

本 科 生 实 验 报 告

**实验课程 软件体系结构与设计**

**学院名称 计算机与网络安全学院（牛津布鲁克斯学院）**

**专业名称 软件工程**

**学生姓名 徐睿航**

**学生学号 202013160210**

**指导教师 单武扬**

**实验地点 C075-05-32**

**实验成绩**

**二〇二二年四月­——二〇二二年六月**

# 实验十 行为型模式应用

## **一、实验名称**

行为型模式应用

## **二、实验目的**

1. 理解“行为型模式”的基本概念、基本原理和应用场景

2. 掌握模板方法模式（Template Method）、观察者模式（Observer）的使用方法

## **三、实验器材**

1. 计算机一台
2. Rational Rose 工具软件
3. Eclipse集成开发环境

## **四、实验内容**

1. 模板
2. 定义一个电子商务网站的购物支付模板：

流程为：用户登陆、选择商品、计算总价、支付、配送

1. 该电子商务网站的用户分为了3个不同的级别

普通用户：

不打折

会员用户：

打8折

可以累计积分，积分按1元积1分计算

积分可以用户日后购物抵扣货款，抵扣规则按照100分对应1元抵扣

VIP用户：

打6折

可以累计积分，积分按照1元积1分计算

积分可以抵扣用户日后购物的货款，抵扣规则按照100分对应 1元抵扣

1. 商品分为两类

电子商品：

无需配送，故不收取配送费用，也没有配送环节

实物商品：

需要物流配送，故收取配送费用每单10元，有物流配送环节

1. 请按照需求，设计出该电子商务网站的购物业务流程
2. 观察者
3. 电子商务网站上的商品价格随时在变化，当我们看中一个商品的时候，我们希望价格变化时通知我们，故选择了关注此商品
4. 当商品价格低于我们的关注价格时进行通知

每个设计作业都要求包含2部分：UML类图和代码实现（java）

## **五、实验步骤**

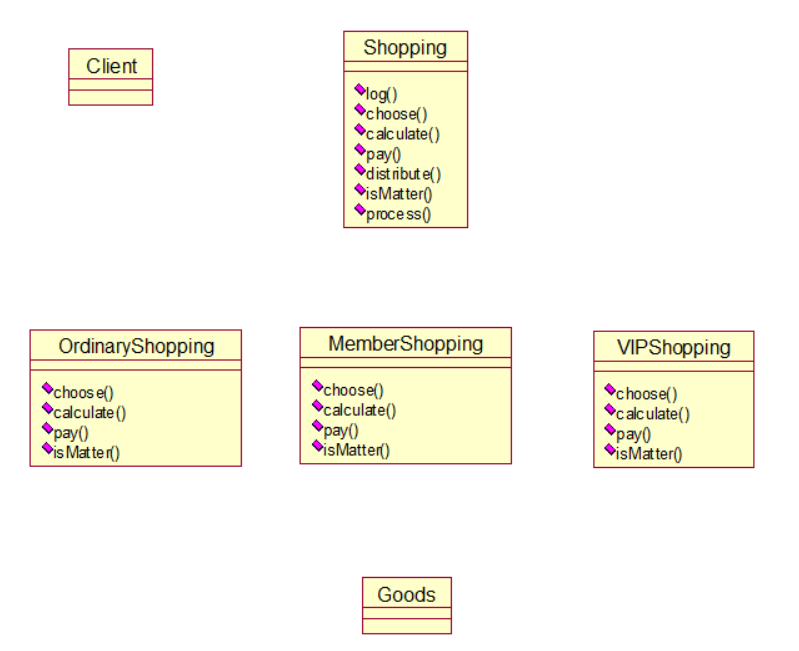
1. 模板

分析：

在购物支付中，Shopping类（抽象类角色）定义抽象的原语操作，具体的子类VIPShopping、MemberShopping、OrdinaryShopping类（具体类角色）将重定义它们以实现购物支付算法各步骤，同时实现一个process()方法（模板方法），定义一个算法的骨架，process()方法不仅调用原语操作，也调用定义在Shopping或其他对象中的操作，OrdinaryShopping类、MemberShopping类、VIPShopping类（具体类角色）实现原语操作以完成购物支付算法中与特定子类相关的步骤。

1. 画UML类图

添加Client、Shopping、OrdinaryShopping、MemberShopping、VIPShopping、Goods类；



为它们建立关系，Client类与Shopping类之间的关联关系， OrdinaryShopping、MemberShopping、VIPShopping类与Shopping类之间的继承关系，与Goods类之间的关联关系。

