

# Rajalakshmi Engineering College

Name: Nethra G  
Email: 240701358@rajalakshmi.edu.in  
Roll no: 240701358  
Phone: 9042026557  
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

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

## 2. Problem Statement

Aryan is developing a voting system for a college election. Each vote is recorded as an entry in an array, where every student's vote is represented by a candidate's ID. Since it's a majority-rule election, the winner is the candidate who receives more than  $n/2$  votes, where  $n$  is the total number of votes cast.

To quickly determine the winner, Aryan decides to use a HashMap to count the occurrences of each vote and identify the candidate who has received more than half of the total votes.

### Example

#### Input

7

2 2 1 2 2 2 3

#### Output

2

#### Explanation

The votes are: 2, 2, 1, 2, 2, 2, 3

Count of each candidate:

2 appears 5 times 1 appears once 3 appears once

The majority element is the one that appears more than  $N/2$  times. Since  $7/2 = 3.5$ , a number must appear at least 4 times to be the majority.

The number 2 appears 5 times, which is greater than 3.5, so the output is 2.

### Input Format

The first line contains an integer  $N$  representing the number of votes cast.

The second line contains  $N$  space-separated integers representing the votes,

where each integer corresponds to a candidate.

### **Output Format**

The output prints an integer representing the majority element (the candidate who received more than  $N/2$  votes).

If no such candidate exists, print -1.

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 7

2 2 1 2 2 2 3

Output: 2

### **Answer**

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

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

```
class MajorityElementFinder {  
    public static int findMajorityElement(int[] arr) {  
        HashMap<Integer, Integer> countMap = new HashMap<>();  
        int n = arr.length;  
        for (int num : arr) {  
            countMap.put(num, countMap.getOrDefault(num, 0) + 1);  
        }  
        for (int key : countMap.keySet()) {  
            if (countMap.get(key) > n / 2) {  
                return key;  
            }  
        }  
        return -1;  
    }  
}
```

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

```
int[] arr = new int[N];

for (int i = 0; i < N; i++) {
    arr[i] = scanner.nextInt();
}

int result = MajorityElementFinder.findMajorityElement(arr);
System.out.println(result);

scanner.close();
}
}
```

**Status :** Correct

**Marks :** 10/10

### 3. Problem Statement

David is managing an employee database where each employee has a unique ID, name, and department. He wants to ensure that duplicate employee IDs are not added to the system. Implement a Java program that allows adding employees to the system, displaying all employees, and checking if an employee exists based on the given ID.

Implement a class `EmployeeDatabase` that contains a `HashSet` to store employee records. The `Employee` class should be a user-defined object containing employee details. The main class should handle user operations and interact with the `EmployeeDatabase` class.

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

The first line contains an integer `n` representing the number of employees to be added.

The next `n` lines follow, each containing:

1. An integer `employee_id`
2. A string `name`
3. A string `department`

The next line contains an integer `m` representing the number of queries.

The next m lines follow, each containing an employee ID to check for existence.

### **Output Format**

The output prints a list of all employees added in the format:

"ID: <employee\_id>, Name: <name>, Department: <department>"

For each query, output "Employee exists" if the ID is found, otherwise "Employee not found".

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 3

101 John IT

102 Alice HR

103 Bob Finance

2

101

104

Output: ID: 101, Name: John, Department: IT

ID: 102, Name: Alice, Department: HR

ID: 103, Name: Bob, Department: Finance

Employee exists

Employee not found

### **Answer**

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

```
class Employee {  
    int employeeId;  
    String name, department;
```

```
    public Employee(int employeeId, String name, String department) {  
        this.employeeId = employeeId;  
        this.name = name;  
        this.department = department;  
    }  
}
```

```

    public int hashCode() {
        return Objects.hash(employeeId);
    }

    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (obj == null || getClass() != obj.getClass()) return false;
        Employee e = (Employee) obj;
        return this.employeeId == e.employeeId;
    }

    public String toString() {
        return "ID: " + employeeId + ", Name: " + name + ", Department: " +
        department;
    }
}

class EmployeeDatabase {
    HashSet<Employee> employees = new HashSet<>();

    public void addEmployee(int id, String name, String department) {
        employees.add(new Employee(id, name, department));
    }

    public void displayEmployees() {
        for (Employee e : employees) {
            System.out.println(e);
        }
    }

    public boolean checkEmployee(int id) {
        return employees.contains(new Employee(id, "", ""));
    }
}

class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        EmployeeDatabase db = new EmployeeDatabase();
        int n = sc.nextInt();
        for (int i = 0; i < n; i++) {
            int id = sc.nextInt();
            String name = sc.next();

```



```

String department = sc.next();
db.addEmployee(id, name, department);
}
db.displayEmployees();
int m = sc.nextInt();
for (int i = 0; i < m; i++) {
    int id = sc.nextInt();
    if (db.checkEmployee(id))
        System.out.println("Employee exists");
    else
        System.out.println("Employee not found");
}
sc.close();
}
}

```

**Status :** Correct

**Marks :** 10/10

#### 4. 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;
    }
}
```

```

    public void markAttendance() {
        this.attendance++;
    }

    public int compareTo(Student s) {
        return Integer.compare(this.rollNo, s.rollNo);
    }

    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;
    }

    public int hashCode() {
        return Objects.hash(rollNo);
    }

    public String toString() {
        return rollNo + " " + name + " " + attendance;
    }
}

```

```

class AttendanceTracker {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        sc.nextLine();
        TreeSet<Student> students = new TreeSet<>();
        for (int i = 0; i < n; i++) {
            String[] command = sc.nextLine().split(" ");
            String operation = command[0];

            if (operation.equals("A")) {
                int rollNo = Integer.parseInt(command[1]);
                String name = command[2];
                students.add(new Student(rollNo, name));
            }
            else if (operation.equals("M")) {
                int rollNo = Integer.parseInt(command[1]);
                for (Student s : students) {

```

```
        if (s.rollNo == rollNo) {  
            s.markAttendance();  
            break;  
        }  
    }  
}  
else if (operation.equals("D")) {  
    for (Student s : students) {  
        System.out.println(s);  
    }  
}  
}  
sc.close();  
}
```

**Status :** Correct

**Marks : 10/10**