

# Rajalakshmi Engineering College

Name: Thivyasri T  
Email: 240701571@rajalakshmi.edu.in  
Roll no: 240701571  
Phone: 9043886744  
Branch: REC  
Department: CSE - Section 1  
Batch: 2028  
Degree: B.E - CSE

Scan to verify results



## 2024\_28\_III\_OOPS Using Java Lab

### REC\_2028\_OOPS using Java\_Week 10\_CY

Attempt : 1  
Total Mark : 40  
Marks Obtained : 30

#### 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 {
    int isbn;
    String title;
    String author;

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

    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Book)) return false;
        Book b = (Book) o;
        return this.isbn == b.isbn;
    }

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

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

class Library {
    private LinkedHashSet<Book> books = new LinkedHashSet<>();

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

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

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

```

}
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

Aryan is developing a voting system for a college election. Each vote is recorded as an entry in an array, where every student's vote is represented by a candidate's ID. Since it's a majority-rule election, the winner is the candidate who receives more than  $n/2$  votes, where  $n$  is the total number of votes cast.

To quickly determine the winner, Aryan decides to use a HashMap to count the occurrences of each vote and identify the candidate who has received more than half of the total votes.

Example

Input

7

2 2 1 2 2 2 3

Output

2

Explanation

The votes are: 2, 2, 1, 2, 2, 3, 2

Count of each candidate:

2 appears 5 times 1 appears once 3 appears once

The majority element is the one that appears more than  $N/2$  times. Since  $7/2 = 3.5$ , a number must appear at least 4 times to be the majority.

The number 2 appears 5 times, which is greater than 3.5, so the output is 2.

### ***Input Format***

The first line contains an integer  $N$  representing the number of votes cast.

The second line contains  $N$  space-separated integers representing the votes, where each integer corresponds to a candidate.

### ***Output Format***

The output prints an integer representing the majority element (the candidate who received more than  $N/2$  votes).

If no such candidate exists, print -1.

Refer to the sample output for formatting specifications.

### ***Sample Test Case***

Input: 7

2 2 1 2 2 2 3

Output: 2

### ***Answer***

```

import java.util.HashMap;
import java.util.Scanner;

import java.util.*;

class MajorityElementFinder {
    public static int findMajorityElement(int[] arr) {
        HashMap<Integer, Integer> map = new HashMap<>();
        for (int num : arr)
            map.put(num, map.getOrDefault(num, 0) + 1);

        int n = arr.length;
        for (Map.Entry<Integer, Integer> e : map.entrySet()) {
            if (e.getValue() > n / 2)
                return e.getKey();
        }
        return -1;
    }
}

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

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

        int result = MajorityElementFinder.findMajorityElement(arr);
        System.out.println(result);

        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) {
        String highestCourse = "", lowestCourse = "";
        double highest = Double.MIN_VALUE, lowest = Double.MAX_VALUE;

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

        Map<String, String> result = new HashMap<>();
        result.put("highest", highestCourse);
        result.put("lowest", lowestCourse);
        return result;
    }
}

```

```

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Map<String, Double> courseRatings = new HashMap<>();

        while (true) {
            String courseName = scanner.nextLine();
            if (courseName.equalsIgnoreCase("done")) {
                break;
            }
            double rating = Double.parseDouble(scanner.nextLine().trim());
            courseRatings.put(courseName, rating);
        }
    }
}

```

```

CourseAnalyzer analyzer = new CourseAnalyzer();
Map<String, String> result =
    analyzer.identifyHighestAndLowestRatedCourses(courseRatings);

```



