程序设计中级实践大作业

22231 学期

一、 任务描述

1. 项目需求变更（新增需求）

在已经完成的 “饿了么” 项目中，增加两个新功能模块，即虚拟钱包模块和积分系统模块。

2. 开发团队（分组）

原则上延续 “软件工程综合实践” 课程中的分组安排，继续由原团队成员完成此次需求变更。个别团队需要调整人员的向老师报备。

3. 项目基础

基于 “饿了么” 项目三的前端（VUE）和 “饿了么” 项目四的后端（Spring Boot）继续开发。

4. 项目目标

新增的两个功能模块与原功能模块融合为一个新的系统。

二、 虚拟钱包模块的需求和设计

1. 功能需求

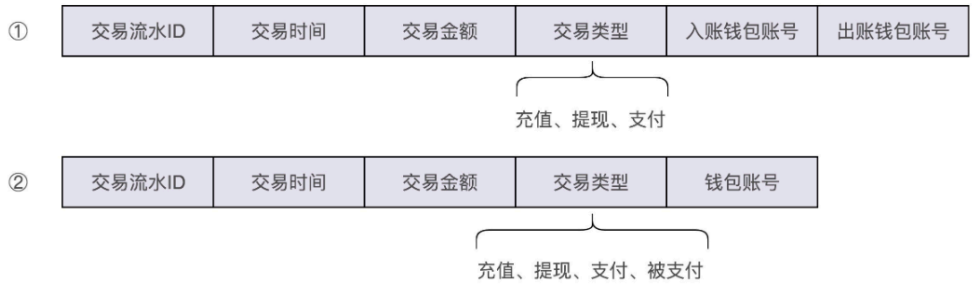
钱包支持充值、提现、支付、查询余额、查询交易流水这五个核心的功能，其他比如冻结、透支、转赠等不常用的功能，暂不考虑。

可以把整个钱包系统的业务划分为两部分，其中一部分单纯跟应用内的虚拟钱包账户打交道（支付、查询余额、查询交易流水），另一部分单纯跟银行账户打交道（充值、提现）。基于此进行业务划分，给系统解耦。

银行系统现实中可能是微信钱包、支付宝或者是网银系统等，在本次实践项目中不必连接真实的银行系统，只需要在虚拟钱包模块中提供接口（HTTP 接口）即可。测试时可以在系统外手工调用该接口完成充值和提现。

2. 设计要求

交易流水如何记录？以下两种方案，经分析，方案 1 不会造成交易记录不完整一致，建议使用方案 1 。



传统的后端项目分为 Repository 层、Service 层、Controller 层。其中，Repository 层负责数据访问，Service 层负责业务逻辑，Controller 层负责暴露接口。

本次实践项目中，要求使用**充血模型**实现 Service 层。应包括 VirtualWalletService 类、VirtualWallet 类和 VirtualWalletTransaction 类，分别代表服务类、钱包类和交易流水类。

贫血模型与充血模型对比说明

- 贫血模型：只包含数据，不包含业务逻辑的类，就叫作贫血模型。

- 充血模型：数据和对应的业务逻辑被封装到同一个类中。

基于充血模型的开发模式实现的代码，也是按照 MVC 三层架构分层的。Controller 层还是负责暴露接口，Repository 层还是负责数据存取，Service 层负责核心业务逻辑。它跟基于贫血模型的传统开发模式的区别主要在 Service 层。

在基于贫血模型的传统开发模式中，Service 层包含 Service 类和 BO 类两部分，BO 是贫血模型，只包含数据，不包含具体的业务逻辑。业务逻辑集中在 Service 类中。

在基于充血模型的开发模式中，Service 层包含 Service 类和 Domain 类两部分。Domain 就相当于贫血模型中的 BO。不过，Domain 与 BO 的区别在于它是基于充血模型开发的，既包含数据，也包含业务逻辑。在这种开发模式下，我们把虚拟钱包 VirtualWallet 类设计成一个充血的 Domain 领域模型，并且将原来在 Service 类中的部分业务逻辑移动到 VirtualWallet 类中，让 Service 类的实现依赖 VirtualWallet 类。

三、 积分系统模块的需求和设计

1. 功能需求

为提高用户粘性，电商平台一般都会有积分系统，请为“饿了么”项目增加一个积分系统，主要考虑的需求包括：

- 积分获取规则

积分获得的途径可能有很多，如下订单后可以根据订单金额获得相应的积分，这里需要制定订单金额与积分的对换规则，如订单 1 元对应增加积分 1 分，不足 1 元的部分不增加积分之类的，这个规则应该是可以调整的。另外，积分获得规则可能与营销相关，如用户新加入、节假日、用户生日等特殊时间可能会有临时调整，比如在特殊时间期间积分双倍等。所以，需要有专门的功能来对积分获取规则进行调整。

获得的积分可能有有效期，所以积分记录并不是简单的在原积分上进行累加，而是应该单独记录一条，包括积分和失效日期。积分的有效期计算规则，同样应该在积分系统中进行设置和调整。

