

# Rajalakshmi Engineering College

Name: Anbarasu V  
Email: 241501018@rajalakshmi.edu.in  
Roll no: 241501018  
Phone: 9488440199  
Branch: REC  
Department: AI & ML - Section 3  
Batch: 2028  
Degree: B.E - AI & ML

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

Mesa, a store manager, needs a program to manage inventory items. Define a class ItemType with private attributes for name, deposit, and cost per day. Create an ArrayList in the Main class to store ItemType objects, allowing input and display.

Note: Use "%-20s%-20s%-20s" for formatting output in tabular format, display double values with 1 decimal place.

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

The first line of input consists of an integer n, representing the number of items.

For each of the n items, there are three lines:

1. The name of the item (a string)

2. The deposit amount (a double value)
3. The cost per day (a double value)

### **Output Format**

The output prints a formatted table with columns for name, deposit and cost per day.

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 3  
Laptop  
10000.0  
250.0  
Light  
1000.0  
50.0  
Fan  
1000.0  
100.0

Output: Name	Deposit	Cost Per Day
Laptop	10000.0	250.0
Light	1000.0	50.0
Fan	1000.0	100.0

### **Answer**

```
import java.util.ArrayList;  
import java.util.List;  
import java.util.Scanner;
```

```
class ItemType {  
    private String name;  
    private double deposit;  
    private double costPerDay;  
  
    public ItemType(String name, double deposit, double costPerDay) {  
        this.name = name;
```

```

        this.deposit = deposit;
        this.costPerDay = costPerDay;
    }

    @Override
    public String toString() {
        return String.format("%-20s%-20.1f%-20.1f", name, deposit, costPerDay);
    }
}

```

```

class ArrayListObjectMain {
    public static void main(String args[]) {
        List<ItemType> items = new ArrayList<>();
        Scanner sc = new Scanner(System.in);
        int n = Integer.parseInt(sc.nextLine());

        for (int i = 0; i < n; i++) {
            String name = sc.nextLine();
            Double deposit = Double.parseDouble(sc.nextLine());
            Double costPerDay = Double.parseDouble(sc.nextLine());
            items.add(new ItemType(name, deposit, costPerDay));
        }
        System.out.format("%-20s%-20s%-20s", "Name", "Deposit", "Cost Per Day");
        System.out.println();

        for (ItemType item : items) {
            System.out.println(item);
        }
    }
}

```

**Status :** Correct

**Marks : 10/10**

## 2. 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.ArrayList;
import java.util.Scanner;

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

**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***

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

```
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();
int[] span = new int[n];
Stack<Integer> stack = new Stack<>();
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++) {
    if (i > 0) System.out.print(" ");
    System.out.print(span[i]);
}
sc.close();
}
}

```

**Status :** Correct

**Marks :** 10/10

#### 4. 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 MergeSortedList {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int m = sc.nextInt();
        LinkedList<Integer> list1 = new LinkedList<>();
        for (int i = 0; i < m; i++) list1.add(sc.nextInt());
        int n = sc.nextInt();
        LinkedList<Integer> list2 = new LinkedList<>();
        for (int i = 0; i < n; i++) list2.add(sc.nextInt());
        LinkedList<Integer> merged = new LinkedList<>();
        merged.addAll(list1);
        merged.addAll(list2);
        Collections.sort(merged);
        for (int i = 0; i < merged.size(); i++) {
            if (i > 0) System.out.print(" ");
            System.out.print(merged.get(i));
        }
        sc.close();
    }
}
```

Status : Correct

Marks : 10/10