Rajalakshmi Engineering College

Name: Nakshatra Pa
Email: 241901062@rajalakshmi.edu.in
Roll no: 241901062
Phone: 8838047354
Branch: REC
Department: CSE (CS) - Section 1
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

2024_28_III_OOPS Using Java Lab

REC_2028_OOPS using Java_Week 6_CY

Attempt : 1
Total Mark : 40
Marks Obtained : 40

Section 1 : Coding

1. Problem Statement

A bank provides two types of deposit schemes: Fixed Deposits (FD) and Recurring Deposits (RD). Customers want to calculate the interest they can earn based on their selected scheme.

Develop a Java program using inheritance to compute the interest for FD and RD. The program should include:

A base class Account with attributes accountHolder and principalAmount, along with a method for interest calculation.A subclass FixedDeposit that calculates interest for FD.A subclass RecurringDeposit that calculates interest for RD.

Formulas Used:

Interest for FD: (principal amount * duration in years * rate of interest) / 100

Interest for RD: (maturity amount * duration in months * rate of interest) / (12 * 100), where maturity amount = monthly deposit * duration in months.

Input Format

The first line of input consists of the choice (1 for FD, 2 for RD).

If the choice is 1, the following lines consist of account holder (string), principal amount (double), duration in years (int), and rate of interest (double).

If the choice is 2, the following lines consist of account holder (string), monthly deposit (int), duration in months (int), and rate of interest (double).

Output Format

The output prints the calculated interest with one decimal place in the following format.

For choice 1: "Interest for FD: <calculated interest >"

For choice 2: "Interest for FD: <calculated interest >"

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
Alice
50000.56
5
6.5

Output: Interest for FD: 16250.2

Answer

```
import java.util.Scanner;

// You are using Java
import java.util.Scanner;

class Account {
  protected String accountHolder;
```

```
    protected double principalAmount;

    public Account(String accountHolder, double principalAmount) {
        this.accountHolder = accountHolder;
        this.principalAmount = principalAmount;
    }

    public double calculateInterest() {
        return 0.0; // To be overridden
    }
}

class FixedDeposit extends Account {
    private int durationYears;
    private double rateOfInterest;

    public FixedDeposit(String accountHolder, double principalAmount, int
durationYears, double rateOfInterest) {
        super(accountHolder, principalAmount);
        this.durationYears = durationYears;
        this.rateOfInterest = rateOfInterest;
    }

    @Override
    public double calculateInterest() {
        return (principalAmount * durationYears * rateOfInterest) / 100;
    }
}

class RecurringDeposit extends Account {
    private int durationMonths;
    private double monthlyDeposit;
    private double rateOfInterest;

    public RecurringDeposit(String accountHolder, double monthlyDeposit, int
durationMonths, double rateOfInterest) {
        super(accountHolder, monthlyDeposit);
        this.monthlyDeposit = monthlyDeposit;
        this.durationMonths = durationMonths;
        this.rateOfInterest = rateOfInterest;
    }
    @Override
    public double calculateInterest() {
        double maturityAmount = monthlyDeposit * durationMonths;
        return (maturityAmount * durationMonths * rateOfInterest) / (12 * 100);
    }
}


public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        int choice = sc.nextInt();

        switch (choice) {
            case 1:
                sc.nextLine();
                String fdName = sc.nextLine();
                double fdPrincipal = sc.nextDouble();
                int fdDuration = sc.nextInt();
                double fdRate = sc.nextDouble();

                FixedDeposit fd = new FixedDeposit(fdName, fdPrincipal, fdDuration,
fdRate);
                System.out.printf("Interest for FD: %.1f", fd.calculateInterest());
                break;

            case 2:
                sc.nextLine();
                String rdName = sc.nextLine();
                int rdDeposit = sc.nextInt();
                int rdDuration = sc.nextInt();
                double rdRate = sc.nextDouble();

                RecurringDeposit rd = new RecurringDeposit(rdName, rdDeposit,
rdDuration, rdRate);
                System.out.printf("Interest for RD: %.1f", rd.calculateInterest());
                break;

            default:
                System.out.println("Invalid Choice");
        }
    }
```

}

Status : Correct *Marks* : *10/10*

2. Problem Statement

A painter needs to determine the cost to paint different shapes based on their surface area. The program should be designed to handle the area of a sphere and calculate the total painting cost using the following formulas:

Area of sphere: Area = 4 * pi * r^2 where pi = 3.14Total painting cost: Cost = cost per square meter * area of sphere

The program will consist of three classes:

Shape class: This class should set the shape type and radius.Area class: This class should extend Shape to calculate the area.Cost class: This class should extend Area to calculate the total painting cost.

