Rajalakshmi Engineering College

Name: Nikitha A
Email: 241901073@rajalakshmi.edu.in
Roll no: 241901073
Phone: 7200177269
Branch: REC
Department: CSE (CS) - Section 2
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

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;


class Employee {
    private String name;
    private double hourlyRate;

    public Employee(String name, double hourlyRate) {
        this.name = name;
        this.hourlyRate = hourlyRate;
    }

    public String getName() {
        return name;
    }

    public double getHourlyRate() {
        return hourlyRate;
    }
}

class HourlyEmployee extends Employee {
    private int hoursWorked;

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

    public int getHoursWorked() {
        return hoursWorked;
    }

    public double calculateWeeklySalary() {
        int regularHours = Math.min(hoursWorked, 40);
        int overtimeHours = Math.max(0, hoursWorked - 40);
        double regularPay = regularHours * getHourlyRate();
        double overtimePay = overtimeHours * (getHourlyRate() * 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**

2. 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;

class BusinessUtility {
    public double calculateMargin(double revenue, double cost) {
```

```
        return ((revenue - cost) / revenue) * 100;
    }
}
class SeasonalBusinessUtility extends BusinessUtility {
    public double calculateMargin(double revenue, double cost) {
        double baseMargin = super.calculateMargin(revenue, cost);
        return baseMargin + 10;

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

3. 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;

class Account {
   String accountHolder;
   double principalAmount;


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

   double calculateInterest() {
      return 0;
   }
}


class FixedDeposit extends Account {
   int durationInYears;
   double rateOfInterest;
   public FixedDeposit(String accountHolder, double principalAmount, int
durationInYears, double rateOfInterest) {
      super(accountHolder, principalAmount);
      this.durationInYears = durationInYears;
      this.rateOfInterest = rateOfInterest;
   }


   double calculateInterest() {
      return (principalAmount * durationInYears * rateOfInterest) / 100;
```

```
    }
}

class RecurringDeposit extends Account {
    int durationInMonths;
    double rateOfInterest;
    int monthlyDeposit;
    public RecurringDeposit(String accountHolder, int monthlyDeposit, int
durationInMonths, double rateOfInterest) {
        super(accountHolder, 0);
        this.monthlyDeposit = monthlyDeposit;
        this.durationInMonths = durationInMonths;
        this.rateOfInterest = rateOfInterest;
    }
    double calculateInterest() {
        double maturityAmount = monthlyDeposit * durationInMonths;
        return (maturityAmount * durationInMonths * 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**

4. Problem Statement

Arun wants to calculate the age gap between the grandfather and the son and determine the father's age after 5 years.

Your task is to assist him in developing a program using three classes: GrandFather, Father, and Son, where the GrandFather stores the grandfather's age, the Father extends GrandFather to include the father's age and calculates his age after 5 years, and Son extends Father to include the son's age and calculate the age difference between the grandfather and the son.

Input Format

The input consists of three integers representing the ages of the grandfather, father, and son, one per line.

Output Format

The first line of output prints "Grandfather and son's age gap:" followed by an integer representing the age gap between the grandfather and the son, ending with "years".

The second line prints "Father's Age:" followed by an integer representing the father's age after 5 years, ending with "years".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 50
30
3
Output: Grandfather and son's age gap: 47 years
Father's Age: 35 years

Answer

```
import java.util.Scanner;

class GrandFather {
    private int grandfatherAge;

    public void setGrandfatherAge(int grandfatherAge) {
        this.grandfatherAge = grandfatherAge;
    }

    public int getGrandfatherAge() {
        return grandfatherAge;
    }
}

class Father extends GrandFather {
    private int fatherAge;

    public void setFatherAge(int fatherAge) {
        this.fatherAge = fatherAge;
    }

    public int getFatherAge() {
        return fatherAge;
    }

    public int calculateFatherAgeAfter5Years() {
        return getFatherAge() + 5;
    }
```

```
}

class Son extends Father {
    private int sonAge;

    public void setSonAge(int sonAge) {
        this.sonAge = sonAge;
    }

    public int getSonAge() {
        return sonAge;
    }

    public int calculateGrandfatherSonAgeDifference() {
        int ageDifferencGrandfatherSon = getGrandfatherAge() - getSonAge();
        return ageDifferencGrandfatherSon;
    }
}

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

        int grandfatherAge = scanner.nextInt();
        son.setGrandfatherAge(grandfatherAge);

        int fatherAge = scanner.nextInt();
        son.setFatherAge(fatherAge);

        int sonAge = scanner.nextInt();
        son.setSonAge(sonAge);

        System.out.println("Grandfather and son's age gap: "+
son.calculateGrandfatherSonAgeDifference() + " years");

        int fatherAgeAfter5Years = son.calculateFatherAgeAfter5Years();
        System.out.println("Father's Age: " + fatherAgeAfter5Years + " years");
    }
}
```

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