HW8

Requirements:

- Create a Java project named yourStudentId_OOP_HW8
- Read instructions and create classes needed. You are supposed to add 1 interface and 6 classes (*Payment*, *Cash, CreditCard*, *Food, Customer, VIP,* and *DMan,*) to the project.
- All instance variables are private. Please use public methods to access private instance variables.

Description:

In a daily life, we often use the Food delivery app like foodpanda or Ubereat to order the food. Supposedly you are a system developer, you need to develop the system for delivers to manage their orders and check the payment of the customers. Hence, you need to use the interface concept to write in code in order to make the payment method more flexible.

The diagram below depicts a simple food delivery system. There are six classes that need to be implemented, where two of them, *Cash* and *CreditCard*, implement the interface *Payment*. *Dman* represents food delivery man. *Customer* is inherited by *VIP*.

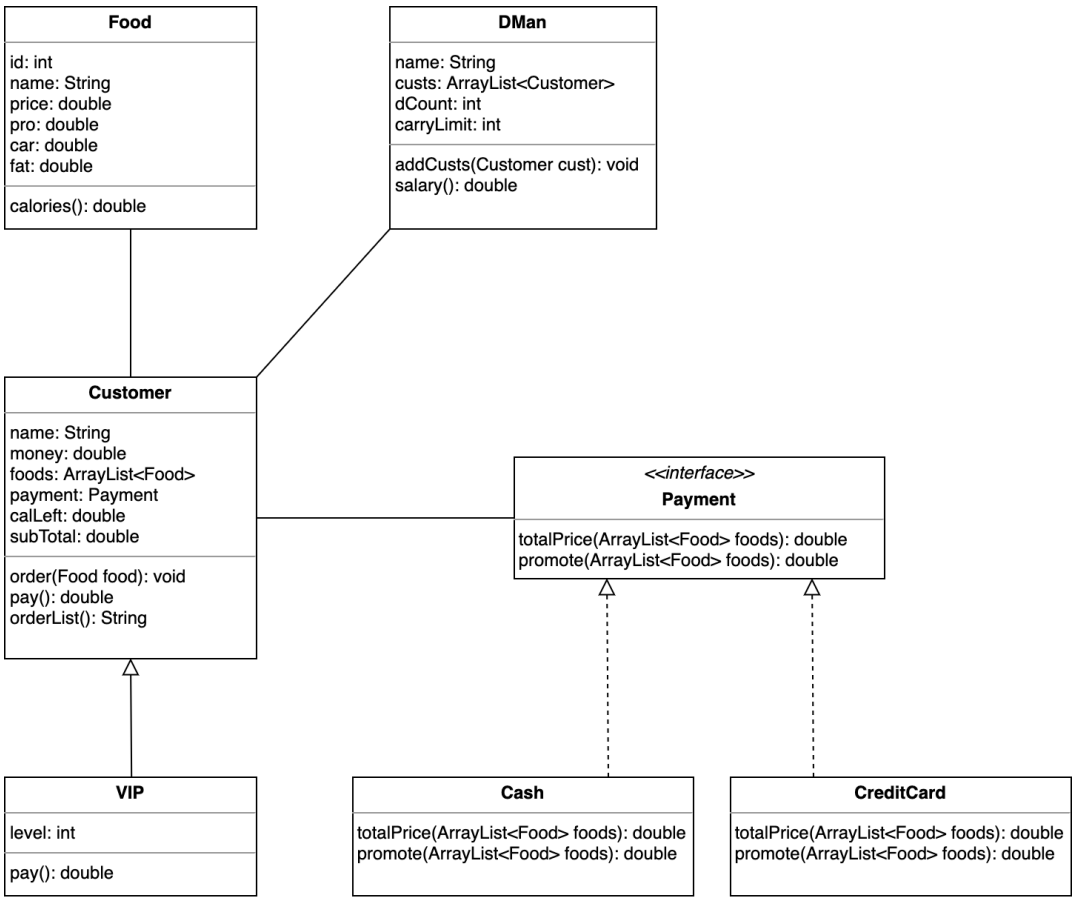

Figure 1. The UML diagram of the question

1. Create **Payment** interface

<<Interface>> **Payment**	
Modifier and type	**Method (or Variable) and description**
Abstract methods	
double	**totalPrice(ArrayList<Food> foods)** Abstract method.
double	**promote(ArrayList<Food> foods)** Abstract method.

2. Create **Cash** class implements **Payment**

Cash	
Modifier and type	**Method (or Variable) and description**
Instance methods	
double	**totalPrice(ArrayList<Food> foods)** Customer who chooses to pay in cash has to pay an extra fee (amount of food*5) USD dollar. Return total price this customer should pay (Don't forget to add extra fee).
double	**promote(ArrayList<Food> foods)** "lobster" (id "777") is on sale. If customer orders three "lobster", this order can be discounted by 50 dollars. Return 0 if condition not fulfilled, else return 50 dollars.

3. Create **CreditCard** class implements **Payment**

CreditCard	
Modifier and type	**Method (or Variable) and description**
Instance methods	
double	**totalPrice(ArrayList<Food> foods)** Customer who chooses to pay with credit card will be charged a two-percent-fee (2%). Return total price after being charged.
double	**promote(ArrayList<Food> foods)** Same as the method in **Cash** (You don't have to write it again).

