

面向对象分析与设计 A 卷答案

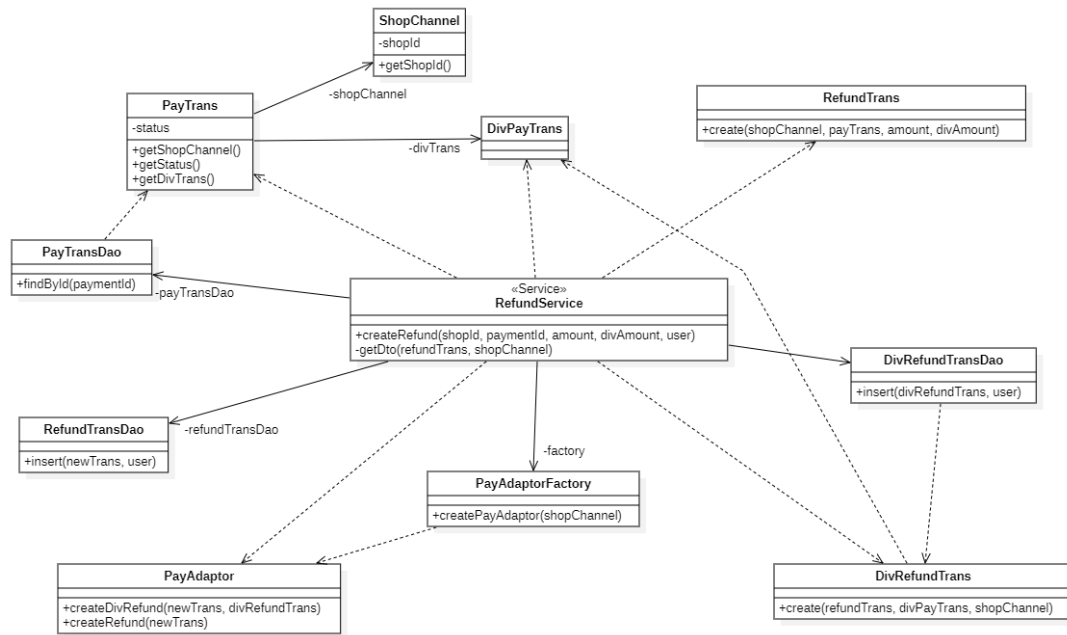
一、 简答题（每题 10 分，共 30 分）

- 1、 结合课程设计中的实际运用，简述层次体系结构中 Dao 层和 Service 层的职责是什么。
 - a. Dao 层和 Service 层是层级体系结构中相连的两层，Service 层代码可以调用 Dao 层代码，但 Dao 层不能调用 Service 层。（2 分）
 - b. 在课程设计中，Service 层的职责是实现系统的业务（4 分）
 - c. Dao 层负责数据存取（3 分），实现系统的对象模型（1 分）
- 2、 简述面向功能设计（结构化编程）、面向对象设计和函数式编程的各自特点是什么。
 - a. 面向功能的设计是以功能出发，将功能分解成为若干小功能，将小功能分解为子功能，直至功能能用函数实现。数据作为函数的参数传入，函数被视作数据的加工和处理，结果作为返回值输出。（3 分）。
 - b. 面向对象的设计是以数据为核心，建立基于需求的领域模型，进而产生对象模型。程序的功能按照 GRASP 原则分配给对象，从而形成对象的方法，每个对象所分配的职责都是基于该对象的属性所能完成的职责，对象方法中所需的数据大多可通过对象属性获得。（4 分）
 - c. 函数式编程把程序视作表达式和转换。与面向功能的设计不同，函数式编程并不将功能进行分解，而是将功能转换为多个函数的连环处理。函数作为程序的基本组成部分，完成计算和转换。每一个函数的输出是下一个函数输入。函数和其所使用的变量形成一个独立的上下文环境。（3 分）
- 3、 简述领域模型和设计类图的相同点和区别。
 - a. 领域模型是从需求中提取概念和概念之间的关系，它采用 UML 类图描述，用类描述概念，类的属性描述概念的属性，未使用类的方法。用类的关联关系描述概念之间的关系，概念之间的关系是双向的。（5 分）
 - b. 设计类图描述的是代码的静态结构类和接口的关系，它同样采用 UML 类图描述，用类描述代码中的类或者接口，用关联关系描述类的特殊属性，用继承关系描述类的继承关系，用依赖关系描述类之间代码的依赖。

