Rajalakshmi Engineering College

Name: akshaya d
Email: 241901002@rajalakshmi.edu.in
Roll no: 241901002
Phone: 9363744696
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_MCQ

Attempt : 1
Total Mark : 15
Marks Obtained : 15

Section 1 : MCQ

1. What is the time complexity of retrieving an element from a HashSet?

Answer

O(1)

Status : Correct ***Marks : 1/1***

2. What will be the output of the following code?

```
import java.util.*;
class Main {
   public static void main(String[] args) {
      HashMap<String, Integer> map = new HashMap<>();
      map.put("A", 1);
      map.put("B", 2);
```

```
        map.put("C", 3);
        System.out.println(map.containsKey("B"));
    }
}
```

Answer

true

Status : Correct ***Marks : 1/1***

3. What will be the output of the following code?

```
import java.util.*;
class Main {
    public static void main(String[] args) {
        HashMap<String, String> map = new HashMap<>();
        map.put("A", "Apple");
        map.put("B", "Banana");
        map.put("C", "Cherry");
        map.replace("B", "Blueberry");
        System.out.println(map);
    }
}
```

Answer

{A=Apple, B=Blueberry, C=Cherry}

Status : Correct ***Marks : 1/1***

4. What happens when you add duplicate elements to a HashSet?

Answer

The duplicate is ignored

Status : Correct ***Marks : 1/1***

5. Which of the following allows null keys in Java?

Answer

HashMap

Status : Correct ***Marks : 1/1***

6. Which method retrieves the lowest key in a TreeMap?

Answer

firstKey()

Status : Correct ***Marks : 1/1***

7. What will happen if you add elements in descending order in a TreeSet?

Answer

They are sorted in ascending order

Status : Correct ***Marks : 1/1***

8. Which statement is true about HashSet and TreeSet?

Answer

TreeSet provides sorted elements

Status : Correct ***Marks : 1/1***

9. Which of the following is true about HashMap?

Answer

It is not synchronized

Status : Correct ***Marks : 1/1***

10. What happens if two keys have the same hash code in a HashMap?

Answer

A linked list is used to store values with the same hash

Status : Correct ***Marks : 1/1***

11. Which method removes all elements from a Set?

Answer

clear()

Status : Correct ***Marks : 1/1***

12. Which of the following is true about TreeMap?

Answer

It maintains natural ordering

Status : Correct ***Marks : 1/1***

13. What will happen if you add a null element to a TreeSet?

Answer

An exception occurs

Status : Correct ***Marks : 1/1***

14. What will be the output of the following code?

```
import java.util.*;
class Main {
   public static void main(String[] args) {
      HashMap<String, Integer> map = new HashMap<>();
      map.put("X", 10);
      map.put("Y", 20);
      map.put("Z", 30);
      map.remove("Y");
      System.out.println(map);
   }
}
```

Answer

{X=10, Z=30}

Status : Correct ***Marks : 1/1***

15. How does HashSet check for duplicate elements?

Answer

Using equals() and hashCode()

Status : Correct ***Marks : 1/1***

Rajalakshmi Engineering College

Name: akshaya d
Email: 241901002@rajalakshmi.edu.in
Roll no: 241901002
Phone: 9363744696
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

2028_REC_OOPS using Java_Week 10_Q1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : COD

1. Problem Statement

A city traffic management system needs to track vehicles entering a toll booth. Each vehicle is uniquely identified by its registration number. The system should allow adding vehicles to a record, ensuring that no duplicate registration numbers exist. The vehicles should be stored in a HashSet, which does not guarantee any specific order.

Your task is to implement a program using a HashSet that allows adding vehicle details and displaying the records.

Input Format

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

The next N lines contain details of each vehicle in the format: "RegNumber

OwnerName VehicleType"

1. RegNumber (String) - A unique registration number (Alphanumeric).
2. OwnerName (String) - The name of the vehicle owner.
3. VehicleType (String, Car, Bike, or Truck) - The type of vehicle.

If a vehicle with the same registration number is already present, ignore the duplicate entry.

Output Format

The output prints the unique vehicle records in any order (since HashSet does not maintain order).

