CE 251: Java Programming

CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY

SOURCE: https://github.com/HARSHDUDHAT07/Assignment-2-JAVA

Prepared by HARSH_DUDHAT_21CE026

PROGRAM 2.1 :	Design a class named Circle containing following attributes and behavior.•Onedouble data field named radius. The default valueis1.•A no-argumentconstructor that creates a default circle.•A Single argument constructor that creates a Circle with the specified radius.•A method named getArea() that returns area of the Circle.•A method named getPerimeter() that returns perimeterof it.
CODE :	

```java
//Prepared by HARSH-DUDHAT_21CE026
import java.util.Scanner;
class circle
{
    double radius=1;
    final double pi=3.14;
    public circle()
    {}
    public circle(double r)
    {

        radius=r;
    }
    public double getArea()
    {

        return pi*radius*radius;
    }
    public double getperimeter()
    {

        return 2*pi*radius;
    }

}
public class pr_1 {
    public static void main(String[] args) {
        circle c1 = new circle();
        System.out.println("Area is " + c1.getArea() + "
Perimeter is " + c1.getperimeter());//take default value of
radius
```

```
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the radius of circle:");
        double r=sc.nextDouble();
        circle c2 = new circle(r);
        System.out.printf("Area is " + c2.getArea() + "
Perimeter is " + c2.getperimeter());

    }
}
```

PROGRAM 2.2 :

Design a class named Account that contains:

•A private int data field namedid for the account (default 0).

•A private double data field named balance for the account (default 500₹).

•A private double data field named annualInterestRate that stores the currentinterest rate (default 7%). Assume all accounts have the same interest rate.

•A private Date data field named dateCreated that stores the date when theaccount was created.

•A no-arg constructor that creates a default account.

•A constructor that creates an account with the specified id and initial balance

.•The accessor and mutator methods for id, balance, and annualInterestRate.

•The accessor method for dateCreated.

•A method named getMonthlyInterestRate() that returns the monthlyinterest rate.

•A method named getMonthlyInterest() that returns the monthly interest.

•A method named withdraw that withdraws a specified amount from theaccount.

•A method named deposit that deposits aspecified amount to the account.

CODE :

```
//Prepared by HARSH_DUDHAT_21CE026
import java.util.*;

class account {
    private int id;
    private double balance;  //balance for account
    private  double annualInterestRate=7;  //store the
cuurent interest rate
    private java.util.Date dateCreated; //stores account
created date.

    public account() {

        dateCreated = new java.util.Date();
    }

    account(int id, double balance) {
```

```
        this.id = id;
        this.balance = balance;
        dateCreated = new java.util.Date();
    }
    //generate mutator(getter and setter) method
    public int getId() {

        return this.id;
    }

    public double getBalance() {

        return this.balance;
    }

    public  double getAnnualInterestRate() {
        return annualInterestRate;
    }

    public void setId(int newId) {

        id = newId;
    }

    public void setBalance(double newBalance) {

        balance = newBalance;
    }

    public  void setAnnualInterestRate(double
newAnnualInterestRate) {

        annualInterestRate = newAnnualInterestRate;
    }

    public double getMonthlyInterestRate() {

        return (annualInterestRate / 100) / 12;
    }

    public double getMonthlyInterest() {

        return balance * getMonthlyInterestRate();
    }

    public void withdraw(double amount) {
        balance = balance - amount;
    }

    public void deposit(double amount) {
        balance = balance + amount;
    }

    public java.util.Date getDateCreated() {
        return dateCreated;
    }

    public void getAccountdetailes() {
        System.out.println("id : " + getId());
        System.out.println("Balance : " + getBalance());
        System.out.println("annualInterestRate : " +
```

```
getAnnualInterestRate());
        System.out.println("Monthly interest is : " +
getMonthlyInterest());
        System.out.println("This account was created at :  "
+ getDateCreated());
    }
}
class pr_2 {
    public static void main(String[] args) {
        Scanner ob=new Scanner(System.in);
        System.out.println("Enter your id:");
        int id=ob.nextInt();
        System.out.println("Enter your balance:");
        double balance=ob.nextDouble();
        System.out.println("Enter your interest:");
         double interest=ob.nextDouble();
        System.out.println("Enter widraw amount:");
         double widrawamount=ob.nextDouble();
        System.out.println("Enter deposit amount:");
        double depositamount=ob.nextDouble();

        account a = new account(id, balance);
        a.setAnnualInterestRate(interest);
        a.withdraw(widrawamount);
        a.deposit(depositamount);
        a.getAccountdetailes();
    }
}
```

PROGRAM 2.3 :

Use the Account class created as above to simulate an ATM machine. Create 10 accounts with id AC001.....AC010 with initial balance 300₹. The system prompts the users to enter an id. If the id is entered incorrectly, ask the user to enter a correct id. Once an id is accepted, display menu with multiple choices.

1.Balance inquiry

2.Withdraw money [Maintain minimum balance 300₹]

3.Deposit money

4.Money Transfer

5.Create Account

6.Deactivate Account

7.Exit Hint: Use ArrayList, which is can shrink and expand with compared to Array.

