

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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

### 2028\_REC\_OOPS using Java\_Week 1\_MCQ

Attempt : 2

Total Mark : 15

Marks Obtained : 15

#### Section 1 : MCQ

1. What will be the output of the following code snippet?

```
class DivisionExample {  
    public static void main(String[] args) {  
        double num1 = 10.5;  
        double num2 = 3;  
        int result = (int)(num1 / num2);  
        System.out.println(result);  
    }  
}
```

**Answer**

3

**Status :** Correct

**Marks :** 1/1

2. What will be the output of the following program?

```
class DataTypesMCQ {  
    public static void main(String[] args) {  
        int a = 10;  
        double b = 5;  
        System.out.println(a / b);  
    }  
}
```

**Answer**

2.0

**Status :** Correct

**Marks :** 1/1

3. What will be the output of the following code?

```
import java.util.*;  
  
class TernaryOperatorExample {  
    public static void main(String[] args) {  
        int a = 5, b = 10;  
        int result = (a > b) ? a : b;  
        System.out.println(result);  
    }  
}
```

**Answer**

10

**Status :** Correct

**Marks :** 1/1

4. What is the output of the following code?

```
class TestClass {  
    public static void main(String[] args) {  
        int x = 5;  
        int X = 10;  
    }  
}
```

```
int sum = x + X;  
int bitwiseResult = x | X;  
  
System.out.println(sum);  
System.out.println(bitwiseResult);  
}  
}
```

**Answer**

1515

**Status :** Correct

**Marks :** 1/1

5. Which of the following is not a primitive data type?

**Answer**

string

**Status :** Correct

**Marks :** 1/1

6. What is the result of the following expression?

```
import java.util.*;  
  
class ComplexExpressionExample {  
    public static void main(String[] args) {  
        int a = 5, b = 2, c = 3, d = 4;  
        int result = a + b * c / d - b;  
  
        System.out.println(result);  
    }  
}
```

**Answer**

4

**Status :** Correct

**Marks :** 1/1

7. Which of the following data types is used to store single characters?

**Answer**

char

**Status :** Correct

**Marks :** 1/1

8. What is the output of the following code?

```
import java.util.*;

class RelationalOperatorExample {
    public static void main(String[] args) {
        int x = 8, y = 4;
        boolean result = (x != y);

        System.out.println(result);
    }
}
```

**Answer**

true

**Status :** Correct

**Marks :** 1/1

9. What is the output of the following code?

```
class TestClass {
    public static void main(String[] args) {
        int a = 5;
        int b = 10;

        int sum = a + b;
        int bitwiseAnd = a & b;
        int bitwiseOr = a | b;

        System.out.println(sum);
        System.out.println(bitwiseAnd);
    }
}
```

```
        System.out.println(bitwiseOr);  
    }  
}
```

**Answer**

15015

**Status :** Correct

**Marks :** 1/1

10. What is the output of the following program?

```
class Arithmetic {  
    public static void main(String[] args) {  
        char ch = 'A';  
        System.out.println(ch);  
    }  
}
```

**Answer**

A

**Status :** Correct

**Marks :** 1/1

11. What is the output of the following code?

```
class TestClass {  
    public static void main(String[] args) {  
        int count = 8;  
        count = count ^ 1;  
  
        System.out.println(count);  
    }  
}
```

**Answer**

9

**Status :** Correct

**Marks :** 1/1

12. What will be the output of the following code snippet?

```
import java.util.*;

class OperatorPrecedenceExample {
    public static void main(String[] args) {
        int a = 5, b = 3, c = 2;
        int result = a + b * c;

        System.out.println(result);
    }
}
```

**Answer**

11

**Status :** Correct

**Marks :** 1/1

13. Which of the following data types is used to store floating-point numbers with greater precision?

**Answer**

double

**Status :** Correct

**Marks :** 1/1

14. What is the output of the following program?

```
class Demo {
    public static void main(String[] args) {
        String text = "Hello, World!";
        System.out.println(text);
    }
}
```

**Answer**

Hello, World!

**Status :** Correct

**Marks :** 1/1

15. What is the output of the following code?

```
class TestClass {  
    public static void main(String[] args) {  
        int a = 10;  
        int b = 3;  
        System.out.println(a / b);  
    }  
}
```

**Answer**

3

**Status :** Correct

**Marks :** 1/1

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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

### 2028\_REC\_OOPS using Java\_Week 1\_Q1

Attempt : 2

Total Mark : 10

Marks Obtained : 10

### Section 1 : Coding

#### 1. Problem Statement

Gloria is responsible for monitoring the performance of two machines in a factory. She needs to determine which of the two machines is operating closest to the optimal temperature of 100 degrees Celsius using the relational operator.

Assist Gloria in displaying the machine's temperature, which is closer to 100, and the difference from 100.

#### ***Input Format***

The first line of input consists of an integer N, representing the temperature of the first machine.

The second line consists of an integer M, representing the temperature of the second machine.



### **Output Format**

The output prints "The integer closer to 100 is X with a difference of Y" where X is the temperature of the closer machine and Y is the difference from 100.

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 90

80

Output: The integer closer to 100 is 90 with a difference of 10

### **Answer**

```
import java.util.Scanner;

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

        // Read input temperatures
        int N = scanner.nextInt(); // Temperature of first machine
        int M = scanner.nextInt(); // Temperature of second machine

        // Calculate differences from 100
        int diffN = Math.abs(100 - N);
        int diffM = Math.abs(100 - M);

        // Determine which machine is closer
        if (diffN < diffM) {
            System.out.println("The integer closer to 100 is " + N + " with a difference of " + diffN);
        } else if (diffM < diffN) {
            System.out.println("The integer closer to 100 is " + M + " with a difference of " + diffM);
        } else {
            int closer = Math.min(N, M);
            System.out.println("The integer closer to 100 is " + closer + " with a difference of " + diffN);
        }
    }
}
```

241901507

```
scanner.close();
```

```
}  
}
```

241901507

241901507

241901507

**Status :** Correct

**Marks :** 10/10

241901507

241901507

241901507

241901507

241901507

241901507

241901507

241901507

241901507

241901507

241901507

241901507

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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

### 2028\_REC\_OOPS using Java\_Week 1\_Q2

Attempt : 1

Total Mark : 10

Marks Obtained : 10

### Section 1 : Coding

#### 1. PROBLEM STATEMENT:

Dave got two students who want help with their doubt. Each hands out an integer and wants to find if one integer is positive while the other is not divisible by 3. Write a program to achieve this and conclude for them.

#### ***Input Format***

The first line of input represents the first integer.

The second line of input represents the second integer.

#### ***Output Format***

The output should display as "One of the integers is positive while the other is not divisible by 3." or "Neither of the integers meets the condition."

Refer to the sample output for the formatting specifications.

**Sample Test Case**

Input: 4

3

Output: One of the integers is positive while the other is not divisible by 3.

**Answer**

// You are using Java

import java.util.\*;

class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int num1 = sc.nextInt();

int num2 = sc.nextInt();

boolean condition1 = (num1 > 0 && num2 % 3 != 0);

boolean condition2 = (num2 > 0 && num1 % 3 != 0);

if (condition1 || condition2) {

System.out.print("One of the integers is positive while the other is not divisible by 3.");

} else {

System.out.print("Neither of the integers meets the condition.");

}

}

}

**Status : Correct**

**Marks : 10/10**

# Rajalakshmi Engineering College

Name: Sachin Krishna B  
Email: 241901507@rajalakshmi.edu.in  
Roll no: 241901507  
Phone: 9025753177  
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

### 2028\_REC\_OOPS using Java\_Week 1\_Q3

Attempt : 1  
Total Mark : 10  
Marks Obtained : 10

#### Section 1 : Coding

##### 1. Problem statement

Manoj, a developer at MoneyMatters Inc., is working on improving the company's financial system. He needs to create a program that takes an integer input, converts it into a double, and displays both the original integer and the converted double value.

##### ***Input Format***

The input consists of a single integer representing a monetary amount.

##### ***Output Format***

The first line of the output displays the "Original Integer: ", followed by an integer representation of the input value.

The second line displays the "Converted Double: ", followed by a double value representing the input as a decimal value.

Refer to the sample output for the formatting specifications.

**Sample Test Case**

Input: 20

Output: Original Integer: 20

Converted Double: 20.0

**Answer**

// You are using Java

import java.util.\*;

```
class Main {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        int number = sc.nextInt();  
  
        double converted = (double) number;  
  
        System.out.println("Original Integer: " + number);  
        System.out.println("Converted Double: " + converted);  
    }  
}
```

**Status :** Correct

**Marks : 10/10**

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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

### 2028\_REC\_OOPS using Java\_Week 1\_Q4

Attempt : 1

Total Mark : 10

Marks Obtained : 10

### Section 1 : Coding

#### 1. Problem Statement

Vishal and Arun are discussing the properties of numbers. Vishal gives Arun two integers. He asks Arun to check if the sum of these two numbers is a multiple of their product.

Can you assist Arun and determine whether the sum is a multiple of the product?

#### ***Input Format***

The input consists of two space-separated integers.

#### ***Output Format***

The output prints:

1. "Sum is Multiple of Product" if the sum of the two numbers is divisible by their product.
2. "Sum is Not Multiple of Product" otherwise.

Refer to the sample output for formatting specifications.

**Sample Test Case**

Input: 1 2

Output: Sum is Not Multiple of Product

**Answer**

```
// You are using Java
import java.util.*;
```

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

        int a = sc.nextInt();
        int b = sc.nextInt();

        int sum = a + b;
        int product = a * b;

        if (sum % product == 0) {
            System.out.println("Sum is Multiple of Product");
        } else {
            System.out.println("Sum is Not Multiple of Product");
        }
    }
}
```

**Status :** Correct

**Marks :** 10/10



# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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

### 2028\_REC\_OOPS using Java\_Week 1\_Q5

Attempt : 1

Total Mark : 10

Marks Obtained : 10

### Section 1 : Coding

#### 1. Problem Statement:

Emily has a beautiful circular garden in her backyard. She's interested in calculating two important measurements for her garden: the circumference and the area. To do this, she needs a program that can take the radius of her circular garden as input and provide the calculated circumference and area as output. The formulas she should use are as follows:

To calculate the circumference (C) of a circle, you can use the formula:

$$C = 2 * \pi * r$$

$$A = \pi * r^2$$

Where:

C represents the circumference.

A represents the area.

$\pi$  (pi) is approximately 3.14159.

r is the radius of the circle.

Emily is not a programmer, and she needs your help to create a program that will make these calculations for her garden.

### ***Input Format***

The first line of input contains a single double-point number radius, representing the radius of the circle.

### ***Output Format***

The output should consist of two lines:

The first line should print the circumference of the circle rounded to 2 decimal places, followed by the unit "meters".

The second line should print the area of the circle rounded to 2 decimal places, followed by the unit "square meters".

Refer to the sample output for formatting specifications.

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

Input: 3.0

Output: Circumference: 18.85 meters

Area: 28.27 square meters

### ***Answer***

```
// You are using Java
import java.util.*;
```

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

```
double pi = 3.14159;

double circumference = 2 * pi * radius;
double area = pi * radius * radius;

System.out.printf("Circumference: %.2f meters\n", circumference);
System.out.printf("Area: %.2f square meters\n", area);
}
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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

### 2028\_REC\_OOPS using Java\_Week 1\_Q6

Attempt : 1

Total Mark : 10

Marks Obtained : 10

### Section 1 : Coding

#### 1. Problem Statement

Joey is learning about bitwise operations and is working on a project that involves extracting specific bits from integers. He needs to write a program that takes an integer and the number of bits N as input and outputs the value of the lowest N bits of the integer.

Help Joey in his project to understand and visualize how bitwise operations work in practical scenarios.

#### ***Input Format***

The first line of input consists of an integer X, representing the given integer.

The second line consists of an integer N, representing the number of bits to extract.

### **Output Format**

The output displays "Result: " followed by an integer representing the value of the lowest N bits of the given integer.

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 85

2

Output: Result: 1

### **Answer**

```
// You are using Java
import java.util.*;
```

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

        int mask = (1 << n) - 1;
        int result = x & mask;

        System.out.println("Result: " + result);
    }
}
```

**Status : Correct**

**Marks : 10/10**

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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

### 2028\_REC\_OOPS using Java\_Week 1\_Q7

Attempt : 1

Total Mark : 10

Marks Obtained : 10

### Section 1 : Coding

#### 1. Problem Statement:

Miles is working on a program that involves analyzing two integers. He wants to check if either one of the integers is both:

Less than or equal to zero, and Odd. Can you help him create a program that identifies whether either of the integers meets these conditions?

#### ***Input Format***

The input consists of two integers on separate lines, denoted as 'input1' and 'input2'.

#### ***Output Format***

A single line with a boolean result (either 'true' or 'false') indicating whether either 'input1' or 'input2' is both less than or equal to zero and odd.

Refer to the sample output for format specifications

**Sample Test Case**

Input: -45

10

Output: true

**Answer**

// You are using Java

```
import java.util.*;
```

```
class Main {
```

```
    public static void main(String[] args) {
```

```
        Scanner sc = new Scanner(System.in);
```

```
        int input1 = sc.nextInt();
```

```
        int input2 = sc.nextInt();
```

```
        boolean result = (input1 <= 0 && input1 % 2 != 0) || (input2 <= 0 && input2 % 2 != 0);
```

```
        System.out.println(result);
```

```
    }
```

```
}
```

**Status :** Correct

**Marks : 10/10**

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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

### 2028\_REC\_OOPS using Java\_Week 1\_Q8

Attempt : 1

Total Mark : 10

Marks Obtained : 4

### Section 1 : Coding

#### 1. Problem Statement

In the Kingdom of Finance, the royal treasury is managed by the treasurer, Sir Cedric. Sir Cedric tracks the daily expenses of the kingdom using an expense report that lists three major categories: food, clothing, and utilities. However, the King wants to know if the average daily expense is greater than at least two of these categories to ensure the kingdom is spending wisely.

Your task is to help Sir Cedric determine if the average daily expense is greater than two of the categories. Specifically, you need to calculate the average of the three expenses and check if it is greater than any two categories.

Note: Use the ternary operator



### ***Input Format***

Three integers a, b, and c represent the daily expenses for food, clothing, and utilities. Each integer is provided on a single line.

### ***Output Format***

The average of the three expenses, rounded to two decimal places.

A message indicating whether the average is greater than at least two of the expense categories.

1. If the average is greater than the two smallest monthly expenses, print "Average is greater than both X and Y," where X and Y are the two smallest expenses.
2. Otherwise, display "Average is not greater than two smallest expenses".

Refer to the sample output for formatting specifications.

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

Input: 4

6

10

Output: 6.67

Average is greater than both 4 and 6

### ***Answer***

```
import java.util.*;
```

```
class Main {
```

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

```
        int a = sc.nextInt();
```

```
        int b = sc.nextInt();
```

```
        int c = sc.nextInt();
```

```
        double average = (a + b + c) / 3.0;
```

```
        int[] expenses = {a, b, c};
```

```
Arrays.sort(expenses); // Now expenses[0] and expenses[1] are the two
smallest
```

```
System.out.printf("%.2f\n", average);
```

```
System.out.println(
    (average > expenses[0] && average > expenses[1])
    ? "Average is greater than both " + expenses[0] + " and " + expenses[1]
    : "Average is not greater than two smallest expense"
```

```
);
```

```
}
```

```
}
```

**Status :** Partially correct

**Marks : 4/10**

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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

### 2028\_REC\_OOPS using Java\_Week 1\_Q9

Attempt : 1

Total Mark : 10

Marks Obtained : 10

### Section 1 : Coding

#### 1. Problem Statement

Phill is a quality control manager at a manufacturing plant. He needs to verify if a sensor reading at a midpoint station (S2) falls exactly halfway between the readings of the previous station (S1) and the next station (S3). Help him by developing a program that checks if the second sensor reading is the average (midpoint) of the first and third sensor readings.

Use the relational operator to solve the program.

#### ***Input Format***

The first line of input consists of an integer S1, representing the sensor reading of the first station.

The second line consists of an integer S2, representing the sensor reading of the midpoint station.

The third line consists of an integer S3, representing the sensor reading of the next station.

### **Output Format**

The first line of output displays a boolean value representing whether the sensor reading at the midpoint station is halfway between the readings of the first and the next stations.

The second line displays one of the following:

1. If the result is true, print "The second integer is halfway between the first and third integers."
2. Otherwise, print "The second integer is not halfway between the first and third integers."

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 1

7

10

Output: false

The second integer is not halfway between the first and third integers.

### **Answer**

```
// You are using Java
import java.util.*;
```

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

        int s1 = sc.nextInt();
        int s2 = sc.nextInt();
        int s3 = sc.nextInt();

        boolean isMidpoint = (s2 * 2) == (s1 + s3);
```

```
System.out.println(isMidpoint);  
System.out.println(isMidpoint  
    ? "The second integer is halfway between the first and third integers."  
    : "The second integer is not halfway between the first and third integers.");  
}  
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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

### 2028\_REC\_OOPS using Java\_Week 1\_Q10

Attempt : 1

Total Mark : 10

Marks Obtained : 10

### Section 1 : Coding

#### 1. Problem Statement

Aishu is supervising a construction project that needs to be completed with the help of three workers: A, B, and C.

She knows how many days each of them would take to complete the entire project individually:

A can complete it in x days, B in y days, C in z days.

Initially, all three workers (A, B, and C) work together for d1 days.

After that, C leaves, and only A and B continue for another d2 days.

Then B also leaves, and A works alone to finish the remaining work.

Your task is to help aishu to implement this functionality using the class WorkDistribution and Method calculateWork(int x, int y, int z, int d1, int d2)

Calculate the total work completed in the first  $d_1$  days by A, B, and C. Calculate the work completed in the next  $d_2$  days by A and B. Determine the remaining work after these  $d_1 + d_2$  days.

**Input Format**

The first line of input contains five space-separated integers:  $x$   $y$   $z$   $d_1$   $d_2$

where:

$x$  represents the Days A takes to complete the work alone

$y$  represents the Days B takes to complete the work alone

$z$  represents the Days C takes to complete the work alone

$d_1$  represents the Days A, B, and C work together

$d_2$  represents the Days A and B work together (after C leaves)

**Output Format**

The first line of output prints "Work done in first  $d_1$  days (A+B+C): " followed by a double value rounded to 2 decimal places.

The second line of output prints "Work done in next  $d_2$  days (A+B): " followed by a double value rounded to 2 decimal places.

The third line prints "Remaining work: " followed by a double value rounded to 2 decimal places.

Refer to the sample output for formatting specifications.

**Sample Test Case**

Input: 10 20 30 2 2

Output: Work done in first  $d_1$  days (A+B+C): 0.37

Work done in next  $d_2$  days (A+B): 0.30

Remaining work: 0.33

**Answer**

```
// You are using Java
import java.util.*;
```

```
class WorkDistribution {
    public void calculateWork(int x, int y, int z, int d1, int d2) {
        // Work done per day
        double rateA = 1.0 / x;
        double rateB = 1.0 / y;
        double rateC = 1.0 / z;

        // Work done in first d1 days by A + B + C
        double work1 = d1 * (rateA + rateB + rateC);

        // Work done in next d2 days by A + B
        double work2 = d2 * (rateA + rateB);

        // Total work done
        double totalWork = work1 + work2;

        // Remaining work
        double remaining = 1.0 - totalWork;

        // Output results rounded to 2 decimal places
        System.out.printf("Work done in first d1 days (A+B+C): %.2f\n", work1);
        System.out.printf("Work done in next d2 days (A+B): %.2f\n", work2);
        System.out.printf("Remaining work: %.2f\n", remaining);
    }
}

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

        int x = sc.nextInt(); // A's days
        int y = sc.nextInt(); // B's days
        int z = sc.nextInt(); // C's days
        int d1 = sc.nextInt(); // Days A+B+C work
        int d2 = sc.nextInt(); // Days A+B work

        WorkDistribution obj = new WorkDistribution();
        obj.calculateWork(x, y, z, d1, d2);
    }
}
```



}

**Status :** Correct

**Marks : 10/10**

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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

### 2028\_REC\_OOPS using Java\_Week 1\_PAH

Attempt : 1

Total Mark : 40

Marks Obtained : 40

### Section 1 : Coding

#### 1. PROBLEM STATEMENT:

Maria, a software developer, is working on a project to create a simple program to determine which of two integers is closest to zero. The integers can be either positive or negative. The program needs to take two integer inputs and calculate which one is closer to zero. If both integers are equidistant from zero, the program should return 0.

#### ***Input Format***

The input contains two lines:

The first line of the input contains an integer, which can be either a positive or a negative integer.

The second line of the input contains an integer, which can be either a positive or a negative integer.

### **Output Format**

The output displays the integer that is closest to zero in the following format:

"The integer closest to zero is: [closest\_integer]"

Here, [closest\_integer] should be replaced with the integer that is closer to zero based on its absolute value.

Refer to the sample output for the formatting specifications.

### **Sample Test Case**

Input: 5

8

Output: The integer closest to zero is: 5

### **Answer**

// You are using Java

import java.util.\*;

class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int num1 = sc.nextInt();

int num2 = sc.nextInt();

int result = (Math.abs(num1) < Math.abs(num2)) ? num1 :  
(Math.abs(num2) < Math.abs(num1)) ? num2 : 0;

System.out.println("The integer closest to zero is: " + result);

}

Status : Correct

Marks : 10/10

## 2. Problem Statement

Mickey and Miney are walking through a magical forest. The forest is full of enchanted stones, each with a unique number. There is a legend that says the magic power of the stones can be revealed by using a special operation. To determine the magic power of a given stone, you need to perform a bitwise AND operation with the number 15.

Each stone's number is represented by an integer, and Mickey needs to find the magic power of each stone by applying this operation.

Your task is to help Mickey compute the result of the bitwise AND operation of the given stone number with 15, and print the result.

### ***Input Format***

The input consists of a single integer.

### ***Output Format***

The output should display a single integer, which is the result of the bitwise AND operation between input and 15.

Refer to the sample output for format specifications.

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

Input: 25

Output: 9

### ***Answer***

```
// You are using Java
import java.util.*;
```

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

```
int stoneNumber = sc.nextInt();  
    int magicPower = stoneNumber & 15;  
    System.out.println(magicPower);  
}  
}
```

**Status :** Correct

**Marks :** 10/10

### 3. PROBLEM STATEMENT:

Maria, a software developer, is working on a program to determine if two given integers which can be either positive or negative integers have the same parity (both even or both odd). She needs your help in writing this program.

Write a program that takes two integers as input and checks if both integers are either even or odd.

#### ***Input Format***

The input consists of two lines:

The first line consists of an integer (input1) which can be either positive or negative.

The second line consists of an integer (input2) which can be either positive or negative.

#### ***Output Format***

The output is displayed in the following format:

If both integers have the same parity (i.e., both even or both odd), print:

"Both integers are either even or odd"

Otherwise, print:

"The integers have different parities"

Refer to the sample output for the formatting specifications.

### **Sample Test Case**

Input: 2

-4

Output: Both integers are either even or odd

### **Answer**

// You are using Java

import java.util.\*;

```
class ParityCheck {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
  
        int input1 = sc.nextInt();  
        int input2 = sc.nextInt();  
  
        if ((input1 % 2 == 0 && input2 % 2 == 0) || (input1 % 2 != 0 && input2 % 2 != 0))  
        {  
            System.out.println("Both integers are either even or odd");  
        } else {  
            System.out.println("The integers have different parities");  
        }  
    }  
}
```

**Status :** Correct

**Marks :** 10/10

### **4. Problem Statement**

In the Kingdom of Delivery Logistics, there is a giant truck used for

transporting packages across the kingdom. The truck has a maximum capacity represented by an integer, and each package also has a specific weight. The truck's efficiency and safety depend on whether the weight of the package is below a certain threshold.

The kingdom's delivery service has a rule: if the weight of a package is less than one-third of the truck's total capacity, the package is eligible for quick processing and dispatch. However, if the weight is too heavy, the package will require special handling.

As a logistics manager, you need to check whether the weight of the package is less than one-third of the truck's total capacity.

Write a program using a ternary operator that helps determine whether the package weight meets the requirement for quick processing or if it needs special handling.

#### ***Input Format***

The first line of input consists of an integer  $p$ , representing the weight of the package.

The second line consists of an integer  $w$ , representing the total weight capacity of the truck.

#### ***Output Format***

The first line of output prints "One-third of Truck: X," where  $X$  is one-third of the truck's total weight capacity as a double value with two decimal places.

The second line of output displays one of the following:

1. If  $p$  is less than one-third of the truck's total weight capacity, print "Package weight is less than one-third of the truck's capacity".
2. Otherwise, print "Package weight is not less than one-third of the truck's capacity".

Refer to the sample output for the formatting specifications.

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

Input: 13  
60

Output: One-third of Truck: 20.00

Package weight is less than one-third of truck's capacity

**Answer**

// You are using Java

import java.util.\*;

```
class PackageCheck {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
  
        int p = sc.nextInt(); // package weight  
        int w = sc.nextInt(); // truck capacity  
  
        double oneThird = w / 3.0;  
  
        System.out.printf("One-third of Truck: %.2f\n", oneThird);  
  
        String result = (p < oneThird)  
            ? "Package weight is less than one-third of truck's capacity"  
            : "Package weight is not less than one-third of truck's capacity";  
  
        System.out.println(result);  
    }  
}
```

**Status : Correct**

**Marks : 10/10**



# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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

### 2028\_REC\_OOPS using Java\_Week 1\_CY

Attempt : 1

Total Mark : 40

Marks Obtained : 40

### Section 1 : Coding

#### 1. Problem Statement:

"Write a program that helps identify the type of a triangle based on the lengths of its three sides. The program prompts the user to input the lengths of sides 'a', 'b', and 'c', and then it classifies the triangle as 'Equilateral' if all sides are equal, 'Isosceles' if two sides are equal, or 'Scalene' if all sides are different. Can you provide the Java code for this task?"

#### ***Input Format***

The first line of the input is an integer 'a' representing the length of side 'a.'

The second line of the input is an integer 'b' representing the length of side 'b.'

The third line of the input is an integer 'c' representing the length of side 'c.'

### Output Format

The program outputs a single line that specifies the type of the triangle: "Equilateral," "Isosceles," or "Scalene."

### Sample Test Case

Input: 3

4

5

Output: The triangle is Scalene

### Answer

// You are using Java

import java.util.\*;

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

        int a = scanner.nextInt();
        int b = scanner.nextInt();
        int c = scanner.nextInt();

        if (a == b && b == c) {
            System.out.println("The triangle is Equilateral");
        } else if (a == b || b == c || a == c) {
            System.out.println("The triangle is Isosceles");
        } else {
            System.out.println("The triangle is Scalene");
        }
    }
}
```

Status : Correct

Marks : 10/10

## 2. Problem Statement

In the faraway land of Arithmetica, there exists an ancient calculator that can only perform bitwise operations. The calculator is locked with a secret code that only works when the number is modified using a special operation called right shifting.

The ruler of Arithmetica, King Thales, needs your help to unlock the calculator. The lock on the calculator is encoded with a number, and the calculator will only open if you apply a right shift by 2 on the number. Your task is to help King Thales determine the magic number that will unlock the ancient calculator.

### ***Input Format***

The first line of input represents an integer.

### ***Output Format***

The output should display the right-shifted value by 2 bits.

Refer to the sample output for formatting specifications.

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

Input: 16

Output: 4

### ***Answer***

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

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

```
        // Right shift the number by 2 bits
        int shifted = n >> 2;
```

```
        // Output the result
        System.out.println(shifted);
```

```
    }
}
```

**Status :** Correct

**Marks :** 10/10

### 3. Problem Statement:

Gilbert is tasked with writing a program that checks whether a given integer is an odd number. An odd number is one that cannot be exactly divided by 2. The program should take an integer as input and determine if it is an odd number or not. The task is to implement the logic to check if the provided integer is odd and return the result.

#### ***Input Format***

The first line of the input contains an integer, "input".

#### ***Output Format***

The output should display a boolean value, "result," which should be set to true if the input integer is an odd number and false if it is even.

Refer to the sample output for formatting specifications.

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

Input: 0

Output: Is the integer odd? false

#### ***Answer***

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

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

        boolean result = (input % 2 != 0);

        System.out.println("Is the integer odd? " + result);
    }
}
```

**Status :** Correct

**Marks :** 10/10

#### 4. Problem Statement

Mandy is working on a cybersecurity project that involves basic encryption techniques. She wants to write a program that takes an integer number and performs a bitwise XOR operation to flip all the bits.

Help Mandy in this encryption using bitwise operations.

##### ***Input Format***

The input consists of an integer N, representing the number to be flipped.

##### ***Output Format***

The output displays "Result: " followed by an integer representing the result of the bitwise XOR operation to flip all the bits.

Refer to the sample output for formatting specifications.

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

Input: 0

Output: Result: 255

##### ***Answer***

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

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

        int result = N ^ 255;

        System.out.println("Result: " + result);
    }
}
```

**Status : Correct**

**Marks : 10/10**

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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

### 2028\_REC\_OOPS using Java\_Week 2\_MCQ

Attempt : 1

Total Mark : 15

Marks Obtained : 10

#### Section 1 : MCQ

1. What will be the output of the following code?

```
class Main {  
    public static void main(String[] args) {  
        for (int i = 5; i > 0; i--) {  
            System.out.print(i + " ");  
        }  
    }  
}
```

**Answer**

5 4 3 2 1

**Status :** Correct

**Marks :** 1/1

2. What will be the output of the following code?

```
class ConditionTest {  
    public static void main(String[] args) {  
        int a = 7;  
        if (a == 7)  
            System.out.print("Match");  
        else  
            System.out.print("No Match");  
    }  
}
```

**Answer**

Match

**Status :** Correct

**Marks :** 1/1

3. What will be the output of the following Java code snippet?

```
public class Main {  
    public static void main(String[] args) {  
        int score = 75;  
        if(score >= 90) {  
            System.out.println("Grade: A");  
        } else if(score >= 80) {  
            System.out.println("Grade: B");  
        } else if(score >= 70) {  
            System.out.println("Grade: C");  
        } else {  
            System.out.println("Grade: D");  
        }  
    }  
}
```

**Answer**

Grade: B

**Status :** Wrong

**Marks :** 0/1

4. What will be the output of the following code?

```
class LoopTest {  
    public static void main(String[] args) {  
        int i = 1;  
        do {  
            System.out.print(i + " ");  
            i *= 2;  
        } while (i <= 8);  
    }  
}
```

**Answer**

1 2 4 8

**Status :** Correct

**Marks :** 1/1

5. What will be the output of the following code?

```
class Test {  
    public static void main(String[] args) {  
        int a = 4, b = 5;  
        if ((a + b) % 2 == 0)  
            System.out.print("Even");  
        else  
            System.out.print("Odd");  
    }  
}
```

**Answer**

Odd

**Status :** Correct

**Marks :** 1/1

6. What will be the output of the following code?

```
public class Main {  
    public static void main(String[] args) {  
        int i = 10;
```



```
do {  
    System.out.print(i + "");  
    i -= 3;  
} while(i > 0);  
}
```

**Answer**

10 7 4 1

**Status :** Correct

**Marks :** 1/1

7. What will be the output of the following code?

```
class Test {  
    public static void main(String[] args) {  
        int x = 5, y = 2;  
        if (x + y == 10)  
            System.out.print("Ten");  
        else if (x - y == 3)  
            System.out.print("Three");  
        else  
            System.out.print("None");  
    }  
}
```

**Answer**

Compilation error

**Status :** Wrong

**Marks :** 0/1

8. What will be the output of the following Java code snippet?

```
public class Main {  
    public static void main(String[] args) {  
        int day = 4;  
        String result = "";  
        switch(day) {  
            case 1:
```

```
        result = "Monday";
        break;
    case 2:
        result = "Tuesday";
        break;
    case 3:
        result = "Wednesday";
        break;
    default:
        result = "Other Day";
    }
    System.out.println(result);
}
```

**Answer**

Other Day

**Status :** Correct

**Marks :** 1/1

9. What will be the output of the following code?

```
public class Main {
    public static void main(String[] args) {
        int sum = 0;
        for(int i = 1; i <= 5; i++) {
            sum += i;
        }
        System.out.println(sum);
    }
}
```

**Answer**

15

**Status :** Correct

**Marks :** 1/1

10. What will be the output of the following code?

```
public class Main {  
    public static void main(String[] args) {  
        int i = 1;  
        while(i < 10) {  
            i += 2;  
        }  
        System.out.println(i);  
    }  
}
```

**Answer**

12

**Status : Wrong**

**Marks : 0/1**

11. What will be the output of the following code?

```
class LoopTest {  
    public static void main(String[] args) {  
        int i = 1;  
        while (i > 0) {  
            System.out.print(i + " ");  
            i++;  
            if (i == 5) break;  
        }  
    }  
}
```

**Answer**

1 2 3 4 5

**Status : Wrong**

**Marks : 0/1**

12. What will be the output of the following code?

```
class Test {  
    public static void main(String[] args) {  
        int num = 15;
```

```
    if (num > 10)
        if (num % 3 == 0)
            System.out.print("Divisible");
        else
            System.out.print("Not Divisible");
    }
}
```

**Answer**

Divisible

**Status :** Correct

**Marks :** 1/1

13. What will be the output of the following code?

```
public class Main {
    public static void main(String[] args) {
        for(int i = 1; i <= 20; i = i * 2) {
            System.out.print(i + " ");
        }
    }
}
```

**Answer**

1 2 4 8 16

**Status :** Correct

**Marks :** 1/1

14. What will be the output of the following code?

```
class ConditionTest {
    public static void main(String[] args) {
        int x = 10;
        if (x > 5)
            System.out.print("High");
    }
}
```

**Answer**

High

Status : Correct

Marks : 1/1

15. What will be the output of the following code?

```
class Loop {  
    public static void main(String[] args) {  
        for (int i = 1; i <= 3; i++) {  
            for (int j = 1; j <= 2; j++) {  
                System.out.print(i + "" + j + " ");  
            }  
        }  
    }  
}
```

Answer

11 12 21 22

Status : Wrong

Marks : 0/1

# Rajalakshmi Engineering College

Name: Sachin Krishna B  
Email: 241901507@rajalakshmi.edu.in  
Roll no: 241901507  
Phone: 9025753177  
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

### 2028\_REC\_OOPS using Java\_Week 2\_Q1

Attempt : 1  
Total Mark : 10  
Marks Obtained : 10

#### Section 1 : Coding

##### 1. Problem Statement

Arun is working on a project to automate the process of determining whether a student has passed or failed based on their subject marks.

He aims to create a simple program that takes positive integers as marks for five subjects from the user. If the average of the marks is greater than or equal to 50, the student has passed the exam. Otherwise, the student has failed.

Help Arun to implement the project.

##### ***Input Format***

The input consists of five space-separated integers, representing the marks in five subjects.

### **Output Format**

The first line of output prints "Average score: " followed by an integer representing the average score.

The second line prints one of the following:

1. If the condition is satisfied, print "The student has passed".
2. Otherwise, the output prints "The student has failed".

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 50 60 70 80 90

Output: Average score: 70

The student has passed

### **Answer**

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

```
public class Main {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
  
        // Read five integers from input  
        int[] marks = new int[5];  
        for (int i = 0; i < 5; i++) {  
            marks[i] = sc.nextInt();  
        }  
  
        // Calculate the average (integer division)  
        int sum = 0;  
        for (int mark : marks) {  
            sum += mark;  
        }  
        int average = sum / 5;  
  
        // Print the average score  
        System.out.println("Average score: " + average);  
    }  
}
```

```
// Print pass/fail based on the average
if (average >= 50) {
    System.out.println("The student has passed");
} else {
    System.out.println("The student has failed");
}

sc.close();
}
```

**Status :** Correct

**Marks :** 10/10



# Rajalakshmi Engineering College

Name: Sachin Krishna B  
Email: 241901507@rajalakshmi.edu.in  
Roll no: 241901507  
Phone: 9025753177  
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

### 2028\_REC\_OOPS using Java\_Week 2\_Q2

Attempt : 1  
Total Mark : 10  
Marks Obtained : 10

#### Section 1 : Coding

##### 1. Problem Statement

Samantha is a diligent math student who is exploring the world of programming. She is learning Java and has recently studied conditional statements. One day, her teacher gives her an interesting problem to solve, which takes a number as input and checks whether it is a multiple of 5 or 7.

Help her complete the task.

##### ***Input Format***

The input consists of a single integer N, representing the number to be checked.

##### ***Output Format***

If the number is a multiple of 5 but not 7, the output prints "N is a multiple of 5"

If the number is a multiple of 7, the output prints "N is a multiple of 7".

Otherwise the output prints "N is neither multiple of 5 nor 7" where N is an entered integer.

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 10

Output: 10 is a multiple of 5

### **Answer**

```
import java.util.Scanner;

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

        int N = sc.nextInt();

        if (N % 7 == 0) {
            System.out.println(N + " is a multiple of 7");
        } else if (N % 5 == 0) {
            System.out.println(N + " is a multiple of 5");
        } else {
            System.out.println(N + " is neither multiple of 5 nor 7");
        }

        sc.close();
    }
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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

### 2028\_REC\_OOPS using Java\_Week 2\_Q3

Attempt : 1

Total Mark : 10

Marks Obtained : 10

### Section 1 : Coding

#### 1. Problem Statement

John is a fitness trainer, and he wants to use the BMI calculator to assess the body mass index of his clients. He has a list of clients based on their height and weight.

John plans to write a program to quickly determine the BMI and provide a classification for each client.

If BMI is less than 18.5, the program will classify it as "Underweight" If BMI is between 18.6 and 24.9, the program will classify it as "Normal Weight" If BMI is between 25.0 and 29.9, the program will classify it as "Overweight" If BMI is 30.0 or higher, the program will classify it as "Obese"

Note: Formula to calculate BMI =  $\text{weight}/(\text{height} \times \text{height})$

**Input Format**

The first line of input consists of a double value, representing the height of the person in meters.

The second line consists of a double value, representing the weight of the person in kilograms.

### ***Output Format***

The first line of output prints "BMI: " followed by a double (rounded to two decimal places) representing the calculated BMI.

The second line prints "Classification: " followed by a string indicating the BMI category (Underweight, Normal Weight, Overweight, or Obese).

Refer to the sample output for formatting specifications.

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

Input: 1.2

45.2

Output: BMI: 31.39

Classification: Obese

### ***Answer***

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

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

```
        // Read inputs
        double height = sc.nextDouble();
        double weight = sc.nextDouble();
```

```
        // Calculate BMI
        double bmi = weight / (height * height);
```

```
        // Determine classification
        String classification;
        if (bmi < 18.5) {
```

```
        classification = "Underweight";
    } else if (bmi >= 18.5 && bmi <= 24.9) {
        classification = "Normal Weight";
    } else if (bmi >= 25.0 && bmi <= 29.9) {
        classification = "Overweight";
    } else {
        classification = "Obese";
    }

    // Print results with formatting
    System.out.printf("BMI: %.2f\n", bmi);
    System.out.println("Classification: " + classification);

    sc.close();
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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

### 2028\_REC\_OOPS using Java\_Week 2\_Q4

Attempt : 1

Total Mark : 10

Marks Obtained : 9

### Section 1 : Coding

#### 1. Problem Statement

Amit wants to evaluate the depreciation of his car over time to understand its current value and categorize it based on that value.

Write a program that helps him determine the current value of his car after a certain number of years of depreciation and classify it into one of three categories:

High: If the current value is greater than 10,000. Medium: If the current value is between 5,000 and 10,000, both inclusive. Low: If the current value is less than 5,000.

The depreciation rate of the car is 15% per year. The program should calculate the current value of the car after applying this depreciation over the given number of years and print the current value along with the category.

### ***Input Format***

The first line of input consists of an integer, representing the initial cost of the car.

The second line consists of an integer, representing the number of years the car has been depreciating.

### ***Output Format***

The first line of output prints a double value, representing the current value of the car, rounded off to two decimal places "Current Value: <value>".

The second line prints its category "Category: <categories>".

Refer to the sample output for formatting specifications.

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

Input: 20000  
5

Output: Current Value: 8874.11  
Category: Medium

### ***Answer***

```
import java.util.Scanner;

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

        int initialCost = sc.nextInt();
        int years = sc.nextInt();

        double rate = 0.15;
        double currentValue = initialCost * Math.pow(1 - rate, years);

        String category;
        if (currentValue > 10000) {
            category = "High";
        } else if (currentValue >= 5000 && currentValue <= 10000) {
```

```
        category = "Medium";  
    } else {  
        category = "Low";  
    }  
  
    System.out.printf("Current Value: %.2f\n", currentValue);  
    System.out.println("Category: " + category);  
  
    sc.close();  
    }  
}
```

**Status :** Partially correct

**Marks :** 9/10



# Rajalakshmi Engineering College

Name: Sachin Krishna B  
Email: 241901507@rajalakshmi.edu.in  
Roll no: 241901507  
Phone: 9025753177  
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

### 2028\_REC\_OOPS using Java\_Week 2\_Q5

Attempt : 1  
Total Mark : 10  
Marks Obtained : 10

#### Section 1 : Coding

##### 1. Problem Statement

Ted, the computer science enthusiast, has accepted the challenge of writing a program that checks if the number of digits in an integer matches the sum of its digits.

Guide Ted in designing and writing the code to solve this problem using a 'do-while' loop.

##### ***Input Format***

The input consists of an integer N, representing the number to be checked.

##### ***Output Format***

If the sum is equal to the number of digits, print "The number of digits in N matches the sum of its digits."

Else, print "The number of digits in N does not match the sum of its digits."

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 20

Output: The number of digits in 20 matches the sum of its digits.

### **Answer**

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

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

        int number = N;
        int sumOfDigits = 0;
        int digitCount = 0;

        // Use do-while loop to count digits and sum digits
        do {
            int digit = number % 10;
            sumOfDigits += digit;
            digitCount++;
            number /= 10;
        } while (number > 0);

        if (sumOfDigits == digitCount) {
            System.out.println("The number of digits in " + N + " matches the sum of
its digits.");
        } else {
            System.out.println("The number of digits in " + N + " does not match the
sum of its digits.");
        }

        sc.close();
    }
}
```

}  
}  
**Status : Correct**

**Marks : 10/10**

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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

### 2028\_REC\_OOPS using Java\_Week 2\_Q6

Attempt : 1

Total Mark : 10

Marks Obtained : 10

### Section 1 : Coding

#### 1. Problem Statement

Maya, a student in an arts and crafts class, wants to create a pattern using stars (\*) in a specific format. She plans to use a program to help her construct the pattern.

Write a program that takes an integer as input and constructs the following pattern using nested for loops.

Input: 5

Output:

\*

\* \*

```
* * *
* * * *
* * * * *
* * * *
* * *
* *
*
*
```

### ***Input Format***

The input consists of a number (integer) representing the number of rows.

### ***Output Format***

The output displays the required pattern.

Refer to the sample output for the formatting specifications.

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

Input: 5

Output: \*

```
* *
* * *
* * * *
* * * * *
* * * *
* * *
* *
*
*
```

### ***Answer***

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

```
public class Main {
    public static void main(String[] args) {
```

```
Scanner sc = new Scanner(System.in);
int rows = sc.nextInt();

// Print increasing stars from 1 to rows
for (int i = 1; i <= rows; i++) {
    for (int j = 1; j <= i; j++) {
        System.out.print("* ");
    }
    System.out.println();
}

// Print decreasing stars from rows-1 down to 1
for (int i = rows - 1; i >= 1; i--) {
    for (int j = 1; j <= i; j++) {
        System.out.print("* ");
    }
    System.out.println();
}

sc.close();
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B  
Email: 241901507@rajalakshmi.edu.in  
Roll no: 241901507  
Phone: 9025753177  
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

### 2028\_REC\_OOPS using Java\_Week 2\_Q7

Attempt : 1  
Total Mark : 10  
Marks Obtained : 10

#### Section 1 : Coding

##### 1. Problem Statement

You are taking part in a coding challenge where your task is to design a program that conjures a mesmerizing numerical pyramid pattern. The enchanting pattern is fashioned using a for loop and is customized based on user input.

