

Rajalakshmi Engineering College

Name: KAVYA SRIRAM

Email: 241901045@rajalakshmi.edu.in

Roll no: 241901045

Phone: 8939657782

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 7_CY

Attempt : 1

Total Mark : 40

Marks Obtained : 40

Section 1 : Coding

1. Problem Statement

A developer aims to create a budget management system using two interfaces, ExpenseRecorder for recording expenses and BudgetCalculator for calculating remaining budgets.

The ExpenseTracker class implements these interfaces, allowing users to input an initial budget and record expenses iteratively until entering 0.0 as a sentinel value.

The program then computes and displays the remaining budget or notifies of budget exceedance.

Example

Input

100.0

20.0 30.0 10.0 0.0

Output

Remaining budget: Rs. 40.00

Explanation

The initial budget is 100.0. Expenses of 20.0, 30.0, and 10.0 are recorded.

Remaining budget is calculated ($100.0 - 20.0 - 30.0 - 10.0 = 40.0$).

Input Format

The first line of input is the initial budget as a double-point number (double type).
The budget is a positive number.

The second line of input consists of individual expenses as double-point numbers. Each expense is separated by space.

To end the input, an expense of 0.0 is used.

Output Format

The output displays the remaining budget, formatted to two decimal places, in the following format:

If the remaining budget (double type) is non-negative, it prints "Remaining budget: Rs. [remainingBudget]".

If the remaining budget is negative, it prints "No remaining budget, You've exceeded your budget!".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 100.0

20.0 30.0 10.0 0.0

Output: Remaining budget: Rs. 40.00

Answer

```
import java.util.Scanner;
```

```
interface ExpenseRecorder {  
    void recordExpense(double expense);  
}
```

```
interface BudgetCalculator {  
    double calculateRemainingBudget();  
}
```

```
class ExpenseTracker implements ExpenseRecorder, BudgetCalculator {  
    private double initialBudget;  
    private double totalExpenses;
```

```
    public ExpenseTracker(double initialBudget) {  
        this.initialBudget = initialBudget;  
        this.totalExpenses = 0.0;  
    }
```

```
    public void recordExpense(double expense) {  
        if (expense != 0.0) {  
            totalExpenses += expense;  
        }  
    }
```

```
    public double calculateRemainingBudget() {  
        return initialBudget - totalExpenses;  
    }  
}
```

```
class Main {  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);  
        double budget = scanner.nextDouble();  
  
        ExpenseTracker tracker = new ExpenseTracker(budget);
```

```

double expense;
do {
    expense = scanner.nextDouble();
    tracker.recordExpense(expense);
} while (expense != 0.0);

double remainingBudget = tracker.calculateRemainingBudget();
if (remainingBudget >= 0) {
    System.out.printf("Remaining budget: Rs. %.2f", remainingBudget);
} else {
    System.out.println("No remaining budget, You've exceeded your
budget!");
}
}
}
}

```

Status : Correct

Marks : 10/10

2. Problem Statement:

Rathish is planning a road trip and needs a program to convert speeds between miles per hour (MPH) and kilometers per hour (KPH).

Create an interface, SpeedConverter, with a method convertSpeed(double mph). Implement the interface with MPHtoKPHConverter class, allowing Rathish to input MPH and receive the converted speed in KPH, rounded to two decimal points.

Formula: Speed in KPH = 1.60934 * Speed in MPH.

Input Format

The input consists of a single double-point number representing the speed in miles per hour (MPH).

Output Format

The output displays the converted speed (double-point number) in kilometers per hour (KPH) rounded off to two decimal points in the following format:

"Speed in KPH: <<converted speed>>".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 1.0

Output: Speed in KPH: 1.61

Answer

```
import java.util.Scanner;

interface SpeedConverter {
    double convertSpeed(double mph);
}

class MPHtoKPHConverter implements SpeedConverter {
    public double convertSpeed(double mph) {
        double kph = 1.60934 * mph;
        return Math.round(kph * 100.0) / 100.0;
    }
}

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

        double speedInMPH = scanner.nextDouble();

        SpeedConverter converter = new MPHtoKPHConverter();

        double speedInKPH = converter.convertSpeed(speedInMPH);

        System.out.printf("Speed in KPH: %.2f\n", speedInKPH);

        scanner.close();
    }
}
```

Status : Correct

Marks : 10/10

3. Problem Statement

John is developing a car loan calculator and has structured his program using two interfaces, Principal and InterestRate, defining methods for principal and interest rate retrieval.

