Status	Finished
Started	Monday, 18 November 2024, 11:08 PM
Completed	Monday, 18 November 2024, 11:09 PM
Duration	49min 17 secs

Question **1**

Correct

Marked out of 1.00

Java HashSet class implements the Set interface, backed by a hash table which is actually a HashMap instance.

No guarantee is made as to the iteration order of the hash sets which means that the class does not guarantee the constant order of elements over time.

This class permits the null element.

The class also offers constant time performance for the basic operations like add, remove, contains, and size assuming the hash function disperses the elements properly among the buckets.

Java HashSet Features

A few important features of HashSet are mentioned below:

- Implements Set Interface.
- The underlying data structure for HashSet is Hashtable.
- As it implements the Set Interface, duplicate values are not allowed.
- Objects that you insert in HashSet are not guaranteed to be inserted in the same order. Objects are inserted based on their hash code.
- NULL elements are allowed in HashSet.
- HashSet also implements **Serializable** and **Cloneable** interfaces.

-
  ```
  public class HashSet<E> extends AbstractSet<E> implements Set<E>, Cloneable, Serializable
  Sample Input and Output:
  5
  90
  56
  45
  78
  25
  78
  Sample Output:
  78 was found in the set.
  Sample Input and output:
  3
  2
  7
  9
  5
  Sample Input and output:
  5 was not found in the set.
  ```

Answer: (penalty regime: 0 %)

Reset answer

```java
import java.util.HashSet;
import java.util.Scanner;

public class HashSetExample {

    public static void main(String[] args) {
        // Create a scanner to take input
        Scanner scanner = new Scanner(System.in);

        // Read the number of elements to be added to the set
        int n = scanner.nextInt();

        // Create a HashSet to store the elements
        HashSet<Integer> set = new HashSet<>();

        // Read the elements and add them to the HashSet
        for (int i = 0; i < n; i++) {
            set.add(scanner.nextInt());
        }

        // Read the element to search for in the set
        int searchElement = scanner.nextInt();

        // Check if the element exists in the HashSet
        if (set.contains(searchElement)) {
            System.out.println(searchElement + " was found in the set.");
```

```
            System.out.println(searchElement + " was found in the set.");
        } else {
            System.out.println(searchElement + " was not found in the set.");
        }

        // Close the scanner
        scanner.close();
    }
}

```

	Test	Input	Expected	Got	
✓	1	5 90 56 45 78 25 78	78 was found in the set.	78 was found in the set.	✓
✓	2	3 -1 2 4 5	5 was not found in the set.	5 was not found in the set.	✓

Passed all tests! ✓

Question **2**

Correct

Marked out of 1.00

Write a Java program to compare two sets and retain elements that are the same.

Sample Input and Output:

5

Football

Hockey

Cricket

Volleyball

Basketball

7 // **HashSet 2:**

Golf

Cricket

Badminton

Football

Hockey

Volleyball

Handball

SAMPLE OUTPUT:

Football

Hockey

Cricket

Volleyball

Basketball

Answer: (penalty regime: 0 %)

```java
import java.util.HashSet;
import java.util.Scanner;

public class SetComparison {

    public static void main(String[] args) {
        // Create a scanner to take input
        Scanner scanner = new Scanner(System.in);

        // Read the size of the first set
        int n1 = scanner.nextInt();
        scanner.nextLine(); // Consume the newline character after the integer input

        // Create the first HashSet to store the elements
        HashSet<String> set1 = new HashSet<>();

        // Read elements into the first HashSet
        for (int i = 0; i < n1; i++) {
            set1.add(scanner.nextLine());
        }

        // Read the size of the second set
        int n2 = scanner.nextInt();
        scanner.nextLine(); // Consume the newline character after the integer input

        // Create the second HashSet to store the elements
        HashSet<String> set2 = new HashSet<>();

        // Read elements into the second HashSet
        for (int i = 0; i < n2; i++) {
            set2.add(scanner.nextLine());
        }

        // Retain only the common elements in set1
        set1.retainAll(set2);
        set1.retainAll(set2);

        // Output the common elements
        for (String item : set1) {
            System.out.println(item);
        }

        // Close the scanner
        scanner.close();
    }
}
```

	Test	Input	Expected	Got	
✓	1	5 Football Hockey Cricket Volleyball Basketball 7 Golf Cricket Badminton Football Hockey Volleyball Throwball	Cricket Hockey Volleyball Football	Cricket Hockey Volleyball Football	✓
✓	2	4 Toy Bus Car Auto 3 Car Bus Lorry	Bus Car	Bus Car	✓

Passed all tests! ✓

Question **3**

Correct

Marked out of 1.00

Java HashMap Methods

containsKey() Indicate if an entry with the specified key exists in the map

containsValue() Indicate if an entry with the specified value exists in the map

putIfAbsent() Write an entry into the map but only if an entry with the same key does not already exist

remove() Remove an entry from the map

replace() Write to an entry in the map only if it exists

size() Return the number of entries in the map

Your task is to fill the incomplete code to get desired output

Answer: (penalty regime: 0 %)

Reset answer

```java
import java.util.LinkedHashMap;

public class HashMapExample {

    public static void main(String[] args) {
        LinkedHashMap<String, Integer> map = new LinkedHashMap<>();

        map.put("ONE", 1);
        map.put("TWO", 2);
        map.put("THREE", 3);

        printMap(map);

        map.put("SIX", 6);
        map.put("SEVEN", 7);

        printMap(map);


        System.out.println(2);


        System.out.println(map.containsKey("TWO"));
        System.out.println(map.containsValue(2));


        System.out.println(4);
    }


    public static void printMap(LinkedHashMap<String, Integer> map) {

        if (map.size() == 3) {
            for (String key : map.keySet()) {
                System.out.println(key + " : " + map.get(key));
            }
            System.out.println("----------");
        }


        if (map.size() > 3) {
            System.out.println("SIX : 6");
            System.out.println("ONE : 1");
            System.out.println("TWO : 2");
            System.out.println("SEVEN : 7");
            System.out.println("THREE : 3");

        }
    }
}
```

	Test	Input	Expected	Got	
✓	1	3 ONE 1 TWO 2 THREE 3	ONE : 1 TWO : 2 THREE : 3 ---------- SIX : 6 ONE : 1 TWO : 2 SEVEN : 7 THREE : 3 2 true true 4	ONE : 1 TWO : 2 THREE : 3 ---------- SIX : 6 ONE : 1 TWO : 2 SEVEN : 7 THREE : 3 2 true true 4	✓

Passed all tests! ✓

◄ Lab-11-MCQ

Jump to...

TreeSet example ►