

**东北师范大学本科生课程作业**

**（**2020春季学期）

**课程名称： 软件质量保证与测试**

**作业题目： 商场收银系统**

**任课老师： 侯鲲**

**姓 名： 梁莉莉**

**专 业： 软件工程**

**学 号： 2017011581**

**学 院： 信息科学与技术学院**

**年 月 日： 2020年5月20日**

目录

[1 项目基本信息 3](#_Toc40952903)

[2项目实现功能 3](#_Toc40952904)

[3 项目结构 4](#_Toc40952905)

[4 Strategy模式分析 4](#_Toc40952906)

[4.1 策略（Strategy）模式 4](#_Toc40952907)

[4.2 模式基本结构图 5](#_Toc40952908)

[4.3具体分析 5](#_Toc40952909)

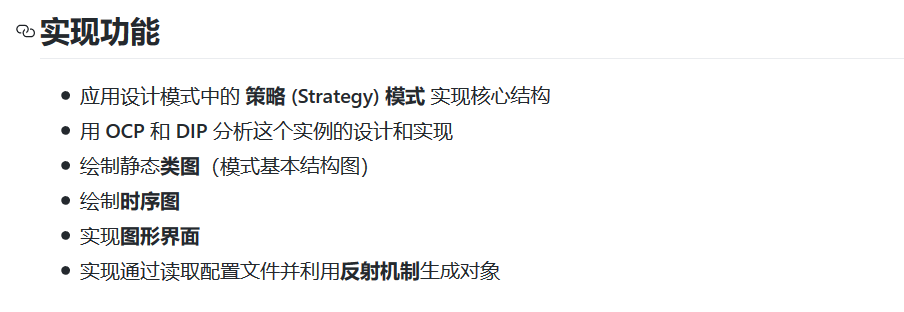
[4.4时序图 6](#_Toc40952910)

[5 运行结果 6](#_Toc40952911)

