Rajalakshmi Engineering College

Name: Sachin Krishna B
Email: 241901507@rajalakshmi.edu.in
Roll no: 241901507
Phone: 9025753177
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. Which method removes all elements from a Set?

Answer

clear()

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

3. How does HashSet check for duplicate elements?

Answer

Using equals() and hashCode()

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. 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***

6. 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***

7. 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***

8. Which of the following is true about TreeMap?

Answer

It maintains natural ordering

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

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

Answer

An exception occurs

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

10. Which of the following is true about HashMap?

Answer

It is not synchronized

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

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

Answer

HashMap

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

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

Answer

firstKey()

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

13. 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***

14. Which statement is true about HashSet and TreeSet?

Answer

TreeSet provides sorted elements

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

15. 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***

Rajalakshmi Engineering College

Name: Sachin Krishna B
Email: 241901507@rajalakshmi.edu.in
Roll no: 241901507
Phone: 9025753177
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 {
    private String regNumber;
    private String ownerName;
    private String vehicleType;

    public Vehicle(String regNumber, String ownerName, String vehicleType) {
        this.regNumber = regNumber;
```

```
        this.ownerName = ownerName;
        this.vehicleType = vehicleType;
    }

    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 regNumber.hashCode();
    }

    public String toString() {
        return regNumber + " " + ownerName + " " + vehicleType;
    }
}

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        HashSet<Vehicle> vehicleSet = new HashSet<>();

        int n = scanner.nextInt();
        scanner.nextLine();

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

            String regNumber = parts[0];
            String ownerName = parts[1];
            String vehicleType = parts[2];

            Vehicle vehicle = new Vehicle(regNumber, ownerName, vehicleType);
            vehicleSet.add(vehicle);
        }

        for (Vehicle vehicle : vehicleSet) {
            System.out.println(vehicle);
```

```
        }

        scanner.close();
    }
}
```

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

Rajalakshmi Engineering College

Name: Sachin Krishna B
Email: 241901507@rajalakshmi.edu.in
Roll no: 241901507
Phone: 9025753177
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.*;
import java.text.DecimalFormat;

public class Main {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      HashMap<String, Double> fruitMap = new HashMap<>();
      DecimalFormat df = new DecimalFormat("0.00");

      while (true) {
         String input = scanner.nextLine();

         if (input.equals("done")) {
            break;
         }

         // Check if input contains ':' as separator
         if (!input.contains(":")) {
            System.out.println("Invalid format");
            return;
         }
```

```
            String[] parts = input.split(":");

            // Check if we have exactly two parts
            if (parts.length != 2) {
                System.out.println("Invalid format");
                return;
            }

            String fruitName = parts[0];
            String quantityStr = parts[1];

            // Try to parse the quantity as double
            try {
                double quantity = Double.parseDouble(quantityStr);
                fruitMap.put(fruitName, quantity);
            } catch (NumberFormatException e) {
                System.out.println("Invalid input");
                return;
            }
        }

        // Calculate total quantity
        double total = 0.0;
        for (double quantity : fruitMap.values()) {
            total += quantity;
        }

        // Print total rounded to 2 decimal places
        System.out.println(df.format(total));

        scanner.close();
    }
}
```

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

Rajalakshmi Engineering College

Name: Sachin Krishna B
Email: 241901507@rajalakshmi.edu.in
Roll no: 241901507
Phone: 9025753177
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

```
// You are using Java
import java.util.*;

class MessageAnalyzer {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        TreeMap<Character, Integer> charFrequency = new TreeMap<>();

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

        for (int i = 0; i < n; i++) {
```

```
        String line = scanner.nextLine();

        for (char ch : line.toCharArray()) {
            if (ch != ' ') {
                if (charFrequency.containsKey(ch)) {
                    charFrequency.put(ch, charFrequency.get(ch) + 1);
                } else {
                    charFrequency.put(ch, 1);
                }
            }
        }

        System.out.println("Character Frequency:");
        for (Map.Entry<Character, Integer> entry : charFrequency.entrySet()) {
            System.out.println(entry.getKey() + ": " + entry.getValue());
        }

        scanner.close();
    }
}
```

