8. Implementation of Sets and Maps

Aim/Objective: To understand the concept and implementation of concepts of sets and maps.

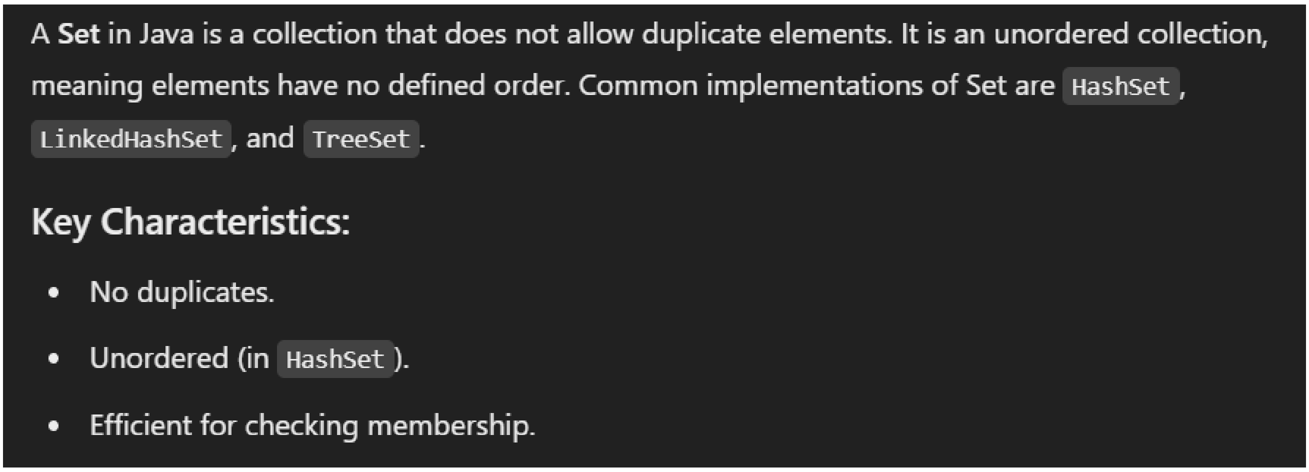
Description: The student will understand the concepts of sets and maps.

Pre-Requisites: Classes and Objects, HashMap, TreeMap, HashSet and TreeSet in JAVA.

Tools: Eclipse IDE for Enterprise Java and Web Developers

Pre-Lab:

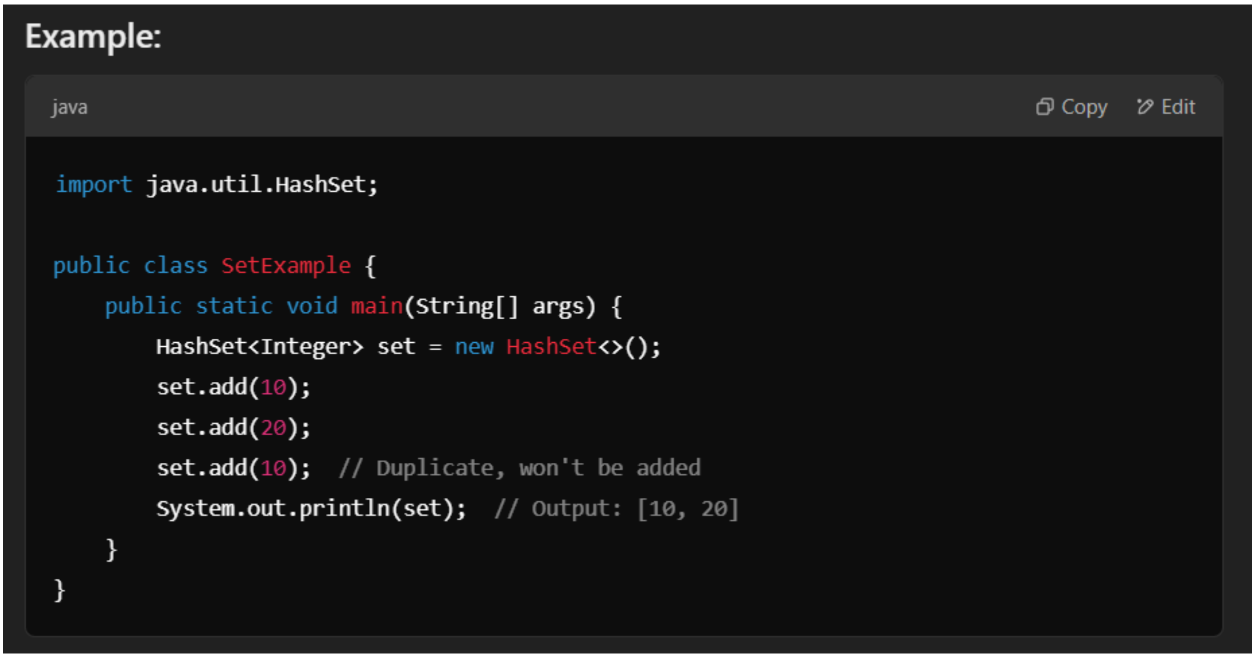
1) Explain the concept of a set in Java. What is the main characteristic of a set? Provide an example code snippet demonstrating the usage of a set.



