

# Rajalakshmi Engineering College

Name: Vetrichelvan M

Email: 241801309@rajalakshmi.edu.in

Roll no: 241801309

Phone: 8438320263

Branch: REC

Department: AI & DS - Section 4

Batch: 2028

Degree: B.E - AI & DS

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

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.*;
public class Main {

    static class Student implements Comparable<Student> {
        int id;
        String name;
        double gpa;

        Student(int id, String name, double gpa) {
            this.id = id;
            this.name = name;
            this.gpa = gpa;
        }
    }
}
```

```

@Override
public int compareTo(Student other) {
    if (this.gpa != other.gpa) {
        return Double.compare(this.gpa, other.gpa);
    }
    int nameCmp = this.name.compareTo(other.name);
    if (nameCmp != 0) return nameCmp;
    return Integer.compare(this.id, other.id);
}
}

```

```

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

    int N = Integer.parseInt(sc.nextLine());
    TreeSet<Student> set = new TreeSet<>();

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

        int id = Integer.parseInt(parts[0]);
        double gpa = Double.parseDouble(parts[parts.length - 1]);

        StringBuilder sb = new StringBuilder();
        for (int j = 1; j < parts.length - 1; j++) {
            sb.append(parts[j]);
            if (j < parts.length - 2) sb.append(" ");
        }
        String name = sb.toString();

        set.add(new Student(id, name, gpa));
    }

    for (Student s : set) {
        System.out.printf("%d %s %.2f\n", s.id, s.name, s.gpa);
    }
}

```

Status : Correct

Marks : 10/10

## 2. 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 {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
  
        int n = Integer.parseInt(sc.nextLine());  
        TreeMap<String, String> events = new TreeMap<>();  
  
        for (int i = 0; i < n; i++) {  
            String line = sc.nextLine().trim();  
            String[] parts = line.split(" ");  
  
            String time = parts[0];  
            String description = parts[1];  
  
            events.putIfAbsent(time, description);  
        }  
  
        System.out.println("Scheduled Events:");  
        for (Map.Entry<String, String> entry : events.entrySet()) {  
            System.out.println(entry.getKey() + " - " + entry.getValue());  
        }  
    }  
}
```

**Status :** Correct

**Marks :** 10/10

### 3. 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 sc = new Scanner(System.in);  
  
        int n = sc.nextInt();  
        String s = sc.next();  
  
        HashMap<Character, Integer> map = new HashMap<>();
```

```
for (char c : s.toCharArray()) {  
    map.put(c, map.getOrDefault(c, 0) + 1);  
}
```

```
for (char c : s.toCharArray()) {  
    if (map.get(c) == 1) {  
        System.out.println(c);  
        return;  
    }  
}
```

```
System.out.println("-1");  
}  
}
```

**Status :** Correct

**Marks :** 10/10