Participants are prompted to unveil the pyramid's magic by specifying its height - essentially dictating the number of rows in this spellbinding creation.

Write a program that employs to weave this captivating numerical pyramid as shown below.

Example

Input:

4

Output:

### ***Input Format***

The input consists of a positive integer n representing the number of rows in the pattern.

### ***Output Format***

The output prints the required pyramid pattern, as shown in the sample output.

Refer to the sample output for the formatting specifications.

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

Input: 4

Output: 1

123

12345

1234567

### ***Answer***

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

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

        for (int i = 1; i <= n; i++) {
            // Print spaces to center the pyramid
            for (int space = 1; space <= n - i; space++) {
                System.out.print(" ");
            }
        }
    }
}
```



```
// Calculate number of digits in this row: 2*i - 1
int digitsCount = 2 * i - 1;

// Print numbers starting from 1 to digitsCount
for (int num = 1; num <= digitsCount; num++) {
    System.out.print(num);
}

System.out.println();
}

sc.close();
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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

### 2028\_REC\_OOPS using Java\_Week 2\_Q8

Attempt : 1

Total Mark : 10

Marks Obtained : 10

### Section 1 : Coding

#### 1. Problem Statement

A bank generates secure codes using 3-digit numbers where each digit is unique, and the code must be divisible by 3. You are tasked with generating the first N such codes based on user input, ensuring the digits are unique and the number is divisible by 3.

Note: Use nested for loops to solve.

#### ***Input Format***

The first line contains an integer N representing the number of valid codes to generate.

#### ***Output Format***

The output prints N lines, each line contains a valid 3-digit code.

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 5

Output: 102

105

108

120

123

### **Answer**

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

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

        int count = 0;

        // Iterate through all 3-digit numbers
        for (int i = 100; i <= 999 && count < N; i++) {
            int hundreds = i / 100;
            int tens = (i / 10) % 10;
            int ones = i % 10;

            // Check if digits are unique
            if (hundreds != tens && tens != ones && hundreds != ones) {
                // Check divisibility by 3
                if (i % 3 == 0) {
                    System.out.println(i);
                    count++;
                }
            }
        }

        sc.close();
    }
}
```

}  
}  
**Status : Correct**

**Marks : 10/10**

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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

### 2028\_REC\_OOPS using Java\_Week 2\_PAH

Attempt : 1

Total Mark : 40

Marks Obtained : 40

### Section 1 : Coding

#### 1. Problem Statement

Rohit is tasked with designing a program to analyze the digits of a given integer.

Write a program to help Rohit that takes an integer as input and identifies the minimum odd digit and the maximum even digit present in the number. If no odd or even digits are present, display appropriate messages.

Implement the solution using a 'while' loop to iterate through the digits of the given number.

#### ***Input Format***

The input consists of an integer n.

#### ***Output Format***

The first line of output prints the message "Minimum odd digit: " followed by an integer representing the smallest odd digit found in the input number.

If no odd digit exists, it prints "There are no odd digits in the number."

The second line of output prints the message "Maximum even digit: " followed by an integer representing the largest even digit found in the input number.

If no even digit exists, it prints "There are no even digits in the number."

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 3465

Output: Minimum odd digit: 3

Maximum even digit: 6

### **Answer**

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

```
public class Main {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        int n = sc.nextInt();  
        sc.close();  
  
        int minOdd = 9; // maximum possible odd digit  
        int maxEven = -1; // minimum possible even digit  
        boolean oddFound = false;  
        boolean evenFound = false;  
  
        while (n > 0) {  
            int digit = n % 10;  
            if (digit % 2 == 0) { // even  
                evenFound = true;  
                if (digit > maxEven) {  
                    maxEven = digit;  
                }  
            } else { // odd
```

```

        oddFound = true;
        if (digit < minOdd) {
            minOdd = digit;
        }
    }
    n /= 10;
}

if (oddFound) {
    System.out.println("Minimum odd digit: " + minOdd);
} else {
    System.out.println("There are no odd digits in the number.");
}

if (evenFound) {
    System.out.println("Maximum even digit: " + maxEven);
} else {
    System.out.println("There are no even digits in the number.");
}
}
}

```

**Status :** Correct

**Marks :** 10/10

## 2. Problem Statement

You are given a number of distribution centers (rows) and are tasked with generating a zigzag shipment route pattern. Each shipment route should alternate between left-to-right and right-to-left, as described below.

The program should print the zigzag pattern with a tab (\t) separating the columns. For each row, the shipment numbers should follow a diagonal pattern where numbers are placed in a zigzag, left to right on odd rows and right to left on even rows.

### **Input Format**

The input consists of an integer N, which represents the number of distribution centers (rows) for the zigzag pattern.

### Output Format

The output prints the zigzag pattern with N rows, formatted with a tab space (\t) separating the columns.

Refer to the sample output for formatting specifications.

### Sample Test Case

Input: 5

Output:     1  
          2  6  
         3  7  10  
         4  8  11  13  
         5  9  12  14  15

### Answer

```
import java.util.Scanner;

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

        for (int i = 0; i < rows; i++) {
            for (int j = 0; j <= i; j++) {
                int value = ((2 * rows - 1 - j) * j) / 2 + (i + 1);
                System.out.print(value + "\t");
            }
            System.out.println();
        }
    }
}
```

Status : Correct

Marks : 10/10

### 3. Problem Statement



Ravi wants to estimate the total utility bill for a household based on the consumption of electricity, water, and gas.

Write a program to calculate the total bill using the following criteria:

The cost per unit for electricity is 0.12, for water is 0.05, and for gas is 0.08. A discount is applied to the total cost based on the following conditions: If the total cost is 100 or more, a 10% discount is applied. If the total cost is between 50 and 99.99, a 5% discount is applied. No discount is applied if the total cost is less than 50.

The program should output the total bill after applying the discount with two decimal places.

#### ***Input Format***

The input consists of three double values, representing the number of units consumed for electricity, water, and gas respectively.

#### ***Output Format***

The output prints a double value, representing the total bill after applying the discount, formatted to two decimal places.

Refer to the sample output for formatting specifications.

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

Input: 1000.0  
200.0  
100.0

Output: 124.20

#### ***Answer***

```
import java.util.Scanner;

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

        double electricityUnits = sc.nextDouble();
```

```

double waterUnits = sc.nextDouble();
double gasUnits = sc.nextDouble();
sc.close();

// Rates
double electricityRate = 0.12;
double waterRate = 0.05;
double gasRate = 0.08;

// Calculate total cost before discount
double totalCost = (electricityUnits * electricityRate) +
    (waterUnits * waterRate) +
    (gasUnits * gasRate);

// Apply discount
if (totalCost >= 100) {
    totalCost -= totalCost * 0.10; // 10% discount
} else if (totalCost >= 50) {
    totalCost -= totalCost * 0.05; // 5% discount
}

// Output with two decimal places
System.out.printf("%.2f", totalCost);
}
}

```

**Status :** Correct

**Marks :** 10/10

#### 4. Problem Statement

Sampad is a high school teacher who wants to convert numeric grades into letter grades.

Write a program that accepts a numeric grade as input. The program should then convert this numeric grade into a letter grade based on specific conditions. The letter grades are A, B, C, D and F.

The conversion is determined by the following conditions:

If the numeric grade is 90 or higher, it's an "A" If the numeric grade is between 80 and 89 (inclusive), it's a "B" If the numeric grade is between 70

and 79 (inclusive), it's a "C" If the numeric grade is between 60 and 69 (inclusive), it's a "D" If the numeric grade is below 60, it's an "F"

### **Input Format**

The input consists of an integer representing the numeric grade of the student.

### **Output Format**

The output prints the letter grade corresponding to the input numeric grade as "Letter Grade: <grade>".

Refer to the sample output for the formatting specifications.

### **Sample Test Case**

Input: 85

Output: Letter Grade: B

### **Answer**

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

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

        String letterGrade;

        if (grade >= 90) {
            letterGrade = "A";
        } else if (grade >= 80) {
            letterGrade = "B";
        } else if (grade >= 70) {
            letterGrade = "C";
        } else if (grade >= 60) {
            letterGrade = "D";
        } else {
            letterGrade = "F";
        }
    }
}
```

```
}  
    System.out.println("Letter Grade: " + letterGrade);  
}  
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B  
Email: 241901507@rajalakshmi.edu.in  
Roll no: 241901507  
Phone: 9025753177  
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

### 2028\_REC\_OOPS using Java\_Week 2\_CY

Attempt : 1  
Total Mark : 40  
Marks Obtained : 40

#### Section 1 : Coding

##### 1. Problem Statement

Ram wants to evaluate the time required to break even on an investment based on initial costs, monthly profits, and monthly expenses. Write a program to calculate the break-even point in months and categorize the return on investment.

Compute the break-even point by using the formula:  $\text{initial cost} / (\text{monthly profit} - \text{monthly expenses})$ . Based on the break-even point, classify the return on investment into one of the following categories: Quick Return: If the break-even point is 3 months or fewer. Average Return: If the break-even point is between 4 and 12 months, inclusive. Long-term Return: If the break-even point exceeds 12 months.

Ram is new to programming, so he seeks your assistance in creating the program.

Note: monthly profit is always greater than monthly expenses.

### ***Input Format***

The first line of input consists of a double value representing the initial cost.

The second line consists of a double value representing the monthly profit.

The third line consists of a double value representing the monthly expenses.

### ***Output Format***

The first line prints "Break-even Point:", followed by the break-even point as a decimal number (of double datatype), formatted to two decimal places.

The second line prints "Category: ", followed by the investment return as a String, which can be one of:

- "Quick Return" if break-even point  $\leq 3$
- "Average Return" if break-even point  $\leq 12$
- "Long-term Return" if break-even point  $> 12$

Refer to the sample output for formatting specifications.

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

Input: 10000.50  
5000.75  
1000.10

Output: Break-even Point: 2.50  
Category: Quick Return

### ***Answer***

```
import java.util.Scanner;

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

        double initialCost = sc.nextDouble();
```

```

double monthlyProfit = sc.nextDouble();
double monthlyExpenses = sc.nextDouble();
sc.close();

// Calculate break-even point
double breakEvenPoint = initialCost / (monthlyProfit - monthlyExpenses);

// Print break-even point with two decimal places
System.out.printf("Break-even Point: %.2f%n", breakEvenPoint);

// Determine category
String category;
if (breakEvenPoint <= 3) {
    category = "Quick Return";
} else if (breakEvenPoint <= 12) {
    category = "Average Return";
} else {
    category = "Long-term Return";
}

// Output category
System.out.println("Category: " + category);
}
}

```

**Status :** Correct

**Marks :** 10/10

## 2. Problem Statement

Raj is solving a physics problem involving projectile motion, where he needs to calculate the time a ball hits the ground using a quadratic equation of the form  $ax^2 + bx + c = 0$ . Depending on the coefficients, the ball may hit the ground once, twice, or not at all in real time.

Help Raj find all real roots of the equation, if any.

Note: discriminant =  $b^2 - 4ac$

### **Input Format**

The input consists of three space-separated doubles a, b, and c, representing the

coefficients of the quadratic equation.

### **Output Format**

If there are two real roots, print:

- "Two real solutions:"
- "Root1 = <value>"
- "Root2 = <value>"

If there is one real root, print:

- "One real solution:"
- "Root = <value>"

If there are no real roots, print:

- "There are no real solutions."

Note: values are rounded to two decimal places.

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 1 6 9

Output: One real solution:

Root = -3.00

### **Answer**

// You are using Java

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

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

```
        double a = sc.nextDouble();  
        double b = sc.nextDouble();  
        double c = sc.nextDouble();  
        sc.close();
```



```

double discriminant = b * b - 4 * a * c;

if (discriminant > 0) {
    double root1 = (-b + Math.sqrt(discriminant)) / (2 * a);
    double root2 = (-b - Math.sqrt(discriminant)) / (2 * a);
    System.out.println("Two real solutions:");
    System.out.printf("Root1 = %.2f%n", root1);
    System.out.printf("Root2 = %.2f%n", root2);
} else if (discriminant == 0) {
    double root = -b / (2 * a);
    System.out.println("One real solution:");
    System.out.printf("Root = %.2f%n", root);
} else {
    System.out.println("There are no real solutions.");
}
}
}
}

```

**Status :** Correct

**Marks :** 10/10

### 3. Problem Statement

Noah is analyzing numbers within a given range [A, B] and wants to calculate a special sum. For each number in the range, he calculates the product of its odd digits (ignoring even digits). If the number contains no odd digits, it is skipped. The sum of these products for all numbers in the range is the result.

Write a program to compute this sum.

Example

Input:

10 12

Output:

3

Explanation:

For 10, odd digits = 1, product = 1.

For 11, odd digits = 1, 1, product =  $1 * 1 = 1$ .

For 12, odd digits = 1, product = 1.

Total sum =  $1 + 1 + 1 = 3$

### ***Input Format***

The input consists of two space-separated integers A and B, representing the inclusive range boundaries.

### ***Output Format***

The output prints a single integer representing the sum of the products of odd digits for all numbers in the range.

Refer to the sample output for the formatting specifications.

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

Input: 10 12

Output: 3

### ***Answer***

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

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

        int A = sc.nextInt();
        int B = sc.nextInt();
        sc.close();

        int totalSum = 0;

        for (int num = A; num <= B; num++) {
            int temp = num;
            int product = 1;
```

```
boolean hasOdd = false;

while (temp > 0) {
    int digit = temp % 10;
    if (digit % 2 != 0) { // odd digit
        hasOdd = true;
        product *= digit;
    }
    temp /= 10;
}

if (hasOdd) {
    totalSum += product;
}

System.out.println(totalSum);
}
```

**Status :** Correct

**Marks : 10/10**

#### 4. Problem Statement

Joe has a favourite number, let's call it X. He wants to check if X is divisible by the sum of its digits. If it is, he considers it a lucky number. If not, he wants to find the closest smaller number, that is divisible by the sum of digits of X. Joe has challenged his friends to solve this puzzle at his birthday party.

Example

Input:

157

Output:

157 is not divisible by the sum of its digits.

The closest smaller number that is divisible: 156

Explanation:

The sum of the digits of X is  $1+5+7=13$ . Since 157 is not divisible by 13, we need to find the closest smaller number that is divisible by 13. 156 is divisible by 13, it is the closest smaller number that meets the requirement.

### ***Input Format***

The input consists of an integer X, representing Joe's favourite number.

### ***Output Format***

If X is a lucky number, then the output must be in the format: "X is divisible by the sum of its digits."

If not, then the output must be in the format:

"X is not divisible by the sum of its digits."

The closest smaller number that is divisible: Y",

where X is the entered number and Y is the closest number.

Refer to the sample output for formatting specifications.

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

Input: 120

Output: 120 is divisible by the sum of its digits.

### ***Answer***

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

```
public class Main {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        int X = sc.nextInt();  
        sc.close();  
  
        int sumOfDigits = 0;  
        int temp = X;
```

```
while (temp > 0) {
    sumOfDigits += temp % 10;
    temp /= 10;
}

if (X % sumOfDigits == 0) {
    System.out.println(X + " is divisible by the sum of its digits.");
} else {
    System.out.println(X + " is not divisible by the sum of its digits.");
    // find closest smaller number divisible by sumOfDigits
    int closest = X - 1;
    while (closest > 0 && closest % sumOfDigits != 0) {
        closest--;
    }
    System.out.println("The closest smaller number that is divisible: " +
closest);
}
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B  
Email: 241901507@rajalakshmi.edu.in  
Roll no: 241901507  
Phone: 9025753177  
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 3\_MCQ

Attempt : 1  
Total Mark : 15  
Marks Obtained : 14

#### Section 1 : MCQ

1. What will be the output of the following code?

```
public class Test {  
    public static void main(String[] args) {  
        int[] x = {4, 8, 12};  
        int result = x[0] * x[2];  
        System.out.println(result);  
    }  
}
```

**Answer**

48

**Status :** Correct

**Marks :** 1/1

2. What will be the output of the following code?

```
class Q {  
    public static void main(String[] args) {  
        int[] nums = {3, 6, 7, 2, 8};  
        int sum = 0;  
        for (int i = 0; i < nums.length; i++) {  
            if (nums[i] % 2 == 0)  
                sum += nums[i];  
        }  
        System.out.println(sum);  
    }  
}
```

**Answer**

16

**Status :** Correct

**Marks :** 1/1

3. What will be the output of the following code?

```
class Q {  
    public static void main(String[] args) {  
        int[] nums = {4, 2, 9, 5};  
        int max = nums[0];  
        for (int i = 1; i < nums.length; i++) {  
            if (nums[i] > max)  
                max = nums[i];  
        }  
        System.out.println(max);  
    }  
}
```

**Answer**

9

**Status :** Correct

**Marks :** 1/1

4. What will be the output of the given code?

```

public class Main {
    public static void main(String[] args) {
        int[] arr = {1, 2, 3, 4, 5};
        int n = arr.length;
        int temp = arr[0];

        for (int i = 0; i < n - 1; i++) {
            arr[i] = arr[i + 1];
        }
        arr[n - 1] = temp;

        for (int num : arr) {
            System.out.print(num + " ");
        }
    }
}

```

**Answer**

2 3 4 5 1

**Status :** Correct

**Marks :** 1/1

5. What will be the output of the following code?

```

class Q {
    public static void main(String[] args) {
        int[] a = {1, 2, 3, 4};
        for (int i = 0; i < a.length / 2; i++) {
            int temp = a[i];
            a[i] = a[a.length - 1 - i];
            a[a.length - 1 - i] = temp;
        }
        System.out.println(a[0]);
    }
}

```

**Answer**

1



Status : Wrong

Marks : 0/1

6. What will be the output of the following code?

```
class M {  
    public static void main(String[] args) {  
        int[][] arr = {  
            {1, 2},  
            {3, 4},  
            {5, 6}  
        };  
        for (int i = 0; i < arr.length; i++) {  
            System.out.print(arr[i][0] + " ");  
        }  
    }  
}
```

Answer

1 3 5

Status : Correct

Marks : 1/1

7. What will be the output of the following code?

```
class Sample {  
    public static void main(String[] args) {  
        int[] a = {1, 2, 3};  
        int product = 1;  
        for (int i = 0; i < a.length; i++) {  
            product *= a[i];  
        }  
        System.out.println(product);  
    }  
}
```

Answer

6

Status : Correct

Marks : 1/1

8. What will be the output of the following code?

```
class ReverseArray {  
    public static void main(String[] args) {  
        int[] a = {1, 2, 3, 4};  
        for (int i = 0; i < a.length / 2; i++) {  
            int temp = a[i];  
            a[i] = a[a.length - 1 - i];  
            a[a.length - 1 - i] = temp;  
        }  
        for (int i : a)  
            System.out.print(i + " ");  
    }  
}
```

Answer

4 3 2 1

Status : Correct

Marks : 1/1

9. What will be the output of the following code?

```
class Q {  
    public static void main(String[] args) {  
        int[] a = {1, 2, 3, 4};  
        for (int i = 0; i < a.length; i++) {  
            if (a[i] % 2 == 0)  
                a[i] = 0;  
        }  
        System.out.println(a[1] + " " + a[3]);  
    }  
}
```

Answer

0 0

Status : Correct

Marks : 1/1

10. What will be the output of the following code?

```
class Sample {  
    public static void main(String[] args) {  
        int[][] data = {  
            {1, 2},  
            {3, 4}  
        };  
        int sum = 0;  
  
        for (int[] row : data) {  
            for (int val : row) {  
                sum += val;  
            }  
        }  
  
        System.out.println("Sum = " + sum);  
    }  
}
```

**Answer**

Sum = 10

**Status :** Correct

**Marks :** 1/1

11. What will be the output of the following code?

```
class Q {  
    public static void main(String[] args) {  
        int[][] a = {  
            {1, 2},  
            {3, 4}  
        };  
        for (int i = 0; i < a.length; i++) {  
            for (int j = 0; j < a[0].length; j++) {  
                System.out.print(a[i][j] + " ");  
            }  
        }  
    }  
}
```

```
}
```

**Answer**

1 2 3 4

**Status :** Correct

**Marks :** 1/1

12. What will be the output of the following code?

```
class Q {  
    public static void main(String[] args) {  
        int[][] arr = {  
            {5, 6, 7},  
            {8, 9, 10}  
        };  
        System.out.println(arr[0][2]);  
    }  
}
```

**Answer**

7

**Status :** Correct

**Marks :** 1/1

13. What will be the output of the following code?

```
class Q {  
    public static void main(String[] args) {  
        int[] a = {1, 2, 1, 3, 1, 4};  
        int count = 0;  
        for (int i = 0; i < a.length; i++) {  
            if (a[i] == 1) count++;  
        }  
        System.out.println(count);  
    }  
}
```

**Answer**

3

**Status :** Correct

**Marks :** 1/1

14. What will be the output of the following code?

```
class Q {  
    public static void main(String[] args) {  
        int[][] a = {  
            {1, 2},  
            {3, 4}  
        };  
        int sum = 0;  
        for (int i = 0; i < a.length; i++)  
            for (int j = 0; j < a[0].length; j++)  
                sum += a[i][j];  
        System.out.println(sum);  
    }  
}
```

**Answer**

10

**Status :** Correct

**Marks :** 1/1

15. What will be the output of the following code?

```
class Sample {  
    public static void main(String[] args) {  
        int[][] matrix = {  
            {1, 2, 3},  
            {4, 5, 6}  
        };  
        System.out.println(matrix[1][2]);  
    }  
}
```

**Answer**

6

**Status :** Correct

**Marks :** 1/1

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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

### 2028\_REC\_OOPS using Java\_Week 3\_Q1

Attempt : 1

Total Mark : 10

Marks Obtained : 10

### Section 1 : Coding

#### 1. Problem Statement

Rosh is intrigued by numerical patterns. Today, she stumbled upon a puzzle while working with arrays. She wants to compute the sum of the third-largest and second-smallest elements from a list of integers. She seeks your help to implement a program that solves this for her efficiently.

#### ***Input Format***

The first line of input is an integer N, representing the size of the array.

The second line of input consists of N space-separated integers, representing the elements of the array.

#### ***Output Format***

The output displays a single integer representing the sum of the third-largest and second-smallest elements in the array.

Refer to the sample output for the formatting specifications.

### **Sample Test Case**

Input: 10

10 20 30 40 50 60 70 80 90 100

Output: 100

### **Answer**

```
import java.util.*;

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

        // Read size of array
        int N = sc.nextInt();
        int[] arr = new int[N];

        // Read elements
        for (int i = 0; i < N; i++) {
            arr[i] = sc.nextInt();
        }

        // Sort array
        Arrays.sort(arr);

        // Find second smallest and third largest
        int secondSmallest = arr[1];
        int thirdLargest = arr[N - 3];

        // Compute sum
        int result = secondSmallest + thirdLargest;

        // Print result
        System.out.println(result);
    }
}
```

Status : Correct

Marks : 10/10



# Rajalakshmi Engineering College

Name: Sachin Krishna B  
Email: 241901507@rajalakshmi.edu.in  
Roll no: 241901507  
Phone: 9025753177  
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

### 2028\_REC\_OOPS using Java\_Week 3\_Q2

Attempt : 1  
Total Mark : 10  
Marks Obtained : 10

#### Section 1 : Coding

##### 1. Problem Statement

Monica is interested in finding a treasure but the key to opening is to get the sum of the main diagonal elements and secondary diagonal elements.

Write a program to help Monica find the diagonal sum of a square 2D array.

Note: The main diagonal of the array consists of the elements traversing from the top-left corner to the bottom-right corner. The secondary diagonal includes elements from the top-right corner to the bottom-left corner.

##### ***Input Format***

The first line of input consists of an integer N, representing the number of rows and columns.

The following N lines consist of N space-separated integers, representing the 2D array elements.

### **Output Format**

The first line of output prints "Sum of the main diagonal: " followed by an integer, representing the sum of the main diagonal.

The second line prints "Sum of the secondary diagonal: " followed by an integer, representing the sum of the secondary diagonal.

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 3

1 2 3

4 5 6

7 8 9

Output: Sum of the main diagonal: 15

Sum of the secondary diagonal: 15

### **Answer**

// You are using Java

import java.util.\*;

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

```
        // Read size of the matrix  
        int N = sc.nextInt();  
        int[][] matrix = new int[N][N];
```

```
        // Read matrix elements  
        for (int i = 0; i < N; i++) {  
            for (int j = 0; j < N; j++) {  
                matrix[i][j] = sc.nextInt();  
            }  
        }  
    }  
}
```

```
int mainDiagonalSum = 0;
int secondaryDiagonalSum = 0;

// Calculate sums
for (int i = 0; i < N; i++) {
    mainDiagonalSum += matrix[i][i];
    secondaryDiagonalSum += matrix[i][N - 1 - i];
}

// Print results
System.out.println("Sum of the main diagonal: " + mainDiagonalSum);
System.out.println("Sum of the secondary diagonal: " +
secondaryDiagonalSum);
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B  
Email: 241901507@rajalakshmi.edu.in  
Roll no: 241901507  
Phone: 9025753177  
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

### 2028\_REC\_OOPS using Java\_Week 3\_Q3

Attempt : 1  
Total Mark : 10  
Marks Obtained : 10

#### Section 1 : Coding

##### 1. Problem Statement

You are developing a warehouse management system for a shipping company. The system uses an integer array to represent the weights of packages in a specific order. To verify that the weight capacity is not exceeded, the program needs to calculate the sum of the weights of the first and last packages in the list.

Task:

Write a code to calculate the sum of the weights of the first and last packages in the list. The program should take an integer array as input and return the total weight of the first and last packages.

##### ***Input Format***

The first line of the input is an integer N representing the size of the array.

The second line of the input is N space-separated integer values.

### **Output Format**

The output is displayed in the following format:

"Sum of the first and last elements: <<Sum>>"

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 5

10 20 30 40 50

Output: Sum of the first and last elements: 60

### **Answer**

```
import java.util.*;
```

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

```
        // Read size of array  
        int N = sc.nextInt();  
        int[] arr = new int[N];
```

```
        // Read array elements  
        for (int i = 0; i < N; i++) {  
            arr[i] = sc.nextInt();  
        }
```

```
        // Sum of first and last element  
        int sum = arr[0] + arr[N - 1];
```

```
        // Print result  
        System.out.println("Sum of the first and last elements: " + sum);
```

```
    }  
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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

### 2028\_REC\_OOPS using Java\_Week 3\_Q4

Attempt : 1

Total Mark : 10

Marks Obtained : 10

### Section 1 : Coding

#### 1. Problem Statement

Sesha is developing a weather monitoring system for a region with multiple weather stations. Each weather station collects temperature data hourly and stores it in a 2D array.

Write a program that can add the temperature data from two different weather stations to create a combined temperature record for the region.

#### ***Input Format***

The first line of input consists of two space-separated integers N and M, representing the number of rows and columns of the matrices, respectively.

The next N lines consist of M space-separated integers, representing the values of the first matrix.

The following N lines consist of M space-separated integers, representing the values of the second matrix.

### **Output Format**

The output prints the addition of the two matrices in N rows and M columns, representing the combined temperature record.

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 3 3

1 2 3

4 5 6

7 8 9

1 1 1

2 2 2

3 3 3

Output: 2 3 4

6 7 8

10 11 12

### **Answer**

```
import java.util.*;
```

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

```
        // Read dimensions  
        int N = sc.nextInt();  
        int M = sc.nextInt();
```

```
        int[][] mat1 = new int[N][M];  
        int[][] mat2 = new int[N][M];  
        int[][] result = new int[N][M];
```

```
        // Read first matrix  
        for (int i = 0; i < N; i++) {  
            for (int j = 0; j < M; j++) {
```



```

        mat1[i][j] = sc.nextInt();
    }
}

// Read second matrix
for (int i = 0; i < N; i++) {
    for (int j = 0; j < M; j++) {
        mat2[i][j] = sc.nextInt();
    }
}

// Perform addition
for (int i = 0; i < N; i++) {
    for (int j = 0; j < M; j++) {
        result[i][j] = mat1[i][j] + mat2[i][j];
    }
}

// Print result
for (int i = 0; i < N; i++) {
    for (int j = 0; j < M; j++) {
        System.out.print(result[i][j] + " ");
    }
    System.out.println();
}
}
}

```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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

### 2028\_REC\_OOPS using Java\_Week 3\_Q5

Attempt : 1

Total Mark : 10

Marks Obtained : 10

### Section 1 : Coding

#### 1. Problem Statement

Sharon is creating a program that finds the first repeated element in an integer array. The program should efficiently identify the first element that appears more than once in the given array. If no such element is found, it should appropriately display a message.

Help Sharon to complete the program.

#### ***Input Format***

The first line of input consists of an integer  $n$ , representing the number of elements in the array.

The second line consists of  $n$  space-separated integers, representing the array elements.

### **Output Format**

If a repeated element is found, print the first element that appears more than once.

If no repeated element is found, print "No repeated element found in the array".

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 8

12 21 13 14 21 36 47 21

Output: 21

### **Answer**

```
import java.util.*;
```

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

```
        // Read size of array  
        int n = sc.nextInt();  
        int[] arr = new int[n];
```

```
        // Read array elements  
        for (int i = 0; i < n; i++) {  
            arr[i] = sc.nextInt();  
        }
```

```
        // Use HashSet to track seen elements  
        Set<Integer> seen = new HashSet<>();  
        boolean found = false;
```

```
        for (int num : arr) {  
            if (seen.contains(num)) {  
                System.out.println(num);  
                found = true;  
                break; // stop at the first repeated element
```

```
    }  
    seen.add(num);  
  }  
  
  if (!found) {  
    System.out.println("No repeated element found in the array");  
  }  
}  
}
```

**Status :** Correct

**Marks : 10/10**

# Rajalakshmi Engineering College

Name: Sachin Krishna B  
Email: 241901507@rajalakshmi.edu.in  
Roll no: 241901507  
Phone: 9025753177  
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 3\_PAH

Attempt : 1  
Total Mark : 40  
Marks Obtained : 40

#### Section 1 : Coding

##### 1. Problem Statement

Priya is building a system to automate image transformations using matrix operations. To do this, she needs to multiply two matrices representing pixel data and transformation rules.

Help Priya perform matrix multiplication and print the resulting matrix if the operation is valid.

##### ***Input Format***

The first line of input consists of two int values, representing the number of rows R1 and columns C1 of the first matrix.

The next R1 × C1 integers represent the elements of the first matrix.

The next line consists of two int values, representing the number of rows R2 and

columns C2 of the second matrix.

The next  $R2 \times C2$  integers represent the elements of the second matrix.

### **Output Format**

If matrix multiplication is possible, print R1 lines, each containing C2 space-separated int values representing the resulting matrix.

Otherwise, print "Matrix multiplication not possible".

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 2 3

1 2 3

4 5 6

3 2

7 8

9 10

11 12

Output: 58 64

139 154

### **Answer**

```
import java.util.*;
```

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

```
        // Read first matrix dimensions  
        int R1 = sc.nextInt();  
        int C1 = sc.nextInt();  
        int[][] A = new int[R1][C1];
```

```
        // Read first matrix  
        for (int i = 0; i < R1; i++) {  
            for (int j = 0; j < C1; j++) {  
                A[i][j] = sc.nextInt();
```

```
}  
}
```

```
// Read second matrix dimensions
```

```
int R2 = sc.nextInt();
```

```
int C2 = sc.nextInt();
```

```
int[][] B = new int[R2][C2];
```

```
// Read second matrix
```

```
for (int i = 0; i < R2; i++) {
```

```
    for (int j = 0; j < C2; j++) {
```

```
        B[i][j] = sc.nextInt();
```

```
    }
```

```
}
```

```
// Check multiplication possibility
```

```
if (C1 != R2) {
```

```
    System.out.println("Matrix multiplication not possible");
```

```
    return;
```

```
}
```

```
// Result matrix
```

```
int[][] result = new int[R1][C2];
```

```
// Matrix multiplication
```

```
for (int i = 0; i < R1; i++) {
```

```
    for (int j = 0; j < C2; j++) {
```

```
        for (int k = 0; k < C1; k++) {
```

```
            result[i][j] += A[i][k] * B[k][j];
```

```
        }
```

```
    }
```

```
}
```

```
// Print result
```

```
for (int i = 0; i < R1; i++) {
```

```
    for (int j = 0; j < C2; j++) {
```

```
        System.out.print(result[i][j] + " ");
```

```
    }
```

```
    System.out.println();
```

```
}
```

```
}
```

```
}
```

Status : Correct

Marks : 10/10

## 2. Problem Statement

Egath is participating in a coding hackathon, and one of the challenges requires him to work with an array of integers. The task is to remove exactly one element from the array such that the sum of the remaining elements is a prime number.

Help Egath find the first possible prime sum by removing one element or determining if no such prime sum can be achieved.

### **Input Format**

The first line of input consists of an integer N, representing the number of elements in the array.

The second line consists of N space-separated integers, representing the array elements.

### **Output Format**

If removing one element results in a prime sum, print the sum.

If no such prime sum can be achieved by removing exactly one element, print "No valid prime sum found".

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 3

1 2 3

Output: 5

### **Answer**

```
// You are using Java
import java.util.*;
```



```

public class Main {
    // Function to check if a number is prime
    static boolean isPrime(int num) {
        if (num <= 1) return false;
        if (num == 2) return true;
        if (num % 2 == 0) return false;
        for (int i = 3; i * i <= num; i += 2) {
            if (num % i == 0) return false;
        }
        return true;
    }

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

        int N = sc.nextInt();
        int[] arr = new int[N];

        int totalSum = 0;
        for (int i = 0; i < N; i++) {
            arr[i] = sc.nextInt();
            totalSum += arr[i];
        }

        boolean found = false;
        for (int i = 0; i < N; i++) {
            int newSum = totalSum - arr[i];
            if (isPrime(newSum)) {
                System.out.println(newSum);
                found = true;
                break; // first possible prime sum
            }
        }

        if (!found) {
            System.out.println("No valid prime sum found");
        }
    }
}

```

**Status :** Correct

**Marks :** 10/10

### 3. Problem Statement

Eminem is a billiard player who enjoys playing billiards and also likes solving mathematical puzzles. He notices that the billiard balls on the table are arranged in a grid, and he is curious to find the sum of the numbers written on each ball.

Write a program to find the sum of all the numbers written on each ball in the grid.

#### ***Input Format***

The first line of input consists of an integer N, representing the number of rows.

The second line consists of an integer M, representing the number of columns.

The following lines N lines consist of M space-separated integers, representing the numbers written on each ball.

#### ***Output Format***

The output prints an integer representing the sum of all the numbers written on each ball.

Refer to the sample output for the formatting specifications.

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

Input: 3

3

1 2 3

4 5 6

7 8 9

Output: 45

#### ***Answer***

```
// You are using Java
import java.util.*;
```

```

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

        // Read dimensions
        int N = sc.nextInt(); // rows
        int M = sc.nextInt(); // columns

        int sum = 0;

        // Read grid and calculate sum
        for (int i = 0; i < N; i++) {
            for (int j = 0; j < M; j++) {
                sum += sc.nextInt();
            }
        }

        // Print result
        System.out.println(sum);
    }
}

```

**Status :** Correct

**Marks :** 10/10

#### 4. Problem Statement

In a customer loyalty program, reward points are logged in a sorted array as customers make transactions. Occasionally, due to system errors, duplicate entries for the same transaction may appear. To ensure accurate reward calculations, it's crucial to remove these duplicates from the list.

Write a program to process the array of reward points, removing any duplicates while preserving the order of unique entries. The program should then display the cleaned list of unique reward points and the total count of these unique points.

#### **Input Format**

The first line of input consists of an integer N, representing the number of reward points.

The second line consists of N space-separated integers, representing the reward points in sorted order.

### **Output Format**

The first line of output prints the cleaned list of unique reward points separated by a space.

The second line of output prints an integer representing the total count of unique reward points.

Refer to the sample output for the formatting specifications.

### **Sample Test Case**

Input: 3  
100 100 200  
Output: 100 200  
2

### **Answer**

```
import java.util.*;

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

        // Read number of elements
        int N = sc.nextInt();
        int[] arr = new int[N];

        for (int i = 0; i < N; i++) {
            arr[i] = sc.nextInt();
        }

        // Remove duplicates while preserving order
        List<Integer> unique = new ArrayList<>();
        unique.add(arr[0]); // first element always unique

        for (int i = 1; i < N; i++) {
            if (arr[i] != arr[i - 1]) {
```

```
        unique.add(arr[i]);
    }
}

// Print cleaned list
for (int num : unique) {
    System.out.print(num + " ");
}
System.out.println();

// Print total count
System.out.println(unique.size());
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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 3\_CY

Attempt : 1

Total Mark : 40

Marks Obtained : 40

### Section 1 : Coding

#### 1. Problem Statement

Robin is a tech-savvy teenager who is diving into programming.

He is working on a project to find special elements in an array called 'leaders.' Leaders are those exceptional elements that are greater than the sum of all the elements to their right.

Assist Robin in writing this program.

Example

Input:

6

16 28 74 19 25 11

Output:

74 25 11

Explanation:

The element 16 is not greater than the sum of elements to its right ( $28 + 74 + 19 + 25 + 11 = 157$ )

The element 28 is not greater than the sum of elements to its right ( $74 + 19 + 25 + 11 = 129$ )

The element 74 is greater than the sum of elements to its right ( $19 + 25 + 11 = 55$ )

The element 19 is not greater than the sum of elements to its right ( $25 + 11 = 36$ )

The element 25 is greater than the sum of elements to its right (11)

The last element 11 is always a leader since there are no elements to its right.

So, the output is {74, 25, 11}.

### ***Input Format***

The first line of input consists of an integer N, representing the number of elements in the array.

The second line consists of N space-separated integers, representing the elements of the array.

### ***Output Format***

The output prints the special elements in the given array, that are greater than the sum of all the elements to their right.

Refer to the sample output for formatting specifications.

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

Input: 5

3 4 2 5 1

Output: 5 1

**Answer**

```
import java.util.*;

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

        // Read input size
        int N = sc.nextInt();
        int[] arr = new int[N];

        for (int i = 0; i < N; i++) {
            arr[i] = sc.nextInt();
        }

        List<Integer> leaders = new ArrayList<>();

        // Traverse the array
        for (int i = 0; i < N; i++) {
            int rightSum = 0;
            for (int j = i + 1; j < N; j++) {
                rightSum += arr[j];
            }

            if (arr[i] > rightSum) {
                leaders.add(arr[i]);
            }
        }

        // Print leaders
        for (int num : leaders) {
            System.out.print(num + " ");
        }
    }
}
```

**Status :** Correct

**Marks :** 10/10

2. Problem Statement:



Emma, a budding computer vision enthusiast, is working on a challenging image processing project. She has a square image represented as a 2D matrix of integers. As part of a special filter operation, she needs to rotate the image by 90 degrees clockwise, but there's a twist — she must perform the rotation in-place, using no extra space.

This means Emma has to rotate the matrix without creating a new one. Your task is to help her implement a Java program that takes this square matrix as input and rotates it within the same structure.

Can you help Emma efficiently rotate the image so that her project can move to the next stage?

### ***Input Format***

The first line of input contains a single integer  $n$ , representing the number of rows and columns of the square matrix (i.e., the matrix is of size  $n \times n$ ).

The next  $n$  lines each contain  $n$  space-separated integers, representing the elements of each row of the 2D array.

### ***Output Format***

The first line of output prints "Rotated 2D Array:"

The next  $n$  lines of output print the rotated matrix.

Each line contains  $n$  space-separated integers representing a row of the rotated matrix.

Refer to the sample output for format specification.

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

Input: 3

1 2 3

4 5 6

7 8 9

Output: Rotated 2D Array:

7 4 1

8 5 2  
9 6 3

**Answer**

// You are using Java  
import java.util.\*;

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

```
        // Read size  
        int n = sc.nextInt();  
        int[][] mat = new int[n][n];
```

```
        // Read matrix  
        for (int i = 0; i < n; i++) {  
            for (int j = 0; j < n; j++) {  
                mat[i][j] = sc.nextInt();  
            }  
        }
```

```
        // Step 1: Transpose the matrix  
        for (int i = 0; i < n; i++) {  
            for (int j = i + 1; j < n; j++) {  
                int temp = mat[i][j];  
                mat[i][j] = mat[j][i];  
                mat[j][i] = temp;  
            }  
        }
```

```
        // Step 2: Reverse each row  
        for (int i = 0; i < n; i++) {  
            int left = 0, right = n - 1;  
            while (left < right) {  
                int temp = mat[i][left];  
                mat[i][left] = mat[i][right];  
                mat[i][right] = temp;  
                left++;  
                right--;  
            }  
        }
```

```

// Output
System.out.println("Rotated 2D Array:");
for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
        System.out.print(mat[i][j] + " ");
    }
    System.out.println();
}
}
}

```

**Status :** Correct

**Marks :** 10/10

### 3. Problem Statement:

Imagine you have an array of integer values, and you're tasked with identifying a pair of elements within the array. This pair of elements should have a sum that is the closest to zero when compared to any other pair in the array.

Your goal is to create a program that solves this problem efficiently. The program should accept an array of integers and return the pair of elements whose sum is closest to zero.

#### **Input Format**

The first line of the input is an integer N representing the size of the array.

The second line of the input contains N space-separated integer values.

#### **Output Format**

The output is displayed in the following format:

"Pair with the sum closest to zero: {value} and {value}"

Refer to the sample output for formatting specifications.

### Sample Test Case

Input: 5

9 10 -3 -5 -2

Output: Pair with the sum closest to zero: 9 and -5

### Answer

// You are using Java

```
import java.util.*;
```

```
public class Main {
```

```
    public static void main(String[] args) {
```

```
        Scanner sc = new Scanner(System.in);
```

```
        int N = sc.nextInt();
```

```
        int[] arr = new int[N];
```

```
        for (int i = 0; i < N; i++) {
```

```
            arr[i] = sc.nextInt();
```

```
        }
```

```
        int minSum = Integer.MAX_VALUE;
```

```
        int val1 = 0, val2 = 0;
```

```
        for (int i = 0; i < N - 1; i++) {
```

```
            for (int j = i + 1; j < N; j++) {
```

```
                int sum = arr[i] + arr[j];
```

```
                if (Math.abs(sum) < Math.abs(minSum)) {
```

```
                    minSum = sum;
```

```
                    val1 = arr[i];
```

```
                    val2 = arr[j];
```

```
                }
```

```
            }
```

```
        }
```

```
        System.out.println("Pair with the sum closest to zero: " + val1 + " and " + val2);
```

```
    }
```

```
}
```

**Status :** Correct

**Marks :** 10/10

#### 4. Problem Statement

Emma is a data analyst working with a grid-based system where each cell contains important numerical data. The grid represents spatial data, inventory records, or structured reports that require periodic updates.

Due to system updates and new requirements, Emma needs to modify the grid in the following ways:

She wants to insert either a new row or a new column at a given position. Later, she needs to delete either a row or a column from the modified matrix.

##### ***Input Format***

The first line contains two integers rows and cols (the dimensions of the matrix).

The next rows lines contain cols space-separated integers representing the initial matrix.

The next line contains two integers insertType and insertIndex:

- insertType = 0 for row insertion, 1 for column insertion.
- insertIndex is the position where the new row/column should be added.

If inserting a row, the next cols integers represent the new row or If inserting a column, the next rows integers represent the new column.

The next line contains two integers deleteType and deleteIndex:

- deleteType = 0 for row deletion, 1 for column deletion.
- deleteIndex is the position to be deleted.

##### ***Output Format***

The first line of output prints the string "After insertion" followed by the modified matrix with the inserted row or column.

