

Rajalakshmi Engineering College

Name: Arya M R
Email: 241901009@rajalakshmi.edu.in
Roll no: 241901009
Phone: 7358633106
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 : 12

Section 1 : MCQ

1. Which statement is true about HashSet and TreeSet?

Answer

TreeSet provides sorted elements

Status : Correct

Marks : 1/1

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

Answer

firstKey()

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

Answer

O(1)

Status : Correct

Marks : 1/1

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

Answer

HashMap

Status : Correct

Marks : 1/1

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

Answer

They are stored in reverse order

Status : Wrong

Marks : 0/1

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

Answer

An exception occurs

Status : Correct

Marks : 1/1

8. 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

9. How does HashSet check for duplicate elements?

Answer

Using equals() and hashCode()

Status : Correct

Marks : 1/1

10. Which of the following is true about HashMap?

Answer

It does not allow duplicate values

Status : Wrong

Marks : 0/1

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

Answer

The duplicate is ignored

Status : Correct

Marks : 1/1

12. 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

13. Which method removes all elements from a Set?

Answer

clear()

Status : Correct

Marks : 1/1

14. Which of the following is true about TreeMap?

Answer

It maintains natural ordering

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, Integer> map = new HashMap<>();  
map.put("A", 1);  
map.put("B", 2);  
map.put("C", 3);  
System.out.println(map.containsKey("B"));  
}  
}
```

Answer

false

Status : Wrong

Marks : 0/1

Rajalakshmi Engineering College

Name: Arya M R
Email: 241901009@rajalakshmi.edu.in
Roll no: 241901009
Phone: 7358633106
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.HashSet;  
import java.util.Scanner;  
import java.util.Objects;
```

```
class Vehicle {  
    String regNumber;  
    String ownerName;  
    String vehicleType;
```

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

```
@Override  
public boolean equals(Object o) {  
    if (this == o) return true;  
    if (!(o instanceof Vehicle)) return false;  
    Vehicle vehicle = (Vehicle) o;  
    return regNumber.equals(vehicle.regNumber);  
}
```

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

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

```
public class Main {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        int n = Integer.parseInt(sc.nextLine());  
  
        HashSet<Vehicle> vehicleSet = new HashSet<>();  
  
        for (int i = 0; i < n; i++) {  
            String[] details = sc.nextLine().split(" ");  
            String regNumber = details[0];  
            String ownerName = details[1];  
            String vehicleType = details[2];  
  
            Vehicle vehicle = new Vehicle(regNumber, ownerName, vehicleType);  
            vehicleSet.add(vehicle);  
        }  
    }  
}
```



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

Status : Correct

Marks : 10/10

Rajalakshmi Engineering College

Name: Arya M R
Email: 241901009@rajalakshmi.edu.in
Roll no: 241901009
Phone: 7358633106
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.HashMap;  
import java.util.Scanner;
```

```
public class Main {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        HashMap<String, Double> fruitMap = new HashMap<>();  
        boolean invalidInput = false;  
        boolean invalidFormat = false;  
  
        while (true) {  
            String line = sc.nextLine();  
            if (line.equals("done")) {  
                break;  
            }  
  
            if (!line.contains(":") || line.contains("-") || line.contains(";") ||  
line.contains(",")) {  
                invalidFormat = true;  
                break;  
            }  
        }  
    }  
}
```

```

String[] parts = line.split(":");
if (parts.length != 2) {
    invalidFormat = true;
    break;
}

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

try {
    double quantity = Double.parseDouble(quantityStr);
    fruitMap.put(fruit, quantity);
} catch (NumberFormatException e) {
    invalidInput = true;
    break;
}

if (invalidFormat) {
    System.out.println("Invalid format");
} else if (invalidInput) {
    System.out.println("Invalid input");
} else {
    double total = 0.0;
    for (double qty : fruitMap.values()) {
        total += qty;
    }
    System.out.printf("%.2f\n", total);
}
}
}

```

Status : Correct

Marks : 10/10

Rajalakshmi Engineering College

Name: Arya M R
Email: 241901009@rajalakshmi.edu.in
Roll no: 241901009
Phone: 7358633106
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.Scanner;  
import java.util.TreeMap;
```

```
class MessageAnalyzer {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        int n = Integer.parseInt(sc.nextLine());  
  
        TreeMap<Character, Integer> freqMap = new TreeMap<>();  
  
        for (int i = 0; i < n; i++) {
```

```
String line = sc.nextLine();
for (char ch : line.toCharArray()) {
    if (Character.isLetter(ch)) {
        freqMap.put(ch, freqMap.getOrDefault(ch, 0) + 1);
    }
}

System.out.println("Character Frequency:");
for (char ch : freqMap.keySet()) {
    System.out.println(ch + ": " + freqMap.get(ch));
}
}
```

