

JAVA程序语言设计

BooK Store 设计模式2



2018-1-3

[裴子祥 计科七班 学号2015211921]

[指导老师：李劼]

目录

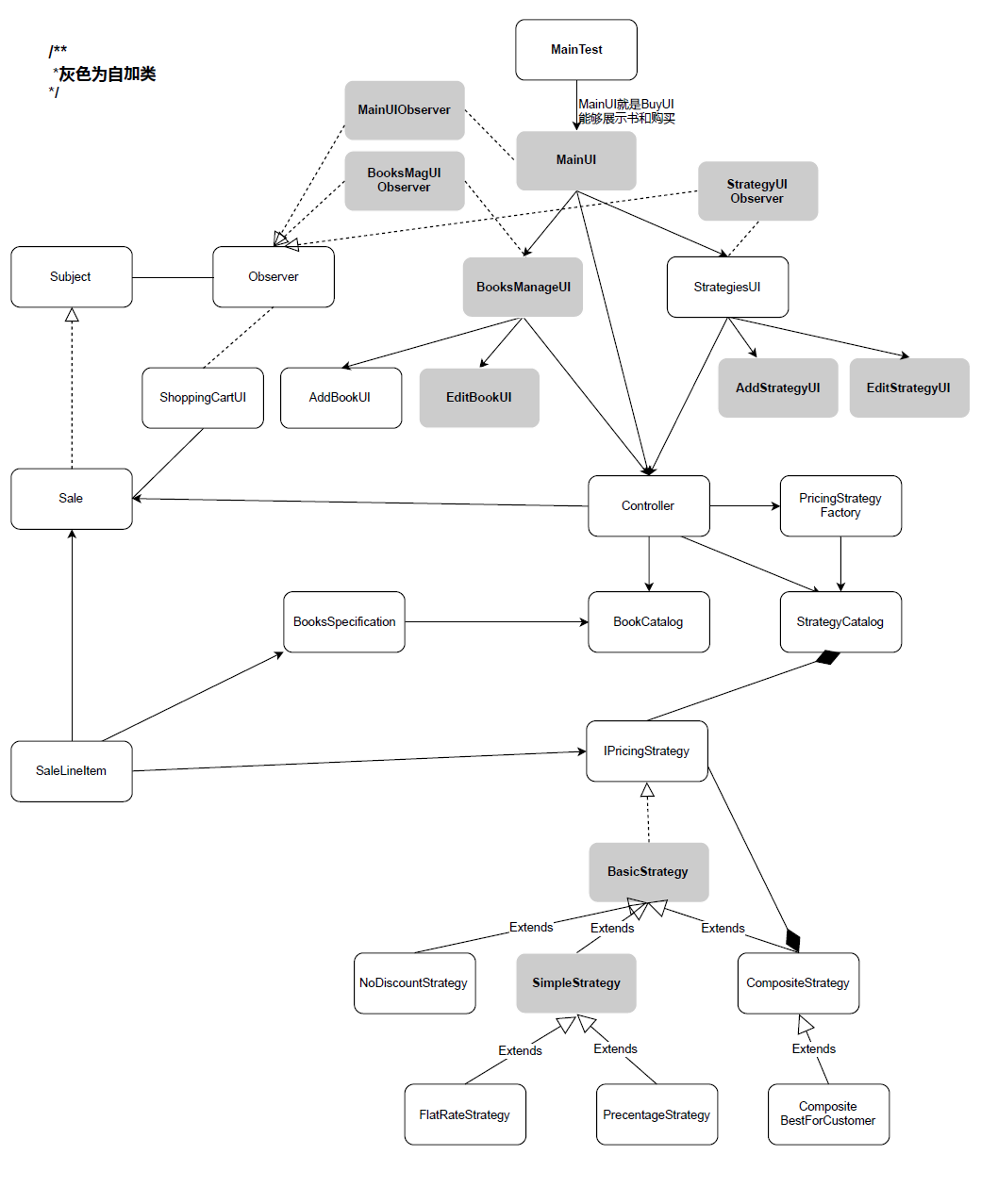
[一．类图 1](#_Toc503175309)

[二． 类说明 2](#_Toc503175310)

