Java collection(Set)

Set in java

The Set Interface is present in java.util package and extends the Collection interface. It is an unordered collection of objects in which duplicate values cannot be stored. It is an interface that implements the mathematical set. This interface adds a feature that restricts the insertion of the duplicate elements.

- No Specific Order: Does not maintain any specific order of elements (Exceptions: LinkedHashSet and TreeSet).
- Allows One Null Element: Most Set implementations allow a single null element.
- Implementation Classes: HashSet , LinkedHashSet and TreeSet.

Method	Description
add(element)	This method is used to add a specific element to the set. The function adds the element only if the specified element is not already present in the set else the function returns False if the element is already present in the Set.
addAll(collection)	This method is used to append all of the elements from the mentioned collection to the existing set. The elements are added randomly without following any specific order.
clear()	This method is used to remove all the elements from the set but not delete the set. The reference for the set still exists.
contains(element)	This method is used to check whether a specific element is present in the Set or not.
containsAll(collection)	This method is used to check whether the set contains all the elements present in the given collection or not. This method returns true if the set contains all the elements and returns false if any of the elements are missing.

Method	Description
hashCode()	This method is used to get the hashCode value for this instance of the Set. It returns an integer value which is the hashCode value for this instance of the Set.
isEmpty()	This method is used to check whether the set is empty or not.
iterator()	This method is used to return the iterator of the set. The elements from the set are returned in a random order.
remove(element)	This method is used to remove the given element from the set. This method returns True if the specified element is present in the Set otherwise it returns False.
removeAll(collection)	This method is used to remove all the elements from the collection which are present in the set. This method returns true if this set changed as a result of the call.
retainAll(collection)	This method is used to retain all the elements from the set which are mentioned in the given collection. This method returns true if this set changed as a result of the call.
size()	This method is used to get the size of the set. This returns an integer value which signifies the number of elements.
toArray()	This method is used to form an array of the same elements as that of the Set.

Ex: Java program Illustrating Set Interface

```
import java.util.*;

public class GFG {

    public static void main(String[] args)

    {
```

```
        // Demonstrating Set using HashSet
        // Declaring object of type String
        Set<String> hash_Set = new HashSet<String>();

        // Adding elements to the Set
        // using add() method
        hash_Set.add("harman");
        hash_Set.add("a");
        hash_Set.add("samsung");
        hash_Set.add("company");
        hash_Set.add("Set");

        // Printing elements of HashSet object
        System.out.println(hash_Set);
    }
}
```

Operations on the Set Interface

The set interface allows the users to perform the basic mathematical operation on the set. Let's take two arrays to understand these basic operations. Let set1 = [1, 3, 2, 4, 8, 9, 0] and set2 = [1, 3, 7, 5, 4, 0, 7, 5]. Then the possible operations on the sets are:

1. Intersection: This operation returns all the common elements from the given two sets. For the above two sets, the intersection would be:

Intersection = [0, 1, 3, 4]

2. Union: This operation adds all the elements in one set with the other. For the above two sets, the union would be:

Union = [0, 1, 2, 3, 4, 5, 7, 8, 9]

3. Difference: This operation removes all the values present in one set from the other set. For the above two sets, the difference would be:

Difference = [2, 8, 9]

Now let us implement the following operations as defined above as follows:

Example: Java Program Demonstrating Operations on the Set

```
// Importing all utility classes
import java.util.*;
public class SetExample {
    public static void main(String args[])
    {
        // Creating an object of Set class
        // Declaring object of Integer type
        Set<Integer> a = new HashSet<Integer>();
        // Adding all elements to List
        a.addAll(Arrays.asList(
            new Integer[] { 1, 3, 2, 4, 8, 9, 0 }));

      // Again declaring object of Set class
      // with reference to HashSet
        Set<Integer> b = new HashSet<Integer>();

      b.addAll(Arrays.asList(
            new Integer[] { 1, 3, 7, 5, 4, 0, 7, 5 }));
```

```
        // To find union
        Set<Integer> union = new HashSet<Integer>(a);
        union.addAll(b);
        System.out.print("Union of the two Set");
        System.out.println(union);

        // To find intersection
        Set<Integer> intersection = new HashSet<Integer>(a);
        intersection.retainAll(b);
        System.out.print("Intersection of the two Set");
        System.out.println(intersection);

        // To find the symmetric difference
        Set<Integer> difference = new HashSet<Integer>(a);
        difference.removeAll(b);
        System.out.print("Difference of the two Set");
        System.out.println(difference);
    }
}
```

Output

```
Union of the two Set[0, 1, 2, 3, 4, 5, 7, 8, 9]
Intersection of the two Set[0, 1, 3, 4]
Difference of the two Set[2, 8, 9]
```