# 1 项目基本信息



# 2项目实现功能



# 3 项目结构

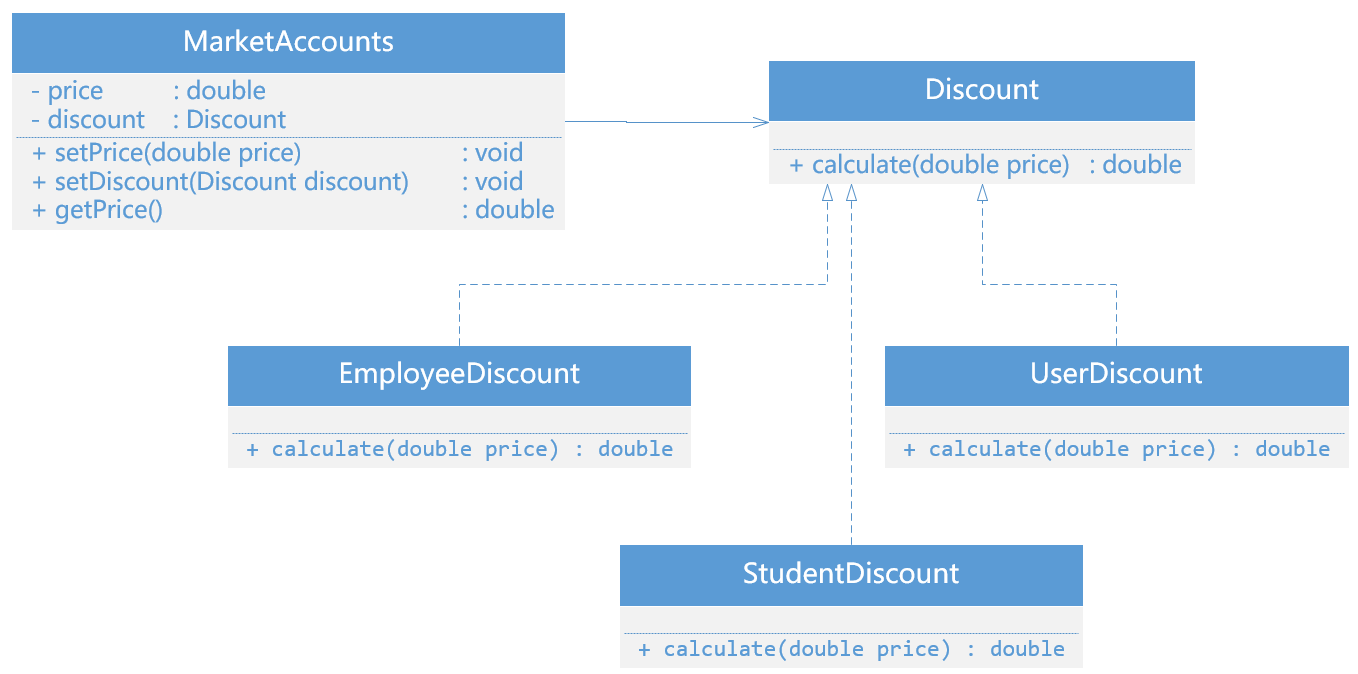


# 4 Strategy模式分析

## 4.1 策略（Strategy）模式

1. 定义：定义一系列算法类，将每一个算法封装起来，并让它们可以相互替换。策略模式让算法独立于使用它的客户而变化。它也被称为政策模式，是一种行为型模式。
2. 角色：
3. Context（环境类）：环境类是使用算法的角色，它在解决某个问题（即实现某个方法）时可以采用多种策略。在环境类中维持一个对抽象策略类的引用实例，用于定义所采用的策略。
4. Strategy（抽象策略类）：它为所支持的算法声明了抽象方法，是所有策略类的父类，它可以是抽象类或具体类，也可以是接口。环境类通过抽象策略类中声明的方法在运行时调用具体策略类中实现的算法。
5. ConcreteStrategy（具体策略类）：它实现了在抽象策略类中声明的算法，在运行时，具体策略类将覆盖在环境类中定义的抽象策略类对象，使用一种具体的算法实现某个业务处理。