Status : Correct

Marks : 10/10

Rajalakshmi Engineering College

Name: Arya M R
Email: 241901009@rajalakshmi.edu.in
Roll no: 241901009
Phone: 7358633106
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 : 9

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.Scanner;
```

```
import java.util.TreeSet;
```

```
public class Main {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        int n = Integer.parseInt(sc.nextLine());  
  
        TreeSet<Integer> seatSet = new TreeSet<>();  
        String[] seats = sc.nextLine().split(" ");  
        for (int i = 0; i < n; i++) {  
            seatSet.add(Integer.parseInt(seats[i]));  
        }  
  
        int m = Integer.parseInt(sc.nextLine());  
  
        if (seatSet.contains(m)) {  
            System.out.println(m + " is present!");  
        } else {  
            System.out.println(m + " is not present!");  
        }  
    }  
}
```

Status : Partially correct

Marks : 9/10

Rajalakshmi Engineering College

Name: Arya M R
Email: 241901009@rajalakshmi.edu.in
Roll no: 241901009
Phone: 7358633106
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 : 25

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.HashMap;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = Integer.parseInt(sc.nextLine());
        String input = sc.nextLine();

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

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

        boolean found = false;
        for (char ch : input.toCharArray()) {
            if (freqMap.get(ch) == 1) {
                System.out.println(ch);
                found = true;
                break;
            }
        }
    }
}
```

```
    if (!found) {  
        System.out.println("-1");  
    }  
}  
}
```

Status : Partially correct

Marks : 5/10

2. 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.Scanner;  
import java.util.TreeMap;
```

```
class EventManager {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        int n = Integer.parseInt(sc.nextLine().trim());  
  
        TreeMap<String, String> eventMap = new TreeMap<>();  
  
        for (int i = 0; i < n; i++) {  
            String line = sc.nextLine().trim();  
            String[] parts = line.split(" ");  
            String time = parts[0];  
            String description = parts[1];  
  
            if (!eventMap.containsKey(time)) {  
                eventMap.put(time, description);  
            }  
        }  
  
        System.out.println("Scheduled Events:");  
        for (String time : eventMap.keySet()) {  
            System.out.println(time + " - " + eventMap.get(time));  
        }  
    }  
}
```

Status : Correct

Marks : 10/10

3. 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.Scanner;  
import java.util.TreeSet;
```

```
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;  
    }  
  
    @Override  
    public int compareTo(Student other) {  
        if (Double.compare(this.gpa, other.gpa) != 0) {  
            return Double.compare(this.gpa, other.gpa);  
        } else {  
            return this.name.compareTo(other.name);  
        }  
    }  
  
    @Override  
    public String toString() {  
        return studentID + " " + name + " " + String.format("%.2f", gpa);  
    }  
}  
  
public class Main {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        int n = Integer.parseInt(sc.nextLine().trim());  
  
        TreeSet<Student> studentSet = new TreeSet<>();
```



```
for (int i = 0; i < n; i++) {  
    String line = sc.nextLine().trim();  
    String[] parts = line.split(" ");  
    int studentID = Integer.parseInt(parts[0]);  
    double gpa = Double.parseDouble(parts[parts.length - 1]);  
    StringBuilder nameBuilder = new StringBuilder();  
    for (int j = 1; j < parts.length - 1; j++) {  
        nameBuilder.append(parts[j]);  
        if (j < parts.length - 2) {  
            nameBuilder.append(" ");  
        }  
    }  
    String name = nameBuilder.toString();  
    studentSet.add(new Student(studentID, name, gpa));  
}  
  
for (Student s : studentSet) {  
    System.out.println(s);  
}  
}
```