A Set in Java is a collection that does not allow duplicate elements. It is an unordered collection, meaning elements have no defined order. Common implementations of Set are `HashSet`, `LinkedHashSet`, and `TreeSet`.

Key Characteristics:

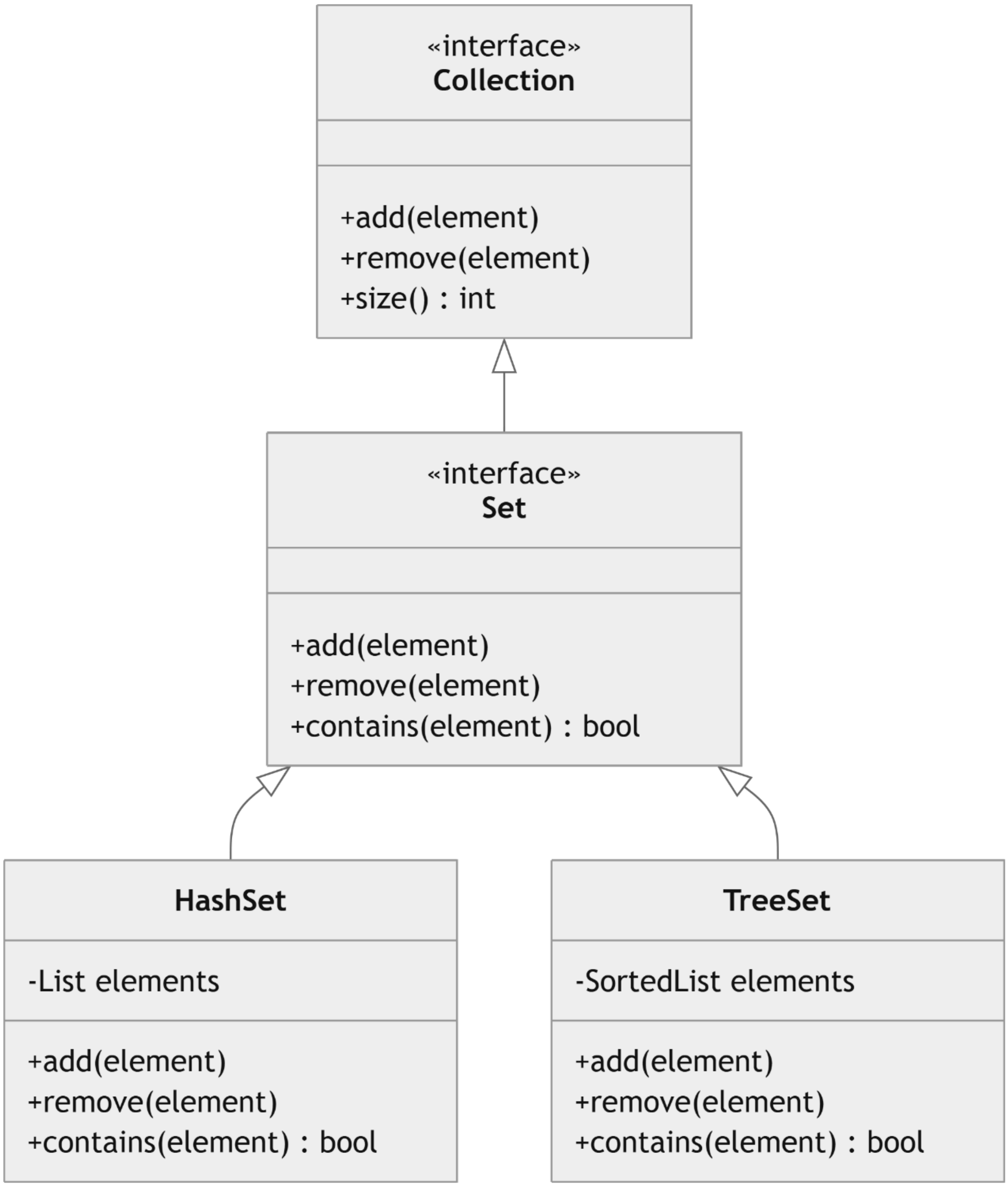
- No duplicates.
- Unordered (in `HashSet`).
- Efficient for checking membership.



