

**软件体系结构与设计**  **报告**

**报告题目 模板,观察者，迭代**

**学院名称 网络安全学院**

**专业名称 软件工程**

**学生姓名 肖骏**

**学生学号 201606150208**

**任课教师 刘明哲**

**报告成绩**

**教务处 制**

年 月 日

模板模式

# 实验分析

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

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

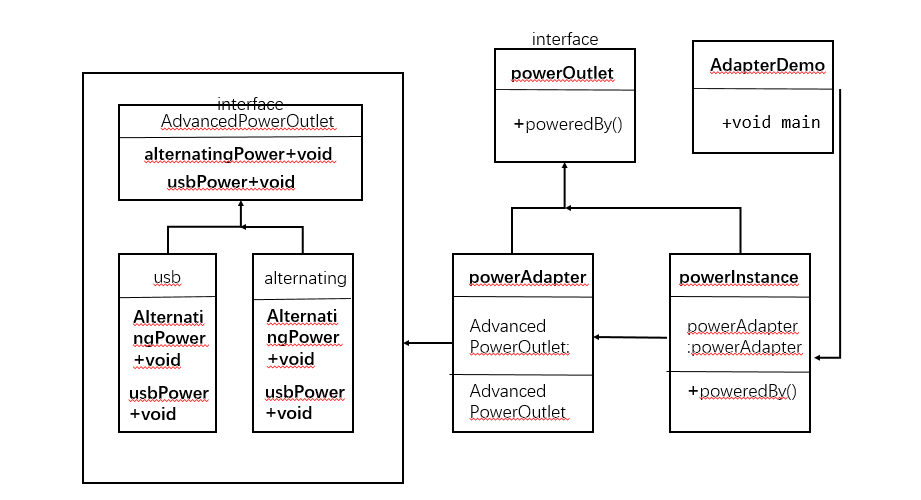
打6折

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

* 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

# UML图



# Java实现与运行结果

**package** TemplateMode;

**public** **class** NormalIUser **extends** ShoppingPayment{

**boolean** n;

**double** price;

@Override

**void** LogIn() {

System.***out***.println("Login...");

}

@Override

**boolean** SelectCommodity(String commodity) {

System.***out***.println("SelectCommodity...");

**if**(commodity=="电子商品") {

n=**false**;

System.***out***.println("Your commodity is 电子商品");

}**else** {

n=**true**;

System.***out***.println("Your commodity is 实物商品");

}

**return** n;

}

@Override

**void** CalculateTotalPrice(**double** price) {

System.***out***.println("No discount...");

System.***out***.println("CalculateTotalPrice...");

**this**.price=price;

}

@Override

**void** Payment(String commodity,**double** price) {

**if**(n) {

System.***out***.println("The total price is:"+**this**.price+10);

System.***out***.println("Payment...");

}**else** {

System.***out***.println("The total price is:"+**this**.price);

System.***out***.println("Payment...");

}

}

@Override

**void** Delivery() {

System.***out***.println("Delivery...");

}

}

**abstract** **void** LogIn();

**abstract** **boolean** SelectCommodity(String commdoityClass);

**abstract** **void** CalculateTotalPrice(**double** price);

**abstract** **void** Payment(String commodity,**double** price);

**abstract** **void** Delivery();

**public** **final** **void** start(**double** price,String commodity) {

**if**(commodity=="实物商品") {

LogIn();

SelectCommodity(commodity);

CalculateTotalPrice(price);

Payment(commodity,price);

Delivery();

}**else** {

LogIn();

SelectCommodity(commodity);

CalculateTotalPrice(price);

Payment(commodity,price);

}

}

}

**package** TemplateMode;

**public** **class** SuperVipUser **extends** ShoppingPayment{

**double** price;

**int** integral;

**boolean** n;

String commodity;

@Override

**void** LogIn() {

System.***out***.println("Login...");

}

@Override

**boolean** SelectCommodity(String commodity) {

System.***out***.println("SelectCommodity...");

**if**(commodity=="电子商品") {

n=**false**;

System.***out***.println("Your commodity is 电子商品");

}**else** {

n=**true**;

System.***out***.println("Your commodity is 实物商品");

}

**return** n;

}

@Override

**void** CalculateTotalPrice(**double** price) {

**this**.price=price;

System.***out***.println("40% off...");

System.***out***.println("CalculateTotalPrice...");

integral=(**int**)**this**.price;

System.***out***.println("Your integral is:"+integral);

**if**(integral>100) {

**this**.price=**this**.price\*0.6-integral/100;

integral=integral%100;

}**else** {

**this**.price=**this**.price\*0.6;

}

}

@Override

**void** Payment(String commodity,**double** price) {

**if**(n) {

**this**.price=**this**.price+10;

System.***out***.println("The total price is:"+**this**.price+10);

System.***out***.println("Payment...");

}**else** {

System.***out***.println("The total price is:"+**this**.price);

System.***out***.println("Payment...");

