Rajalakshmi Engineering College

Name: Anbarasu V
Email: 241501018@rajalakshmi.edu.in
Roll no: 241501018
Phone: 9488440199
Branch: REC
Department: AI & ML - Section 3
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_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

```
// You are using Java
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 this.rollNo - other.rollNo;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Student s = (Student) o;
        return rollNo == s.rollNo;
    }

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

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++) {
            String cmd = sc.next();
            if (cmd.equals("A")) {
                int rollNo = sc.nextInt();
                String name = sc.next();
                students.add(new Student(rollNo, name));
            } else if (cmd.equals("M")) {
                int rollNo = sc.nextInt();
                for (Student s : students) {
```

```
                if (s.rollNo == rollNo) {
                    s.attendance++;
                    break;
                }
            }
        } else if (cmd.equals("D")) {
            for (Student s : students) {
                System.out.print(s.rollNo + " " + s.name + " " + s.attendance + " ");
            }
        }
    }
    sc.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, 3, 2

Count of each candidate:

2 appears 5 times1 appears once3 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> map = new HashMap<>();
      for (int num : arr) {
```

```
            map.put(num, map.getOrDefault(num, 0) + 1);
        }
        int n = arr.length;
        for (int key : map.keySet()) {
            if (map.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 id;
    String name;
    String department;
    Employee(int id, String name, String department) {
        this.id = id;
        this.name = name;
        this.department = department;
    }
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Employee)) return false;
        Employee e = (Employee) o;
        return id == e.id;
    }
    @Override
    public int hashCode() {
        return Objects.hash(id);
    }
}

class EmployeeDatabase {
    HashSet<Employee> employees = new HashSet<>();
    void addEmployee(int id, String name, String department) {
        employees.add(new Employee(id, name, department));
    }
    void displayEmployees() {
        for (Employee e : employees) {
            System.out.print("ID: " + e.id + ", Name: " + e.name + ", Department: " +
e.department + " ");
        }
    }
    boolean checkEmployee(int id) {
```

```
        for (Employee e : employees) {
            if (e.id == id) return true;
        }
        return false;
    }
}

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 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 {
    void classifyWords(List<String> words) {
```

```
        TreeMap<Character, List<String>> map = new TreeMap<>();
        for (String word : words) {
            char ch = word.charAt(0);
            map.putIfAbsent(ch, new ArrayList<>());
            map.get(ch).add(word);
        }
        System.out.print("Grouped Words by Starting Letter: ");
        for (char key : map.keySet()) {
            System.out.print(key + ": ");
            for (String w : map.get(key)) {
                System.out.print(w + " ");
            }
        }
    }
}

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**