**实验四 行为型设计模式**

学生姓名： 黄丹 学 号： 6130116093 专业班级：航天班

实验类型：□ 验证 □ 综合 □ 设计 □ 创新 实验日期： 2019/4/30 实验成绩：

一、实验项目名称

行为型设计模式

二、实验目的

1、了解回顾熟悉行为型设计模式，熟练使用相关设计模式；

2、掌握模式结构，学习运用模式解决实际设计中遇到的问题；

3、熟练使用动态多态。

三、实验基本原理

四、主要仪器设备及耗材

PC微机

Windows 操作系统

IDEA/JDK1.8

Github

五、实验内容

1、某物资管理系统中物资采购需要分级审批，主任可以审批1万元及以下的采购单，部门经理可以审批5万元及以下的采购单，副总经理可以审批10万元及以下的采购单，总经理可以审批20万元及以下的采购单，20万元以上的采购单需要开会确定。现使用职责链模式设计该系统，绘制类图并编程实现。

2、某系统需要提供一个命令集合（注：可使用ArrayList等集合对象实现），用于存储一些列命令对象，并通过该命令集合实现多次undo()和redo()操作，可使用计算器来模拟实现。

3、某教务管理系统中一个班级(Class)包含多个学生(Student)，使用Java内置迭代器实现对学生信息的遍历，要求按学生由大到小的次序输出学生信息。用Java实现该过程。

4、使用中介者模式来说明联合国的作用，要求绘制相应的类图并分析每个类的作用（注：可以将联合国定义为抽象中介者类，联合国下属机构如WTO、WFC、WHO等作为具体中介者类，国家可以作为抽象同事类，而将中国、美国、日本、英国等国家作为具体同事类）。

5、改进“用户信息操作撤销”实例，使得系统可以实现多次撤销操作（可以使用集合对象如HashMap、ArrayList等来实现）。

6、某高校教学管理系统需要实现如下功能，如果某个系的系名发生改变，则该系所有教师和学生的所属系名称也将发生改变。使用Java语言提供的观察者类和观察目标类实现该功能，绘制类图并编程实现。

7、某纸牌游戏软件中，人物角色具有入门级(Primary)、熟练级(Secondary)、高手级(Professional)和骨灰级(Final)四种等级，角色的等级与其积分对应，游戏胜利将增加积分，失败则扣除积分。入门级具有最基本的游戏功能play()，熟练级增加了游戏胜利积分加倍功能doubleScore()，高手级在熟练级基础上再增加换牌功能changeCards()，骨灰级在高手级基础上再添加偷看他人的牌功能peekCards()。现使用状态模式来设计该系统，绘制类图并编程实现。

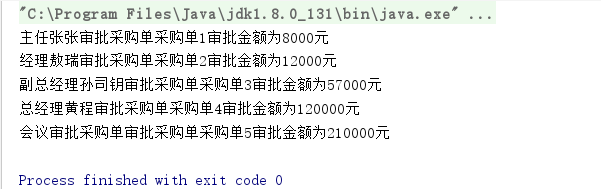
8、设计一个网上书店，该系统中所有的计算机类图书(ComputerBook)每本都有10%的折扣，所有的语言类图书(LanguageBook)每本都有2元的折扣，小说类图书(NovelBook)每100元有10元的折扣。现使用策略模式来设计该系统，绘制类图并编程实现。

9、如果网上商店中商品（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。

10、一部手机，在其接受到来电的时候，会发出声音来提醒主人。而现在我们需要为该手机添加一项功能，在接收来电的时候，产生震动，为令其更加高级，不仅发声，而且振动，而且有灯光闪烁。

