Rajalakshmi Engineering College

Name: Arilli Jagadeesh A
Email: 241901502@rajalakshmi.edu.in
Roll no: 241901502
Phone: 9361250488
Branch: REC
Department: CSE (CS) - Section 2
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

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.*;

// You are using Java
```

```
class Employee
{

    int id;
    String name;
    String department;

    Employee(int id, String name, String department)

{

        this.id = id;
        this.name = name;
        this.department = department;


}

    @Override
    public int hashCode()

{

        return id;


}

    @Override
    public boolean equals(Object obj)

{


        if (this == obj) return true;
        if (!(obj instanceof Employee)) return false;
        Employee e = (Employee) obj;
        return this.id == e.id;
```

```
}
}
class EmployeeDatabase
{
    HashSet<Employee> set = new HashSet<>();
    public void addEmployee(int id, String name, String department)
{
        set.add(new Employee(id, name, department));
}
    public void displayEmployees()
{
        for (Employee e : set)
{
            System.out.println("ID: " + e.id + ", Name: " + e.name + ", Department: " +
e.department);
}
}
```

```
    public boolean checkEmployee(int id)
{

        for (Employee e : set)

{


            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
                System.out.println("Employee not found");
        }
        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;

// write here code
class CourseAnalyzer

{


   public Map<String, String>
identifyHighestAndLowestRatedCourses(Map<String, Double> courseRatings)

{



      String highestCourse = null;
      String lowestCourse = null;

      double highest = -1;
      double lowest = Double.MAX_VALUE;

      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

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.*;

// You are using 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;


}
    @Override
    public int hashCode()

{


        return 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;


}
}
```

```
class Library
{

   HashSet<Book> set = new HashSet<>();
   ArrayList<Book> order = new ArrayList<>();

   public void addBook(int isbn, String title, String author)

{

      Book b = new Book(isbn, title, author);
      if (set.add(b))

{

         order.add(b);   // maintain insertion order


}


}

   public void removeBook(int isbn)

{


      Book temp = new Book(isbn, "", "");
      if (set.remove(temp))

{


         // remove from order list also
         for (int i = 0; i < order.size(); i++)
```

```
{
            if (order.get(i).isbn == isbn)
{
                order.remove(i);
                break;
}
}
}
}
    public void displayBooks()
{
        if (order.isEmpty())
{
            System.out.println("No books available");
            return;
}
        for (Book b : order)
{
```

```
            System.out.println("ISBN: " + b.isbn + ", Title: " + b.title + ", Author: " +
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**

4. 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.*;

// write your code here
class WordClassifier
```

```
{

    public void classifyWords(List<String> words)

{


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

        for (String w : words)

{

            char c = w.charAt(0);

            map.putIfAbsent(c, new ArrayList<>());
            map.get(c).add(w);


}

        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***