

# Rajalakshmi Engineering College

Name: DARSHAN M  
Email: 241501040@rajalakshmi.edu.in  
Roll no: 241501040  
Phone: 9360697087  
Branch: REC  
Department: AI & ML - Section 1  
Batch: 2028  
Degree: B.E - AI & ML

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.*;
```

```
class Student implements Comparable<Student> {  
    int studentId;  
    String name;  
    double gpa;
```

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

```

@Override
public int compareTo(Student other) {
    // First compare by GPA
    int gpaComparison = Double.compare(this.gpa, other.gpa);
    if (gpaComparison != 0) {
        return gpaComparison;
    }
    // If GPA is same, compare by name
    return this.name.compareTo(other.name);
}

@Override
public String toString() {
    return studentId + " " + name + " " + String.format("%.2f", gpa);
}
}

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

        TreeSet<Student> students = new TreeSet<>();

        for (int i = 0; i < n; i++) {
            int id = sc.nextInt();
            String name = sc.next();
            double gpa = sc.nextDouble();
            students.add(new Student(id, name, gpa));
        }

        for (Student student : students) {
            System.out.println(student);
        }

        sc.close();
    }
}

```

**Status :** Correct

**Marks :** 10/10

## 2. 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 str = sc.next();
```

```

HashMap<Character, Integer> map = new HashMap<>();

// Count frequency of each character
for (int i = 0; i < str.length(); i++) {
    char ch = str.charAt(i);
    map.put(ch, map.getOrDefault(ch, 0) + 1);
}

// Find first non-repeating character
for (int i = 0; i < str.length(); i++) {
    char ch = str.charAt(i);
    if (map.get(ch) == 1) {
        System.out.println(ch);
        sc.close();
        return;
    }
}

// If no non-repeating character found
System.out.println(-1);
sc.close();
}
}

```

**Status :** Correct

**Marks :** 10/10

### 3. 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 {  
    private TreeMap<String, String> events;
```

```
    public EventManager() {  
        events = new TreeMap<>();  
    }  
}
```

```

public void addEvent(String time, String description) {
    // Only add if time doesn't exist (avoid duplicates)
    if (!events.containsKey(time)) {
        events.put(time, description);
    }
}

public void printEvents() {
    System.out.println("Scheduled Events:");
    for (Map.Entry<String, String> entry : events.entrySet()) {
        System.out.println(entry.getKey() + " - " + entry.getValue());
    }
}

}

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        sc.nextLine(); // Consume newline

        EventManager manager = new EventManager();

        for (int i = 0; i < n; i++) {
            String line = sc.nextLine();
            String[] parts = line.split(" ", 2);
            String time = parts[0];
            String description = parts[1];
            manager.addEvent(time, description);
        }

        manager.printEvents();
        sc.close();
    }
}

```

**Status :** Correct

**Marks :** 10/10