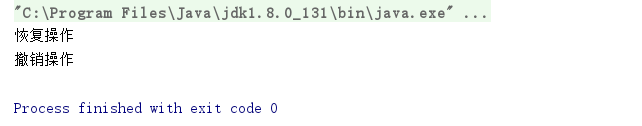
六、实验数据及处理结果

1.物资采购





2.命令集合

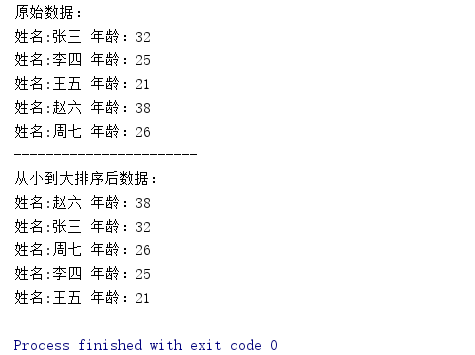


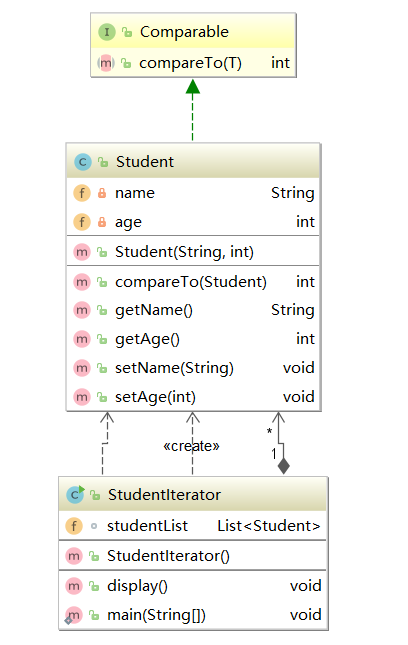


**public class** CompositeCommand **extends** Command{  
 **private** Command[] **commands**;  
  
 **public** CompositeCommand(Command[] commands){  
 **this**.**commands**=commands;  
 }  
  
 **public void** execute(){  
 **for**(**int** i=0;i<**commands**.**length**;++i){  
 **commands**[i].execute();  
 }  
 }  
}

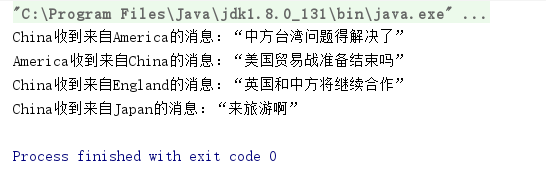
一开始把书上的例子弄混了，硬是没分清楚command用ArrayList表示的方法，后来将命令模式看明白才下手。