**前言**

本次实验通过使用观察者模式、工厂模式、策略模式、组合模式、单例模式等设计模式以及图形化界面、类的继承与接口等 Java 知识实现了一个简单的BookStore，在完成基本要求的基础上，添加了策略与图书的增删编辑的功能，也完善了许多小细节，如各种错误处理，还有删除策略后相关图书价格的变化等。对这次实践也比较满意，确实学到了许多知识，理解了模式对程序的健壮性与可扩展性的支持，也初步掌握 Java 面向对象的编程方法以及设计模式中封装变化、多用组合，少用继承和为交互对象之间的松耦合设计而努力等原则。

# **一．类图**



1. 类说明

|  |  |
| --- | --- |
| **public class MainTest** | |
| **方法名** | **方法功能说明** |
| public static void main(String args[]) | Main函数，声明并调用MainUI以打开初始界面 |

|  |  |
| --- | --- |
| **public class MainUI extends JFrame** | |
| **属性名** | **属性说明** |
| private Observer observer | 观察者，当图书发生改变时，更新显示 |
| private JTable table | 显示图书目录表格 |
| private ShoppingCartUI shopCartUI | 购物车窗口 |
| private JSpinner spinner | Spin控件，控制购买数量 |
| private JLabel bookSelected | 显示当前选中书本信息Label |
| class ModelTable extends DefaultTableModel | 内部类，用于显示JTable数据信息 |
| class MainUIObserver implements Observer | 内部类，MainUI的观察者，Observer的一个实现  其内部对update()具体化，用于更新MainUI |
| **方法名** | **方法功能说明** |
| public MainUI() | MainUI构造函数 |
| private void initializeUI() | 初始化MainUI界面，布局、界面、初始化信息等  布局中有一个JTable显示图书信息  一个Spin空间设置购买数量  一个Label显示当前当前选中图书  一个“打开购物车”按钮，一个“添加购物车”按钮  上方菜单栏，一级菜单“管理”，能够二级菜单进入“图书管理”与策略管理；一级菜单“购物车”，能够二级菜单“清空购物车”。 |
| private void openShopCartUI() | 打开购物车窗口，点击“打开购物车”按钮响应事件 |
| public void closeShopCartUI() | 关闭购物车窗口,把内部shopCartUI置空 |
| private void addToShopCart() | 点击“添加至购物车”按钮响应事件 |

|  |  |
| --- | --- |
| **public class ShoppingCartUI extends JFrame** | |
| **属性名** | **属性说明** |
| private Observer observer | 观察者，当发生Sale信息改变时，更新显示 |
| private JTable table | 显示当前选购图书信息 |
| private JLabel cost | 用于显示Sale(购物车)内总价Label |
| class ModelShopCart extends DefaultTableModel | 内部类，用于显示JTable数据信息 |
| class CartObserver implements Observer | 内部类，ShoppingCartUI观察者，Observer的一个实现  其内部对update()具体化，用于更新ShoppingCartUI |
| **方法名** | **方法功能说明** |
| public ShoppingCartUI(MainUI parent) | 构造函数，构造在主界面之上 |
| private void initializeUI() | 初始化ShoppingCartUI界面  JTable显示当前Sale（购物车内容）  一个“删除”按钮，能够删除选中记录  一个“关闭购物车”按钮  一个显示当前总价的Label |
| public void closeShopCart() | 关闭购物出车窗口，点击 “关闭购物车”按钮 时响应该事件 |
| private void delSaleLineItem() | 选中购物车内一条记录，点击“删除”按钮，响应事件，则将选中图书从购物车内移除 |

|  |  |
| --- | --- |
| **public class BooksManageUI extends JFrame** | |
| **属性名** | **属性说明** |
| private Observer observer | 观察者，当图书信息改变时，更新显示 |
| private JTable table | 显示当前所有图书信息 |
| class ModelTable extends DefaultTableModel | 内部类，用于显示JTable数据信息 |
| class BooksMagUIObserver implements Observer | 内部类，BooksManageUI观察者，Observer的一个实现  其内部对update()具体化，用于更新BooksManageUI |
| **方法名** | **方法功能说明** |
| public BooksManageUI(MainUI parent) | 构造函数，构造在主界面之上 |
| private void initializeUI() | 初始化BooksManageUI界面  JTable显示所有图书信息  一个“删除”按钮，能够删除选中图书  一个“退回主界面”按钮  一个“编辑”按钮  一个“添加图书”按钮 |
| private void delBook() | 点击“删除”按钮，响应事件，从BookCatalog删除选中图书 |
| private void addBook() | 点击“添加新书”按钮，响应事件，弹出图书添加界面 |
| private void editBook() | 点击“编辑选中图书”按钮，响应事件，弹出图书编辑界面 |