```
System.out.printf("Highest Rated Course: %s\n", result.get("highest"));
System.out.printf("Lowest Rated Course: %s", result.get("lowest"));

    scanner.close();
}
}
```

**Status :** Correct

**Marks :** 10/10

#### 4. 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.*;
```

```
public class Main {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        HashMap<String, Integer> map = new HashMap<>();  
        boolean invalidFormat = false;  
        boolean invalidInput = false;  
  
        while (true) {  
            String input = sc.nextLine();  
  
            if (input.equals("done")) break;  
  
            if (!input.contains(":") || input.indexOf(":") != input.lastIndexOf(":")) {  
                invalidFormat = true;  
                break;  
            }  
  
            for (char ch : input.toCharArray()) {  
                if (!Character.isLetterOrDigit(ch) && ch != ':') {  
                    invalidFormat = true;  
                    break;  
                }  
            }  
  
            if (invalidFormat) break;  
  
            String[] parts = input.split(":");  
            String name = parts[0];  
            String scoreStr = parts[1];  
  
            if (!scoreStr.matches("\\d+")) {  
                invalidInput = true;  
                break;  
            }  
        }  
    }  
}
```

```
        int score = Integer.parseInt(scoreStr);
        map.put(name, score);
    }
```

```
    if (invalidFormat) {
        System.out.println("Invalid format");
        return;
    }
```

```
    if (invalidInput) {
        System.out.println("Invalid input");
        return;
    }
```

```
    String maxPlayer = "";
    int maxScore = -1;
```

```
    for (String player : map.keySet()) {
        int score = map.get(player);
        if (score > maxScore) {
            maxScore = score;
            maxPlayer = player;
        }
    }
```

```
    if (!maxPlayer.equals("")) {
        System.out.println(maxPlayer);
    }
}
```

```
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 : Wrong**

**Marks : 0/10**

# Rajalakshmi Engineering College

Name: Thivyasri T  
Email: 240701571@rajalakshmi.edu.in  
Roll no: 240701571  
Phone: 9043886744  
Branch: REC  
Department: CSE - Section 1  
Batch: 2028  
Degree: B.E - CSE

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

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 {  
    public void scheduleEvents(int n, Scanner sc) {  
        TreeMap<String, String> events = new TreeMap<>();  
        for (int i = 0; i < n; i++) {  
            String[] parts = sc.nextLine().split(" ");  
            String time = parts[0];  
            String desc = parts[1];  
            events.putIfAbsent(time, desc);  
        }  
        System.out.println("Scheduled Events:");  
        for (Map.Entry<String, String> e : events.entrySet())
```

```

        System.out.println(e.getKey() + " - " + e.getValue());
    }
}

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

```

**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.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        String s = sc.next();
        HashMap<Character, Integer> map = new HashMap<>();
        for (char c : s.toCharArray())
            map.put(c, map.getOrDefault(c, 0) + 1);
        char result = '-';
        for (char c : s.toCharArray()) {
            if (map.get(c) == 1) {
                result = c;
                break;
            }
        }
        if (result == '-') System.out.println(-1);
        else System.out.println(result);
    }
}
```

**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.*;
class Student implements Comparable<Student> {
    int id;
    String name;
```

```

double gpa;

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

public int compareTo(Student s) {
    if (this.gpa != s.gpa)
        return Double.compare(this.gpa, s.gpa);
    int nameCompare = this.name.compareTo(s.name);
    if (nameCompare != 0)
        return nameCompare;
    return Integer.compare(this.id, s.id);
}

public String toString() {
    return id + " " + 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());
        TreeSet<Student> students = new TreeSet<>();
        for (int i = 0; i < n; i++) {
            String[] parts = sc.nextLine().split(" ");
            int id = Integer.parseInt(parts[0]);
            double gpa = Double.parseDouble(parts[parts.length - 1]);
            String name = String.join(" ", Arrays.copyOfRange(parts, 1, parts.length -
1));
            students.add(new Student(id, name, gpa));
        }
        for (Student s : students) System.out.println(s);
    }
}

```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Thivyasri T  
Email: 240701571@rajalakshmi.edu.in  
Roll no: 240701571  
Phone: 9043886744  
Branch: REC  
Department: CSE - Section 1  
Batch: 2028  
Degree: B.E - CSE

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

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Thivyasri T  
Email: 240701571@rajalakshmi.edu.in  
Roll no: 240701571  
Phone: 9043886744  
Branch: REC  
Department: CSE - Section 1  
Batch: 2028  
Degree: B.E - CSE