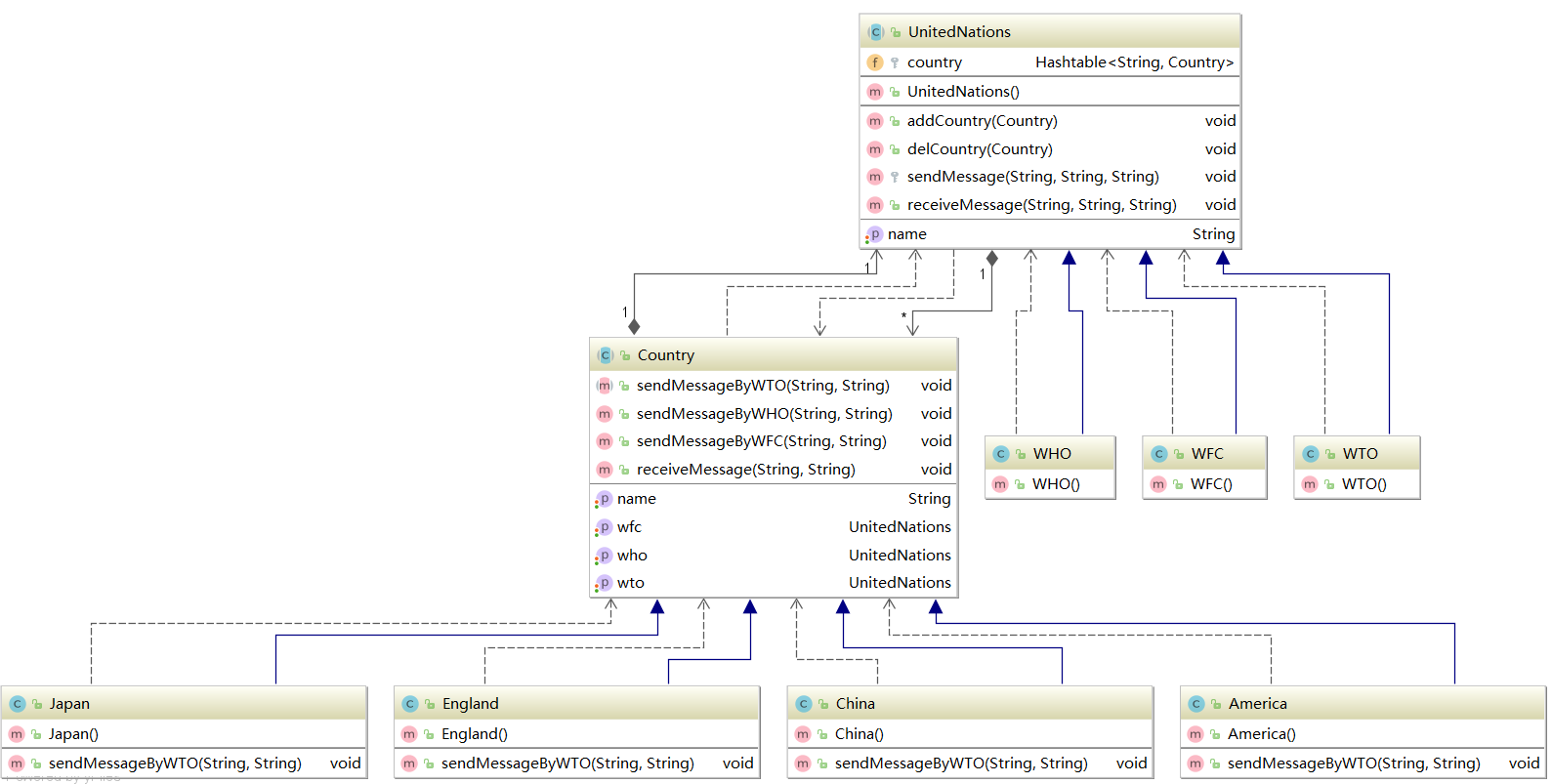
3.遍历学生信息





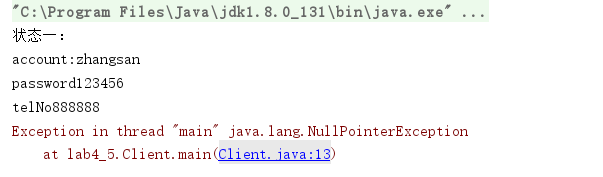
4.联合国的作用（中介者模式）



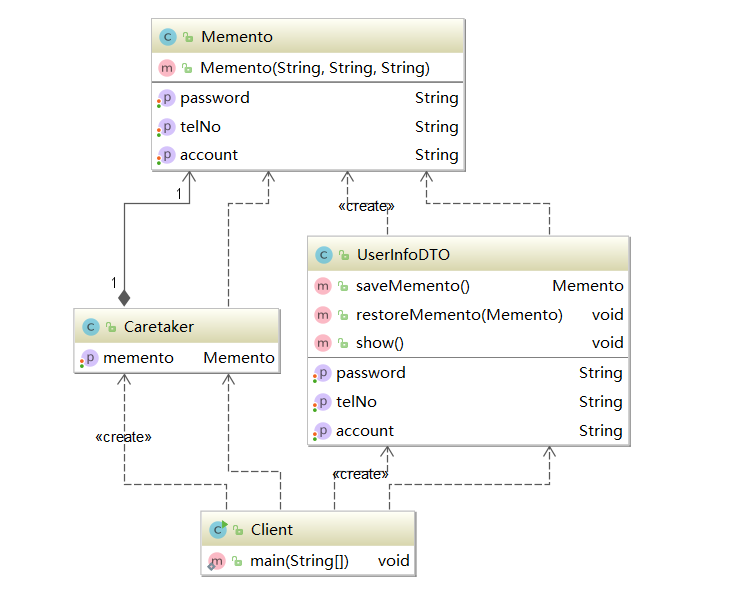


UnitedNations类定义了Country类的集合，用于存储Country类对象并提供加入、移除、发送信息、接收信息方法；WFC、WHO、WTO继承UnitedNations，可调用其发送信息、接受信息等方法，在类中只定义构造方法；Country类定义了UnitedNations类的集合，通过UnitedNations类的对象调用发送信息和接收信息方法；China、America、England、Japan是其子类。主要是通过UnitedNations类的对象发送信息和接收信息。

