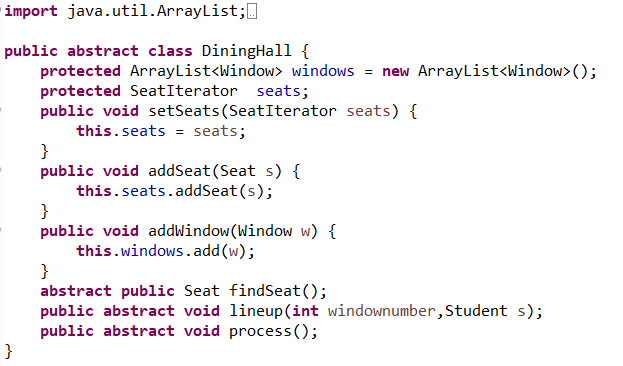
设计思路：1 对于服务员打蔡和学生点餐，同时对菜品采用不同的操作所以使用访问者模式

2 对于食堂学生在食堂可以找座位，选窗口排队等，把这些功能整合到一个食堂类中，学生调用食堂类提供的接口采用外观模式

3 学生点餐需要排队，排队可以看成学生的集合，对集合进行遍历采用迭代器模式

4 食堂的窗口有的比较高级，有的比较差，考虑可以对窗口进行升级因此采用了装饰模式，同理食堂也可以进行升级，对食堂也采用装饰模式。

源码架构：



**abstract** **public** **class** DiningHallDecorator **extends** DiningHall {

**protected** DiningHall dininghall;

DiningHallDecorator(DiningHall dininghall){

**this**.dininghall = dininghall;

}

}

**public** **class** TAODiningHall **extends** DiningHall {

@Override

**public** Seat findSeat() {

// **TODO** 自动生成的方法存根

Seat s = **this**.seats.getEmptySeat();

**if**(s == **null** ) {

System.***out***.println("没座位了");

**return** **null**;

}

**return** s;

}

@Override

**public** **void** lineup(**int** windownumber,Student s) {

// **TODO** 自动生成的方法存根

**if**(windownumber>= **this**.windows.size()) {

System.***out***.println("没有这个窗口");

**return**;

}

**this**.windows.get(windownumber).addStudent(s);

}

@Override

**public** **void** process() {

// **TODO** 自动生成的方法存根

// 调用每个窗口的process函数

**int** i =0;

**for**(Window w:**this**.windows) {

System.***out***.println("窗口"+i+"工作中");

w.process();

i++;

}

}

}

**public** **class** MEIDiningHall **extends** DiningHallDecorator {

**public** MEIDiningHall(DiningHall dininghall) {

**super**(dininghall);

// **TODO** 自动生成的构造函数存根

}

@Override

**public** Seat findSeat() {

// **TODO** 自动生成的方法存根

System.***out***.println("梅园的椅子更加舒适");

**return** **this**.dininghall.findSeat();

}

@Override

**public** **void** lineup(**int** windownumber, Student s) {

// **TODO** 自动生成的方法存根

**this**.dininghall.lineup(windownumber, s);

}

@Override

**public** **void** process() {

// **TODO** 自动生成的方法存根

System.***out***.println("梅园的饭菜更加可口，服务更好，打的菜更多");

**this**.dininghall.process();

}

}

**public** **class** Seat {

**protected** **boolean** state =**false**;

**public** **boolean** getState() {

**return** state;

}

**public** **void** setState(Boolean state) {

**this**.state = state;

}

}

**import** java.util.ArrayList;

**public** **class** SeatList {

**protected** ArrayList<Seat> seats = **new** ArrayList<Seat>();

**public** SeatList() {

**int** i=0;

**for**(i=0;i<3;i++) {

seats.add(**new** Seat()); //默认有3把椅子

}

}

**public** SeatIterator createSeatIterator() {

**return** **new** SeatIterator();

}

**class** SeatIterator{ //迭代器模式

**public** Seat getEmptySeat() {

Seat currentSeat = **null**;

**for**(Seat s :seats) {

**if**(s.getState()) { //如果是空的

currentSeat =s;

}

}

**return** currentSeat;

}

**public** **void** addSeat(Seat s) {

seats.add(s);

}

}

}

**public** **abstract** **class** Window {

**protected** Visitor server;

**public** **void** setServer(Visitor server) {

**this**.server = server;

}

**private** StudentQueue sq = **new** StudentQueue();

**protected** QueueIterator queue = **this**.sq.createQueueIterator();

**public** **void** setQueue(QueueIterator queue) {

**this**.queue = queue;

}

**protected** HashMap<String,Food> foodList = **new** HashMap<String,Food>() ; //采用哈希表将Food对象和 Food名称（字符串）相匹配，为了符合模拟场景，这个用在process函数和student类的takeorder函数中