Each row of the matrix is printed on a new line with space-separated integers.

The next line prints the string "After deletion" followed by the final matrix after the specified deletion operation.

Each row of the resulting matrix is printed on a new line with space-separated integers.

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 3 3

1 2 3

4 5 6

7 8 9

0 1

10 11 12

1 2

Output: After insertion

1 2 3

10 11 12

4 5 6

7 8 9

After deletion

1 2

10 11

4 5

7 8

### **Answer**

```
import java.util.*;
```

```
public class Main {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
  
        int rows = sc.nextInt();  
        int cols = sc.nextInt();  
  
        // Read initial matrix  
        List<List<Integer>> matrix = new ArrayList<>();  
        for (int i = 0; i < rows; i++) {  
            List<Integer> row = new ArrayList<>();  
            for (int j = 0; j < cols; j++) {
```

```

        row.add(sc.nextInt());
    }
    matrix.add(row);
}

// Insertion
int insertType = sc.nextInt();
int insertIndex = sc.nextInt();

if (insertType == 0) { // Row insertion
    List<Integer> newRow = new ArrayList<>();
    for (int j = 0; j < cols; j++) {
        newRow.add(sc.nextInt());
    }
    matrix.add(insertIndex, newRow);
    rows++;
} else { // Column insertion
    List<Integer> newCol = new ArrayList<>();
    for (int i = 0; i < rows; i++) {
        newCol.add(sc.nextInt());
    }
    for (int i = 0; i < rows; i++) {
        matrix.get(i).add(insertIndex, newCol.get(i));
    }
    cols++;
}

System.out.println("After insertion");
printMatrix(matrix);

// Deletion
int deleteType = sc.nextInt();
int deleteIndex = sc.nextInt();

if (deleteType == 0) { // Row deletion
    matrix.remove(deleteIndex);
    rows--;
} else { // Column deletion
    for (int i = 0; i < rows; i++) {
        matrix.get(i).remove(deleteIndex);
    }
    cols--;
}

```

```
    System.out.println("After deletion");  
    printMatrix(matrix);
```

```
    sc.close();  
}
```

```
private static void printMatrix(List<List<Integer>> matrix) {  
    for (List<Integer> row : matrix) {  
        for (int val : row) {  
            System.out.print(val + " ");  
        }  
        System.out.println();  
    }  
}
```

**Status :** Correct

**Marks :** 10/10



# Rajalakshmi Engineering College

Name: Sachin Krishna B  
Email: 241901507@rajalakshmi.edu.in  
Roll no: 241901507  
Phone: 9025753177  
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 4\_MCQ

Attempt : 1  
Total Mark : 15  
Marks Obtained : 14

#### Section 1 : MCQ

1. What will be the output of the following program?

```
class Main {  
    public static void main(String[] args) {  
        String s = new String("5");  
        System.out.println(1 + 1111 + s + 1 + 1010);  
    }  
}
```

**Answer**

1112511010

**Status : Correct**

**Marks : 1/1**

2. What will be the output of the following program?

```

class Main {
    public static void main(String[] args) {
        String s1 = "EDUCATION";
        String s2 = new String("EDUCATION");
        String s3 = "EDUCATION";
        if (s1 == s2) {
            System.out.println("s1 and s2 equal");
        }
        else {
            System.out.println("s1 and s2 not equal");
        }
        if (s1 == s3) {
            System.out.println("s1 and s3 equal");
        }
        else {
            System.out.println("s1 and s3 not equal");
        }
    }
}

```

**Answer**

s1 and s2 not equals1 and s3 equal

**Status :** Correct

**Marks :** 1/1

3. Predict the output for the following code:

```

public class Main {
    public static void main(String[] args) {
        float a = 10.0f;
        String temp = Float.toString(a);
        System.out.println(temp);
    }
}

```

**Answer**

10.0

**Status :** Correct

**Marks :** 1/1

4. What will be the output of the following code?

```
class Main {  
    public static void main(String args[]) {  
        char c[] = {'j', 'a', 'v', 'a'};  
        String s1 = new String(c);  
        String s2 = new String(s1);  
        System.out.println(s1);  
        System.out.println(s2);  
    }  
}
```

**Answer**

javajava

**Status :** Correct

**Marks :** 1/1

5. Predict the output for the following code:

```
class Main {  
    public static void main(String args[]) {  
        StringBuffer sb = new StringBuffer("I Java!");  
        sb.insert(5, "like ");  
        System.out.println(sb);  
    }  
}
```

**Answer**

I Java like a!

**Status :** Wrong

**Marks :** 0/1

6. What is the output of the following code?

```
class Main  
{  
    public static void main(String args[])  
    {  
        StringBuffer c = new StringBuffer("Hello");
```

```
c.delete(0,2);  
System.out.println(c);  
}  
}
```

**Answer**

llo

**Status :** Correct

**Marks :** 1/1

7. What will be the output of the following program?

```
public class Main {  
    public static void main(String[] args) {  
        String str = "1234.34";  
        int a = Integer.parseInt(str);  
        System.out.println(a);  
    }  
}
```

**Answer**

NumberFormatException

**Status :** Correct

**Marks :** 1/1

8. What will be the output of the following program?

```
class Main {  
    public static void main(String args[]) {  
        String name="Work Hard";  
        name.concat("Success");  
        System.out.println(name);  
    }  
}
```

**Answer**

Work Hard

**Status :** Correct

**Marks :** 1/1

9. What will be the output for the following code?

```
class Main {  
    public static void main(String[] args) {  
        String languages[] = { "C", "C++", "Java", "Python", "Ruby"};  
        for (String sample: languages) {  
            System.out.println(sample);  
        }  
    }  
}
```

**Answer**

CC++JavaPythonRuby

**Status :** Correct

**Marks :** 1/1

10. What will be the output of the following code?

```
class Main {  
    public static void main(String args[]) {  
        String s1 = "Hello i love java";  
        String s2 = new String(s1);  
        System.out.println((s1 == s2) + " " + s1.equals(s2));  
    }  
}
```

**Answer**

false true

**Status :** Correct

**Marks :** 1/1

11. What will be the output of the following code?

```
class Main {  
    public static void main(String args[])  
    {  
        StringBuffer sb = new StringBuffer("Hello");  
        System.out.println("buffer before = " + sb);  
        System.out.println("charAt(1) before = " + sb.charAt(1));  
    }  
}
```

```

        sb.setCharAt(1, 'i');
        sb.setLength(2);
        System.out.println("buffer after = " + sb);
        System.out.println("charAt(1) after = " + sb.charAt(1));
    }
}

```

**Answer**

buffer before = Hello charAt(1) before = e buffer after = Hi charAt(1) after = i

**Status :** Correct

**Marks :** 1/1

12. What will be the output of the following program?

```

class Main {
    public static void main(String args[]) {
        StringBuffer sb = new StringBuffer("Hello");
        System.out.println("buffer = " + sb);
        System.out.println("length = " + sb.length());
        System.out.println("capacity = " + sb.capacity());
    }
}

```

**Answer**

buffer = Hello length = 5 capacity = 21

**Status :** Correct

**Marks :** 1/1

13. Predict the output for the following code.

```

public class Main {
    public static void main(String[] args) {
        String a = "java";
        char temp = a.charAt(1);
        System.out.println(temp);
    }
}

```

**Answer**

a

Status : Correct

Marks : 1/1

14. Predict the output for the following code.

```
class Main {  
    public static void main(String[] fruits) {  
        String fruit1 = new String("apple");  
        String fruit2 = new String("orange");  
        String fruit3 = new String("pear");  
        fruit3 = fruit1;  
        fruit2 = fruit3;  
        fruit1 = fruit2;  
        System.out.println(fruit1);  
        System.out.println(fruit2);  
        System.out.println(fruit3);  
    }  
}
```

Answer

appleappleapple

Status : Correct

Marks : 1/1

15. What will be the output of the following program?

```
class Main {  
    public static void main(String[] args) {  
        String greet = "Welcome\n";  
        System.out.print("String: " + greet);  
        int length = greet.length();  
        System.out.print("Length: " + length);  
    }  
}
```

Answer

String: WelcomeLength: 8

Status : Correct

Marks : 1/1

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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

### 2028\_REC\_OOPS using Java\_Week 4\_Q1

Attempt : 1

Total Mark : 10

Marks Obtained : 10

### Section 1 : Coding

#### 1. Problem Statement

In a publishing company, editors often need to quickly analyze passages of text to check for punctuation usage. To assist them, you are asked to write a program that counts the number of specific punctuation marks in each passage.

The punctuation marks of interest are:

Commas (,) Periods (.) Question marks (?)

#### ***Input Format***

The first line of input contains an integer T, representing the number of test cases (passages).

Each of the next T lines contains a single passage of text.



### **Output Format**

For each test case, print three integers separated by spaces, representing the number of commas, periods, and question marks in the passage.

The first line of output corresponds to the first passage, the second line to the second passage, and so on.

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 1

Hello, world. How are you?

Output: 1 1 1

### **Answer**

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

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

```
        int T = Integer.parseInt(sc.nextLine()); // Number of test cases
```

```
        for (int i = 0; i < T; i++) {  
            String passage = sc.nextLine();
```

```
            int commas = 0, periods = 0, questions = 0;
```

```
            for (char ch : passage.toCharArray()) {  
                if (ch == ',') {  
                    commas++;  
                } else if (ch == '.') {  
                    periods++;  
                } else if (ch == '?') {  
                    questions++;  
                }  
            }  
        }  
    }  
}
```

```
        System.out.println(commas + " " + periods + " " + questions);  
    }  
    sc.close();  
}  
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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

### 2028\_REC\_OOPS using Java\_Week 4\_Q2

Attempt : 1

Total Mark : 10

Marks Obtained : 10

### Section 1 : Coding

#### 1. Problem Statement

Anu is developing a tool for a conference registration system. Participants submit keywords related to their fields of interest. The organizer wants to sort these keywords alphabetically to generate tags for session grouping.

Write a program that accepts at least five keywords as input arguments and outputs them in sorted alphabetical order.

#### ***Input Format***

The first line of input contains an integer n, representing the number of keywords.

The second line of input contains n space-separated keywords (string).

#### ***Output Format***

The output prints n space separated strings representing the sorted keyword in alphabetical order.

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 5

Blockchain Cloud AI Data Cybersecurity

Output: AI Blockchain Cloud Cybersecurity Data

### **Answer**

```
// You are using Java
import java.util.*;
```

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

        int n = Integer.parseInt(sc.nextLine()); // number of keywords
        String[] keywords = sc.nextLine().split(" "); // read keywords

        Arrays.sort(keywords); // sort alphabetically

        for (int i = 0; i < n; i++) {
            System.out.print(keywords[i]);
            if (i < n - 1) {
                System.out.print(" ");
            }
        }
        sc.close();
    }
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B  
Email: 241901507@rajalakshmi.edu.in  
Roll no: 241901507  
Phone: 9025753177  
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

### 2028\_REC\_OOPS using Java\_Week 4\_Q3

Attempt : 1  
Total Mark : 10  
Marks Obtained : 10

#### Section 1 : Coding

##### 1. Problem Statement

Bechan Chacha is seeking help to filter out valid mobile numbers from a list provided by his crush. He can only pick his crush's number if the list contains valid mobile numbers.

A mobile number is considered valid if:

It has exactly 10 digits. It consists only of numeric values (0–9). It does not begin with zero.

Your task is to determine whether each mobile number in the list is valid or not.

##### ***Input Format***

The first line contains an integer T, representing the number of mobile numbers

to check.

The next T lines each contain a string S, representing a mobile number.

**Output Format**

For each mobile number S, the output print "YES" if it is valid.

Otherwise, print "NO".

Refer to the sample output for formatting specifications.

**Sample Test Case**

Input: 1  
9876543210

Output: YES

**Answer**

```
import java.util.*;
```

```
public class Main {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
  
        int T = Integer.parseInt(sc.nextLine()); // number of test cases  
  
        for (int i = 0; i < T; i++) {  
            String s = sc.nextLine().trim();  
  
            if (isValidMobile(s)) {  
                System.out.println("YES");  
            } else {  
                System.out.println("NO");  
            }  
        }  
  
        sc.close();  
    }  
}
```

```
// Function to validate mobile number
```

```
public static boolean isValidMobile(String s) {  
    // must be exactly 10 digits  
    if (s.length() != 10) {  
        return false;  
    }  
    // must contain only digits  
    for (char ch : s.toCharArray()) {  
        if (!Character.isDigit(ch)) {  
            return false;  
        }  
    }  
    // must not start with 0  
    if (s.charAt(0) == '0') {  
        return false;  
    }  
    return true;  
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B  
Email: 241901507@rajalakshmi.edu.in  
Roll no: 241901507  
Phone: 9025753177  
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

### 2028\_REC\_OOPS using Java\_Week 4\_Q4

Attempt : 1  
Total Mark : 10  
Marks Obtained : 10

#### Section 1 : Coding

##### 1. Problem Statement

Arjun is learning how to filter words from a sentence based on grammar rules. He wants to identify the valid words in a sentence.

A word is considered valid if it satisfies all these conditions:

The word contains only alphabets (a-z, A-Z). The word length is at least 2 characters. The word should not contain digits or special characters.

Your task is to read a sentence and print all the valid words in it.

##### ***Input Format***

The input contains a single line containing a sentence S.

##### ***Output Format***



The output prints all the valid words separated by spaces.

If no valid word exists, print "No valid words."

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: Hello world1 123 ab" @#\$ Hi

Output: Hello Hi

### **Answer**

```
import java.util.*;

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

        String sentence = sc.nextLine();
        String[] words = sentence.split(" "); // split by space

        List<String> validWords = new ArrayList<>();

        for (String word : words) {
            if (isValidWord(word)) {
                validWords.add(word);
            }
        }

        if (validWords.isEmpty()) {
            System.out.println("No valid words.");
        } else {
            System.out.println(String.join(" ", validWords));
        }

        sc.close();
    }

    // Function to check if word is valid
    public static boolean isValidWord(String word) {
```

```
    if (word.length() < 2) {  
        return false;  
    }  
    for (char ch : word.toCharArray()) {  
        if (!Character.isLetter(ch)) {  
            return false;  
        }  
    }  
    return true;  
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B  
Email: 241901507@rajalakshmi.edu.in  
Roll no: 241901507  
Phone: 9025753177  
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

### 2028\_REC\_OOPS using Java\_Week 4\_Q5

Attempt : 1  
Total Mark : 10  
Marks Obtained : 10

#### Section 1 : Coding

##### 1. Problem Statement

In a secure banking system, customers are required to create PIN codes for accessing their accounts. The bank wants to validate these PIN codes before accepting them.

A PIN code is considered valid if:

It consists of exactly 4 digits. All characters must be numeric (0–9). It cannot contain all identical digits (e.g., 1111 is invalid).

Your task is to determine whether each PIN code in the list is valid or not.

##### ***Input Format***

The first line of input contains an integer T, representing the number of PIN codes to check.

The next T lines each contain a string S, representing a PIN code.

### **Output Format**

For each PIN code S, the output print "YES" if it is valid.

Otherwise, the output print "NO".

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 1

1234

Output: YES

### **Answer**

// You are using Java

import java.util.\*;

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

        int T = Integer.parseInt(sc.nextLine()); // number of PIN codes

        for (int i = 0; i < T; i++) {
            String pin = sc.nextLine().trim();

            if (isValidPIN(pin)) {
                System.out.println("YES");
            } else {
                System.out.println("NO");
            }
        }

        sc.close();
    }

    // Function to validate PIN
```

```
public static boolean isValidPIN(String pin) {  
    // must be exactly 4 characters  
    if (pin.length() != 4) {  
        return false;  
    }  
  
    // must contain only digits  
    for (char ch : pin.toCharArray()) {  
        if (!Character.isDigit(ch)) {  
            return false;  
        }  
    }  
  
    // check if all digits are identical  
    char first = pin.charAt(0);  
    boolean allSame = true;  
    for (char ch : pin.toCharArray()) {  
        if (ch != first) {  
            allSame = false;  
            break;  
        }  
    }  
  
    return !allSame; // valid only if not all digits are same  
}
```

**Status :** Correct

**Marks : 10/10**

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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 4\_PAH

Attempt : 1

Total Mark : 40

Marks Obtained : 40

### Section 1 : Coding

#### 1. Problem Statement

Ravi is analyzing text messages for his research on typing patterns. He wants to count the number of uppercase letters, lowercase letters, and digits in a sentence to understand typing trends.

Your task is to help Ravi by writing a program that takes a sentence and prints the count of uppercase letters, lowercase letters, and digits.

#### ***Input Format***

The input contains a single line containing a sentence (string).

#### ***Output Format***

The output prints three integers separated by spaces:

- Number of uppercase letters
- Number of lowercase letters
- Number of digits

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: Hello World 123

Output: 2 8 3

### **Answer**

```
import java.util.*;

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

        int uppercase = 0, lowercase = 0, digits = 0;

        for (char ch : sentence.toCharArray()) {
            if (Character.isUpperCase(ch)) {
                uppercase++;
            } else if (Character.isLowerCase(ch)) {
                lowercase++;
            } else if (Character.isDigit(ch)) {
                digits++;
            }
        }

        System.out.println(uppercase + " " + lowercase + " " + digits);
        sc.close();
    }
}
```

**Status :** Correct

**Marks :** 10/10

## **2. Problem Statement**

At a digital library, the system needs to analyze passages to identify the frequency of vowels, since they are key for linguistic research. You are asked to write a program that counts the number of vowels in each passage of text.

The vowels of interest are:

a, e, i, o, u (both uppercase and lowercase).

### ***Input Format***

The first line of input contains an integer T, representing the number of test cases (passages).

Each of the next T lines contains a single passage of text.

### ***Output Format***

For each test case, print a single integer representing the total number of vowels in the passage.

The first line of output corresponds to the first passage, the second line to the second passage, and so on.

Refer to the sample output for formatting specifications.

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

Input: 1  
Hello World

Output: 3

### ***Answer***

```
// You are using Java
import java.util.*;
```

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



```

int T = Integer.parseInt(sc.nextLine()); // number of test cases

for (int i = 0; i < T; i++) {
    String passage = sc.nextLine();
    int vowelCount = 0;

    for (char ch : passage.toCharArray()) {
        if (isVowel(ch)) {
            vowelCount++;
        }
    }

    System.out.println(vowelCount);
}

sc.close();
}

// Function to check if a character is a vowel
public static boolean isVowel(char ch) {
    ch = Character.toLowerCase(ch);
    return (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u');
}
}

```

**Status :** Correct

**Marks :** 10/10

### 3. Problem Statement

Sana is analyzing text for a secret code. She wants to find all words in a sentence that start and end with the same letter. These words are considered "special words" for her analysis.

Your task is to write a program that extracts and prints all words that start and end with the same letter (case-insensitive).

If no such word exists, print "No special words found".

#### **Input Format**

The input contains a single line containing a sentence with multiple words.

### **Output Format**

The output prints all words that start and end with the same letter separated by a space.

If no word satisfies the condition, print "No special words found".

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: Anna went to the civic center

Output: Anna civic

### **Answer**

```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String sentence = sc.nextLine();
        String[] words = sentence.split(" "); // split by space

        List<String> specialWords = new ArrayList<>();

        for (String word : words) {
            if (word.length() > 0) {
                char first = Character.toLowerCase(word.charAt(0));
                char last = Character.toLowerCase(word.charAt(word.length() - 1));

                if (first == last) {
                    specialWords.add(word);
                }
            }
        }

        if (specialWords.isEmpty()) {
            System.out.println("No special words found");
        } else {
            System.out.println(String.join(" ", specialWords));
        }
    }
}
```

```
}  
    sc.close();  
}  
}
```

**Status :** Correct

**Marks :** 10/10

#### 4. Problem Statement

Riya is preparing a puzzle game for her friends. She wants to include a feature that highlights special words in a sentence – specifically, palindromic words (words that read the same forward and backward).

Your task is to help Riya by writing a program that extracts all palindrome words from the given sentence. If there are no palindromes, print "No palindromes found".

##### ***Input Format***

The input contains a single string S representing a sentence.

##### ***Output Format***

The output prints all palindromic words separated by a space.

If no palindrome exists, print "No palindromes found".

Refer to the sample output for formatting specifications.

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

Input: madam went to school

Output: madam

##### ***Answer***

```
// You are using Java  
import java.util.*;
```

```

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String sentence = sc.nextLine();
        String[] words = sentence.split(" "); // split by space

        List<String> palindromes = new ArrayList<>();

        for (String word : words) {
            if (isPalindrome(word)) {
                palindromes.add(word);
            }
        }

        if (palindromes.isEmpty()) {
            System.out.println("No palindromes found");
        } else {
            System.out.println(String.join(" ", palindromes));
        }

        sc.close();
    }

    // Helper method to check palindrome
    public static boolean isPalindrome(String word) {
        int left = 0, right = word.length() - 1;
        while (left < right) {
            if (word.charAt(left) != word.charAt(right)) {
                return false;
            }
            left++;
            right--;
        }
        return true;
    }
}

```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B  
Email: 241901507@rajalakshmi.edu.in  
Roll no: 241901507  
Phone: 9025753177  
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 4\_CY

Attempt : 1  
Total Mark : 40  
Marks Obtained : 40

#### Section 1 : Coding

##### 1. Problem Statement

In a university library, librarians need to track the usage of special characters in students' notes.

To help them, you are asked to write a program that counts the number of specific symbols in each passage of text.

The symbols of interest are:

Exclamation marks (!)Colons (: )Semicolons (;)

##### ***Input Format***

The first line of input contains an integer T, representing the number of test cases (passages).

Each of the next T lines contains a single passage of text.

### **Output Format**

For each test case, print three integers separated by spaces, representing the number of exclamation marks, colons, and semicolons in the passage.

The first line of output corresponds to the first passage, the second line to the second passage, and so on.

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 1  
Hello! How are you  
Output: 1 0 0

### **Answer**

```
import java.util.*;

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

        int T = Integer.parseInt(sc.nextLine()); // number of test cases

        for (int i = 0; i < T; i++) {
            String passage = sc.nextLine();

            int exclamations = 0, colons = 0, semicolons = 0;

            for (char ch : passage.toCharArray()) {
                if (ch == '!') {
                    exclamations++;
                } else if (ch == ':') {
                    colons++;
                } else if (ch == ';') {
                    semicolons++;
                }
            }
        }
    }
}
```

```
        System.out.println(exclamations + " " + colons + " " + semicolons);
    }
    sc.close();
}
```

**Status :** Correct

**Marks :** 10/10

## 2. Problem Statement

Riya is preparing for a vocabulary test. Her teacher told her to focus on long words in her practice sentences, specifically words that have at least 5 letters.

Riya wants to write a program that will help her identify such words quickly.

Your task is to help Riya by printing all the words in a given sentence that have a length greater than or equal to 5.

If no such word exists, display "No long words found".

### **Input Format**

The input contains a single line containing a sentence with multiple words.

### **Output Format**

The output prints all words having length  $\geq 5$ , separated by a space.

If no such word is found, print "No long words found".

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: The quick brown fox jumps over the lazy dog

Output: quick brown jumps

### Answer

```
// You are using Java
import java.util.*;
```

```
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String sentence = sc.nextLine();
        String[] words = sentence.split(" "); // split by space

        List<String> longWords = new ArrayList<>();

        for (String word : words) {
            if (word.length() >= 5) {
                longWords.add(word);
            }
        }

        if (longWords.isEmpty()) {
            System.out.println("No long words found");
        } else {
            // print words in the same order, separated by space
            for (String w : longWords) {
                System.out.print(w + " ");
            }
            System.out.println();
        }

        sc.close();
    }
}
```

**Status :** Correct

**Marks :** 10/10

### 3. Problem Statement

In a college, students are required to create unique usernames for accessing the digital library.

The librarian needs your help to verify whether the usernames entered by students are valid.



A username is considered valid if:

It contains only letters (a–z, A–Z) and digits (0–9). Its length is between 5 and 15 characters (inclusive). It must start with a letter (not a digit).

Your task is to determine whether each username in the list is valid or not.

### ***Input Format***

The first line of input contains an integer T, representing the number of usernames to check.

The next T lines each contain a string S, representing a username.

### ***Output Format***

For each username S, the output print "YES" if it is valid.

Otherwise, the output print "NO".

Refer to the sample output for formatting specifications.

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

Input: 1

Alice123

Output: YES

### ***Answer***

```
// You are using Java
import java.util.*;
```

```
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int T = Integer.parseInt(sc.nextLine());

        for (int i = 0; i < T; i++) {
            String username = sc.nextLine();

            if (isValidUsername(username)) {
```

```

        System.out.println("YES");
    } else {
        System.out.println("NO");
    }
}
sc.close();
}

public static boolean isValidUsername(String s) {
    // Length check
    if (s.length() < 5 || s.length() > 15) {
        return false;
    }
    // First character must be a letter
    if (!Character.isLetter(s.charAt(0))) {
        return false;
    }
    // Only letters and digits allowed
    for (char c : s.toCharArray()) {
        if (!Character.isLetterOrDigit(c)) {
            return false;
        }
    }
    return true;
}
}

```

**Status :** Correct

**Marks :** 10/10

#### 4. Problem Statement

Meera is practicing her English vocabulary. She wants to focus on words that have more vowels in them, as they help improve her pronunciation. She decides to extract only those words from a sentence that contain at least two vowels.

Your task is to help Meera by writing a program that finds such words from

the given sentence.

### ***Input Format***

The input contains a string representing the sentence.

### ***Output Format***

The output prints all the words that contain at least two vowels, separated by a space.

If no such word exists, print "No words with two vowels".

Refer to the sample output for formatting specifications.

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

Input: This is an example sentence

Output: example sentence

### ***Answer***

```
import java.util.*;

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

        String[] words = sentence.split(" ");
        List<String> result = new ArrayList<>();

        for (String word : words) {
            if (countVowels(word) >= 2) {
                result.add(word);
            }
        }

        if (result.isEmpty()) {
            System.out.println("No words with two vowels");
        } else {
            System.out.println(String.join(" ", result));
        }
    }
}
```

```
}  
}  
  
public static int countVowels(String word) {  
    int count = 0;  
    for (char c : word.toLowerCase().toCharArray()) {  
        if ("aeiou".indexOf(c) != -1) {  
            count++;  
        }  
    }  
    return count;  
}  
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B  
Email: 241901507@rajalakshmi.edu.in  
Roll no: 241901507  
Phone: 9025753177  
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

### 2028\_REC\_OOPS using Java\_Week 5\_MCQ

Attempt : 1  
Total Mark : 15  
Marks Obtained : 15

#### Section 1 : MCQ

1. What will be the output of the following code?

```
class Alpha {  
    void greet(String name) {  
        System.out.println("Hello " + name);  
    }  
}
```

```
public class Main {  
    public static void main(String[] args) {  
        Alpha obj = new Alpha();  
        obj.greet("Anu");  
    }  
}
```

**Answer**

Hello Anu

**Status :** Correct

**Marks :** 1/1

2. What will be the output of the following code?

```
class MathUtils {  
    int add(int x) {  
        return x + x;  
    }  
}  
  
public class Main {  
    public static void main(String[] args) {  
        MathUtils m = new MathUtils();  
        System.out.println(m.add(5));  
    }  
}
```

**Answer**

10

**Status :** Correct

**Marks :** 1/1

3. What will be the output of the following code?

```
class Person {  
    int age = 18;  
}  
  
public class Main {  
    public static void main(String[] args) {  
        Person p = new Person();  
        p.age += 2;  
        System.out.println("Age: " + p.age);  
    }  
}
```

**Answer**

Age: 20

Status : Correct

Marks : 1/1

4. What will be the output of the following code?

```
class Sample {  
    int x = 10;  
  
    void display() {  
        System.out.println("x = " + x);  
    }  
  
    public static void main(String[] args) {  
        Sample s = new Sample();  
        s.display();  
    }  
}
```

Answer

x = 10

Status : Correct

Marks : 1/1

5. What is the output of the following code?

```
class Box {  
    int height;  
    Box(int height) {  
        this.height = height;  
    }  
    void modifyHeight(Box b) {  
        b.height += 10;  
    }  
}  
  
public class Main {  
    public static void main(String[] args) {  
        Box b1 = new Box(20);  
        b1.modifyHeight(b1);  
    }  
}
```

```
        System.out.println(b1.height);
    }
}
```

**Answer**

30

**Status :** Correct

**Marks :** 1/1

6. What will be the output of the following code?

```
class A {
    int p = 5;
    int q = 2;
}

class Main {
    public static void main(String[] args) {
        A obj = new A();
        System.out.println(obj.p + obj.q);
    }
}
```

**Answer**

7

**Status :** Correct

**Marks :** 1/1

7. What will be the output of the following code?

```
class A {
    int y = 30;
}

public class Main {
    public static void main(String[] args) {
        A a1 = new A();
        A a2 = new A();
        a1.y = 50;
    }
}
```



```
        System.out.println(a2.y);  
    }  
}
```

**Answer**

30

**Status :** Correct

**Marks :** 1/1

8. What will be the output of the following code?

```
class A {  
    int val = 20;  
}  
  
public class Main {  
    public static void main(String[] args) {  
        A obj1 = new A();  
        A obj2 = obj1;  
        obj2.val += 5;  
        System.out.println(obj1.val);  
    }  
}
```

**Answer**

25

**Status :** Correct

**Marks :** 1/1

9. What will be the output of the following code?

```
class Test {  
    private int value;  
    Test(int value) {  
        this.value = value;  
    }  
    public int getValue() {  
        return value;  
    }  
}
```

```
}  
public class Main {  
    public static void main(String[] args) {  
        Test obj = new Test(10);  
        System.out.println(obj.value);  
    }  
}
```

**Answer**

Compile-time error

**Status :** Correct

**Marks :** 1/1

10. What will be the output of the following code?

```
class Box {  
    int volume(int l, int b, int h) {  
        return l * b * h;  
    }  
}  
  
public class Main {  
    public static void main(String[] args) {  
        Box b = new Box();  
        System.out.println(b.volume(2, 3, 4));  
    }  
}
```

**Answer**

24

**Status :** Correct

**Marks :** 1/1

11. What will be the output of the following code?

```
class Box {  
    int length = 5;  
    int width = 4;
```

```
int area() {  
    return length * width;  
}  
  
public static void main(String[] args) {  
    Box b = new Box();  
    System.out.println("Area = " + b.area());  
}  
}
```

**Answer**

Area = 20

**Status :** Correct

**Marks :** 1/1

12. What will be the output of the following code?

```
class Demo {  
    void printMessage() {  
        System.out.println("Hello from Demo");  
    }  
}  
  
public class Main {  
    public static void main(String[] args) {  
        Demo d = new Demo();  
        d.printMessage();  
    }  
}
```

**Answer**

Hello from Demo

**Status :** Correct

**Marks :** 1/1

13. What will be the output of the following code?

```
class Ball {  
    int size = 11;
```

```
}  
class Game {  
    public static void main(String[] args) {  
        Ball b1 = new Ball();  
        Ball b2 = new Ball();  
        b2.size = 10;  
        System.out.println(b1.size);  
    }  
}
```

**Answer**

11

**Status :** Correct

**Marks :** 1/1

14. What will be the output of the following code?

```
class A {  
    int x = 50;  
}  
  
public class Main {  
    public static void main(String[] args) {  
        A obj1 = new A();  
        A obj2 = obj1;  
        obj2.x = 100;  
        System.out.println(obj1.x);  
    }  
}
```

**Answer**

100

**Status :** Correct

**Marks :** 1/1

15. What will be the output of the following code?

```
class Person {
```

```
String name;  
void setName(String n){  
    name = n;  
}  
void printName() {  
    System.out.println(name);  
}  
}  
  
class Test {  
    public static void main(String[] args) {  
        Person p = new Person();  
        p.printName();  
    }  
}
```

**Answer**

null

**Status :** Correct

**Marks :** 1/1

# Rajalakshmi Engineering College

Name: Sachin Krishna B  
Email: 241901507@rajalakshmi.edu.in  
Roll no: 241901507  
Phone: 9025753177  
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

### 2028\_REC\_OOPS using Java\_Week 5\_Q2

Attempt : 1  
Total Mark : 10  
Marks Obtained : 10

#### Section 1 : Coding

##### 1. Problem Statement

You are working as a developer for CityBank, which wants to build a basic account management system.

Each customer at the bank has:

An Account Number (integer) A Customer Name (string) An Initial Balance (double)

The bank allows two types of transactions:

Deposit – increases the balance. Withdrawal – decreases the balance only if enough funds are available.

If the withdrawal amount is greater than the balance, the withdrawal should not happen, and the balance should remain the same.

You are required to implement this system using:

A class with attributes for account details. A constructor to initialize account details. Setter methods to update details if needed. Getter methods to retrieve details. Objects of the class to represent customers.

Finally, display each customer's account details after all transactions.

### ***Input Format***

The first line of input contains an integer N, representing the number of customers.

For each customer:

- The next line contains the account number (integer).
- The following line contains the customer name (string).
- The next line contains the initial balance (double).
- The next line contains the deposit amount (double).
- The next line contains the withdrawal amount (double).

### ***Output Format***

For each customer, print the details in the following format:

1. Account Number: <account\_number>
2. Customer Name: <customer\_name>
3. Final Balance: <final\_balance> (rounded to one decimal place)

Refer to the sample output for formatting specifications.

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

Input: 1

1234

Rahul Sharma

5000

2000

3000

Output: Account Number: 1234

Customer Name: Rahul Sharma

Final Balance: 4000.0

**Answer**

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

```
class Account {  
    private int accountNumber;  
    private String customerName;  
    private double balance;  
  
    // Constructor  
    public Account(int accountNumber, String customerName, double balance) {  
        this.accountNumber = accountNumber;  
        this.customerName = customerName;  
        this.balance = balance;  
    }  
  
    // Getter methods  
    public int getAccountNumber() {  
        return accountNumber;  
    }  
  
    public String getCustomerName() {  
        return customerName;  
    }  
  
    public double getBalance() {  
        return balance;  
    }  
  
    // Setter methods  
    public void setAccountNumber(int accountNumber) {  
        this.accountNumber = accountNumber;  
    }  
  
    public void setCustomerName(String customerName) {  
        this.customerName = customerName;  
    }  
  
    public void setBalance(double balance) {  
        this.balance = balance;  
    }  
}
```



```

// Deposit
public void deposit(double amount) {
    if (amount >= 0) {
        balance += amount;
    }
}

// Withdraw
public void withdraw(double amount) {
    if (amount <= balance) {
        balance -= amount;
    }
}

}

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int N = Integer.parseInt(sc.nextLine()); // number of customers

        for (int i = 0; i < N; i++) {
            int accNo = Integer.parseInt(sc.nextLine());
            String name = sc.nextLine();
            double initBal = Double.parseDouble(sc.nextLine());
            double depositAmt = Double.parseDouble(sc.nextLine());
            double withdrawAmt = Double.parseDouble(sc.nextLine());

            Account acc = new Account(accNo, name, initBal);

            acc.deposit(depositAmt);
            acc.withdraw(withdrawAmt);

            // Print details
            System.out.println("Account Number: " + acc.getAccountNumber());
            System.out.println("Customer Name: " + acc.getCustomerName());
            System.out.printf("Final Balance: %.1f%n", acc.getBalance());
        }

        sc.close();
    }
}

```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B  
Email: 241901507@rajalakshmi.edu.in  
Roll no: 241901507  
Phone: 9025753177  
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

### 2028\_REC\_OOPS using Java\_Week 5\_Q3

Attempt : 1  
Total Mark : 10  
Marks Obtained : 10

#### Section 1 : Coding

##### 1. Problem Statement

Neha is working as a developer for CityElectricity Board, which wants to build a household electricity billing system.

Each customer's electricity account has:

A Customer ID (integer) A Customer Name (string) Units Consumed (double)

The electricity bill is calculated based on these rules:

For the first 100 units 5 units charge per unit For the next 100 units (101–200) 7 units charge per unit For units above 200 10 units charge per unit If the total bill exceeds 2000 units, a 5% discount is applied on the final bill.

Neha has been asked to implement this system using:

A class with attributes for customer details. A constructor to initialize customer details. Setter methods to update details if needed. Getter methods to retrieve details. Objects of the class to represent customers.

Finally, display each customer's details and final bill amount.

### ***Input Format***

The first line of input contains an integer N, representing the number of customers.

For each customer:

- The next line contains the Customer ID (integer).
- The following line contains the Customer Name (string).
- The next line contains the Units Consumed (double).

### ***Output Format***

For each customer, print the details in the following format:

Customer ID: <customer\_id>

Customer Name: <customer\_name>

Final Bill: <final\_bill> (rounded to one decimal place)

Refer to the sample output for formatting specifications.

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

Input: 1

1001

Ravi Kumar

80

Output: Customer ID: 1001

Customer Name: Ravi Kumar

Final Bill: 400.0

### ***Answer***

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

```
class Customer {
    private int customerId;
    private String customerName;
    private double unitsConsumed;
    private double finalBill;

    // Constructor
    public Customer(int customerId, String customerName, double
unitsConsumed) {
        this.customerId = customerId;
        this.customerName = customerName;
        this.unitsConsumed = unitsConsumed;
        this.finalBill = calculateBill();
    }

    // Getters
    public int getCustomerId() {
        return customerId;
    }

    public String getCustomerName() {
        return customerName;
    }

    public double getUnitsConsumed() {
        return unitsConsumed;
    }

    public double getFinalBill() {
        return finalBill;
    }

    // Setters
    public void setCustomerId(int customerId) {
        this.customerId = customerId;
    }

    public void setCustomerName(String customerName) {
        this.customerName = customerName;
    }
}
```

```
public void setUnitsConsumed(double unitsConsumed) {  
    this.unitsConsumed = unitsConsumed;  
    this.finalBill = calculateBill(); // Recalculate when updated  
}
```

```
// Bill Calculation
```

```
private double calculateBill() {  
    double bill = 0;
```

```
    if (unitsConsumed <= 100) {  
        bill = unitsConsumed * 5;  
    } else if (unitsConsumed <= 200) {  
        bill = (100 * 5) + (unitsConsumed - 100) * 7;  
    } else {  
        bill = (100 * 5) + (100 * 7) + (unitsConsumed - 200) * 10;  
    }
```

```
    // Apply 5% discount if bill > 2000  
    if (bill > 2000) {  
        bill = bill - (bill * 0.05);  
    }
```

```
    return bill;
```

```
}
```

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

```
        int N = Integer.parseInt(sc.nextLine()); // number of customers
```

```
        for (int i = 0; i < N; i++) {  
            int id = Integer.parseInt(sc.nextLine());  
            String name = sc.nextLine();  
            double units = Double.parseDouble(sc.nextLine());
```

```
            Customer c = new Customer(id, name, units);
```

```
            // Output as per format  
            System.out.println("Customer ID: " + c.getCustomerId());  
            System.out.println("Customer Name: " + c.getCustomerName());
```

```
        System.out.printf("Final Bill: %.1f%n", c.getFinalBill());
    }
    sc.close();
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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

### 2028\_REC\_OOPS using Java\_Week 5\_Q4

Attempt : 1

Total Mark : 10

Marks Obtained : 10

### Section 1 : Coding

#### 1. Problem Statement

You are working as a developer for CityCab, a taxi service company that wants to build a ride fare management system.

Each customer booking has:

A Booking ID (integer) A Customer Name (string) A Distance Travelled in km (double)

The fare calculation rules are:

Base Fare = 50 units (flat charge for every ride). Per km charge = 10 units/km. If the distance is greater than 20 km, a 10% discount is applied on the total fare.

You are required to implement this system using:



A class with attributes for booking details. A constructor to initialize booking details. Setter methods to update details if needed. Getter methods to retrieve details. Objects of the class to represent customer rides.

Finally, display each booking's details and final fare.

### ***Input Format***

The first line of input contains an integer N, representing the number of bookings.

For each booking:

- The next line contains the booking ID (integer).
- The following line contains the customer's name (string).
- The next line contains the distance travelled (double).

### ***Output Format***

For each booking, print the details in the following format:

1. Booking ID: <booking\_id>
2. Customer Name: <customer\_name>
3. Final Fare: <final\_fare> (rounded to one decimal place)

Refer to the sample output for formatting specifications.

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

Input: 1

1234

Rahul Sharma

15

Output: Booking ID: 1234

Customer Name: Rahul Sharma

Final Fare: 200.0

### ***Answer***

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

```
class Booking {
```

```
private int bookingId;
private String customerName;
private double distance;
private double finalFare;

// Constructor
public Booking(int bookingId, String customerName, double distance) {
    this.bookingId = bookingId;
    this.customerName = customerName;
    this.distance = distance;
    this.finalFare = calculateFare();
}

// Getters
public int getBookingId() {
    return bookingId;
}

public String getCustomerName() {
    return customerName;
}

public double getDistance() {
    return distance;
}

public double getFinalFare() {
    return finalFare;
}

// Setters
public void setBookingId(int bookingId) {
    this.bookingId = bookingId;
}

public void setCustomerName(String customerName) {
    this.customerName = customerName;
}

public void setDistance(double distance) {
    this.distance = distance;
    this.finalFare = calculateFare(); // recalc on update
}
```

```

    }
    // Fare Calculation
    private double calculateFare() {
        double baseFare = 50;
        double perKmCharge = distance * 10;
        double total = baseFare + perKmCharge;

        if (distance > 20) {
            total = total - (total * 0.10); // 10% discount
        }
        return total;
    }
}

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

        int N = Integer.parseInt(sc.nextLine()); // number of bookings

        for (int i = 0; i < N; i++) {
            int id = Integer.parseInt(sc.nextLine());
            String name = sc.nextLine();
            double distance = Double.parseDouble(sc.nextLine());

            Booking b = new Booking(id, name, distance);

            // Output in required format
            System.out.println("Booking ID: " + b.getBookingId());
            System.out.println("Customer Name: " + b.getCustomerName());
            System.out.printf("Final Fare: %.1f%n", b.getFinalFare());
        }

        sc.close();
    }
}

```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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

### 2028\_REC\_OOPS using Java\_Week 5\_Q5

Attempt : 1

Total Mark : 10

Marks Obtained : 10

### Section 1 : Coding

#### 1. Problem Statement

Ram is working as a developer for BrightEdu Coaching Center, which wants to build a student fee management system.

Each student's enrollment has:

An Enrollment ID (integer) A Student Name (string) The Number of Subjects (integer)

The fee calculation rules are:

Registration Fee = 1000 units (flat for every student). Per Subject Fee = 800 units. If the student enrolls in more than 5 subjects, a 20% scholarship (discount) is applied on the total fee.

Ram has been asked to implement this system using:

A class with attributes for student details. A constructor to initialize student details. Setter methods to update details if needed. Getter methods to retrieve details. Objects of the class to represent student enrollments.

Finally, display each student's details and final fee.

### ***Input Format***

The first line of input contains an integer N, representing the number of students.

For each student:

- The next line contains the Enrollment ID (integer).
- The following line contains the student's name (string).
- The next line contains the Number of subjects (integer).

### ***Output Format***

For each student, print the details in the following format:

- Enrollment ID: <enrollment\_id>
- Student Name: <student\_name>
- Final Fee: <final\_fee> (rounded to one decimal place)

Refer to the sample output for formatting specifications.

