

# Rajalakshmi Engineering College

Name: Nithin Daniel

Email: 241501133@rajalakshmi.edu.in

Roll no: 241501133

Phone: 8056422379

Branch: REC

Department: AI & ML - Section 2

Batch: 2028

Degree: B.E - AI & ML

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 : 14

#### **Section 1 : MCQ**

- 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

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, 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

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

Answer

HashMap

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("A", 1);
        map.put("B", 2);
        map.put("C", 3);
        System.out.println(map.containsKey("B"));
    }
}
```

**Answer**

true

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 when you add duplicate elements to a HashSet?

**Answer**

The duplicate is ignored

Status : Correct

Marks : 1/1

8. Which statement is true about HashSet and TreeSet?

**Answer**

TreeSet provides sorted elements

Status : Correct

Marks : 1/1

9. Which of the following is true about TreeMap?

**Answer**

It maintains natural ordering

**Status : Correct**

**Marks : 1/1**

10. Which method removes all elements from a Set?

**Answer**

clear()

**Status : Correct**

**Marks : 1/1**

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

**Answer**

O(1)

**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 will happen if you add a null element to a TreeSet?

**Answer**

An exception occurs

**Status : Correct**

**Marks : 1/1**

14. Which of the following is true about HashMap?

**Answer**

It does not allow duplicate values

**Status : Wrong**

**Marks : 0/1**

15. How does HashSet check for duplicate elements?

**Answer**

Using equals() and hashCode()

**Status : Correct**

**Marks : 1/1**

# Rajalakshmi Engineering College

Name: Nithin Daniel

Email: 241501133@rajalakshmi.edu.in

Roll no: 241501133

Phone: 8056422379

Branch: REC

Department: AI & ML - Section 2

Batch: 2028

Degree: B.E - AI & ML

Scan to verify results



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

### 2028\_REC\_OOPS using Java\_Week 10\_Q1

Attempt : 1

Total Mark : 10

Marks Obtained : 10

#### **Section 1 : COD**

##### **1. Problem Statement**

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

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

##### ***Input Format***

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

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

OwnerName VehicleType"

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

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

### ***Output Format***

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

Output format: "RegNumber OwnerName VehicleType"

Refer to the sample output for formatting specifications.

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

Input: 5

KA01AB1234 John Car

MH02CD5678 Alice Bike

DL03EF9012 Bob Truck

TN04GH3456 Mike Car

KA01AB1234 John Car

Output: TN04GH3456 Mike Car

KA01AB1234 John Car

MH02CD5678 Alice Bike

DL03EF9012 Bob Truck

### ***Answer***

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

```
class Vehicle {
```

```
    String regNumber;
```

```
    String ownerName;
```

```
    String vehicleType;
```

```
    Vehicle(String regNumber, String ownerName, String vehicleType) {
```

```
        this.regNumber = regNumber;
```

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

    // Override equals() and hashCode() to prevent duplicates based on
    regNumber
    @Override
    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);
    }

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

    @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 = sc.nextInt();
        HashSet<Vehicle> vehicles = new HashSet<>();

        for (int i = 0; i < n; i++) {
            String regNumber = sc.next();
            String ownerName = sc.next();
            String vehicleType = sc.next();

            Vehicle v = new Vehicle(regNumber, ownerName, vehicleType);
            vehicles.add(v); // duplicates ignored automatically
        }
    }
}
```

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

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Nithin Daniel

Email: 241501133@rajalakshmi.edu.in

Roll no: 241501133

Phone: 8056422379

Branch: REC

Department: AI & ML - Section 2

Batch: 2028

Degree: B.E - AI & ML

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 sc = new Scanner(System.in);
        HashMap<String, Double> fruits = new HashMap<>();
        DecimalFormat df = new DecimalFormat("0.00");
        double total = 0.0;
        boolean invalidInput = false;
        boolean invalidFormat = false;

        while (true) {
            String input = sc.next();

            if (input.equalsIgnoreCase("done"))
                break;

            // Check for correct format (must contain only one ':')
            if (!input.contains(":") || input.indexOf(':') != input.lastIndexOf(':')) {
                invalidFormat = true;
                break;
            }

            String[] fruitValue = input.split(":");
            String fruit = fruitValue[0];
            String valueStr = fruitValue[1];
            double value;
            try {
                value = Double.parseDouble(valueStr);
            } catch (NumberFormatException e) {
                invalidInput = true;
                break;
            }

            if (invalidInput)
                System.out.println("Invalid input");
            else if (invalidFormat)
                System.out.println("Invalid format");
            else
                fruits.put(fruit, value);
        }

        double sum = fruits.values().stream().mapToDouble(v -> v).sum();
        System.out.printf("Output: %.2f\n", sum);
    }
}
```

```
        }

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

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

        // Check for special characters other than ':'
        if (!fruitName.matches("[A-Za-z]+")) {
            invalidFormat = true;
            break;
        }

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

    if (invalidFormat) {
        System.out.println("Invalid format");
    } else if (invalidInput) {
        System.out.println("Invalid input");
    } else {
        for (double q : fruits.values()) {
            total += q;
        }
        System.out.println(df.format(total));
    }
}
```

Status : Correct

Marks : 10/10

# Rajalakshmi Engineering College

Name: Nithin Daniel

Email: 241501133@rajalakshmi.edu.in

Roll no: 241501133

Phone: 8056422379

Branch: REC

Department: AI & ML - Section 2

Batch: 2028

Degree: B.E - AI & ML

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 = sc.nextInt();
        sc.nextLine(); // consume newline

        TreeMap<Character, Integer> freqMap = new TreeMap<>();
        for (int i = 0; i < n; i++) {
```

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

for (char ch : line.toCharArray()) {
    // consider only alphabets, ignore spaces
    if (Character.isLetter(ch)) {
        freqMap.put(ch, freqMap.getOrDefault(ch, 0) + 1);
    }
}

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

Status : Correct

Marks : 10/10

# Rajalakshmi Engineering College

Name: Nithin Daniel

Email: 241501133@rajalakshmi.edu.in

Roll no: 241501133

Phone: 8056422379

Branch: REC

Department: AI & ML - Section 2

Batch: 2028

Degree: B.E - AI & ML

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(); // number of available seats  
        TreeSet<Integer> seats = new TreeSet<>();  
  
        // Read available seat numbers  
        for (int i = 0; i < n; i++) {  
            seats.add(sc.nextInt());  
        }  
  
        int m = sc.nextInt(); // seat number to search  
  
        // Check availability  
        if (seats.contains(m)) {  
            System.out.println(m + " is present!");  
        } else {  
            System.out.println(m + "is not present!");  
        }  
    }  
}
```

}

**Status : Correct**

**Marks : 10/10**