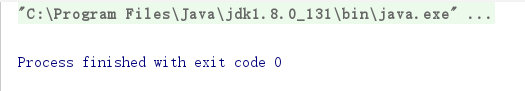
5.多次撤销操作



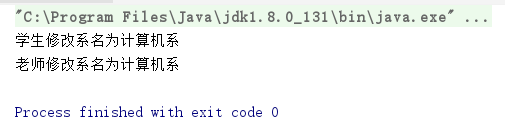
错误原因是初始化负责人时用的数组出现错误，改正后即可



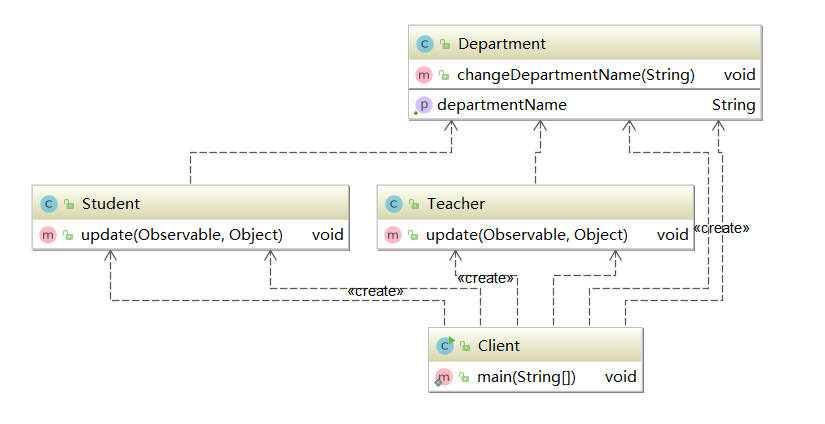
6.教学管理系统



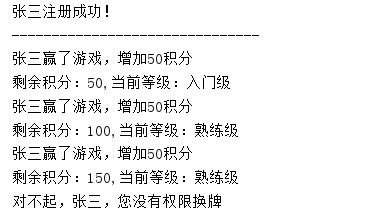
没有将老师和学生类加入观察者类集合

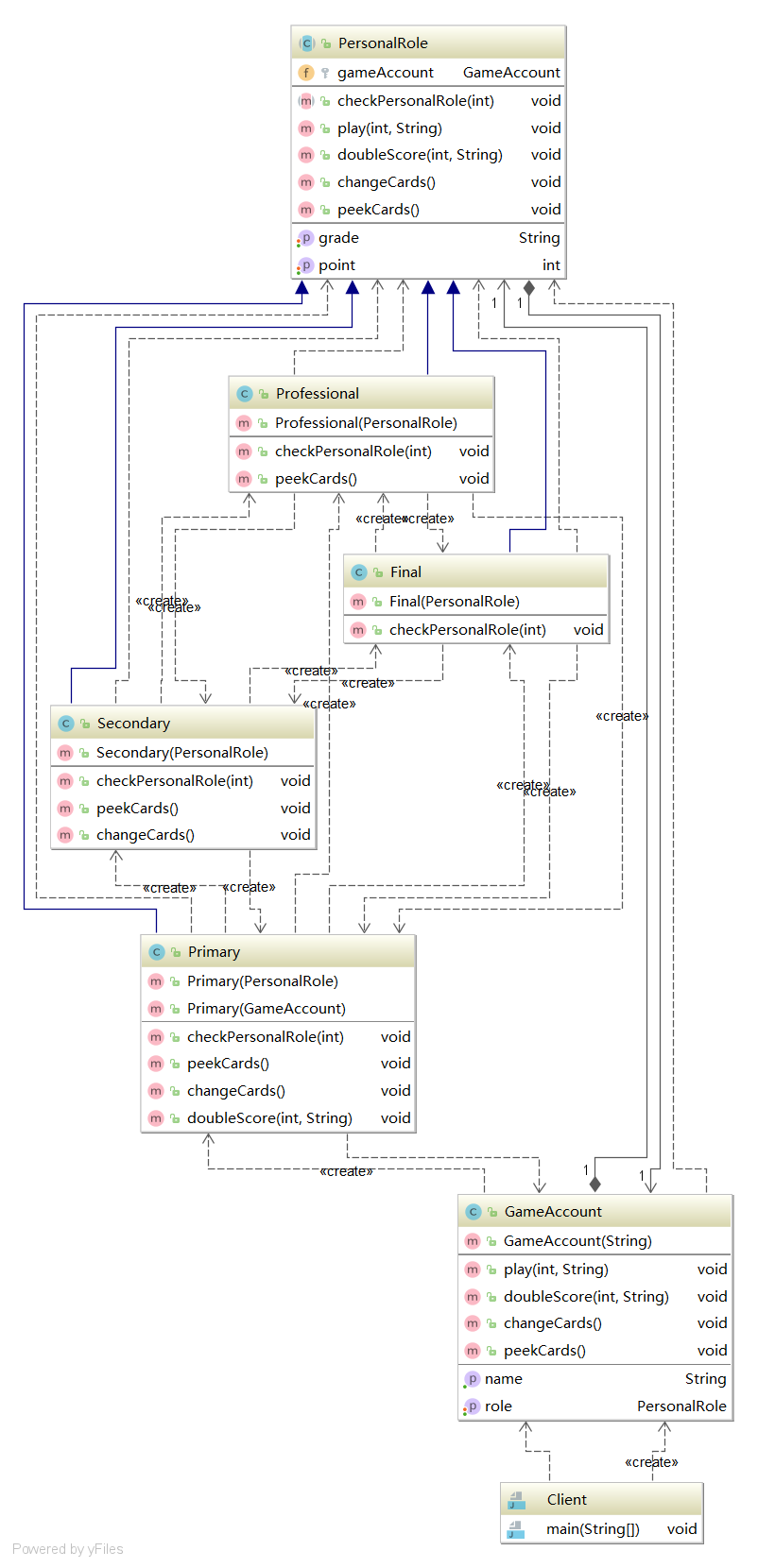


**public static void** main(String args[]){  
 Department department=**new** Department();  
 Teacher teacher=**new** Teacher();  
 Student student=**new** Student();  
 department.addObserver(teacher);  
 department.addObserver(student);  
 department.changeDepartmentName(**"计算机系"**);  
}