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

Input: 1

1234

Ravi Kumar

3

Output: Enrollment ID: 1234

Student Name: Ravi Kumar

Final Fee: 3400.0

### ***Answer***

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

```
class Student {  
    private int enrollmentId;
```

```
private String studentName;
private int numSubjects;
private double finalFee;

// Constructor
public Student(int enrollmentId, String studentName, int numSubjects) {
    this.enrollmentId = enrollmentId;
    this.studentName = studentName;
    this.numSubjects = numSubjects;
    this.finalFee = calculateFee();
}

// Getters
public int getEnrollmentId() {
    return enrollmentId;
}

public String getStudentName() {
    return studentName;
}

public int getNumSubjects() {
    return numSubjects;
}

public double getFinalFee() {
    return finalFee;
}

// Setters
public void setEnrollmentId(int enrollmentId) {
    this.enrollmentId = enrollmentId;
}

public void setStudentName(String studentName) {
    this.studentName = studentName;
}

public void setNumSubjects(int numSubjects) {
    this.numSubjects = numSubjects;
    this.finalFee = calculateFee(); // Recalculate fee when updated
}
```

```

// Fee Calculation
private double calculateFee() {
    double registrationFee = 1000;
    double perSubjectFee = numSubjects * 800;
    double totalFee = registrationFee + perSubjectFee;

    // Apply 20% scholarship if more than 5 subjects
    if (numSubjects > 5) {
        totalFee = totalFee - (totalFee * 0.20);
    }

    return totalFee;
}

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

        int N = Integer.parseInt(sc.nextLine()); // number of students

        for (int i = 0; i < N; i++) {
            int id = Integer.parseInt(sc.nextLine());
            String name = sc.nextLine();
            int subjects = Integer.parseInt(sc.nextLine());

            Student s = new Student(id, name, subjects);

            // Print details in required format
            System.out.println("Enrollment ID: " + s.getEnrollmentId());
            System.out.println("Student Name: " + s.getStudentName());
            System.out.printf("Final Fee: %.1f%n", s.getFinalFee());
        }

        sc.close();
    }
}

```

**Status :** Correct

**Marks : 10/10**

# Rajalakshmi Engineering College

Name: Sachin Krishna B  
Email: 241901507@rajalakshmi.edu.in  
Roll no: 241901507  
Phone: 9025753177  
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

### 2028\_REC\_OOPS using Java\_Week 5\_PAH

Attempt : 1  
Total Mark : 50  
Marks Obtained : 50

#### Section 1 : Coding

##### 1. Problem Statement

Ravi is working as a developer for SecureLogin Systems, which wants to build a system to evaluate the strength of user passwords.

Each user record has:

User ID (integer) User Name (string) Password (string)

The system must calculate whether a password is strong or weak.

A password is considered strong if it meets all of the following conditions:

At least 8 characters long. Contains at least one uppercase letter. Contains at least one lowercase letter. Contains at least one digit. Contains at least one special character (from !@#\$%^&\*).



Ravi has been asked to implement this system using:

A class with attributes for user details. A constructor to initialize user details. Getter and setter methods to retrieve or update user details. A method to check whether the password is strong. Objects of the class to represent users.

Finally, display each user's details and indicate whether their password is Strong or Weak.

### ***Input Format***

The first line contains an integer N, representing the number of users.

For each user:

The next line contains the User ID (integer).

The next line contains the User Name (string).

The next line contains the Password (string).

### ***Output Format***

For each user, print the details in the following format:

User ID: <user\_id>

User Name: <user\_name>

Password: <password>

Password Strength: <Strong/Weak>

Refer to the sample output for formatting specifications.

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

Input: 1

1001

Ravi Kumar

Abc@1234

Output: User ID: 1001

User Name: Ravi Kumar

Password: Abc@1234

Password Strength: Strong

### **Answer**

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

```
class User {  
    private int userId;  
    private String userName;  
    private String password;  
  
    // Constructor  
    public User(int userId, String userName, String password) {  
        this.userId = userId;  
        this.userName = userName;  
        this.password = password;  
    }  
  
    // Getters  
    public int getUserId() {  
        return userId;  
    }  
  
    public String getUserName() {  
        return userName;  
    }  
  
    public String getPassword() {  
        return password;  
    }  
  
    // Setters  
    public void setUserId(int userId) {  
        this.userId = userId;  
    }  
  
    public void setUserName(String userName) {  
        this.userName = userName;  
    }  
}
```

```

public void setPassword(String password) {
    this.password = password;
}

// Method to check password strength
public String checkPasswordStrength() {
    if (password.length() < 8) return "Weak";

    boolean hasUpper = false;
    boolean hasLower = false;
    boolean hasDigit = false;
    boolean hasSpecial = false;

    String specialChars = "!@#$%^&*";

    for (int i = 0; i < password.length(); i++) {
        char ch = password.charAt(i);
        if (Character.isUpperCase(ch)) hasUpper = true;
        else if (Character.isLowerCase(ch)) hasLower = true;
        else if (Character.isDigit(ch)) hasDigit = true;
        else if (specialChars.indexOf(ch) >= 0) hasSpecial = true;
    }

    if (hasUpper && hasLower && hasDigit && hasSpecial) {
        return "Strong";
    } else {
        return "Weak";
    }
}
}

```

```

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

        int N = Integer.parseInt(sc.nextLine()); // number of users

        for (int i = 0; i < N; i++) {
            int id = Integer.parseInt(sc.nextLine());
            String name = sc.nextLine();
            String pwd = sc.nextLine();

```

```
User u = new User(id, name, pwd);

// Print details
System.out.println("User ID: " + u.getUserId());
System.out.println("User Name: " + u.getUserName());
System.out.println("Password: " + u.getPassword());
System.out.println("Password Strength: " + u.checkPasswordStrength());
}

sc.close();
}
```

**Status :** Correct

**Marks : 10/10**

## 2. Problem Statement

Each customer at the bank has an Account Number, Customer Name, and an Initial Balance. The bank allows two types of transactions:

Deposit – Increases the balance. Withdrawal – Decreases the balance, but only if enough funds are available. If the withdrawal amount exceeds the available balance, the transaction should be skipped, and the balance should remain unchanged.

You are required to implement this banking system by:

Creating a class with the necessary attributes to store account details.

Using a constructor to initialize the account details when a new account is created. Providing setter methods to update the details if required. Providing getter methods to retrieve account details. Creating objects of this class to represent different customers, where each customer can perform deposits and withdrawals.

Instructions:

Implement the class to store account details. Implement the logic for performing deposit and withdrawal transactions. Ensure that withdrawals don't exceed the available balance. After performing the transactions, print

the account number, customer name, and final balance.

### **Input Format**

The first line of input contains an integer N, representing the number of customers.

For each customer:

- The next line contains the account number (integer).
- The following line contains the customer name (string).
- The next line contains the initial balance (double).
- The next line contains the deposit amount (double).
- The next line contains the withdrawal amount (double).

### **Output Format**

For each customer, print the details in the following format:

1. Account Number: <account\_number>
2. Customer Name: <customer\_name>
3. Final Balance: <final\_balance> (rounded to one decimal place)

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 1

1234

Rahul Sharma

5000

2000

3000

Output: Account Number: 1234

Customer Name: Rahul Sharma

Final Balance: 4000.0

### **Answer**

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

```
class Account {
```

```
private int accountNumber;
private String customerName;
private double balance;

// Constructor
public Account(int accountNumber, String customerName, double balance) {
    this.accountNumber = accountNumber;
    this.customerName = customerName;
    this.balance = balance;
}

// Getters
public int getAccountNumber() {
    return accountNumber;
}

public String getCustomerName() {
    return customerName;
}

public double getBalance() {
    return balance;
}

// Setters
public void setAccountNumber(int accountNumber) {
    this.accountNumber = accountNumber;
}

public void setCustomerName(String customerName) {
    this.customerName = customerName;
}

public void setBalance(double balance) {
    this.balance = balance;
}

// Deposit method
public void deposit(double amount) {
    if (amount >= 0) {
        balance += amount;
    }
}
```

```

    }
    // Withdrawal method
    public void withdraw(double amount) {
        if (amount <= balance) {
            balance -= amount;
        }
    }
}

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

        int N = Integer.parseInt(sc.nextLine()); // number of customers

        for (int i = 0; i < N; i++) {
            int accNo = Integer.parseInt(sc.nextLine());
            String name = sc.nextLine();
            double initialBalance = Double.parseDouble(sc.nextLine());
            double depositAmount = Double.parseDouble(sc.nextLine());
            double withdrawalAmount = Double.parseDouble(sc.nextLine());

            Account acc = new Account(accNo, name, initialBalance);

            // Perform transactions
            acc.deposit(depositAmount);
            acc.withdraw(withdrawalAmount);

            // Print final details
            System.out.println("Account Number: " + acc.getAccountNumber());
            System.out.println("Customer Name: " + acc.getCustomerName());
            System.out.printf("Final Balance: %.1f%n", acc.getBalance());
        }

        sc.close();
    }
}

```

**Status :** Correct

**Marks : 10/10**

### 3. Problem Statement

Anjali is working as a developer for CityFitness Gym, which wants to build a system to calculate monthly membership fees for gym members based on the type of membership and the number of personal training sessions booked.

Each member's record has:

Member ID (integer) Member Name (string) Membership Type (string: "Basic", "Premium", "Elite") Number of Personal Training Sessions (integer)

The monthly fees are:

Basic – 1000 units Premium – 1500 units Elite – 2000 units

The cost of personal training sessions is 500 units per session.

The calculation rules:

Total Amount = Membership Fee + (Number of Personal Training Sessions × 500) If the number of sessions is more than 5, a 10% discount is applied on the total amount. If the member has Elite membership and the total amount exceeds 4000, an additional 5% service tax is added after discount.

Anjali has been asked to implement this system using:

A class with attributes for member details. A constructor to initialize member details. Getter and Setter methods to retrieve and update member details if required. A method to calculate the final monthly fee. Objects of the class to represent members.

Finally, display each member's details and the final monthly fee.

#### ***Input Format***

The first line contains an integer N, representing the number of members.

For each member:

- Next line contains Member ID (integer)
- Next line contains Member Name (string)



- Next line contains Membership Type ("Basic", "Premium", "Elite")
- Next line contains Number of Personal Training Sessions (integer)

### **Output Format**

For each member, print:

- Member ID: <member\_id>
- Member Name: <member\_name>
- Final Monthly Fee: <final\_fee> (The final fee must be rounded to one decimal place)

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 1

1001

Ravi Kumar

Basic

3

Output: Member ID: 1001

Member Name: Ravi Kumar

Final Monthly Fee: 2500.0

### **Answer**

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

```
class Member {  
    private int memberId;  
    private String memberName;  
    private String membershipType;  
    private int personalSessions;
```

```
// Constructor
```

```
    public Member(int memberId, String memberName, String membershipType,  
int personalSessions) {  
        this.memberId = memberId;  
        this.memberName = memberName;  
        this.membershipType = membershipType;  
        this.personalSessions = personalSessions;
```

```
}  
  
// Getters  
public int getMemberId() {  
    return memberId;  
}  
  
public String getMemberName() {  
    return memberName;  
}  
  
public String getMembershipType() {  
    return membershipType;  
}  
  
public int getPersonalSessions() {  
    return personalSessions;  
}  
  
// Setters  
public void setMemberId(int memberId) {  
    this.memberId = memberId;  
}  
  
public void setMemberName(String memberName) {  
    this.memberName = memberName;  
}  
  
public void setMembershipType(String membershipType) {  
    this.membershipType = membershipType;  
}  
  
public void setPersonalSessions(int personalSessions) {  
    this.personalSessions = personalSessions;  
}  
  
// Method to calculate final monthly fee  
public double calculateFinalFee() {  
    double membershipFee = 0;  
    switch (membershipType) {  
        case "Basic":  
            membershipFee = 1000;  
    }  
}
```

```

        break;
    case "Premium":
        membershipFee = 1500;
        break;
    case "Elite":
        membershipFee = 2000;
        break;
    }

    double total = membershipFee + (personalSessions * 500);

    // Apply 10% discount if more than 5 sessions
    if (personalSessions > 5) {
        total = total - (total * 0.10);
    }

    // Apply 5% service tax if Elite and total > 4000
    if (membershipType.equals("Elite") && total > 4000) {
        total = total + (total * 0.05);
    }

    return total;
}

}

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

        int N = Integer.parseInt(sc.nextLine()); // number of members

        for (int i = 0; i < N; i++) {
            int id = Integer.parseInt(sc.nextLine());
            String name = sc.nextLine();
            String type = sc.nextLine();
            int sessions = Integer.parseInt(sc.nextLine());

            Member m = new Member(id, name, type, sessions);

            // Print output as required
            System.out.println("Member ID: " + m.getMemberId());
            System.out.println("Member Name: " + m.getMemberName());

```

```
        System.out.printf("Final Monthly Fee: %.1f%n", m.calculateFinalFee());
    }
    sc.close();
}
```

**Status :** Correct

**Marks :** 10/10

#### 4. Problem Statement

Neha is working as a developer for CityQuiz Platform, which wants to build a system to calculate quiz scores and identify top scorers among participants.

Each participant's record has:

Participant ID (integer) Participant Name (string) An array of scores in 5 quiz rounds (integers, each between 0 and 100)

The system must calculate:

Total Score = sum of scores in all 5 rounds. Average Score = Total Score ÷ 5. If a participant scores above 80 in all rounds, a bonus of 10 points is added to the total score. Identify the Top Scorer among all participants. If two participants have the same total score, the one with the lower Participant ID is considered the top scorer.

Neha has been asked to implement this system using:

A class with attributes for participant details. A constructor to initialize participant details. Getter and setter methods to retrieve or update participant details. A method to calculate total score and average score (including bonus if applicable). Objects of the class to represent participants.

Finally, display each participant's details and announce the Top Scorer.

#### **Input Format**

The first line of input contains an integer N, representing the number of participants.

For each participant:

- Next line: Participant ID (integer)
- Next line: Participant Name (string)
- Next line: 5 integers separated by spaces (scores for 5 quiz rounds)

### **Output Format**

For each participant:

- Participant ID: <participant\_id>
- Participant Name: <participant\_name>
- Total Score: <total\_score>
- Average Score: <average\_score>

Finally, print "Top Scorer: <participant\_name> with <total\_score> points"

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 1

1001

Ravi Kumar

85 90 88 92 87

Output: Participant ID: 1001

Participant Name: Ravi Kumar

Total Score: 452

Average Score: 90

Top Scorer: Ravi Kumar with 452 points

### **Answer**

// You are using Java

import java.util.Scanner;

```
class Participant {  
    private int participantId;
```

```
private String participantName;
private int[] scores; // scores for 5 rounds
private int totalScore;
private int averageScore;

// Constructor
public Participant(int participantId, String participantName, int[] scores) {
    this.participantId = participantId;
    this.participantName = participantName;
    this.scores = scores;
    calculateScores();
}

// Getters
public int getParticipantId() {
    return participantId;
}

public String getParticipantName() {
    return participantName;
}

public int getTotalScore() {
    return totalScore;
}

public int getAverageScore() {
    return averageScore;
}

// Setters
public void setParticipantId(int participantId) {
    this.participantId = participantId;
}

public void setParticipantName(String participantName) {
    this.participantName = participantName;
}

public void setScores(int[] scores) {
    this.scores = scores;
    calculateScores();
}
```

```

    }
    // Calculate total and average score, including bonus if applicable
    private void calculateScores() {
        totalScore = 0;
        boolean bonusEligible = true;
        for (int score : scores) {
            totalScore += score;
            if (score <= 80) {
                bonusEligible = false;
            }
        }
        // Add bonus if all scores > 80
        if (bonusEligible) {
            totalScore += 10;
        }
        averageScore = totalScore / 5;
    }
}

```

```

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

        int N = Integer.parseInt(sc.nextLine());
        Participant[] participants = new Participant[N];

        for (int i = 0; i < N; i++) {
            int id = Integer.parseInt(sc.nextLine());
            String name = sc.nextLine();
            String[] scoreStr = sc.nextLine().split(" ");
            int[] scores = new int[5];
            for (int j = 0; j < 5; j++) {
                scores[j] = Integer.parseInt(scoreStr[j]);
            }

            participants[i] = new Participant(id, name, scores);
        }

        // Find top scorer
        Participant topScorer = participants[0];
        for (int i = 1; i < N; i++) {

```

```

        if (participants[i].getTotalScore() > topScorer.getTotalScore() ||
            (participants[i].getTotalScore() == topScorer.getTotalScore() &&
             participants[i].getParticipantId() < topScorer.getParticipantId())) {
            topScorer = participants[i];
        }
    }

    // Print participant details
    for (Participant p : participants) {
        System.out.println("Participant ID: " + p.getParticipantId());
        System.out.println("Participant Name: " + p.getParticipantName());
        System.out.println("Total Score: " + p.getTotalScore());
        System.out.println("Average Score: " + p.getAverageScore());
    }

    System.out.println("Top Scorer: " + topScorer.getParticipantName() +
        " with " + topScorer.getTotalScore() + " points");

    sc.close();
}
}

```

**Status :** Correct

**Marks :** 10/10

## 5. Problem Statement

Neha is working as a developer for CityMovie Theatre, which wants to build a system to calculate total ticket cost for movie-goers based on the number of tickets and type of seats booked.

Each customer's booking has:

Booking ID (integer) Customer Name (string) Number of Tickets (integer) Seat Type (string: "Standard", "Premium", "VIP")

The ticket prices are:

Standard – 250 units per ticket Premium – 400 units per ticket VIP – 600 units per ticket

The calculation rules:



Total Amount = Number of Tickets × Seat Price

If a customer books more than 4 tickets, they get a 10% discount on the total amount.

If the booking is for VIP seats and the total amount exceeds 3000 units, a 5% luxury tax is added after any discount.

Neha has been asked to implement this system using:

A class with attributes for booking details. A constructor to initialize booking details. Getter and Setter methods to retrieve and update booking details if required. A method to calculate the final ticket cost. Objects of the class to represent bookings.

Finally, display each customer's details and final ticket amount.

### ***Input Format***

The first line contains an integer N, representing the number of bookings.

For each booking:

- The next line contains the Booking ID (integer).
- The next line contains the Customer Name (string).
- The next line contains Number of Tickets (integer).
- The next line contains Seat Type ("Standard", "Premium", or "VIP").

### ***Output Format***

For each booking, print:

- Booking ID: <booking\_id>
- Customer Name: <customer\_name>
- Final Ticket Amount: <final\_amount> (rounded to one decimal place)

Refer to the sample output for formatting specifications.

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

Input: 1

1001

Ravi Kumar

3

Standard

Output: Booking ID: 1001

Customer Name: Ravi Kumar

Final Ticket Amount: 750.0

### **Answer**

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

```
class Booking {  
    private int bookingId;  
    private String customerName;  
    private int numberOfTickets;  
    private String seatType;  
  
    // Constructor  
    public Booking(int bookingId, String customerName, int numberOfTickets,  
String seatType) {  
        this.bookingId = bookingId;  
        this.customerName = customerName;  
        this.numberOfTickets = numberOfTickets;  
        this.seatType = seatType;  
    }  
  
    // Getters  
    public int getBookingId() {  
        return bookingId;  
    }  
  
    public String getCustomerName() {  
        return customerName;  
    }  
  
    public int getNumberOfTickets() {  
        return numberOfTickets;  
    }  
  
    public String getSeatType() {  
        return seatType;  
    }  
}
```

// Setters

```
public void setBookingId(int bookingId) {  
    this.bookingId = bookingId;  
}
```

```
public void setCustomerName(String customerName) {  
    this.customerName = customerName;  
}
```

```
public void setNumberOfTickets(int numberOfTickets) {  
    this.numberOfTickets = numberOfTickets;  
}
```

```
public void setSeatType(String seatType) {  
    this.seatType = seatType;  
}
```

// Method to calculate final ticket amount

```
public double calculateFinalAmount() {  
    double seatPrice = 0;  
    switch (seatType) {  
        case "Standard":  
            seatPrice = 250;  
            break;  
        case "Premium":  
            seatPrice = 400;  
            break;  
        case "VIP":  
            seatPrice = 600;  
            break;  
    }  
}
```

```
double total = numberOfTickets * seatPrice;
```

// Apply 10% discount if more than 4 tickets

```
if (numberOfTickets > 4) {  
    total = total - (total * 0.10);  
}
```

// Apply 5% luxury tax for VIP if total > 3000

```
if (seatType.equals("VIP") && total > 3000) {
```

```

        total = total + (total * 0.05);
    }

    return total;
}

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

        int N = Integer.parseInt(sc.nextLine()); // number of bookings

        for (int i = 0; i < N; i++) {
            int id = Integer.parseInt(sc.nextLine());
            String name = sc.nextLine();
            int tickets = Integer.parseInt(sc.nextLine());
            String seat = sc.nextLine();

            Booking booking = new Booking(id, name, tickets, seat);

            System.out.println("Booking ID: " + booking.getBookingId());
            System.out.println("Customer Name: " + booking.getCustomerName());
            System.out.printf("Final Ticket Amount: %.1f%n",
                booking.calculateFinalAmount());
        }

        sc.close();
    }
}

```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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 5\_CY

Attempt : 1

Total Mark : 40

Marks Obtained : 40

### Section 1 : Coding

#### 1. Problem Statement

Anjali is now working as a developer for the City Marathon Association, which wants to build a system to track and find the fastest runner among marathon participants.

Each runner's record has:

Runner ID (integer) Runner Name (string) An array of times (in minutes) taken in 5 marathon events (integers)

The system must calculate:

The average time of each runner (sum of all times / 5). Identify the fastest runner (the one with the lowest average time). If two or more runners have the same average time, the one with the lower Runner ID is considered the

fastest runner.

Anjali has been asked to implement this system using:

A class with attributes for runner details. A constructor to initialize runner details. Getter and Setter methods to retrieve and update runner details if required. A method to calculate the average time. Objects of the class to represent runners.

Finally, display each runner's details and announce the Fastest Runner.

### ***Input Format***

The first line of input contains an integer N (number of runners).

For each runner:

- The next line contains the Runner ID (integer).
- The following line contains the Runner Name (string).
- The next line contains 5 integers separated by spaces (times in minutes for 5 marathon events).

### ***Output Format***

For each runner the output prints the following details:

- Runner ID: <runner\_id>
- Runner Name: <runner\_name>
- Average Time: <average\_time>

Finally, print "Fastest Runner: <runner\_name> with <average\_time> minutes"

Refer to the sample output for formatting specifications.

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

Input: 1

1001

Ravi Kumar

240 250 245 255 260

Output: Runner ID: 1001

Runner Name: Ravi Kumar

Average Time: 250

Fastest Runner: Ravi Kumar with 250 minutes

### **Answer**

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

```
class Runner {  
    private int runnerId;  
    private String runnerName;  
    private int[] times; // times in 5 events  
    private int averageTime;  
  
    // Constructor  
    public Runner(int runnerId, String runnerName, int[] times) {  
        this.runnerId = runnerId;  
        this.runnerName = runnerName;  
        this.times = times;  
        calculateAverageTime();  
    }  
  
    // Getters  
    public int getRunnerId() {  
        return runnerId;  
    }  
  
    public String getRunnerName() {  
        return runnerName;  
    }  
  
    public int getAverageTime() {  
        return averageTime;  
    }  
  
    // Setters  
    public void setRunnerId(int runnerId) {  
        this.runnerId = runnerId;  
    }  
  
    public void setRunnerName(String runnerName) {
```

```

        this.runnerName = runnerName;
    }

    public void setTimes(int[] times) {
        this.times = times;
        calculateAverageTime();
    }

    // Method to calculate average time
    private void calculateAverageTime() {
        int sum = 0;
        for (int t : times) {
            sum += t;
        }
        averageTime = sum / times.length;
    }
}

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

        int N = Integer.parseInt(sc.nextLine());
        Runner[] runners = new Runner[N];

        for (int i = 0; i < N; i++) {
            int id = Integer.parseInt(sc.nextLine());
            String name = sc.nextLine();
            String[] timeStr = sc.nextLine().split(" ");
            int[] times = new int[5];
            for (int j = 0; j < 5; j++) {
                times[j] = Integer.parseInt(timeStr[j]);
            }
            runners[i] = new Runner(id, name, times);
        }

        // Find fastest runner
        Runner fastest = runners[0];
        for (int i = 1; i < N; i++) {
            if (runners[i].getAverageTime() < fastest.getAverageTime() ||
                (runners[i].getAverageTime() == fastest.getAverageTime() &&
                 runners[i].getRunnerId() < fastest.getRunnerId())) {

```



```

        fastest = runners[i];
    }
}

// Print runner details
for (Runner r : runners) {
    System.out.println("Runner ID: " + r.getRunnerId());
    System.out.println("Runner Name: " + r.getRunnerName());
    System.out.println("Average Time: " + r.getAverageTime());
}

System.out.println("Fastest Runner: " + fastest.getRunnerName() +
    " with " + fastest.getAverageTime() + " minutes");

sc.close();
}
}

```

**Status :** Correct

**Marks :** 10/10

## 2. Problem Statement

Anjali is working as a developer for the City Basketball Association, which wants to build a system to track and find the top scorer among basketball players.

Each player's record has:

Player ID (integer) Player Name (string) An array of points scored in 5 matches (integers)

The system must calculate:

The total score of each player (sum of all match points). Identify the highest scorer among all players. If two or more players have the same total score, the one with the lower Player ID is considered the top scorer.

Anjali has been asked to implement this system using:

A class with attributes for player details. A constructor to initialize player

details. Getter and Setter methods to retrieve and update player details if required. A method to calculate the total score. Objects of the class to represent players.

Finally, display each player's details and announce the Top Scorer.

### ***Input Format***

The first line of input contains an integer N (number of players).

For each player:

- The next line contains the Player ID (integer).
- The following line contains the Player Name (string).
- The next line contains 5 integers separated by spaces (points scored in 5 matches).

### ***Output Format***

For each player the output prints the following details:

- Player ID: <player\_id>
- Player Name: <player\_name>
- Total Score: <total\_score>

Finally, print "Top Scorer: <player\_name> with <total\_score> points"

Refer to the sample output for formatting specifications.

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

Input: 1

1001

Ravi Kumar

10 20 30 40 50

Output: Player ID: 1001

Player Name: Ravi Kumar

Total Score: 150

Top Scorer: Ravi Kumar with 150 points

**Answer**

```
import java.util.Scanner;

class Player {
    private int playerId;
    private String playerName;
    private int[] points; // points in 5 matches
    private int totalScore;

    // Constructor
    public Player(int playerId, String playerName, int[] points) {
        this.playerId = playerId;
        this.playerName = playerName;
        this.points = points;
        calculateTotalScore();
    }

    // Getters
    public int getPlayerId() {
        return playerId;
    }

    public String getPlayerName() {
        return playerName;
    }

    public int getTotalScore() {
        return totalScore;
    }

    // Setters
    public void setPlayerId(int playerId) {
        this.playerId = playerId;
    }

    public void setPlayerName(String playerName) {
        this.playerName = playerName;
    }

    public void setPoints(int[] points) {
```

```

        this.points = points;
        calculateTotalScore();
    }

    // Calculate total score
    private void calculateTotalScore() {
        totalScore = 0;
        for (int p : points) {
            totalScore += p;
        }
    }
}

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

        int N = Integer.parseInt(sc.nextLine());
        Player[] players = new Player[N];

        for (int i = 0; i < N; i++) {
            int id = Integer.parseInt(sc.nextLine());
            String name = sc.nextLine();
            String[] pointStr = sc.nextLine().split(" ");
            int[] points = new int[5];
            for (int j = 0; j < 5; j++) {
                points[j] = Integer.parseInt(pointStr[j]);
            }
            players[i] = new Player(id, name, points);
        }

        // Find top scorer
        Player topScorer = players[0];
        for (int i = 1; i < N; i++) {
            if (players[i].getTotalScore() > topScorer.getTotalScore() ||
                (players[i].getTotalScore() == topScorer.getTotalScore() &&
                 players[i].getPlayerId() < topScorer.getPlayerId())) {
                topScorer = players[i];
            }
        }

        // Print player details
    }
}

```

```

    for (Player p : players) {
        System.out.println("Player ID: " + p.getPlayerId());
        System.out.println("Player Name: " + p.getPlayerName());
        System.out.println("Total Score: " + p.getTotalScore());
    }

    System.out.println("Top Scorer: " + topScorer.getPlayerName() +
        " with " + topScorer.getTotalScore() + " points");

    sc.close();
}
}

```

**Status :** Correct

**Marks : 10/10**

### 3. Problem Statement

Arjun is working as a developer for CityWater Supply Board, which wants to build a household water billing system.

Each household's water account has:

A Customer ID (integer) A Customer Name (string) Liters Consumed (double)

The water bill is calculated based on these rules:

For the first 500 liters     2 per liter For the next 500 liters (501–1000)     3 per liter  
 For liters above 1000     5 per liter If the total bill exceeds 3000, a 10% discount is applied on the final bill.

Arjun has been asked to implement this system using:

A class with attributes for customer details. A constructor to initialize customer details. Setter methods to update details if needed. Getter methods to retrieve details. Objects of the class to represent customers.

Finally, display each customer's details and final bill amount.

**Input Format**

The first line of input contains an integer N, representing the number of customers.

For each customer:

- The next line contains the Customer ID (integer).
- The following line contains the Customer Name (string).
- The next line contains the Liters Consumed (double).

### ***Output Format***

For each customer, print the details in the following format:

Customer ID: <customer\_id>

Customer Name: <customer\_name>

Final Bill: <final\_bill> (rounded to one decimal place)

Refer to the sample output for formatting specifications.

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

Input: 1

1001

Ravi Kumar

300

Output: Customer ID: 1001

Customer Name: Ravi Kumar

Final Bill: 600.0

### ***Answer***

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

```
class Customer {  
    private int customerId;  
    private String customerName;  
    private double litersConsumed;  
  
    // Constructor  
    public Customer(int customerId, String customerName, double
```

```
litersConsumed) {
    this.customerId = customerId;
    this.customerName = customerName;
    this.litersConsumed = litersConsumed;
}

// Getters
public int getCustomerId() {
    return customerId;
}

public String getCustomerName() {
    return customerName;
}

public double getLitersConsumed() {
    return litersConsumed;
}

// Setters
public void setCustomerId(int customerId) {
    this.customerId = customerId;
}

public void setCustomerName(String customerName) {
    this.customerName = customerName;
}

public void setLitersConsumed(double litersConsumed) {
    this.litersConsumed = litersConsumed;
}

// Method to calculate final bill
public double calculateBill() {
    double bill = 0;
    double remaining = litersConsumed;

    // First 500 liters
    if (remaining > 0) {
        double firstTier = Math.min(remaining, 500);
        bill += firstTier * 2;
        remaining -= firstTier;
    }
}
```

```

    }

    // Next 500 liters (501-1000)
    if (remaining > 0) {
        double secondTier = Math.min(remaining, 500);
        bill += secondTier * 3;
        remaining -= secondTier;
    }

    // Above 1000 liters
    if (remaining > 0) {
        bill += remaining * 5;
    }

    // Apply 10% discount if bill > 3000
    if (bill > 3000) {
        bill = bill - (bill * 0.10);
    }

    return bill;
}

}

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

        int N = Integer.parseInt(sc.nextLine());
        Customer[] customers = new Customer[N];

        for (int i = 0; i < N; i++) {
            int id = Integer.parseInt(sc.nextLine());
            String name = sc.nextLine();
            double liters = Double.parseDouble(sc.nextLine());

            customers[i] = new Customer(id, name, liters);
        }

        // Print customer details and final bill
        for (Customer c : customers) {
            System.out.println("Customer ID: " + c.getCustomerId());
            System.out.println("Customer Name: " + c.getCustomerName());
        }
    }
}

```



```
        System.out.printf("Final Bill: %.1f%n", c.calculateBill());
    }
    sc.close();
}
}
```

**Status :** Correct

**Marks : 10/10**

#### 4. Problem Statement

You are working as a developer for CityMobile, which wants to build a basic mobile data usage management system.

Each customer has:

A Customer ID (integer) A Customer Name (string) An Initial Data Balance (in GB, double)

The company allows two types of operations:

Recharge – increases the data balance. Usage – decreases the data balance only if enough data is available.

If the usage amount is greater than the available data balance, the usage should not happen, and the balance should remain the same.

You are required to implement this system using:

A class with attributes for customer details. A constructor to initialize customer details. Setter methods to update details if needed. Getter methods to retrieve details. Objects of the class to represent customers.

Finally, display each customer's details after all operations.

#### **Input Format**

The first line of input contains an integer N, representing the number of customers.

For each customer:

- The next line contains the Customer ID (integer).
- The following line contains the Customer Name (string).
- The next line contains the Initial Data Balance (double).
- The next line contains the Recharge Amount in GB (double).
- The next line contains the Usage Amount in GB (double).

### **Output Format**

For each customer, print the details in the following format:

Customer ID: <customer\_id>

Customer Name: <customer\_name>

Final Data Balance: <final\_data\_balance> GB (The final balance must be rounded to one decimal place.)

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 1

1234

Ravi Kumar

5.0

2.0

3.0

Output: Customer ID: 1234

Customer Name: Ravi Kumar

Final Data Balance: 4.0 GB

### **Answer**

// You are using Java

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

```
class Customer {  
    private int customerId;  
    private String customerName;  
    private double dataBalance;
```

// Constructor

```
public Customer(int customerId, String customerName, double dataBalance) {  
    this.customerId = customerId;  
    this.customerName = customerName;  
    this.dataBalance = dataBalance;  
}
```

// Getters

```
public int getCustomerId() {  
    return customerId;  
}
```

```
public String getCustomerName() {  
    return customerName;  
}
```

```
public double getDataBalance() {  
    return dataBalance;  
}
```

// Setters

```
public void setCustomerId(int customerId) {  
    this.customerId = customerId;  
}
```

```
public void setCustomerName(String customerName) {  
    this.customerName = customerName;  
}
```

```
public void setDataBalance(double dataBalance) {  
    this.dataBalance = dataBalance;  
}
```

// Recharge method

```
public void recharge(double amount) {  
    if (amount >= 0) {  
        dataBalance += amount;  
    }  
}
```

// Usage method

```
public void useData(double amount) {
```

```

        if (amount <= dataBalance && amount >= 0) {
            dataBalance -= amount;
        }
    }
}

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

        int N = Integer.parseInt(sc.nextLine());
        Customer[] customers = new Customer[N];

        for (int i = 0; i < N; i++) {
            int id = Integer.parseInt(sc.nextLine());
            String name = sc.nextLine();
            double initialBalance = Double.parseDouble(sc.nextLine());
            double rechargeAmount = Double.parseDouble(sc.nextLine());
            double usageAmount = Double.parseDouble(sc.nextLine());

            customers[i] = new Customer(id, name, initialBalance);
            customers[i].recharge(rechargeAmount);
            customers[i].useData(usageAmount);
        }

        // Print customer details
        for (Customer c : customers) {
            System.out.println("Customer ID: " + c.getCustomerId());
            System.out.println("Customer Name: " + c.getCustomerName());
            System.out.printf("Final Data Balance: %.1f GB%n", c.getDataBalance());
        }

        sc.close();
    }
}

```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B  
Email: 241901507@rajalakshmi.edu.in  
Roll no: 241901507  
Phone: 9025753177  
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\_MCQ

Attempt : 1  
Total Mark : 15  
Marks Obtained : 15

#### Section 1 : MCQ

1. Select the correct keyword for implementing inheritance through the class.

**Answer**

extends

**Status : Correct**

**Marks : 1/1**

2. Which of the following is true about method overriding in Java?

**Answer**

The method must have the same name, same parameters, and must be in different classes with an inheritance relationship

**Status : Correct**

**Marks : 1/1**

3. What will be the output of the following code?

```
class A {  
    int sum(int x) {  
        return x + 2;  
    }  
}  
  
class B extends A {  
    int sum(int x) {  
        return super.sum(x) * 2;  
    }  
}  
  
class C extends B {  
    int sum(int x) {  
        return super.sum(x) - 3;  
    }  
}  
  
class Test {  
    public static void main(String[] args) {  
        C obj = new C();  
        System.out.println(obj.sum(4));  
    }  
}
```

**Answer**

9

**Status :** Correct

**Marks :** 1/1

4. What will be the output of the following code?

```
class A {  
    void display() {  
        System.out.println("Display A");  
    }  
}
```

```
class B extends A {  
    void display() {  
        System.out.println("Display B");  
    }  
}
```

```
class C extends B {  
    void display() {  
        super.display();  
    }  
}
```

```
class Test {  
    public static void main(String[] args) {  
        C obj = new C();  
        obj.display();  
    }  
}
```

**Answer**

Display B

**Status :** Correct

**Marks :** 1/1

5. What will be the output of the following Java program?

```
class A {  
    int value = 10;  
    void display() {  
        System.out.println("A's display: " + value);  
    }  
}  
class B extends A {  
    int value = 20;  
    void display() {  
        System.out.println("B's display: " + value);  
    }  
}
```

```
class Test {  
    public static void main(String[] args) {  
        A obj = new B();  
        obj.display();  
        System.out.println("Value: " + obj.value);  
    }  
}
```

**Answer**

B's display: 20 Value: 10

**Status :** Correct

**Marks :** 1/1

6. What will be the output of the following program?

```
class A {  
    int x = 10;  
}  
  
class B extends A {  
    int x = 20;  
}  
  
class C extends B {  
    int x = 30;  
  
    void display() {  
        System.out.println(x);  
        System.out.println(super.x);  
    }  
}  
  
class Test {  
    public static void main(String[] args) {  
        C obj = new C();  
        obj.display();  
    }  
}
```



**Answer**

3020

**Status :** Correct

**Marks :** 1/1

7. What will be the output of the following program?

```
class Vehicle {  
    String type = "Vehicle";  
}  
  
class Car extends Vehicle {  
    String type = "Car";  
}  
  
class Test {  
    public static void main(String[] args) {  
        Car c = new Car();  
        System.out.println(c.type);  
    }  
}
```

**Answer**

Car

**Status :** Correct

**Marks :** 1/1

8. What will be the output of the following program?

```
class A {  
    public int i;  
    private int j;  
}  
  
class B extends A {  
    void display() {  
        super.j = super.i + 1;  
        System.out.println(super.i + " " + super.j);  
    }  
}
```

```
}  
class inheritance {  
    public static void main(String args[]) {  
        B obj = new B();  
        obj.i=1;  
        obj.j=2;  
        obj.display();  
    }  
}
```

**Answer**

Compile Time Error

**Status :** Correct

**Marks :** 1/1

9. What will be the output of the following Java program?

```
class Vehicle {  
    void startEngine() {  
        System.out.println("Vehicle engine started");  
    }  
}
```

```
class Car extends Vehicle {  
    void startEngine() {  
        System.out.println("Car engine started");  
    }  
}
```

```
class Main {  
    public static void main(String[] args) {  
        Vehicle myVehicle = new Car();  
        myVehicle.startEngine();  
    }  
}
```

**Answer**

Car engine started

**Status :** Correct

**Marks :** 1/1

10. What will be the output of the following Java program?

```
class Parent {  
    void show() {  
        System.out.println("Parent class");  
    }  
}  
class Child extends Parent {  
    void show() {  
        System.out.println("Child class");  
    }  
}  
class Test {  
    public static void main(String[] args) {  
        Parent obj = new Child();  
        obj.show();  
    }  
}
```

**Answer**

Child class

**Status :** Correct

**Marks :** 1/1

11. What will be the output of the following Java program?

```
class Vehicle {  
    void start() {  
        System.out.println("Vehicle starts");  
    }  
}  
class Car extends Vehicle {  
  
    void start() {  
        System.out.println("Car starts");  
    }  
}  
class ElectricCar extends Car {  
    void start() {
```

```

        System.out.println("Electric Car starts silently");
    }
}
class Test {
    public static void main(String[] args) {
        Vehicle v = new ElectricCar();
        v.start();
    }
}

```

**Answer**

Electric Car starts silently

**Status :** Correct

**Marks :** 1/1

12. What will be the output of the following Java program?

```

class Test {
    void show(int a) {
        System.out.println("Integer method");
    }
    void show(String s) {
        System.out.println("String method");
    }
    public static void main(String[] args) {
        Test obj = new Test();
        obj.show(null);
    }
}

```

**Answer**

String method

**Status :** Correct

**Marks :** 1/1

13. What will be the output of the following Java program?

```

class A {
    void display() {

```

```

        System.out.println("Class A");
    }
}

class B extends A {
    void show() {
        System.out.println("Class B");
    }
}

class C extends B {
    void print() {
        System.out.println("Class C");
    }
}

class Test {
    public static void main(String[] args) {
        C obj = new C();
        obj.display();
        obj.show();
        obj.print();
    }
}

```

**Answer**

Class AClass BClass C

**Status :** Correct

**Marks :** 1/1

14. What will be the output of the following Java program?

```

class Test {
    void display(int a, int b) {
        System.out.println("Method 1");
    }
    void display(double a, double b) {
        System.out.println("Method 2");
    }
}

```

```
public static void main(String[] args) {  
    Test obj = new Test();  
    obj.display(10, 10.0);  
}  
}
```

**Answer**

Method 2

**Status :** Correct

**Marks :** 1/1

15. Which of the following is the correct way for class B to inherit from class A?

**Answer**

class B extends A {}

**Status :** Correct

**Marks :** 1/1

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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

### 2028\_REC\_OOPS using Java\_Week 6\_Q1

Attempt : 1

Total Mark : 10

Marks Obtained : 10

### Section 1 : Coding

#### 1. Problem Statement

Elsa subscribes to a premium service with a base monthly cost, a service tax and an extra feature cost. Assist her in writing an inheritance program that takes input for these values and calculates the total monthly cost.

Refer to the below class diagram:

#### ***Input Format***

The first line of input consists of a double value, representing the base monthly cost.

The second line consists of a double value, representing the service tax.

The third line consists of a double value, representing the extra feature cost.

### **Output Format**

The output prints "Rs. X" where X is a double value, rounded off to two decimal places.

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 10.0

2.5

5.0

Output: Rs. 17.50

### **Answer**

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

```
class Subscription {
```

```
    protected double baseCost;
```

```
    protected double serviceTax;
```

```
    public Subscription(double baseCost, double serviceTax) {
```

```
        this.baseCost = baseCost;
```

```
        this.serviceTax = serviceTax;
```

```
    }
```

```
    public double calculateMonthlyCost() {
```

```
        return baseCost + serviceTax;
```

```
    }
```

```
}
```

```
class PremiumSubscription extends Subscription {
```

```
    private double extraFeatureCost;
```

```
    public PremiumSubscription(double baseCost, double serviceTax, double  
extraFeatureCost) {
```

```
        super(baseCost, serviceTax);
```

```
        this.extraFeatureCost = extraFeatureCost;
```

```
    }
```

```
@Override
```



```
public double calculateMonthlyCost() {  
    return super.calculateMonthlyCost() + extraFeatureCost;  
}  
}  
  
public class Main {  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);  
  
        double baseMonthlyCost = scanner.nextDouble();  
        double serviceTax = scanner.nextDouble();  
        double extraFeatureCost = scanner.nextDouble();  
  
        PremiumSubscription premiumSubscription = new  
PremiumSubscription(baseMonthlyCost, serviceTax, extraFeatureCost);  
  
        double totalMonthlyCost = premiumSubscription.calculateMonthlyCost();  
  
        System.out.printf("Rs. %.2f%n", totalMonthlyCost);  
  
        scanner.close();  
    }  
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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

### 2028\_REC\_OOPS using Java\_Week 6\_Q2

Attempt : 1

Total Mark : 10

Marks Obtained : 10

### Section 1 : Coding

#### 1. Problem Statement

Alice is managing an online store and wants to implement a program using inheritance to calculate the selling price of products after applying discounts.

Guide her by following the instructions:

Create a base class called Product with a public double attribute price. Create a subclass called DiscountedProduct, which extends Product and includes a private double attribute discount rate. This subclass has a method called calculateSellingPrice() to determine the final selling price after applying the discount.

Formula: Discounted selling price = price \* (1 - discount rate)

**Input Format**

The first line of input consists of a double value p, the initial price of the product.

The second line consists of a double value d, the discount rate.

### **Output Format**

The output prints "Rs. X", where X is a double value, representing the calculated discounted selling price, rounded off to two decimal places.

If the discount rate is greater than 1, print "Not applicable".

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 50.00

0.20

Output: Rs. 40.00

### **Answer**

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

```
class Product {  
    public double price;  
  
    public Product(double price) {  
        this.price = price;  
    }  
}
```

```
// Subclass
```

```
class DiscountedProduct extends Product {  
    private double discountRate;  
  
    public DiscountedProduct(double price, double discountRate) {  
        super(price);  
        this.discountRate = discountRate;  
    }  
  
    public boolean isValidDiscount() {  
        return discountRate <= 1.0;  
    }  
}
```

```
}  
    public double calculateSellingPrice() {  
        return price * (1 - discountRate);  
    }  
}  
  
class ProductPricing {  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);  
  
        double initialPrice = scanner.nextDouble();  
        double discountRate = scanner.nextDouble();  
        DiscountedProduct discountedProduct = new  
DiscountedProduct(initialPrice, discountRate);  
        double sellingPrice = discountedProduct.calculateSellingPrice();  
  
        if (sellingPrice >= 0) {  
            System.out.printf("Rs. %.2f%n", sellingPrice);  
        } else {  
            System.out.println("Not applicable");  
        }  
        scanner.close();  
    }  
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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

### 2028\_REC\_OOPS using Java\_Week 6\_Q3

Attempt : 1

Total Mark : 10

Marks Obtained : 10

### Section 1 : Coding

#### 1. Problem Statement

Preethi is working on a project to automate sales tax calculations for items in a store. She wants to create a program that takes the price of an item and the sales tax rate as input and calculates the final price of the item after applying the sales tax.