}

}

@Override

**void** Delivery() {

System.***out***.println("Delivery...");

}

}

**package** TemplateMode;

**public** **class** TemplateModeDemo {

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

ShoppingPayment normalUser=**new** NormalIUser();

ShoppingPayment vipUser=**new** VipUser();

ShoppingPayment superVipUser=**new** SuperVipUser();

System.***out***.println("普通用户。。。");

normalUser.start(100,"电子商品");

System.***out***.println("会员用户。。。");

vipUser.start(200,"实物商品");

System.***out***.println("vip用户。。。");

superVipUser.start(100,"电子商品");

}

}

**package** TemplateMode;

**public** **class** VipUser **extends** ShoppingPayment{

**int** integral;

String commodity;

**boolean** n;

**double** price;

@Override

**void** LogIn() {

System.***out***.println("Login...");

}

@Override

**boolean** SelectCommodity(String commodity) {

System.***out***.println("SelectCommodity...");

**if**(commodity=="电子商品") {

n=**false**;

System.***out***.println("Your commodity is 电子商品");

}**else** {

n=**true**;

System.***out***.println("Your commodity is 实物商品");

}

**return** n;

}

@Override

**void** CalculateTotalPrice(**double** price) {

**this**.price=price;

System.***out***.println("20% off...");

System.***out***.println("CalculateTotalPrice...");

integral=(**int**)**this**.price;

System.***out***.println("Your integral is:"+integral);

**if**(integral>100) {

**this**.price=**this**.price\*0.8-integral/100;

integral=integral%100;

}**else** {

**this**.price=**this**.price\*0.8;

}

}

@Override

**void** Payment(String commodity,**double** price) {

**if**(n) {

**this**.price=**this**.price+10;

System.***out***.println("The total price is:"+**this**.price);

System.***out***.println("Payment...");

}**else** {

System.***out***.println("The total price is:"+**this**.price);

System.***out***.println("Payment...");

}

}

@Override

**void** Delivery() {

System.***out***.println("Delivery...");

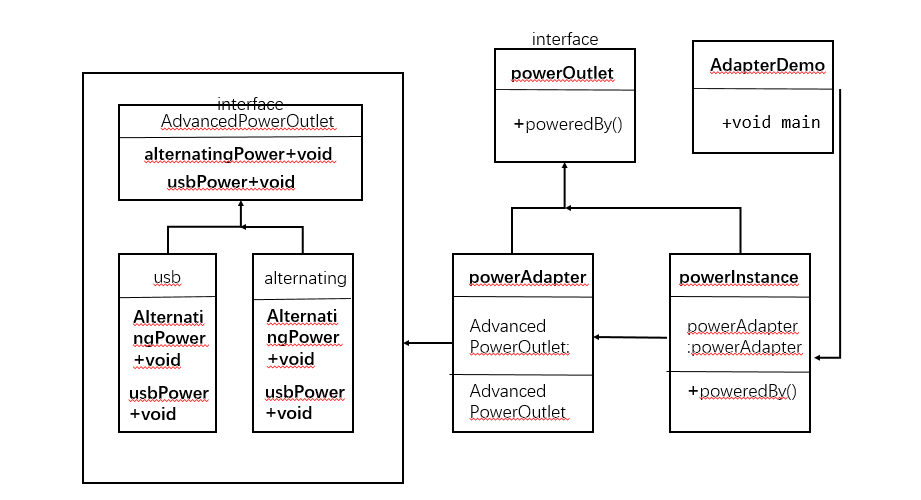
}

观察者模式

# 1.实验分析

* 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
* 当商品价格低于我们的关注价格时进行通知

# 2.UML图



# 3.Java实现与运行结果

**package** ObserverMode;

**public** **class** CommodityObserver **extends** PriceObserver{

@Override

**public** **void** PriceUpdate() {

System.***out***.println("the price is "+subject.getPrice());

}

**public** CommodityObserver(Subject subject) {

**this**.subject=subject;

**this**.subject.attach(**this**);

}

}

**package** ObserverMode;

**import** java.util.List;

**import** java.util.ArrayList;

**public** **abstract** **class** PriceObserver {

**protected** Subject subject;

**public** **abstract** **void** PriceUpdate();

}

**package** ObserverMode;

**public** **class** PriceObserverModeDemo {

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

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

**new** CommodityObserver(subject);

subject.setPrice(10);

subject.setPrice(15);

}

}

package ObserverMode;

import java.util.List;

import java.util.ArrayList;

public class Subject {

private List<PriceObserver> priceObservers=new ArrayList<PriceObserver>();

private double price;

public double getPrice() {

return price;

}

public void setPrice(double price) {

this.price=price;

notifyAllPriceObservers();

}

public void attach(PriceObserver priceObserver) {

priceObservers.add(priceObserver);

}

public void notifyAllPriceObservers() {

for(PriceObserver priceObserver:priceObservers) {

priceObserver.PriceUpdate();