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

Rajalakshmi Engineering College

Name: Sachin Krishna B
Email: 241901507@rajalakshmi.edu.in
Roll no: 241901507
Phone: 9025753177
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

```
// You are using Java
import java.util.*;

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

        int n = scanner.nextInt();

        for (int i = 0; i < n; i++) {
            int seat = scanner.nextInt();
            availableSeats.add(seat);
        }

        int m = scanner.nextInt();

        if (availableSeats.contains(m)) {
            System.out.println(m + " is present!");
        } else {
            System.out.println(m + " is not present!");
        }
```

```
        scanner.close();
    }
}
```

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

Rajalakshmi Engineering College

Name: Sachin Krishna B
Email: 241901507@rajalakshmi.edu.in
Roll no: 241901507
Phone: 9025753177
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

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.*;

public class Main {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);

      int n = scanner.nextInt();
      scanner.nextLine(); // consume newline
      String str = scanner.nextLine();

      HashMap<Character, Integer> charCount = new HashMap<>();

      // First pass: count frequency of each character
      for (int i = 0; i < n; i++) {
         char ch = str.charAt(i);
         charCount.put(ch, charCount.getOrDefault(ch, 0) + 1);
      }

      // Second pass: find first character with count 1
      char result = '-';
      boolean found = false;

      for (int i = 0; i < n; i++) {
         char ch = str.charAt(i);
```

```
            if (charCount.get(ch) == 1) {
                result = ch;
                found = true;
                break;
            }
        }

        if (found) {
            System.out.println(result);
        } else {
            System.out.println(-1);
        }

        scanner.close();
    }
}
```

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

2. 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> {
    private int studentID;
    private String name;
    private double gpa;

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

    public int getStudentID() { return studentID; }
    public String getName() { return name; }
    public double getGpa() { return gpa; }

    @Override
```

```
    public int compareTo(Student other) {
        // First compare by GPA
        if (this.gpa != other.gpa) {
            return Double.compare(this.gpa, other.gpa);
        }
        // If GPA is same, compare by name
        return this.name.compareTo(other.name);
    }

    @Override
    public String toString() {
        return String.format("%d %s %.2f", studentID, name, gpa);
    }

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

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

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

        int n = scanner.nextInt();
        scanner.nextLine(); // consume newline

        for (int i = 0; i < n; i++) {
            String line = scanner.nextLine();
            String[] parts = line.split(" ", 3);

            int studentID = Integer.parseInt(parts[0]);
            String name = parts[1];
```

```
            double gpa = Double.parseDouble(parts[2]);

            Student student = new Student(studentID, name, gpa);
            studentSet.add(student);
        }

        // Print all students in sorted order
        for (Student student : studentSet) {
            System.out.println(student);
        }

        scanner.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 {
    private TreeMap<String, String> events;

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

    public void addEvent(String time, String description) {
        // Use putIfAbsent to ignore duplicates
        events.putIfAbsent(time, description);
    }

    public void displayEvents() {
        System.out.println("Scheduled Events:");
        for (Map.Entry<String, String> entry : events.entrySet()) {
            System.out.println(entry.getKey() + " - " + entry.getValue());
        }
```

```
    }
}

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        EventManager eventManager = new EventManager();

        int n = scanner.nextInt();
        scanner.nextLine(); // consume newline

        for (int i = 0; i < n; i++) {
            String line = scanner.nextLine();
            String[] parts = line.split(" ", 2);

            String time = parts[0];
            String description = parts[1];
            eventManager.addEvent(time, description);
        }

        eventManager.displayEvents();
        scanner.close();
    }
}
```

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

Rajalakshmi Engineering College

Name: Sachin Krishna B
Email: 241901507@rajalakshmi.edu.in
Roll no: 241901507
Phone: 9025753177
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 : 2
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

```
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() {
        this.attendanceCount++;
    }

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

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

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

    @Override
```