Write a program using the class SalesTaxCalculator, which contains an overloaded method named calculateFinalPrice to handle both integer and double inputs. The program should also include a Main class that takes user input, calls the appropriate method from SalesTaxCalculator, and prints the final price of the item.

Formula Used: Final price = price + ((price \* sales tax rate) / 100)

**Input Format**

The first line of input consists of an integer price (the price of the item for integer inputs).

The second line of input consists of an integer taxRate (the sales tax rate for integer inputs).

The third line of input consists of a double price (the price of the item for double inputs).

The fourth line of input consists of a double taxRate (the sales tax rate for double inputs).

### ***Output Format***

The first line of output prints an integer, representing the final price of the item after applying the sales tax for integer inputs (a and b).

The second line prints a double value, representing the final price of the item after applying the sales tax for double-value inputs (m and n), rounded to two decimal places.

Refer to the sample output for formatting specifications.

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

Input: 100

10

100.0

5.0

Output: 110

105.00

### ***Answer***

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

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

```
// Class with overloaded methods
```

```
class SalesTaxCalculator {
```

```
// Overloaded method for integer inputs
```

```
    public static int calculateFinalPrice(int price, int taxRate) {
```

```
        return price + (price * taxRate) / 100;
    }

    // Overloaded method for double inputs
    public static double calculateFinalPrice(double price, double taxRate) {
        return price + (price * taxRate) / 100.0;
    }
}
```

```
class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int intPrice = scanner.nextInt();
        int intTaxRate = scanner.nextInt();
        double doublePrice = scanner.nextDouble();
        double doubleTaxRate = scanner.nextDouble();

        int finalPriceInt = SalesTaxCalculator.calculateFinalPrice(intPrice,
intTaxRate);
        double finalPriceDouble =
SalesTaxCalculator.calculateFinalPrice(doublePrice, doubleTaxRate);

        System.out.println(finalPriceInt);
        System.out.format("%.2f", finalPriceDouble);
    }
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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

### 2028\_REC\_OOPS using Java\_Week 6\_Q4

Attempt : 1

Total Mark : 10

Marks Obtained : 10

### Section 1 : Coding

#### 1. Problem Statement

Mr.Kapoor wants to create a program to calculate the volume of a Cuboid and a Cube using method overriding.

Implements a base class Cuboid with attributes for length, width, and height. Include a method calculateVolume() that computes the volume of the cuboid.

Extends the base class with a subclass Cube representing a cube, where all sides are equal. Override the calculateVolume() method in the Cube class to compute the volume of the cube.

The program should take user input for the dimensions of the cuboid and the side length of the cube and display the calculated volumes with two decimal places.



### ***Input Format***

The first line of input consists of 3 space-separated double values, representing the cuboid length, width, and height, respectively.

The second line consists of a double value, representing the side length of the cube.

### ***Output Format***

The first line of output prints the volume of the cuboid, rounded off to two decimal places.

The second line prints the volume of the cube, rounded off to two decimal places.

Refer to the sample output for formatting specifications.

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

Input: 60.0 60.0 60.0  
50.0

Output: Volume of Cuboid: 216000.00  
Volume of Cube: 125000.00

### ***Answer***

```
import java.util.Scanner;

class Cuboid {
    protected double length;
    protected double width;
    protected double height;

    // Constructor
    public Cuboid(double length, double width, double height) {
        this.length = length;
        this.width = width;
        this.height = height;
    }

    // Method to calculate volume of Cuboid
```

```

    public double calculateVolume() {
        return length * width * height;
    }
}

// Subclass Cube extending Cuboid
class Cube extends Cuboid {
    private double side;

    // Constructor (all sides are equal for a cube)
    public Cube(double side) {
        super(side, side, side); // call Cuboid constructor
        this.side = side;
    }

    // Overriding calculateVolume()
    @Override
    public double calculateVolume() {
        return side * side * side;
    }
}

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

        double cuboidLength = scanner.nextDouble();
        double cuboidWidth = scanner.nextDouble();
        double cuboidHeight = scanner.nextDouble();

        // Regular object instantiation for Cuboid
        Cuboid cuboid = new Cuboid(cuboidLength, cuboidWidth, cuboidHeight);
        System.out.printf("Volume of Cuboid: %.2f\n", cuboid.calculateVolume());

        double cubeSide = scanner.nextDouble();

        // Upcasting - Using superclass reference for subclass object (DMD)
        Cuboid cube = new Cube(cubeSide); // Upcasting
        System.out.printf("Volume of Cube: %.2f", cube.calculateVolume()); // Calls
        Cube's method dynamically

        scanner.close();
    }
}

```

}

**Status :** Correct

**Marks : 10/10**

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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

### 2028\_REC\_OOPS using Java\_Week 6\_Q5

Attempt : 1

Total Mark : 10

Marks Obtained : 10

### Section 1 : Coding

#### 1. Problem statement:

Tim was tasked with developing a grocery shopping app. You have a class hierarchy that includes Item, Produce, and OrganicProduce. Your goal is to calculate the total cost of a shopping list, which may contain a mix of regular produce and organic produce items. Additionally, you need to apply discounts to organic items. Apply a 10% discount on organic produce items

Class Hierarchy:

Item: Base class for all items.

Produce: Subclass of Item for regular produce items.

OrganicProduce: Subclass of Produce for organic produce items.

### ***Input Format***

The first line of input consists of an integer, 'n'.

For each 'n' item, the user will provide:

- A string 'type' representing the item type ('Regular' or 'Organic').
- A string 'name' represents the item name.
- A double 'price' represents the item price.

### ***Output Format***

The output will display the total cost of the shopping list, including discounts on organic items.

Refer to the sample output for format specifications.

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

Input: 1

Regular Banana 1.99

Output: 1.99

### ***Answer***

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

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

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

```
// Base class
```

```
abstract class Item {  
    protected String name;  
    protected double price;
```

```
  
    public Item(String name, double price) {  
        this.name = name;  
        this.price = price;  
    }
```

```
// Method to be overridden
```

```
public abstract double calculateCost();
```

```
}
```

```
// Subclass for regular produce
```

```
class Produce extends Item {  
    public Produce(String name, double price) {  
        super(name, price);  
    }  
}
```

```
@Override
```

```
public double calculateCost() {  
    return price; // No discount for regular items  
}
```

```
}
```

```
// Subclass for organic produce
```

```
class OrganicProduce extends Produce {  
    public OrganicProduce(String name, double price) {  
        super(name, price);  
    }  
}
```

```
@Override
```

```
public double calculateCost() {  
    return price * 0.90; // 10% discount  
}
```

```
}
```

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

```
        int n = sc.nextInt();  
        sc.nextLine(); // Consume newline
```

```
        double totalCost = 0.0;
```

```
        for (int i = 0; i < n; i++) {  
            String type = sc.next();  
            String name = sc.next();  
            double price = sc.nextDouble();
```

```
            if (type.equals("Regular")) {  
                Item item = new Produce(name, price);  
                totalCost += item.calculateCost();  
            }  
        }  
    }  
}
```

```
        } else if (type.equals("Organic")) {  
            Item item = new OrganicProduce(name, price);  
            totalCost += item.calculateCost();  
        }  
    }  
  
    System.out.printf("%.2f%n", totalCost);  
}  
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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\_PAH

Attempt : 1

Total Mark : 40

Marks Obtained : 40

### Section 1 : Coding

#### 1. Problem Statement

John is planning a long road trip and wants to calculate the distance his car can travel based on its speed and fuel capacity. As John knows that different cars have different fuel efficiencies, he wants a program that can help him estimate the travel distance for any given car.

To do this, you are tasked with creating a program that calculates the travel distance of a car based on its speed and fuel capacity. The calculation is simple and follows the formula:

Travel Distance = Speed \* Fuel Capacity

You need to model this system using a Vehicle class and a Car class. The Vehicle class will have attributes for the speed and fuel capacity, while the Car class will inherit from the Vehicle class and include a method to



calculate the travel distance.

### ***Input Format***

The first line of input consists of a double value representing the speed of the car in km/h.

The second line of input consists of a double value representing the fuel capacity of the car in liters.

### ***Output Format***

The first line should print "Speed: X km/h", where X is the speed of the car, rounded to two decimal places.

The second line should print "Fuel Capacity: Y liters", where Y is the fuel capacity of the car, rounded to two decimal places.

The third line should print "Travel Distance: Z km", where Z is the total travel distance the car can cover, rounded to two decimal places.

Refer to the sample output for formatting specifications.

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

Input: 10.0

1.0

Output: Speed: 10.00 km/h

Fuel Capacity: 1.00 liters

Travel Distance: 10.00 km

### ***Answer***

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

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

```
class Vehicle {
```

```
    double speed;
```

```
    double fuelCapacity;
```

```
// Constructor
```

```
Vehicle(double speed, double fuelCapacity) {
```

```
        this.speed = speed;
        this.fuelCapacity = fuelCapacity;
    }
}
```

```
// Derived class
class Car extends Vehicle {

    // Constructor calls parent constructor
    Car(double speed, double fuelCapacity) {
        super(speed, fuelCapacity);
    }

    // Method to calculate travel distance
    double calculateTravelDistance() {
        return speed * fuelCapacity;
    }
}

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

        double speed = scanner.nextDouble();
        double fuelCapacity = scanner.nextDouble();

        Car car = new Car(speed, fuelCapacity);

        System.out.println("Speed: " + String.format("%.2f", car.speed) + " km/h");
        System.out.println("Fuel Capacity: " + String.format("%.2f", car.fuelCapacity)
+ " liters");
        System.out.println("Travel Distance: " + String.format("%.2f",
car.calculateTravelDistance()) + " km");

        scanner.close();
    }
}
```

**Status :** Correct

**Marks :** 10/10

## 2. Problem Statement

Ram is designing a program to calculate the Body Mass Index (BMI). Your task is to assist him by following the given specifications.

Create a base class BMIcalculator with a method calculateBMI() to compute BMI using the formula  $\text{weight} / (\text{height} * \text{height})$ .

Extend the class with a subclass CustomBMIcalculator that overrides the method calculateBMI() to calculate BMI based on custom criteria, assigning categories such as "Underweight," "Normal Weight," "Overweight," or "Obese."

BMI < 18.5, category = "Underweight" BMI >= 18.5 & < 24.9, category = "Normal Weight" BMI >= 25 & < 29.9, category = "Overweight" else category = "Obese"

Implement user input for weight and height and display both the standard and custom BMI calculations.

#### ***Input Format***

The first line of input consists of a double value, representing the weight in kgs.

The second line consists of a double value, representing the height in meters.

#### ***Output Format***

The first line of output prints: "Standard BMI Calculation:"

The second line of output prints: "BMI: " followed by the calculated BMI value (to two decimal places).

The third line of output prints: "Custom BMI Calculation:"

The fourth line of output prints: "Category: " followed by the BMI category.

Refer to the sample output for formatting specifications.

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

Input: 69.7

2.6

Output: Standard BMI Calculation:

BMI: 10.31

Custom BMI Calculation:

Category: Underweight

**Answer**

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

```
class BMIcalculator {  
    double weight;  
    double height;
```

```
    // Constructor
```

```
    BMIcalculator(double weight, double height) {  
        this.weight = weight;  
        this.height = height;  
    }
```

```
    // Method to calculate BMI
```

```
    double calculateBMI() {  
        return weight / (height * height);  
    }
```

```
    // Method to display standard BMI
```

```
    void displayBMI() {  
        double bmi = calculateBMI();  
        System.out.println("BMI: " + String.format("%.2f", bmi));  
    }  
}
```

```
// Subclass for custom BMI calculation
```

```
class CustomBMIcalculator extends BMIcalculator {
```

```
    CustomBMIcalculator(double weight, double height) {  
        super(weight, height);  
    }
```

```
    // Method to display BMI category
```

```
    void displayCustomBMI() {  
        double bmi = calculateBMI();  
        String category;
```

```

        if (bmi < 18.5) {
            category = "Underweight";
        } else if (bmi >= 18.5 && bmi < 24.9) {
            category = "Normal Weight";
        } else if (bmi >= 25 && bmi < 29.9) {
            category = "Overweight";
        } else {
            category = "Obese";
        }

        System.out.println("Category: " + category);
    }
}

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

        double weight = scanner.nextDouble();
        double height = scanner.nextDouble();

        BMIcalculator bmiCalculator = new BMIcalculator(weight, height);
        System.out.println("Standard BMI Calculation:");
        bmiCalculator.displayBMI();

        CustomBMIcalculator customBMIcalculator = new
        CustomBMIcalculator(weight, height);
        System.out.println("Custom BMI Calculation:");
        customBMIcalculator.displayCustomBMI();

        scanner.close();
    }
}

```

**Status :** Correct

**Marks :** 10/10

### 3. Problem Statement

Sharon, a software developer, is working on a project to automate velocity calculations for various objects. She wants to create a class named

VelocityCalculator with overloaded methods calculateVelocity to calculate the velocity. One method will accept distance in meters and time in seconds as integers, while another will accept distance and time as doubles.

Help her in completing the project.

Formula:  $\text{Velocity} = \text{distance} / \text{time}$

### ***Input Format***

The first line of input consists of an integer, representing the distance in meters (for the integer method).

The second line consists of an integer, representing the time in seconds (for the integer method).

The third line consists of a double value, representing the distance in meters (for the double method).

The fourth line consists of a double value, representing the time in seconds (for the double method).

### ***Output Format***

The first line prints the velocity calculated using the integer inputs in the format:

Velocity with integer inputs: <velocity> m/s

The second line prints the velocity calculated using the double inputs in the format:

Velocity with double inputs: <velocity> m/s

Note:

The velocity for the double inputs should be printed with two decimal places.

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 100

10

100.5

10.2

Output: Velocity with integer inputs: 10 m/s

Velocity with double inputs: 9.85 m/s

### **Answer**

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

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

```
class VelocityCalculator {
```

```
    // Method for integer inputs
```

```
    static int calculateVelocity(int distance, int time) {
```

```
        return distance / time;
```

```
    }
```

```
    // Method for double inputs
```

```
    static double calculateVelocity(double distance, double time) {
```

```
        return distance / time;
```

```
    }
```

```
}
```

```
public class Main {
```

```
    public static void main(String[] args) {
```

```
        Scanner scanner = new Scanner(System.in);
```

```
        int distanceInt = scanner.nextInt();
```

```
        int timeInt = scanner.nextInt();
```

```
        double distanceDouble = scanner.nextDouble();
```

```
        double timeDouble = scanner.nextDouble();
```

```
        int velocityInt = VelocityCalculator.calculateVelocity(distanceInt, timeInt);
```

```
        double velocityDouble =
```

```
        VelocityCalculator.calculateVelocity(distanceDouble, timeDouble);
```

```
        System.out.println("Velocity with integer inputs: " + velocityInt + " m/s");
```

```
System.out.printf("Velocity with double inputs: %.2f m/s", velocityDouble);  
    scanner.close();  
}  
}
```

**Status :** Correct

**Marks :** 10/10

#### 4. Problem Statement

In a company, each manager has a unique employee ID and a monthly salary. You are required to design a program that will calculate and display the annual(12 months) salary of a manager based on the input details provided by the user.

Implement the solution using a single inheritance approach.

Employee: The base class with attributes name and employeeID.

Manager: The derived class inheriting from Employee, with an additional attribute salary.

##### ***Input Format***

The first line of input consists of a string name, representing the manager's name.

The second line of input consists of an integer employeeID, representing the manager's employee ID.

The third line of input consists of a double salary, representing the manager's monthly salary.

##### ***Output Format***

The first line of output prints: Name: <name>

The second line of output prints: Annual Salary: Rs. <annual\_salary> (rounded to two decimal places).



Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: Davis

234

28750.75

Output: Name: Davis

Annual Salary: Rs. 345009.00

### **Answer**

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

// Base class
class Employee {
    String name;
    int employeeID;

    // Constructor
    Employee(String name, int employeeID) {
        this.name = name;
        this.employeeID = employeeID;
    }
}

// Derived class
class Manager extends Employee {
    double salary; // Monthly salary

    // Constructor
    Manager(String name, int employeeID, double salary) {
        super(name, employeeID);
        this.salary = salary;
    }

    // Method to calculate annual salary
    double calculateAnnualSalary() {
        return salary * 12;
    }
}

class Main {
```

```
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    DecimalFormat df = new DecimalFormat("0.00");

    String name = scanner.nextLine();
    int employeeID = scanner.nextInt();
    double salary = scanner.nextDouble();

    Manager manager = new Manager(name, employeeID, salary);

    System.out.println("Name: " + manager.name);
    System.out.println("Annual Salary: Rs. " +
df.format(manager.calculateAnnualSalary()));
    scanner.close();
}
}
```

**Status :** Correct

**Marks : 10/10**

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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 : 2

Total Mark : 40

Marks Obtained : 40

### Section 1 : Coding

#### 1. Problem Statement

Teena is launching a new airline, Boeing747, and needs to calculate the total revenue generated from ticket sales based on the ticket cost and seat availability. Teena's airline offers two types of seats: regular and premium. The ticket cost and seat availability for both types of seats need to be considered for revenue calculation.

To help with this, Teena wants to implement a system using multilevel inheritance with three classes:

**Airline:** This class will have the ticket cost as an attribute and defines the method `setCost(double cost)` and `double getCost()`. **Indigo:** This class will extend **Airline** and add the seat availability attribute and defines the method `getSeatAvailability()` and `setSeatAvailability(int seatAvailability)`. **Boeing747:** This class will extend **Indigo** and include a

method `calculateTotalRevenue()` based on the ticket cost and seat availability .

Teena needs to calculate the total revenue using the formula:

$\text{Total Revenue} = \text{ticket cost} * \text{seat availability}$

Help Teena implement this system for calculating the revenue of her airline.

### ***Input Format***

The first line of input consists of a double value, representing the flight's ticket cost.

The second line consists of an integer, representing seat availability.

### ***Output Format***

The first line of output prints "Ticket Cost: Rs. " followed by a double value representing the ticket cost rounded to one decimal place.

The second line of output prints "Seat Availability: X seats" where X is an integer value representing the seat availability.

The third line of output prints "Total Revenue: Rs. " followed by a double value representing the total revenue rounded to one decimal place.

Refer to the sample output for the exact text and format.

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

Input: 1000.0  
100

Output: Ticket Cost: Rs. 1000.0  
Seat Availability: 100 seats  
Total Revenue: Rs. 100000.0

### ***Answer***

```
import java.util.Scanner;  
  
class Airline {  
    private double ticketCost;
```

```
public void setCost(double cost) {
    this.ticketCost = cost;
}

public double getCost() {
    return ticketCost;
}
}
```

// Derived class

```
class Indigo extends Airline {
    private int seatAvailability;
```

```
    public void setSeatAvailability(int seatAvailability) {
        this.seatAvailability = seatAvailability;
    }
```

```
    public int getSeatAvailability() {
        return seatAvailability;
    }
}
```

// Derived class from Indigo

```
class Boeing747 extends Indigo {
```

```
    public double calculateTotalRevenue() {
        return getCost() * getSeatAvailability();
    }
}
```

```
public class Main {
```

```
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Boeing747 plane = new Boeing747();
```

```
        double ticketCost = scanner.nextDouble();
        plane.setCost(ticketCost);
        int seatAvailability = scanner.nextInt();
        plane.setSeatAvailability(seatAvailability);
```

```
        System.out.printf("Ticket Cost: Rs. %.1f\n", plane.getCost());
```

```
        System.out.println("Seat Availability: " + plane.getSeatAvailability() + "
seats");
    }
}
```

```
        System.out.printf("Total Revenue: Rs. %.1f\n",  
plane.calculateTotalRevenue());  
    }  
}
```

**Status :** Correct

**Marks :** 10/10

## 2. Problem Statement

Bob has been tasked with creating a program using CircleUtils class to calculate and display the circumference and area of the circle.

The program should allow Bob to input the radius of a circle as both an integer and a double and compute both the circumference and area of the circle using separate overloaded methods:

calculateCircumference- To calculate the circumference using the formula  $2 * 3.14 * \text{radius}$

calculateArea- To calculate the area  $3.14 * \text{radius} * \text{radius}$

Write a program to help Bob.

### **Input Format**

The first line of input consists of an integer m, representing the radius of the circle as a whole number.

The second line consists of a double value n, representing the radius of the circle as a decimal number.

### **Output Format**

The first line of output displays two space-separated double values, rounded to two decimal places, representing the circumference of the circle with the integer radius and the double radius, respectively.

The second line displays two space-separated double values, rounded to two decimal places, representing the area of the circle with the integer radius and the double radius, respectively.

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 5

3.50

Output: 31.40 21.98

78.50 38.47

### **Answer**

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

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

```
class CircleUtils {
```

```
    public double calculateCircumference(int radius) {
```

```
        return 2 * 3.14 * radius;
```

```
    }
```

```
    public double calculateCircumference(double radius) {
```

```
        return 2 * 3.14 * radius;
```

```
    }
```

```
    public double calculateArea(int radius) {
```

```
        return 3.14 * radius * radius;
```

```
    }
```

```
    public double calculateArea(double radius) {
```

```
        return 3.14 * radius * radius;
```

```
    }
```

```
class Main {
```

```
    public static void main(String[] args) {
```

```
        Scanner scanner = new Scanner(System.in);
```

```
        int radiusInt = scanner.nextInt();
```

```
        double radiusDouble = scanner.nextDouble();
```

```
        CircleUtils circleUtils = new CircleUtils();
```

```
        double circumferenceInt = circleUtils.calculateCircumference(radiusInt);
```

```
        double circumferenceDouble =
```

```
        circleUtils.calculateCircumference(radiusDouble);
```

```
        double areaInt = circleUtils.calculateArea(radiusInt);
```

```

double areaDouble = circleUtils.calculateArea(radiusDouble);

System.out.format("%.2f %.2f\n", circumferenceInt, circumferenceDouble);
System.out.format("%.2f %.2f", areaInt, areaDouble);

scanner.close();
}
}

```

**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:

$$\text{Margin} = ((\text{Revenue} - \text{Cost}) / \text{Revenue}) \times 100$$

$$\text{Seasonal Margin} = \text{Margin} + 10$$

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

// Derived class
class SeasonalBusinessUtility extends BusinessUtility {
    @Override
    public double calculateMargin(double revenue, double cost) {
        double baseMargin = super.calculateMargin(revenue, cost);
```

```

        return baseMargin + 10; // seasonal adjustment of 10%
    }
}

// Class to check profitability
class ProfitabilityChecker {
    public void checkProfitability(double regularMargin) {
        if (regularMargin >= 10.0) {
            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

Teena's retail store has implemented a Loyalty Points System to reward

customers based on their spending. The program calculates and displays the loyalty points based on whether the customer is a regular or a premium customer.

For regular customers (class Customer), the loyalty points are calculated as:

$\text{Loyalty points} = \text{amount spent} / 10$

For premium customers (class PremiumCustomer, which inherits from Customer), the loyalty points are calculated as:

$\text{Loyalty points} = 2 * (\text{amount spent} / 10)$

The program should use method overriding for premium customers to calculate their loyalty points. The method that needs to be overridden is calculateLoyaltyPoints in the Customer class.

### ***Input Format***

The first line of input consists of an integer representing the amount spent by the customer.

The second line consists of a string representing the premium customer status:

- "yes" if the customer is a premium customer.
- "no" if the customer is not a premium customer.

### ***Output Format***

The output should display the loyalty points earned based on the amount spent and the customer type.

Refer to the sample output for formatting specifications.

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

Input: 50

yes

Output: 10

### ***Answer***

```
import java.util.Scanner;

class Customer {
    public int calculateLoyaltyPoints(int amountSpent) {
        return amountSpent / 10;
    }
}

class PremiumCustomer extends Customer {
    public int calculateLoyaltyPoints(int amountSpent) {
        return 2 * (amountSpent / 10);
    }
}

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

        int amountSpent = scanner.nextInt();

        String isPremium = scanner.next().toLowerCase();

        Customer customer;

        if (isPremium.equals("yes")) {
            customer = new PremiumCustomer();
        } else {
            customer = new Customer();
        }

        int loyaltyPoints = customer.calculateLoyaltyPoints(amountSpent);

        System.out.println(loyaltyPoints);
    }
}
```

**Status :** Correct

**Marks : 10/10**

# Rajalakshmi Engineering College

Name: Sachin Krishna B  
Email: 241901507@rajalakshmi.edu.in  
Roll no: 241901507  
Phone: 9025753177  
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 7\_MCQ

Attempt : 1  
Total Mark : 15  
Marks Obtained : 13

#### Section 1 : MCQ

1. Which of the following statements is true regarding default methods in Java interfaces?

**Answer**

A default method can be overridden in a class implementing the interface.

**Status : Correct**

**Marks : 1/1**

2. If a class implements two interfaces that have the same default method, what must the class do?

**Answer**

The class must declare the method as abstract.

**Status : Wrong**

**Marks : 0/1**

3. What is the output of the following code?

```
interface A {  
    static void display() {  
        System.out.println("Static method in A");  
    }  
}
```

```
class B implements A {  
    static void display() {  
        System.out.println("Static method in B");  
    }  
}
```

```
public class Main {  
    public static void main(String[] args) {  
        B.display();  
    }  
}
```

**Answer**

Static method in A

**Status : Wrong**

**Marks : 0/1**

4. What happens when an implementing class does not override a default method from an interface?

**Answer**

The default method's implementation from the interface will be used.

**Status : Correct**

**Marks : 1/1**

5. What is the primary purpose of static methods in Java interfaces?

**Answer**

They allow an interface to provide helper methods without requiring an implementing class.

**Status :** Correct

**Marks :** 1/1

6. What is the output of the following code?

```
interface X {  
    default void show() {  
        System.out.println("X's Default Method");  
    }  
}
```

```
interface Y {  
    default void show() {  
        System.out.println("Y's Default Method");  
    }  
}
```

```
class Z implements X, Y {  
    public void show() {  
        System.out.println("Z's Method");  
    }  
}
```

```
public class Main {  
    public static void main(String[] args) {  
        Z obj = new Z();  
        obj.show();  
    }  
}
```

**Answer**

Z's Method

**Status :** Correct

**Marks :** 1/1

7. How do you call a static method from an interface MyInterface?

**Answer**

MyInterface.staticMethod();

**Status :** Correct

**Marks :** 1/1

8. Can a Java interface contain both default and static methods?

**Answer**

Yes, an interface can have both default and static methods.

**Status :** Correct

**Marks :** 1/1

9. What is the output of the following code?

```
interface A {  
    default void show() {  
        System.out.println("A's Default Method");  
    }  
}
```

```
class B {  
    public void show() {  
        System.out.println("B's Method");  
    }  
}
```

```
class C extends B implements A {  
}
```

```
public class Main {  
    public static void main(String[] args) {  
        C obj = new C();  
        obj.show();  
    }  
}
```

**Answer**

B's Method

**Status :** Correct

**Marks :** 1/1



10. What is the output of the following code?

```
interface A {  
    default void show() {  
        System.out.println("A's Default Method");  
    }  
}
```

```
interface B {  
    default void show() {  
        System.out.println("B's Default Method");  
    }  
}
```

```
class C implements A, B {  
    public void show() {  
        A.super.show();  
    }  
}
```

```
public class Main {  
    public static void main(String[] args) {  
        C obj = new C();  
        obj.show();  
    }  
}
```

**Answer**

A's Default Method

**Status :** Correct

**Marks :** 1/1

11. Consider a class implementing an interface and extending a class, both having a method with the same name. Which method gets called?

**Answer**

The method from the superclass

**Status :** Correct

**Marks :** 1/1

12. Which of the following statements about Java interfaces is true?

**Answer**

A class can implement multiple interfaces.

**Status :** Correct

**Marks :** 1/1

13. What is the output of the following code?

```
interface MathOperations {  
    static int square(int x) {  
        return x * x;  
    }  
}  
  
public class Main {  
    public static void main(String[] args) {  
        System.out.println(MathOperations.square(5));  
    }  
}
```

**Answer**

25

**Status :** Correct

**Marks :** 1/1

14. How can a class explicitly call a default method from an interface if there is a naming conflict?

**Answer**

Using InterfaceName.super.methodName();

**Status :** Correct

**Marks :** 1/1

15. Which of the following is the correct way to declare an interface in Java?

**Answer**

```
interface Vehicle { void start();}
```

**Status :** Correct

**Marks :** 1/1

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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

### 2028\_REC\_OOPS using Java\_Week 7\_Q1

Attempt : 1

Total Mark : 10

Marks Obtained : 10

### Section 1 : Coding

#### 1. Problem Statement:

Rajiv is analyzing the energy consumption in his household and wants to calculate the total cost based on the daily energy usage. He is given the rate per unit of electricity and the energy consumed for multiple days. To structure this calculation efficiently, he decides to use an interface-based approach.

Implement an interface CostCalculator with the necessary methods to retrieve energy details and compute the cost. The calculations should be handled in the EnergyConsumptionTracker class, while the EnergyConsumptionApp class should only handle input and output.

Formula

Energy Cost for one day = Energy Consumed per day \* Rate Per Unit

### ***Input Format***

The first line of input consists of the rate per unit as an 'R' (a double value).

The second line of input consists of the number of days 'N' (an integer).

The third line of input consists of the daily energy consumption values for each day 'D' (double values), separated by space.

### ***Output Format***

The first line of the output prints: "Day-wise Energy Cost:"

The next N lines of the output print the day-wise energy costs(double type) and the total energy cost (double type) in Indian Rupees in the following format: "Day [day\_number]: Rs. [energy\_cost]"

The last line of the output prints: "Total Energy Cost: Rs. [total\_cost]"

Note: energy\_cost and total\_cost are rounded off to two decimal points

Refer to the sample output for the formatting specifications.

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

Input: 0.01

3

10.0 20.0 30.0

Output: Day-wise Energy Cost:

Day 1: Rs. 0.10

Day 2: Rs. 0.20

Day 3: Rs. 0.30

Total Energy Cost: Rs. 0.60

### ***Answer***

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

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

```
interface CostCalculator {  
    void getEnergyDetails(Scanner scanner);  
    void calculateAndDisplayCost();  
}
```

```
class EnergyConsumptionTracker implements CostCalculator {  
    private double ratePerUnit;  
    private int numDays;  
    private double[] dailyConsumption;
```

```
    public EnergyConsumptionTracker(double ratePerUnit, int numDays) {  
        this.ratePerUnit = ratePerUnit;  
        this.numDays = numDays;  
        this.dailyConsumption = new double[numDays];  
    }
```

```
    @Override  
    public void getEnergyDetails(Scanner scanner) {  
        for (int i = 0; i < numDays; i++) {  
            dailyConsumption[i] = scanner.nextDouble();  
        }  
    }
```

```
    @Override  
    public void calculateAndDisplayCost() {  
        System.out.println("Day-wise Energy Cost:");  
        double totalCost = 0.0;  
        for (int i = 0; i < numDays; i++) {  
            double dayCost = dailyConsumption[i] * ratePerUnit;  
            totalCost += dayCost;  
            System.out.printf("Day %d: Rs. %.2f%n", i + 1, dayCost);  
        }  
        System.out.printf("Total Energy Cost: Rs. %.2f%n", totalCost);  
    }  
}
```

```
class EnergyConsumptionApp {  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);  
  
        double ratePerUnit = scanner.nextDouble();  
        int numDays = scanner.nextInt();
```

```
CostCalculator tracker = new EnergyConsumptionTracker(ratePerUnit,  
numDays);
```

```
    tracker.getEnergyDetails(scanner);  
    tracker.calculateAndDisplayCost();
```

```
    scanner.close();
```

```
    }  
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B  
Email: 241901507@rajalakshmi.edu.in  
Roll no: 241901507  
Phone: 9025753177  
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

### 2028\_REC\_OOPS using Java\_Week 7\_Q2

Attempt : 1  
Total Mark : 10  
Marks Obtained : 10

#### Section 1 : Coding

##### 1. Problem Statement

Jaheer is working on a health monitoring system to help individuals calculate their Body Mass Index (BMI). He has implemented a basic BMI calculator and an interface called HealthCalculator. It should have a method called calculateBMI.

You are tasked with creating a program that takes weight and height as input, calculates the BMI using the BMICalculator class, and displays the result. If the height or weight is less than or equal to zero, then return -1.

Formula:  $BMI = \text{weight} / (\text{height} * \text{height})$

##### ***Input Format***

The first line of input consists of a double value W, the person's weight in kilograms.



The second line consists of a double value H, the height of the person in meters.

### **Output Format**

The output displays "BMI: " followed by a double value, representing the calculated BMI, rounded off to two decimal places.

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 70.0

1.75

Output: BMI: 22.86

### **Answer**

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

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

```
interface HealthCalculator {  
    double calculateBMI(double weight, double height);  
}
```

```
class BMICalculator implements HealthCalculator {  
    @Override  
    public double calculateBMI(double weight, double height) {  
        if (weight <= 0 || height <= 0) {  
            return -1;  
        }  
        return weight / (height * height);  
    }  
}
```

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

```
BMI Calculator bmiCalculator = new BMI Calculator();  
double bmi = bmiCalculator.calculateBMI(weight, height);  
System.out.printf("BMI: %.2f\n", bmi);  
  
    scanner.close();  
}  
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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

### 2028\_REC\_OOPS using Java\_Week 7\_Q3

Attempt : 1

Total Mark : 10

Marks Obtained : 10

### Section 1 : Coding

#### 1. Problem Statement

A financial analyst, Alex, needs a program to calculate simple interest for various financial transactions. He requires a straightforward tool that takes in the principal amount, interest rate, and time in years and computes the interest.

The formula to be used is:  $\text{Interest} = \text{Principal} \times \text{Rate} \times \text{Time} / 100$

Implement this functionality using the InterestCalculator interface and the SimpleInterestCalculator class.

#### **Input Format**

The first line of input consists of the principal amount P as a double value.

The second line of input consists of the annual interest rate  $r$  as a double value.

The third line of input consists of the number of years  $t$  as a positive integer, which is an integer value.

### ***Output Format***

The output displays the calculated simple interest in the following format: "Simple Interest: [interest\_value]", Here, [interest\_value] should be replaced with the actual interest value calculated by the program.

Refer to the sample output for the formatting specifications.

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

Input: 1000.00

5.00

2

Output: Simple Interest: 100.0

### ***Answer***

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

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

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

```
interface InterestCalculator
```

```
{
```

```
    double simpleInterest(double principal, double rate, int time);
```

```
}
```

```
class SimpleInterestCalculator implements InterestCalculator
```

```
{
```

```
    public double simpleInterest(double principal, double rate, int time)
```

```
{  
    return (principal * rate * time) / 100.0;  
}  
}  
class Main {  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);  
        double principal = scanner.nextDouble();  
        double rate = scanner.nextDouble();  
        int time = scanner.nextInt();  
        InterestCalculator calculator = new SimpleInterestCalculator();  
        double interest = calculator.simpleInterest(principal, rate, time);  
        System.out.println("Simple Interest: " + interest);  
    }  
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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

### 2028\_REC\_OOPS using Java\_Week 7\_Q4

Attempt : 1

Total Mark : 10

Marks Obtained : 10

### Section 1 : Coding

#### 1. Problem Statement

Maria, a software developer, is working on an inventory management system project using Java that utilizes an inventory interface to manage a store's products.

The interface should define two methods: `addProduct`, which adds a product by accepting its name, price, and quantity, and `calculateTotalValue`, which computes the total value of all products in the inventory. Implement the interface in a class called `SimpleInventory`, which internally manages a list of `Product` objects.

Each `Product` object should encapsulate the product's name, price, and quantity and include a method to calculate its value as  $\text{price} \times \text{quantity}$ . The system should allow users to dynamically add products to the inventory and calculate the total value of all products stored.

Help Maria achieve the task.

### ***Input Format***

The first line of input consists of an integer to choose one of the following options:

- 1 - to add a product to the inventory.
- 2 - to calculate and view the total inventory value.
- 3 - to exit the program.

For Choice 1 (Add Product):

The next input line is the string representing the product name as a string (single or multi-word, without quotes).

The next line is a double value representing the price as a decimal value

The next line is an integer value representing the quantity as an integer

For Choices 2 and 3, no additional input is required

### ***Output Format***

The output displays the results of the commands as follows:

- For the addProduct command, the program should display "Product added to inventory."
- For choice 2, the program should display "Total inventory value [totalvalue].  
"The total value should be displayed with one decimal place. If there is no product in the inventory, print the total as 0.0.
- For choice 3, the program should exit

If the choice is not 1, 2, or 3, then print "Invalid choice. Please select a valid option (1/2/3).".

Refer to the sample output for the formatting specifications.

### **Sample Test Case**

Input: 1

Laptop

800.0

3

2

5

3

Output: Product added to inventory.

Total inventory value: \$2400.0

Invalid choice. Please select a valid option (1/2/3).

### **Answer**

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

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

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

```
import java.util.ArrayList;
```

```
import java.util.List;
```

```
interface Inventory
```

```
{
```

```
    void addProduct(String name, double price, int quantity);
```

```
    double calculateTotalValue();
```

```
}
```

```
class Product
```

```
{
```

```
    private String name;
```

```
    private double price;
```

```
    private int quantity;
```

```
    public Product(String name, double price, int quantity)
```

```
{
```



```
    this.name = name;  
    this.price = price;  
    this.quantity = quantity;
```

```
}
```

```
    public double getValue()
```

```
{
```

```
        return price * quantity;
```

```
}
```

```
}
```

```
class SimpleInventory implements Inventory
```

```
{
```

```
    private List<Product> products;
```

```
    public SimpleInventory(int capacity)
```

```
{
```

```
        products = new ArrayList<>(capacity);
```

```
}
```

```
    public void addProduct(String name, double price, int quantity)
```

```
{
```

```
products.add(new Product(name, price, quantity));  
System.out.println("Product added to inventory.");
```

```
}
```

```
public double calculateTotalValue()
```

```
{
```

```
    double total = 0.0;  
    for (Product p : products)
```

```
{
```

```
        total += p.getValue();
```

```
}
```

```
    return Math.round(total * 10) / 10.0; // round to 1 decimal place
```

```
}
```

```
}
```

```
public class Main {
```

```
    public static void main(String[] args) {
```

```
        Scanner scanner = new Scanner(System.in);
```

```
        Inventory inventory = new SimpleInventory(10);
```

```
        while (true) {
```

```
            int choice = scanner.nextInt();
```

```
            if (choice == 1) {
```

```
                scanner.nextLine();
```

```
                String productName = scanner.nextLine();
```

```
                double price = scanner.nextDouble();
```

```
                int quantity = scanner.nextInt();
```

```
                inventory.addProduct(productName, price, quantity);
```

```
            } else if (choice == 2) {
```

```
                double totalValue = inventory.calculateTotalValue();
```

```
        System.out.println("Total inventory value: $" + totalValue);
    } else if (choice == 3) {
        break;
    } else {
        System.out.println("Invalid choice. Please select a valid option
(1/2/3).");
    }
}
scanner.close();
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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

### 2028\_REC\_OOPS using Java\_Week 7\_Q5

Attempt : 1

Total Mark : 10

Marks Obtained : 10

### Section 1 : Coding

#### 1. Problem Statement

Raj is curious about how old he is in the current year.

He has asked you to create a simple program that calculates a person's age based on their birth year. You decide to implement this functionality using the AgeCalculator interface and the HumanAgeCalculator class.

Note: The current year is 2024. Calculate the current age by using the formula: current year - birth year.

#### ***Input Format***

The input consists of an integer representing the birth year.

#### ***Output Format***

The output displays "You are X years old." where X is an integer representing the calculated age based on the entered birth year.

Refer to the sample output for formatting specifications.

**Sample Test Case**

Input: 1934

Output: You are 90 years old.

**Answer**

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

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

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

```
interface AgeCalculator
```

```
{
```

```
    int calculateAge(int birthYear);
```

```
}
```

```
class HumanAgeCalculator implements AgeCalculator
```

```
{
```

```
    private static final int CURRENT_YEAR = 2024;
```

```
    public int calculateAge(int birthYear)
```

```
{
```

```
        return CURRENT_YEAR - birthYear;
```

```
}  
}  
class AgeCalculatorApp {  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);  
  
        AgeCalculator ageCalculator = new HumanAgeCalculator();  
  
        int birthYear = scanner.nextInt();  
        int age = ageCalculator.calculateAge(birthYear);  
  
        System.out.println("You are " + age + " years old.");  
    }  
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B  
Email: 241901507@rajalakshmi.edu.in  
Roll no: 241901507  
Phone: 9025753177  
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 7\_PAH

Attempt : 1  
Total Mark : 40  
Marks Obtained : 30

#### Section 1 : Coding

##### 1. Problem Statement

Develop a program for managing employee information that caters to both full-time and part-time employees. The program should be capable of computing the salary for each category of employee and presenting their particulars. To achieve this, create two classes, FullTimeEmployee and PartTimeEmployee, that adhere to the Employee interface.

The program is expected to accept input data, including the name and monthly salary for full-time employees, as well as the name, hourly rate, and hours worked for part-time employees. Subsequently, it should calculate and exhibit the employee details and their respective salaries.

For Full-Time employees, the annual salary should be calculated as 12 times the monthly salary.

For Part-Time employees, the salary calculation should be based on the formula: hourly rate \* hours worked.

### ***Input Format***

The first line of input should be a string representing the name of a full-time employee.

The second line of input should be an integer representing the monthly salary of the full-time employee.

The third line of input should be a string representing the name of a part-time employee.

The fourth line of input should be an integer representing the hourly rate of the part-time employee.

The fifth line of input should be an integer representing the number of hours worked by the part-time employee.

### ***Output Format***

The output displays the following details:

#### Full-Time Employee Details:

Name: [Full-Time Employee Name] (string)

Monthly Salary: \$[Monthly Salary] (integer)

Annual Salary: \$[12 times Monthly Salary] (integer)

#### Part-Time Employee Details:

Name: [Part-Time Employee Name] (string)

Hourly Rate: \$[Hourly Rate] (integer)



Hours Worked: [Hours Worked] hours (integer)

Monthly Salary: \$[Calculated Monthly Salary] (integer)

Refer to the sample output for the formatting specifications.

