

Rajalakshmi Engineering College

Name: Mohammed Rizwan
Email: 240701327@rajalakshmi.edu.in
Roll no: 240701327
Phone: 9944383207
Branch: REC
Department: CSE - Section 9
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

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.*;
```

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

```
class WordClassifier {
```

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

```
        TreeMap<Character, List<String>> map = new TreeMap<>();
```

```
        for (String word : words) {
```

```
            if (word == null || word.isEmpty()) {
```

```
                continue;
```

```
            }
```

```
            char start = word.charAt(0);
```

```
            map.computeIfAbsent(start, k -> new ArrayList<>()).add(word);
```

```

        System.out.println("Grouped Words by Starting Letter:");
        for (Map.Entry<Character, List<String>> entry : map.entrySet()) {
            System.out.print(entry.getKey() + ": ");
            for (String w : entry.getValue()) {
                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

2. 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.HashMap;
import java.util.Map;
import java.util.Scanner;

class ScoreTracker {
    Map<String, Integer> scoreMap = new HashMap<>();
    public boolean processInput(String input) {
        if (!input.contains(":")) {
            System.out.println("Invalid format");
            return false;
        }
        String[] parts = input.split(":", 2);
        if (parts.length != 2) {
            System.out.println("Invalid format");
            return false;
        }
    }
}
```

```

    }
    String player = parts[0];
    String scoreStr = parts[1];
    int score;
    try {
        score = Integer.parseInt(scoreStr);
    } catch (NumberFormatException e) {
        System.out.println("Invalid input");
        return false;
    }

    scoreMap.put(player, score);
    return true;
}

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

3. 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 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;
```

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

```
class MajorityElementFinder {  
    public static int findMajorityElement(int[] arr) {  
        int n = arr.length;  
        HashMap<Integer, Integer> map = new HashMap<>();  
        for (int num : arr) {  
            map.put(num, map.getDefault(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

4. 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.*;
```

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

```
import java.text.DecimalFormat;
```

```
class Solution {
```

```
    public static void checkSubset(TreeSet<Integer> setA, TreeSet<Integer> setB,  
    int totalCount, long totalSum) {
```

```
        if (setA.containsAll(setB)) {
```

```
            System.out.print("YES ");
```

```
            for (int num : setB) {
```

```
                System.out.print(num + " ");
```

```
            }
```

```
        } else {
```

```
            double avg = (double) totalSum / totalCount;
```

```

        DecimalFormat df = new DecimalFormat("0.00");
        System.out.print("NO " + df.format(avg));
    }
}

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