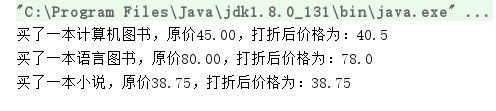


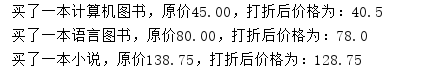
7.纸牌游戏

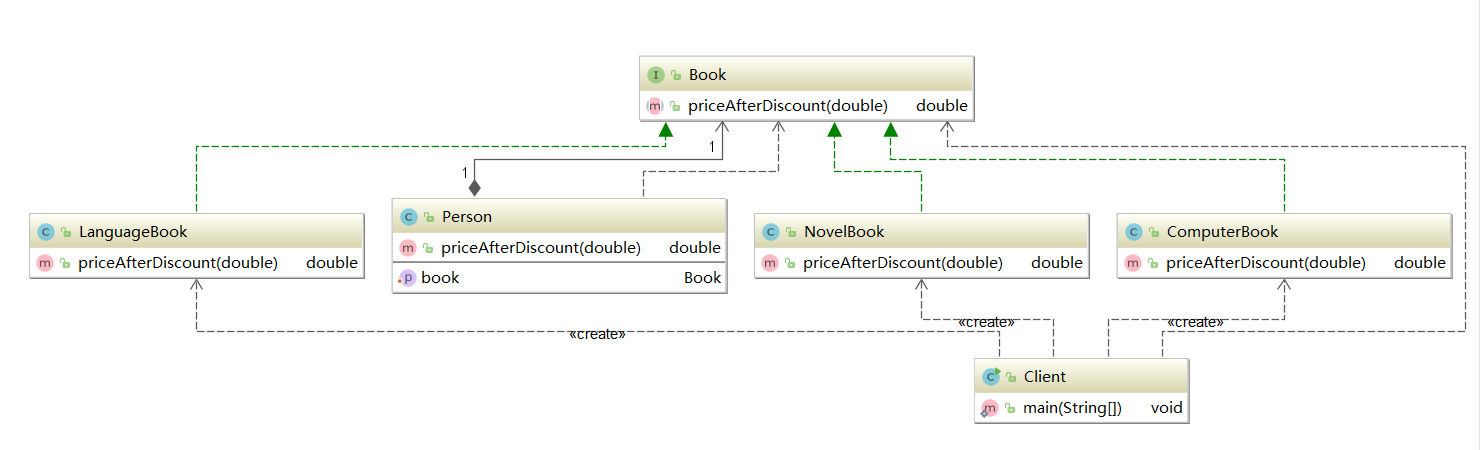




8.网上书店

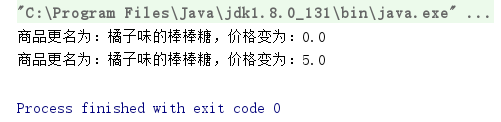


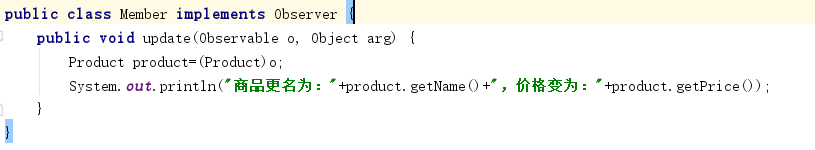




9.网上商店

设置了一个观察者，其结果为：在我需要的正确通知前，价格初始化为了0





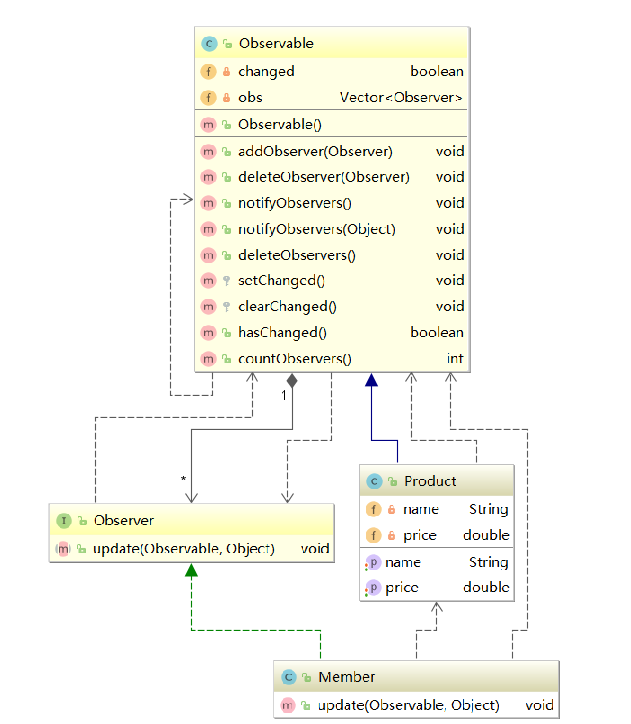
设置两个观察者没有问题



更改代码，设置一个观察者也可以成功

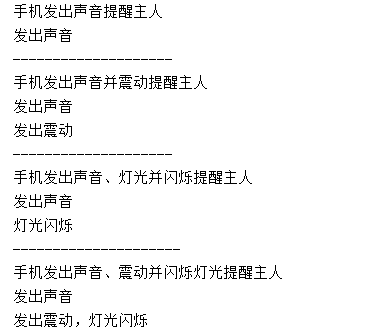


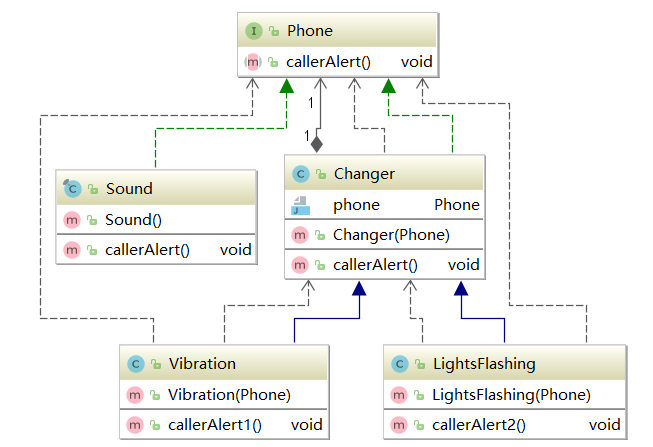
**public class** Member **implements** Observer {  
 **public void** update(Observable o, Object arg) {  
 Product product=(Product)o;  
