Rajalakshmi Engineering College

Name: UMAR FAROOK
Email: 240701573@rajalakshmi.edu.in
Roll no: 240701573
Phone: 9791730398
Branch: REC
Department: CSE - Section 5
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

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.*;

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);
            }
            System.out.println();
        } else {
            double average = (double) sum / totalElements;
            System.out.printf("NO %.2f%n", 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***

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 id;
    String name;
    String department;

    public Employee(int id, String name, String department) {
        this.id = id;
        this.name = name;
        this.department = department;
    }

    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (!(obj instanceof Employee)) return false;
        Employee other = (Employee) obj;
        return this.id == other.id;
    }

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

    public String toString() {
        return "ID: " + id + ", Name: " + name + ", Department: " + department;
    }
}

class EmployeeDatabase {
    private HashSet<Employee> employees;

    public EmployeeDatabase() {
        employees = new HashSet<>();
    }

    public void addEmployee(int id, String name, String department) {
        employees.add(new Employee(id, name, department));
    }

    public void displayEmployees() {
        ArrayList<Employee> list = new ArrayList<>(employees);
        list.sort(Comparator.comparingInt(e -> e.id));
        for (Employee e : list) {
            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

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 {
    public void classifyWords(List<String> words) {
        TreeMap<Character, List<String>> map = new TreeMap<>();
        for (String word : words) {
            char firstChar = word.charAt(0);
            if (!map.containsKey(firstChar)) {
                map.put(firstChar, new ArrayList<>());
            }
            map.get(firstChar).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***

4. 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"

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.*;

class Book {
   int isbn;
   String title;
   String author;

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

   public boolean equals(Object obj) {
      if (this == obj) return true;
      if (!(obj instanceof Book)) return false;
      Book other = (Book) obj;
      return this.isbn == other.isbn;
   }

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

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

class Library {
   private LinkedHashSet<Book> books;
```

```
    public Library() {
        books = new LinkedHashSet<>();
    }

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

    public void removeBook(int isbn) {
        books.remove(new Book(isbn, "", ""));
    }

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

```
}
```

Status : Correct *Marks : 10/10*