The Loan class implements these interfaces, taking principal and annual interest rates as parameters. User input is solicited for these values, and the program ensures their validity before performing calculations. If input values are invalid (less than or equal to zero), an error message is displayed.

Note: Total interest = principal * interest rate * years

Input Format

The first line of input consists of a double value P, representing the principal.

The second line consists of a double value R, representing the annual interest rate.

The third line consists of an integer value N, representing the loan duration in years.

Output Format

If the input values are valid, print "Total interest paid: Rs. " followed by a double value, representing the total interest paid, rounded off to two decimal places.

If the input values are invalid (negative or zero values for principal, annual interest rate, or loan duration), print "Invalid input values!".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 20000.00

0.05

5

Output: Total interest paid: Rs.5000.00

Answer

```
import java.util.Scanner;

interface Principal {
    double getPrincipal();
}

interface InterestRate {
    double getInterestRate();
}

class Loan implements Principal, InterestRate {
    private double principal;
    private double interestRate;

    public Loan(double principal, double interestRate) {
        this.principal = principal;
        this.interestRate = interestRate;
    }

    public double getPrincipal() {
        return principal;
    }

    public double getInterestRate() {
        return interestRate;
    }

    public double calculateTotalInterest(int years) {
        return principal * interestRate * years;
    }
}
```

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

        double carPrice = scanner.nextDouble();

        double annualInterestRate = scanner.nextDouble();
    }
}
```

```

        int loanDuration = scanner.nextInt();

        if (carPrice <= 0 || annualInterestRate <= 0 || loanDuration <= 0) {
            System.out.println("Invalid input values!");
            return;
        }

        Loan carLoan = new Loan(carPrice, annualInterestRate);
        double totalInterest = carLoan.calculateTotalInterest(loanDuration);

        System.out.printf("Total interest paid: Rs.%.2f%n", totalInterest);
    }
}

```

Status : Correct

Marks : 10/10

4. Problem Statement

Alex and Bob are designing a control system for household appliances, and one of the appliances is a washing machine. You want to create a program to help them that models the washing machine as a motor and calculates its electricity consumption based on its capacity.

Define an interface named Motor with the following methods:

void run() double consume(double capacity)

Create a class called WashingMachine that implements the Motor interface.

In the WashingMachine class:

Implement the run() method to print "Washing machine is running." Implement a consume() method to print "Washing machine is consuming electricity." Implement the consume(double capacity) method to calculate the electricity consumption (in kWh) of the washing machine based on its capacity. The formula for electricity consumption is (capacity * 0.05).

Input Format

The input consists of a double value representing the capacity of the washing machine in kW.

Output Format

The first line of output prints "Washing machine is running."

The second line prints "Washing machine is consuming electricity."

The third line prints "Electricity consumption: X kWh" where X is a double value, rounded off to two decimal places, representing the electricity consumption.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 2.5

Output: Washing machine is running.

Washing machine is consuming electricity.

Electricity consumption: 0.13 kWh

Answer

```
import java.util.Scanner;
```

```
interface Motor {  
    void run();  
    void consume();  
    double consume(double capacity);  
}
```

```
class WashingMachine implements Motor {  
    public void run() {  
        System.out.print("Washing machine is running. ");  
    }  
  
    public void consume() {  
        System.out.print("Washing machine is consuming electricity. ");  
    }  
  
    public double consume(double capacity) {  
        return capacity * 0.05;  
    }  
}
```

```
}  
}  
public class Main {  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);  
  
        WashingMachine washingMachine = new WashingMachine();  
  
        double capacity = scanner.nextDouble();  
  
        washingMachine.run();  
        washingMachine.consume();  
  
        double consumption = washingMachine.consume(capacity);  
        System.out.printf("Electricity consumption: %.2f kWh", consumption);  
  
        scanner.close();  
    }  
}
```

Status : Correct

Marks : 10/10