- 积分消费规则

积分消费的方法，包括单独使用积分兑换礼品，或者在下单时抵扣部分金额等，如 100 积分可以抵扣 1 元等。积分与抵扣金额的比例关系应该也是可以调整的。另外，要能够自动选择临近失效的积分消费。

- 积分及其明细查询

应能够区分显示积分的获取流水记录和积分的使用流水记录。

2. 功能模块（纵向切分）

电商平台中，可能会包括以下几个与积分相关的模块：

- 营销系统，负责各种促销活动。
- 业务系统，如订单系统、评论系统等，可以有机会获得积分。
- 换购商城、支付系统等，可以消费积分。

模块之间功能的划分方法主要有三种情况：

第一种划分方式是：积分赚取渠道及兑换规则、消费渠道及兑换规则的管理和维护（增删改查），不划分到积分系统中，而是放到更上层的营销系统中。这样积分系统就会变得非常简单，只需要负责增加积分、减少积分、查询积分、查询积分明细等这几个工作。

比如，用户通过下订单赚取积分。订单系统通过异步发送消息或者同步调用接口的方式，告知营销系统订单交易成功。营销系统根据拿到的订单信息，查询订单对应的积分兑换规则（兑换比例、有效期等），计算得到订单可兑换的积分数量，然后调用积分系统的接口给用户增加积分。

第二种划分方式是：积分赚取渠道及兑换规则、消费渠道及兑换规则的管理和维护，分散在各个相关业务系统中，比如订单系统、评论系统、签到系统、换购商城、优惠券系统等。

还是刚刚那个下订单赚取积分的例子，在这种情况下，用户下订单成功之后，订单系统根据商品对应的积分兑换比例，计算所能兑换的积分数量，然后直接调用积分系统给用户增加积分。

第三种划分方式是：所有的功能都划分到积分系统中，包括积分赚取渠道及兑换规则、消费渠道及兑换规则的管理和维护。还是同样的例子，用户下订单成功之后，订单系统直接告知积分系统订单交易成功，积分系统根据订单信息查询积分兑换规则，给用户增加积分。

本次实验中模块划分时，请使用**第三种划分方式**完成。

3. 模块与模块之间的交互关系

常见的系统之间的交互方式有两种：一种是同步接口调用，另一种是利用消息中间件异步调用。第一种方式简单直接，第二种方式的解耦效果更好。上下层系统之间的调用倾向于通过同步接口，同层之间的调用倾向于异步消息调用。比如，营销系统和积分系统是上下层关系，它们之间就比较推荐使用同步接口调用。

本次实验，请使用同步接口调用方法实现即可。

4. 代码分层（横向切分）

大部分业务系统的开发都可以分为三层：Contoller 层、Service 层、Repository 层。其中 Controller 用来与前端用户界面交互，Service 层用来处理业务逻辑，Repository 层用来与数据库交互。这样做的好处是：

1. 分层能起到代码复用的作用
2. 分层能起到隔离变化的作用
3. 分层能起到隔离关注点的作用
4. 分层能提高代码的可测试性
5. 分层能应对系统的复杂性

5. 数据对象 BO、VO、Entity

Controller、Service、Repository 三层，每层都会定义相应的数据对象，它们分别是 VO（View Object）、BO（Business Object）、Entity，例如 UserVo、UserBo、UserEntity。VO、BO、Entity 三个类虽然代码重复，但功能语义不重复。可以使用继承或组合，解决代码重复问题。

四、 设计要求

在完成程序设计中级实践大作业时，重点要放在设计环节中如何体现面向对象设计原则，常见的设计原则包含但不限于下列原则：

1) 高内聚、松耦合

2) 面向接口而非实现编程
3) 多用组合加委托少用继承
4) SRP 单一职责原则
5) OCP 开闭原则
6) LSP 里氏替换原则
7) ISP 接口隔离原则
8) DIP 依赖倒置原则
9) KISS 原则
10) DRY 原则
11) LoD 迪米特法则，又称 LKP 最少知识原则
12) RoT 原则

五、 需完成的材料

1. 在“饿了么”项目三前端和项目四后端基础上，增加虚拟钱包模块和积分系统模块，形成改进版完整的项目代码。
2. 改进版项目的相关文档。以下内容合并在一份文档中提交。其中应包括：
   (一) 项目需求/设计/测试/部署文档，即《软件需求规格说明书》《软件设计文档》《软件测试文档》《软件部署文档》等。其中：
      a) 《软件需求规格说明书》参考《UML 大战需求分析》一书附录 1 格式完成。
      b) 《软件设计文档》需要至少包括：软件架构与分层设计说明、接口说明（包括前端与后端之间的大接口，也包括前端内部和后端内部各分层之间的小接口）、业务对象设计说明和数据库设计及其它要说明的设计内容。
      c) 《软件部署文档》可以参考“饿了么”项目中的部署手册。建议考虑从开发环境切换到生产环境下进行部署。
   (二) 项目中体现了哪些面向对象设计原则以及是如何实现的。
   (三) 小组成员及分工任务描述。
   (四) 项目开发过程描述，包含组会、协作、Git 记录等。
   (五) 项目特色之处。
   (六) 项目进展过程中的问题点/解决对策/结果反思等。