4. Create **Food** class

Food	
Modifier and type	**Method (or Variable) and description**
Instance variable	
String	**name** The name of food.
int	**id** The id of food.
double	**price** The price of food.
double	**pro** The protein content of food.
double	**car** The carbohydrate content of food.
double	**fat** The fat content of food.
Constructor	
Food (int id,String name,double price,double pro,double car,double fat) **Constructs an Food object with given id, name, price, pro, car and fat.**	
Instance methods	
double	**calories**() Calculate the calories of food and then return the result. *Each gram of protein, carbohydrate and fat will generate 4, 4, 9 calories respectively.

5. Create **Customer** class

Customer	
Modifier and type	**Method (or Variable) and description**
Instance variable	
String	**name** Name of the customer.
double	**money** The money this customer got.
ArrayList<Food>	**foods** An ArrayList contains foods that this customer ordered before.
Payment	**payment** An object declared from Payment.

double	**calLeft** The maximum cal that customer can accept. Or in other words, calLeft represents "how many cal that customer still could take in". Every time when customer orders food, we should re-calculate "how many cal that customer still could take in".
double	**subtotal** Total price of current order list.
Constructor	
Customer (String name, double money, double calLeft, Payment payment) **Constructs an Customer object with given name, money, calLeft, payment, set subtotal to 0, and initialize foods.**	
Instance methods	
void	**order(Food food)** step1.Check if customer has enough money. If has, go to step2. Otherwise, print "XXX don't have enough money!" (XXX=customer's name). step2.Check if the food is too high in calorie. If it isn't too high, go to step3. Otherwise, print ""Too fat for XXX" (XXX=customer's name). step3.Add the food in to **foods** (ArrayList), and re-calculate **calLeft.**
String	**orderList()** Return a String description of order list (please refer to the output in Tester). Please format you output (reserved space is 10, and round the price to two decimal places).
double	**pay()** Return the final charge (total price minus promotion discount). But still, it is necessary to check if this customer has enough money to pay this bill. If he/she does, return final charge and calculate his/her remaining money. If he/she does not, print out "XXX doesn't have enough money!" (XXX=customer's name) and also return 0;

6. Create **VIP** class **extends Customer** class

VIP	
Modifier and type	**Method (or Variable) and description**
int	**level** The VIP level of this customer. VIP will get a discount (level*10).
Instance methods	
double	**pay()** Same as pay() in Customer class except that the price should deduct VIP discount (level*10).

7. Create **DMan** class

DMan	
Modifier and type	**Method (or Variable) and description**
Instance variable	
String	**name** The name of delivery man.
int	**dCount** The counter to calculates how many times this DMan delivers foods.
int	**carryLimit** Maximum amount of foods this DMan could affords.
ArrayList<Customer>	**custs** An ArrayList contains customers.
Constructor	
DMan(String name, int carryLimit) **Constructs a DMan object with given name, carryLimit, set dCount to 0, and initialize custs.**	
Instance methods	
double	**salary()** Calculate the salary of this DMan (10 dollars per customer).
void	**addCusts(Customer cust)** Receive the order from customers, add customer into custs. Note that if receiving this order will cause the **carryLimit** to be exceeded, print" Too many to carry!" in the console, and don't forget to reject this order.

Tester code	Tester output

`Payment p1 = new Cash();`
`Payment p2 = new CreditCard();`
`DMan d1 = new DMan("Toolman", 12);`
`Food f1 = new Food(111, "Apple", 50, 5, 5, 5);`
`Food f2 = new Food(555, "Steak", 500, 200, 200, 200);`
`Food f3 = new Food(777, "lobster", 400, 100, 100,100);`
`Customer c1 = new Customer("Peter", 100, 171, p2);`
`VIP c2 = new VIP("Dick", 9000, 8670, p1, 5);`
`c1.order(f1);`
`c1.order(f1);`
`c1.order(f1);//Peter don't have enough money!`
`c2.order(f1);`
`c2.order(f1);`
`c2.order(f2);`
`c2.order(f2);`
`c2.order(f3);`
`c2.order(f3);`
`d1.addCusts(c2);`
`d1.addCusts(c1);`
`System.out.print(c1.orderList());`
`System.out.println("--------------------------");`
	`Peter don't have enough money!`
	`Too fat for Dick`
	`Apple 50.00`
	`Apple 50.00`
	`Total 100.00`
	`--------------------------`
	`Apple 50.00`
	`Apple 50.00`
	`Steak 500.00`
	`Steak 500.00`
	`lobster 400.00`
	`Total 1500.00`
	`Peter doesn't have enough money!`
	`C1 pays 0.0 USD dollars`
	`c1 left 100.0 USD dollars`
	`C2 pays 1475.0 USD dollars`
	`c2 left 7525.0 USD dollars`

```
System.out.print(c2.orderList());
//c1 cannot afford the payment because of the extra fee
System.out.println("C1 pays "+c1.pay()+" USD dollars");
System.out.println("c1    left    "+c1.getMoney()+"    USD
dollars");
System.out.println("C2 pays "+c2.pay()+" USD dollars");
System.out.println("c2    left    "+c2.getMoney()+"    USD
dollars");
System.out.println("d1  got  salary  "+d1.salary()+"  USD
dollar");
```

Submission: Submit your project as **".zip file"** via Moodle. No other submissions will be graded.

Reminder: Please zip the whole project.

Deadline: 2021/04/05 (Mon56) or 2021/04/06 (for Tue23)