Example:

```java
import java.util.HashSet;

public class SetExample {
    public static void main(String[] args) {
        HashSet<Integer> set = new HashSet<>();
        set.add(10);
        set.add(20);
        set.add(10);  // Duplicate, won't be added
        System.out.println(set);  // Output: [10, 20]
    }
}
```

2) Draw a UML diagram showing the relationship between the Set interface, the Collection interface, and two implementations of Set: HashSet and TreeSet.



In-Lab:

1) You are tasked with developing a contact management application where you need to store a collection of contacts. Each contact has a name and a phone number. Implement a class called ContactManager that uses a map to store the contacts, with the contact's name as the key and the phone number as the value. Write code to demonstrate the implementation of the ContactManager class.

Requirements

1. ContactManager Class:
   - Use a map to store contacts.
   - The key of the map is the contact's name (String).
   - The value of the map is the phone number (String).
   - Provide methods to add, remove, and retrieve contacts.
   - Provide a method to list all contacts.
2. Demonstration:
   - Create an instance of ContactManager.
   - Add several contacts.
   - Retrieve and display a contact's phone number.
   - Remove a contact.
   - List all contacts.

Procedure/Program:

```
import java.util.*;

class ContactManager {
   private Map<String, String> contacts = new HashMap<>();

   void add(String name, String phone) { contacts.put(name, phone); }
   void remove(String name) { contacts.remove(name); }
   String get(String name) { return contacts.get(name); }
   void list() { contacts.forEach((n, p) -> System.out.println("Name: " + n + ", 
Phone: " + p)); }
}

public class Main {
   public static void main(String[] args) {
      ContactManager cm = new ContactManager();
```

```
         cm.add("Alice", "123-456-7890");
         cm.add("Bob", "987-654-3210");
        cm.add("Charlie", "555-555-5555");

        System.out.println("Phone number of Alice: " + cm.get("Alice"));
        cm.remove("Bob");

        System.out.println("All contacts:");
        cm.list();
    }
}
```

OUTPUT

```
Phone number of Alice: 123-456-7890
All contacts:
Name: Alice, Phone: 123-456-7890
Name: Charlie, Phone: 555-555-5555
```

2) You are working on a program that stores student grades. Each student has a unique ID assigned to them. Design a data structure using sets and maps to efficiently store and retrieve student grades based on their ID. Write the code for adding a student's grade to the data structure given their ID and grade.

