

[Dashboard](#) / [My courses](#) / [CS23333-OOPUJ-2023](#) / [Lab-11-Set, Map](#) / [Lab-11-Logic Building](#)

Status	Finished
---------------	----------

Started	Tuesday, 19 November 2024, 3:29 AM
----------------	------------------------------------

Completed	Tuesday, 19 November 2024, 4:21 AM
------------------	------------------------------------

Duration	51 mins 59 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

```
1
2 import java.util.HashSet;
3 import java.util.Scanner;
4
5 public class HashSetExample {
6     public static void main(String[] args) {
7         // Create a scanner object to take input
8         Scanner scanner = new Scanner(System.in);
9
10        // Read the number of elements to be inserted into the set
11        int n = scanner.nextInt();
12
13        // Create a HashSet to store the elements
14        HashSet<Integer> set = new HashSet<>();
15
16        // Insert the elements into the set
```

```

16 // Insert the elements into the set
17 for (int i = 0; i < n; i++) {
18     int element = scanner.nextInt();
19     set.add(element);
20 }
21
22 // Read the element to be searched in the set
23 int searchElement = scanner.nextInt();
24
25 // Check if the element is in the set and print the result
26 if (set.contains(searchElement)) {
27     System.out.println(searchElement + " was found in the set.");
28 } else {
29     System.out.println(searchElement + " was not found in the set.");
30 }
31
32 // Close the scanner
33 scanner.close();
34 }
35 }
36

```

	Test	Input	Expected	Got	
✓	1	5 90 56 45 78 25 78	78 was found in the set.	78 was found in the set.	✓

	Test	Input	Expected	Got	
✓	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 %)

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

```

3 import java.util.Set;
4
5 public class SetComparison {
6     public static void main(String[] args) {
7         Scanner scanner = new Scanner(System.in);
8
9         // Read size and elements for the first set
10        int n1 = scanner.nextInt();
11        Set<String> set1 = new HashSet<>();
12        scanner.nextLine(); // Consume the newline after the integer input
13        for (int i = 0; i < n1; i++) {
14            set1.add(scanner.nextLine());
15        }
16
17        // Read size and elements for the second set
18        int n2 = scanner.nextInt();
19        Set<String> set2 = new HashSet<>();
20        scanner.nextLine(); // Consume the newline after the integer input
21        for (int i = 0; i < n2; i++) {
22            set2.add(scanner.nextLine());
23        }
24
25        // Retain only common elements
26        set1.retainAll(set2);
27
28        // Output the common elements
29        for (String element : set1) {
30            System.out.println(element);
31        }
32
33        scanner.close();
34    }
35 }

```

	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

```
1 import java.util.HashMap;
2 import java.util.Map.Entry;
3 import java.util.Set;
4 import java.util.Scanner;
5
6 class prog {
7     public static void main(String[] args) {
8         // Creating a HashMap with default initial capacity and load factor
9         HashMap<String, Integer> map = new HashMap<String, Integer>();
10
11         String name;
12         int num;
13         Scanner sc = new Scanner(System.in);
14
15         int n = sc.nextInt(); // Read the number of key-value pairs to be entered
16
17         for (int i = 0; i < n; i++) {
18             name = sc.next(); // Read the name (key)
19             num = sc.nextInt(); // Read the number (value)
20             map.put(name, num); // Add the key-value pair to the HashMap
21         }
22
23         // Printing key-value pairs from the first map
```

```
24 ▼ for (Entry<String, Integer> entry : map.entrySet()) {
25     System.out.println(entry.getKey() + " : " + entry.getValue());
26 }
27
28 System.out.println("-----"); // Separator line
29
30 // Creating another HashMap
31 HashMap<String, Integer> anotherMap = new HashMap<String, Integer>();
32
33 // Inserting key-value pairs to anotherMap using put() method
34 anotherMap.put("SIX", 6);
35 anotherMap.put("SEVEN", 7);
36
37 // Inserting key-value pairs of map to anotherMap using putAll() method
38 anotherMap.putAll(map); // Add all entries from 'map' to 'anotherMap'
39
40 // Printing key-value pairs of anotherMap
41 ▼ for (Entry<String, Integer> entry : anotherMap.entrySet()) {
42     System.out.println(entry.getKey() + " : " + entry.getValue());
43 }
44
45 // Adds key-value pair 'FIVE'-5 only if it is not present in map
46 map.putIfAbsent("FIVE", 5);
47
48 // Retrieving a value associated with key 'TWO'
49 int value = map.get("TWO");
50 System.out.println(value);
51
52 // Checking whether key 'ONE' exists in map
```

	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 ▶](#)