Output format: "RegNumber OwnerName VehicleType"

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
KA01AB1234 John Car
MH02CD5678 Alice Bike
DL03EF9012 Bob Truck
TN04GH3456 Mike Car
KA01AB1234 John Car

Output: TN04GH3456 Mike Car
KA01AB1234 John Car
MH02CD5678 Alice Bike
DL03EF9012 Bob Truck

Answer

```
import java.util.*;
class Vehicle {
   String regNumber;
   String ownerName;
   String vehicleType;
   private static HashSet<Vehicle> vehicleSet = new HashSet<>();
   public Vehicle(String regNumber, String ownerName, String vehicleType) {
      this.regNumber = regNumber;
      this.ownerName = ownerName;
```

```
        this.vehicleType = vehicleType;
    }
    public static void addVehicle(String regNumber, String ownerName, String
vehicleType) {
        vehicleSet.add(new Vehicle(regNumber, ownerName, vehicleType));
    }
    public static void displayVehicles() {
        for (Vehicle v : vehicleSet) {
            System.out.println(v);
        }
    }
    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (obj == null || getClass() != obj.getClass()) return false;
        Vehicle vehicle = (Vehicle) obj;
        return regNumber.equals(vehicle.regNumber);
    }
    public int hashCode() {
        return Objects.hash(regNumber);
    }
    public String toString() {
        return regNumber + " " + ownerName + " " + vehicleType;
    }
}class TollBoothSystem {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        sc.nextLine();
        for (int i = 0; i < n; i++) {
            String regNumber = sc.next();
            String ownerName = sc.next();
            String vehicleType = sc.next();
            Vehicle.addVehicle(regNumber, ownerName, vehicleType);
        }
        Vehicle.displayVehicles();
        sc.close();
    }
}
```

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

Rajalakshmi Engineering College

Name: akshaya d
Email: 241901002@rajalakshmi.edu.in
Roll no: 241901002
Phone: 9363744696
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

2028_REC_OOPS using Java_Week 10_Q2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : COD

1. Problem Statement

John is organizing a fruit festival, and the quantities of various fruits are stored in a HashMap where fruit names are keys and quantities are values.

Help him develop a program to find the total quantity of fruits for the festival by summing up the values in the HashMap.

Input Format

The input consists of fruit quantities in the format 'fruitName:quantity', where fruitName is the name of the fruit(a string), and quantity is a double value representing the quantity.

The input is terminated by entering "done".

Output Format

The output prints a double value, representing the sum of values in the HashMap, rounded off to two decimal places.

If the value is not numeric, print "Invalid input".

If any special characters other than ':' are entered, print "Invalid format".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: Banana:15.2
Orange:56.3
Mango:47.3
done
Output: 118.80

Answer

```
import java.util.Scanner;
import java.util.Map;
import java.util.HashMap;
class ValueProcessor {
   public static Map<String, Double> readValues(Scanner scanner) {
      Map<String, Double> valueMap = new HashMap<>();
      while (true) {
         String input = scanner.nextLine();
         if (input.toLowerCase().equals("done")) {
            break;
         }
         String[] pair = input.split(":");
         if (pair.length == 2) {
            String key = pair[0].trim();
            try {
               double value = Double.parseDouble(pair[1].trim());
               valueMap.put(key, value);
            } catch (NumberFormatException e) {
               System.out.println("Invalid input");
               return null;
            }
         } else {
                System.out.println("Invalid format");
                return null;
            }
        }
        return valueMap;
    }

    public static double calculateSum(Map<String, Double> valueMap) {
        double sum = 0;
        for (double value : valueMap.values()) {
            sum += value;
        }
        return sum;
    }
}
class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Map<String, Double> valueMap = ValueProcessor.readValues(scanner);
        if (valueMap != null) {
            double sum = ValueProcessor.calculateSum(valueMap);
            System.out.printf("%.2f\n", sum);
        }
        scanner.close();
    }
}
```

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

Rajalakshmi Engineering College

Name: akshaya d
Email: 241901002@rajalakshmi.edu.in
Roll no: 241901002
Phone: 9363744696
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

2028_REC_OOPS using Java_Week 10_Q3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : COD

1. Problem Statement

Priya is analyzing encrypted messages in a research project. She wants to analyze the frequency of each character in a given paragraph. The characters should be stored in a TreeMap so that the output is sorted in ascending order of characters automatically.