Input Format

The input consists of a string representing the shape type, a double value representing the radius, and another double value representing the cost per square meter on each line.

Output Format

For a valid shape type of "Sphere":

- The first line prints: "Area of Sphere is: <calculated_area>" rounded to two decimal places.
- The second line prints: "Cost to paint the shape is: <total_painting_cost>" rounded to two decimal places.

For any other shape types, print: "Invalid type".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: Sphere

```
3.4
5.8
Output: Area of Sphere is: 145.19
Cost to paint the shape is: 842.12
```

Answer

```
import java.util.Scanner;

class Shape{
   String type;
   Scanner scanner;
   double radius;
   public  void setShape(String type, Scanner scanner){
      this.type=type;
      this.scanner=scanner;
      if(type.equals("Sphere"))
      radius=scanner.nextDouble();
      else
      System.out.println("Invalid Type");
   }

}

class Area extends Shape{
   double area;
   public void calculateArea(){

   if(type.equals("Sphere")){
       area=4*3.14*radius*radius;
      System.out.printf("Area of sphere is: %.2f",area);
   }
   else{
      return;
   }
   }
}
class Cost extends Area{
   double costToPaint;
   public  void setCost(double costToPaint){
      this.costToPaint= costToPaint;
   }
   public  void calculateCost(){
    if(type.equals("Sphere")){
      System.out.printf("Cost to paint the shape is: %.2f",(area* costToPaint));
  }
  else{
    return;
  }

  }
}


public class Main {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    String s = scanner.next();
    Cost shape = new Cost();
    shape.setShape(s, scanner);
    double costToPaint = scanner.nextDouble();
    shape.calculateArea();
    shape.setCost(costToPaint);
    shape.calculateCost();
  }
}
```

Status : Correct ***Marks : 10/10***

3. Problem Statement

Mary is managing a business and wants to analyze its profitability. She operates both a regular business model and a seasonal business model. To assess profitability, she uses a program that calculates and compares the profit margins for both models based on revenue and cost.

The program defines:

BusinessUtility class with a method calculateMargin(double revenue, double cost).SeasonalBusinessUtility (inherits from BusinessUtility) and overrides calculateMargin(double revenue, double cost), adding a seasonal adjustment of 10% to the base margin.ProfitabilityChecker class with a method checkProfitability(double regularMargin), which prints "Business is profitable." if the regular margin is 10% or more, otherwise prints "Business is not profitable.".

Mary inputs revenue and cost, and the program compute and display the regular and seasonal margins using:

Margin = ((Revenue − Cost) / Revenue) × 100

Seasonal Margin = Margin + 10

Input Format

The first line of input consists of a double value r, representing the revenue.

The second line consists of a double value c, representing the cost.

Output Format

The first line prints a double value, representing the regular profit margin, rounded to two decimal places, in the format: "Regular Margin: X. XX%", where X.XX denotes the calculated regular margin.

The second line prints a double value, representing the seasonal profit margin, rounded to two decimal places, in the format: "Seasonal Margin: X. XX%", where X.XX denotes the calculated seasonal margin.

The third line prints a string, indicating whether the business is profitable or not profitable, based on the regular margin.

If the regular margin is less than 10, print "Business is not profitable.". If it is 10 or greater, print "Business is profitable."

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 1000.0
800.0

Output: Regular Margin: 20.00%
Seasonal Margin: 30.00%
Business is profitable.

