Rajalakshmi Engineering College

Name: Kaaviya Sri PS
Email: 240701222@rajalakshmi.edu.in
Roll no: 240701222
Phone: 8838174850
Branch: REC
Department: CSE - Section 6
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

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> {
    int rollNo;
    String name;
    int attendance;
```

```
    public Student(int rollNo, String name) {
        this.rollNo = rollNo;
        this.name = name;
        this.attendance = 0;
    }

    @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 + " " + attendance;
    }
}

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

        for (int i = 0; i < N; i++) {
            String line = sc.nextLine();
            String[] parts = line.split(" ");
```

```
        char command = parts[0].charAt(0);

        if (command == 'A') {
            int rollNo = Integer.parseInt(parts[1]);
            String name = parts[2];
            Student newStudent = new Student(rollNo, name);

            if (!students.contains(newStudent)) {
                students.add(newStudent);
            }
        }
        else if (command == 'M') {
            int rollNo = Integer.parseInt(parts[1]);
            for (Student s : students) {
                if (s.rollNo == rollNo) {
                    s.attendance++;
                    break;
                }
            }
        }
        else if (command == 'D') {
            for (Student s : students) {
                System.out.println(s);
            }
        }
    }
    sc.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 firstChar = word.charAt(0);

            map.putIfAbsent(firstChar, new ArrayList<>());
            map.get(firstChar).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 w : entry.getValue()) {
                System.out.print(w + " ");
            }
            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 totalCount, long totalSum) {
        if (setA.containsAll(setB)) {
            System.out.print("YES ");
            for (int num : setB) {
```

```
            System.out.print(num + " ");
        }
    } else {
        double avg = (double) totalSum / totalCount;
        System.out.printf("NO %.2f", avg);
    }
  }
}

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

    if (courseRatings.isEmpty()) {
        result.put("highest", "None");
        result.put("lowest", "None");
        return result;
    }

    String highestCourse = null;
    String lowestCourse = null;
    double highestRating = Double.NEGATIVE_INFINITY;
    double lowestRating = Double.POSITIVE_INFINITY;

    for (Map.Entry<String, Double> entry : courseRatings.entrySet()) {
        String course = entry.getKey();
        double rating = entry.getValue();

        if (rating > highestRating) {
            highestRating = rating;
            highestCourse = course;
        }
        if (rating < lowestRating) {
            lowestRating = rating;
            lowestCourse = course;
        }
    }

    result.put("highest", highestCourse);
    result.put("lowest", lowestCourse);
    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**