二、 画图题（30 分）

图 1 是支付模块中建立退款的交易的相关代码，请根据代码画出相应的设计类图和时序图，要求除以下内容外，图 1 中的其余内容均需要画出。

- 1、 类图中可以省略属性的类型
- 2、 类图中可以省略方法的参数类型及返回类型

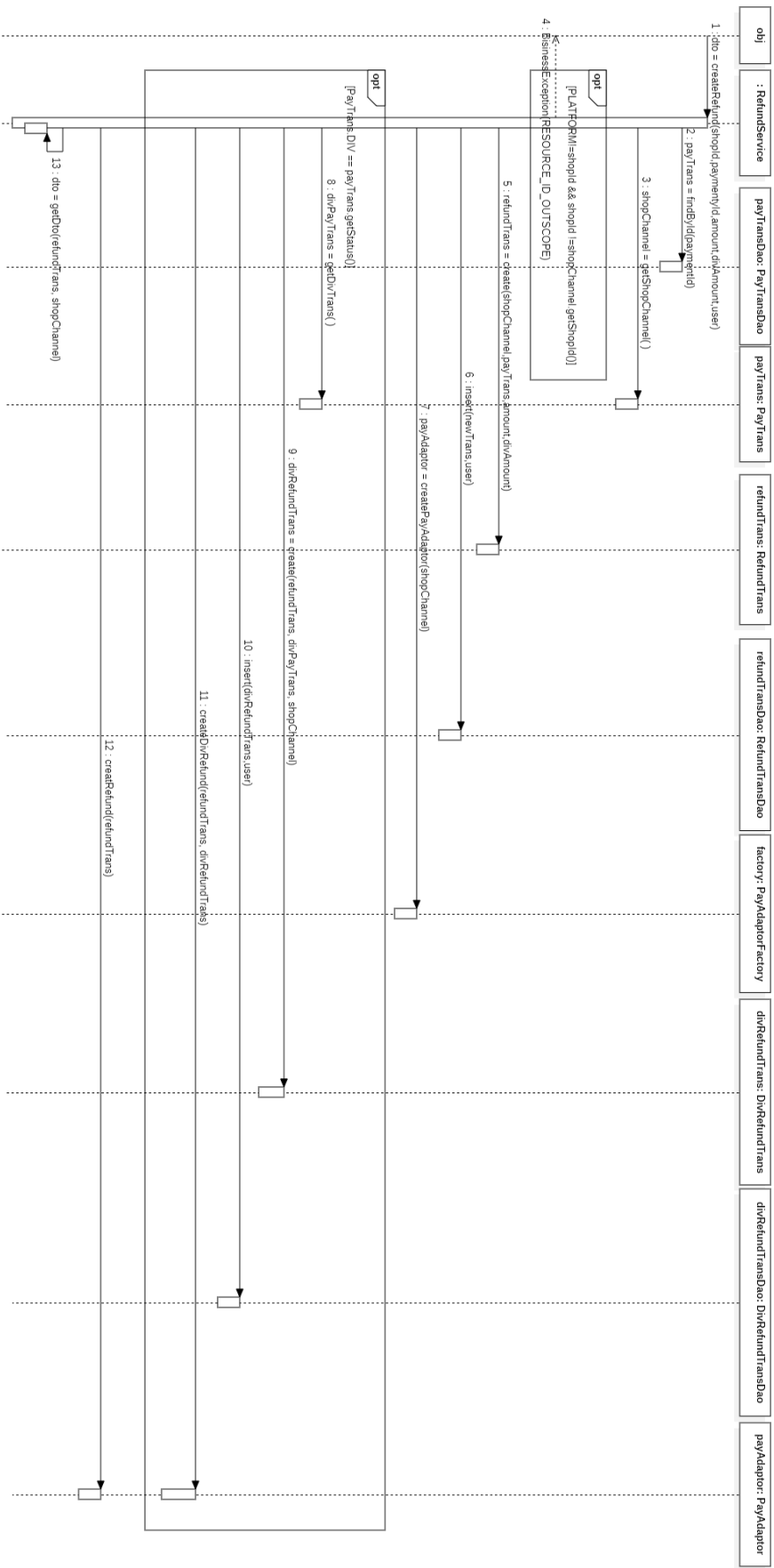


1、设计类图合计 15 分

- 依赖关系 (3 分)
- 关联关系 (5 分)
- 类的定义 (7 分)，其中方法的定义 (3 分)