## 4.2 模式基本结构图

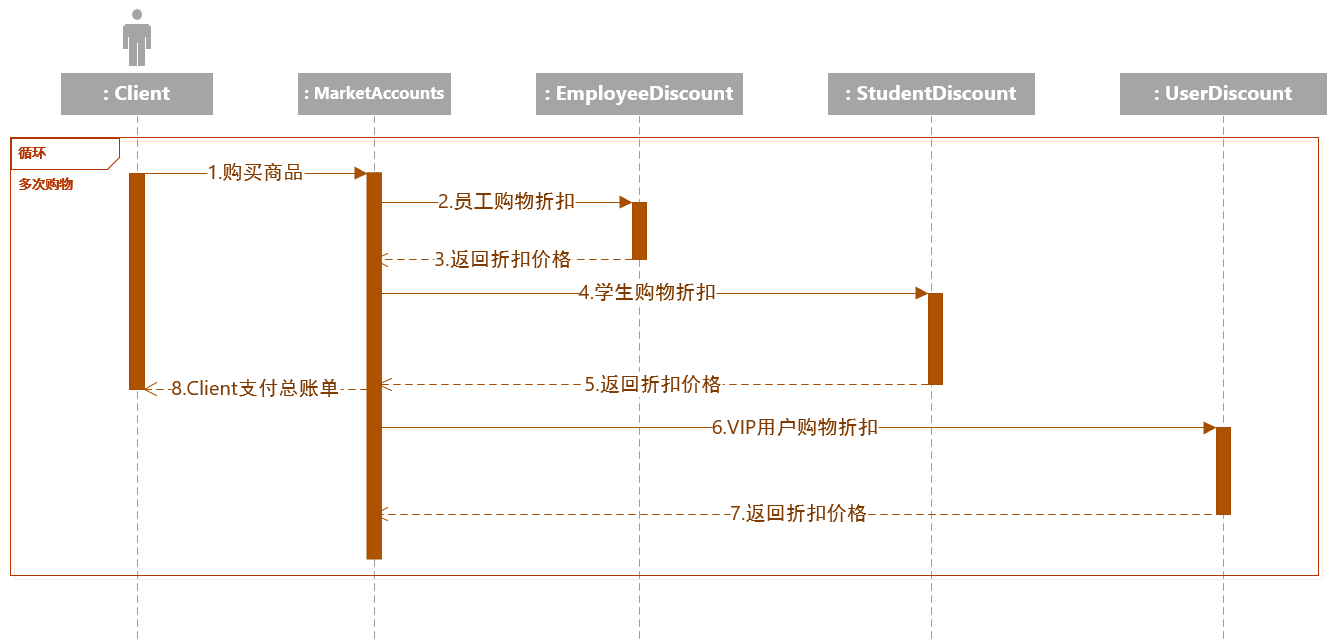


## 4.3具体分析

1. 在本例中，MarketAccounts 充当环境类角色，Discount 充当抽象策略角色， EmployeeDiscount、StudentDiscount 和 UserDiscount 充当具体策略角色。
2. 将算法的定义放在专门的 Discount 策略类中，每一个具体策略类封装了一种实现算法，使用算法的 MarketAccounts 环境类针对抽象策略类进行编程，符合“依赖倒转 (DIP) 原则”。
3. 用户可以在不修改原有系统的基础上选择具体算法或行为，也可灵活地增加新的算法或行为，符合“开放-封闭原则 (OCP) 原则”。
4. 此外，为了提高系统的灵活性和可扩展性，将具体策略类的类名存储在配置文件 (路径： src\com\nenu\market\res\config.xml) 中，通过工具类 XmlUtil 来读取配置文件并反射生成对象。

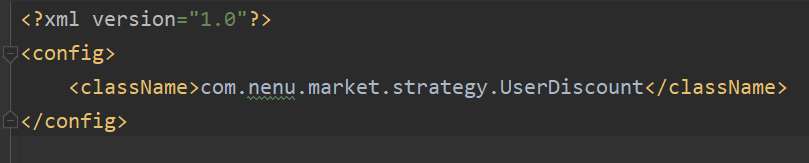
## 4.4时序图

Client选购商品，确定商品价格与选够商品的数量，MarketAccounts确定折扣方式，进行打折，并返回折扣价格。Client可在商场中多次购物，进行循环。

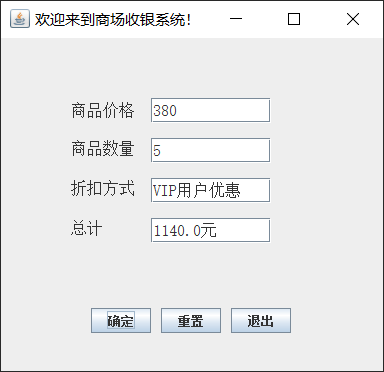


# 5 运行结果

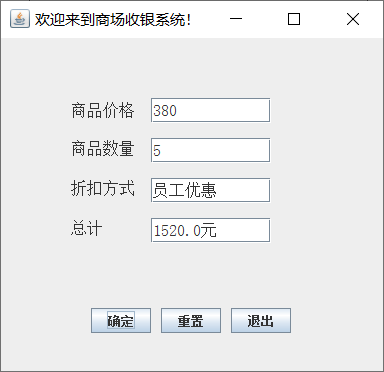
1. 在配置文件中选择具体策略类：例如选择VIP用户优惠（6折优惠，并增加积分）



1. 运行界面：用户在图形界面中填入**商品价格**与**选购的商品数量**，点击确定，页面返回折扣后的总价格



1. 可在配置文件中更换折扣方式
2. 员工折扣（8折优惠）：



1. 学生折扣（每满100减免10元）：