### **Sample Test Case**

Input: John Smith

15000

Mary Johnson

100

100

Output: Full-Time Employee Details:

Name: John Smith

Monthly Salary: \$15000

Annual Salary: \$180000

Part-Time Employee Details:

Name: Mary Johnson

Hourly Rate: \$100

Hours Worked: 100 hours

Monthly Salary: \$10000

### **Answer**

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

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

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

```
interface Calculator
```

```
{
```

```
    double add(double a, double b);
```

```
    double subtract(double a, double b);
```

```
}
```

```
class SimpleCalculator implements Calculator
```

```
{
```

```
    public double add(double a, double b)
```

```
    {
```

```
        return a + b;
```

```
    }
```

```
    public double subtract(double a, double b)
```

```
    {
```

```
        return a - b;
```

```
    }
```

```
}
```

```
class EmployeeInheritanceDemo {
```

```
    public static void main(String[] args) {
```

```
        Scanner scanner = new Scanner(System.in);
```

```
        String fullName = scanner.nextLine();
```

```
        int fullTimeSalary = scanner.nextInt();
```

```
        scanner.nextLine();
```

```
        String partTimeName = scanner.nextLine();
```

```
        int hourlyRate = scanner.nextInt();
```

```
        int hoursWorked = scanner.nextInt();
```

```
        FullTimeEmployee fullTimeEmployee = new FullTimeEmployee(fullName,  
fullTimeSalary);
```

```
        PartTimeEmployee partTimeEmployee = new  
PartTimeEmployee(partTimeName, hourlyRate, hoursWorked);
```

```
        fullTimeEmployee.displayDetails();
```

```
        System.out.println();
```

```
partTimeEmployee.displayDetails();
scanner.close();
}
}
```

**Status :** Wrong

**Marks :** 0/10

## 2. Problem Statement

Oviya is fascinated by automorphic numbers and wants to create a program to determine whether a given number is an automorphic number or not.

An automorphic number is a number whose square ends with the same digits as the number itself. For example,  $25 = (25)^2 = 625$

Oviya has defined two interfaces: NumberInput for taking user input and AutomorphicChecker for checking if a given number is automorphic. The class AutomorphicNumber implements both interfaces.

Help her complete the task.

### ***Input Format***

The input consists of a single integer n.

### ***Output Format***

If the input number is an automorphic number, print "n is an automorphic number". Otherwise, print "n is not an automorphic number".

Refer to the sample output for formatting specifications.

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

Input: 25

Output: 25 is an automorphic number

### ***Answer***

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

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

```
interface NumberInput
```

```
{
```

```
    int getInput();
```

```
}
```

```
interface AutomorphicChecker
```

```
{
```

```
    boolean checkAutomorphic(int n);
```

```
}
```

```
class AutomorphicNumber implements NumberInput, AutomorphicChecker
```

```
{
```

```
    public int getInput()
```

```
{
```

```
        Scanner scanner = new Scanner(System.in);  
        return scanner.nextInt();
```

```
}
```

```
    public boolean checkAutomorphic(int n)
```

```
{
```

```

        int square = n * n;
        String strN = String.valueOf(n);
        String strSquare = String.valueOf(square);
        // Check if square ends with n
        return strSquare.endsWith(strN);
    }

}

public class Main {
    public static void main(String[] args) {
        AutomorphicNumber automorphicNumber = new AutomorphicNumber();
        int inputNumber = automorphicNumber.getInput();

        boolean isAutomorphic =
            automorphicNumber.checkAutomorphic(inputNumber);

        if (isAutomorphic) {
            System.out.println(inputNumber+" is an automorphic number");
        } else {
            System.out.println(inputNumber+" is not an automorphic number");
        }
    }
}

```

**Status :** Correct

**Marks :** 10/10

### 3. Problem Statement

Sophia is developing a matrix analysis tool for a data analytics company. The tool needs to analyze square matrices and extract insights from the matrix diagonals.

To organize the code properly, Sophia creates an interface named Matrix that declares a method for finding the smallest and largest elements along the principal and secondary diagonals of the matrix.

Sophia then creates a class named MatrixAnalyzer that implements the Matrix interface. This class provides the logic to process a given square

matrix and print:

The smallest and largest elements in the principal diagonal (from top-left to bottom-right). The smallest and largest elements in the secondary diagonal (from top-right to bottom-left).

Your task is to implement the Matrix interface and the MatrixAnalyzer class. The main driver program (in the class Main) will read the input matrix, create an instance of MatrixAnalyzer, and invoke its method to display the results.

### ***Input Format***

The first line contains an integer n, representing the size of the square matrix.

The next n lines each contain n integers separated by spaces, representing the elements of the matrix.

### ***Output Format***

The output prints the four lines:

"Smallest Element - 1: <smallest element in the principal diagonal>" (integer)

"Largest Element - 1: <largest element in the principal diagonal>" (integer)

"Smallest Element - 2: <smallest element in the secondary diagonal>" (integer)

"Largest Element - 2: <largest element in the secondary diagonal>" (integer)

Refer to the sample output for the formatting specifications.

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

Input: 5

7 8 9 0 1

2 3 4 5 6

5 4 2 0 8

23 5 6 8 9

12 5 6 7 32

Output: Smallest Element - 1: 2

Largest Element - 1: 32

Smallest Element - 2: 1

Largest Element - 2: 12

**Answer**

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

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

```
interface Matrix {  
    void diagonalsMinMax(int[][] matrix);  
}
```

```
class MatrixAnalyzer implements Matrix {
```

```
    public void diagonalsMinMax(int[][] matrix) {  
        int n = matrix.length;
```

```
        int principalMin = matrix[0][0];  
        int principalMax = matrix[0][0];  
        int secondaryMin = matrix[0][n - 1];  
        int secondaryMax = matrix[0][n - 1];
```

```
        for (int i = 0; i < n; i++) {  
            int principalElement = matrix[i][i];  
            int secondaryElement = matrix[i][n - 1 - i];
```

```
            if (principalElement < principalMin) principalMin = principalElement;  
            if (principalElement > principalMax) principalMax = principalElement;  
            if (secondaryElement < secondaryMin) secondaryMin =  
secondaryElement;  
            if (secondaryElement > secondaryMax) secondaryMax =  
secondaryElement;  
        }
```

```
        System.out.println("Smallest Element - 1: " + principalMin);  
        System.out.println("Largest Element - 1: " + principalMax);  
        System.out.println("Smallest Element - 2: " + secondaryMin);  
        System.out.println("Largest Element - 2: " + secondaryMax);
```

```
    }  
}
```

```
public class Main {
```

```

public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int n = sc.nextInt();
    int[][] matrix = new int[n][n];
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            matrix[i][j] = sc.nextInt();
        }
    }
    MatrixAnalyzer analyzer = new MatrixAnalyzer();
    analyzer.diagonalsMinMax(matrix);
}
}

```

**Status :** Correct

**Marks :** 10/10

#### 4. Problem Statement:

Alice has been tasked with implementing a simple calculator interface and a corresponding class for performing basic addition and subtraction operations. The task is to create an interface called Calculator with two methods: add and subtract. The add method should take two numbers as input and return their sum, while the subtract method should take two numbers as input and return their difference.

Implement a class called SimpleCalculator that implements the Calculator interface. This class should provide the functionality for adding and subtracting numbers. Write a code that satisfies the above requirements.

#### **Input Format**

The first line of input consists of a single integer, representing the choice

If the choice is 1 or 2, the next two lines consist of 2 double values, representing the numbers to do addition or subtraction.

#### **Output Format**

The output prints a float-value with one decimal value representing the sum of two number or difference of two number.



Refer to the sample output for format specification.

**Sample Test Case**

Input: 1

5.5

3.5

Output: Result: 9.0

**Answer**

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

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

```
interface Calculator
```

```
{
```

```
    double add(double a, double b);
```

```
    double subtract(double a, double b);
```

```
}
```

```
class SimpleCalculator implements Calculator
```

```
{
```

```
    public double add(double a, double b)
```

```
{
```

```
        return a + b;
```

```
}
```

```
    public double subtract(double a, double b)
```

```

    {
        return a - b;
    }
}

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

        SimpleCalculator calculator = new SimpleCalculator();

        int choice = scanner.nextInt();

        if (choice == 1) {
            double num1 = scanner.nextDouble();
            double num2 = scanner.nextDouble();
            double result = calculator.add(num1, num2);
            System.out.println("Result: " + result);
        } else if (choice == 2) {
            double num1 = scanner.nextDouble();
            double num2 = scanner.nextDouble();
            double result = calculator.subtract(num1, num2);
            System.out.println("Result: " + result);
        } else {
            System.out.println("Invalid choice. Please choose 1 for addition or 2 for subtraction.");
        }

        scanner.close();
    }
}

```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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 7\_CY

Attempt : 1

Total Mark : 40

Marks Obtained : 40

### Section 1 : Coding

#### 1. 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;
```

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

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

```
class WashingMachine implements Motor {
```

```
    public void run() {
```

```

        System.out.println("Washing machine is running.");
    }

    // Method to indicate the machine is consuming electricity
    public void consume() {
        System.out.println("Washing machine is consuming electricity.");
    }

    // Method to calculate electricity consumption
    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

## 2. 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;
```

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

```
// Interface for tax calculation
```

```
interface TaxCalculator {  
    int calculateTax(int salary);  
}
```

```
// Implementation class for progressive tax calculation
```

```
class SimpleTaxCalculator implements TaxCalculator {  
    @Override  
    public int calculateTax(int salary) {  
        double tax;
```

```
        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

### 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;
```

```
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;
```

```
}
```

```
    @Override
```

```
    public double getPrincipal()
```

```
{
```

```
        return principal;
```

```
}
```

```
    @Override
```

```
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

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 budget;  
    private double totalExpenses;
```

```
    public ExpenseTracker(double budget) {
```

```

        this.budget = budget;
        this.totalExpenses = 0.0;
    }

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

    @Override
    public double calculateRemainingBudget() {
        return budget - 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

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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 8\_MCQ

Attempt : 1

Total Mark : 15

Marks Obtained : 15

#### Section 1 : MCQ

1. What will happen if a checked custom exception is thrown inside a method without being caught or declared?

**Answer**

Compilation Error

**Status : Correct**

**Marks : 1/1**

2. What will be the output for the following code?

```
class InvalidUsernameException extends Exception {  
    public InvalidUsernameException(String message) {  
        super(message);  
    }  
}
```

```

class Test {
    public static void main(String[] args) {
        try {
            String username = "abc";
            if (username.length() < 5) {
                throw new InvalidUsernameException("Username must be at
least 5 characters long");
            }
        } catch (InvalidUsernameException e) {
            System.out.println(e.getMessage());
        }
    }
}

```

**Answer**

Username must be at least 5 characters long

**Status :** Correct

**Marks :** 1/1

3. What will be the output for the following code?

```

import java.io.*;

```

```

class TemperatureTooHighException extends Exception {
    public TemperatureTooHighException(String message) {
        super(message);
    }
}

```

```

class Test {
    public static void main(String[] args) {
        try {
            int temperature = 110;
            if (temperature > 100) {
                throw new TemperatureTooHighException("Temperature too
high");
            }
        } catch (TemperatureTooHighException e) {

```



```
        System.out.println(e.getMessage());
    }
}
```

**Answer**

Temperature too high

**Status :** Correct

**Marks :** 1/1

4. What will be the output for the following code?

```
class InvalidVotingAgeException extends Exception {
    public InvalidVotingAgeException(String message) {
        super(message);
    }
}

class Test {
    public static void main(String[] args) {
        try {
            int age = 15;
            if (age < 18) {
                throw new InvalidVotingAgeException("You are not eligible to
vote");
            }
            System.out.println("Eligible to vote");
        } catch (InvalidVotingAgeException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

**Answer**

You are not eligible to vote

**Status :** Correct

**Marks :** 1/1

5. What will be the output for the following code?

```
import java.io.*;
```

```
class OutOfStockException extends Exception {  
    public OutOfStockException(String message) {  
        super(message);  
    }  
}
```

```
class Test {  
    public static void main(String[] args) {  
        try {  
            int stock = 0;  
            if (stock == 0) {  
                throw new OutOfStockException("Item is out of stock");  
            }  
        } catch (OutOfStockException e) {  
            System.out.println(e.getMessage());  
        }  
    }  
}
```

**Answer**

Item is out of stock

**Status :** Correct

**Marks :** 1/1

6. What will be the output of the following code?

```
class MyException extends Exception {  
    public MyException() {  
        super("Default Exception Message");  
    }  
}
```

```
class Test {  
    public static void main(String[] args) {  
        try {  
            throw new MyException();  
        }  
    }  
}
```

```
} catch (MyException e) {  
    System.out.println(e.getMessage());  
}  
}  
}
```

**Answer**

Default Exception Message

**Status :** Correct

**Marks :** 1/1

7. What is the purpose of a custom exception in Java?

**Answer**

To create user-defined exceptions for specific scenarios

**Status :** Correct

**Marks :** 1/1

8. Which of the following is true about custom exceptions?

**Answer**

Custom exceptions must extend either Exception or RuntimeException

**Status :** Correct

**Marks :** 1/1

9. what is the output of the following code?

```
class MyException extends Exception {  
    public MyException(String message) {  
        super(message);  
    }  
}  
  
class Test {  
    public static void main(String[] args) {  
        try {  
            throw new MyException("Error occurred");  
        } catch (MyException e) {
```

```
        System.out.println(e);
    }
}
}
```

**Answer**

MyException: Error occurred

**Status :** Correct

**Marks :** 1/1

10. What will be the output for the following code?

```
import java.io.*;

class UnderageException extends Exception {
    public UnderageException(String message) {
        super(message);
    }
}

class Test {
    public static void main(String[] args) {
        try {
            int age = 17;
            if (age < 18) {
                throw new UnderageException("Underage, cannot proceed");
            }
        } catch (UnderageException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

**Answer**

Underage, cannot proceed

**Status :** Correct

**Marks :** 1/1

11. Which keyword is used to explicitly throw a custom exception?

**Answer**

throw

**Status :** Correct

**Marks :** 1/1

12. How do you create an unchecked custom exception?

**Answer**

By extending RuntimeException

**Status :** Correct

**Marks :** 1/1

13. what is the output of the following code?

```
class MyException extends Exception {  
    public MyException(String message) {  
        super(message);  
    }  
}  
  
class Test {  
    static void check() throws MyException {  
        throw new MyException("Custom Exception Occurred");  
    }  
  
    public static void main(String[] args) {  
        try {  
            check();  
        } catch (Exception e) {  
            System.out.println(e.getMessage());  
        }  
    }  
}
```

**Answer**

Custom Exception Occurred

**Status :** Correct

**Marks :** 1/1

14. What will be the output for the following code?

```
import java.io.*;

class NegativeAgeException extends Exception {
    public NegativeAgeException(String message) {
        super(message);
    }
}

class Test {
    public static void main(String[] args) {
        try {
            int age = -5;
            if (age < 0) {
                throw new NegativeAgeException("Age cannot be negative");
            }
        } catch (NegativeAgeException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

**Answer**

Age cannot be negative

**Status :** Correct

**Marks :** 1/1

15. What will be the output for the following code?

```
class NegativeBalanceException extends Exception {
    public NegativeBalanceException(String message) {
        super(message);
    }
}
```

```
class Test {
    public static void main(String[] args) {
        try {
```

```
double balance = -500;  
if (balance < 0) {  
    throw new NegativeBalanceException("Balance cannot be  
negative");  
}  
} catch (NegativeBalanceException e) {  
    System.out.println("Error: " + e.getMessage());  
}  
}  
}
```

**Answer**

Error: Balance cannot be negative

**Status :** Correct

**Marks :** 1/1

# Rajalakshmi Engineering College

Name: Sachin Krishna B  
Email: 241901507@rajalakshmi.edu.in  
Roll no: 241901507  
Phone: 9025753177  
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

### 2028\_REC\_OOPS using Java\_Week 8\_Q1

Attempt : 1  
Total Mark : 10  
Marks Obtained : 10

#### Section 1 : Coding

##### 1. Problem Statement

Write a program to validate the email address and display suitable exceptions if there is any mistake.

Create 3 custom exception classes as below

DotException AtTheRateException DomainException

A typical email address should have a "." character, and a "@" character, and also the domain name should be valid. Valid domain names for practice be 'in', 'com', 'net', or 'biz'.

Display Invalid Dot usage, Invalid @ usage, or Invalid Domain message based on email id.

Get the email address from the user, validate the email by checking the



above-mentioned criteria, and print the validity status of the input email address.

### ***Input Format***

The first line of input contains the email to be validated.

### ***Output Format***

The output prints a Valid email address or an Invalid email address along with the suitable exception

If email ends with . or contains not exactly one . after @, it throws:

DotException: Invalid Dot usage

Invalid email address

If @ appears not exactly once, it throws:

AtTheRateException: Invalid @ usage

Invalid email address

If the part after the last dot is not among accepted domains:

DomainException: Invalid Domain

Invalid email address

If all conditions satisfied then print:

Valid email address

Refer to the sample input and output for format specifications.

**Sample Test Case**

Input: sample@gmail.com

Output: Valid email address

**Answer**

```
// You are using Java
import java.util.*;
class DotException extends Exception {
    public DotException(String message) {
        super(message);
    }
}
class AtTheRateException extends Exception {
    public AtTheRateException(String message) {
        super(message);
    }
}
class DomainException extends Exception {
    public DomainException(String message) {
        super(message);
    }
}
public class Main {
    public static void main(String[] args) {
```

```
Scanner sc = new Scanner(System.in);
```

```
String email = sc.nextLine().trim();
```

```
try {  
    validateEmail(email);  
    System.out.println("Valid email address");  
} catch (DotException e) {  
    System.out.println("DotException: " + e.getMessage());  
    System.out.println("Invalid email address");  
} catch (AtTheRateException e) {  
    System.out.println("AtTheRateException: " + e.getMessage());  
    System.out.println("Invalid email address");  
} catch (DomainException e) {  
    System.out.println("DomainException: " + e.getMessage());  
    System.out.println("Invalid email address");  
}  
}
```

```
public static void validateEmail(String email) throws DotException,  
AtTheRateException, DomainException {
```

```
    // Check for '@' usage
```

```
    int atCount = email.length() - email.replace("@", "").length();
```

```
    if (atCount != 1) {
```

```
        throw new AtTheRateException("Invalid @ usage");  
    }
```

```
    if (email.startsWith("@") || email.endsWith("@") || email.contains("@@")) {  
        throw new AtTheRateException("Invalid @ usage");  
    }
```

```
    // Check for dot usage
```

```
    if (email.startsWith(".") || email.endsWith(".") || email.contains("..")) {  
        throw new DotException("Invalid Dot usage");  
    }
```

```
    int atIndex = email.indexOf('@');
```

```
    int lastDotIndex = email.lastIndexOf('.');
```

```
    // There must be at least one '.' after '@'
```

```
    if (lastDotIndex < atIndex + 1 || lastDotIndex == email.length() - 1) {  
        throw new DotException("Invalid Dot usage");  
    }
```

```
}  
// Check domain  
String domain = email.substring(lastDotIndex + 1);  
List<String> validDomains = Arrays.asList("in", "com", "net", "biz");  
if (!validDomains.contains(domain)) {  
    throw new DomainException("Invalid Domain");  
}  
}  
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B  
Email: 241901507@rajalakshmi.edu.in  
Roll no: 241901507  
Phone: 9025753177  
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

### 2028\_REC\_OOPS using Java\_Week 8\_Q2

Attempt : 1  
Total Mark : 10  
Marks Obtained : 10

#### Section 1 : Coding

##### 1. Problem Statement

Elsa, a busy professional, is using a scheduling application to plan her meetings efficiently. The application requires users to input meeting durations in minutes, ensuring that the duration is a positive integer and does not exceed 240 minutes (4 hours). Elsa needs a program to assist her in scheduling meetings securely with proper exception handling.

Create a Java class named ElsaMeetingScheduler. Implement a custom exception: InvalidDurationException for invalid meeting duration entries. Implement the main method to interactively take user input for a meeting duration. Implement the validateMeetingDuration method to validate the meeting duration based on the specified rules and throw a custom exception if the validation fails. Print appropriate success or error messages based on the meeting duration.

Implement a custom exception, `InvalidDurationException`, to handle cases where the entered meeting duration does not meet the specified criteria.

### ***Input Format***

The input consists of an integer value 'n', representing the meeting duration.

### ***Output Format***

The output is displayed in the following format:

If the entered meeting duration meets the specified criteria, the program outputs

"Meeting scheduled successfully!"

If the entered meeting duration is invalid, the program outputs an error message indicating the issue.

"Error: Invalid meeting duration. Please enter a positive integer not exceeding 240 minutes (4 hours)."

Refer to the sample output for formatting specifications.

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

Input: 120

Output: Meeting scheduled successfully!

### ***Answer***

```
import java.util.*;
class InvalidDurationException extends Exception {

    public InvalidDurationException(String message) {

        super(message);

    }

}

public class Main {
```

```
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int duration = sc.nextInt();

    try {
        validateMeetingDuration(duration);

        System.out.println("Meeting scheduled successfully!");
    } catch (InvalidDurationException e) {
        System.out.println("Error: " + e.getMessage());
    }
}

public static void validateMeetingDuration(int duration) throws
InvalidDurationException {
    if (duration <= 0 || duration > 240) {
        throw new InvalidDurationException("Invalid meeting duration. Please enter
a positive integer not exceeding 240 minutes (4 hours).");
    }
}
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B  
Email: 241901507@rajalakshmi.edu.in  
Roll no: 241901507  
Phone: 9025753177  
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

### 2028\_REC\_OOPS using Java\_Week 8\_Q3

Attempt : 1  
Total Mark : 10  
Marks Obtained : 10

#### Section 1 : Coding

##### 1. Problem Statement

In a user registration system, there is a requirement to implement a username validation module. Users attempting to register must adhere to specific criteria for their usernames to be considered valid.

Your task is to develop a program that takes user input for a desired username and validates it according to the following rules:

The username must not contain any spaces. The username must be at least 5 characters long.

Implement a custom exception, `InvalidUsernameException`, to handle cases where the entered username does not meet the specified criteria.

##### ***Input Format***



The input consists of a string S, representing the desired username.

### **Output Format**

If the username is valid, print "Username is valid: [S]".

If the username is invalid:

1. If the username is short, print "Invalid Username: Username must be at least 5 characters long"
2. If the username contains spaces, print "Invalid Username: Username cannot contain spaces"

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: John

Output: Invalid Username: Username must be at least 5 characters long

### **Answer**

```
import java.util.*;
class InvalidUsernameException extends Exception {

    public InvalidUsernameException(String message) {
        super(message);
    }

}

public class Main {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        String username = sc.nextLine();
        try {
            validateUsername(username);
            System.out.println("Username is valid: " + username);
        }
    }
}
```

```
    } catch (InvalidUsernameException e) {  
        System.out.println(e.getMessage());  
    }  
}
```

```
public static void validateUsername(String username) throws  
InvalidUsernameException {  
    if (username.contains(" ")) {  
        throw new InvalidUsernameException("Invalid Username: Username cannot  
contain spaces");  
    } else if (username.length() < 5) {  
        throw new InvalidUsernameException("Invalid Username: Username must  
be at least 5 characters long");  
    }  
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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

### 2028\_REC\_OOPS using Java\_Week 8\_Q4

Attempt : 1

Total Mark : 10

Marks Obtained : 10

### Section 1 : Coding

#### 1. Problem Statement

A local municipality is implementing an online voting system for a community event and wants to ensure that only eligible voters (those aged 18 or older) can participate.

Your task is to develop a program that validates the age of individuals attempting to vote online. If the user's age is below 18, the program should throw a custom exception, `InvalidAgeException`, preventing them from casting their vote. If the input is invalid, catch the appropriate `InputMismatchException` and print the in-built exception message.

#### ***Input Format***

The input consists of an integer representing the age.

#### ***Output Format***

If the age is 18 or older, print "Eligible to vote"

If the age is below 18, print "Exception occurred: InvalidAgeException: Age is not valid to vote"

If there is any other type of exception, print "An error occurred: " followed by the in-built exception message.

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 20

Output: Eligible to vote

### **Answer**

```
// You are using Java
import java.util.*;
```

```
class InvalidAgeException extends Exception {
    public InvalidAgeException(String message) {
        super(message);
    }
}
```

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

            if (age < 18) {
                throw new InvalidAgeException("Age is not valid to vote");
            } else {
                System.out.println("Eligible to vote");
            }
        } catch (InvalidAgeException e) {
            System.out.println("Exception occurred: InvalidAgeException: " +
                e.getMessage());
        }
    }
}
```

```
} catch (InputMismatchException e) {  
    System.out.println("An error occurred: " + e);  
} catch (Exception e) {  
    System.out.println("An error occurred: " + e);  
}  
}  
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B  
Email: 241901507@rajalakshmi.edu.in  
Roll no: 241901507  
Phone: 9025753177  
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

### 2028\_REC\_OOPS using Java\_Week 8\_Q5

Attempt : 1  
Total Mark : 10  
Marks Obtained : 10

#### Section 1 : Coding

##### 1. Problem Statement

In a file management system, users are required to provide a valid file name when creating new files. The system enforces specific rules for file names to maintain consistency and avoid potential issues. Your task is to implement a Java program named FileNameValidator that takes user input for a file name and validates it according to the specified rules.

Rules for Valid File Name:

The file name must consist of alphanumeric characters (letters and digits) only. The file name must have a minimum length of 3 characters.

Implement a custom exception, FileNameValidator, to handle cases where the entered filename does not meet the specified criteria.

***Input Format***

The input consists of a string S, representing the desired filename.

### **Output Format**

The output is displayed in the following format:

If the entered file name meets the specified criteria, the program outputs

"Valid file name"

If the entered file name does not meet the criteria and triggers the `InvalidFileNameException`, the program outputs

"Error: Invalid file name. It must be alphanumeric and have a minimum length of 3 characters."

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: myfile123

Output: Valid file name

### **Answer**

```
// You are using Java
import java.util.*;
```

```
class InvalidFileNameException extends Exception {
    public InvalidFileNameException(String message) {
        super(message);
    }
}
```

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

        try {
            validateFileName(fileName);
        }
    }
}
```

```
        System.out.println("Valid file name");
    } catch (InvalidFileNameException e) {
        System.out.println("Error: Invalid file name. It must be alphanumeric and
have a minimum length of 3 characters.");
    }
}
```

```
public static void validateFileName(String name) throws
InvalidFileNameException {
    if (name.length() < 3 || !name.matches("[A-Za-z0-9]+")) {
        throw new InvalidFileNameException("Invalid");
    }
}
}
```

**Status :** Correct

**Marks :** 10/10



# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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 8\_PAH

Attempt : 1

Total Mark : 40

Marks Obtained : 30

### Section 1 : Coding

#### 1. Problem Statement

Daniel is developing a program to verify the age of users. He wants to ensure that the entered age is within a valid range. Write a program to help Daniel implement this age-checking feature using custom exceptions.

Daniel needs a program that takes an integer input representing a person's age. If the age is between 0 and 150 (inclusive), the program should print "Age is valid!". If the age is less than 0 or greater than 150, the program should throw a custom exception (InvalidAgeException) with the message "Invalid age. Please enter an age between 0 and 150."

Implement a custom exception, InvalidAgeException, to handle cases where the entered age does not meet the specified criteria.

#### **Input Format**

The input consists of an integer value 'n', representing the age.

### **Output Format**

The output is displayed in the following format:

If the age is valid (between 0 and 150, inclusive), print

"Age is valid!".

If the age is invalid, print

"Error: Invalid age. Please enter an age between 0 and 150."

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 45

Output: Age is valid!

### **Answer**

```
// You are using Java
import java.util.*;
```

```
class InvalidAgeException extends Exception {
    public InvalidAgeException(String message) {
        super(message);
    }
}
```

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

```
        try {
            validateAge(age);
            System.out.println("Age is valid!");
        } catch (InvalidAgeException e) {
```

```
        System.out.println("Error: Invalid age. Please enter an age between 0 and  
150.");  
    }  
}  
  
    public static void validateAge(int age) throws InvalidAgeException {  
        if (age < 0 || age > 150) {  
            throw new InvalidAgeException("Invalid age. Please enter an age between  
0 and 150.");  
        }  
    }  
}
```

**Status :** Correct

**Marks : 10/10**

## 2. Problem Statement

You are tasked to create a program that defines a custom exception `GradeException`. The program should include a `Student` class with fields for the student's name, age, and grade. Implement a method in the `Student` class that checks the grade, and if the grade is below 40, it should throw a `GradeException`. Otherwise, it should display the student's details.

### *Input Format*

The input consists of three parameters in separate lines:

1. A string representing the student's name.
2. An integer representing the student's age.
3. An integer representing the student's grade.

### *Output Format*

The output will display the student's details if the grade is valid.

If the grade is below 40, the program will display an error message "Grade is below 40".

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: Alice

20

85

Output: Name: Alice

Age: 20

Grade: 85

### **Answer**

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

```
// Custom Exception
```

```
class GradeException extends Exception {  
    public GradeException(String message) {  
        super(message);  
    }  
}
```

```
// Student Class
```

```
class Student {  
    String name;  
    int age;  
    int grade;
```

```
    public Student(String name, int age, int grade) {  
        this.name = name;  
        this.age = age;  
        this.grade = grade;  
    }
```

```
    public void checkGrade() throws GradeException {  
        if (grade < 40) {  
            throw new GradeException("Grade is below 40");  
        } else {  
            System.out.println("Name: " + name);  
            System.out.println("Age: " + age);  
            System.out.println("Grade: " + grade);  
        }  
    }
```

```
    }  
}
```

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

        String name = scanner.nextLine();
        int age = Integer.parseInt(scanner.nextLine());
        int grade = Integer.parseInt(scanner.nextLine());

        Student student = new Student(name, age, grade);

        try {
            student.checkGrade();
        } catch (GradeException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

**Status : Wrong**

**Marks : 0/10**

### 3. Problem Statement

An HR software system is being developed to process employee payrolls. During payroll processing, the system must ensure that no employee has a negative salary and that no employee's salary exceeds 2,00,000. If either condition occurs, the system should throw a custom exception.

Create a custom exception `InvalidSalaryException` and a class `Employee` that processes salary according to the following rules:

If  $\text{salary} < 0$ , throw `InvalidSalaryException` with the message: "Salary cannot be negative". If  $\text{salary} > 200000$ , throw `InvalidSalaryException` with the message: "Salary exceeds threshold limit". Otherwise, display: "Salary processed successfully for <empName>: <salary>".

The payroll processing should always display: "Payroll process completed" at the end, regardless of whether an exception occurs.

**Input Format**

The first line of input contains an integer representing the employee ID.

The second line contains a string representing the employee's name.

The third line contains a floating-point number representing the salary of the employee.

### **Output Format**

If the salary is valid: "Salary processed successfully for <empName>: <salary>"

"Payroll process completed"

If the salary is invalid: "<Exception Message>"

"Payroll process completed"

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 101

Rahul

150000.0

Output: Salary processed successfully for Rahul: 150000.0

Payroll process completed

### **Answer**

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

```
// Custom Exception
class InvalidSalaryException extends Exception {
    public InvalidSalaryException(String message) {
        super(message);
    }
}
```

```
// Employee Class
class Employee {
    int empId;
```

```
String empName;  
double salary;
```

```
public Employee(int empId, String empName, double salary) {  
    this.empId = empId;  
    this.empName = empName;  
    this.salary = salary;  
}
```

```
public void processSalary() throws InvalidSalaryException {  
    if (salary < 0) {  
        throw new InvalidSalaryException("Salary cannot be negative");  
    } else if (salary > 200000) {  
        throw new InvalidSalaryException("Salary exceeds threshold limit");  
    } else {  
        System.out.println("Salary processed successfully for " + empName + " : "  
+ salary);  
    }  
}
```

```
// Main Class
```

```
public class Main {  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);  
  
        int empId = Integer.parseInt(scanner.nextLine());  
        String empName = scanner.nextLine();  
        double salary = Double.parseDouble(scanner.nextLine());  
  
        Employee emp = new Employee(empId, empName, salary);  
  
        try {  
            emp.processSalary();  
        } catch (InvalidSalaryException e) {  
            System.out.println(e.getMessage());  
        } finally {  
            System.out.println("Payroll process completed");  
        }  
    }  
}
```

Status : Correct

Marks : 10/10

#### 4. Problem Statement

Enigma is developing a simple web application that takes a user-input URL, validates it, and throws a custom exception `InvalidURLException` if the URL does not start with "http://" or "https://".

The main method prompts the user for input, validates the URL, and prints whether it is valid or not.

##### *Input Format*

The input consists of a string, representing the URL entered by the user.

##### *Output Format*

The output displays one of the following results:

If the entered URL is valid according to the specified format, the program prints:

"[URL] is a valid URL"

If the entered URL is not valid according to the specified format, the program prints:

"Invalid URL format: [URL]"

Refer to the sample output for formatting specifications.

##### *Sample Test Case*

Input: `http://www.example.com`

Output: `http://www.example.com is a valid URL`

##### *Answer*



```
import java.util.*;
```

```
class InvalidURLException extends Exception {  
    public InvalidURLException(String message) {  
        super(message);  
    }  
}
```

```
class URLValidator {  
    public static void validateURL(String url) throws InvalidURLException {  
        if (!(url.startsWith("http://") || url.startsWith("https://"))) {  
            throw new InvalidURLException("Invalid URL format: " + url);  
        }  
    }  
}
```

```
public class Main {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        String url = sc.nextLine();  
        sc.close();  
  
        try {  
            URLValidator.validateURL(url);  
            System.out.println(url + " is a valid URL");  
        } catch (InvalidURLException e) {  
            System.out.println(e.getMessage());  
        }  
    }  
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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 8\_CY

Attempt : 1

Total Mark : 40

Marks Obtained : 40

### Section 1 : Coding

#### 1. Problem Statement

Hemanth is designing a banking system for XYZ Bank. The system should allow customers to perform deposit, withdrawal, and balance inquiry operations. Implement exception handling for scenarios involving invalid transaction amounts or insufficient funds.

Create two custom exception classes, InvalidAmountException and InsufficientFundsException, both extending the Exception class. Throw an InvalidAmountException with a message if the deposit amount is less than or equal to zero. Throw an InsufficientFundsException if the withdrawal amount is greater than the available balance. Deduct the withdrawal amount from the balance if the withdrawal is successful.

Assist Hemanth in designing the program.

### ***Input Format***

The first line of input consists of a double value B, representing the initial balance.

The second line consists of a double value D, representing the deposit amount.

The third line consists of a double value W, representing the withdrawal amount.

### ***Output Format***

If the withdrawal is successful, print the amount withdrawn and the current balance, rounded off to one decimal place.

If an `InvalidAmountException` occurs, print "Error: [D] is not valid".

If an `InsufficientFundsException` occurs, print "Error: Insufficient funds".

Refer to the sample output for formatting specifications.

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

Input: 1050.1

270.2

150.3

Output: Amount Withdrawn: 150.3

Current Balance: 1170.0

### ***Answer***

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

```
// Custom exception for invalid deposit amount
```

```
class InvalidAmountException extends Exception {  
    public InvalidAmountException(String message) {  
        super(message);  
    }  
}
```

```
// Custom exception for insufficient funds
```

```
class InsufficientFundsException extends Exception {  
    public InsufficientFundsException(String message) {
```

```

        super(message);
    }
}

public class Main {

    // Method to handle deposit
    public static double deposit(double balance, double amount) throws
InvalidAmountException {
        if (amount <= 0) {
            throw new InvalidAmountException(amount + " is not valid");
        }
        return balance + amount;
    }

    // Method to handle withdrawal
    public static double withdraw(double balance, double amount) throws
InsufficientFundsException {
        if (amount > balance) {
            throw new InsufficientFundsException("Insufficient funds");
        }
        return balance - amount;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        double balance = sc.nextDouble();
        double depositAmount = sc.nextDouble();
        double withdrawAmount = sc.nextDouble();

        try {
            // Deposit process
            balance = deposit(balance, depositAmount);

            // Withdrawal process
            balance = withdraw(balance, withdrawAmount);

            System.out.println("Amount Withdrawn: " + withdrawAmount);
            System.out.printf("Current Balance: %.1f", balance);
        } catch (InvalidAmountException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}

```

```
} catch (InsufficientFundsException e) {  
    System.out.println("Error: " + e.getMessage());  
}  
}  
}
```

**Status :** Correct

**Marks :** 10/10

## 2. Problem Statement

Tim was tasked with creating a user profile system that validates the user's date of birth input. The system should throw a custom exception, `InvalidDateOfBirthException`, if the date is not in the specified format "dd-mm-yyyy" or if it represents an invalid calendar date.

The main method takes user input, validates the date of birth, and prints whether it is valid or not.

### ***Input Format***

The input consists of a string, representing the date of birth of the user.

### ***Output Format***

The output displays one of the following results:

If the entered date of birth is valid according to the specified format, the program prints:

"[Date] is a valid date of birth"

If the entered date of birth is not valid according to the specified format, the program prints:

"Invalid date: [Date]"

Refer to the sample output for formatting specifications.

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

Input: 01-01-2000

Output: 01-01-2000 is a valid date of birth

**Answer**

```
import java.util.*;
import java.text.*;

class InvalidDateOfBirthException extends Exception {
    public InvalidDateOfBirthException(String message) {
        super(message);
    }
}

class DateValidator {
    public static void validateDate(String dateStr) throws
    InvalidDateOfBirthException {
        SimpleDateFormat sdf = new SimpleDateFormat("dd-MM-yyyy");
        sdf.setLenient(false); // ensures strict date validation
        try {
            Date date = sdf.parse(dateStr);
        } catch (ParseException e) {
            throw new InvalidDateOfBirthException("Invalid date: " + dateStr);
        }
    }
}

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

        try {
            DateValidator.validateDate(dateStr);
            System.out.println(dateStr + " is a valid date of birth");
        } catch (InvalidDateOfBirthException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

**Status : Correct**

**Marks : 10/10**

### 3. Problem Statement

Theo is trying to update his payment information on a subscription-based streaming service. To proceed, the system requires Theo to provide a valid credit card number consisting of 16 digits. However, Theo wants to make sure that the credit card number he enters meets the specified criteria with proper exception handling.

The credit card number must consist of exactly 16 digits. If the entered credit card number does not meet the specified criteria, the program should throw a custom exception, `InvalidCreditCardException`, and provide Theo with specific error messages: If the length of the credit card number is not 16 digits, the exception message should be: "Invalid credit card number length." If the credit card number contains non-numeric characters, the exception message should be: "Invalid credit card number format."

Implement a custom exception, `InvalidCreditCardException`, to fulfill Theo's requirements and keep his payment information secure.

#### ***Input Format***

The input consists of a string value 's', consisting of the 16-digit credit card number.

#### ***Output Format***

The output is displayed in the following format:

If the entered credit card number is valid, the program should output a success message:

"Payment information updated successfully!"

If the entered credit card has more than 16 digits or less than 16 digits it displays

"Error: Invalid credit card number length."

If the entered 16-digit credit card has non-integers it displays

"Error: Invalid credit card number format."

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 1234567890123456

Output: Payment information updated successfully!

### **Answer**

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

```
// Custom exception class
```

```
class InvalidCreditCardException extends Exception {  
    public InvalidCreditCardException(String message) {  
        super(message);  
    }  
}
```

```
// Main class
```

```
public class Main {
```

```
    // Method to validate credit card number
```

```
    public static void validateCreditCard(String cardNumber) throws  
InvalidCreditCardException {  
        if (cardNumber.length() != 16) {  
            throw new InvalidCreditCardException("Invalid credit card number  
length.");  
        }  
        if (!cardNumber.matches("\\d{16}")) {  
            throw new InvalidCreditCardException("Invalid credit card number  
format.");  
        }  
    }  
}
```

```
public static void main(String[] args) {
```

```
    Scanner sc = new Scanner(System.in);
```

```
    String cardNumber = sc.nextLine();
```

```
    try {
```

```
        validateCreditCard(cardNumber);
```

```
        System.out.println("Payment information updated successfully!");
```

```
    } catch (InvalidCreditCardException e) {
```

```
        System.out.println("Error: " + e.getMessage());
```



**Status :** Correct

**Marks :** 10/10

#### 4. Problem Statement

Faustus is managing his bank account and wants to create a program to update his account balance based on certain conditions. However, he needs to handle specific scenarios related to invalid inputs and insufficient balances. Faustus wants to update his account balance. He inputs the current balance and the amount to be updated.

The initial account balance should be positive. If Faustus enters a negative initial balance, the program should throw an `InvalidAmountException` with the message "Invalid amount. Please enter a positive initial balance." If the amount to be updated is negative, the program should check if the subtraction results in a negative balance. If so, it should throw an `InsufficientBalanceException` with the message "Insufficient balance." If the amount to be updated is positive, it should be added to the current balance, and the new balance should be printed.

Implement a custom exception, `InvalidAmountException`, and `InsufficientBalanceException`, to manage his bank account.

##### **Input Format**

The first line of input consists of a double value 'd', representing the initial account balance.

The second line of input consists of a double value 'd1', representing the amount to be updated.

##### **Output Format**

The output is displayed in the following format:

If the validation passes, print

"Account balance updated successfully! New balance: {new\_balance}"

where {new\_balance} is the updated account balance

If the initial bank amount is negative it displays

"Error: Invalid amount. Please enter a positive initial balance."

If the updated amount exceeds the initial account balance in withdrawal it displays

"Error: Insufficient balance."

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 1000

500

Output: Account balance updated successfully! New balance: 1500.0

### **Answer**

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

```
// Custom exception for invalid initial amount
```

```
class InvalidAmountException extends Exception {  
    public InvalidAmountException(String message) {  
        super(message);  
    }  
}
```

```
// Custom exception for insufficient balance
```

```
class InsufficientBalanceException extends Exception {  
    public InsufficientBalanceException(String message) {  
        super(message);  
    }  
}
```

```
public class Main {
```

```
    // Method to update account balance
```

```

public static double updateBalance(double balance, double amount)
    throws InvalidAmountException, InsufficientBalanceException {

    if (balance < 0) {
        throw new InvalidAmountException("Invalid amount. Please enter a
positive initial balance.");
    }

    if (amount < 0) {
        if (balance + amount < 0) {
            throw new InsufficientBalanceException("Insufficient balance.");
        } else {
            balance += amount; // withdrawing
        }
    } else {
        balance += amount; // depositing
    }

    return balance;
}

public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    double balance = sc.nextDouble();
    double amount = sc.nextDouble();

    try {
        double newBalance = updateBalance(balance, amount);
        System.out.println("Account balance updated successfully! New balance:
" + newBalance);
    } catch (InvalidAmountException | InsufficientBalanceException e) {
        System.out.println("Error: " + e.getMessage());
    }
}

```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B  
Email: 241901507@rajalakshmi.edu.in  
Roll no: 241901507  
Phone: 9025753177  
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 9\_MCQ

Attempt : 1  
Total Mark : 15  
Marks Obtained : 14

#### Section 1 : MCQ

1. What will be the output of the following code?