CODE :

```
// Prepared by HARSH_DUDHAT_21CE026
import java.util.Scanner;

public class pr_3 {
    static double acb = 300;
    private static final Scanner in = new Scanner(System.in);
    public static void main(String[] args) {
        Account[] accounts = new Account[10];
        for (int i = 1; i < 11; i++) {
```

```
            accounts[i - 1] = new Account(i, 300.0);
        }

        System.out.print("Enter an id (1 - 10): ");
        int id = in.nextInt();
        if (id < 1 || id > 10) {
            System.out.println("enter a correct id,written id
is incorrect");
        }

        while (true) {
            menuDisplay();
            System.out.print("Enter a choice: ");
            int choice = in.nextInt();
            Account a=new Account();
         switch(choice)
         {
            case 1:
            // a.balanceinquiry();
            break;
            case 2:
            a.withdraw();
            break;
            case 3:
            a.deposit();
            break;
            case 4:
            // a.transfer();
            break;
            case 5:
            // a.create_account();
            break;
            case 6:
            // a.Deactivate_Account();
            break;
            case 7:
            break;
            default:
            break;

         }
        }
    }
    public static void menuDisplay() {

        System.out.println("****Main menu****");
        System.out.println("1: Balance Inqury");
        System.out.println("2: Withdraw money");
```

```
        System.out.println("3: Deposit money");
        System.out.println("4: Transfer");
        System.out.println("5: Create account");
        System.out.println("6: deactivate Account");
        System.out.println("7: Exit");

    }
}
```

PROGRAM 2.4 :

(Subclasses of Account) In Programming Exercise 2, the Account class wasdefined to model a bank account. An account has the properties account number,balance, annual interestrate, and date created, and methods to deposit and withdrawfunds. Create two subclasses for checking and saving accounts. A checkingaccount has an overdraft limit, but a savings account cannot be overdrawn.Draw the UML diagram for the classes and then implement them. Writea test program that creates objects of Account, SavingsAccount, andCheckingAccount and invokes their toString() methods.

CODE :

```
//Prepared by HARSH_DUDHAT_21CE026

class Account
{
    private int id = 0;
    double balance = 500, annualInterest = 7, amount;
    String dateCreated;

    Account()
    {
        id = 0;
        balance = 50000;
        annualInterest = 7;
    }
    Account(int i, double bal)
    {
        id = i;
        balance = bal;
    }
    void setdata(int i, double bal, double aInt, String dt)
    {
        id = i;
        balance = bal;
        annualInterest = aInt;
        dateCreated = dt;
    }
    int getId()
    {
        return id;
    }
    double getBal()
    {
        return balance;
    }
    double getAnn()
    {
        return annualInterest;
    }
```

```
    double getMonthlyInterestRate()
    {
        return (annualInterest * 100) / 12;
    }
    double getMonthlyInterest()
    {
        return balance * (annualInterest * 100) / 12;
    }
    String getDt()
    {
        return dateCreated;
    }
    void withdraw(double amount)
    {
        balance -= amount;
        if (balance > 0)
        {
            System.out.println("The balance left after
withdrawal of Rs " + amount + " is Rs. " + balance);
        }
        else
        {
            System.out.println("Withdrawal of Rs " + amount +
" is not possible ");
        }
    }
    void deposit(double amount)
    {
        balance += amount;
        System.out.println("The balance left after deposit of
Rs." + amount + " is Rs. " + balance);
    }
}
class SavingAccount extends Account
{
    SavingAccount(double a)
    {
        amount = a;
        balance -= amount;
    }
    public String toString()
    {
        if (balance >= 3000)
        {
            return "The balance left after withdrawal of Rs "
+ amount + " is Rs. " + balance;
        }
        else
        {
            return "Minimum balance of Rs. 3000 is
required.";
        }
    }
}
class CheckingAccount extends Account
{
    CheckingAccount(double am)
    {
        amount=am;
        balance-=amount;
    }
```

```
    public String toString()
    {
        System.out.println("Withdrawal successful");
        return "Now the balance left is Rs. "+balance+" after
the withdrawal of Rs. "+amount;
    }
}
class pr_4
{
    public static void main(String[] args)
    {
        Account a1=new Account();
        Account a2=new Account(123456,100000);
        a2.setdata(1289031,100000,5.6,"2-8-2022");
        System.out.println("Account Details: ");
        System.out.println("Balance : "+a2.balance);
        System.out.println("Annual Interest : "+a2.getAnn());
        System.out.println("Monthly Interest Rate :
"+a2.getMonthlyInterestRate());
        System.out.println("Monthly Interest :
"+a2.getMonthlyInterest());
        System.out.println("Account was created on :
"+a2.getDt());
        a2.withdraw(12000);
        a2.deposit(15000);
        System.out.print("-------------------------\n");
        SavingAccount a=new SavingAccount(900);
        CheckingAccount b=new CheckingAccount(1000);
        System.out.println("For Saving Account : ");
        System.out.println(a);
        System.out.print("-------------------------\n");
        System.out.println("For Checking Account : ");
        System.out.println(b);
    }
}
```

PROGRAM 2.5 : Develop a Program that illustrate method overloading concept.

CODE :

```
//Prepared by HARSH_DUDHAT_21CE026

import java.util.Scanner;
public class pr_5
{
    void record(String t )
    {
        System.out.println(t);
    }
    void record(String studentName,char grade)
    {
        System.out.println("Student name is "+studentName);
        System.out.println("Student grade is "+grade);
    }
    void record(int id,String studentName,char grade)
    {
        System.out.println("Student ID is "+id);
        System.out.println("Student name is "+studentName);
        System.out.println("Student grade is "+grade);
    }
    public static void main(String[]args)
```

```
    {
        pr_5 O =new pr_5();
        Scanner ob=new Scanner(System.in);
        System.out.println("Enter the record of Students:");
        String name=ob.next();
        String a=ob.next();
        int ID=ob.nextInt();
        char gd=ob.next().charAt(0);
        System.out.println("string is:");
        O.record(a);
        System.out.println("Enter a Student name and
grade:");
        O.record(name,gd);
        System.out.println("Enter a id and name and grade:");
        O.record(ID,name,gd);

      //  System.out.println("\n\n by ID : 21CE026");

    }
}
```