Answer

```
import java.util.Scanner;

// You are using Java
import java.util.Scanner;

class BusinessUtility {
   public double calculateMargin(double revenue, double cost) {
      return ((revenue - cost) / revenue) * 100;
   }
}

class SeasonalBusinessUtility extends BusinessUtility {
   @Override
   public double calculateMargin(double revenue, double cost) {
      double baseMargin = super.calculateMargin(revenue, cost);
      return baseMargin + 10; // Adding 10% seasonal adjustment
   }
}

class ProfitabilityChecker {
   public void checkProfitability(double regularMargin) {
      if (regularMargin >= 10) {
         System.out.println("Business is profitable.");
      } else {
         System.out.println("Business is not profitable.");
      }
   }
}


class Main {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      double revenue = scanner.nextDouble();
      double cost = scanner.nextDouble();
      BusinessUtility business = new BusinessUtility();
      SeasonalBusinessUtility seasonalBusiness = new
SeasonalBusinessUtility();
      double regularMargin = business.calculateMargin(revenue, cost);
      double seasonalMargin = seasonalBusiness.calculateMargin(revenue,
cost);

      System.out.printf("Regular Margin: %.2f%%\n", regularMargin);
```

```
        System.out.printf("Seasonal Margin: %.2f%%\n", seasonalMargin);

        ProfitabilityChecker checker = new ProfitabilityChecker();
        checker.checkProfitability(regularMargin);
        scanner.close();
    }
}
```

Status : Correct ***Marks : 10/10***

4. Problem Statement

Adams has a reputation company with a great number of employees. He must calculate the salary weekly according to the hourly rate and working hours. Create a program to define a class Employee with attributes name and hourly rate. Create a subclass HourlyEmployee that calculates the weekly salary based on the number of hours worked.

(The first 40 hours are based on the regular hour rate. If the work hours are greater than 40 then the work wage is 1.5 times the hourly rate)

Note: Use Math(Math.max, Math.min) functions .

Example

Input:

Chris

10

45

Output:

Weekly Salary: Rs.475.00

Explanation:

Calculation:

The first 40 hours are paid normally: 40 × 10 = 400.00The extra 5 hours are paid at 1.5 times the hourly rate: 5 × (10×1.5) = 5 × 15 = 75.00Total salary:

400.00 + 75.00 = 475.00

Input Format

The first line of input consists of a string that represents the name of the employee.

The second line consists of a double value that represents the rate for an hour.

The last line consists of an integer that represents the total hours worked.

Output Format

The output displays the total salary of the employee, where salary is rounded to two decimal places in the format: "Weekly Salary: Rs.<double value>".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: Dave
10.0
40
Output: Weekly Salary: Rs.400.00

Answer

```
import java.util.Scanner;
import java.text.DecimalFormat;

// You are using Java
import java.util.Scanner;

class Employee {
   protected String name;
   protected double hourlyRate;
   protected int hoursWorked;

   public Employee(String name, double hourlyRate, int hoursWorked) {
      this.name = name;
      this.hourlyRate = hourlyRate;
      this.hoursWorked = hoursWorked;
   }
}
class HourlyEmployee extends Employee {

   public HourlyEmployee(String name, double hourlyRate, int hoursWorked) {
      super(name, hourlyRate, hoursWorked);
   }

   public double calculateWeeklySalary() {
      int regularHours = Math.min(hoursWorked, 40);
      int extraHours = Math.max(hoursWorked - 40, 0);

      double regularPay = regularHours * hourlyRate;
      double overtimePay = extraHours * (hourlyRate * 1.5);

      return regularPay + overtimePay;
   }
}


public class Main {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);

      String name = scanner.nextLine();
      double hourlyRate = scanner.nextDouble();
      int hoursWorked = scanner.nextInt();

      HourlyEmployee employee = new HourlyEmployee(name, hourlyRate,
hoursWorked);

      double weeklySalary = employee.calculateWeeklySalary();
      DecimalFormat df = new DecimalFormat("#.00");
      String formattedSalary = df.format(weeklySalary);
      System.out.println("Weekly Salary: Rs." + formattedSalary);
      scanner.close();
   }
}
```

Status : Correct   ***Marks : 10/10***