Rajalakshmi Engineering College

Name: Shreenidhi T
Email: 240701503@rajalakshmi.edu.in
Roll no: 240701503
Phone: 9150942326
Branch: REC
Department: CSE - Section 10
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

Bob wants to develop a score-tracking application for a gaming tournament. Each player's score is stored in a HashMap with the player's name as the key and the score as the value.

Write a program to assist Bob that takes user input to enter player scores, calculates the maximum score from the HashMap, and prints the player with the highest score.

Input Format

The input consists of strings representing player details in the format "playerName:score".

The input is terminated by entering "done".

Output Format

The output displays a string, representing the player's name who scored the maximum.

If the value is not numeric, print "Invalid input".

If any special characters other than ':' are given, print "Invalid format".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: Alice:15
Bob:56
done

Output: Bob

Answer

```
import java.util.*;


class ScoreTracker {
 Map<String, Integer> scoreMap = new HashMap<>();

 boolean processInput(String input) {
 if (input.split(":").length != 2) {
 System.out.println("Invalid format");
 return false;
 }

 String[] parts = input.split(":");
 String playerName = parts[0].trim();
 String scoreStr = parts[1].trim();

 try {
 int score = Integer.parseInt(scoreStr);

 if (score < 1 || score > 100) {
 System.out.println("Invalid input");
 return false;
 }

 scoreMap.put(playerName, score);
 return true;
 } catch (NumberFormatException e) {
 System.out.println("Invalid input");
 return false;
 }
 }

 String findTopPlayer() {
 int maxScore = Integer.MIN_VALUE;
 String topPlayer = "";

 for (Map.Entry<String, Integer> entry : scoreMap.entrySet()) {
 if (entry.getValue() > maxScore) {
 maxScore = entry.getValue();
 topPlayer = entry.getKey();

 }
 }

 return topPlayer;
 }
}

public class Main {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    ScoreTracker tracker = new ScoreTracker();
    boolean validInput = true;

    while (true) {
      String input = scanner.nextLine();

      if (input.toLowerCase().equals("done")) {
        break;
      }

      if (!tracker.processInput(input)) {
        validInput = false;
        break;
      }
    }

    if (validInput && !tracker.scoreMap.isEmpty()) {
      System.out.println(tracker.findTopPlayer());
    }

    scanner.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.*;

// You are using Java
class WordClassifier {
    public 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.println("Grouped Words by Starting Letter:");
        for(char Key : map.keySet()){
            System.out.print(Key+": ");
            for(String w:map.get(Key)){
                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

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 Integer.compare(this.rollNo, other.rollNo);
   }
```

```
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Student)) return false;
        Student other = (Student) o;
        return this.rollNo == other.rollNo;
    }

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

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

        for (int i = 0; i < N; i++) {
            String[] parts = sc.nextLine().split(" ");
            String command = parts[0];
            if (command.equals("A")) {
                int roll = Integer.parseInt(parts[1]);
                String name = parts[2];
                Student s = new Student(roll, name);
                students.add(s);
            } else if (command.equals("M")) {
                int roll = Integer.parseInt(parts[1]);

                for (Student s: students) {
                    if (s.rollNo == roll) {
                        s.attendance++;
                        break;
                    }
                }
            } else if (command.equals("D")) {
                for (Student s: students) {
                    System.out.println(s.rollNo + " " + s.name + " " + s.attendance);
                }
            }
        }
        sc.close();
    }
}
```

Status : Correct ***Marks : 10/10***

4. 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;

// You are using Java
class MajorityElementFinder {
  public static int findMajorityElement(int[] arr){
      HashMap<Integer, Integer>
      map=new HashMap<>();
      int n = arr.length;
      for(int num:arr){
         map.put(num,map.getOrDefault(num,0)+1);
}
       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***