

# Rajalakshmi Engineering College

Name: Neha R N  
Email: 240701354@rajalakshmi.edu.in  
Roll no: 240701354  
Phone: 9080137196  
Branch: REC  
Department: CSE - Section 7  
Batch: 2028  
Degree: B.E - CSE

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;
```

```
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 budget) {  
        this.initialBudget = budget;  
        this.totalExpenses = 0.0;  
    }
```

```
    @Override  
    public void recordExpense(double expense) {  
        totalExpenses += expense;  
    }
```

```
    @Override  
    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

Maria, an online store owner, is looking to implement a pricing system that calculates the final price of products after applying discounts. She needs a program that takes the original price of a product and the discount percentage as input and computes the final discounted price. The discount is applied as a percentage of the original price. Maria wants to ensure that the final price is formatted to display exactly two decimal places.

Implement this functionality using the PriceCalculator interface and the DiscountCalculator class.

### **Input Format**

The first line of input consists of the original price (a double value).

The second line of input consists of a discount percentage (a double value).

### **Output Format**

The output displays the final price after the discount, adhering to the following format: "Final Price after discount: \$[final\_price]".

Here, [final\_price] should be replaced with the calculated final price, formatted as a currency value with two decimal places.

Refer to the sample output for the formatting specifications.

### **Sample Test Case**

Input: 100.0

10.0

Output: Final Price after discount: \$90.00

### **Answer**

```
import java.util.Scanner;

// You are using Java
interface PriceCalculator {
    double calculatePrice(double originalPrice, double discountPercentage);
}

class DiscountCalculator implements PriceCalculator {
    public double calculatePrice(double originalPrice, double discountPercentage)
    {
        double discountAmount = (originalPrice * discountPercentage) / 100;
        return originalPrice - discountAmount;
    }
}

class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        double originalPrice = scanner.nextDouble();
        double discount = scanner.nextDouble();
        PriceCalculator calculator = new DiscountCalculator();
        double finalPrice = calculator.calculatePrice(originalPrice, discount);
        System.out.printf("Final Price after discount: $%.2f\n", finalPrice); //
        // Formats output to 2 decimal places
    }
}
```

Status : Correct

Marks : 10/10

### 3. Problem Statement:

Ray is developing a tax calculation program in Java. The program includes an interface named TaxCalculator with a method to calculate tax based on salary. The SimpleTaxCalculator class implements this interface and determines the tax to be paid based on the salary amount using progressive tax slabs.

Your task is to implement this system. The program first takes an integer T representing the number of test cases, followed by T salary values. For each salary, calculate the total tax to be paid based on the following progressive tax rules:

For the first 50,000 of salary, the tax rate is 5%. For the next 50,000 (i.e., from 50,001 to 1,00,000), the tax rate is 10%. For any amount above 1,00,000, the tax rate is 20%. (That is, only the amount above 1,00,000 is taxed at 20%.)

Example

Input

3

78000

110000

23000

Output

5300

9500

1150

Explanation

For Salary Rs. 78,000

$$\text{Tax} = 0.1 * (78,000 - 50,000) + 0.05 * 50,000 = 5,300$$

For Salary Rs. 1,10,000

$$\text{Tax} = 0.2 * (110000 - 100000) + 0.1 * 50,000 + 0.05 * 50,000 = 9,500$$

For Salary Rs. 23,000

$$\text{Tax} = 0.05 * 23,000 = 1,150$$

### ***Input Format***

The first line of the input consists of an integer, T, representing the number of test cases.

The next T lines of the input consist of a single integer, representing the annual salary of an individual, separated by a line.

### ***Output Format***

The output displays the calculated tax as an integer for each test case, separated by a line.

Refer to the sample output for the formatting specifications.

### ***Sample Test Case***

Input: 2

100

300

Output: 5

15

### ***Answer***

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

```
// You are using Java
```

```
interface TaxCalculator {
```

```
    int calculateTax(int salary);
```

```
}
```

```
class SimpleTaxCalculator implements TaxCalculator {
```

```
    public int calculateTax(int salary) {
```

```

double tax = 0;
if(salary <= 50000) {
    tax = 0.05 * salary;
} else if (salary <= 100000) {
    tax = (0.05 * 50000) + (0.10 * (salary - 50000));
} else {
    tax = (0.05 * 50000) + (0.10 * 50000) + (0.20 * (salary - 100000));
}

return (int) tax;
}
}

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

        TaxCalculator taxCalculator = new SimpleTaxCalculator();

        for (int i = 0; i < T; i++) {
            int salary = scanner.nextInt();
            int tax = taxCalculator.calculateTax(salary);
            System.out.println(tax);
        }

        scanner.close();
    }
}

```

**Status :** Correct

**Marks :** 10/10

#### 4. 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;

// You are using Java
interface SpeedConverter {
    double convertSpeed(double mph);
}

class MPHtoKPHConverter implements SpeedConverter {
    public double convertSpeed(double mph) {
        return 1.60934 * mph;
    }
}

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