|  |  |
| --- | --- |
| **public class AddBookUI extends JFrame** | |
| **属性名** | **属性说明** |
| private JTextField bookIsbnField | 输入ISBN号文本编辑栏 |
| private JTextField bookTitleField | 输入书名文本编辑栏 |
| private JTextField bookPriceField | 输入原价文本编辑栏 |
| private JComboBox<String> bookTypeCombox; | 书类选择栏 |
| **方法名** | **方法功能说明** |
| public AddBookUI(BooksManageUI parent) | 构造函数，构造在BooksManageUI界面之上 |
| private void initializeUI() | 初始化AddBookUI界面  除了输入外，还有“确认添加”按钮、“退回”按钮 |
| private void confirmAddBook() | 点击“确认添加”按钮，响应事件，将要添加图书加入BookCatalog中。 |

|  |  |
| --- | --- |
| **public class EditBookUI extends JFrame** | |
| **属性名** | **属性说明** |
| private JTextField bookIsbnField | 输入ISBN号文本编辑栏，不可修改 |
| private JTextField bookTitleField | 输入书名文本编辑栏，可修改 |
| private JTextField bookPriceField | 输入原价文本编辑栏，可修改 |
| private JComboBox<String> bookTypeCombox; | 书类选择栏，不可修改 |
| **方法名** | **方法功能说明** |
| public EditBookUI(BooksManageUI parent, String isbn) | 构造函数，构造在BooksManageUI界面之上，并且传入当前选中图书信息ISBN号，以确定编辑哪本图书信息 |
| private void initializeUI(String isbn) | 初始化EditBookUI界面 |
| private void confirmEditBook() | 点击“确认修改”按钮，响应事件，将要新的图书信写回BookCatalog中 |

|  |  |
| --- | --- |
| **public class StrategiesUI extends JFrame** | |
| **属性名** | **属性说明** |
| private Observer observer | 观察者，当策略信息改变时，更新显示 |
| private JTable table | 显示当前所有策略信息 |
| class ModelTable extends DefaultTableModel | 内部类，用于显示JTable数据信息 |
| class StrategyUIObserver implements Observer | 内部类，StrategiesUI观察者，Observer的一个实现  其内部对update()具体化，用于更新StrategiesUI |
| **方法名** | **方法功能说明** |
| public StrategiesUI(MainUI parent) | 构造函数，构造在主界面之上 |
| private void initializeUI() | 初始化StrategiesUI界面  JTable显示所有策略信息  一个“删除”按钮，能够删除选中策略  一个“退回主界面”按钮  一个“编辑策略”按钮  一个“添加策略”按钮 |
| private void delStrategy() | 点击“删除”按钮，响应事件，从StrategiesCatalog删除选中策略 |
| private void addStrategy() | 点击“添加新策略”按钮，响应事件，弹出策略添加界面 |
| private void editStrategy() | 点击“编辑选中策略”按钮，响应事件，弹出策略编辑界面 |

|  |  |
| --- | --- |
| **public class AddStrategyUI extends JFrame** | |
| **属性名** | **属性说明** |
| private JTextField strategyNameField | 输入策略名称文本编辑栏 |
| private JTextField strategyValueField | 输入策略值文本编辑栏 |
| private JComboBox<String> bookTypeCombox | 适用图书种类选择栏 |
| private JComboBox<String> strategyTypeCombox | 策略种类选择栏 |
| **方法名** | **方法功能说明** |
| public AddStrategyUI(StrategiesUI parent) | 构造函数，构造在StrategiesUI界面之上 |
| private void initializeUI() | 初始化AddStrategyUI界面 |
| private void confirmAddStrategy() | 点击“确认添加”按钮，响应事件，将新的策略信息写回StrategiesCatalog中 |

