实验九 观察者模式的应用

1 实验目的

1) 掌握外观模式（Observer）的特点

2) 分析具体问题，使用外观模式进行设计。

2 实验内容和要求

网上商店中如果商品（product）在名称（name）、价格（price）等方面有变化，系统能自动通知会员，将是网上商店区别传统商店的一大特色。如何设计实现？ 说明你所选择的设计模式，画出类关系图并指明各个类的角色。应用外观模式，用C#或Java控制台应用程序改进该设计。绘制该模式的UML图。

3 UML图

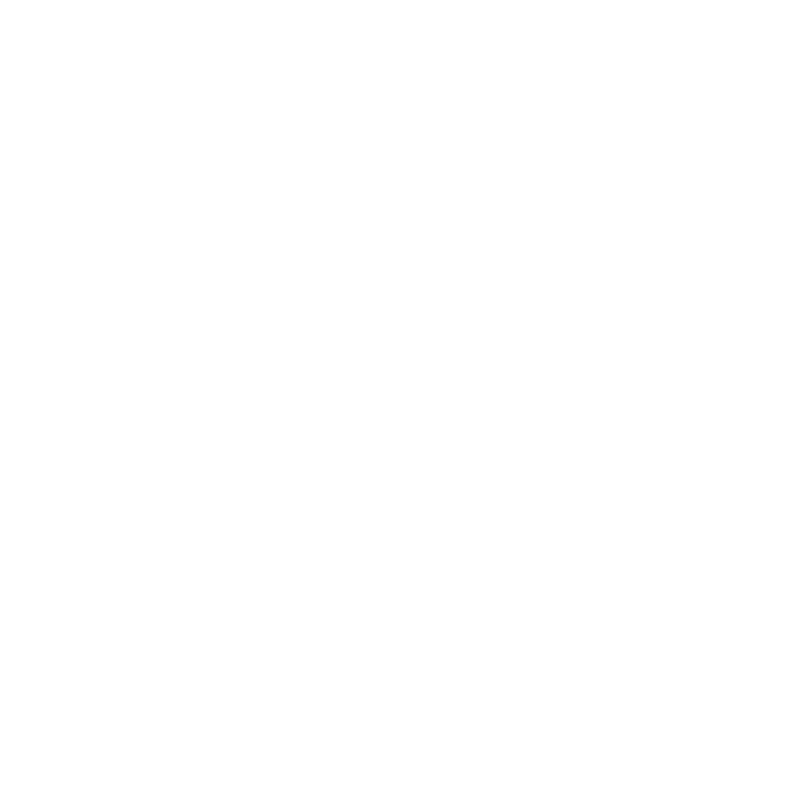


图9.3.1 观察者模式类图

4 代码实现

public class Product {  
 private String name;  
 private double price;  
 List<Observer> observers = new ArrayList<Observer>();  
  
 public Product(String name, double price) {  
 this.name = name;  
 this.price = price;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 notifyObservers();  
 }  
  
 public void setPrice(double price) {  
 this.price = price;  
 notifyObservers();  
 }  
  
 void notifyObservers() {  
 for (Observer o : observers) {  
 o.response(this);  
 }  
 }  
  
 public void addObserver(Observer observer) {  
 observers.add(observer);  
 }  
  
 public void deleteObserver(Observer observer) {  
 observers.remove(observer);  
 }  
}

interface Observer {  
 void response(Object o);  
}

public class Main {  
 public static void main(String[] args) {  
 Product p = new Product("小米手机", 5999.99);  
  
 p.addObserver((o) -> {  
 o = (Product) o;  
 System.*out*.println("A: 我来看看是降价了吗？");  
 });  
  
 p.addObserver((o)->{  
 o = (Product) o;  
 System.*out*.println("B: 我来看看是改名了吗？");  
 });  
  
