Market store assignment explanation

Assignment

A market store offers to their clients three different types of discount cards: bronze, silver and gold. Each card stores information about its owner, the turnover for the previous month and the initial discount rate. With each card you can calculate the discount of the current purchase. In general this is done using the formula: value\_of\_purchase \* discount\_rate.

The bronze card comes with no discount rate if the turnover for the previous month is below $100. If it is between $100 and $300, the discount rate is 1%. And if it is above $300, the rate is 2.5%.

The silver card comes with an initial discount rate of 2%. In case, the turnover is over $300, the rate is 3.5%.

The gold card comes with an initial discount rate of 2%. The discount rate grows 1% for each $100 from the turnover, capping at 10%.

My task is:

* to write an object-oriented model of this market store system;
* to create a class called PayDesk containing static method(s) that output the discount rate, the discount and the total purchase value in my application entry point:
* create instances with sample data as shown in Example outputs section;
* invoke method(s) of the PayDesk class to print the required information.

Example outputs

Bronze:

a. Mock data: turnover $0, purchase value $150;

b. Output:

Purchase value: $150.00

Discount rate: 0.0%

Discount: $0.00

Total: $150.00

Silver:

a. Mock data: turnover $600, purchase value $850;

b. Output:

Purchase value: $850.00

Discount rate: 3.5%

Discount: $29.75

Total: $820.25

Gold:

a. Mock data: turnover $1500, purchase value $1300;

b. Output:

Purchase value: $1300.00

Discount rate: 10.0%

Discount: $130.00

Total: $1170.00

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

To begin with, there are given three different types of discount cards, but they have almost the same characteristics. This means that I have to create one parent class for a discount card which has the following attributes = > owner, turnover and discount rate. First of all, for the owner I created class Client so I can store information for his/her name, phone, email and address in separate class, then in need it will be easier to extract the data for the owner.

**public** **class** Client {

String name;

String email;

String phone;

String address;

Client(String name, String email, String phone, String address){

**this**.name = name;

**this**.email = email;

**this**.phone = phone;

**this**.address = address;

}

Client(String name, String phone){

**this**.name = name;

**this**.phone = phone;

}

}

It is good to use type String for the email and phone attribute, because phone can start with “+” or can also contain other symbols, email contains “@ . ”.What is more, I created two constructors for initializing the attributes. The first one is also enough, but I wanted to have another one in order not to pass all the parameters to the constructor when I call it in the main class.

Next, I created abstract class DiscountCard.

**public** **abstract** **class** DiscountCard {

Client owner;

**float** turnover;

**float** discountRate;

DiscountCard(Client owner, **float** turnover, **float** discountRate){

**this**.owner = owner;

**this**.turnover = turnover;

**this**.discountRate = discountRate;

}

DiscountCard(Client owner, **float** turnover){

**this**.owner = owner;

**this**.turnover = turnover;

}

**abstract** **float** getDiscountRate();

**abstract** **float** calculateDiscount(**float** purchaseValue);

}

It is abstract because I have methods that I am going to use in other classes for the different types of cards where the rate and calculation will be different for the different types. For the discount rate variable I use float instead of double, because it is less room and less precision. I have declared two constructors again, but I am going to use the second one. Now I have the superclass and I can create the subclasses Bronze, Silver and Gold cards.

**public** **class** BronzeCard **extends** DiscountCard{

BronzeCard(Client owner, **float** turnover) {

**super**(owner, turnover);

}

**float** getDiscountRate() {

**if**(turnover<100) **return** 0f;

**else** **if**(turnover>=100 && turnover <=300) **return** 0.01f;

**else** **return** 0.025f;

}

**float** calculateDiscount(**float** purchaseValue) {

**return** (purchaseValue \* getDiscountRate());

}

}

The BronzeCard class inherits the attributes and methods from the DiscountCard class. In the getDiscountRate() method I have an algorithm that return 0 if the turnover is less than $100, when the turnover is between $100 and $300, the rate is 1% = 1/100 = 0.01 (there is f because of the type float) and finally it checks if the turnover is more than $300 and if it is right, then the result is 2.5= 2.5/100 = 0.025. The discount formula is the same for all the cards, but the discount rate is different, this is why I made it abstract and override in all the classes.