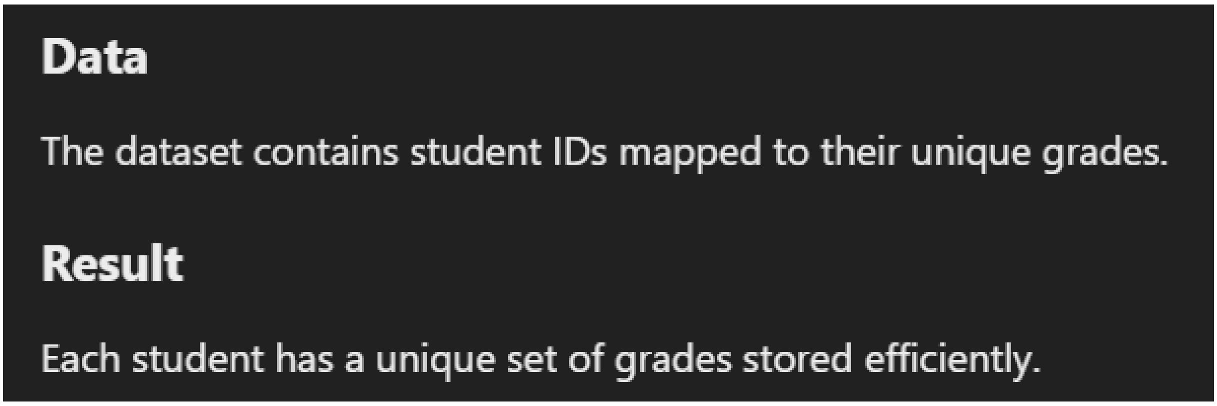
Procedure/Program:

```
import java.util.*;

public class StudentGrades {
   private Map<Integer, Set<Double>> studentGradesMap;

   public StudentGrades() {
      studentGradesMap = new HashMap<>();
   }

   public void addStudentGrade(int studentId, double grade) {
      studentGradesMap.putIfAbsent(studentId, new HashSet<>());
      studentGradesMap.get(studentId).add(grade);
   }

   public void printStudentGrades() {
      studentGradesMap.forEach((id, grades) ->
         System.out.println("Student ID: " + id + ", Grades: " + grades));
   }

   public static void main(String[] args) {
      StudentGrades sg = new StudentGrades();
      sg.addStudentGrade(101, 85.5);
      sg.addStudentGrade(101, 90.0);
      sg.addStudentGrade(102, 78.5);
      sg.addStudentGrade(102, 88.0);

      sg.printStudentGrades();
   }
}
```

OUTPUT

```
Student ID: 101, Grades: [85.5, 90.0]
Student ID: 102, Grades: [78.5, 88.0]
```

- ✓ **Data and Results:**



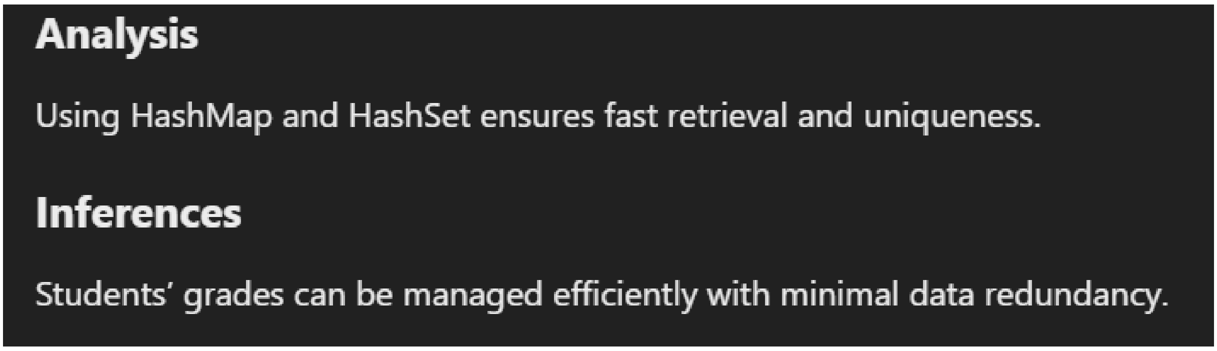
Data

The dataset contains student IDs mapped to their unique grades.

Result

Each student has a unique set of grades stored efficiently.

- ✓ **Analysis and Inferences:**



Analysis

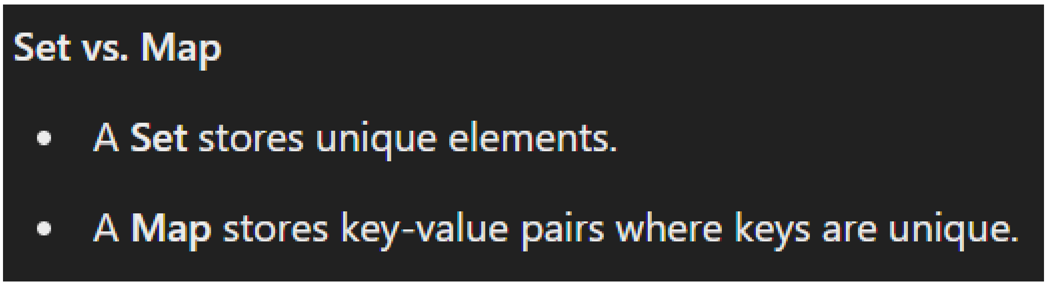
Using HashMap and HashSet ensures fast retrieval and uniqueness.

