**CE 251: Java Programming**

|  |  |  |
| --- | --- | --- |
| |  | | --- | | **CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY** | |  |

# PART:2

|  |  |
| --- | --- |
| Progra m : 2.1 | Design a class named Circle containing following attributes and behavior.   * • One double data field named radius. The default value is 1. * • A no-argument constructor 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 perimeter of it. |
|  | /\*Design a class named Circle containing following attributes and behavior.  • One double data field named radius. The default value is 1.  • A no-argument constructor 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 perimeter of it.  21CE002 - Andrew  \*/  import javax.sound.sampled.SourceDataLine;  class Circle{      private double radius;      public Circle(){          this.radius = 1;      }      public Circle(double r){          this.radius = r;      }      public double getArea(){          return 3.14\*radius\*radius;      }      public double getPerimeter(){          return 2\*3.14\*radius;      }  }  public class q1 {      public static void main(String[] args) {          Circle c1 = new Circle();          Circle c2 = new Circle(10);          System.out.println(c1.getArea());          System.out.println(c2.getPerimeter());      }    } |

|  |  |
| --- | --- |
| Program: 2.2 | Design a class named Account that contains: • • A private int data field named id 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 current interest rate (default 7%). Assume all accounts have the same interest rate. * • A private Date data field named dateCreated that stores the date when the account 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 monthly interest rate. * • A method named getMonthlyInterest() that returns the monthly interest. * • A method named withdraw that withdraws a specified amount from the account. * • A method named deposit that deposits a specified amount to the account. |
|  | /\*Design a class named Account that contains:  • A private int data field named id 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 current interest rate (default 7%). Assume all  accounts have the same interest rate.  • A private Date data field named dateCreated that  stores the date when the account 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 monthly interest rate.  • A method named getMonthlyInterest() that returns the  monthly interest.  • A method named withdraw that withdraws a specified  amount from the account.  • A method named deposit that deposits a specified  amount to the account.\*/  //21CE002 - Andrew  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 q2{  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();  }  } |

