

# Rajalakshmi Engineering College

Name: Manisha M  
Email: 240701304@rajalakshmi.edu.in  
Roll no: 240701304  
Phone: 9363972879  
Branch: REC  
Department: CSE - Section 9  
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. How does HashSet check for duplicate elements?

**Answer**

Using equals() and hashCode()

**Status : Correct**

**Marks : 1/1**

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

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

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

**Answer**

The duplicate is ignored

**Status :** Correct

**Marks :** 1/1

4. Which of the following is true about TreeMap?

**Answer**

It maintains natural ordering

**Status :** Correct

**Marks :** 1/1

5. Which statement is true about HashSet and TreeSet?

**Answer**

TreeSet provides sorted elements

**Status :** Correct

**Marks :** 1/1

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

**Answer**

HashMap

**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 method removes all elements from a Set?

**Answer**

clear()

**Status :** Correct

**Marks :** 1/1

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

**Answer**

O(1)

**Status :** Correct

**Marks :** 1/1

10. 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

11. 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

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

**Answer**

An exception occurs

**Status :** Correct

**Marks :** 1/1

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

**Answer**

firstKey()

**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

# Rajalakshmi Engineering College

Name: Manisha M  
Email: 240701304@rajalakshmi.edu.in  
Roll no: 240701304  
Phone: 9363972879  
Branch: REC  
Department: CSE - Section 9  
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 int hashCode() {
        return regNumber.hashCode();
    }

    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (obj == null || getClass() != obj.getClass()) return false;
        Vehicle v = (Vehicle) obj;
        return regNumber.equals(v.regNumber);
    }

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

```

**Status :** Correct

**Marks :** 10/10



# Rajalakshmi Engineering College

Name: Manisha M  
Email: 240701304@rajalakshmi.edu.in  
Roll no: 240701304  
Phone: 9363972879  
Branch: REC  
Department: CSE - Section 9  
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**

```
// You are using Java
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 total = 0.0;
        boolean invalidInput = false, invalidFormat = false;

        while (true) {
            String input = sc.nextLine().trim();
            if (input.equalsIgnoreCase("done"))
                break;

            if (!input.contains(":")) {
                invalidFormat = true;
                break;
            }

            String[] parts = input.split(":");
            if (parts.length != 2 || !parts[0].matches("[A-Za-z]+")) {
```

```
        invalidFormat = true;
        break;
    }

    try {
        double quantity = Double.parseDouble(parts[1]);
        fruits.put(parts[0], 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 qty : fruits.values())
        total += qty;
    System.out.printf("%.2f\n", total);
}
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Manisha M  
Email: 240701304@rajalakshmi.edu.in  
Roll no: 240701304  
Phone: 9363972879  
Branch: REC  
Department: CSE - Section 9  
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**

```
// You are using Java
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 ch : line.toCharArray()) {  
    if (Character.isLetter(ch)) {  
        map.put(ch, map.getOrDefault(ch, 0) + 1);  
    }  
}  
  
System.out.println("Character Frequency:");  
for (Map.Entry<Character, Integer> entry : map.entrySet()) {  
    System.out.println(entry.getKey() + ": " + entry.getValue());  
}  
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Manisha M  
Email: 240701304@rajalakshmi.edu.in  
Roll no: 240701304  
Phone: 9363972879  
Branch: REC  
Department: CSE - Section 9  
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**

// You are using Java

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