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 static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        int n = Integer.parseInt(sc.nextLine());  
        TreeMap<Character, Integer> map = new TreeMap<>();  
        for (int i = 0; i < n; i++) {  
            String line = sc.nextLine();  
            for (char c : line.toCharArray()) {  
                if (Character.isLetter(c))
```

```
        map.put(c, map.getOrDefault(c, 0) + 1);
    }
}
System.out.println("Character Frequency:");
for (Map.Entry<Character, Integer> e : map.entrySet())
    System.out.println(e.getKey() + ": " + e.getValue());
}
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Thivyasri T  
Email: 240701571@rajalakshmi.edu.in  
Roll no: 240701571  
Phone: 9043886744  
Branch: REC  
Department: CSE - Section 1  
Batch: 2028  
Degree: B.E - CSE

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.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        HashMap<String, Double> fruits = new HashMap<>();
        double sum = 0;
        boolean invalidInput = false, invalidFormat = false;
        while (true) {
            String input = sc.nextLine();
            if (input.equals("done")) break;
            if (!input.contains(":") || input.matches("[^a-zA-Z0-9:. ]")) {
                invalidFormat = true;
                break;
            }
            String[] parts = input.split(":");
            if (parts.length != 2) {
                invalidFormat = true;
                break;
            }
            String fruit = parts[0];
            try {
                double qty = Double.parseDouble(parts[1]);
```

```
        fruits.put(fruit, qty);
    } catch (Exception e) {
        invalidInput = true;
        break;
    }
}
if (invalidFormat) System.out.println("Invalid format");
else if (invalidInput) System.out.println("Invalid input");
else {
    for (double v : fruits.values()) sum += v;
    System.out.printf("%.2f", sum);
}
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Thivyasri T  
Email: 240701571@rajalakshmi.edu.in  
Roll no: 240701571  
Phone: 9043886744  
Branch: REC  
Department: CSE - Section 1  
Batch: 2028  
Degree: B.E - CSE

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, ownerName, vehicleType;
```

```
    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 instanceof Vehicle)) return false;
        Vehicle v = (Vehicle) obj;
        return regNumber.equals(v.regNumber);
    }

    public int hashCode() {
        return regNumber.hashCode();
    }

    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> vehicles = new HashSet<>();
        for (int i = 0; i < n; i++) {
            String[] d = sc.nextLine().split(" ");
            vehicles.add(new Vehicle(d[0], d[1], d[2]));
        }
        for (Vehicle v : vehicles) System.out.println(v);
    }
}

```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Thivyasri T  
Email: 240701571@rajalakshmi.edu.in  
Roll no: 240701571  
Phone: 9043886744  
Branch: REC  
Department: CSE - Section 1  
Batch: 2028  
Degree: B.E - CSE

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

**Answer**

They are sorted in ascending order

**Status : Correct**

**Marks : 1/1**

2. How does HashSet check for duplicate elements?

**Answer**

Using equals() and hashCode()

**Status : Correct**

**Marks : 1/1**

3. Which statement is true about HashSet and TreeSet?

**Answer**

TreeSet provides sorted elements

**Status :** Correct

**Marks :** 1/1

4. 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

5. Which method removes all elements from a Set?

**Answer**

clear()

**Status :** Correct

**Marks :** 1/1

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

**Answer**

An exception occurs

**Status :** Correct

**Marks :** 1/1

7. Which of the following is true about TreeMap?

**Answer**

It maintains natural ordering

**Status :** Correct

**Marks :** 1/1

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

**Answer**

firstKey()

**Status :** Correct

**Marks :** 1/1

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

**Answer**

The duplicate is ignored

**Status :** Correct

**Marks :** 1/1

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

**Answer**

HashMap

**Status :** Correct

**Marks :** 1/1

11. 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

12. 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

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

**Answer**

$O(1)$

**Status :** Correct

**Marks :** 1/1

14. Which of the following is true about HashMap?

**Answer**

It is not synchronized

**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**

true

**Status :** Correct

**Marks :** 1/1