**public** **class** SilverCard **extends** DiscountCard{

SilverCard(Client owner, **float** turnover) {

**super**(owner, turnover);

}

**float** getDiscountRate() {

**if**(turnover>300) **return** 0.035f;

**else** **return** 0.02f;

}

**float** calculateDiscount(**float** purchaseValue) {

**return** (purchaseValue \* getDiscountRate());

}

}

The SilverCard class also inherits the attributes and methods from the DiscountCard class. In the getDiscountRate() method I have an algorithm that return 3.5% = 3.5/100= 0.035 if the turnover is more than $300, if it is not true, it returns 2% = 2/100 = 0.02 .

**public** **class** GoldCard **extends** DiscountCard{

GoldCard(Client owner, **float** turnover) {

**super**(owner, turnover);

}

**int** value=(((**int**)turnover/100)+2);

**float** getDiscountRate() {

**if**(turnover<100) **return** 0.02f;

**else** **if**(turnover>=800) **return** 0.1f;

**else** **return** (**float**)value/100;

}

**float** calculateDiscount(**float** purchaseValue) {

**return** (purchaseValue \* getDiscountRate());

}

}

The GoldCard is a subclass of DiscountCard class. In the getDiscountRate() method here I check if the turnover is less than 100, in order to return the initial discount rate. The discount rate grows with 1% for each $100 from the turnover. If the first statement is not true, I check if the turnover is more than $800, because 10% is over $800, if it is not true I define an int variable value and calculate it, as I took the first digit of the turnover and sum it with 2 to get the percent, then in the return statement I divide it by 100 and cast it into float.

After all, it is time to create the main class PayDesk.

**import** java.text.DecimalFormat;

**import** java.text.NumberFormat;

**public** **class** PayDesk {

**public** **static** **void** printInfo( DiscountCard discountCard, **float** purchaseValue) {

NumberFormat formatter = **new** DecimalFormat("#0.00");

**float** discountRate = discountCard.getDiscountRate() \* 100;

**float** discount = discountCard.calculateDiscount(purchaseValue);

**switch**(discountCard.getClass().getName().toString()) {

**case** "BronzeCard": {System.***out***.print("Bronze: "); **break**;}

**case** "SilverCard": {System.***out***.print("Silver: "); **break**;}

**case** "GoldCard": {System.***out***.print("Gold: "); **break**;}

**default**: {System.***out***.print("Discount card: "); **break**;}

}

System.***out***.println("\na.Mock data: turnover $"+discountCard.turnover+", purchase value $"+purchaseValue);

System.***out***.println("b.Output:\n\n\* Purchase value: $"+ formatter.format(purchaseValue)+

"\n\* Discount rate: "+ discountRate +

"%\n\* Discount: $"+formatter.format(discount)+

"\n\* Total: $"+formatter.format(purchaseValue-discount)+"\n");

}

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

Client firstOwner = **new** Client("Sanie Mehmet","+359899616894");

Client secondOwner = **new** Client("Maria Daneva","0886524789");

Client thirdOwner = **new** Client("Ivan Ivanov","0872341569");

BronzeCard bronzeCard = **new** BronzeCard(firstOwner,0);

SilverCard silverCard = **new** SilverCard(secondOwner,600);

GoldCard goldCard = **new** GoldCard(thirdOwner,1500);

*printInfo*(bronzeCard,150);

*printInfo*(silverCard,850);

*printInfo*(goldCard,1300);

}

}

First of all, I created a static method printInfo () with parameters of type DiscountCard discountCard (in order to pass cards from all types) and float purchaseValue.

NumberFormat formatter = **new** DecimalFormat("#0.00");

In order to use NumberFormat and DecimalFormat I imported some packages. The formatter is necessary, because I want to output the values with 2 digits after the point. Then I declare variables for discount rate and discount and using the methods of the class DiscountCard appropriate values to them. I multiply the discount rate by 100 in order to get the value in percentage.

My classes are called BronzeCard, SilverCard, GoldCard and when I want to print their name by using getClass ().getName () it will print BronzeCard, etc. I want the output to be only one word like bronze, silver or gold, and this is why I have a switch clause where I check the name of the classes and output the appropriate word. The next ones are the output statements for all the information (discount rate, discount, total, etc.).

In the main, I created objects of type Client, three card objects with sample data as shown in the assignment. Finally, I invoke the method I created to print the cards information.

To start the application, click Run.