

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

Status	Finished
Started	Wednesday, 20 November 2024, 7:22 PM
Completed	Wednesday, 20 November 2024, 7:57 PM
Duration	34 mins 25 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 import java.util.HashSet;
2 import java.util.Scanner;
3
4 public class Prog {
5     public static void main(String[] args) {
6         Scanner sc = new Scanner(System.in);
7
8
9         int n = sc.nextInt();
10        HashSet<Integer> numbers = new HashSet<>();
11
12        for (int i = 0; i < n; i++) {
13            numbers.add(sc.nextInt());
14        }
15
16        int skey = sc.nextInt();
17
18        if (numbers.contains(skey)) {
19            System.out.println(skey + " was found in the set.");
20        } else {
21            System.out.println(skey + " was not found in the set.");
22        }
23
24        sc.close();
25    }
26 }
```

	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 %)

```

1 import java.util.HashSet;
2 import java.util.Scanner;
3 import java.util.Set;
4
5 public class CompareSets {
6     public static void main(String[] args) {
7         Scanner scanner = new Scanner(System.in);
8
9         // Read the size of the first set
10
11         int size1 = Integer.parseInt(scanner.nextLine());
12
13         // Create a HashSet to store the first set of elements
14         Set<String> set1 = new HashSet<>();
15         for (int i = 0; i < size1; i++) {
16             set1.add(scanner.nextLine());
17         }
18
19         // Read the size of the second set
20
21         int size2 = Integer.parseInt(scanner.nextLine());
22
23         // Create a HashSet to store the second set of elements
24         Set<String> set2 = new HashSet<>();
25
26         for (int i = 0; i < size2; i++) {
27             set2.add(scanner.nextLine());
28         }
29
30         // Retain common elements using the retainAll() method
31         set1.retainAll(set2);
32
33         // Print the common elements
34
35         for (String element : set1) {
36             System.out.println(element);
37         }
38     }
39 }

```

```

36         System.out.println(element);
37     }
38     // Close the scanner
39     scanner.close();
40 }
41 }

```

	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 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         int n = sc.nextInt();
15         for (int i = 0; i < n; i++) {
16             name = sc.next();
17             num = sc.nextInt();
18             map.put(name, num);
19         }
20
21         // Printing key-value pairs
22         Set<Entry<String, Integer>> entrySet = map.entrySet();
23         for (Entry<String, Integer> entry : entrySet) {
24             System.out.println(entry.getKey() + " : " + entry.getValue());
25         }
26         System.out.println("-----");
27
28         // Creating another HashMap
29         HashMap<String, Integer> anotherMap = new HashMap<String, Integer>();
30
31         // Inserting key-value pairs to anotherMap using put() method
32         anotherMap.put("SIX", 6);
33         anotherMap.put("SEVEN", 7);
34
35         // Inserting key-value pairs of map to anotherMap using putAll() method
36         anotherMap.putAll(map); // Filling in the missing code here
37         entrySet = anotherMap.entrySet();
38         for (Entry<String, Integer> entry : entrySet) {
39             System.out.println(entry.getKey() + " : " + entry.getValue());
40         }
41         map.putIfAbsent("FIVE", 5);
42
43         // Retrieving a value associated with key 'TWO'
44         Integer value = map.get("TWO"); // Using Integer instead of int to handle null case
45         System.out.println( (value != null ? value : "Key not found"));
46
47         // Checking whether key 'ONE' exists in map
48         System.out.println( map.containsKey("ONE"));
49
50         // Checking whether value '3' exists in map
51         System.out.println( map.containsValue(3));
52

```

	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 ▶](#)