Rajalakshmi Engineering College

Name: naveenraj s
Email: 240701350@rajalakshmi.edu.in
Roll no: 240701350
Phone: 6379711376
Branch: REC
Department: CSE - Section 3
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 : 36.5

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;

import java.util.*;

class CourseAnalyzer {

   public Map<String, String>
identifyHighestAndLowestRatedCourses(Map<String, Double> ratings) {
      String highest = "";
      String lowest = "";
      double max = -1;
      double min = Double.MAX_VALUE;
```

240701350

```
        for (Map.Entry<String, Double> e : ratings.entrySet()) {
            double r = e.getValue();
            if (r > max) {
                max = r;
                highest = e.getKey();
            }
            if (r < min) {
                min = r;
                lowest = e.getKey();
            }
        }

        Map<String, String> result = new HashMap<>();
        result.put("highest", highest);
        result.put("lowest", lowest);
        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();
    }
}
```

240701350

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

2. Problem Statement

Arjun is working on a program that checks if one set of numbers is a subset of another. If Set B is a subset of Set A, the program should print "YES" followed by the sorted elements of Set B. If Set B is not a subset of Set A, the program should print "NO" followed by the average of all elements from both sets combined, rounded to two decimal places.

Implement a class Solution with the required method to perform the subset check using TreeSet in Java.

Input Format

The first line contains an integer n - the number of elements in Set A.

The second line contains n space-separated integers - the elements of Set A.

The third line contains an integer m - the number of elements in Set B.

The fourth line contains m space-separated integers - the elements of Set B.

Output Format

If Set B is a subset of Set A, print "YES" followed by the sorted values of Set B.

Otherwise, print "NO" followed by the average of all numbers in both sets (rounded to two decimal places).

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
1 2 3 4 5
3
2 3 5
Output: YES 2 3 5

Answer

```
import java.util.*;

// You are using Java
class Solution {
    public static void checkSubset(TreeSet<Integer> setA, TreeSet<Integer> setB,
                                   int totalElements, long sum) {
        if (setA.containsAll(setB)) {
            System.out.print("YES ");
            for (int num : setB) {
                System.out.print(num + " ");
            }
        } else {
            double average = (double) sum / totalElements;
            System.out.printf("NO %.2f", average);
        }
    }
}

class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        TreeSet<Integer> setA = new TreeSet<>();
        long sum = 0;
        for (int i = 0; i < n; i++) {
            int num = sc.nextInt();
            setA.add(num);
            sum += num;
        }
        int m = sc.nextInt();
        TreeSet<Integer> setB = new TreeSet<>();
        for (int i = 0; i < m; i++) {
            int num = sc.nextInt();
            setB.add(num);
            sum += num;
        }
        Solution.checkSubset(setA, setB, n + m, sum);
        sc.close();
    }
}
```

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

3. Problem Statement

The city library maintains a record of books available for lending. Each book is uniquely identified by its ISBN number, along with its title and author. The librarian wants to efficiently store and manage these records, ensuring books can be listed in the order they were added.

Your task is to implement a Library Management System using HashSet where:

The librarian adds books with ISBN, title, and author.The librarian can remove books by providing an ISBN.Finally, the librarian displays the available books in the order they were added.

Implement a class Library that will handle these operations. The main function should manage user input and interact with the Library class accordingly.

Input Format

The first line contains an integer n – the number of books to be added.

The next n lines contain three values: ISBN (integer), Title (string without spaces), and Author (string without spaces).

1. An integer employee_id
2. A string title
3. A string author name

The next line contains an integer m – the number of books to be removed.

The next m lines follow, each contains an ISBN number to remove.

Output Format

The output prints a list of books available in the library after performing all operations in the format:

"ISBN: <isbn>, Title: <title>, Author: <author>"

If no books remain, print: "No books available"

240701350

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 3
1234 JavaCompleteGuide JohnDoe
5678 PythonBasics JaneDoe
9012 DataStructures AliceSmith
1
5679

Output: ISBN: 1234, Title: JavaCompleteGuide, Author: JohnDoe
ISBN: 9012, Title: DataStructures, Author: AliceSmith
ISBN: 5678, Title: PythonBasics, Author: JaneDoe

Answer

```
import java.util.*;

// You are using Java
class Book {
    int isbn;
    String title, author;

    Book(int isbn, String title, String author) {
        this.isbn = isbn;
        this.title = title;
        this.author = author;
    }

    public boolean equals(Object o) {
        return isbn == ((Book)o).isbn;
    }

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

    public String toString() {
        return "ISBN: " + isbn + ", Title: " + title + ", Author: " + author;
    }
}
```

```
class Library {
    HashSet<Book> books = new HashSet<>();

    void addBook(int isbn, String title, String author) {
        books.add(new Book(isbn, title, author));
    }

    void removeBook(int isbn) {
        Book remove = null;
        for (Book b : books) {
            if (b.isbn == isbn) {
                remove = b;
                break;
            }
        }
        if (remove != null) books.remove(remove);
    }

    void displayBooks() {
        if (books.isEmpty()) {
            System.out.println("No books available");
        } else {
            for (Book b : books)
                System.out.println(b);
        }
    }
}
class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Library library = new Library();
        int n = sc.nextInt();
        for (int i = 0; i < n; i++) {
            int isbn = sc.nextInt();
            String title = sc.next();
            String author = sc.next();
            library.addBook(isbn, title, author);
        }
        int m = sc.nextInt();
        for (int i = 0; i < m; i++) {
            int isbn = sc.nextInt();
            library.removeBook(isbn);
```

240701350

```
        }
        library.displayBooks();
        sc.close();
    }
}
```

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

4. 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.*;
```

```
import java.util.*;

class ScoreTracker {
   public LinkedHashMap<String, Integer> scoreMap = new LinkedHashMap<>();
   private boolean invalidInput = false;
   private boolean invalidFormat = false;

   public boolean processInput(String line) {
      if (line == null) return true;
      line = line.trim();
      if (line.equals("done")) return true;

      if (line.indexOf(':') == -1 || line.indexOf(':') != line.lastIndexOf(':')) {
         invalidFormat = true;
         return false;
      }

      String[] parts = line.split(":", -1);
      if (parts.length != 2) {
         invalidFormat = true;
         return false;
      }

      String name = parts[0].trim();
      String scoreStr = parts[1].trim();

      if (name.isEmpty() || scoreStr.isEmpty()) {
         invalidFormat = true;
         return false;
      }

      if (!name.matches("[A-Za-z]{1,20}")) {
         invalidFormat = true;
         return false;
      }
```

```
        if (!scoreStr.matches("[0-9]+")) {
            if (scoreStr.matches(".*[A-Za-z].*")) invalidInput = true;
            else invalidFormat = true;
            return false;
        }

        try {
            int score = Integer.parseInt(scoreStr);
            scoreMap.put(name, score);
            return true;
        } catch (NumberFormatException e) {
            invalidInput = true;
            return false;
        }
    }

    public String findTopPlayer() {
        if (invalidFormat) return "Invalid format";
        if (invalidInput) return "Invalid input";
        if (scoreMap.isEmpty()) return "";

        int max = Integer.MIN_VALUE;
        List<String> tied = new ArrayList<>();

        for (Map.Entry<String, Integer> e : scoreMap.entrySet()) {
            int val = e.getValue();
            if (val > max) {
                max = val;
                tied.clear();
                tied.add(e.getKey());
            } else if (val == max) {
                tied.add(e.getKey());
            }
        }

        return Collections.max(tied);
    }
}

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 : Partially correct **Marks : 6.5/10**

240701350