Rajalakshmi Engineering College

Name: HARI PRASANNA S
Email: 241501064@rajalakshmi.edu.in
Roll no: 241501064
Phone: 9042038178
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

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;
    }

    // equals() and hashCode() to ensure uniqueness by ISBN
    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (!(obj instanceof Book))
            return false;
        Book b = (Book) obj;
        return this.isbn == b.isbn;
    }

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

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

// Library class
class Library {
    private LinkedHashSet<Book> books = new LinkedHashSet<>();
    private HashSet<Integer> whitelist = new HashSet<>(); // stores ISBNs to
ensure uniqueness

    // Add book if ISBN not already present
    public void addBook(int isbn, String title, String author) {
        if (!whitelist.contains(isbn)) {
```

```
            books.add(new Book(isbn, title, author));
            whitelist.add(isbn);
        }
    }

    // Remove book by ISBN
    public void removeBook(int isbn) {
        Book toRemove = null;
        for (Book b : books) {
            if (b.isbn == isbn) {
                toRemove = b;
                break;
            }
        }
        if (toRemove != null) {
            books.remove(toRemove);
            whitelist.remove(isbn);
        }
    }

    // Display books in order of addition
    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**

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 {
   HashMap<String, Integer> scoreMap = new HashMap<>();

   // Method to process each player input
   public boolean processInput(String input) {
      // Check for invalid special characters (only ':' allowed)
      if (!input.contains(":") || input.matches(".*[^a-zA-Z0-9: ].*")) {
         System.out.println("Invalid format");
         return false;
      }

      String[] parts = input.split(":");
      if (parts.length != 2) {
         System.out.println("Invalid format");
         return false;
      }

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

      // Validate score is numeric
      try {
         int score = Integer.parseInt(scoreStr);
         if (score < 1 || score > 100) {
            System.out.println("Invalid input");
            return false;
         }
         scoreMap.put(player, score);
      } catch (NumberFormatException e) {
         System.out.println("Invalid input");
         return false;
      }
```

```
        return true;
    }

    // Method to find player with max score
    public String findTopPlayer() {
        String topPlayer = "";
        int maxScore = -1;

        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

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) {
      Map<String, String> result = new HashMap<>();

      if (courseRatings.isEmpty()) {
         result.put("highest", "None");
         result.put("lowest", "None");
         return result;
      }

      String highestCourse = "";
      String lowestCourse = "";
      double highestRating = Double.MIN_VALUE;
      double lowestRating = Double.MAX_VALUE;

      for (Map.Entry<String, Double> entry : courseRatings.entrySet()) {
         String course = entry.getKey();
         double rating = entry.getValue();

         if (rating > highestRating) {
            highestRating = rating;
            highestCourse = course;
         }
         if (rating < lowestRating) {
            lowestRating = rating;
            lowestCourse = course;
         }
      }
```

```
        result.put("highest", highestCourse);
        result.put("lowest", lowestCourse);
        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**

4. 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;
    }

    // Overriding equals() and hashCode() to prevent duplicate IDs
    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null || getClass() != obj.getClass())
            return false;
        Employee emp = (Employee) obj;
        return id == emp.id;
    }

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

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

// EmployeeDatabase class
```

```
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) {
        for (Employee e : employees) {
            if (e.id == id)
                return true;
        }
        return false;
    }
}

// Main class

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***