You are required to build a Java program that:

Uses a TreeMap<Character, Integer> to count how many times each character appears in the message.Ignores spaces and considers only alphabets (case-sensitive).Outputs the frequencies of characters in sorted order.

You must use a TreeMap in the class named MessageAnalyzer.

Input Format

The first line of input contains an integer n, the number of lines in the message.

The next n lines each contain a string (the encrypted message line).

Output Format

The first line of output prints: "Character Frequency:"

Then print each character and its frequency in the format: "<character>: <count>"

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 2
Hello World
Java

Output: Character Frequency:
H: 1
J: 1
W: 1
a: 2
d: 1
e: 1
l: 3
o: 2
r: 1
v: 1

Answer

```
import java.util.*;
class MessageAnalyzer {
    public void analyzeMessageFrequency(List<String> lines) {
        TreeMap<Character, Integer> frequencyMap = new TreeMap<>();

        for (String line : lines) {
            for (char ch : line.toCharArray()) {
                if (Character.isLetter(ch)) {
                    frequencyMap.put(ch, frequencyMap.getOrDefault(ch, 0) + 1);
                }
            }
        }

        System.out.println("Character Frequency:");
        for (Map.Entry<Character, Integer> entry : frequencyMap.entrySet()) {
            System.out.println(entry.getKey() + ": " + entry.getValue());
        }
    }
}
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = Integer.parseInt(sc.nextLine());
        List<String> lines = new ArrayList<>();
        for (int i = 0; i < n; i++) {
            lines.add(sc.nextLine());
        }

        MessageAnalyzer analyzer = new MessageAnalyzer();
        analyzer.analyzeMessageFrequency(lines);
    }
}
```

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

Rajalakshmi Engineering College

Name: akshaya d
Email: 241901002@rajalakshmi.edu.in
Roll no: 241901002
Phone: 9363744696
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

2028_REC_OOPS using Java_Week 10_Q4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : COD

1. Problem Statement

In a ticket reservation system, you store the available seat numbers in a TreeSet. Users input their desired seat number, and the program checks whether the chosen seat is available.

Using a TreeSet ensures quick and efficient verification of seat availability, ensuring a smooth and organized ticket booking process.

Input Format

The first line of input contains a single integer n, representing the number of available seats.

The second line contains n space-separated integers, representing the available seat numbers.

The third line contains an integer m, representing the seat number that needs to be searched.

Output Format

The output displays "[m] is present!" if the given seat is available. Otherwise, it displays "[m] is not present!"

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 4
2 4 5 6
5
Output: 5 is present!

Answer

```
import java.util.Set;
import java.util.TreeSet;
import java.util.Scanner;
class NumberChecker {
   private Set<Integer> numberSet;

   public NumberChecker(Set<Integer> numberSet) {
      this.numberSet = numberSet;
   }

   public void addNumbers(int[] numbers) {
      for (int number : numbers) {
         numberSet.add(number);
      }
   }

   public String checkNumber(int number) {
      return numberSet.contains(number) ? number + " is present!" : number + " is
not present!";
   }
}
class Main {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      int numberOfElements = scanner.nextInt();
      int[] numbers = new int[numberOfElements];

      for (int i = 0; i < numberOfElements; i++) {
         numbers[i] = scanner.nextInt();
      }

      int elementToCheck = scanner.nextInt();
      scanner.close();

      Set<Integer> numberSet = new TreeSet<>();
      NumberChecker numberChecker = new NumberChecker(numberSet);
      numberChecker.addNumbers(numbers);

      System.out.println(numberChecker.checkNumber(elementToCheck));
   }
}
```

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

Rajalakshmi Engineering College

Name: akshaya d
Email: 241901002@rajalakshmi.edu.in
Roll no: 241901002
Phone: 9363744696
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_PAH

Attempt : 1
Total Mark : 30
Marks Obtained : 30

Section 1 : Coding

1. Problem Statement

A university maintains a list of student records and wants to store them in a sorted manner based on their GPA. If two students have the same GPA, they should be further sorted by their name in lexicographical order. Implement a program that uses a TreeSet to store student records and ensures unique student IDs.

Input Format

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

The next N lines contain details of each student in the format: "StudentID Name GPA"

- StudentID (Integer) - A unique identifier.
- Name (String) - The student's name (can contain spaces).

- GPA (Double) - The Grade Point Average.

Output Format

The output prints the list of students in ascending order of GPA.