```
import java.util.*;
class Main {
    public static void main(String[] args) {
        ArrayList<Integer> list = new ArrayList<>();
        list.add(10);
        list.add(20);
        list.add(30);
        list.remove(1);
        System.out.println(list);
    }
}
```

**Answer**

[10, 30]

**Status :** Correct

**Marks :** 1/1

2. What is Collection in Java?

**Answer**

A group of objects

**Status :** Correct

**Marks :** 1/1

3. What will be the output of the following code?

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Stack<Integer> stack = new Stack<>();
        for (int i = 1; i <= 3; i++)
            stack.push(i * 2);
        stack.pop();
        stack.push(10);
        System.out.println(stack.peek());
    }
}
```

**Answer**

10

**Status :** Correct

**Marks :** 1/1

4. How can you access the first element of an ArrayList named as list?

**Answer**

list.get(0);

**Status :** Correct

**Marks :** 1/1

5. What will be the output of the following code?

```
import java.util.*;
class Main {
    public static void main(String[] args) {
        ArrayList<Integer> list = new ArrayList<>();
        list.add(1);
        list.add(2);
        list.add(3);
        list.add(4);
        list.add(5);
        System.out.println(list.get(3));
    }
}
```

**Answer**

4

**Status :** Correct

**Marks :** 1/1

6. What will be the output of the following code?

```
import java.util.*;
class Main {
    public static void main(String[] args) {
        ArrayList<Integer> list = new ArrayList<>();
        list.add(1);
        list.add(2);
        list.add(3);
        list.add(4);
        list.set(2, 10);
        System.out.println(list);
    }
}
```

**Answer**

[1, 2, 10, 4]

**Status :** Correct

**Marks :** 1/1

7. What will be the output of the following code?

```
import java.util.ArrayList;

public class Main {
    public static void main(String[] args) {
        ArrayList<Integer> list = new ArrayList<>();
        list.add(10);
        list.add(20);
        list.add(30);
        System.out.println("Size of the list: " + list.size());
    }
}
```

**Answer**

Size of the list: 3

**Status :** Correct

**Marks :** 1/1

8. What does the addFirst() method of LinkedList do?

**Answer**

Adds an element to the beginning of the list

**Status :** Correct

**Marks :** 1/1

9. What is the correct way to create an ArrayList in Java?

**Answer**

ArrayList<int> list = new ArrayList<>();

**Status :** Wrong

**Marks :** 0/1

10. Which method is used to add an element to the top of the stack?

**Answer**

push()

**Status :** Correct

**Marks :** 1/1

11. What will be the output of the following code?

```
import java.util.*;
class Main {
    public static void main(String[] args) {
        ArrayList<String> list = new ArrayList<>();
        list.add("apple");
        list.add("banana");
        list.add("cherry");
        list.add("banana");
        System.out.println(list.lastIndexOf("banana"));
    }
}
```

**Answer**

3

**Status :** Correct

**Marks :** 1/1

12. Which of the following methods removes and returns the last element from a LinkedList?

**Answer**

removeLast()

**Status :** Correct

**Marks :** 1/1

13. What will be the output of the following code?

```
import java.util.ArrayList;

public class Main {
    public static void main(String[] args) {
        ArrayList<String> list = new ArrayList<>();
        list.add("Apple");
        list.add("Banana");
    }
}
```



```
list.remove("Apple");  
System.out.println(list);  
}  
}
```

**Answer**

[Banana]

**Status :** Correct

**Marks :** 1/1

14. What will be the output of the following code?

```
import java.util.*;  
class Main {  
    public static void main(String[] args) {  
        ArrayList<String> list = new ArrayList<>();  
        list.add("Java");  
        list.add("Python");  
        list.add("Java");  
        list.add("C++");  
        System.out.println(list.indexOf("Java"));  
    }  
}
```

**Answer**

0

**Status :** Correct

**Marks :** 1/1

15. What will be the output of the following code?

```
import java.util.*;  
public class Main {  
    public static void main(String[] args) {  
        Stack<Integer> s = new Stack<>();  
        s.push(10);  
        s.push(20);  
        s.push(30);
```

```
        System.out.println(s.peek());  
    }  
}
```

**Answer**

30

**Status :** Correct

**Marks :** 1/1

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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

### 2028\_REC\_OOPS using Java\_Week 9\_Q1

Attempt : 1

Total Mark : 10

Marks Obtained : 10

### Section 1 : Coding

#### 1. Problem Statement

Bobby is tasked with processing a sequence of numbers from a monitoring system. He needs to extract a strictly increasing subsequence using an ArrayList. The program should dynamically add numbers to the ArrayList only if they are greater than the last number currently stored in the list. Bobby aims to efficiently utilize the dynamic resizing and indexing features of the ArrayList to solve this problem.

Help Bobby implement this solution.

#### ***Input Format***

The first line of input consists of an integer N, representing the number of elements.

The second line consists of N space-separated integers, representing the elements.

### **Output Format**

The output prints the list of integers in increasing sequence, ignoring out-of-order elements.

Refer to the sample output for the formatting specifications.

### **Sample Test Case**

Input: 7

3 5 9 1 11 7 13

Output: [3, 5, 9, 11, 13]

### **Answer**

```
// You are using Java
import java.util.*;
```

```
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int N = sc.nextInt();
        ArrayList<Integer> list = new ArrayList<>();

        for (int i = 0; i < N; i++) {
            int num = sc.nextInt();
            if (list.isEmpty() || num > list.get(list.size() - 1)) {
                list.add(num);
            }
        }

        System.out.print(list);
    }
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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

### 2028\_REC\_OOPS using Java\_Week 9\_Q2

Attempt : 1

Total Mark : 10

Marks Obtained : 10

### Section 1 : Coding

#### 1. Problem Statement

Vikram loves listening to music and wants to create a simple playlist manager using Java Collections. The playlist supports the following operations:

"ADD <song>" Adds the song to the end of the playlist. "REMOVE <song>" Removes the first occurrence of the song from the playlist. If the song is not found, do nothing. "SHOW" Displays all songs in the playlist in order. If the playlist is empty, print "EMPTY". "NEXT" Moves to the next song in the playlist and prints its name. If the playlist is empty, print "EMPTY".

The playlist maintains a "current song" position that starts at the first song when it's added. The NEXT command moves to the next song and prints it, wrapping around to the first song after reaching the last song. When removing songs, the current position adjusts accordingly to maintain

proper navigation.

Help Vikram implement this playlist manager.

### ***Input Format***

The first line of the input consists of an integer n, the number of operations.

The next n lines, each containing a command:

- "ADD <song>"
- "REMOVE <song>"
- "SHOW"
- "NEXT"

### ***Output Format***

For each "SHOW" command, print the songs in order, separated by spaces.

For each "NEXT" command, print the next song in the playlist.

If no song exists, print "EMPTY".

Refer to the sample output for formatting specifications.

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

Input: 7

ADD song1

ADD song2

SHOW

NEXT

REMOVE song2

SHOW

NEXT

Output: song1 song2

song2

song1

song1

### ***Answer***

```
import java.util.*;
```

```
public class Main {
```

```
    public static void main(String[] args) {
```

```
        Scanner sc = new Scanner(System.in);
```

```
        int n = Integer.parseInt(sc.nextLine());
```

```
        LinkedList<String> playlist = new LinkedList<>();
```

```
        int currentIndex = 0;
```

```
        for (int i = 0; i < n; i++) {
```

```
            String command = sc.nextLine();
```

```
            if (command.startsWith("ADD")) {
```

```
                String song = command.substring(4);
```

```
                playlist.add(song);
```

```
                if (playlist.size() == 1) {
```

```
                    currentIndex = 0;
```

```
                }
```

```
            }
```

```
            else if (command.startsWith("REMOVE")) {
```

```
                String song = command.substring(7);
```

```
                if (!playlist.isEmpty() && playlist.contains(song)) {
```

```
                    int removeIndex = playlist.indexOf(song);
```

```
                    playlist.remove(song);
```

```
                    if (playlist.isEmpty()) {
```

```
                        currentIndex = 0;
```

```
                    } else if (removeIndex <= currentIndex && currentIndex > 0) {
```

```
                        currentIndex--;
```

```
                    }
```

```
            }
```

```
        }
```

```
        else if (command.equals("SHOW")) {
```

```
            if (playlist.isEmpty()) {
```

```
                System.out.println("EMPTY");
```

```
            } else {
```

```
                for (String s : playlist) {
```

```
                    System.out.print(s + " ");
```

```
                }
```

```
                System.out.println();
```

```
            }
```

```
        }
```

```
        else if (command.equals("NEXT")) {
```

```
        if (playlist.isEmpty()) {  
            System.out.println("EMPTY");  
        } else {  
            currentIndex = (currentIndex + 1) % playlist.size();  
            System.out.println(playlist.get(currentIndex));  
        }  
    }  
}  
}
```

**Status :** Correct

**Marks : 10/10**



# Rajalakshmi Engineering College

Name: Sachin Krishna B  
Email: 241901507@rajalakshmi.edu.in  
Roll no: 241901507  
Phone: 9025753177  
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

### 2028\_REC\_OOPS using Java\_Week 9\_Q3

Attempt : 1  
Total Mark : 10  
Marks Obtained : 10

#### Section 1 : Coding

##### 1. Problem Statement

Assist Pranitha in developing a program that takes an integer N as input, representing the number of names to be read. Then read N names and store them in an ArrayList. Finally, input a search string and output the frequency of that string in the list of names.

Note: Some parts of the code are provided as snippets, and you need to complete the remaining sections by writing the necessary code.

##### ***Input Format***

The first line of input consists of an integer N, representing the number of names to be read.

The following N lines consist of N names, as a string.

The last line consists of a string, representing the name to be searched.

### **Output Format**

The output prints a single integer, representing the frequency of the specified name in the given list.

If the specified name is not found, print 0.

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 5

Alice

Bob

Ankit

Alice

Pranitha

Alice

Output: 2

### **Answer**

// You are using Java

```
import java.util.*;
```

```
public class Main {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        int N = sc.nextInt();  
        sc.nextLine(); // consume the newline
```

```
        ArrayList<String> names = new ArrayList<>();
```

```
        for (int i = 0; i < N; i++) {  
            String name = sc.nextLine();  
            names.add(name);  
        }
```

```
        String searchName = sc.nextLine();  
        int count = 0;
```

```
        for (String name : names) {  
            if (name.equals(searchName)) {  
                count++;  
            }  
        }  
  
        System.out.println(count);  
    }  
}
```

**Status :** Correct

**Marks : 10/10**

# Rajalakshmi Engineering College

Name: Sachin Krishna B  
Email: 241901507@rajalakshmi.edu.in  
Roll no: 241901507  
Phone: 9025753177  
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 9\_PAH

Attempt : 1  
Total Mark : 30  
Marks Obtained : 30

#### Section 1 : Coding

##### 1. Problem Statement

Aditi is analyzing stock market trends and wants to find the Next Greater Element (NGE) for each stock price in a list. The Next Greater Element for an element  $x$  in an array is the first element to the right that is greater than  $x$ . If no greater element exists, return -1 for that position.

Your task is to help Aditi by efficiently computing the Next Greater Element for each element in the given array using a Stack.

Example:

Input:

6

4 5 2 10 8 6

Output:

5 10 10 -1 -1 -1

Explanation:

For each element:

4 5 (next greater element)5 102 1010 -1 (No greater element)8 -16 -1

### ***Input Format***

The first line contains an integer n, representing the number of elements.

The second line contains n space-separated integers arr[i], where arr[i] is the stock price on the i-th day.

### ***Output Format***

The output prints n space-separated integers representing the Next Greater Element for each element in the array.

Refer to the sample output for formatting specifications.

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

Input: 6

4 5 2 10 8 6

Output: 5 10 10 -1 -1 -1

### ***Answer***

```
// You are using Java
import java.util.*;
```

```
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int arr[] = new int[n];
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }
    }
}
```

```

int result[] = new int[n];
Stack<Integer> stack = new Stack<>();

for (int i = n - 1; i >= 0; i--) {
    while (!stack.isEmpty() && stack.peek() <= arr[i]) {
        stack.pop();
    }
    if (stack.isEmpty()) {
        result[i] = -1;
    } else {
        result[i] = stack.peek();
    }
    stack.push(arr[i]);
}

for (int i = 0; i < n; i++) {
    System.out.print(result[i] + " ");
}
}

```

**Status :** Correct

**Marks :** 10/10

## 2. Problem Statement

Rekha is a teacher who wants to calculate the average of marks scored by her students in a test. She needs to store all the marks dynamically because the number of students may vary each time. Using an ArrayList allows her to easily add any number of marks without worrying about the initial size.

Help her implement the task.

### **Input Format**

The first line of input is an integer  $n$ , representing the number of students..

The second line of input consists of  $n$  double values, representing the marks of each student, separated by a space.

### **Output Format**

The output prints: "Average of the list: " followed by the average value formatted to two decimal places.

Refer to the sample output for the formatting specifications.

### **Sample Test Case**

Input: 5

1.0 2.0 3.0 4.0 5.0

Output: Average of the list: 3.00

### **Answer**

```
// You are using Java
import java.util.*;
```

```
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        ArrayList<Double> marks = new ArrayList<>();

        for (int i = 0; i < n; i++) {
            marks.add(sc.nextDouble());
        }

        double sum = 0;
        for (double m : marks) {
            sum += m;
        }

        double average = sum / n;
        System.out.printf("Average of the list: %.2f", average);
    }
}
```

**Status :** Correct

**Marks :** 10/10

### **3. Problem Statement**

Arun is building a task manager to keep track of tasks using a LinkedList. The task manager supports the following operations:

"ADD <task>" Adds the given task to the end of the list. "REMOVE" Removes the first task from the list. "SHOW" Displays all tasks in the list in order. If the list is empty, print "EMPTY".

Help Arun implement this functionality using a LinkedList.

### ***Input Format***

The first line of the input consists of an integer n, the number of operations.

The next n lines, each containing a command:

- "ADD <task>"
- "REMOVE"
- "SHOW"

### ***Output Format***

For each "SHOW" command, the output prints the tasks in order, separated by spaces.

If no tasks exist, print "EMPTY".

Refer to the sample output for formatting specifications.

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

Input: 5

ADD homework

ADD project

SHOW

REMOVE

SHOW

Output: homework project  
project

### ***Answer***



```

import java.util.*;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = Integer.parseInt(sc.nextLine());
        LinkedList<String> tasks = new LinkedList<>();

        for (int i = 0; i < n; i++) {
            String command = sc.nextLine();

            if (command.startsWith("ADD")) {
                String task = command.substring(4);
                tasks.add(task);
            }
            else if (command.equals("REMOVE")) {
                if (!tasks.isEmpty()) {
                    tasks.removeFirst();
                }
            }
            else if (command.equals("SHOW")) {
                if (tasks.isEmpty()) {
                    System.out.println("EMPTY");
                }
                else {
                    for (String t : tasks) {
                        System.out.print(t + " ");
                    }
                    System.out.println();
                }
            }
        }
    }
}

```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B  
Email: 241901507@rajalakshmi.edu.in  
Roll no: 241901507  
Phone: 9025753177  
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 9\_CY

Attempt : 1  
Total Mark : 40  
Marks Obtained : 40

#### Section 1 : Coding

##### 1. Problem Statement

Sanjay is working on a program to merge two sorted linked lists into a single sorted list using Java's LinkedList class from the Collections framework. Given two sorted linked lists, he wants to merge them while maintaining the sorted order.

Write a Java program that:

Reads two sorted linked lists. Merges them into a single sorted linked list. Prints the merged list in ascending order.

##### ***Input Format***

The first line contains an integer  $m$  (the size of the first linked list).

The second line contains  $m$  space-separated integers (sorted).

The third line contains an integer n (the size of the second linked list).

The fourth line contains n space-separated integers (sorted).

### **Output Format**

The output prints the merged linked list as space-separated integers.

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 2

5 10

3

1 3 8

Output: 1 3 5 8 10

### **Answer**

```
import java.util.*;
class MergeSortedLinkedLists {

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

        // Read size of first list
        int m = sc.nextInt();
        LinkedList<Integer> list1 = new LinkedList<>();
        for (int i = 0; i < m; i++) {
            list1.add(sc.nextInt());
        }

        // Read size of second list
        int n = sc.nextInt();
        LinkedList<Integer> list2 = new LinkedList<>();
        for (int i = 0; i < n; i++) {
            list2.add(sc.nextInt());
        }
    }
}
```

```

// Merge both lists
LinkedList<Integer> mergedList = new LinkedList<>();
mergedList.addAll(list1);
mergedList.addAll(list2);

// Sort merged list
Collections.sort(mergedList);

// Print merged sorted list
for (int num : mergedList) {
    System.out.print(num + " ");
}
}
}

```

**Status :** Correct

**Marks :** 10/10

## 2. Problem Statement

Sarah, a warehouse manager, is managing a list of product names in her store's inventory system. She needs to perform basic operations like adding (inserting) new products, removing products that are sold out or discontinued, displaying all the products in stock, and searching for a specific product in the inventory list.

Sarah's goal is to manage the inventory using a list of product names (strings). The system allows her to perform the following operations using ArrayList:

Insert a Product: Sarah adds a new product to the inventory. Delete a Product: Sarah removes a product from the inventory when it's sold or discontinued. Display the Inventory: Sarah checks all the products currently available in the inventory. Search for a Product: Sarah searches for a specific product in the inventory to check if it's available.

### **Input Format**

The input consists of multiple space-separated values representing different operations on a product list. Each operation follows a specific format:

1 <product\_name> - Adds <product\_name> to the product list.

2 <product\_name> - Removes <product\_name> from the product list if it exists.

3 - Print all products currently on the list.

4 <product\_name> - Checks if <product\_name> exists in the list.

### ***Output Format***

The output displays,

For (choice 1) prints, " <item> has been added to the list."

For (choice 2) prints, " <item> has been removed from the list."

For (choice 3) prints, "Items in the list:" followed by each item in the list on a new line, or "The list is empty." if the list is empty.

For (choice 4) prints, " <item> is found in the list." or " <item> not found in the list."

Refer to the sample output for formatting specifications.

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

Input: 1 apple 1 banana 2 apple 3 4 apple

Output: apple has been added to the list.

banana has been added to the list.

apple has been removed from the list.

Items in the list:

banana

apple not found in the list.

### ***Answer***

```
import java.util.ArrayList;
```

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

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

```
class StringListOperations {
```

```
    // Insert item
```

```

public static void insertItem(ArrayList<String> list, String item) {
    list.add(item);
    System.out.println(item + " has been added to the list.");
}

// Delete item
public static void deleteItem(ArrayList<String> list, String item) {
    if (list.remove(item)) {
        System.out.println(item + " has been removed from the list.");
    } else {
        System.out.println(item + " not found in the list.");
    }
}

// Display list
public static void displayList(ArrayList<String> list) {
    if (list.isEmpty()) {
        System.out.println("The list is empty.");
    } else {
        System.out.println("Items in the list:");
        for (String item : list) {
            System.out.println(item);
        }
    }
}

// Search item
public static void searchItem(ArrayList<String> list, String item) {
    if (list.contains(item)) {
        System.out.println(item + " is found in the list.");
    } else {
        System.out.println(item + " not found in the list.");
    }
}

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

        String input = sc.nextLine();
        String[] commands = input.split(" ");
    }
}

```

```

int i = 0;
while (i < commands.length) {
    int choice = Integer.parseInt(commands[i]);
    switch (choice) {
        case 1:
            if (i + 1 < commands.length) {
                StringListOperations.insertItem(list, commands[i + 1]);
                i += 2;
            } else {
                System.out.println("No string provided for insertion.");
                i++;
            }
            break;
        case 2:
            if (i + 1 < commands.length) {
                StringListOperations.deleteItem(list, commands[i + 1]);
                i += 2;
            } else {
                System.out.println("No string provided for deletion.");
                i++;
            }
            break;
        case 3:
            StringListOperations.displayList(list);
            i += 1;
            break;
        case 4:
            if (i + 1 < commands.length) {
                StringListOperations.searchItem(list, commands[i + 1]);
                i += 2;
            } else {
                System.out.println("No string provided for searching.");
                i++;
            }
            break;
    }
}
}
}
}

```

**Status :** Correct

**Marks :** 10/10

### 3. Problem Statement

Rahul, a stock trader, wants to analyze the stock prices of a company over several days. For each day, he wants to determine the stock span, which is the number of consecutive days (including the current day) where the stock price is less than or equal to the price on that day.

The stock span helps him understand how long a stock has been continuously increasing or staying the same. You need to help Rahul by computing the stock span for each day using a Stack data structure efficiently.

Example:

Input:

7

100 80 60 70 60 75 85

Output:

1 1 1 2 1 4 6

Explanation:

For each day:

Day 1: Price = 100    Span = 1 (Only this day)  
Day 2: Price = 80    Span = 1 (Only this day)  
Day 3: Price = 60    Span = 1 (Only this day)  
Day 4: Price = 70    Span = 2 (Includes today and previous day)  
Day 5: Price = 60    Span = 1 (Only this day)  
Day 6: Price = 75    Span = 4 (Includes today and previous three days)  
Day 7: Price = 85    Span = 6 (Includes today and previous five days)

#### ***Input Format***

The first line contains an integer  $n$ , the number of days.

The second line contains  $n$  space-separated integers  $prices[i]$ , where  $prices[i]$  represents the stock price on the  $i$ -th day.

#### ***Output Format***

The output prints  $n$  space-separated integers representing the stock span for each day.



Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 7

100 80 60 70 60 75 85

Output: 1 1 1 2 1 4 6

### **Answer**

```
import java.util.*;
```

```
public class Main {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        int n = sc.nextInt();  
        int[] prices = new int[n];  
        for (int i = 0; i < n; i++) {  
            prices[i] = sc.nextInt();  
        }  
  
        Stack<Integer> stack = new Stack<>();  
        int[] span = new int[n];  
  
        for (int i = 0; i < n; i++) {  
            while (!stack.isEmpty() && prices[stack.peek()] <= prices[i]) {  
                stack.pop();  
            }  
            span[i] = (stack.isEmpty()) ? (i + 1) : (i - stack.peek());  
            stack.push(i);  
        }  
  
        for (int i = 0; i < n; i++) {  
            System.out.print(span[i]);  
            if (i < n - 1) System.out.print(" ");  
        }  
    }  
}
```

**Status : Correct**

**Marks : 10/10**

#### 4. Problem Statement

Raman, a computer science teacher, is responsible for registering students for his programming class. To streamline the registration process, he wants to develop a program that stores students' names and allows him to retrieve a student's name based on their index in the list.

Raman has decided to use an ArrayList to store the names of students, as it provides efficient dynamic resizing and indexing.

Write a program that enables Raman to input the names of students and fetch a student's name using the specified index. If the entered index is invalid, the program should return an appropriate message.

##### ***Input Format***

The first line of input consists of an integer  $n$ , representing the number of students to register.

The next  $n$  lines of input consist of the names of each student, one by one.

The last line of input is an integer, representing the index (0-indexed) of the element to retrieve.

##### ***Output Format***

If the index is valid (within the bounds of the ArrayList), print "Element at index [index]: " followed by the element (student name as string).

If the index is invalid, print "Invalid index".

Refer to the sample output for formatting specifications.

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

Input: 5

Alice

Bob

Ankit

Alice

Prajit

2

Output: Element at index 2: Ankit

**Answer**

// You are using Java

import java.util.\*;

```
public class Main {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        int n = sc.nextInt();  
        sc.nextLine(); // consume newline  
        ArrayList<String> students = new ArrayList<>();  
  
        for (int i = 0; i < n; i++) {  
            students.add(sc.nextLine());  
        }  
  
        int index = sc.nextInt();  
  
        if (index >= 0 && index < students.size()) {  
            System.out.println("Element at index " + index + ": " + students.get(index));  
        } else {  
            System.out.println("Invalid index");  
        }  
    }  
}
```

**Status : Correct**

**Marks : 10/10**

# Rajalakshmi Engineering College

Name: Sachin Krishna B  
Email: 241901507@rajalakshmi.edu.in  
Roll no: 241901507  
Phone: 9025753177  
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 10\_MCQ

Attempt : 1  
Total Mark : 15  
Marks Obtained : 15

#### Section 1 : MCQ

1. What is the time complexity of retrieving an element from a HashSet?

**Answer**

O(1)

**Status : Correct**

**Marks : 1/1**

2. Which method removes all elements from a Set?

**Answer**

clear()

**Status : Correct**

**Marks : 1/1**

3. How does HashSet check for duplicate elements?

**Answer**

Using equals() and hashCode()

**Status : Correct**

**Marks : 1/1**

4. What happens when you add duplicate elements to a HashSet?

**Answer**

The duplicate is ignored

**Status : Correct**

**Marks : 1/1**

5. What will be the output of the following code?

```
import java.util.*;
class Main {
    public static void main(String[] args) {
        HashMap<String, Integer> map = new HashMap<>();
        map.put("X", 10);
        map.put("Y", 20);
        map.put("Z", 30);
        map.remove("Y");
        System.out.println(map);
    }
}
```

**Answer**

{X=10, Z=30}

**Status : Correct**

**Marks : 1/1**

6. What will happen if you add elements in descending order in a TreeSet?

**Answer**

They are sorted in ascending order

**Status :** Correct

**Marks :** 1/1

7. What happens if two keys have the same hash code in a HashMap?

**Answer**

A linked list is used to store values with the same hash

**Status :** Correct

**Marks :** 1/1

8. Which of the following is true about TreeMap?

**Answer**

It maintains natural ordering

**Status :** Correct

**Marks :** 1/1

9. What will happen if you add a null element to a TreeSet?

**Answer**

An exception occurs

**Status :** Correct

**Marks :** 1/1

10. Which of the following is true about HashMap?

**Answer**

It is not synchronized

**Status :** Correct

**Marks :** 1/1

11. Which of the following allows null keys in Java?

**Answer**

HashMap

**Status :** Correct

**Marks :** 1/1

12. Which method retrieves the lowest key in a TreeMap?

**Answer**

firstKey()

**Status :** Correct

**Marks :** 1/1

13. What will be the output of the following code?

```
import java.util.*;
class Main {
    public static void main(String[] args) {
        HashMap<String, Integer> map = new HashMap<>();
        map.put("A", 1);
        map.put("B", 2);
        map.put("C", 3);
        System.out.println(map.containsKey("B"));
    }
}
```

**Answer**

true

**Status :** Correct

**Marks :** 1/1

14. Which statement is true about HashSet and TreeSet?

**Answer**

TreeSet provides sorted elements

**Status :** Correct

**Marks :** 1/1

15. What will be the output of the following code?

```
import java.util.*;
class Main {
    public static void main(String[] args) {
        HashMap<String, String> map = new HashMap<>();
    }
}
```

```
map.put("A", "Apple");  
map.put("B", "Banana");  
map.put("C", "Cherry");  
map.replace("B", "Blueberry");  
System.out.println(map);  
}  
}
```

**Answer**

{A=Apple, B=Blueberry, C=Cherry}

**Status :** Correct

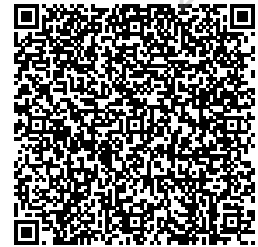
**Marks :** 1/1



# Rajalakshmi Engineering College

Name: Sachin Krishna B  
Email: 241901507@rajalakshmi.edu.in  
Roll no: 241901507  
Phone: 9025753177  
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

### 2028\_REC\_OOPS using Java\_Week 10\_Q1

Attempt : 1  
Total Mark : 10  
Marks Obtained : 10

#### Section 1 : COD

##### 1. Problem Statement

A city traffic management system needs to track vehicles entering a toll booth. Each vehicle is uniquely identified by its registration number. The system should allow adding vehicles to a record, ensuring that no duplicate registration numbers exist. The vehicles should be stored in a HashSet, which does not guarantee any specific order.

Your task is to implement a program using a HashSet that allows adding vehicle details and displaying the records.

##### ***Input Format***

The first line of input contains an integer N - the number of vehicles.

The next N lines contain details of each vehicle in the format: "RegNumber

OwnerName VehicleType"

1. RegNumber (String) - A unique registration number (Alphanumeric).
2. OwnerName (String) - The name of the vehicle owner.
3. VehicleType (String, Car, Bike, or Truck) - The type of vehicle.

If a vehicle with the same registration number is already present, ignore the duplicate entry.

### **Output Format**

The output prints the unique vehicle records in any order (since HashSet does not maintain order).

Output format: "RegNumber OwnerName VehicleType"

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 5

KA01AB1234 John Car

MH02CD5678 Alice Bike

DL03EF9012 Bob Truck

TN04GH3456 Mike Car

KA01AB1234 John Car

Output: TN04GH3456 Mike Car

KA01AB1234 John Car

MH02CD5678 Alice Bike

DL03EF9012 Bob Truck

### **Answer**

```
import java.util.*;
```

```
class Vehicle {
```

```
    private String regNumber;
```

```
    private String ownerName;
```

```
    private String vehicleType;
```

```
    public Vehicle(String regNumber, String ownerName, String vehicleType) {
```

```
        this.regNumber = regNumber;
```

```

        this.ownerName = ownerName;
        this.vehicleType = vehicleType;
    }

    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (obj == null || getClass() != obj.getClass()) return false;
        Vehicle vehicle = (Vehicle) obj;
        return regNumber.equals(vehicle.regNumber);
    }

    public int hashCode() {
        return regNumber.hashCode();
    }

    public String toString() {
        return regNumber + " " + ownerName + " " + vehicleType;
    }
}

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

        int n = scanner.nextInt();
        scanner.nextLine();

        for (int i = 0; i < n; i++) {
            String line = scanner.nextLine();
            String[] parts = line.split(" ");

            String regNumber = parts[0];
            String ownerName = parts[1];
            String vehicleType = parts[2];

            Vehicle vehicle = new Vehicle(regNumber, ownerName, vehicleType);
            vehicleSet.add(vehicle);
        }

        for (Vehicle vehicle : vehicleSet) {
            System.out.println(vehicle);
        }
    }
}

```

```
    scanner.close();  
  }  
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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

### 2028\_REC\_OOPS using Java\_Week 10\_Q2

Attempt : 1

Total Mark : 10

Marks Obtained : 10

#### Section 1 : COD

##### 1. Problem Statement

John is organizing a fruit festival, and the quantities of various fruits are stored in a HashMap where fruit names are keys and quantities are values.

Help him develop a program to find the total quantity of fruits for the festival by summing up the values in the HashMap.

##### ***Input Format***

The input consists of fruit quantities in the format 'fruitName:quantity', where fruitName is the name of the fruit(a string), and quantity is a double value representing the quantity.

The input is terminated by entering "done".

##### ***Output Format***

The output prints a double value, representing the sum of values in the HashMap, rounded off to two decimal places.

If the value is not numeric, print "Invalid input".

If any special characters other than ':' are entered, print "Invalid format".

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: Banana:15.2

Orange:56.3

Mango:47.3

done

Output: 118.80

### **Answer**

```
import java.util.*;
```

```
import java.text.DecimalFormat;
```

```
public class Main {  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);  
        HashMap<String, Double> fruitMap = new HashMap<>();  
        DecimalFormat df = new DecimalFormat("0.00");  
  
        while (true) {  
            String input = scanner.nextLine();  
  
            if (input.equals("done")) {  
                break;  
            }  
  
            // Check if input contains ':' as separator  
            if (!input.contains(":")) {  
                System.out.println("Invalid format");  
                return;  
            }  
        }  
    }  
}
```

```

String[] parts = input.split(":");

// Check if we have exactly two parts
if (parts.length != 2) {
    System.out.println("Invalid format");
    return;
}

String fruitName = parts[0];
String quantityStr = parts[1];

// Try to parse the quantity as double
try {
    double quantity = Double.parseDouble(quantityStr);
    fruitMap.put(fruitName, quantity);
} catch (NumberFormatException e) {
    System.out.println("Invalid input");
    return;
}

// Calculate total quantity
double total = 0.0;
for (double quantity : fruitMap.values()) {
    total += quantity;
}

// Print total rounded to 2 decimal places
System.out.println(df.format(total));

scanner.close();
}
}

```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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

### 2028\_REC\_OOPS using Java\_Week 10\_Q3

Attempt : 1

Total Mark : 10

Marks Obtained : 10

#### Section 1 : COD

##### 1. Problem Statement

Priya is analyzing encrypted messages in a research project. She wants to analyze the frequency of each character in a given paragraph. The characters should be stored in a TreeMap so that the output is sorted in ascending order of characters automatically.

You are required to build a Java program that:

Uses a `TreeMap<Character, Integer>` to count how many times each character appears in the message. Ignores spaces and considers only alphabets (case-sensitive). Outputs the frequencies of characters in sorted order.

You must use a TreeMap in the class named MessageAnalyzer.

**Input Format**



The first line of input contains an integer n, the number of lines in the message.

The next n lines each contain a string (the encrypted message line).

### **Output Format**

The first line of output prints: "Character Frequency:"

Then print each character and its frequency in the format: "<character>: <count>"

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 2  
Hello World  
Java

Output: Character Frequency:

H: 1

J: 1

W: 1

a: 2

d: 1

e: 1

l: 3

o: 2

r: 1

v: 1

### **Answer**

```
// You are using Java
import java.util.*;
```

```
class MessageAnalyzer {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        TreeMap<Character, Integer> charFrequency = new TreeMap<>();

        int n = Integer.parseInt(scanner.nextLine());

        for (int i = 0; i < n; i++) {
```

```

String line = scanner.nextLine();

for (char ch : line.toCharArray()) {
    if (ch != ' ') {
        if (charFrequency.containsKey(ch)) {
            charFrequency.put(ch, charFrequency.get(ch) + 1);
        } else {
            charFrequency.put(ch, 1);
        }
    }
}

System.out.println("Character Frequency:");
for (Map.Entry<Character, Integer> entry : charFrequency.entrySet()) {
    System.out.println(entry.getKey() + ": " + entry.getValue());
}

scanner.close();
}
}

```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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

### 2028\_REC\_OOPS using Java\_Week 10\_Q4

Attempt : 1

Total Mark : 10

Marks Obtained : 10

#### Section 1 : COD

##### 1. Problem Statement

In a ticket reservation system, you store the available seat numbers in a TreeSet. Users input their desired seat number, and the program checks whether the chosen seat is available.

Using a TreeSet ensures quick and efficient verification of seat availability, ensuring a smooth and organized ticket booking process.

##### ***Input Format***

The first line of input contains a single integer  $n$ , representing the number of available seats.

The second line contains  $n$  space-separated integers, representing the available seat numbers.

The third line contains an integer  $m$ , representing the seat number that needs to be searched.

### **Output Format**

The output displays "[ $m$ ] is present!" if the given seat is available. Otherwise, it displays "[ $m$ ] is not present!"

Refer to the sample output for the formatting specifications.

### **Sample Test Case**

Input: 4

2 4 5 6

5

Output: 5 is present!

### **Answer**

// You are using Java

import java.util.\*;

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

        int n = scanner.nextInt();

        for (int i = 0; i < n; i++) {
            int seat = scanner.nextInt();
            availableSeats.add(seat);
        }

        int m = scanner.nextInt();

        if (availableSeats.contains(m)) {
            System.out.println(m + " is present!");
        } else {
            System.out.println(m + " is not present!");
        }
    }
}
```

```
}  
    scanner.close();  
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B  
Email: 241901507@rajalakshmi.edu.in  
Roll no: 241901507  
Phone: 9025753177  
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 10\_PAH

Attempt : 1  
Total Mark : 30  
Marks Obtained : 30

#### Section 1 : Coding

##### 1. Problem Statement

Sarah is working on a spam detection system that analyzes incoming messages for unique patterns. Spammers often use repetitive character sequences, making it important to identify the first non-repeating character in a message.

Given a string, Sarah needs to determine the first character that appears only once. If all characters repeat, the system should return -1.

She decides to use a HashMap to efficiently track character frequencies and find the solution.

##### ***Input Format***

The first line contains an integer N representing , the length of the string.

The second line contains a string of N lowercase English letters (a-z).

### **Output Format**

The output prints a character representing the first non-repeating character. If none exist, print -1.

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 10  
abacabadac

Output: d

### **Answer**

```
import java.util.*;

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

        int n = scanner.nextInt();
        scanner.nextLine(); // consume newline
        String str = scanner.nextLine();

        HashMap<Character, Integer> charCount = new HashMap<>();

        // First pass: count frequency of each character
        for (int i = 0; i < n; i++) {
            char ch = str.charAt(i);
            charCount.put(ch, charCount.getOrDefault(ch, 0) + 1);
        }

        // Second pass: find first character with count 1
        char result = '-';
        boolean found = false;
        for (int i = 0; i < n; i++) {
            char ch = str.charAt(i);
```

```
        if (charCount.get(ch) == 1) {
            result = ch;
            found = true;
            break;
        }
    }

    if (found) {
        System.out.println(result);
    } else {
        System.out.println(-1);
    }

    scanner.close();
}
```

**Status :** Correct

**Marks :** 10/10

## 2. Problem Statement

A university maintains a list of student records and wants to store them in a sorted manner based on their GPA. If two students have the same GPA, they should be further sorted by their name in lexicographical order. Implement a program that uses a TreeSet to store student records and ensures unique student IDs.

### **Input Format**

The first line contains an integer N - the number of students.

The next N lines contain details of each student in the format: "StudentID Name GPA"

- StudentID (Integer) - A unique identifier.
- Name (String) - The student's name (can contain spaces).
- GPA (Double) - The Grade Point Average.

### **Output Format**

The output prints the list of students in ascending order of GPA.



If two students have the same GPA, sort them by name.

Print details in the format: "StudentID Name GPA" in the output, GPA is rounded to two decimal places.

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 5

101 John 8.5

102 Alice 9.1

103 Bob 8.5

104 Zoe 7.3

105 Charlie 9.1

Output: 104 Zoe 7.30

103 Bob 8.50

101 John 8.50

102 Alice 9.10

105 Charlie 9.10

### **Answer**

```
import java.util.*;
```

```
class Student implements Comparable<Student> {  
    private int studentID;  
    private String name;  
    private double gpa;  
  
    public Student(int studentID, String name, double gpa) {  
        this.studentID = studentID;  
        this.name = name;  
        this.gpa = gpa;  
    }  
  
    public int getStudentID() { return studentID; }  
    public String getName() { return name; }  
    public double getGpa() { return gpa; }  
  
    @Override
```

```
    @Override
```

```
public int compareTo(Student other) {  
    // First compare by GPA  
    if (this.gpa != other.gpa) {  
        return Double.compare(this.gpa, other.gpa);  
    }  
    // If GPA is same, compare by name  
    return this.name.compareTo(other.name);  
}
```

```
@Override  
public String toString() {  
    return String.format("%d %s %.2f", studentID, name, gpa);  
}
```

```
@Override  
public boolean equals(Object obj) {  
    if (this == obj) return true;  
    if (obj == null || getClass() != obj.getClass()) return false;  
    Student student = (Student) obj;  
    return studentID == student.studentID;  
}
```

```
@Override  
public int hashCode() {  
    return Objects.hash(studentID);  
}  
}
```

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

```
        int n = scanner.nextInt();  
        scanner.nextLine(); // consume newline
```

```
        for (int i = 0; i < n; i++) {  
            String line = scanner.nextLine();  
            String[] parts = line.split(" ", 3);  
  
            int studentID = Integer.parseInt(parts[0]);  
            String name = parts[1];
```

```

double gpa = Double.parseDouble(parts[2]);

Student student = new Student(studentID, name, gpa);
studentSet.add(student);
}

// Print all students in sorted order
for (Student student : studentSet) {
    System.out.println(student);
}

scanner.close();
}
}

```

**Status :** Correct

**Marks :** 10/10

### 3. Problem Statement

Riya is building a calendar event scheduler where each event is stored in chronological order using a TreeMap. The key represents the event time in 24-hour format (HH:MM), and the value is the event description.

She wants the system to:

Automatically sort events by time. Avoid duplicate time entries — if a duplicate time is entered, ignore the new entry. Print all scheduled events in order.

Implement this logic using a class named EventManager.

#### **Input Format**

The first line of the input contains an integer  $n$ , representing the number of events.

The next  $n$  lines each contain a string in the format: "HH:MM Description"

(Example: 09:00 TeamMeeting).

