Rajalakshmi Engineering College

Name: SIVAGURU D
Email: 240701517@rajalakshmi.edu.in
Roll no: 240701517
Phone: 9345616842
Branch: REC
Department: CSE - Section 7
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

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***

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.*;

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**

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 initial = word.charAt(0);
            if (!map.containsKey(initial)) {
                map.put(initial, new ArrayList<>());
            }
            map.get(initial).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 word : entry.getValue()) {
                System.out.print(word + " ");
            }
            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, 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 == null || getClass() != obj.getClass()) return false;
        Book book = (Book) obj;
        return isbn == book.isbn;
    }

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

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) {
        books.removeIf(book -> book.isbn == isbn);
    }

    void displayBooks() {
        if (books.isEmpty()) {
            System.out.println("No books available");
        } else {
            for (Book book : books) {
                System.out.println("ISBN: " + book.isbn + ", Title: " + book.title + ",
Author: " + book.author);
            }
        }
    }
}

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***