 p.setName("小米12 Ultra");  
 p.setPrice(49999.99);  
 }  
}

**测试结果**

A: 我来看看是降价了吗？

B: 我来看看是改名了吗？

A: 我来看看是降价了吗？

B: 我来看看是改名了吗？

5 要点总结

1. 当一个对象被修改时，则会自动通知它的依赖对象。属于行为型模式。

实验十 策略模式的应用

1 实验目的

1) 掌握策略模式（Strategy）的特点

2) 分析具体问题，使用策略模式进行设计。

2 实验内容和要求

在图书销售时，根据不同类型的图书有不同的折扣，计算金额时必须区别对待，例如计算机类图书7折，英语类图书6折。应用策略模式，用C#或Java控制台应用程序实现该设计。绘制该模式的UML图。

3 UML图

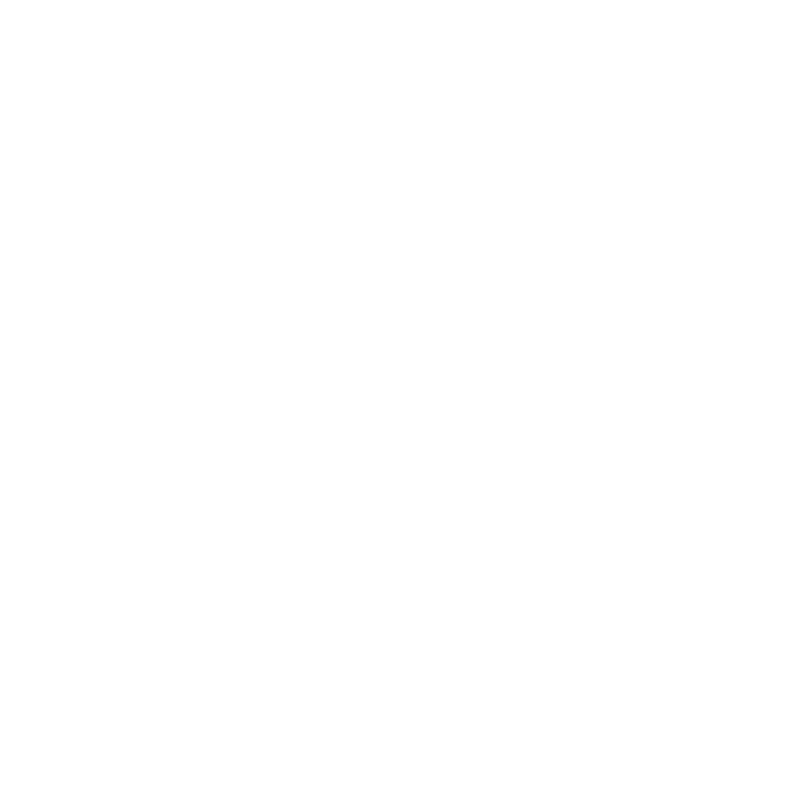


图10.3.1 策略模式类图

4 代码实现

public class Book {  
 String name;  
 Double price;  
 Discount discount;  
  
 public Book(String name, Double price, Discount discount) {  
 this.name = name;  
 this.price = price;  
 this.discount = discount;  
 }  
  
 public void setPrice(Double price) {  
 this.price = price;  
 }  
  
 public void setDiscount(Discount discount) {  
 this.discount = discount;  
 }  
  
 public double getDiscountedPrice() {  
 return discount.compute(price);  
 }  
}

interface Discount {  
 double compute(Double price);  
}

public class Main {  
 public static void main(String[] args) {  
 Discount CS = price -> price \* 0.7;  
 Discount En = price -> price \* 0.6;  
  
 Book b1 = new Book("具体数学", 100.0, CS);  
 Book b2 = new Book("大学英语语法", 288.9, En);  
  
 System.*out*.println(b1.getDiscountedPrice());  
 System.*out*.println(b2.getDiscountedPrice());  
 }  
}

**测试结果**

70.0

173.33999999999997

5 要点总结

1. 策略模式(Strategy Pattern)指的是定义一系列的算法，把它们一个个封装起来，目的就是将算法的使用与算法的实现分离开来