}

}

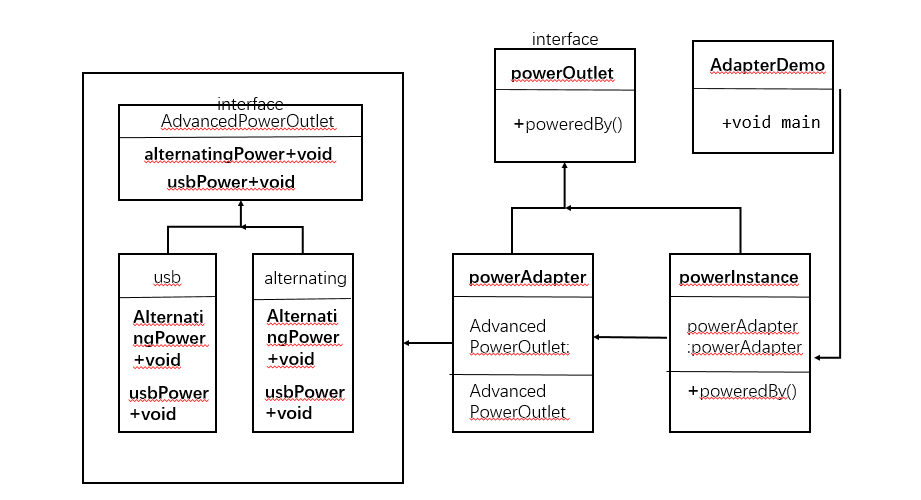
}

迭代器模式

# 1.实验分析

* 我们有很多种商品，用户可以选择不同的排序方式进行浏览
  + 1、按销量
  + 2、按好评
  + 3、按时间
  + 4、其他任意可扩展的方式，如按购买用户所属地等等

# 2.UML图



# 3.Java实现与运行结果

**package** IteratorMode;

**public** **interface** Container {

**public** Iterator getIterator();

}

**package** IteratorMode;

**public** **interface** Iterator {

**public** **boolean** hasNext();

**public** Object next();

}

**package** IteratorMode;

**import** java.sql.Date;

**public** **class** IteratorModeDemo {

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

SaleVolumesRepository saleVolumesRepository=**new** SaleVolumesRepository();

PraiseRepositary praiseRepositary=**new** PraiseRepositary();

TimeRepository timeRepository =**new** TimeRepository();

**for**(Iterator iter1=saleVolumesRepository.getIterator();iter1.hasNext();) {

**int** saleVolumes=(**int**)iter1.next();

System.***out***.println("saleVolumes:"+saleVolumes);

}

**for**(Iterator iter2=praiseRepositary.getIterator();iter2.hasNext();) {

**int** Praise=(**int**)iter2.next();

System.***out***.println("Praise:"+Praise);

}

**for**(Iterator iter3=timeRepository.getIterator();iter3.hasNext();) {

Date time=(Date)iter3.next();

System.***out***.println("time:"+time);

}

}

}

**package** IteratorMode;

**public** **class** PraiseRepositary **implements** Container{

**public** **int** PraiseNumbers[]= {78,23,55,32};

@Override

**public** Iterator getIterator() {

// **TODO** Auto-generated method stub

**return** **new** PraiseIterator();

}

**private** **class** PraiseIterator **implements** Iterator{

**int** index;

@Override

**public** **boolean** hasNext() {

**if**(index<PraiseNumbers.length) {

**return** **true**;

}

**return** **false**;

}

@Override

**public** Object next() {

**if**(**this**.hasNext()) {

**return** PraiseNumbers[index++];

}

**return** **null**;

}

}

}

**package** IteratorMode;

**public** **class** SaleVolumesRepository **implements** Container{

**public** **int** saleVolume[]= {44,22,224,653};

@Override

**public** Iterator getIterator() {

// **TODO** Auto-generated method stub

**return** **new** SaleVolumesIterator();

}

**private** **class** SaleVolumesIterator **implements** Iterator{

**int** index;

@Override

**public** **boolean** hasNext() {

**if**(index<saleVolume.length) {

**return** **true**;

}

**return** **false**;

}

@Override

**public** Object next() {

**if**(**this**.hasNext()) {

**return** saleVolume[index++];

}

**return** **null**;

}

}

}

**package** IteratorMode;

**import** java.sql.Date;

**public** **class** TimeRepository **implements** Container{

Date t1=**new** Date(100000000);

Date t2=**new** Date(200000000);

Date t3=**new** Date(300000000);

Date t4=**new** Date(400000000);

**public** Date Time[] = {t1,t2,t3,t4};

@Override

**public** Iterator getIterator() {

// **TODO** Auto-generated method stub

**return** **new** TimeIterator();

}

**private** **class** TimeIterator **implements** Iterator{

**int** index;

@Override

**public** **boolean** hasNext() {

**if**(index<Time.length) {

**return** **true**;

}

**return** **false**;

}

@Override

**public** Object next() {

**if**(**this**.hasNext()) {

**return** Time[index++];

}

**return** **null**;

}

}

}