**public** **void** addFood(Food f) {

**this**.foodList.put(f.getName(), f);

}

**public** **void** addStudent(Student s) {

**this**.queue.addStudent(s);

}

**public** **int** getNum() {

**return** **this**.queue.getNum();

}

**abstract** **public** **void** process(); //按顺序排队点餐函数，由子类实现

}

**public** **abstract** **class** WindowDecorator **extends** Window {

**protected** Window window;

**public** WindowDecorator(Window w){

**this**.window = w;

}

}

**public** **class** BaseWindow **extends** Window {

@Override

**public** **void** process() {

// **TODO** 自动生成的方法存根

**do**{

Student currentStudent = **this**.queue.currentStudent();

**if** (currentStudent==**null**) {

**return**;

}

String foodName = currentStudent.takeOrder(); //设置hash表和 student里的 wishFood就是为了这里

Food f = **this**.foodList.get(foodName);

**if**(f==**null**) {

System.***out***.println("没有学生想要吃的菜");

}**else** {

f.accept(currentStudent); //访问者模式

f.accept(**this**.server);

}

System.***out***.println("请下一位学生取餐");

}**while**(**this**.queue.next());

}

}

**package** DiningHallSimulate;

**public** **class** AdvancedWindow **extends** WindowDecorator {

**public** AdvancedWindow(Window w) {

**super**(w);

// **TODO** 自动生成的构造函数存根

}

@Override

**public** **void** process() {

// **TODO** 自动生成的方法存根

System.***out***.println("窗口前有显示所有菜品的LED屏幕");

**this**.window.process();

}

}

**public** **abstract** **class** Food {

**protected** Double price;

**public** Double getPrice() {

**return** price;

}

**public** String getName() {

**return** name;

}

**protected** String name;

**public** Food(Double price,String name) {

**this**.price = price;

**this**.name = name;

}

**abstract** **public** **void** accept(Visitor v);

}

**public** **class** HangMenJi **extends** Food {

**public** HangMenJi(Double price, String name) {

**super**(price, name);

// **TODO** 自动生成的构造函数存根

}

@Override

**public** **void** accept(Visitor v) {

// **TODO** 自动生成的方法存根

v.visit(**this**);

}

}

**public** **class** DiaoZhaBing **extends** Food {

**public** DiaoZhaBing(Double price, String name) {

**super**(price, name);

// **TODO** 自动生成的构造函数存根

}

@Override

**public** **void** accept(Visitor v) {

// **TODO** 自动生成的方法存根

v.visit(**this**);

}

}

**public** **class** NiuRouMian **extends** Food {

**public** NiuRouMian(Double price, String name) {

**super**(price, name);

// **TODO** 自动生成的构造函数存根

}

@Override

**public** **void** accept(Visitor v) {

// **TODO** 自动生成的方法存根

v.visit(**this**);

}

}

**public** **abstract** **interface** Visitor {

**public** **abstract** **void** visit(HangMenJi f);

**public** **abstract** **void** visit(DiaoZhaBing f);

**public** **abstract** **void** visit(NiuRouMian f);

}

**public** **class** Server **implements** Visitor {

//食堂工作人员类

@Override

**public** **void** visit(HangMenJi f) {

// **TODO** 自动生成的方法存根

System.***out***.println("食堂工作人员打了"+f.getName());

}

@Override

**public** **void** visit(DiaoZhaBing f) {

// **TODO** 自动生成的方法存根

System.***out***.println("食堂工作人员打了"+f.getName());

}

@Override

**public** **void** visit(NiuRouMian f) {

// **TODO** 自动生成的方法存根

System.***out***.println("食堂工作人员打了"+f.getName());

}

}

**public** **class** Student **implements** Visitor {

**private** String name;

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** PayMethod getPayMethod() {

**return** payMethod;

}

**public** **void** setPayMethod(PayMethod payMethod) {

**this**.payMethod = payMethod;

}

**private** PayMethod payMethod;

**public** String wishFood=**null**;

**public** Student(String name) {

**this**.name=name;

}

@Override

**public** **void** visit(HangMenJi f) {

// **TODO** 自动生成的方法存根

System.***out***.println("学生点了"+f.getName());

**this**.pay(f.price);

}

@Override

**public** **void** visit(DiaoZhaBing f) {

// **TODO** 自动生成的方法存根

System.***out***.println("学生点了"+f.getName());

**this**.pay(f.price);

}

@Override

**public** **void** visit(NiuRouMian f) {

// **TODO** 自动生成的方法存根

System.***out***.println("学生点了"+f.getName());