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

class AttendanceSystem {
    private TreeSet<Student> students;

    public AttendanceSystem() {
        students = new TreeSet<>();
    }

    public void addStudent(int rollNo, String name) {
        Student newStudent = new Student(rollNo, name);
        if (!students.contains(newStudent)) {
            students.add(newStudent);
        }
    }

    public void markAttendance(int rollNo) {
        Student tempStudent = new Student(rollNo, "");
        Student foundStudent = students.floor(tempStudent);
        if (foundStudent != null && foundStudent.getRollNo() == rollNo) {
            foundStudent.markAttendance();
        }
    }

    public void displayStudents() {
        for (Student student : students) {
            System.out.println(student);
        }
    }
}

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        AttendanceSystem system = new AttendanceSystem();

        int N = scanner.nextInt();
        scanner.nextLine();
```

```
        for (int i = 0; i < N; i++) {
            String line = scanner.nextLine();
            String[] parts = line.split(" ");

            String command = parts[0];

            if (command.equals("A")) {
                int rollNo = Integer.parseInt(parts[1]);
                String name = parts[2];
                system.addStudent(rollNo, name);
            } else if (command.equals("M")) {
                int rollNo = Integer.parseInt(parts[1]);
                system.markAttendance(rollNo);
            } else if (command.equals("D")) {
                system.displayStudents();
            }
        }

        scanner.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;

    public ScoreTracker() {
        scoreMap = new HashMap<>();
    }

    public boolean processInput(String input) {
        // Check if input contains exactly one colon
        if (countOccurrences(input, ':') != 1) {
            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].trim();
        String scoreStr = parts[1].trim();

        // Validate player name (only letters and spaces)
        if (!playerName.matches("[a-zA-Z ]+")) {
            System.out.println("Invalid format");
            return false;
        }

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

        return true;
    }

    public String findTopPlayer() {
        if (scoreMap.isEmpty()) {
            return "";
        }

        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;
    }

    private int countOccurrences(String str, char ch) {
        int count = 0;
        for (char c : str.toCharArray()) {
            if (c == ch) {
                count++;
            }
        }
        return count;
    }
}

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

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 {
    private int id;
    private String name;
    private String department;

    public Employee(int id, String name, String department) {
        this.id = id;
        this.name = name;
        this.department = department;
    }

    public int getId() {
        return id;
    }

    public String getName() {
        return name;
    }

    public String getDepartment() {
        return department;
    }

    @Override
```

```
    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (obj == null || getClass() != obj.getClass()) return false;
        Employee employee = (Employee) obj;
        return id == employee.id;
    }

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

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

class EmployeeDatabase {
    private HashSet<Employee> employees;

    public EmployeeDatabase() {
        employees = new HashSet<>();
    }

    public void addEmployee(int id, String name, String department) {
        Employee employee = new Employee(id, name, department);
        employees.add(employee);
    }

    public void displayEmployees() {
        for (Employee employee : employees) {
            System.out.println(employee);
        }
    }

    public boolean checkEmployee(int id) {
        for (Employee employee : employees) {
            if (employee.getId() == 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**

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.*;


class WordClassifier {
   private TreeMap<Character, List<String>> wordMap;

   public WordClassifier() {
      wordMap = new TreeMap<>();
```

```
    }

    public void classifyWords(List<String> words) {
        for (String word : words) {
            if (word == null || word.isEmpty()) {
                continue;
            }

            char firstChar = word.charAt(0);

            // If the key doesn't exist, create a new list
            if (!wordMap.containsKey(firstChar)) {
                wordMap.put(firstChar, new ArrayList<>());
            }

            // Add the word to the list for that character
            wordMap.get(firstChar).add(word);
        }

        displayGroupedWords();
    }

    private void displayGroupedWords() {
        System.out.println("Grouped Words by Starting Letter:");

        for (Map.Entry<Character, List<String>> entry : wordMap.entrySet()) {
            char letter = entry.getKey();
            List<String> wordList = entry.getValue();

            System.out.print(letter + ": ");

            // Print all words for this letter
            for (int i = 0; i < wordList.size(); i++) {
                System.out.print(wordList.get(i));
                if (i < wordList.size() - 1) {
                    System.out.print(" ");
                }
            }
            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**