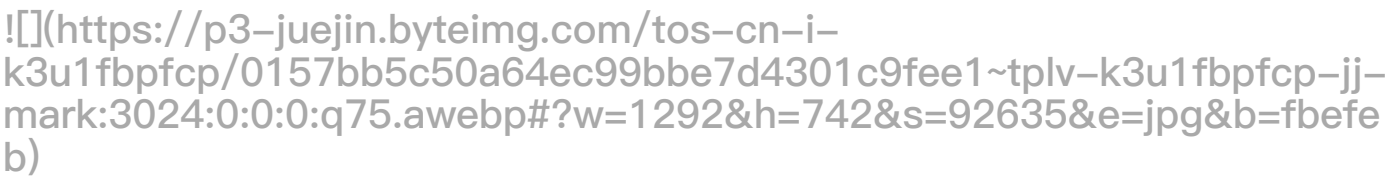
8.执行重构：根据选择的重构技术，逐步修改代码。确保每次修改后的代码仍然通过之前编写的测试用例。

9.运行测试用例：在每次重构之后，运行之前编写的测试用例，确保重构后的代码仍然正确。

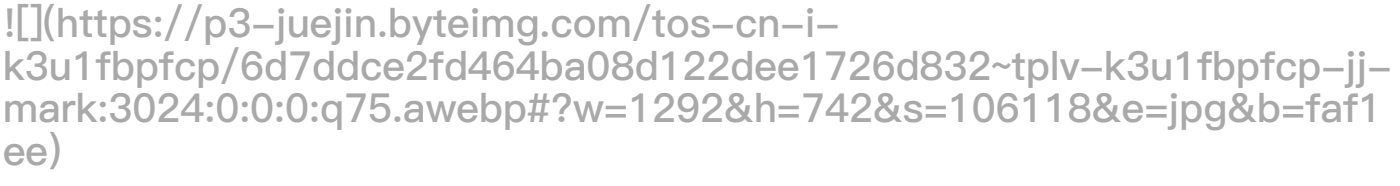
10.重构后的代码评估：评估重构后的代码是否达到了预期的目标，例如是否提高了代码的可读性、可维护性或性能。

2.2、重构方案

2.2.1、重构前系统交互图



2.2.2、重构后系统交互图

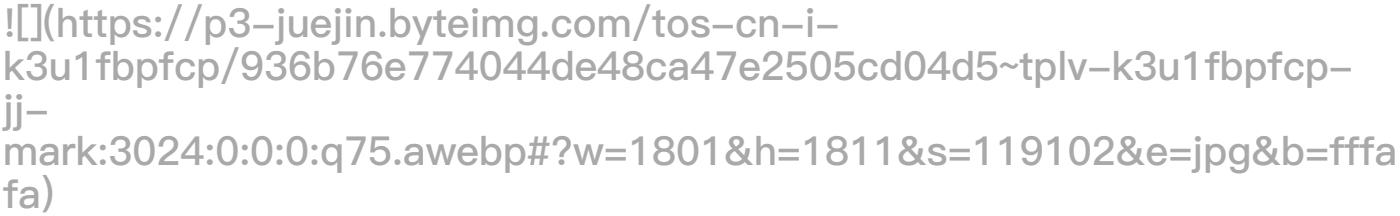


退款业务强耦合到售后系统中，并且业务代码分散到各个业务层，严重缺乏系统的领域边界和分层设计，重构后退款业务逻辑不强依赖售后核心业务逻辑，做到可以独立部署。

2.2.3、重构前金额计算流程图

k3u1fbpfcp/7441d81194244eadbed4b92ceaa25a52~tplv-k3u1fbpfcp-jj-mark:3024:0:0:0:q75.awebp#?w=2781&h=2701&s=260675&e=jpg&b=d5e8d4)