2、时序图合计 15 分，关注以下问题

- 时序图所有对象和变量的命名是否前后一致，并与设计类图和代码对应 (5 分)
- 时序图的格式是否正确 (7 分)
- 消息的格式是否正确，并与设计类图和代码对应 (3 分)



三、 分析题（40 分）：

图 2-10 的设计类图和时序图描述的是 Product 获取关联的 Onsale 对象的设计。

1、 请简述该设计的目标。（10 分）

由于 Product 和 Onsale 是一对多的关系,在对象模型中获取一对多的关系效率较低,因此在对象模型中设计的 Product 和 Onsale 的关系是一对一的。(3 分)设计目标是实现在不同场景下 Product 对象关联不同的 Onsale 的对象,满足系统的开闭原则,可方便扩展和修改。(7 分)

2、 运用 GRASP 方法和软件设计原则分析该设计目标是如何实现的。（30 分）

在三个场景有各自不同的需求,因此在 ProductDao 中分别用 findProductById 实现根据 productId 获取 Product 并关联当前有效的 Onsale 对象;用 findProductByOnsaleId 实现根据 onsaleId 获取关联特定 onsale 的 Product 对象;用 findProductByBeginEnd 实现关联在 beginTime 和 endTime 之间有效的 onsale 对象的 Product 对象。在三个方法方法中虽然关联的 onsale 对象不同,但是统一调用 findWithGetBo 来关联不同的 onsale 对象(2 分)。

由于三个场景需要关联不同的 onsale 对象,在 onsaleDao 中实现了三个方法, findLatestOnsaleByProductId 查到 product 的最近有效的 onsale 对象; findById 查到指定的 onsale 对象; findOverlapOnsale 查到时间段内最早有效的 onsale 对象。但这个三个方法的参数不同,没办法统一为一个接口。因此采用了 GRASP 中的虚拟的方法,引入 ValidOnsaleExecutor、SpecOnsaleExecutor 和 OverlapOnsaleExecutor 对象,将不同的参数存在对象属性里,把对 OnsaleDao 的三个方法的调用统一为无参数的 execute() (5 分)。这样就可以采用 GRASP 中的多态的方法定义 OnsaleExecutor 接口,让三个 Executor 实现 OnsaleExecutor 接口,从而通过 OnsaleExecutor 的接口调用不同的 OnsaleDao 的方法(5 分)。此处的设计满足 Liskov 可替换原则,任意替换 Executor 类不会影响 Product 的 getValidOnsale 方法(2 分)。基于 Liskov 可替换原则系统满足了开闭原则,即未来扩展新的场景时只需增加继承 OnsaleExecutor 接口的 XXXExecutor 类,这是系统的扩展的开放性;如果 OnsaleDao 的对应方法发生了修改,只需修改对应的 Executor 类,不会影响代码的其他部分,这是系统修改的封闭性(5 分)。同时调用者使用的是 OnsaleExecutor 接口,整个设计中都不会直接调用具体的实现类,这满足了依赖倒置原则(2 分)。引起每个 Executor 类发生变化的唯一原因是其对应的 OnsaleDao 的方法发生了变化,这满足了单一职责原则(2 分)。

在 findWithGetBo 中和 findProductById 中,需要在 getBo 和 setBo 方法中根据不同的场景给 Product 对象设置不同的 OnsaleExecutor 对象,以便于在 Product 对象 getValidOnsale 方法中可以获得不同的 Onsale 对象。在这里采用了 GRASP 中的多态的方法,将 getExecutor 方法由三个子类继承,以便于产生三个不同的 OnsaleExecutor 对象(5 分)。同样满足 Liskov 可替换原则和系统的开闭原则(2 分)。