|  |  |
| --- | --- |
| **public class EditStrategyUI extends JFrame** | |
| **属性名** | **属性说明** |
| private JTextField strategyNameField | 输入策略名称文本编辑栏 |
| private JTextField strategyValueField | 输入策略值文本编辑栏 |
| private JComboBox<String> bookTypeCombox | 适用图书种类选择栏 |
| private JComboBox<String> strategyTypeCombox | 策略种类选择栏，不可修改 |
| **方法名** | **方法功能说明** |
| public EditStrategyUI(StrategiesUI parent, int strategyNum) | 构造函数，构造在StrategiesUI界面之上，并且传入当前选中策略编号，以确定编辑哪个策略 |
| private void initializeUI(int strategyNum) | 初始化EditStrategyUI界面 |
| private void confirmEditStrategy(int strategyNum, int bType) | 点击“确认修改”按钮，响应事件，将更新后的策略信息写回StrategiesCatalog中。 |

|  |  |
| --- | --- |
| **public class Controller** | |
| **属性名** | **属性说明** |
| private Sale sale | 销售记录（购物车所有商品记录） |
| private static Controller instance | 单例模式，只有一个全局控制器 |
| **方法名** | **方法功能说明** |
| public static Controller getInstance() | 获取当前控制器 |
| public Sale getSale() | 获取当前购书记录 |
| public IPricingStrategy getStrategy(int strategyNum) | 根据策略号获得策略 |
| public void addStrategy(IPricingStrategy strategy) | 传递添加策略信息 |
| public void removeStrategy(int strateNum) | 传递 根据策略编号 移除策略信息 |
| public void editStrategy(int strategyNum, IPricingStrategy strategy) | 传递 策略编号 与 策略信息 以更新策略 |
| public BookSpecification getBook(String isbn) | 根据ISBN号获得书籍 |
| public boolean addBook(BookSpecification book) | 添加图书至BookCatalog |
| public boolean removeBook(String isbn) | 根据ISBN号移除书籍 |
| public boolean addSaleItem(String isbn, int copies) | 根据ISBN号与数量，添加一条记录到Sale中 |
| public void addSaleItem(SaleLineItem item) | 传入一条记录，把它加入Sale中 |
| public boolean removeSaleItem(SaleLineItem item) | 根据传入记录，移除一条记录 |
| public boolean removeSaleItem(int saleLineNum) | 根据记录编号，移除该条记录 |
| public void editBook(BookSpecification book) | 根据传入图书，修改此书信息 |
| public void updateBook() | 根性图书观察者 |
| public void updateStrategy() | 更新策略观察者 |
| public void updateSale() | 更新购书记录 |
| public void clearShopCart() | 清空购物车 |
| public void messageBox(String mess, String title) | 全局信息弹出窗口，“成功或错误” |
| public void initInfo() | 初始化信息，将一些图书和策略信息写入 |
| public String bookTypeToStr(int bookType) | 将书的种类由整型对应成字符串型 |

|  |  |
| --- | --- |
| **public class BookSpecification** | |
| **属性名** | **属性说明** |
| public static final int NOT\_EXIST = 0 | 类别常量，此书不存在 |
| public static final int NOT\_COMPUTER\_TEACHING = 1 | 类别常量，非教材计算机 |
| public static final int TEACHING = 2 | 类别常量，教材类 |
| public static final int COMIC = 3 | 类别常量，连环画 |
| public static final int HEALTH = 4 | 类别常量，健康类 |
| public static final int OTHER = 5 | 类别常量，其它 |
| private String isbn | ISBN号 |
| private String title | 书名 |
| private double price | 原价 |
| private int type | 书的类别 |
| **方法名** | **方法功能说明** |
| public BookSpecification(String isbn,  String title,double price, int type) | 构造函数，根据基本信息创建一本书实例 |
| public BookSpecification() | 基本构造函数，创建一个空书实例，类别为NOT\_EXIST |
| public String getTypeName() | 根据书的种类获取对应种类字符串名称 |
| 各个私有属性的get/set函数 | 基本get/set功能 |