Inferences

Students’ grades can be managed efficiently with minimal data redundancy.

VIVA-VOCE Questions (In-Lab):

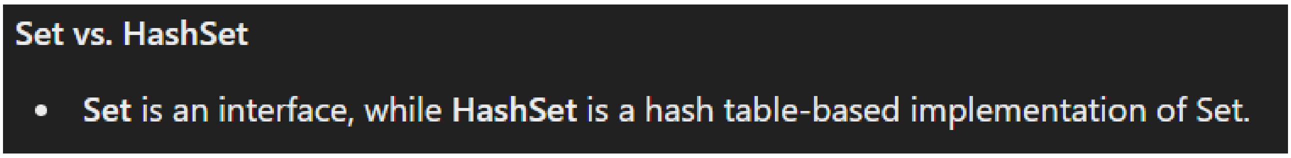
1) Explain the difference between a set and a map in programming.



Set vs. Map

- A **Set** stores unique elements.
- A **Map** stores key-value pairs where keys are unique.

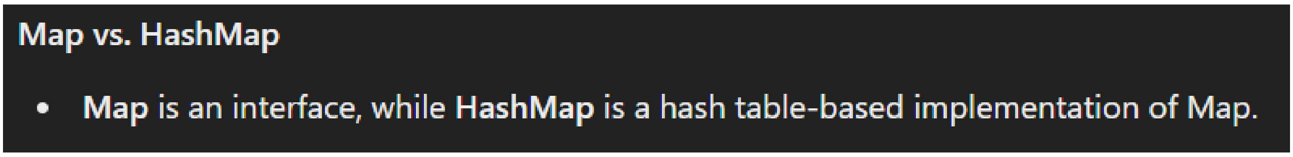
2) What is the difference between a Set and a HashSet?



Set vs. HashSet

- **Set** is an interface, while **HashSet** is a hash table-based implementation of Set.

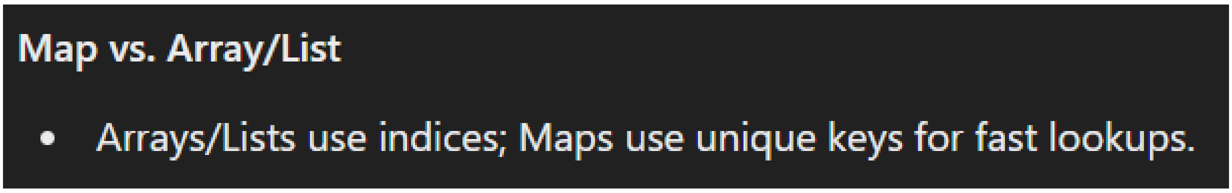
3) What is the difference between a Map and a HashMap?



Map vs. HashMap

- **Map** is an interface, while **HashMap** is a hash table-based implementation of Map.

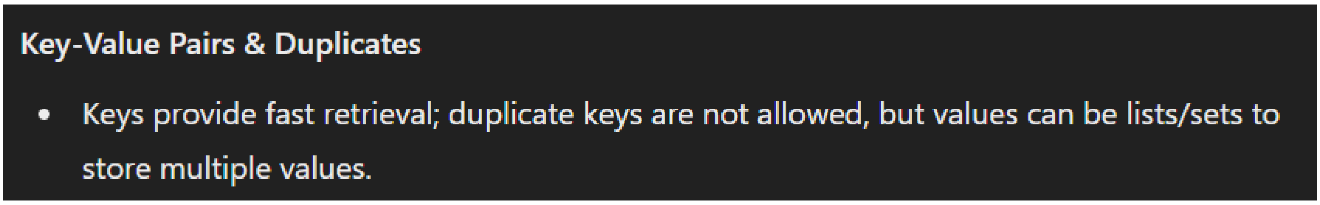
4) How does a map differ from an array or a list?



Map vs. Array/List

- Arrays/Lists use indices; Maps use unique keys for fast lookups.

5) What is the purpose of the key-value pairs in a map? Can the same key have multiple values?



Key-Value Pairs & Duplicates

- Keys provide fast retrieval; duplicate keys are not allowed, but values can be lists/sets to store multiple values.

