Java Assignments

Chapter: Inheritance and Polymorphism

Level : Beginner

1. Create a class Book with the following attributes :

   Isbn : text, title:text, price:double ,stock:long

   Override toString() to print all the details with a ":" as delimiter.

2. Create a group of classes (Person,Address,HR,Employee,Customer,Manager,Biller) as per the following rules.
   1. Class Relationship have to be logical
   2. All persons have an address.
   3. Address has plotNo,streetNo,city.
   4. HR maintains the details of Employee
   5. Biller generates a Bill for any Customer, printing his complete details.
   6. All the classes except Billerclass have a print() method to print the details.

   Test the above scenario with 1 instance of every class.

   (Note : Employee instance should print all details of Person also).

3. Create an abstract class Payment with functionality makePayment(..) that has 2 implementations in CashPayment and CreditCardPayment.

   Class PaymentUtility has a method printReceipt(Customer c,Payment p) That prints the customer details and Payment Details.

   If Payment is Cash prints the message that Customer gets a discount of 10% and if Payment is CreditCard he gets 5% reward points.

Level : Intermediate

4. In Class Book override equals() and hashCode() methods such that no 2 books can have same HashCode.

Ex: if b1 and b2 are Book Objects b1.equals(b2) should return true of isbn of b1 and b2 are same.

Level : Expert

Case Study Topic: Java Classes and their relations.

A

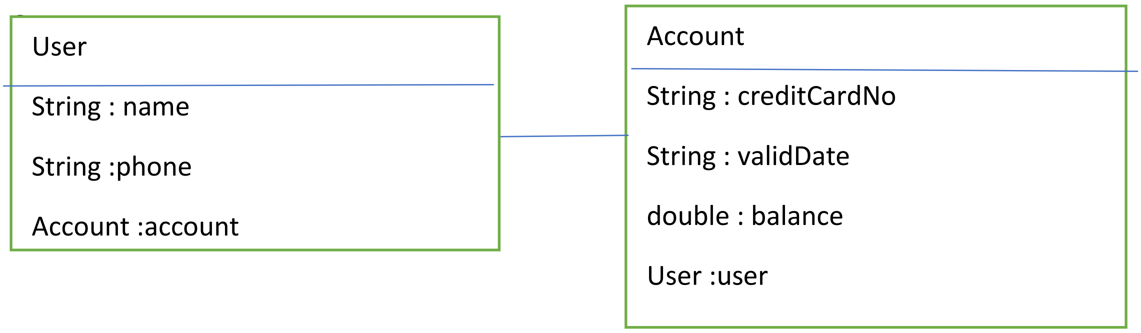
In an EWallet app there are different types of Users.

User registers for the first time and an account is created .

he logins every time he uses the app.

Create Classes for the above scenario and test it with a client code.

Refer the following class diagram.



User

String : name

String :phone

Account :account

Account

String : creditCardNo

String : validDate

double : balance

User :user

Sample Client Code:

```
User newUser = User("sam","8998772222");

Account userAccount= new Account("2223 4445 3333 2222","12/7/2020",500,newUser);

//print the details of the newUser from userAccount.
```

B User can be a Customer,Employee or a Student.

Customer has a custId that is autogenerated and email id.

Employee has an empCode and dept he works in.

Student has college name.

Create classes for the above and test it with a client program,.

C

User has a functionality giveOffers() that gives offers to different types of Users.

For Customers 10% cash back.

Employees will get a 20% cash back and Students get a 25% cash back.

D Since giveOffers() is not implemented in User and should be available as a contract for the different users to implement it can be marked as abstract and User class as abstract.

E

Modify the above application for the following sample client code:

//Read the type of user (customer.employee,student) in userType.

User user = Factory.getUser(userType);

UtilityClass.printDetails(user);

User.giveOffers();

F Create an Interface Payment that has a method printInvoice(double amount) that is implemented by customer class.

printInvoice(..) method prints the Customer details along with the current date and time and the amount.