**this**.pay(f.price);

}

//点餐函数

**public** String takeOrder() {

//这个函数从学生出获得想要吃的食物的名称...这里假设从类里的一个变量获得，这个变量可以随便指定

**return** **this**.wishFood;

}

//付钱

**public** **void** pay(Double d) {

**this**.payMethod.pay(d);

}

}

**public** **class** StudentQueue {

**private** ArrayList<Student> students=**new** ArrayList<Student>();

**public** QueueIterator createQueueIterator() {

**return** **new** QueueIterator();

}

**class** QueueIterator{

**public** **boolean** next() {

**if**(!students.isEmpty()) {

students.remove(0);

**return** **true**;

}**else** {

System.***out***.println("没有学生排队了");

**return** **false**;

}

}

**public** Student currentStudent() {

**if**(! students.isEmpty()) {

**return** students.get(0);

}**else** {

System.***out***.println("没有学生排队了");

**return** **null**;

}

}

**public** **void** addStudent(Student s) {

students.add(s);

}

**public** **int** getNum() {

**return** students.size();

}

}

}

**public** **abstract** **interface** PayMethod {

**abstract** **public** **void** pay(Double d);

}

**public** **class** Cash **implements** PayMethod {

@Override

**public** **void** pay(Double d) {

// **TODO** 自动生成的方法存根

System.***out***.println("使用现金支付了 "+d+" 元");

}

}

**public** **class** Card **implements** PayMethod {

@Override

**public** **void** pay(Double d) {

// **TODO** 自动生成的方法存根

System.***out***.println("使用校园卡支付了 "+d+" 元");

}

}

**public** **class** test {

**public** **static** **void** main(String[] args) {

// **TODO** 自动生成的方法存根

//测试类

//菜品 有三种 黄焖鸡 掉渣饼和牛肉面

HangMenJi food1 = **new** HangMenJi(13.0, "黄焖鸡");

DiaoZhaBing food2 = **new** DiaoZhaBing(11.5, "掉渣饼");

NiuRouMian food3 = **new** NiuRouMian(10.0,"牛肉面");

//食堂

//桃园食堂

TAODiningHall taoyuan = **new** TAODiningHall();

//假设有2个窗口

BaseWindow w1 = **new** BaseWindow();

BaseWindow tempW = **new** BaseWindow();

BaseWindow w2 = **new** BaseWindow(); //窗口2 比窗口1更加高级 装饰模式

//一个窗口有一个厨师

Server s1 = **new** Server();

Server s2 = **new** Server();

//设置窗口服务员和菜品

w1.setServer(s1);

w2.setServer(s2);

w1.addFood(food1); //窗口1 卖黄焖鸡

w2.addFood(food2); //窗口 2、3买掉渣饼和牛肉面

w2.addFood(food3);

//设置椅子

SeatList seats1 = **new** SeatList();

SeatList seats2 = **new** SeatList();

seats2.createSeatIterator().addSeat(**new** Seat()); //添加椅子

//组装桃园食堂

taoyuan.setSeats(seats1.createSeatIterator());

//taoyuan.addWindow(w1);

taoyuan.addWindow(w2);

// 梅园食堂配置和桃园食堂基本相似只是增加了一些新的功能

MEIDiningHall meiyuan = **new** MEIDiningHall(taoyuan);

//学生

//付款方式

Cash cash = **new** Cash();

Card card = **new** Card();

Student student1 = **new** Student("小王");

student1.setPayMethod(card);

student1.wishFood="黄焖鸡";

Student student2 = **new** Student("小明");

student2.setPayMethod(card);

student2.wishFood="牛肉面";

Student student3 = **new** Student("小刚");

student3.setPayMethod(cash);

student3.wishFood="掉渣饼";

//学生去食堂吃饭了

taoyuan.lineup(0, student1);

taoyuan.lineup(0, student2);

taoyuan.lineup(0, student3);

//开始点餐 一个个顺序进行，没有用多线程

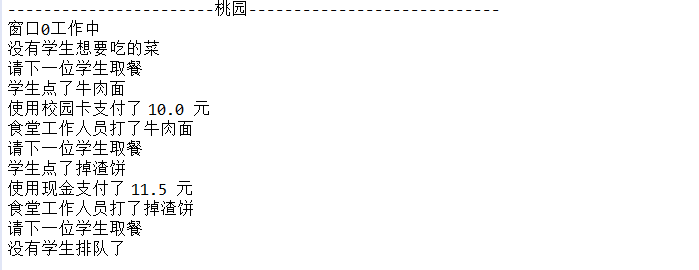
System.***out***.println("-----------------------桃园----------------------------");

taoyuan.process();

}

}

部分结果



UML图