|  |  |
| --- | --- |
| **public class BookCatalog** | |
| **属性名** | **属性说明** |
| private static BookCatalog instance = null | 单例模式 |
| private ArrayList<BookSpecification> books | 存所有书籍信息 |
| private ArrayList<Observer> observerList | 观察者模式，所有观察者放入列表 |
| **方法名** | **方法功能说明** |
| private BookCatalog() | 构造函数 |
| public static BookCatalog getInstance() | 获取全局BookCatalog,只能有一个 |
| public boolean addBook(BookSpecification book) | 根据传入图书，将其加入书籍列表中 |
| public ArrayList<BookSpecification> getBooks() | 获取整个图书列表 |
| public BookSpecification getBook(String isbn) | 根据ISBN号获取图书 |
| public boolean removeBook(String isbn) | 根据ISBN号从书籍列表移除此图书，移除成功返回true，否则返回false |
| public void editBook(BookSpecification book) | 根据传入图书，修改其图书信息（ISBN不变） |
| public void registerObserver(Observer observer) | 添加观察者 |
| public boolean removeObserver(Observer observer) | 移除观察者 |
| public void notifyObservers() | 更新每个观察者界面 |

|  |  |
| --- | --- |
| **public interface IPricingStrategy** | |
| **方法名** | **方法功能说明** |
| public double getSubTotal(SaleLineItem item) | 算取一条记录的价格 |
| public int getStrategyType() | 获取策略种类 |
| public int getStrategyNum() | 获取策略编号 |
| public String getStrategyName() | 获取策略名称 |
| public BookSpecification getBook(String isbn) | 根据ISBN号获取图书 |
| public String getStrategyValue() | 获取策略优惠关键值（字符串） |
| public String getStrategyTypeStr() | 获取策略类型 |
| public int getBookType() | 获取策略适用图书类别 |
| public boolean setStrategyName(String name) | 设置策略名称 |
| public boolean setStrategyValue(String value) | 设置策略优惠关键值 |
| public boolean setBookType(int booktype) | 设置适用图书类别 |

|  |  |
| --- | --- |
| **public abstract class BasicStrategy implements IPricingStrategy** | |
| **属性名** | **属性说明** |
| public static int strategyNumSequence = 0 | 策略编号基值，每生成一个策略，自动+1 |
| protected int strategyNum | 策略编号 |
| protected String strategyName | 策略名称 |
| protected int bookType | 适用图书类别 |
| **方法名** | **方法功能说明** |
| public abstract double getSubTotal(SaleLineItem item) | 获取一条记录使用策略后的价格 |
| public boolean setStrategyName(String name) | 设置策略名称实现 |
| public String getStrategyName() | 获取策略名称实现 |
| public int getStrategyNum() | 获取策略编号实现 |
| public boolean setBookType(int type) | 设置适用图书类型实现 |
| public int getBookType() | 获取适用图书类型实现 |
| public abstract String getStrategyValue() | 获取策略关键值 |

|  |  |
| --- | --- |
| **public class NoDiscountStrategy extends BasicStrategy** | |
| **属性名** | **属性说明** |
| private static NoDiscountStrategy instance = null | 单例模式，全局只有一个无策略的策略（即普通状态） |
| protected int strategyNum | 策略编号 |
| protected String strategyName | 策略名称 |
| protected int bookType | 适用图书类别 |
| **方法名** | **方法功能说明** |
| private NoDiscountStrategy() | 构造函数，初始化 无策略的性质 |
| public static NoDiscountStrategy getInstance() | 获取全局“无策略” |
| public String getStrategyTypeStr() | 返回策略名称“无优惠” |
| public String getStrategyValue() | 返回策略关键字“无任何折扣” |
| public int getStrategyType() | 获取策略种类，0，0号策略 |
| public double getSubTotal(SaleLineItem item) | 计算原价下，记录的价格 |

|  |  |
| --- | --- |
| **public abstract class SimpleStrategy extends BasicStrategy** | |
| **属性名** | **属性说明** |
| public static final int PERCENTAGE = 1 | 策略类别常量，简单策略中的百分比折扣策略 |
| public static final int FLAT\_RATE = 2 | 策略类别常量，简单策略中的绝对值折扣策略 |
| protected int strategyType | 策略种类 |
| **方法名** | **方法功能说明** |
| public int getStrategyType() | 获得简单策略种类 |
| public String getStrategyTypeStr() | 获得简单策略种类字符串名称 |