图示

描述已自动生成

1. 编写Java代码

编写Goods类，其中有数据成员type表示商品种类，name表示商品名称，price表示商品价格，还有getter()、setter()方法；

编写User类，其中有数据成员id表示用户id，level表示用户级别，points表示用户积分，还有getter()、setter()方法；

编写Shopping类，抽象类，其数据成员为User类的对象user和Goods类的goods，还有choose()、calculate()、pay()抽象方法，isMatter()钩子方法，log()、distribute()、process()具体方法；

编写OrdinaryShopping类继承Shopping类并实现抽象方法和钩子方法；

VIPShopping、MemberShopping类与之类似；

编写Client类，进行购物支付。

1. 观察者

分析：

在电子商务网站上，有Subject（抽象观察目标）和Observer（抽象观察者角色），选择关注Goods（具体观察目标），当它价格低于关注的价格时会通知User（具体观察者角色）。

1. 画UML类图

添加Subject、Observer、Goods、User类；

图示

描述已自动生成

为它们建立关系，Subject类与Goods类之间的继承关系，Observer类与User类之间的继承关系，Subject类与Observer类之间的聚合关系，User类与Goods类之间的关联关系。

图示

描述已自动生成

1. 编写Java代码

编写Observer类，抽象类，有成员Goods类对象goods，有Update()方法；

编写User类，继承Observer类，实现Update()方法；

编写Subject类，抽象类，其中有数据成员为ArrayList<Observer>的observers，有attach()、detach()、Notify()方法；

编写Goods类继承Subject类，有数据成员originalPrice和currentPrice分别表示最先的价格和现在的价格，有getter()、setter()方法；

编写Client类，进行关注商品，降价通知。

## **六、实验效果或核心代码**

1. 模板

图示

描述已自动生成

Goods.java

**package** templete;

**public** **class** Goods {

**private** String name;

**private** String type;

**private** **double** price;

**public** Goods(String name, String type, **double** price) {

**this**.name = name;

**this**.type = type;

**this**.price = price;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getType() {

**return** type;

}

**public** **void** setType(String type) {

**this**.type = type;

}

**public** **double** getPrice() {

**return** price;

}

**public** **void** setPrice(**double** price) {

**this**.price = price;

}

}

User.java

**package** templete;

**public** **class** User {

**private** String id;

**private** String level;

**private** **double** points;

**public** User(String id, String level, **double** points) {

**this**.id = id;

**this**.level = level;

**this**.points = points;

}

**public** String getId() {

**return** id;

}

**public** **void** setId(String id) {

**this**.id = id;

}

**public** String getLevel() {

**return** level;

}

**public** **void** setLevel(String level) {

**this**.level = level;

}

**public** **double** getPoints() {

**return** points;

}

**public** **void** setPoints(**double** points) {

**this**.points = points;

}

}

Shopping.java

**package** templete;

**public** **abstract** **class** Shopping {

User user;

Goods goods;

**public** **void** process(Goods cgoods) {

log();

choose(cgoods);

calculate();

pay();

**if**(isMatter()) {

distribute();

}

}

**public** **void** log() {

System.***out***.println(user.getId() + "号用户登录！");

}

**public** **abstract** **void** choose(Goods cgoods);

**public** **abstract** **void** calculate();

**public** **abstract** **void** pay();

**public** **boolean** isMatter() {

**return** **false**;

}

**public** **void** distribute() {

System.***out***.println("进行配送" + goods.getName());

}

}

OridianryShopping.java

**package** templete;

**public** **class** OrdinaryShoping **extends** Shopping {

**public** **void** choose(Goods cgoods) {

**this**.goods = cgoods;

System.***out***.println("普通用户选择了" + goods.getName());

}

**public** **void** calculate() {

**double** total;

**if**(isMatter()) {

total = goods.getPrice() + 10;

System.***out***.println("商品价格" + goods.getPrice() + "元，普通用户不打折，还需配送费10元，共需支付：" + total + "元");

}

**else** {

total = goods.getPrice();

System.***out***.println("商品价格" + goods.getPrice() + "普通用户不打折，共需支付：" + total + "元");

}

}

**public** **void** pay() {

System.***out***.println("进行支付！");