 **if**(arg **instanceof** String){  
 *//name=(String)arg;* System.***out***.println(**"商品更名为："**+product.getName());  
 }  
 **if**(arg **instanceof** Double){  
 *//price=((Double)arg).doubleValue();* System.***out***.println(**"商品价格变为："**+product.getPrice());  
 }  
 *//System.out.println("商品更名为："+product.getName()+"，价格变为："+product.getPrice());* }  
}

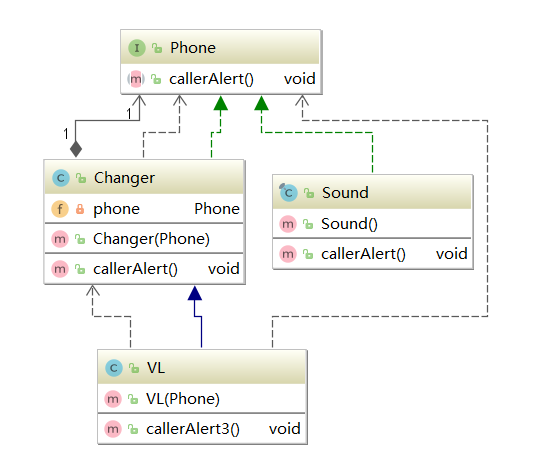


自定义两个类，一个产品类继承自Observable，作为被观察的对象；会员类继承自Observer作为观察者，当产品的名称和价格发生变化时提醒会员，会员即可得知这些消息。