|  |  |
| --- | --- |
| **public class PrecentageStrategy extends SimpleStrategy** | |
| **属性名** | **属性说明** |
| private int discountPercentage | 策略关键值，0~100的整数，表明折扣百分比力度 |
| **方法名** | **方法功能说明** |
| public PrecentageStrategy(String strategyName, int discountPercentage) | 百分比折扣策略构造函数 |
| public PrecentageStrategy(int strategyNum, String strategyName, int discountPercentage) | 构造函数，添加策略编号，修改策略属性时用到 |
| public double getSubTotal(SaleLineItem item) | 在百分比折扣下，一条记录的价格 |
| public String getStrategyValue() | 获取策略关键值字符串形式 |
| public boolean setStrategyValue(String value) | 根据参数，设置百分比折扣策略关键值 |

|  |  |
| --- | --- |
| **public class FlatRateStrategy extends SimpleStrategy** | |
| **属性名** | **属性说明** |
| private double discountPerBook | 策略关键值，浮点数，绝对值优惠 |
| **方法名** | **方法功能说明** |
| public FlatRateStrategy(String strategyName, double discountPerBook) | 绝对值优惠策略构造函数 |
| public FlatRateStrategy(int strategyNum , String strategyName, double discountPerBook) | 构造函数，添加策略编号，修改策略属性时用到 |
| public double getSubTotal(SaleLineItem item) | 在绝对值优惠下，一条记录的价格 |
| public String getStrategyValue() | 获取策略关键值字符串形式 |
| public boolean setStrategyValue(String value) | 根据参数，设置绝对值优惠策略关键值 |

|  |  |
| --- | --- |
| **public abstract class CompositeStrategy extends BasicStrategy** | |
| **属性名** | **属性说明** |
| public static final int CompositeBestForCustom = 3 | 策略类别常量，组合策略中的用户最优策略 |
| protected ArrayList<Integer> strategyNUMList | 子策略列表，由策略编号代表组成策略的子策略 |
| protected String values | 策略关键值以（1|2）其它策略编号组成形式表示 |
| **方法名** | **方法功能说明** |
| public CompositeStrategy(String strategyName) | 复杂策略构造函数，传入 |
| public CompositeStrategy(int strategyNum, String strategyName) | 复杂策略构造函数，新传入策略编号，修改策略属性时用到 |
| public abstract double getSubTotal(SaleLineItem item) | 获得记录价格，子类必须实现 |
| private boolean initStrategies() | 根据策略关键值，初始化子策略编号列表 |
| public String getStrategyValue() | 获取策略关键值 |
| public boolean setStrategyValue(String value) | 设置策略关键值 |
| public String getStrategyTypeStr() | 获取“组合优惠”的名称字符串 |
| public int getStrategyType() | 返回策略类别 |

|  |  |
| --- | --- |
| **public class CompositeBestForCustomer extends CompositeStrategy** | |
| **方法名** | **方法功能说明** |
| public CompositeBestForCustomer(int strategyNum, String strategyName) | 继承父类调用函数，构造一个用户最优策略 |
| public CompositeBestForCustomer(String strategyName) | 继承父类调用函数 |
| public double getSubTotal(SaleLineItem item) | 在用户最优策略下，一条记录的价格，是子策略中最便宜的一个值 |

|  |  |
| --- | --- |
| **public class PricingStrategyFactory** | |
| **属性名** | **属性说明** |
| private static PricingStrategyFactory instance = null | 策略工厂，单例模式 |
| private StrategyCatalog catalog | 策略目录 |
| **方法名** | **方法功能说明** |
| private PricingStrategyFactory() | 初始化构造函数，初始化策略目录 |
| public static PricingStrategyFactory getInstance() | 获取策略工厂唯一全局实例 |
| public IPricingStrategy getPricingStrategy(int bookType) | 根据书的类别，获取它使用的策略 |
| public void setCatalog(StrategyCatalog catalog) | 设置策略目录 |