Status : Correct

Marks : 10/10

Rajalakshmi Engineering College

Name: Arya M R
Email: 241901009@rajalakshmi.edu.in
Roll no: 241901009
Phone: 7358633106
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.Scanner;  
import java.util.TreeSet;
```

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

```
int attendance;
```

```
public Student(int rollNo, String name) {  
    this.rollNo = rollNo;  
    this.name = name;  
    this.attendance = 0;  
}
```

```
@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 other = (Student) obj;  
    return this.rollNo == other.rollNo;  
}
```

```
@Override  
public int hashCode() {  
    return Integer.hashCode(rollNo);  
}  
}
```

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

```
        TreeSet<Student> students = new TreeSet<>();
```

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

```
            if (parts[0].equals("A")) {  
                int rollNo = Integer.parseInt(parts[1]);  
                String name = parts[2];  
                Student newStudent = new Student(rollNo, name);
```

```

        students.add(newStudent);
    } else if (parts[0].equals("M")) {
        int rollNo = Integer.parseInt(parts[1]);
        for (Student s : students) {
            if (s.rollNo == rollNo) {
                s.attendance++;
                break;
            }
        }
    } else if (parts[0].equals("D")) {
        for (Student s : students) {
            System.out.println(s.rollNo + " " + s.name + " " + s.attendance);
        }
    }
}
}
}
}
}

```

Status : Correct

Marks : 10/10

2. 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 {  
    public void classifyWords(List<String> words) {  
        TreeMap<Character, List<String>> groupedWords = new TreeMap<>();
```

```
        for (String word : words) {  
            char firstChar = word.charAt(0);  
            groupedWords.putIfAbsent(firstChar, new ArrayList<>());  
            groupedWords.get(firstChar).add(word);  
        }
```

```
        System.out.println("Grouped Words by Starting Letter:");  
        for (Map.Entry<Character, List<String>> entry : groupedWords.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

3. 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(":") || input.contains("-") || input.contains(";") ||
            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 name = parts[0];
        String scoreStr = parts[1];

        try {
            int score = Integer.parseInt(scoreStr);
            scoreMap.put(name, score);
        }
    }
}
```



```

    } catch (NumberFormatException e) {
        System.out.println("Invalid input");
        return false;
    }

    return true;
}

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

4. Problem Statement

David is managing an employee database where each employee has a unique ID, name, and department. He wants to ensure that duplicate employee IDs are not added to the system. Implement a Java program that allows adding employees to the system, displaying all employees, and checking if an employee exists based on the given ID.

Implement a class `EmployeeDatabase` that contains a `HashSet` to store employee records. The `Employee` class should be a user-defined object containing employee details. The main class should handle user operations and interact with the `EmployeeDatabase` class.

Input Format

The first line contains an integer `n` representing the number of employees to be added.

The next `n` lines follow, each containing:

1. An integer `employee_id`
2. A string `name`
3. A string `department`

The next line contains an integer `m` representing the number of queries.

The next `m` lines follow, each containing an employee ID to check for existence.

Output Format

The output prints a list of all employees added in the format:

"ID: <employee_id>, Name: <name>, Department: <department>"

For each query, output "Employee exists" if the ID is found, otherwise "Employee not found".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 3

101 John IT

102 Alice HR

103 Bob Finance

2

101

104

Output: ID: 101, Name: John, Department: IT

ID: 102, Name: Alice, Department: HR

ID: 103, Name: Bob, Department: Finance

Employee exists

Employee not found

Answer

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

```
class Employee {  
    int employeeId;  
    String name;  
    String department;
```

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

```
    @Override  
    public boolean equals(Object obj) {
```

```
    if (this == obj) return true;
    if (!(obj instanceof Employee)) return false;
    Employee other = (Employee) obj;
    return this.employeeId == other.employeeId;
}
```

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

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

```
class EmployeeDatabase {
    HashSet<Employee> employeeSet = new HashSet<>();

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

    public void displayEmployees() {
        for (Employee emp : employeeSet) {
            System.out.println(emp);
        }
    }

    public boolean checkEmployee(int id) {
        for (Employee emp : employeeSet) {
            if (emp.employeeId == 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