Post-Lab:

1) Given a set of integers, write a code snippet to find the maximum value in the set.

Procedure/Program:

```
import java.util.Set;

public class MaxValueFinder {
    public static int findMax(Set<Integer> numbers) {
        if (numbers == null || numbers.isEmpty()) {
            throw new IllegalArgumentException("Set is empty or null");
        }

        int max = Integer.MIN_VALUE;
        for (int num : numbers) {
            if (num > max) {
                max = num;
            }
        }
        return max;
    }

    public static void main(String[] args) {
        Set<Integer> numbers = Set.of(5, 12, 8, 20, 3);
        System.out.println("Maximum value: " + findMax(numbers));
    }
}
```

OUTPUT

```
Maximum value: 20
```

2) Write a Java program that uses a Set to store a list of fruits. The program should then iterate over the Set and print out the fruits in alphabetical order.

Procedure/Program:

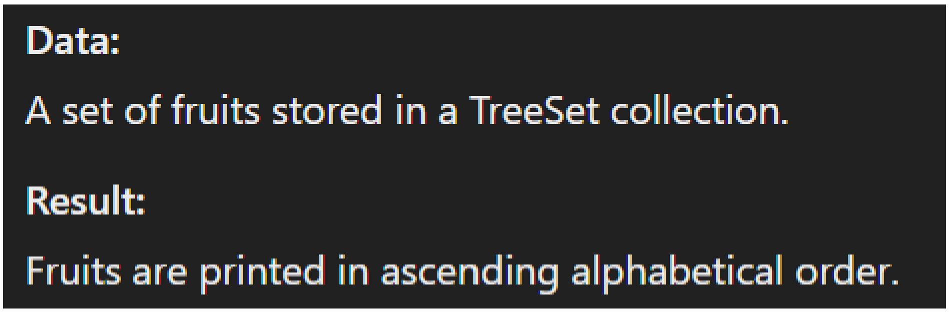
```
import java.util.Set;
import java.util.TreeSet;

public class FruitSet {
    public static void main(String[] args) {
        Set<String> fruits = new TreeSet<>();
        fruits.add("Banana");
        fruits.add("Apple");
        fruits.add("Mango");
        fruits.add("Orange");
        fruits.add("Grapes");

        for (String fruit : fruits) {
            System.out.println(fruit);
        }
    }
}
```

OUTPUT

```
Apple
Banana
Grapes
Mango
Orange
```

Experiment#		**Student ID**	
Date		**Student Name**	[@KLWKS_bot] THANOS

- ✓ **Data and Results:**



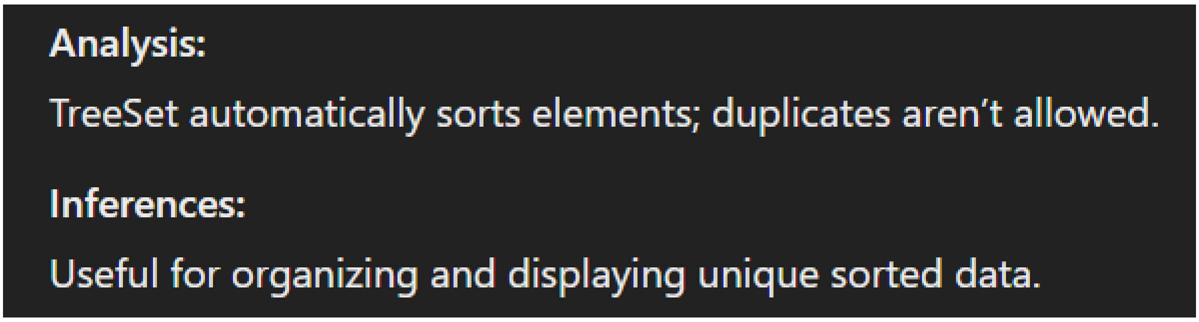
Data:

A set of fruits stored in a TreeSet collection.

Result:

Fruits are printed in ascending alphabetical order.

- ✓ **Analysis and Inferences:**



Analysis:

TreeSet automatically sorts elements; duplicates aren't allowed.

Inferences:

Useful for organizing and displaying unique sorted data.

Evaluator Remark (if Any):	**Marks Secured _____ out of 50**
	Signature of the Evaluator with Date

Evaluator MUST ask Viva-voce prior to signing and posting marks for each experiment.