实验三 行为型设计模式实验

一、实验目的

1．综合实例，熟练绘制常见的行为型设计模式结构图。

2. 结合实例，使用 Java 实现常见的行为型设计模式。

3. 通过实验，理解不同行为型设计模式的使用动机，掌握不同行为型设计模式的特点和运用场合，学习如何使用代码实现这些设计模式。

二、实验要求

1．独立完成实验

2．书写实验报告书

三、实验内容

1. 迭代器模式的运用

（1）案例背景：

课堂教学中学习了如何使用迭代器模式来模拟电视遥控器的实现，并使用了内部类的方式来实现迭代器。在实验中，请将迭代器从具体聚合类（电视机类）中分离出来，重新实现电视遥控器的模拟，请画出类图并编程实现。

（2）实现步骤：

- 参照教材中实例，画出电视机遥控器的类图
- 根据类图，实现上述类的具体代码以及用户类 Client，请注意将迭代器分离出来，形成单独的类，此外还需要实现 XMLUtil 类来从 XML 配置文件中读取具体的电视遥控器类。
- 编译并运行代码，实现电视遥控器的模拟。

（3）案例总结：

在以下情况下可以使用迭代器模式：

- 需要访问一个聚合对象的内容而无需暴露它的内部表示时
- 需要为一个聚合对象提供多种遍历方式时
- 为遍历不同的聚合结构提供一个统一的接口，在该接口的实现类中为不同的聚合结构提供不同的遍历方式，而客户端可以一致性地操作该接口

2.观察者模式的运用

（1）案例背景：

某在线股票系统需要提供以下功能：当股票购买者所购买的某只股票价格变化幅度达到 5%时，系统将自动发送通知（包括新价格）给购买该股票的股民。现使用观察者模式设计该系统，绘制类图并编程实现

（2）实现步骤：

- 根据题意，画出在线股票系统的类图，类图中应包括目标类 Stock，抽象观察者 Investor 以及具体观察者 ConcreteInvestor。Stock 类中应该包含添加观察者的功能 attach（），移除观察者的功能 detach（），获取股票名称 getStockName（），设定股票名称 setStockName（），设定股票价格 setPrice（），获取股票功能 getPrice（）以及通知观察者的功能 notifyInvestor（）；观察者应该有能够根据观察目标的改变作出反应的 upDate（）方法
- 根据类图，实现上述类的具体代码以及用户类 Client。
- 编译并运行程序，使得股民能够在价格变化超过 5%的时候收到通知。

（4）案例总结：

在以下情况可以使用观察者模式：

- 一个抽象模型有两个方面，其中一个方面依赖于另一个方面，将这两个方面封装在独立的对象中使它们可以各自独立地改变和复用
- 一个对象的改变将导致一个或多个其他对象发生改变，并且不知道具体有多少对象将发生改变，也不知道这些对象是谁
- 需要在系统中创建一个触发链。

3. 策略模式的运用

（1）案例背景：

在介绍策略模式时，我们讲到了从不同角度出发，可以使用不同的出行策略的例子，教材中已经提供了“旅游出行策略”的类图，用 Java 代码模拟实现“旅游出行策略”实例，要求使用配置文件存储具体策略类的类名。在此基础上再增加一种新的旅游出行方式，如徒步旅行（WalkStrategy），修改原有类图及方法，并编程实现。（教材 403 页第 1 题）

（2）实现步骤：

- 根据书上“旅游出行策略”类图，增加新的徒步旅行方式，画出新的类图。
- 根据类图，编写并实现代码，使用 XMLUtil 类来从 XML 文件中读取相应类名。
- 编译并运行代码，使代码能够模拟旅游出行策略。

（3）案例总结：

在以下情况下可以使用策略模式：

- 如果在一个系统里面有许多类，他们之间的区别仅在于他们的行为，使用策略模式可以动态的让一个对象在许多行为中选择一种行为。
- 一个系统需要动态的在几种算法中选择一种，那么可以将这些算法封装到一个个的具体算法类中，而这些算法类是一个抽象算法类的子类。这些具体算法类有统一的接口，由于多态性原则，客户端可以选择任意一个具体算法类，并只需要维持一个抽象算法类的对象。
- 如果一个对象有很多的行为，可以使用策略模式把这些行为转移到相应的具体策略类里面，这样可以避免使用难以维护的多重条件选择语句。
- 不需要客户端直到复杂的，与算法相关的数据结构，在具体策略类中封装算法和相关的数据结构，提高算法的保密性和安全性。