# Lecture 4 - 实物类: Vending Machine & Juke Box - Factory & Adaptor Pattern

* 实物类OOD题型
  + Vending machine
  + Jukebox
  + CD Player
  + Coffee maker
  + ATM
* 实物类
  + 考虑对于实物的输入输出

Graphical user interface, diagram

Description automatically generated

* 技巧
  + State pattern
  + Decorate pattern
  + Factory pattern

## Vending Machine & ATM MachineDiagram Description automatically generated with medium confidence

## Diagram Description automatically generated

### Graphical user interface, text Description automatically generated with medium confidence

Graphical user interface, text, application, email

Description automatically generated

Table

Description automatically generated

如果用上面的方式定义vendingmachine存在的问题是map里是用静态的数据“比如A1，B4”来对应一个动态的Item，那如果当客户拿到了Item的信息就可以对它进行修改。解决方法是可以加一个静态的class ItemInfo

Diagram

Description automatically generated

上图漏了一个点，Item class下面应该包括itemInfo信息

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Diagram

Description automatically generated

pair就相当于是可以实现几返回item也返回需要的找零

Graphical user interface, text, application, email

Description automatically generated

Diagram

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

Text

Description automatically generated

直接把add method放到stock这个class里，nested class把常用的借口封装起来

Diagram

Description automatically generated with low confidence

小错误：item中的iteminfo info应该是protected，这样的话spirit，coke继承的时候可以自己贴商标

到1:10:00

### State Design Pattern

* States
  + HAS\_SELECTION - NO\_SELECTION
  + COINS\_INSERTED - NO\_COIN
  + SOLD
  + SOLD\_OUT
* State related actions:
  + select item
  + insert coin
  + execute transaction
  + cancel transaction

public interface State {

public oid selectItem(String selection);

public void insertMoney(int value);

public void executeTransaction();

pulic int cancelTransaction();

}

public class NoSelectionState implements State {

VendingMachine vendingMachine;

public NoSectionsState(VendingMachine vendingMachine) {

this.vendingMachine = vendingMachine;

}

//Implement, and change state

@Override

public void selectItem(String selection) {

vendingMachine.setSelectedItem(selection);

vendingMachine.changeToHasSelectionState();

}

}

public class VendingMachine {

private State state;

private NoSelectionState noSelectionState;

privte HasSelectionState hasSelectionState

//Constructor: init all state, and set inital state

public VendingMachine() {

noSelectionState = new NoSelectionState(this);

hasSelectionState = new hasSelectionState(this);

state = noSelectonState;

}

//Expose to other State class for changing

public void changeToHasSelectionState() {

state = hasSelectionState();

}

//API, use different state to implement

public void selectItem(String selection) {

state.selectItem(selection);

}

}

## Coffee Machine

### Decorator Design Pattern

public interface Coffee {

public double getCost();

public String getDescripton();

}

public class Expresso implements Coffee {

public Expresso() {

description = "Expresso";

}

public float getCost() {

return 1.99;

}

public String getDecription() {

return description;

}

}

public abstract class CoffeeDecorator implements Coffee {

protected final Coffe decoratedCoffee;

public CoffeeDecorator(Coffee c) {

this.decoratedCoffee = c;

}

public double getCost() {

return decoratedCoffee.getCost();

}

public double getDescription() {

return decoratedCoffee.getDescription();

}

}

public class WithMocha extends CoffeeDecorator {

public WithMocha(Coffee coffee) {

super(c);

}

public String getDescription() {

return super.getDescription() + ", Mocha";

}

public float getCost() {

return super.getCost() + 0.5;

}

}

public void test() {

Coffee c = new Expresso();

c = new WithMocha(c);

c = new WithMocha(c);

// Expresso, Mocha, Mocha $2.99

System.out.println(c.getDescription() + "$ " + c.getCost());

}

## Kindle

### Factory Design Pattern

* Simple Factory: 将if-else放入一个单独的factory里面
* Factory method
* Abstract Factory