|  |  |
| --- | --- |
| 实验内容 | 股票交易中心现上线一支股票“New\_Fashion”,用户A，B，C注册并关注了这支股票。当股票价格发生变动的时候，交易中心会推送该股票的变化情况给A,B,C三个用户，请用观察者模式演示该功能。  请设计对应的类图，并给出实现代码。 |
| 类图 |  |
| 实验过程记 录 | 观察者模式的实现关键在于目标类实现通知方法时会调用观察者的响应方法。在这个实例中，我首先定义了一个Observers接口，用来充当抽象观察者类，声明了update()更新状态的方法和getter和setter方法；具体观察者类Investor实现了Observers接口中的方法；再以Observable类作为抽象目标类，定义一个集合用于存储股民，基于ArrayList的add()和remove()方法，定义了增加成员和移除成员的方法，再声明了notice()方法；TradingCenter交易中心类继承了Observable类，实现了通知方法，先打印股票情况更新的信息，再调用集合中每一个股民的update()方法。最后，客户端创建A、B、C三个股民，加入集合，并调用交易中心的通知方法，就实现了观察者模式的主要功能。 |
| 总结 | 观察者模式的优点是实现了表示层和数据逻辑层的分离，在观察目标和观察者之间建立了抽象的耦合，支持广播通信，且符合开闭原则。但是如果观察者很多，通知所有观察者会很费时；如果在观察者和观察目标之间存在循环依赖，可能会导致系统崩溃；而且观察者模式缺乏让观察者知道观察对象如何变化的机制。  在这个例子中，因为只牵涉到股票状态的通知，我没有在具体观察者和具体目标之间维持对象引用。如果情况更加复杂，可能在具体观察者类执行update()方法时还需要用到具体目标类的状态，那么可能需要在具体观察者类中定义一个具体目标类实例，具体观察者和具体目标类之间就可能存在关联关系。如果要扩展不同类型的观察者做出不同的响应，可以在抽象观察者接口下定义不同的具体观察者，并以不同的方法实现update()；如果要扩展不同的交易中心，也可以定义新的具体目标类。 |