2.2.4、重构后金额计算流程图

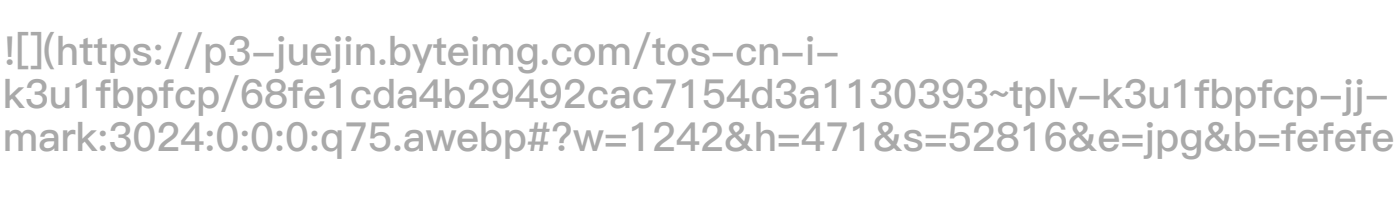


将2套金额计算业务逻辑利用设计模式将其合并为1套金额计算业务逻辑，打造防腐层

2.3、重构设计类图

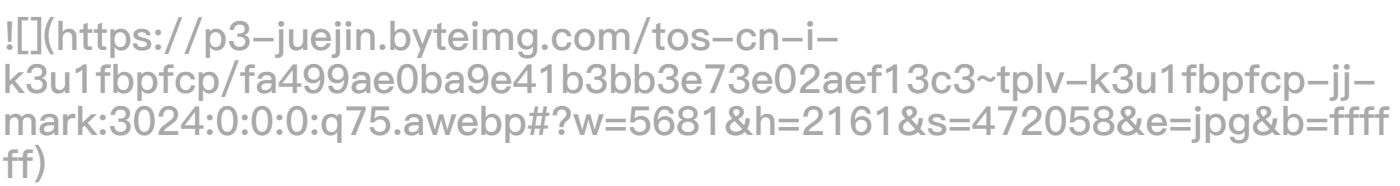
依据上述制定的设计方案流程图，我进行了UML类图的绘制，以下是关于金额计算业务模块的类图

2.3.1、抽象工厂+策略模式类图



)

2.3.2、责任链模式类图



三、系统稳定性保障

3.1、小步重构

将售后重构分成退款、金额计算、逆向账三个步骤，并在每个步骤之后运行测试用例。这样可以及时发现并修复引入的错误，避免错误在整个系统中蔓延

3.2、逐步验证

在每个重构步骤之后，进行系统的逐步验证。分批次进行上线灰度，灰度配置绝对隔离，不能复用。确保系统的各个部分在重构过程中都能正常运行，并与其他部分协调良好。

3.3、监控和性能测试

在重构完成后，进行系统的监控和性能测试，确保重构没有引入性能问题或影响系统的稳定性。如果发现问题，及时进行修复和优化。

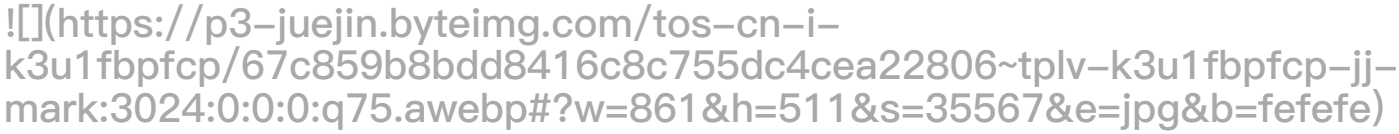
3.4、团队代码审查和测试

在进行重构时，与团队成员进行合作，并进行代码审查。多个人的视角和经验可以帮助发现潜在的问题，并提供改进的建议；针对重构代码进行深度解刨，能更有效地保障重构的安全性。

重构业务及时通知测试人员，使测试人员能够评估到测试点，更加完善测试用例

3.5、灰度步骤

3.5.1、bcp持续比对校验



3.5.2、按照商家灰度

依据售后单量 小->中->大 的顺序逐步进行灰度切量，观察其退款、金额计算等售后单数据是否异常

四、重构成果

1.降低开发、维护成本

2.提升代码质量、系统稳定性

3.系统扩展性和灵活性的加强；

4.系统应用、业务边界定位更加清晰

5.统一和规范售后核心业务脉络，降低业务学习成本，提升开发效率

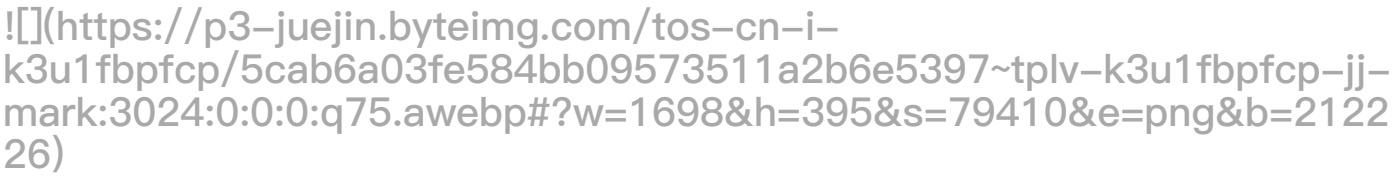
6.提升自己的技术能力、代码质量意识、问题解决能力、团队合作和沟通能力；经典著作《重构》这本书中有这么一段话：