If two students have the same GPA, sort them by name.

Print details in the format: "StudentID Name GPA" in the output, GPA is rounded to two decimal places.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
101 John 8.5
102 Alice 9.1
103 Bob 8.5
104 Zoe 7.3
105 Charlie 9.1

Output: 104 Zoe 7.30
103 Bob 8.50
101 John 8.50
102 Alice 9.10
105 Charlie 9.10

Answer

```
import java.util.*;
class Student implements Comparable<Student> {
    int studentID;
    String name;
    double gpa;

    public Student(int studentID, String name, double gpa) {
        this.studentID = studentID;
        this.name = name;
        this.gpa = gpa;
    }

    public int compareTo(Student other) {
        if (this.gpa != other.gpa) {
            return Double.compare(this.gpa, other.gpa);
        }
        return this.name.compareTo(other.name);
    }

    public String toString() {
        return studentID + " " + name + " " + String.format("%.2f", gpa);
    }
}
class UniversityRecords {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        sc.nextLine();
        TreeSet<Student> studentSet = new TreeSet<>();
        for (int i = 0; i < n; i++) {
            int id = sc.nextInt();
            String name = sc.next();
            double gpa = sc.nextDouble();
            studentSet.add(new Student(id, name, gpa));
        }
        for (Student s : studentSet) {
            System.out.println(s);
        }
        sc.close();
    }
}
```

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

2. Problem Statement

Sarah is working on a spam detection system that analyzes incoming messages for unique patterns. Spammers often use repetitive character sequences, making it important to identify the first non-repeating character in a message.

Given a string, Sarah needs to determine the first character that appears only once. If all characters repeat, the system should return -1.

She decides to use a HashMap to efficiently track character frequencies and find the solution.

Input Format

The first line contains an integer N representing , the length of the string.

The second line contains a string of N lowercase English letters (a-z).

Output Format

The output prints a character representing the first non-repeating character. If none exist, print -1.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 10
abacabadac

Output: d

Answer

```
import java.util.*;
class NonRepeatingCharacterFinder {

  public char findFirstNonRepeatingCharacter(String str) {
     HashMap<Character, Integer> charCount = new HashMap<>();

     for (char ch : str.toCharArray()) {
        charCount.put(ch, charCount.getOrDefault(ch, 0) + 1);
     }

     for (char ch : str.toCharArray()) {
        if (charCount.get(ch) == 1) {
           return ch;
        }
     }

     return '\0';
  }
}
class FirstNonRepeatingCharacter {
  public static void main(String[] args) {
     Scanner sc = new Scanner(System.in);

     int N = sc.nextInt();
     String str = sc.next();

     NonRepeatingCharacterFinder finder = new NonRepeatingCharacterFinder();
     char result = finder.findFirstNonRepeatingCharacter(str);

     if (result == '\0') {
        System.out.println(-1);
     } else {
        System.out.println(result);
     }

     sc.close();
  }
}
```

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

3. Problem Statement

Riya is building a calendar event scheduler where each event is stored in chronological order using a TreeMap. The key represents the event time in 24-hour format (HH:MM), and the value is the event description.

She wants the system to:

Automatically sort events by time.Avoid duplicate time entries — if a duplicate time is entered, ignore the new entry.Print all scheduled events in order.

Implement this logic using a class named EventManager.

Input Format

The first line of the input contains an integer n, representing the number of events.

The next n lines each contain a string in the format: "HH:MM Description"

(Example: 09:00 TeamMeeting).

Output Format

The first line of the output prints "Scheduled Events:"

The next k lines print each event in the format: "HH:MM - Description"

Refer to the sample output for formatting specifications.