|  |  |
| --- | --- |
| **public class StrategyCatalog** | |
| **属性名** | **属性说明** |
| private HashMap<Integer, IPricingStrategy> strategies | 哈希表存储，策略编号与策略的一一映射 |
| private ArrayList<Observer> observers | 观察者列表 |
| private static StrategyCatalog instance | 单例模式，唯一实例 |
| **方法名** | **方法功能说明** |
| private StrategyCatalog() | 策略目录初始构造 |
| public static StrategyCatalog getInstance() | 获取策略目录唯一全局实例 |
| public void registerObserver(Observer obs) | 新增观察者 |
| public boolean removeObserver(Observer obs) | 移除观察者 |
| public void notifyObserver() | 更新观察者列表中的所有UI |
| public boolean isOccupied(int bookType) | 根据书的种类，判断此书是否已经有策略了，若有返回true，否则返回false |
| public IPricingStrategy getStrategy(int bookType) | 按书的类别找到策略 |
| public IPricingStrategy getStrategyOnNum(int num) | 按策略编号找到该策略 |
| public void addStrategy(IPricingStrategy strategy) | 传入一个策略，并添加至策略目录 |
| public IPricingStrategy removeStrategy(IPricingStrategy strategy) | 传入一个策略，将这个策略从策略目录移除 |
| public void removeStrategy(Integer strateNum) | 传入一个策略编号，将该策略从策略目录移除 |
| public void editStrategy(Integer strategyNum, IPricingStrategy strategy) | 根据传入策略编号，更新策略信息，（策略编号不变） |
| public ArrayList<IPricingStrategy> getStrategies() | 获得所有策略列表 |

|  |  |
| --- | --- |
| **public class SaleLineItem** | |
| **属性名** | **属性说明** |
| private static int numSequence = 0 | 一条购书记录基准值，每加入购物车一次自动+1 |
| private int saleItemNum | 购书记录号 |
| private int copies | 购买数量 |
| private BookSpecification prodSpec | 购买书的实例 |
| private IPricingStrategy strategy | 此书对应的策略 |
| **方法名** | **方法功能说明** |
| public SaleLineItem(int copies, BookSpecification proSpec) | 初始化构造函数，根据购买图书，与其数量，就能生成一条记录 |
| public int getSaleItemNum() | 获取购书记录号 |
| public double getSubTotal() | 获取这条记录的花费金额 |
| public int getCopies() | 获得购书数量 |
| public BookSpecification getProdSpec() | 获取购书实例 |
| public IPricingStrategy getStrategy() | 获取对应策略信息 |
| public void updateStrategy() | 更新记录中的策略 |
| public void setCopies(int copies) | 设置记录中的购书数量 |

|  |  |
| --- | --- |
| **public class Sale implements Subject** | |
| **属性名** | **属性说明** |
| private ArrayList<SaleLineItem> items | 当前购物车内所有购书记录列表 |
| private ArrayList<Observer> observers | 观察者列表 |
| **方法名** | **方法功能说明** |
| public Sale() | 初始化构造函数 |
| public double getTotal() | 获取购物车内所有图书总价 |
| public ArrayList<SaleLineItem> getItems() | 获取购物车内购书记录列表 |
| public void addItem(SaleLineItem item) | 添加一次购书记录，若已经拥有，则只改变数量 |
| public boolean removeItem(SaleLineItem item) | 删除选中的购书记录 |
| public void clear() | 一次性清空购物车 |
| public void updateStrategy(int strategyNum) | 更新策略号为strategyNum的记录 |
| public void updateStrategy() | 更新购物车已经包含的全体策略信息 |
| public void registerObserver(Observer observer) | 添加观察者 |
| public void removeObserver(Observer observer) | 移除观察者 |
| public void notifyObservers() | 更新观察者列表中的全体UI |

|  |  |
| --- | --- |
| **public interface Subject** | |
| **方法名** | **方法功能说明** |
| public void registerObserver(Observer observer) | 添加某个观察者 |
| public void removeObserver(Observer observer) | 移除某个观察者 |
| public void notifyObservers() | 数据发生变化，更新所有观察者的UI |

|  |  |
| --- | --- |
| **public interface Observer** | |
| **方法名** | **方法功能说明** |
| public void update() | 对相应的UI进行更新，受Subject调用 |