}

**public** **boolean** isMatter() {

**if**(goods.getType() == "实物商品") {

**return** **true**;

}

**else** {

**return** **false**;

}

}

}

MemberShopping.java

**package** templete;

**public** **class** MemberShopping **extends** Shopping {

**public** **void** choose(Goods cgoods) {

**this**.goods = cgoods;

System.***out***.println("会员用户选择了" + goods.getName());

}

**public** **void** calculate() {

**double** total;

**if**(isMatter()) {

total = 0.8 \* goods.getPrice() + 10;

System.***out***.println("商品价格" + goods.getPrice() + "元，会员用户打8折，还需配送费10元，共需支付：" + total + "元");

}

**else** {

total = 0.8 \* goods.getPrice();

System.***out***.println("商品价格" + goods.getPrice() + "会员用户打8折，共需支付：" + total + "元");

}

}

**public** **void** pay() {

**double** points = 0.8 \* goods.getPrice();

System.***out***.println("进行支付！本单增加积分：" + points + "分！");

points += user.getPoints();

user.setPoints(points);

}

**public** **boolean** isMatter() {

**if**(goods.getType() == "实物商品") {

**return** **true**;

}

**else** {

**return** **false**;

}

}

}

VIPShopping.java

**package** templete;

**public** **class** VIPShopping **extends** Shopping {

**public** **void** choose(Goods cgoods) {

**this**.goods = cgoods;

System.***out***.println("VIP用户选择了" + goods.getName());

}

**public** **void** calculate() {

**double** total;

**if**(isMatter()) {

total = 0.6 \* goods.getPrice() + 10;

System.***out***.println("商品价格" + goods.getPrice() + "元，VIP用户打6折，还需配送费10元，共需支付：" + total + "元");

}

**else** {

total = 0.6 \* goods.getPrice();

System.***out***.println("商品价格" + goods.getPrice() + "会员用户打6折，共需支付：" + total + "元");

}

}

**public** **void** pay() {

**double** points = 0.6 \* goods.getPrice();

System.***out***.println("进行支付！本单增加积分：" + points + "分！");

points += user.getPoints();

user.setPoints(points);

}

**public** **boolean** isMatter() {

**if**(goods.getType() == "实物商品") {

**return** **true**;

}

**else** {

**return** **false**;

}

}

}

Client.java

**package** templete;

**public** **class** Client {

**public** **static** **void** main(String[] args) {

User user = **new** User("135", "会员用户", 0);

Goods cgoods = **new** Goods("零食","实物商品", 46.5);

Shopping shopping = **null**;

**if**(user.getLevel() == "普通用户") {

shopping = **new** OrdinaryShoping();

}

**else** **if**(user.getLevel() == "会员用户") {

shopping = **new** MemberShopping();

}

**else** **if**(user.getLevel() == "VIP用户") {

shopping = **new** VIPShopping();

}

shopping.user = user;

shopping.process(cgoods);