|  |  |
| --- | --- |
| Progra m : 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₹]**   1. **Deposit money** 2. **Money Transfer** 3. **Create Account** 4. **Deactivate Account** 5. **Exit**     **Hint**: Use **ArrayList**, which is can shrink and expand with compared to Array. |
|  | /\*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  21CE002 - Andrew   \*/  import java.util.Scanner;  import java.util.ArrayList;  import java.lang.NullPointerException;  class q3  {  public static void main(String[] args)throws NullPointerException  {  int i;  ArrayList<Account> arrL=new ArrayList<Account>();  for(i=0;i<10;i++)  {  Account ac=null;  if(i<9)  ac=new Account("AC00"+(i+1));  else  ac=new Account("AC0"+(i+1));  arrL.add(ac);  }  Account ac=null;  Scanner sc=new Scanner(System.in);  int temp;  do  {  Account.menu();  System.out.print("\nEnter the choice : ");  temp=sc.nextInt();  if(temp==7)  System.exit(0);  if(temp==5)  {  System.out.print("\nEnter the Account id : ");  String id=sc.next();  ac=new Account(id);  arrL.add(ac);  System.out.println("Your account has been successfully created.");  continue;  }  if(temp==6)  {  System.out.print("\nEnter the Account id : ");  String id=sc.next();  for(i=0;i<arrL.size();i++)  {  ac=arrL.get(i);  if(id.equals(ac.getId()))  {  arrL.remove(i);  break;  }  }  System.out.println("Your account has been successfully deleted.");  continue;  }  do  {  System.out.print("\nEnter the Account id : ");  String id=sc.next();  for(i=0;i<arrL.size();i++)  {  ac=arrL.get(i);  if(id.equals(ac.getId()))  break;  }  if(i==arrL.size())  System.out.println("Enter valid id ");  else  break;  }while(true);  switch(temp)  {  case 1:  ac.balanceInquiry();  break;  case 2:  ac.withdrawMoney();  break;  case 3:  ac.depositMoney();  break;  case 4:  int j;  System.out.print("Enter the Transfered Account id : ");  String id1=sc.next();  System.out.print("Enter the Transfered amount : ");  double db=sc.nextDouble();  Account ac1=null;  for(j=0;j<10;j++)  {  ac1=arrL.get(j);  if((ac1.getId()).equals(id1))  break;  }  ac.moneyTransfer(ac1,db);  break;  default:  System.out.println("Please Enter the valid choice.");  }  }while(true);  }  }  class Account  {  private Scanner sc=new Scanner(System.in);  private String id="";  private double balance=300;  public Account(String id)  {  this.id=id;  }  public Account(String id,double balance)  { this.id=id;  this.balance=balance;  }  public Account()  {  System.out.print("\nEnter the Account id : ");  id=sc.next();  System.out.print("\nEnter the initial amount : ");  balance=sc.nextDouble();  }  public void balanceInquiry()  {  System.out.println("Your Bank balance is : "+balance);  }  public void withdrawMoney()  {  int flag=0;  do  {  System.out.print("\nEnter the withdral amount : ");  double temp=sc.nextDouble();  if(temp<0)  System.out.println("Please enter the valid withdraw amount.");  else if(balance-temp>=300)  { balance-=temp;  break; }  else  System.out.println("you are not able to withdraw money.");  }while(true);  System.out.println("Your available balance is : "+balance);  }  public void depositMoney()  {  do  {  System.out.print("\nEnter the deposit amount : ");  double temp=sc.nextDouble();  if(temp<0)  System.out.println("Please enter the valid deposit amount.");  else  { balance+=temp;  break;  }  }while(true);  System.out.println("Your available balance is : "+balance);  }  public void moneyTransfer(Account ac1,double bal)  {  if(this.balance-bal>=300)  {  this.balance-=bal;  ac1.setBalance(ac1.getBalance()+bal);  System.out.println("Money Transfer is successfully completed.");  }  else  System.out.println("Money Transfer is not successfully completed.");  }  public void createAccount(Account ac)  { ac=new Account();  System.out.println("Your Account has been successfully created.");  }  public static void menu()  {  System.out.println("\n 1. Balance inquiry \n 2. Withdraw money \n 3. Deposit money\n 4. Money Transfer\n 5. Create Account\n 6. Deactivate Account\n 7. Exit");  }  public void setBalance(double balance)  {  this.balance=balance;  }  public double getBalance()  {  return balance;  }  public String getId()  {  return id;  }  } |
| Progra m : 2.4 | (Subclasses of Account) In Programming Exercise 2, the Account class was defined to model a bank account. An account has the properties account number, balance, annual interest rate, and date created, and methods to deposit and withdraw funds. Create two subclasses for checking and saving accounts. A checking account has an overdraft limit, but a savings account cannot be overdrawn. Draw the UML diagram for the classes and then implement them. Write a test program that creates objects of Account, SavingsAccount, and CheckingAccount and invokes their toString() methods. |
|  | /\*(Subclasses of Account) In Programming Exercise 2, the Account class  was defined to model a bank account.  An account has the properties account number, balance, annual  interest rate, and date created,  and methods to deposit and withdraw funds. Create two  subclasses for checking and saving accounts.  A checking account has an overdraft limit, but a savings  account cannot be overdrawn. Draw the UML diagram for the  classes and then implement them. Write a test program that  creates objects of Account, SavingsAccount, and  CheckingAccount and invokes their toString() methods.\*/  //21CE002 - Andrew  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 q4  {  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);  }  } |

|  |  |
| --- | --- |
| Progra m : 2.5 | Develop a Program that illustrate method overloading concept. |
|  | //Develop a Program that illustrate method overloading concept.  //21CE002 - Andrew  import java.util.Scanner;  public class q5  {  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)  { q5 O =new q5();  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);  }  } |