Rajalakshmi Engineering College

Name: varsha s
Email: 241501237@rajalakshmi.edu.in
Roll no:
Phone: 9342191041
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_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

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***

3. 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***

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;

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***