#### **Output Format**

The first line of the output prints "Scheduled Events:"

The next k lines print each event in the format: "HH:MM - Description"

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 5

09:00 TeamMeeting

13:30 LunchBreak

11:00 ProjectUpdate

09:00 Standup

15:00 ClientCall

Output: Scheduled Events:

09:00 - TeamMeeting

11:00 - ProjectUpdate

13:30 - LunchBreak

15:00 - ClientCall

### **Answer**

```
import java.util.*;
```

```
class EventManager {  
    private TreeMap<String, String> events;  
  
    public EventManager() {  
        events = new TreeMap<>();  
    }  
  
    public void addEvent(String time, String description) {  
        // Use putIfAbsent to ignore duplicates  
        events.putIfAbsent(time, description);  
    }  
  
    public void displayEvents() {  
        System.out.println("Scheduled Events:");  
        for (Map.Entry<String, String> entry : events.entrySet()) {  
            System.out.println(entry.getKey() + " - " + entry.getValue());  
        }  
    }  
}
```

```
}  
}  
public class Main {  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);  
        EventManager eventManager = new EventManager();  
  
        int n = scanner.nextInt();  
        scanner.nextLine(); // consume newline  
  
        for (int i = 0; i < n; i++) {  
            String line = scanner.nextLine();  
            String[] parts = line.split(" ", 2);  
  
            String time = parts[0];  
            String description = parts[1];  
            eventManager.addEvent(time, description);  
        }  
  
        eventManager.displayEvents();  
        scanner.close();  
    }  
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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 10\_CY

Attempt : 2

Total Mark : 40

Marks Obtained : 40

#### Section 1 : COD

##### 1. Problem Statement

A college professor wants to keep track of students who attend classes. Each student has a unique roll number and their attendance count increases every time they attend a class. The system should allow adding a student, marking their attendance, and displaying all students with their total attendance.

Your task is to implement a Java program using TreeSet to maintain students in sorted order of roll numbers and track their attendance count.

Operations:

A roll\_no name Add a student with roll number and name (if not already added).M roll\_no Mark attendance for the student with the given roll number (increase their count by 1).D Display all students in ascending order of roll number along with their attendance count.

### ***Input Format***

The first line contains an integer N - the number of students.

The next N lines contain one of the following commands:

A roll\_no name

M roll\_no

D

- A (Add) Adds a new student with a unique roll number and name.
- M (Mark) Increases attendance count for the given roll number.
- D (Display) Prints all students in ascending order of roll number.

### ***Output Format***

For D, output prints each student's roll number, name, and attendance count in ascending order of roll number.

Refer to the sample output for formatting specifications.

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

Input: 5

A 101 Alice

A 102 Bob

M 101

M 101

D

Output: 101 Alice 2

102 Bob 0

### ***Answer***

```
import java.util.*;
```

```
class Student implements Comparable<Student> {  
    private int rollNo;  
    private String name;  
    private int attendanceCount;
```

```
public Student(int rollNo, String name) {  
    this.rollNo = rollNo;  
    this.name = name;  
    this.attendanceCount = 0;  
}
```

```
public int getRollNo() {  
    return rollNo;  
}
```

```
public String getName() {  
    return name;  
}
```

```
public int getAttendanceCount() {  
    return attendanceCount;  
}
```

```
public void markAttendance() {  
    this.attendanceCount++;  
}
```

```
@Override  
public int compareTo(Student other) {  
    return Integer.compare(this.rollNo, other.rollNo);  
}
```

```
@Override  
public boolean equals(Object obj) {  
    if (this == obj) return true;  
    if (obj == null || getClass() != obj.getClass()) return false;  
    Student student = (Student) obj;  
    return rollNo == student.rollNo;  
}
```

```
@Override  
public int hashCode() {  
    return Objects.hash(rollNo);  
}
```

```
@Override
```



```

    public String toString() {
        return rollNo + " " + name + " " + attendanceCount;
    }
}

class AttendanceSystem {
    private TreeSet<Student> students;

    public AttendanceSystem() {
        students = new TreeSet<>();
    }

    public void addStudent(int rollNo, String name) {
        Student newStudent = new Student(rollNo, name);
        if (!students.contains(newStudent)) {
            students.add(newStudent);
        }
    }

    public void markAttendance(int rollNo) {
        Student tempStudent = new Student(rollNo, "");
        Student foundStudent = students.floor(tempStudent);
        if (foundStudent != null && foundStudent.getRollNo() == rollNo) {
            foundStudent.markAttendance();
        }
    }

    public void displayStudents() {
        for (Student student : students) {
            System.out.println(student);
        }
    }
}

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

        int N = scanner.nextInt();
        scanner.nextLine();
    }
}

```

```

for (int i = 0; i < N; i++) {
    String line = scanner.nextLine();
    String[] parts = line.split(" ");

    String command = parts[0];

    if (command.equals("A")) {
        int rollNo = Integer.parseInt(parts[1]);
        String name = parts[2];
        system.addStudent(rollNo, name);
    } else if (command.equals("M")) {
        int rollNo = Integer.parseInt(parts[1]);
        system.markAttendance(rollNo);
    } else if (command.equals("D")) {
        system.displayStudents();
    }
}

scanner.close();
}
}

```

**Status :** Correct

**Marks :** 10/10

## 2. Problem Statement

Bob wants to develop a score-tracking application for a gaming tournament. Each player's score is stored in a HashMap with the player's name as the key and the score as the value.

Write a program to assist Bob that takes user input to enter player scores, calculates the maximum score from the HashMap, and prints the player with the highest score.

### **Input Format**

The input consists of strings representing player details in the format "playerName:score".

The input is terminated by entering "done".

### **Output Format**

The output displays a string, representing the player's name who scored the maximum.

If the value is not numeric, print "Invalid input".

If any special characters other than ':' are given, print "Invalid format".

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: Alice:15

Bob:56

done

Output: Bob

### **Answer**

```
import java.util.*;
```

```
class ScoreTracker {  
    HashMap<String, Integer> scoreMap;  
  
    public ScoreTracker() {  
        scoreMap = new HashMap<>();  
    }  
  
    public boolean processInput(String input) {  
        // Check if input contains exactly one colon  
        if (countOccurrences(input, ':') != 1) {  
            System.out.println("Invalid format");  
            return false;  
        }  
  
        String[] parts = input.split(":");  
        if (parts.length != 2) {  
            System.out.println("Invalid format");  
            return false;  
        }  
    }  
}
```

```

    }

    String playerName = parts[0].trim();
    String scoreStr = parts[1].trim();

    // Validate player name (only letters and spaces)
    if (!playerName.matches("[a-zA-Z ]+")) {
        System.out.println("Invalid format");
        return false;
    }

    // Validate and parse score
    try {
        int score = Integer.parseInt(scoreStr);
        if (score < 1 || score > 100) {
            System.out.println("Invalid input");
            return false;
        }
        scoreMap.put(playerName, score);
    } catch (NumberFormatException e) {
        System.out.println("Invalid input");
        return false;
    }

    return true;
}

public String findTopPlayer() {
    if (scoreMap.isEmpty()) {
        return "";
    }

    String topPlayer = "";
    int maxScore = Integer.MIN_VALUE;

    for (Map.Entry<String, Integer> entry : scoreMap.entrySet()) {
        if (entry.getValue() > maxScore) {
            maxScore = entry.getValue();
            topPlayer = entry.getKey();
        }
    }
}

```

```

        return topPlayer;
    }

    private int countOccurrences(String str, char ch) {
        int count = 0;
        for (char c : str.toCharArray()) {
            if (c == ch) {
                count++;
            }
        }
        return count;
    }
}

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

        while (true) {
            String input = scanner.nextLine();

            if (input.toLowerCase().equals("done")) {
                break;
            }

            if (!tracker.processInput(input)) {
                validInput = false;
                break;
            }
        }

        if (validInput && !tracker.scoreMap.isEmpty()) {
            System.out.println(tracker.findTopPlayer());
        }

        scanner.close();
    }
}

```

**Status :** Correct

**Marks :** 10/10

### 3. Problem Statement

David is managing an employee database where each employee has a unique ID, name, and department. He wants to ensure that duplicate employee IDs are not added to the system. Implement a Java program that allows adding employees to the system, displaying all employees, and checking if an employee exists based on the given ID.

Implement a class `EmployeeDatabase` that contains a `HashSet` to store employee records. The `Employee` class should be a user-defined object containing employee details. The main class should handle user operations and interact with the `EmployeeDatabase` class.

#### ***Input Format***

The first line contains an integer  $n$  representing the number of employees to be added.

The next  $n$  lines follow, each containing:

1. An integer `employee_id`
2. A string `name`
3. A string `department`

The next line contains an integer  $m$  representing the number of queries.

The next  $m$  lines follow, each containing an employee ID to check for existence.

#### ***Output Format***

The output prints a list of all employees added in the format:

"ID: <employee\_id>, Name: <name>, Department: <department>"

For each query, output "Employee exists" if the ID is found, otherwise "Employee not found".

Refer to the sample output for formatting specifications.

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

Input: 3

101 John IT

102 Alice HR

103 Bob Finance

2

101

104

Output: ID: 101, Name: John, Department: IT

ID: 102, Name: Alice, Department: HR

ID: 103, Name: Bob, Department: Finance

Employee exists

Employee not found

**Answer**

```
import java.util.*;
```

```
class Employee {
```

```
    private int id;
```

```
    private String name;
```

```
    private String department;
```

```
    public Employee(int id, String name, String department) {
```

```
        this.id = id;
```

```
        this.name = name;
```

```
        this.department = department;
```

```
    }
```

```
    public int getId() {
```

```
        return id;
```

```
    }
```

```
    public String getName() {
```

```
        return name;
```

```
    }
```

```
    public String getDepartment() {
```

```
        return department;
```

```
    }
```

```
    @Override
```

```
public boolean equals(Object obj) {  
    if (this == obj) return true;  
    if (obj == null || getClass() != obj.getClass()) return false;  
    Employee employee = (Employee) obj;  
    return id == employee.id;  
}
```

```
@Override  
public int hashCode() {  
    return Objects.hash(id);  
}
```

```
@Override  
public String toString() {  
    return "ID: " + id + ", Name: " + name + ", Department: " + department;  
}  
}
```

```
class EmployeeDatabase {  
    private HashSet<Employee> employees;
```

```
    public EmployeeDatabase() {  
        employees = new HashSet<>();  
    }
```

```
    public void addEmployee(int id, String name, String department) {  
        Employee employee = new Employee(id, name, department);  
        employees.add(employee);  
    }
```

```
    public void displayEmployees() {  
        for (Employee employee : employees) {  
            System.out.println(employee);  
        }  
    }
```

```
    public boolean checkEmployee(int id) {  
        for (Employee employee : employees) {  
            if (employee.getId() == id) {  
                return true;  
            }  
        }  
    }
```



```

        return false;
    }
}

class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        EmployeeDatabase db = new EmployeeDatabase();
        int n = sc.nextInt();
        for (int i = 0; i < n; i++) {
            int id = sc.nextInt();
            String name = sc.next();
            String department = sc.next();
            db.addEmployee(id, name, department);
        }
        db.displayEmployees();
        int m = sc.nextInt();
        for (int i = 0; i < m; i++) {
            int id = sc.nextInt();
            if (db.checkEmployee(id))
                System.out.println("Employee exists");
            else
                System.out.println("Employee not found");
        }
        sc.close();
    }
}

```

**Status :** Correct

**Marks : 10/10**

#### 4. Problem Statement

A linguist named Meera is classifying a list of words based on their first character. She wants to store words grouped by their starting letter using a TreeMap so that the groups appear in sorted order of characters (i.e., 'a' to 'z'). For each letter, all words starting with that letter should be stored in the order they appear.

Implement the logic inside a class named WordClassifier using the `TreeMap<Character, List<String>>` collection.

### ***Input Format***

The first line of the input contains an integer n, representing the number of words.

The next n lines each contain a word.

### ***Output Format***

The first line of the output prints: "Grouped Words by Starting Letter:"

The next lines print each character key and its list of words in the format:

"letter: word1 word2 word3..."

...

Refer to the sample output for formatting specifications.

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

Input: 5

dog

deer

cat

cow

camel

Output: Grouped Words by Starting Letter:

c: cat cow camel

d: dog deer

### ***Answer***

```
import java.util.*;
```

```
class WordClassifier {  
    private TreeMap<Character, List<String>> wordMap;  
  
    public WordClassifier() {  
        wordMap = new TreeMap<>();  
    }  
}
```

```

    }

    public void classifyWords(List<String> words) {
        for (String word : words) {
            if (word == null || word.isEmpty()) {
                continue;
            }

            char firstChar = word.charAt(0);

            // If the key doesn't exist, create a new list
            if (!wordMap.containsKey(firstChar)) {
                wordMap.put(firstChar, new ArrayList<>());
            }

            // Add the word to the list for that character
            wordMap.get(firstChar).add(word);
        }

        displayGroupedWords();
    }

    private void displayGroupedWords() {
        System.out.println("Grouped Words by Starting Letter:");

        for (Map.Entry<Character, List<String>> entry : wordMap.entrySet()) {
            char letter = entry.getKey();
            List<String> wordList = entry.getValue();

            System.out.print(letter + ": ");

            // Print all words for this letter
            for (int i = 0; i < wordList.size(); i++) {
                System.out.print(wordList.get(i));
                if (i < wordList.size() - 1) {
                    System.out.print(" ");
                }
            }
            System.out.println();
        }
    }
}

```

```
public class Main {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        int n = Integer.parseInt(sc.nextLine());  
  
        List<String> words = new ArrayList<>();  
        for (int i = 0; i < n; i++) {  
            words.add(sc.nextLine());  
        }  
  
        WordClassifier classifier = new WordClassifier();  
        classifier.classifyWords(words);  
    }  
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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 11

Attempt : 1

Total Mark : 20

Marks Obtained : 20

### Section 1 : Project

#### 1. Problem Statement

Create a JDBC-based Hospital Management System that handles runtime input to manage patient records. The system should allow users to:

Add a new patient (patient ID, name, age, status).

Update a patient's status.

View a specific patient's record by patient ID.

Display all patient records in the database.

Exit the application.

The system should connect to a MySQL database using the following default credentials:

DB URL: jdbc:mysql://localhost/ri\_db

USER: test

PWD: test123

The patients table has already been created with the following structure:

Table Name: patients

### ***Input Format***

The first line of input consists of an integer choice, representing the operation to be performed:

(1 for Add Patient, 2 for Update Patient Status, 3 for View Patient Record, 4 for Display All Patients, 5 for Exit)

For choice 1 (Add Patient):

- The second line consists of an integer patient\_id.
- The third line consists of a string name.
- The fourth line consists of an integer age.
- The fifth line consists of a string status.

For choice 2 (Update Patient Status):

- The second line consists of an integer patient\_id.
- The third line consists of a string new\_status.

For choice 3 (View Patient Record):

- The second line consists of an integer patient\_id.

For choice 4 (Display All Patients):

- No additional inputs are required.

For choice 5 (Exit):

- No additional inputs are required.

### **Output Format**

For choice 1 (Add Patient):

- Print "Patient added successfully" if the patient was added.
- Print "Failed to add patient." if the insertion failed.

For choice 2 (Update Patient Status):

- Print "Patient status updated successfully" if the update was successful.
- Print "Patient not found." if the specified patient ID does not exist.

For choice 3 (View Patient Record):

- Display the patient details in the format:
  - ID: [patient\_id] | Name: [name] | Age: [age] | Status: [status]
- Print "Patient not found." if the specified patient ID does not exist.

For choice 4 (Display All Patients):

- Display each patient on a new line in the format:
  - ID | Name | Age | Status
- If no records are available, print nothing (or handle it with an appropriate message if desired).

For choice 5 (Exit):

- Print "Exiting Hospital Management System."

For invalid input:

- Print "Invalid choice. Please try again."

### **Sample Test Case**

Input: 1

101

John Doe

45

Admitted

4

5

Output: Patient added successfully

ID | Name | Age | Status

101 | John Doe | 45 | Admitted

Exiting Hospital Management System.

### **Answer**

```
import java.sql.*;
```

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

```
class HospitalManagementSystem {
```

```
    public static void main(String[] args) {
```

```
        try (Connection conn = DriverManager.getConnection("jdbc:mysql://localhost/ri_db", "test", "test123"));
```

```
            Scanner scanner = new Scanner(System.in)) {
```

```
                boolean running = true;
```

```
                while (running) {
```

```
                    int choice = scanner.nextInt();
```

```
                    switch (choice) {
```

```
                        case 1:
```

```
                            addPatient(conn, scanner);
```

```
                            break;
```

```
                        case 2:
```

```
                            updatePatientStatus(conn, scanner);
```

```
                            break;
```

```
                        case 3:
```

```
                            viewPatientRecord(conn, scanner);
```

```
                            break;
```

```
                        case 4:
```

```
                            displayAllPatients(conn);
```

```
                            break;
```

```
                        case 5:
```

```
                            System.out.println("Exiting Hospital Management System.");
```

```
                            running = false;
```

```
                            break;
```

```
                    default:
```

```
                        System.out.println("Invalid choice. Please try again.");
```

```
                }
```



```

    }
} catch (SQLException e) {
    e.printStackTrace();
}
}

```

```

public static void addPatient(Connection conn, Scanner scanner) {
    int patientId = scanner.nextInt();
    scanner.nextLine(); // consume newline
    String name = scanner.nextLine();
    int age = scanner.nextInt();
    scanner.nextLine(); // consume newline
    String status = scanner.nextLine();

```

```

    String sql = "INSERT INTO patients (patient_id, name, age, status) VALUES
    (?, ?, ?, ?)";

```

```

    try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
        pstmt.setInt(1, patientId);
        pstmt.setString(2, name);
        pstmt.setInt(3, age);
        pstmt.setString(4, status);

        int rowsAffected = pstmt.executeUpdate();
        if (rowsAffected > 0) {
            System.out.println("Patient added successfully");
        } else {
            System.out.println("Failed to add patient.");
        }
    } catch (SQLException e) {
        System.out.println("Failed to add patient.");
    }
}

```

```

public static void updatePatientStatus(Connection conn, Scanner scanner) {
    int patientId = scanner.nextInt();
    scanner.nextLine(); // consume newline
    String newStatus = scanner.nextLine();

```

```

    String sql = "UPDATE patients SET status = ? WHERE patient_id = ?";

```

```

try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
    pstmt.setString(1, newStatus);
    pstmt.setInt(2, patientId);

    int rowsAffected = pstmt.executeUpdate();
    if (rowsAffected > 0) {
        System.out.println("Patient status updated successfully");
    } else {
        System.out.println("Patient not found.");
    }
} catch (SQLException e) {
    System.out.println("Patient not found.");
}

public static void viewPatientRecord(Connection conn, Scanner scanner) {
    int patientId = scanner.nextInt();
    scanner.nextLine(); // consume newline

    String sql = "SELECT * FROM patients WHERE patient_id = ?";

    try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
        pstmt.setInt(1, patientId);
        ResultSet rs = pstmt.executeQuery();

        if (rs.next()) {
            int id = rs.getInt("patient_id");
            String name = rs.getString("name");
            int age = rs.getInt("age");
            String status = rs.getString("status");

            System.out.println("ID: " + id + " | Name: " + name + " | Age: " + age + " | Status: " + status);
        } else {
            System.out.println("Patient not found.");
        }
    } catch (SQLException e) {
        System.out.println("Patient not found.");
    }

    public static void displayAllPatients(Connection conn) {

```

```

String sql = "SELECT * FROM patients";

try (Statement stmt = conn.createStatement();
    ResultSet rs = stmt.executeQuery(sql)) {

    System.out.println("ID | Name | Age | Status");
    while (rs.next()) {
        int id = rs.getInt("patient_id");
        String name = rs.getString("name");
        int age = rs.getInt("age");
        String status = rs.getString("status");

        System.out.println(id + " | " + name + " | " + age + " | " + status);
    }
} catch (SQLException e) {
    // If no records are available, print nothing as per requirements
}

}
}

```

**Status :** Correct

**Marks :** 10/10

## 2. Problem Statement

In Café Central, the menu is cataloged and stored in a database.

To efficiently manage the restaurant's menu using Java and JDBC, you must build a Restaurant Management System that supports:

Adding new menu items

Updating menu item prices

Viewing details of a menu item

Displaying all menu items in sorted order

You are given two files:

File 1: MenuItem.java (POJO Class)

This class represents the MenuItem entity.

A MenuItem contains the following details:

Field	Description
itemId	Unique Menu Item ID (Integer)
name	Item Name (String)
category	Item Category (String)
price	Item Price (Double)

Students must write code in the marked area:

```
class MenuItem {  
    private int itemId;  
    private String name;  
    private String category;  
    private double price;  
  
    public MenuItem() {}  
  
    public MenuItem(int itemId, String name, String category, double price) {  
        // write your code here  
    }  
  
    // Include getters and setters  
}
```

Expected in this part:

Assign parameter values to instance variables inside the constructor.

Add getters and setters for all attributes.

File 2: MenuItemDAO.java (Data Access Layer)

This class handles all database operations using JDBC.

Students must complete the missing JDBC logic in the following methods:

```
class MenuItemDAO {

    public void addItem(Connection conn, MenuItem menuItem)
    throws SQLException {

        // write your code here

    }

    public void updateItemPrice(Connection conn, int itemId, double
    newPrice) throws SQLException {

        // write your code here

    }

    public void deleteMenuItem(Connection conn, int itemId) throws
    SQLException {

        // write your code here

    }

    public MenuItem viewItemDetails(Connection conn, int itemId) throws
    SQLException {

        // write your code here

    }

    public List<MenuItem> displayAllMenuItems(Connection conn) throws
    SQLException {

        // write your code here

    }

    private MenuItem mapToMenuItem(ResultSet rs) throws SQLException {
        return new MenuItem(
```

```
// write your code here
);
}
}
```

Expected in this part:

Write SQL queries for INSERT, UPDATE, DELETE, SELECT.

Execute queries using PreparedStatement or Statement.

Map ResultSet rows to MenuItem objects using mapToMenuItem().

Return a List<MenuItem> where required.

The system should connect to a MySQL database using the following default credentials:

DB URL: jdbc:mysql://localhost/ri\_db

USER: test

PWD: test123

The menu table has already been created with the following structure:

Table Name: menu

### ***Input Format***

The first line of input consists of an integer choice, representing the operation to be performed (1 for Add Item, 2 for Restock item, 3 for reduce item, 4 for Display, 5 for Exit).

For choice 1 (Add Menu Item):

- The second line consists of an integer item\_id.
- The third line consists of a string name.
- The fourth line consists of a string category.
- The fifth line consists of a double price.

For choice 2 (Update Item Price):

- The second line consists of an integer item\_id.
- The third line consists of a double new\_price.

For choice 3 (View Item Details):

- The second line consists of an integer item\_id.

For choice 4 (Display All Menu Items):

- No additional inputs are required.

For choice 5 (Exit):

- No additional inputs are required.

### ***Output Format***

For choice 1 (Add Menu Item):

- Print "Menu item added successfully" if the item was added.
- Print "Failed to add item." if the insertion failed.

For choice 2 (Update Item Price):

- Print "Item price updated successfully" if the price update was successful.
- Print "Item not found." if the specified item ID does not exist.

For choice 3 (View Item Details):

- Display the item details in the format:
- ID: [item\_id] | Name: [name] | Category: [category] | Price: [price]
- Print "Item not found." if the specified item ID does not exist.

For choice 4 (Display All Menu Items):

- Display each item on a new line in the format:
- ID | Name | Category | Price
- If no items are available, print nothing (or handle with an appropriate message if desired).

For choice 5 (Exit):

- Print "Exiting Restaurant Management System."

For invalid input:

- Print "Invalid choice. Please try again."

### **Sample Test Case**

Input: 1

11

Margherita Pizza

Main Course

12.99

4

5

Output: Menu item added successfully

ID | Name | Category | Price

11 | Margherita Pizza | Main Course | 12.99

Exiting Restaurant Management System.

### **Answer**

```
import java.sql.*;
```

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

```
class RestaurantManagementSystem {
```

```
    public static void main(String[] args) {
```

```
        try (Connection conn = DriverManager.getConnection("jdbc:mysql://localhost/ri_db", "test", "test123");
```

```
            Scanner scanner = new Scanner(System.in)) {
```

```
            boolean running = true;
```

```
            while (running) {
```

```
                int choice = scanner.nextInt();
```

```
                switch (choice) {
```

```
                    case 1:
```

```
                        addMenuItem(conn, scanner);
```

```
                        break;
```

```
                    case 2:
```

```
                        updateItemPrice(conn, scanner);
```

```
                        break;
```



```

        case 3:
            viewItemDetails(conn, scanner);
            break;
        case 4:
            displayAllMenuItems(conn);
            break;
        case 5:
            System.out.println("Exiting Restaurant Management System.");
            running = false;
            break;
        default:
            System.out.println("Invalid choice. Please try again.");
    }
}
} catch (SQLException e) {
    e.printStackTrace();
}
}
}

```

```

static java.util.List<MenuItem> menuList = new java.util.ArrayList<>();
// ===== CASE 1: ADD MENU ITEM =====
public static void addItem(Connection conn, Scanner scanner) {
    int id = scanner.nextInt();
    scanner.nextLine();
    String name = scanner.nextLine();
    String category = scanner.nextLine();
    double price = scanner.nextDouble();
    scanner.nextLine();

    menuList.add(new MenuItem(id, name, category, price));

    System.out.println("Menu item added successfully");
}

```

```

// ===== CASE 2: UPDATE PRICE =====
public static void updateItemPrice(Connection conn, Scanner scanner) {
    int id = scanner.nextInt();
    double newPrice = scanner.nextDouble();
    scanner.nextLine();

    for (MenuItem item : menuList) {

```

```

        if (item.getItemId() == id) {
            item.setPrice(newPrice);
            System.out.println("Item price updated successfully");
            return;
        }
    }
    System.out.println("Item not found.");
}

// ===== CASE 3: VIEW ITEM =====
public static void viewItemDetails(Connection conn, Scanner scanner) {
    int id = scanner.nextInt();
    scanner.nextLine();

    for (MenuItem item : menuList) {
        if (item.getItemId() == id) {
            System.out.println(
                "ID: " + item.getItemId() +
                " | Name: " + item.getName() +
                " | Category: " + item.getCategory() +
                " | Price: " + item.getPrice()
            );
            return;
        }
    }
    System.out.println("Item not found.");
}

// ===== CASE 4: DISPLAY ALL =====
public static void displayAllMenuItems(Connection conn) {

    if (menuList.isEmpty()) {
        return;
    }

    // Sort items by ID
    menuList.sort(java.util.Comparator.comparing(MenuItem::getItemId));

    // Print expected header
    System.out.println("ID | Name | Category | Price");

    for (MenuItem item : menuList) {

```

```

        System.out.println(
            item.getItemId() + " | " +
            item.getName() + " | " +
            item.getCategory() + " | " +
            String.format("%.2f", item.getPrice())
        );
    }
}

// =====
//      MENU ITEM POJO
// =====
class MenuItem {
    private int itemId;
    private String name;
    private String category;
    private double price;

    public MenuItem() {}

    public MenuItem(int itemId, String name, String category, double price) {
        this.itemId = itemId;
        this.name = name;
        this.category = category;
        this.price = price;
    }

    public int getItemId() { return itemId; }
    public String getName() { return name; }
    public String getCategory() { return category; }
    public double getPrice() { return price; }

    public void setItemId(int itemId) { this.itemId = itemId; }
    public void setName(String name) { this.name = name; }
    public void setCategory(String category) { this.category = category; }
    public void setPrice(double price) { this.price = price; }
}

//

```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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\_Week 12\_Java\_Lamba Expressions\_MCQ

Attempt : 1

Total Mark : 10

Marks Obtained : 9

#### Section 1 : MCQ

1. Which of the following interfaces is NOT a functional interface in Java?

**Answer**

Iterable

**Status : Correct**

**Marks : 1/1**

2. Which functional interface is commonly used with lambda expressions in Java?

**Answer**

Runnable

**Status : Correct**

**Marks : 1/1**

3. What is the syntax for a basic lambda expression in Java?

**Answer**

(parameters) -> expression

**Status : Correct**

**Marks : 1/1**

4. Can a lambda expression in Java have a body with multiple statements?

**Answer**

Yes, if the statements are enclosed in curly braces

**Status : Correct**

**Marks : 1/1**

5. Can a lambda expression have more than one parameter?

**Answer**

Yes, it can have multiple parameters

**Status : Correct**

**Marks : 1/1**

6. Which of the following is a valid lambda expression in Java?

**Answer**

(x) -> {return x \* 2;}

**Status : Wrong**

**Marks : 0/1**

7. What is a lambda expression in Java?

**Answer**

A way to define anonymous methods

**Status : Correct**

**Marks : 1/1**

8. Which functional interface in Java takes two arguments and returns a

result?

**Answer**

BiFunction

**Status :** Correct

**Marks :** 1/1

9. What is the return type of a lambda expression in Java?

**Answer**

The return type is inferred from the context

**Status :** Correct

**Marks :** 1/1

10. Can a lambda expression in Java have a body with multiple statements?

**Answer**

Yes, if the statements are enclosed in curly braces

**Status :** Correct

**Marks :** 1/1

# Rajalakshmi Engineering College

Name: Sachin Krishna B  
Email: 241901507@rajalakshmi.edu.in  
Roll no: 241901507  
Phone: 9025753177  
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

### 2028\_REC\_OOPS using Java\_Week 12\_Q1

Attempt : 1  
Total Mark : 10  
Marks Obtained : 10

#### Section 1 : Coding

##### 1. Problem Statement

Sabrina is working on a project that involves analyzing a set of numbers. In her exploration, she encounters scenarios where extracting even numbers and finding their sum is essential.

Create a program that calculates the sum of even numbers from a given array of integers using a lambda expression.

##### ***Input Format***

The first line of input consists of an integer N, representing the size of the array.

The second line consists of N space-separated integers, representing the elements of the array.

##### ***Output Format***

The output prints the sum of the even integers from the array.

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 3

29 37 45

Output: 0

### **Answer**

// You are using Java

import java.util.\*;

import java.util.stream.\*;

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

```
  
        int N = sc.nextInt(); // size of array  
        int[] arr = new int[N];
```

```
  
        for (int i = 0; i < N; i++) {  
            arr[i] = sc.nextInt();  
        }
```

```
  
        // Using lambda and stream to filter even numbers and sum them  
        int sum = Arrays.stream(arr)  
            .filter(x -> x % 2 == 0)  
            .sum();
```

```
  
        System.out.println(sum);  
    }  
}
```

**Status :** Correct

**Marks :** 10/10



# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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

### 2028\_REC\_OOPS using Java\_Week 12\_Q2

Attempt : 1

Total Mark : 10

Marks Obtained : 10

### Section 1 : Coding

#### 1. Problem Statement

Alex is learning about Java's functional interfaces and lambda expressions.

He wants to write a simple program that prints the square of each number in an array using a predefined functional interface.

Help Alex complete this task using the Consumer functional interface.

#### ***Input Format***

- The first line contains an integer N, the number of elements in the array.
- The second line contains N space-separated integers.

#### ***Output Format***

- Print the squares of all elements in the array, separated by a space.

Refer to the sample output for formatting specifications.

**Sample Test Case**

Input: 4

1 2 3 4

Output: 1 4 9 16

**Answer**

```
import java.util.*;
import java.util.function.*;

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

        int N = sc.nextInt();
        int[] arr = new int[N];

        for (int i = 0; i < N; i++) {
            arr[i] = sc.nextInt();
        }

        // Use IntConsumer instead of Consumer<Integer>
        IntConsumer printSquare = x -> System.out.print((x * x) + " ");

        Arrays.stream(arr).forEach(printSquare);
    }
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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

### 2028\_REC\_OOPS using Java\_Week 12\_Q3

Attempt : 1

Total Mark : 10

Marks Obtained : 10

### Section 1 : Coding

#### 1. Problem Statement

In the mystical realm of programming, there exists a magical incantation to reveal hidden words.

Elara, the skilled enchantress, wishes to summon a word using her spell and then reverse its characters to uncover its enchanted reflection.

Write a program that uses the predefined functional interface `Supplier<String>` and a lambda expression to:

Supply (generate) a string, and

Display its reversed form.

**Input Format**

No input is required from the user.

The string must be supplied internally using a Supplier<String>.

### **Output Format**

Print the reversed version of the supplied string.

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: Wizard!!

Output: !!draziW

### **Answer**

```
import java.util.function.Supplier;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String word = sc.nextLine(); // Read the string (since test cases supply
        different words)

        // Supplier that provides the input string
        Supplier<String> supplier = () -> word;

        // Get the string and reverse it
        String reversed = new StringBuilder(supplier.get()).reverse().toString();

        // Print the reversed string
        System.out.println(reversed);
    }
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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

### 2028\_REC\_OOPS using Java\_Week 12\_Q4

Attempt : 1

Total Mark : 10

Marks Obtained : 10

### Section 1 : Coding

#### 1. Problem Statement

Abi is working on a text analysis project where she needs to categorize words based on their length.

Words that have three or fewer characters are considered "Short", while

words with more than three characters are classified as "Long."

Write a Java program that takes a sentence as input, analyzes each word, and prints a list showing whether each word is "Short" or "Long."

Use the predefined functional interface `Function<String, String>` along with a lambda expression for categorization.

**Input Format**

A single line containing a sentence (words separated by spaces).

### **Output Format**

- A single line with each word categorized as "Short" or "Long", separated by spaces.

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: I love my cat

Output: Short Long Short Short

### **Answer**

```
import java.util.*;
import java.util.function.Function;

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

        // Function to categorize each word
        Function<String, String> categorize = word -> word.length() <= 3 ? "Short" :
"Long";

        // Split the sentence into words
        String[] words = sentence.split(" ");

        // Analyze and print each word's category
        for (String word : words) {
            System.out.print(categorize.apply(word) + " ");
        }
    }
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B  
Email: 241901507@rajalakshmi.edu.in  
Roll no: 241901507  
Phone: 9025753177  
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\_Week 12\_Java\_Lamba Expressions\_PAH

Attempt : 1  
Total Mark : 40  
Marks Obtained : 40

#### Section 1 : COD

##### 1. Problem Statement

Sneha is developing a feature for an e-commerce application that helps display product details after applying a seasonal discount.

She decides to use lambda expressions with the Consumer functional interface to print each product's name, original price, and discounted price neatly.

The program should:

Accept a list of product names and their prices. Apply a 15% discount on all products. Use a Consumer lambda expression to display the details in a formatted manner.

**Input Format**

The first line of input consists of an integer n, representing the number of products.

The next n lines each contain a String (product name) and a double (price) separated by a space.

### **Output Format**

For each product, print the details in the format:

Product: <name>, Original Price: <price>, Discounted Price: <discounted price>

If there are no products, print:

No products available

### **Sample Test Case**

Input: 1

Phone 60000

Output: Product: Phone, Original Price: 60000.0, Discounted Price: 51000.0

### **Answer**

```
import java.util.*;  
import java.util.function.Consumer;
```

```
class Product {  
    private String name;  
    private double price;  
  
    public Product(String name, double price) {  
        this.name = name;  
        this.price = price;  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    public double getPrice() {  
        return price;  
    }  
  
    public double getDiscountedPrice() {
```



```

        return price * 0.85; // 15% discount
    }
}

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        List<Product> products = new ArrayList<>();

        int n = scanner.nextInt();

        if (n == 0) {
            System.out.println("No products available");
            scanner.close();
            return;
        }

        for (int i = 0; i < n; i++) {
            String name = scanner.next();
            double price = scanner.nextDouble();
            products.add(new Product(name, price));
        }

        // Consumer lambda expression to display product details
        Consumer<Product> productConsumer = product -> {
            System.out.printf("Product: %s, Original Price: %.1f, Discounted Price: %.1f
%n",
                product.getName(),
                product.getPrice(),
                product.getDiscountedPrice());
        };

        // Apply the consumer to each product
        products.forEach(productConsumer);

        scanner.close();
    }
}

```

**Status :** Correct

**Marks :** 10/10

## 2. Problem Statement

Aditya is developing a reading app that recommends books to users based on a predefined list.

Each time a user opens the app, it should supply the next book title in the list, one at a time, using a lambda expression and the Supplier functional interface.

When all books have been recommended, the list should start again from the beginning.

### ***Input Format***

The first line contains an integer  $n$  — the total number of available book titles.

The next  $n$  lines each contain a book title (a string).

The next line contains an integer  $m$  — the number of times users open the app (i.e., the number of recommendations to be made).

### ***Output Format***

Print the supplied book title for each recommendation, one per line.

If  $m > n$ , repeat the list from the start.

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

Input: 3

The Alchemist

Atomic Habits

Ikigai

5

Output: The Alchemist

Atomic Habits

Ikigai

The Alchemist

Atomic Habits

### ***Answer***

```
// You are using Java
import java.util.*;
```

```

import java.util.function.Supplier;

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

        // Read the number of books
        int n = scanner.nextInt();
        scanner.nextLine(); // consume newline

        // Read all book titles
        List<String> books = new ArrayList<>();
        for (int i = 0; i < n; i++) {
            books.add(scanner.nextLine());
        }

        // Read the number of recommendations
        int m = scanner.nextInt();

        // Supplier lambda expression to provide books in cyclic order
        final int[] currentIndex = {0};

        Supplier<String> bookSupplier = () -> {
            String book = books.get(currentIndex[0]);
            currentIndex[0] = (currentIndex[0] + 1) % n; // Cycle back to start
            return book;
        };

        // Generate and print m recommendations
        for (int i = 0; i < m; i++) {
            System.out.println(bookSupplier.get());
        }

        scanner.close();
    }
}

```

**Status :** Correct

**Marks :** 10/10

### 3. Problem Statement

Emily, an analyst at a data processing firm, is tasked with cleaning up datasets to remove duplicate values from lists of integers.

Create a Java program that allows Emily to input a series of integers, with the program then utilizing a lambda expression to efficiently remove any duplicates.

### ***Input Format***

The first line of input consists of an integer N, representing the size of the array.

The second line consists of N space-separated integers, each denoting an array element.

### ***Output Format***

The output prints the array elements after removing the duplicates inside the square bracket separated by a comma and space.

Refer to the sample output for formatting specifications.

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

Input: 15

1 2 3 4 3 2 1 2 3 4 4 4 5 5 6

Output: [1, 2, 3, 4, 5, 6]

### ***Answer***

```
import java.util.*;
import java.util.stream.Collectors;

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

        int N = scanner.nextInt();
        List<Integer> numbers = new ArrayList<>();

        for (int i = 0; i < N; i++) {
            numbers.add(scanner.nextInt());
        }
    }
}
```

```

    }

    // Remove duplicates while maintaining order
    List<Integer> uniqueNumbers = numbers.stream()
        .distinct()
        .collect(Collectors.toList());

    // Print in the exact required format
    System.out.print("[");
    for (int i = 0; i < uniqueNumbers.size(); i++) {
        System.out.print(uniqueNumbers.get(i));
        if (i < uniqueNumbers.size() - 1) {
            System.out.print(", ");
        }
    }
    System.out.println("]");

    scanner.close();
}
}

```

**Status :** Correct

**Marks :** 10/10

#### 4. Problem Statement

Rishi is working as an HR analyst in a software company. He wants to filter a list of employees based on their salary using modern Java techniques. He has a list of employee names and salaries and wants to use lambda expressions to filter those who earn more than a specific threshold.

Implement a program using lambda expressions and functional interfaces to print the names of employees whose salary is greater than or equal to 50,000.

##### **Input Format**

The first line of input consists of an integer  $n$ , representing the number of employees.

The next  $n$  lines. Each line contains a String (employee name) and an int (salary).

### **Output Format**

The output prints the names of employees whose salary is greater than or equal to 50000, each on a new line.

If no employee found with salary greater than 50000, print: No employee found with salary  $\geq$  50000

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 4

Amit 45000

Sneha 50000

Ravi 60000

Priya 30000

Output: Sneha

Ravi

### **Answer**

```
import java.util.*;
```

```
import java.util.function.Predicate;
```

```
public class Main {
```

```
    public static void main(String[] args) {
```

```
        Scanner scanner = new Scanner(System.in);
```

```
        int n = scanner.nextInt();
```

```
        List<String> employeeNames = new ArrayList<>();
```

```
        List<Integer> employeeSalaries = new ArrayList<>();
```

```
        // Read employee data
```

```
        for (int i = 0; i < n; i++) {
```

```
            employeeNames.add(scanner.next());
```

```
            employeeSalaries.add(scanner.nextInt());
```

```
        }
```

```
        // Predicate lambda to filter employees with salary  $\geq$  50000
```

```
        Predicate<Integer> salaryFilter = salary -> salary  $\geq$  50000;
```

```
List<String> filteredEmployees = new ArrayList<>();

// Apply the filter using lambda
for (int i = 0; i < n; i++) {
    if (salaryFilter.test(employeeSalaries.get(i))) {
        filteredEmployees.add(employeeNames.get(i));
    }
}

// Display results
if (filteredEmployees.isEmpty()) {
    System.out.println("No employee found with salary >= 50000");
} else {
    filteredEmployees.forEach(System.out::println);
}

scanner.close();
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Sachin Krishna B

Email: 241901507@rajalakshmi.edu.in

Roll no: 241901507

Phone: 9025753177

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\_Week 12\_Java\_Lambda Expressions\_CY

Attempt : 1

Total Mark : 40

Marks Obtained : 40

#### Section 1 : Coding

##### 1. Problem Statement

Nethra is a researcher working on a project that involves analyzing experimental data. As part of her analysis, she needs to determine whether a given word is a palindrome or not.

Create a Java program that allows Nethra to input a word, and then check and display whether the entered word is a palindrome. Use lambda expressions to perform the palindrome check.

##### ***Input Format***

The first line of input consists of a word.

##### ***Output Format***

The output prints whether the given word is a palindrome or not in the following format:



"<input> is palindrome" or "<input> is not palindrome".

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: malayalam

Output: malayalam is palindrome

### **Answer**

```
import java.util.*;
import java.util.function.Predicate;

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

        String word = scanner.nextLine();

        // Predicate lambda to check if a string is palindrome
        Predicate<String> isPalindrome = str -> {
            String reversed = new StringBuilder(str).reverse().toString();
            return str.equals(reversed);
        };

        // Check using the lambda
        if (isPalindrome.test(word)) {
            System.out.println(word + " is palindrome");
        } else {
            System.out.println(word + " is not palindrome");
        }

        scanner.close();
    }
}
```

**Status :** Correct

**Marks :** 10/10

## 2. Problem Statement

### Problem Statement

Sophia, a data analyst, is studying experimental results collected from various lab sensors. Each sensor provides a list of numeric readings, and Sophia wants to calculate the average of these readings to analyze consistency.

She decides to use lambda expressions and the Function functional interface to compute the average of all the recorded values efficiently.

### Your Task

Write a Java program that:

Reads the total number of measurements. Reads all the measurement values as doubles. Uses a `Function<double[], Double>` lambda expression to calculate the average value. Displays the final average, formatted to two decimal places.

### ***Input Format***

The first line of input consists of an integer N, representing the number of measurements.

The second line contains N space-separated double values.

### ***Output Format***

Print the average of the entered values, rounded to two decimal places.

Refer to the sample output for formatting specifications.

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

Input: 6

2.2 1.2 5.4 4.6 2.9 55.7

Output: 12.00

### ***Answer***

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

```

import java.util.*;
import java.util.function.Function;

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

        int N = scanner.nextInt();
        double[] measurements = new double[N];

        for (int i = 0; i < N; i++) {
            measurements[i] = scanner.nextDouble();
        }

        // Function lambda to calculate average
        Function<double[], Double> averageCalculator = arr -> {
            double sum = 0.0;
            for (double value : arr) {
                sum += value;
            }
            return sum / arr.length;
        };

        // Calculate average using the lambda function
        double average = averageCalculator.apply(measurements);

        // Format to two decimal places
        System.out.printf("%.2f\n", average);

        scanner.close();
    }
}

```

**Status :** Correct

**Marks :** 10/10

### 3. Problem Statement

A company named TechNova is collecting feedback from its customers. Each customer gives a feedback score (an integer between 1 and 10) along with their name.

The company wants to:

Display each customer's name along with their feedback in a formatted way using a lambda expression and a Consumer functional interface. After displaying all feedbacks, calculate and display the average feedback score. You need to implement this functionality using Java lambda expressions and streams, emphasizing the Consumer interface for displaying formatted output.

### ***Input Format***

The first line of input contains an integer  $n$ , representing the number of customers.

The next  $n$  lines each contain a String (customer name) followed by an int (feedback score).

### ***Output Format***

- Each line prints a customer's name and feedback in the format:
- Customer: <name>, Feedback Score: <score>

- After all customers are displayed, print the average feedback as:
- Average Feedback: <average\_value>

(Average should be displayed up to two decimal places.)

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

Input: 3

Ravi 7

Ananya 9

Kiran 8

Output: Customer: Ravi, Feedback Score: 7

Customer: Ananya, Feedback Score: 9

Customer: Kiran, Feedback Score: 8

Average Feedback: 8.00

### Answer

// You are using Java

```
import java.util.*;
```

```
import java.util.function.Consumer;
```

```
public class Main {
```

```
    public static void main(String[] args) {
```

```
        Scanner scanner = new Scanner(System.in);
```

```
        int n = scanner.nextInt();
```

```
        List<String> customerNames = new ArrayList<>();
```

```
        List<Integer> feedbackScores = new ArrayList<>();
```

```
        // Read customer data
```

```
        for (int i = 0; i < n; i++) {
```

```
            customerNames.add(scanner.next());
```

```
            feedbackScores.add(scanner.nextInt());
```

```
        }
```

```
        // Consumer lambda to display formatted customer feedback
```

```
        Consumer<Integer> displayFeedback = index -> {
```

```
            String name = customerNames.get(index);
```

```
            int score = feedbackScores.get(index);
```

```
            System.out.println("Customer: " + name + ", Feedback Score: " + score);
```

```
        };
```

```
        // Display all feedbacks using the consumer
```

```
        for (int i = 0; i < n; i++) {
```

```
            displayFeedback.accept(i);
```

```
        }
```

```
        // Calculate average feedback score
```

```
        double average = feedbackScores.stream()
```

```
            .mapToInt(Integer::intValue)
```

```
            .average()
```

```
            .orElse(0.0);
```

```
        // Print average with two decimal places
```

```
        System.out.printf("Average Feedback: %.2f\n", average);
```

```
        scanner.close();
```

```
    }
```

}

**Status :** Correct

**Marks :** 10/10

#### 4. Problem Statement

Riya is developing a college admission system that assigns unique roll numbers to each newly admitted student.

Each roll number should follow this fixed format:

<DEPT>-<YEAR>-<4-digit-sequence>

where:

<DEPT> is the department code (in uppercase, e.g., CSE, ECE, MECH). <YEAR> is the admission year (e.g., 2025). <4-digit-sequence> starts from a given number and increases sequentially for each student. Write a Java program using a Supplier<String> lambda to generate and print the roll numbers for n students.

#### **Input Format**

First line: integer n — number of roll numbers to generate

Second line: string DEPT — department code (uppercase letters only)

Third line: integer YEAR — admission year

Fourth line: integer start — starting sequence number ( $0 \leq \text{start} \leq 9999$ )

#### **Output Format**

Print n roll numbers, one per line, in the required format

Sequence must be zero-padded to 4 digits

If sequence exceeds 9999, wrap around to 0000

#### **Sample Test Case**

Input: 5  
CSE  
2025

98

Output: CSE-2025-0098

CSE-2025-0099

CSE-2025-0100

CSE-2025-0101

CSE-2025-0102

**Answer**

```
import java.util.*;
import java.util.function.Supplier;

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

        int n = scanner.nextInt();
        String dept = scanner.next();
        int year = scanner.nextInt();
        int start = scanner.nextInt();

        // Supplier lambda to generate sequential roll numbers
        final int[] currentSequence = {start};

        Supplier<String> rollNumberSupplier = () -> {
            // Format sequence as 4-digit zero-padded number
            String sequence = String.format("%04d", currentSequence[0]);

            // Increment sequence and wrap around if exceeds 9999
            currentSequence[0] = (currentSequence[0] + 1) % 10000;

            return dept + "-" + year + "-" + sequence;
        };

        // Generate and print n roll numbers
        for (int i = 0; i < n; i++) {
            System.out.println(rollNumberSupplier.get());
        }

        scanner.close();
    }
}
```

Status : Correct

Marks : 10/10