Rajalakshmi Engineering College

Name: KAVYA SRIRAM
Email: 241901045@rajalakshmi.edu.in
Roll no: 241901045
Phone: 8939657782
Branch: REC
Department: CSE (CS) - Section 1
Batch: 2028
Degree: B.E - CSE (CS)

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.*;

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

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

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

    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (obj == null || getClass() != obj.getClass()) return false;
        Book b = (Book) obj;
        return isbn == b.isbn;
    }
}

class Library {
    Set<Book> books = new LinkedHashSet<>();

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

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

    void displayBooks() {
        if (books.isEmpty()) {
            System.out.print("No books available");
            return;
        }
        for (Book b : books) {
```

```java
        System.out.printf("ISBN: %d, Title: %s, Author: %s\n", b.isbn, b.title,
b.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*

2. 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) {
        String highestCourse = "";
        String lowestCourse = "";
```

```java
        double highest = -1;
        double lowest = 6;

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

            if (rating > highest) {
                highest = rating;
                highestCourse = course;
            }
            if (rating < lowest) {
                lowest = rating;
                lowestCourse = course;
            }
        }

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

3.   Problem Statement

A college professor wants to keep track of students who attend classes.
Each student has a unique roll number and their attendance count
increases every time they attend a class. The system should allow adding
a student, marking their attendance, and displaying all students with their
total attendance.

Your task is to implement a Java program using TreeSet to maintain
students in sorted order of roll numbers and track their attendance count.

Operations:

A roll_no name   Add a student with roll number and name (if not already
added).M roll_no   Mark attendance for the student with the given roll
number (increase their count by 1).D   Display all students in ascending
order of roll number along with their attendance count.

Input Format

The first line contains an integer N - the number of students.

The next N lines contain one of the following commands:

A roll_no name

M roll_no

D

- A (Add)   Adds a new student with a unique roll number and name.
- M (Mark)   Increases attendance count for the given roll number.
- D (Display)   Prints all students in ascending order of roll number.

Output Format

For D, output prints each student's roll number, name, and attendance count in ascending order of roll number.

Refer to the sample output for formatting specifications.

Sample Test Case

```
Input: 5
A 101 Alice
A 102 Bob
M 101
M 101
D
Output: 101 Alice 2
102 Bob 0
```

Answer

```java
import java.util.*;

class Student implements Comparable<Student> {
    int roll;
    String name;
    int attendance;

    Student(int roll, String name) {
        this.roll = roll;
        this.name = name;
        this.attendance = 0;
    }

    public int compareTo(Student s) {
        return Integer.compare(this.roll, s.roll);
    }

    public String toString() {
        return roll + " " + name + " " + attendance;
    }
}
```

```java
class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int n = sc.nextInt();

    TreeSet<Student> set = new TreeSet<>();
    Map<Integer, Student> map = new HashMap<>();

    for (int i = 0; i < n; i++) {
      String cmd = sc.next();

      if (cmd.equals("A")) {
        int roll = sc.nextInt();
        String name = sc.next();
        if (!map.containsKey(roll)) {
          Student s = new Student(roll, name);
          set.add(s);
          map.put(roll, s);
        }
      }
      else if (cmd.equals("M")) {
        int roll = sc.nextInt();
        if (map.containsKey(roll)) {
          map.get(roll).attendance++;
        }
      }
      else if (cmd.equals("D")) {
        for (Student s : set) {
          System.out.print(s + " ");
        }
      }
    }
  }
}
```

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

```java
import java.util.*;

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;
    }

    @Override
    public boolean equals(Object o) {
        if (o instanceof Employee) {
            return this.id == ((Employee) o).id;
        }
        return false;
    }

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

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

```java
        }
    }

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;
    }
}


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
```

```java
            System.out.println("Employee not found");
        }
        sc.close();
    }
}
```

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