

# Rajalakshmi Engineering College

Name: Elamvazhuthi MS

Email: 240701133@rajalakshmi.edu.in

Roll no: 240701133

Phone: 7010282287

Branch: REC

Department: CSE - Section 4

Batch: 2028

Degree: B.E - CSE

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

###### 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***

```
import java.util.*;  
import java.util.function.Function;
```

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

```
        int n = sc.nextInt();  
        double[] values = new double[n];
```

```
        for (int i = 0; i < n; i++) {  
            values[i] = sc.nextDouble();  
        }
```

```
        // Function to calculate average using lambda expression
```

```
        Function<double[], Double> calculateAverage = arr -> {
```

```
            double sum = 0;
```

```
            for (double val : arr) {
```

```

        sum += val;
    }
    return sum / arr.length;
};

double average = calculateAverage.apply(values);
System.out.printf("%.2f", average);

sc.close();
}
}

```

**Status :** Correct

**Marks :** 10/10

## 2. 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;
```

```
class Main {
```

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

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

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

```
        String dept = sc.next();
```

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

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

```
        final int[] current = { start };
```

```
        Supplier<String> rollNumberSupplier = () -> {
```

```
            String roll = String.format("%s-%d-%04d", dept, year, current[0]);
```

```
            current[0] = (current[0] + 1) % 10000;
```

```
            return roll;
```

```
        };
```

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

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

```
        }
```

```
        sc.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**

```
import java.util.*;
import java.util.function.Consumer;

class Customer {
    String name;
    int feedback;

    Customer(String name, int feedback) {
        this.name = name;
        this.feedback = feedback;
    }
}

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

        int n = sc.nextInt();
        List<Customer> customers = new ArrayList<>();

        for (int i = 0; i < n; i++) {
            String name = sc.next();
            int feedback = sc.nextInt();
            customers.add(new Customer(name, feedback));
        }

        Consumer<Customer> displayFeedback = c ->
            System.out.println("Customer: " + c.name + ", Feedback Score: " +
                c.feedback);
    }
}
```

```
customers.forEach(displayFeedback);

double avgFeedback = customers.stream()
    .mapToInt(c -> c.feedback)
    .average()
    .orElse(0.0);

System.out.printf("Average Feedback: %.2f", avgFeedback);
sc.close();
}
}
```

**Status :** Correct

**Marks : 10/10**

#### 4. 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.Scanner;
import java.util.function.Predicate;

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

        String inputString = scanner.nextLine();

        Predicate<String> isPalindrome = str -> {
            String reversed = new StringBuilder(str).reverse().toString();
            return str.equalsIgnoreCase(reversed);
        };

        if (isPalindrome.test(inputString)) {
            System.out.println(inputString + " is palindrome");
        } else {
            System.out.println(inputString + " is not palindrome");
        }

        scanner.close();
    }
}
```

**Status : Correct**

**Marks : 10/10**