Sample Test Case

```
Input: 5
09:00 TeamMeeting
13:30 LunchBreak
11:00 ProjectUpdate
09:00 Standup
15:00 ClientCall
Output: Scheduled Events:
09:00 - TeamMeeting
11:00 - ProjectUpdate
13:30 - LunchBreak
15:00 - ClientCall
```

Answer

```
import java.util.*;
class EventManager {
   TreeMap<String, String> schedule;

   public EventManager() {
      schedule = new TreeMap<>();
   }

   public void addEvent(String time, String description) {
      if (!schedule.containsKey(time)) {
         schedule.put(time, description);
      }
   }
```

```
    public void printSchedule() {
        System.out.println("Scheduled Events:");
        for (Map.Entry<String, String> entry : schedule.entrySet()) {
            System.out.println(entry.getKey() + " - " + entry.getValue());
        }
    }
}
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = Integer.parseInt(sc.nextLine());

        EventManager manager = new EventManager();

        for (int i = 0; i < n; i++) {
            String line = sc.nextLine();
            int spaceIndex = line.indexOf(' ');
            String time = line.substring(0, spaceIndex);
            String desc = line.substring(spaceIndex + 1);
            manager.addEvent(time, desc);
        }

        manager.printSchedule();
    }
}
```

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

Rajalakshmi Engineering College

Name: akshaya d
Email: 241901002@rajalakshmi.edu.in
Roll no: 241901002
Phone: 9363744696
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

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 {
    private int isbn;
    private String title;
    private String author;

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

    public int getIsbn() {
        return isbn;
    }

    public String getTitle() {
        return title;
    }

    public String getAuthor() {
        return author;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Book book = (Book) o;
        return isbn == book.isbn;
    }

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

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

```
class Library {
   private LinkedHashSet<Book> books;

   public Library() {
      books = new LinkedHashSet<>();
   }

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

   public void removeBook(int isbn) {
      books.removeIf(book -> book.getIsbn() == isbn);
   }

   public void displayBooks() {
      if (books.isEmpty()) {
         System.out.println("No books available");
      } else {
         for (Book book : books) {
            System.out.println(book);
         }
      }
   }
}
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<>();

   public boolean processInput(String input) {
      if (!input.contains(":")) {
         System.out.println("Invalid format");
         return false;
      }

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

      String playerName = parts[0];
      String scoreStr = parts[1];

      try {
         int score = Integer.parseInt(scoreStr);
         scoreMap.put(playerName, score);
         return true;
      } catch (NumberFormatException e) {
         System.out.println("Invalid input");
         return false;
      }
   }

   public String findTopPlayer() {
      String topPlayer = "";
      int maxScore = Integer.MIN_VALUE;

      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", "No courses available");
        result.put("lowest", "No courses available");
        return result;
    }

    String highestRatedCourse = null;
    String lowestRatedCourse = null;
    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;
            highestRatedCourse = course;
        }

        if (rating < lowestRating) {
            lowestRating = rating;
            lowestRatedCourse = course;
        }
    }

    result.put("highest", highestRatedCourse);
    result.put("lowest", lowestRatedCourse);

    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

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

```
import java.util.*;

class Student implements Comparable<Student> {
   private int rollNo;
   private String name;
   private int attendanceCount;

   public Student(int rollNo, String name) {
      this.rollNo = rollNo;
      this.name = name;
```

```
        this.attendanceCount = 0;
    }

    public int getRollNo() {
        return rollNo;
    }

    public String getName() {
        return name;
    }

    public int getAttendanceCount() {
        return attendanceCount;
    }

    public void markAttendance() {
        attendanceCount++;
    }

    @Override
    public int compareTo(Student other) {
        return Integer.compare(this.rollNo, other.rollNo);
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Student student = (Student) o;
        return rollNo == student.rollNo;
    }

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

    @Override
    public String toString() {
        return rollNo + " " + name + " " + attendanceCount;
    }
}
```

```
class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        TreeSet<Student> students = new TreeSet<>();

        int n = Integer.parseInt(scanner.nextLine().trim());

        for (int i = 0; i < n; i++) {
            String[] input = scanner.nextLine().trim().split(" ");
            String command = input[0];

            if (command.equals("A")) {
                int rollNo = Integer.parseInt(input[1]);
                String name = input[2];
                boolean exists = false;
                for (Student student : students) {
                    if (student.getRollNo() == rollNo) {
                        exists = true;
                        break;
                    }
                }

                if (!exists) {
                    students.add(new Student(rollNo, name));
                }

            } else if (command.equals("M")) {
                int rollNo = Integer.parseInt(input[1]);

                // Find student and mark attendance
                for (Student student : students) {
                    if (student.getRollNo() == rollNo) {
                        student.markAttendance();
                        break;
                    }
                }

            } else if (command.equals("D")) {
                for (Student student : students) {
                    System.out.println(student);
                }
            }
        }

        scanner.close();
    }
}
```

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