}

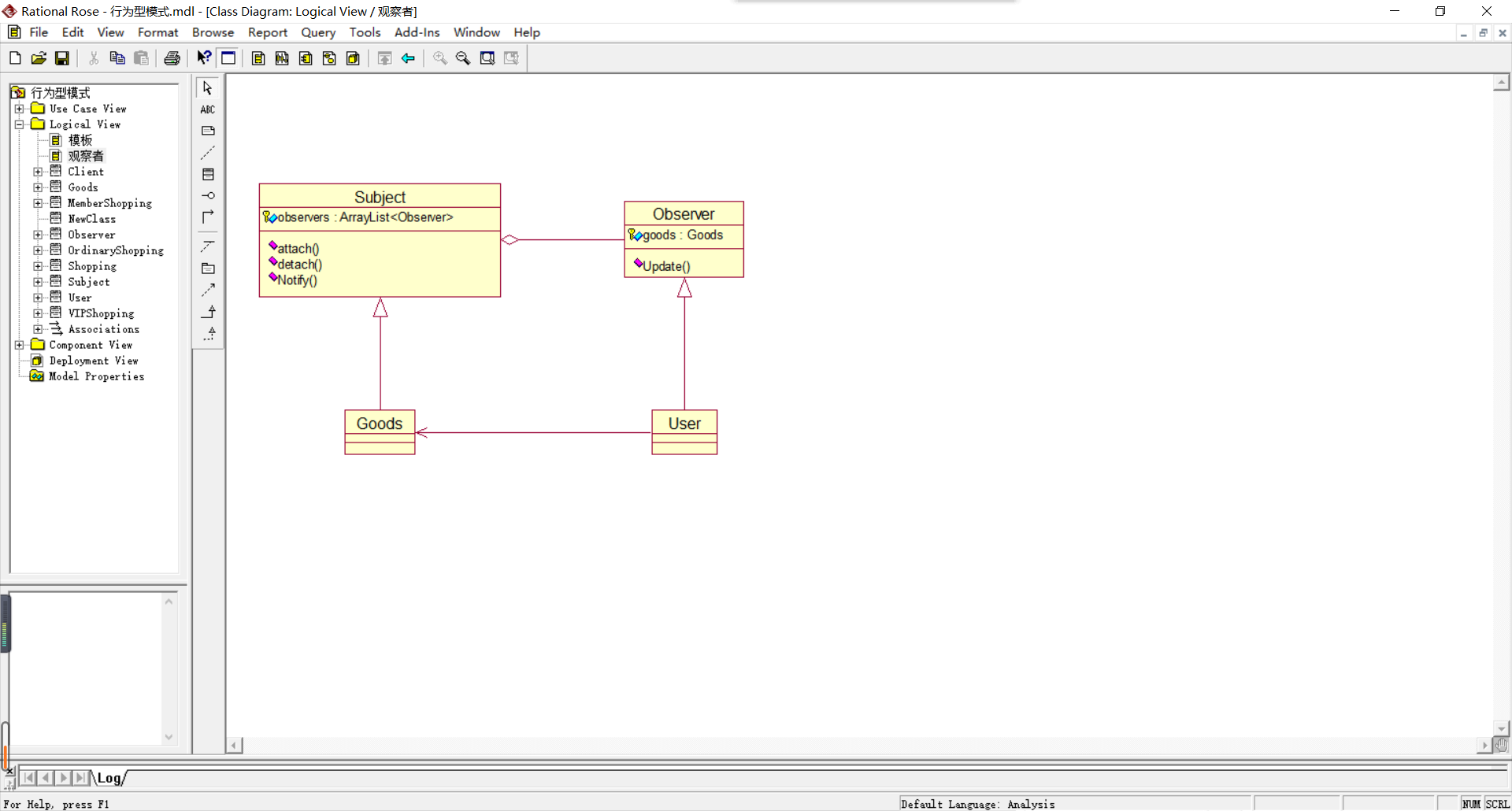
}

运行结果：

文本

描述已自动生成

1. 观察者



Observer.java

**package** observer;

**public** **abstract** **class** Observer {

**protected** Goods goods;

**public** **abstract** **void** Update();

}

Subject.java

**package** observer;

**import** java.util.ArrayList;

**public** **abstract** **class** Subject {

**protected** ArrayList<Observer> observers = **new** ArrayList<Observer>();

**public** **void** attach(Observer observer) {

observers.add(observer);

}

**public** **void** detach(Observer observer) {

observers.remove(observer);

}

**public** **void** Notify() {

**for**(Observer observer:observers) {

observer.Update();

}

}

}

Goods.java

**package** observer;

**public** **class** Goods **extends** Subject {

**private** **double** originalPrice;

**private** **double** currentPrice;

**public** **double** getOriginalPrice() {

**return** originalPrice;

}

**public** **void** setOriginalPrice(**double** originalPrice) {

**this**.originalPrice = originalPrice;

}

**public** **double** getCurrentPrice() {

**return** currentPrice;

}

**public** **void** setCurrentPrice(**double** currentPrice) {

**this**.currentPrice = currentPrice;

}

}

User.java

**package** observer;

**public** **class** User **extends** Observer {

**public** **void** Update() {

**double** orginalPrice = goods.getOriginalPrice();

**double** currentPrice = goods.getCurrentPrice();

**if**(currentPrice < orginalPrice) {

System.***out***.println("商品降价了，通知用户查看");

}

**else** {

System.***out***.println("商品未降价，不通知用户");

}

}

}

Client.java

**package** observer;

**public** **class** Client {

**public** **static** **void** main(String[] args) {

Subject subject = **new** Goods();

Observer observer = **new** User();

subject.attach(observer);

Goods goods = **new** Goods();

observer.goods = goods;

goods.setOriginalPrice(150);

goods.setCurrentPrice(133);

subject.Notify();

}

}

运行结果：

图片包含 文本

描述已自动生成

|  |  |
| --- | --- |
| **学生实验心德** | 通过这次实验我明白了行为者模式的应用。  学生（签名）：徐睿航  2022年6月10日 |
| **指导教师评语** | 成绩评定：  指导教师（签名）：  年 月 日 |