10.手机来电提醒设置







七、思考讨论题

1、比较创建型设计模式、结构型设计模式和行为型设计模式；

答：

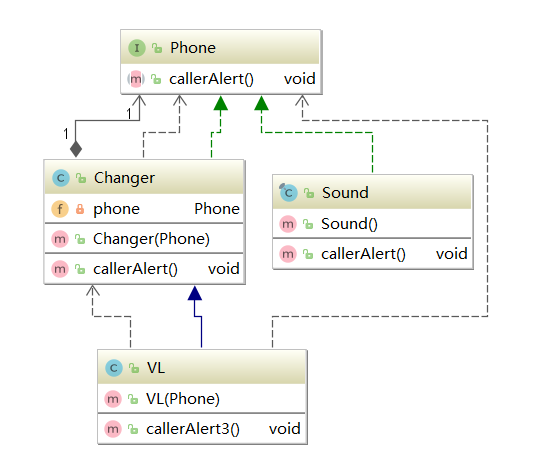
（1）创建型模式对类的实现化过程进行了抽象，能够使软件模块做到与对象的创建和组织无关。

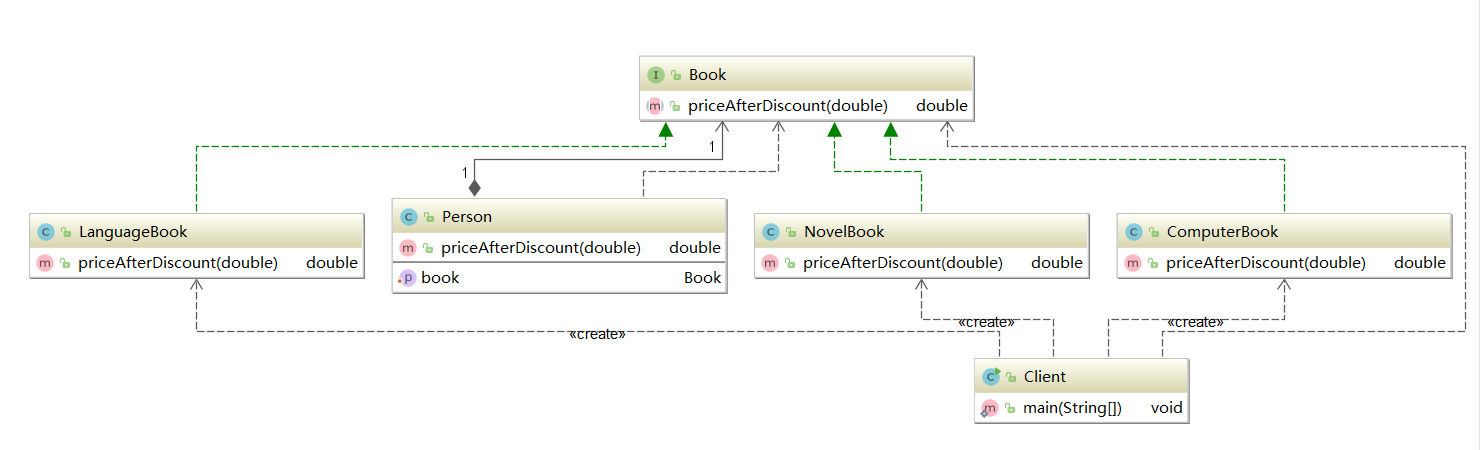
（2）结构型模式描述类和对象之间如何进行有效的组织，以形成良好的软件体系结构，主要的方式是使用继承关系来组织各个类，一个最容易的例子就是如何用多个继承组织两个以上的类，结果产生的类结合了父类所有的属性，结构型模式特别适用于和独立的类库一起工作。

（3）行为型设计模式描述类和对象之间如何交互及如何分配职责，实际上它所牵涉的不仅仅是类或对象的设计模式，还有它们之间的通信模式。

2、对比结构型设计模式与行为型设计模式中部分模式结构图的类似结构图；

答：如装饰模式和策略模式





3、了解符合面向对象的设计原则。

答：单一职责原则，开闭原则，里氏代换原则，依赖倒转原则，接口隔离原则，合成复用原则，迪米特原则。

八、参考资料

九、实现代码

<https://github.com/inocentnine/JAVA-BehavioralPattern>