* Low Level Design

**Introduction:**

On a retail website, the following discounts apply:

1.If the user is an employee of the store, he gets a 30% discount

2.If the user is an affiliate of the store, he gets a 10% discount

3.If the user has been a customer for over 2 years, he gets a 5% discount.

4.For every $100 on the bill, there would be a $ 5 discount

(e.g. for $ 990, you get $ 45 as discount).

5.The percentage based discounts do not apply on groceries.

6.A user can get only one of the percentage based discounts on a bill.

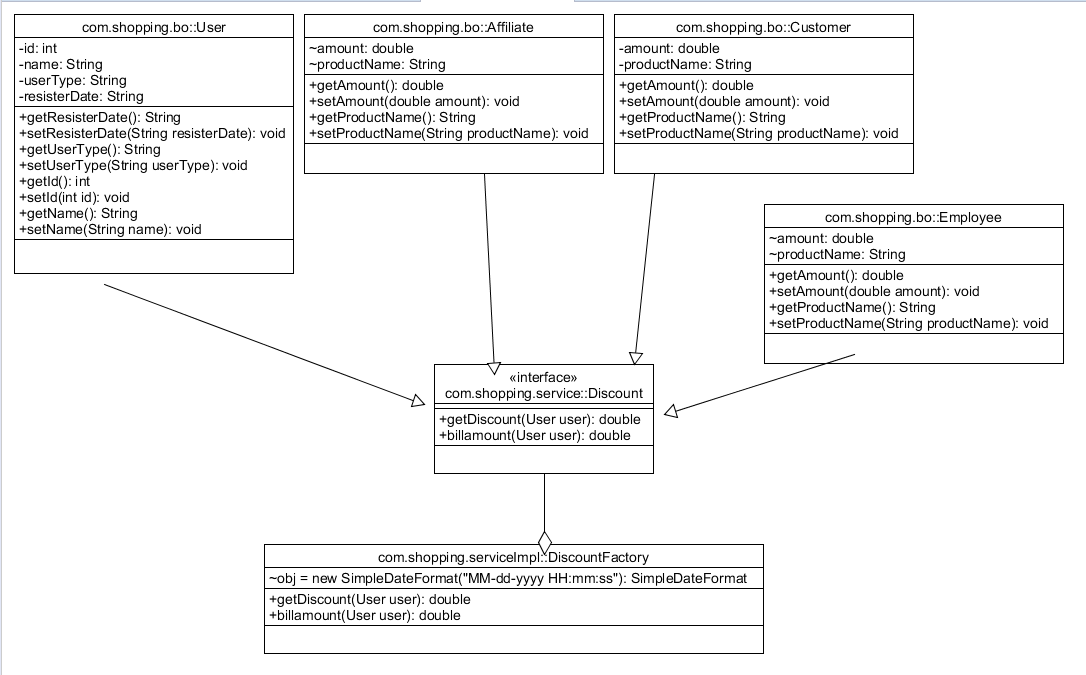
There is 3 type User exist :

1. Employee
2. Affiliate
3. Customer

**Component Changes:**

1. Affiliate.java
2. Customer.java
3. Employee.java
4. User.java
5. Discount.java
6. DiscountFactory

**Class Diagram**



**Code snipet :**

Create a Maven project as Cart Application.

2 source(src) folder is created.

1. Src/main/java – which contain the business logic
2. Src/test/java- which contain the Junit testcase.

Use case: 3 types of Users (1. Employee ,2. Customer, 3-Affiliate)

User.java

**private** **int** id;

**private** String name;

**private** String userType;

**private** String resisterDate;

Employee ,Customer,Affiliate extend User class

**double** amount;

String productName;

If User type is Employee or Customer or Affiliate, Based on user discount create the user object.

If User type is employee, then create the Employee Object, Call the appropriate discount for this user.

So, for this create an interface Discount. Java

**public** **interface** Discount {

**public** **double** userType(User user);

}

Implementation class ,

**public** **class** DiscountFactory **implements** Discount

@Override

**public** **double** userType (User user) {

String UserType = user.getUserType();

If userType as an Employee, then call the getDiscount method particular for the Employee related. Based on User type Object will create and call the appropriate business logic.

**if** (UserType.equals("Employee ")) {

double amount = getDiscount(user);

}

If userType as an Customer, then call the getDiscount method particular for the Customer related discount logic.

**if** (UserType.equals("Customer")) {

double amount = getDiscount(user);

}

If userType as an Affiliate, then call the getDiscount method particular for the Affiliate related discount logic.

So Added getDiscount method inside the interface .

**public** **interface** Discount {

**public** **double** userType(User user);

**public** **double** getDiscount(User user);

}

Inside the implementation class, Call the discount business logic based on the user type.

Check if condition , if user is an instance of Employee , then go to discount applied for employee or else it will other if condition for other user

**Below code snippet**

**public** **double** getDiscount(User user) {

**double** amount = 0.00;

**if** (user **instanceof** Employee) {

String productName = ((Employee) user).getProductName();

amount = ((Employee) user).getAmount();

**if** (!productName.equals("groceries")) {

**double** quotient = (amount \* 30) / 100;

amount = amount - quotient;

System.***out***.println("Total Bill::" + amount);

System.***out***.println("Employee 30% Discount Apply" + quotient);

System.***out***.println("After discount::" + amount);

} **else** {

**if** (amount > 100) {

amount = extracted(amount);

}

}

**Else if** (user **instanceof** Customer) {

Apply discount for customer

If customer cart having not grocery, then check if

Customer resister before 2 years, Then applied 5 % discount .

Else part if Customer cart not having grocery, Then applied 5$ discount

} **Else if** (user **instanceof** Affiliate) {

Apply discount for Affiliate

If Affiliate cart having not grocery, then

apply 10 % discount .

Else part if Customer cart not having grocery, then applied 5$ discount

**}**

**Assumptions:**

Write a test Junit test case :

Create a class DiscountFactoryTest. And create a method

@Test

**void** testBillamountCustomer() {

}

Using stub data, create Employee, customer and Affiliate object.

Check the actual amount and expected amount pass or fails

Example

@Test

**void** testBillamountOtherCustome\_before2year\_resister() {

DiscountFactory discountFactory = **new** DiscountFactory();

Customer c = **new** Customer();

c.setId(1001);

c.setAmount(1000.00);

c.setProductName("Dress");

c.setName("sushil");

c.setUserType("Customer");

c.setResisterDate("14-09-2021 07:15:50");

**double** expectedAmt= discountFactory.billamount(c);

*assertEquals*(951.00, expectedAmt);

}

Similarly check the Employee and Affiliate discount .

Also need to test for all different scenario, Like Null check, Exception, performance, timeout Error .