## **Objective:**

To understand the concepts of Inheritance, Constructor chaining, Overriding and Polymorphism, instanceof

## **Class Demos**

1. Create a class Person that has name and address.
2. Test equals() and Hashcode()
3. Create class Student that has regno that is autogenerated.
4. Create class Teacher that has department and subjects handled.
5. Create an abstract class Shape that has draw() abstract method. With 2 implementations.
6. Create a factory class ShapeFactory that has method getShape(choice) to get the shape object.
7. Client calls getShape(choice) and invokes draw()
8. An Online Store has offers for different types of users. User can be a registered customer, Employee of the Store or a Student.
   1. giveOffers() is a common functionality that gives 10% discount to all Customers , 10% reward points to Employees and 20% discount to all Students.
   2. User has name. Customer has Email.Employee has empCode and Student has regNo.
   3. Create a Utility Class that calls giveOffers based on the type of User and also prints all the details of User.

## **Java Assignments – as003**

## **Chapter: Inheritance ,Polymorphism, Overriding**

## **Level : Beginner**

1. Create a class Customer that extends User with (name,address,id).
   1. If User class is in a different project create a jar and link it
   2. Customer has email and customerId which is autogenerated.
   3. Customer has an Account
   4. Account has username, password. Where password is autogenerated while creating but can be updated. Username cannot be updated and should be minimum 8 characters
   5. Account account = **new** Account("sam12345");
      1. Displays the password
   6. Create a class Employee that extends User and Has empId and dept.
   7. Customer and Employee has toString() Overridden.
   8. Create a Client class with main(..) that has the following sample code, and add printDetails() accordingly.

Customer c=new Customer(….); // displays the customerId.

Employee emp1=new Employee(..);

printDetails(c);

printDetails(emp1);

printDetails() is a single method that prints email for Customer , dept for Employee along with name and address.

1. Create a class Book with isbn, title, price and stock as attributes with constructors ,getters and setters.
   1. Test it in Client as follows.
   2. Book b1=new Book(“2341”,”C”,250.25,100)
   3. Book b2=new Book(“2341”,”C”,250.25,100)
   4. b1.equals(b2) should print true
   5. hascode of b1 and b2 should be same.

(concept of hashcode() and equals() contract)

1. 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

## 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.

Sam

Account

String : creditCardNo

String : validDate

double : balance

User :user

User

String : name

String :phone

Account :account

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();