一开始，我所做的重构都停留在细枝末节上。随着代码趋向简洁，我发现自己可以看到一些设计层面的东西了，这些是我以前理解不到的，如果没有重构，我达不到这种高度

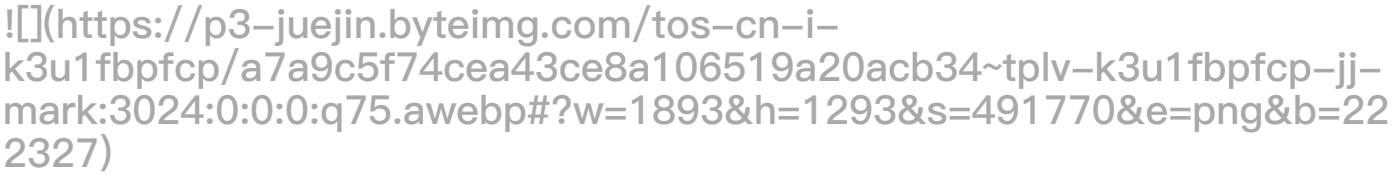
五、code show
==========

5.1、重构前金额计算

京东秒送售后单金额计算service方法

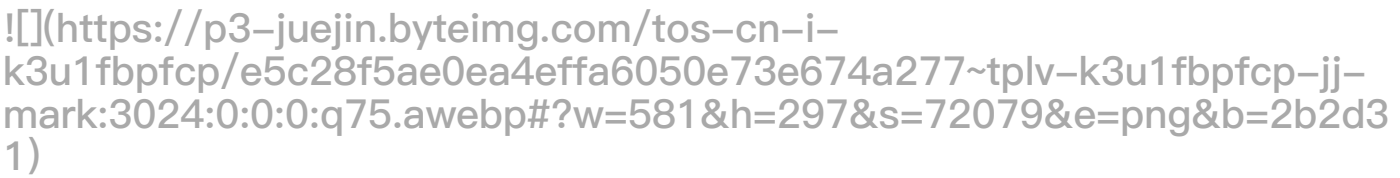


一个大的金额计算class类就有1000多行代码，每个方法中都有几百行代码，以下是京东秒送售后单金额计算部分代码

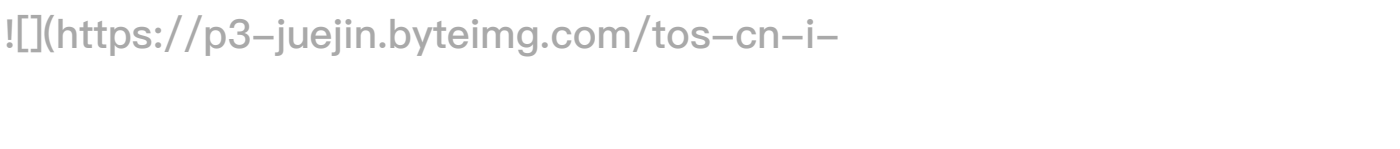


5.2、重构后金额计算

京东秒送售后单金额计算用同一个接口才承接业务实现，并且使用策略+抽象工厂模式实现秒送、天选业务的金额计算



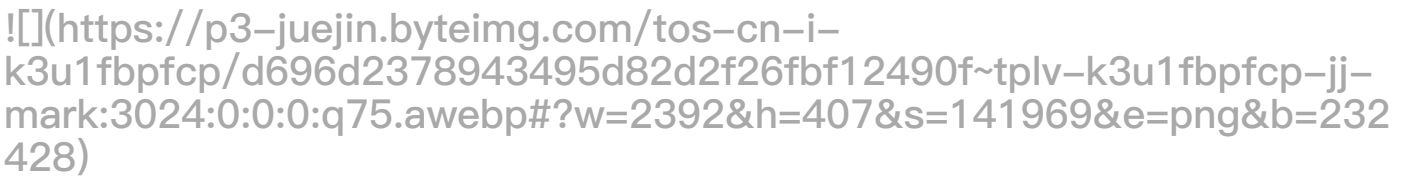
策略模式获取金额拆分结果集



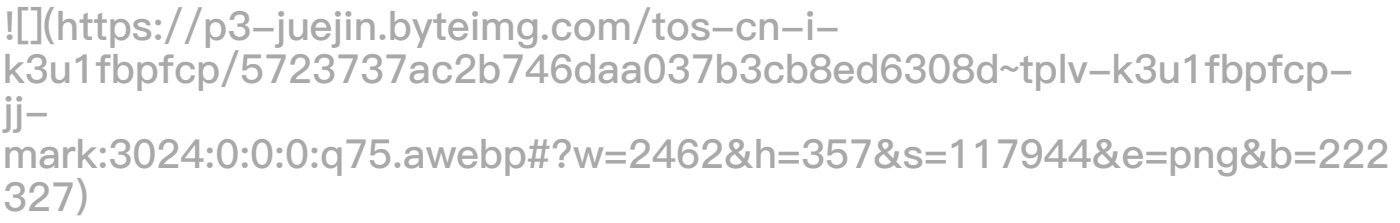
k3u1fbpfcp/37b4bdc28af34f3184f47a93f7efaac4~tplv-k3u1fbpfcp-jj-mark:3024:0:0:0:q75.awebp#?w=1886&h=1057&s=269977&e=png&b=212226)

金额计算核心方法只有4步骤

其中金额计算的核心则采用的是责任链业务进行计算



在件维度、sku维度针对不同的业务又采用了责任链模式进行金额计算



六、参考文献

代码的坏味道： [www.qinglite.cn/doc/8703647…](http://cxyroad.com/”https://www.qinglite.cn/doc/87036476d565d55f9”)

《重构改善既有代码的设计》：[美]MartinFowler

《敏捷软件开发》：[美]RobertC.Martin

作者：京东零售 高凯

来源：京东云开发者社区

原文链接: https://juejin.cn/post/7369248993106444340