

Rajalakshmi Engineering College

Name: MRIDHULA DEVI M
Email: 240701337@rajalakshmi.edu.in
Roll no:
Phone: 9840329629
Branch: REC
Department: CSE - Section 8
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

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

```
// You are using Java
```

```
// You are using Java
```

```
import java.util.*;
```

```
class Student implements Comparable<Student>{  
    int roll; String name; int attendance=0;
```

```

Student(int r,String n){roll=r;name=n;}
public int compareTo(Student s){return this.roll-s.roll;}
public String toString(){return roll+" "+name+" "+attendance+" ";}
}

public class Main{
    public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        TreeSet<Student> set=new TreeSet<>();
        for(int i=0;i<n;i++){
            String cmd=sc.next();
            if(cmd.equals("A")){
                int r=sc.nextInt(); String nm=sc.next();
                set.add(new Student(r,nm));
            }else if(cmd.equals("M")){
                int r=sc.nextInt();
                for(Student s:set) if(s.roll==r){s.attendance++; break;}
            }else if(cmd.equals("D")){
                for(Student s:set) System.out.print(s);
            }
        }
    }
}

```

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) {
```

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

```
        if (courseRatings == null || courseRatings.isEmpty()) {
```

```

        result.put("highest", "");
        result.put("lowest", "");
        return result;
    }

    String highestCourse = "";
    String lowestCourse = "";
    double highestRating = Double.NEGATIVE_INFINITY;
    double lowestRating = Double.POSITIVE_INFINITY;

    for (Map.Entry<String, Double> entry : courseRatings.entrySet()) {
        double rating = entry.getValue();
        if (rating > highestRating) {
            highestRating = rating;
            highestCourse = entry.getKey();
        }
        if (rating < lowestRating) {
            lowestRating = rating;
            lowestCourse = entry.getKey();
        }
    }

    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 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.*;
```

```
// You are using Java
```

```
// You are using Java
```

```
class WordClassifier {
    void classifyWords(List<String> words) {
        TreeMap<Character, List<String>> map = new TreeMap<>();
        for (String w : words)
            map.computeIfAbsent(w.charAt(0), k -> new ArrayList<>()).add(w);

        System.out.println("Grouped Words by Starting Letter: ");
        map.forEach((ch, list) -> {
            System.out.print(ch + ": ");
            for (String s : list) System.out.print(s + " ");
            System.out.print("\n");
        });
    }
}

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;
```

```
    Book(int isbn, String title, String author) {
```

```
        this.isbn = isbn;
```

```
        this.title = title;
```

```
        this.author = author;
```

```
    }
```

```

@Override
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;
}

```

```

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

```

```

class Library {
    private LinkedHashSet<Book> 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.print("No books available");
            return;
        }
        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