

# Rajalakshmi Engineering College

Name: RANNESH KHUMAR B R  
Email: 240701422@rajalakshmi.edu.in  
Roll no: 2116240701  
Phone: 9042350670  
Branch: REC  
Department: CSE - Section 8  
Batch: 2028  
Degree: B.E - CSE

Scan to verify results



## 2024\_28\_III\_OOPS Using Java Lab

### REC\_2028\_OOPS using Java\_Week 10\_CY

Attempt : 1  
Total Mark : 40  
Marks Obtained : 40

#### Section 1 : COD

##### 1. 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 {
    Map<String, Integer> scoreMap = new HashMap<>();

    boolean processInput(String input) {
        if (input.split(":").length != 2) {
            System.out.println("Invalid format");
            return false;
        }

        String[] parts = input.split(":");
        String playerName = parts[0].trim();
        String scoreStr = parts[1].trim();

        try {
            int score = Integer.parseInt(scoreStr);

            if (score < 1 || score > 100) {
                System.out.println("Invalid input");
                return false;
            }
        }
    }
}
```

```

        scoreMap.put(playerName, score);
        return true;
    } catch (NumberFormatException e) {
        System.out.println("Invalid input");
        return false;
    }
}

String findTopPlayer() {
    int maxScore = Integer.MIN_VALUE;
    String topPlayer = "";

    for (Map.Entry<String, Integer> entry : scoreMap.entrySet()) {
        if (entry.getValue() > maxScore) {
            maxScore = entry.getValue();
            topPlayer = entry.getKey();
        }
    }

    return topPlayer;
}

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

## 2. 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 {
    public void classifyWords(List<String> words) {
        TreeMap<Character, List<String>> map = new TreeMap<>();

        for (String word : words) {
            char initial = word.charAt(0);
            if (!map.containsKey(initial)) {
                map.put(initial, new ArrayList<>());
            }
            map.get(initial).add(word);
        }

        System.out.println("Grouped Words by Starting Letter:");
        for (Map.Entry<Character, List<String>> entry : map.entrySet()) {
            System.out.print(entry.getKey() + ": ");
            for (String word : entry.getValue()) {
                System.out.print(word + " ");
            }
            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

### 3. Problem Statement

Arjun is working on a program that checks if one set of numbers is a subset of another. If Set B is a subset of Set A, the program should print "YES" followed by the sorted elements of Set B. If Set B is not a subset of Set A, the program should print "NO" followed by the average of all elements from both sets combined, rounded to two decimal places.

Implement a class Solution with the required method to perform the subset check using TreeSet in Java.

#### **Input Format**

The first line contains an integer n - the number of elements in Set A.

The second line contains n space-separated integers - the elements of Set A.

The third line contains an integer m - the number of elements in Set B.

The fourth line contains m space-separated integers - the elements of Set B.

#### **Output Format**

If Set B is a subset of Set A, print "YES" followed by the sorted values of Set B.

Otherwise, print "NO" followed by the average of all numbers in both sets (rounded to two decimal places).

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 5

1 2 3 4 5

3

2 3 5

Output: YES 2 3 5

### **Answer**

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

```
class Solution {
```

```
    public static void checkSubset(TreeSet<Integer> setA, TreeSet<Integer> setB,  
    int totalElements, double sum) {
```

```
        if (setA.containsAll(setB)) {
```

```
            System.out.print("YES ");
```

```
            for (int num : setB) {
```

```
                System.out.print(num + " ");
```

```
            }
```

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

```
        } else {
```

```
            double average = sum / totalElements;
```

```
            System.out.printf("NO %.2f%n", average);
```

```
        }
```

```
    }
```

```
}
```

```
class Main {
```

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

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

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

```
        TreeSet<Integer> setA = new TreeSet<>();
```

```
        long sum = 0;
```

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

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

```
            setA.add(num);
```

```
            sum += num;
```

```
        }
```

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

```
        TreeSet<Integer> setB = new TreeSet<>();
```

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

```
int num = sc.nextInt();
setB.add(num);
sum += num;
}
Solution.checkSubset(setA, setB, n + m, sum);
sc.close();
}
}
```

**Status :** Correct

**Marks :** 10/10

#### 4. Problem Statement

Tony is an e-learning platform administrator, he oversees the user ratings for various online courses offered in the platform.

To enhance user experience, you should assist him in utilizing a HashMap to store course ratings given by learners. Regularly, he analyzes this data to identify the highest and lowest-rated courses, enabling targeted improvements and ensuring the quality of the educational content. This process assists in maintaining a competitive and engaging online learning environment for the users.

##### **Input Format**

The input consists of a string representing the course name followed by a double value representing the course's rating, in separate lines.

The input is terminated by entering "done".

##### **Output Format**

The first line of output prints the string "Highest Rated Course: " followed by the highest-rated course.

The second line prints the string "Lowest Rated Course: " followed by the lowest-rated courses.

Refer to the sample output for formatting specifications.



### Sample Test Case

Input: DSA

4.0

OOPS

4.2

C

3.2

done

Output: Highest Rated Course: OOPS

Lowest Rated Course: C

### Answer

```
import java.util.HashMap;
```

```
import java.util.Map;
```

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

```
class CourseAnalyzer {
```

```
    public Map<String, String>
```

```
    identifyHighestAndLowestRatedCourses(Map<String, Double> courseRatings) {
```

```
        double highestRating = Double.MIN_VALUE;
```

```
        double lowestRating = Double.MAX_VALUE;
```

```
        String highestRatedCourse = "";
```

```
        String lowestRatedCourse = "";
```

```
        for (Map.Entry<String, Double> entry : courseRatings.entrySet()) {
```

```
            String course = entry.getKey();
```

```
            double rating = entry.getValue();
```

```
            if (rating > highestRating) {
```

```
                highestRating = rating;
```

```
                highestRatedCourse = course;
```

```
            }
```

```
            if (rating < lowestRating) {
```

```
                lowestRating = rating;
```

```
                lowestRatedCourse = course;
```

```
            }
```

```
        }
```

```
        Map<String, String> result = new HashMap<>();
```

```
        result.put("highest", highestRatedCourse);
```

```
        result.put("lowest", lowestRatedCourse);
```

```
        return result;
```

```

    }
}

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Map<String, Double> courseRatings = new HashMap<>();

        while (true) {
            String courseName = scanner.nextLine();
            if (courseName.equalsIgnoreCase("done")) {
                break;
            }
            double rating = Double.parseDouble(scanner.nextLine().trim());
            courseRatings.put(courseName, rating);
        }

        CourseAnalyzer analyzer = new CourseAnalyzer();
        Map<String, String> result =
        analyzer.identifyHighestAndLowestRatedCourses(courseRatings);

        System.out.printf("Highest Rated Course: %s\n", result.get("highest"));
        System.out.printf("Lowest Rated Course: %s", result.get("lowest"));

        scanner.close();
    }
}

```

**Status :** Correct

**Marks :** 10/10