JAVA - THE COLLECTION INTERFACE

The Collection interface is the foundation upon which the collections framework is built. It declares the core methods that all collections will have. These methods are summarized in the following table.

Because all collections implement Collection, familiarity with its methods is necessary for a clear understanding of the framework. Several of these methods can throw an **UnsupportedOperationException**.

SN	Methods with Description
1	**boolean add(Object obj)** Adds obj to the invoking collection. Returns true if obj was added to the collection. Returns false if obj is already a member of the collection, or if the collection does not allow duplicates.
2	**boolean addAll(Collection c)** Adds all the elements of c to the invoking collection. Returns true if the operation succeeded (i.e., the elements were added). Otherwise, returns false.
3	**void clear( )** Removes all elements from the invoking collection.
4	**boolean contains(Object obj)** Returns true if obj is an element of the invoking collection. Otherwise, returns false.
5	**boolean containsAll(Collection c)** Returns true if the invoking collection contains all elements of c. Otherwise, returns false.
6	**boolean equals(Object obj)** Returns true if the invoking collection and obj are equal. Otherwise, returns false.
7	**int hashCode( )** Returns the hash code for the invoking collection.
8	**boolean isEmpty( )** Returns true if the invoking collection is empty. Otherwise, returns false.
9	**Iterator iterator( )** Returns an iterator for the invoking collection.
10	**boolean remove(Object obj)** Removes one instance of obj from the invoking collection. Returns true if the element was removed. Otherwise, returns false.

11 **boolean removeAll(Collection c)**

Removes all elements of c from the invoking collection. Returns true if the collection changed (i.e., elements were removed). Otherwise, returns false.

12 **boolean retainAll(Collection c)**

Removes all elements from the invoking collection except those in c. Returns true if the collection changed (i.e., elements were removed). Otherwise, returns false

13 **int size( )**

Returns the number of elements held in the invoking collection.

14 **Object[ ] toArray( )**

Returns an array that contains all the elements stored in the invoking collection. The array elements are copies of the collection elements.

15 **Object[ ] toArray(Object array[ ])**

Returns an array containing only those collection elements whose type matches that of array.

Example:

Following is the example to explain few methods from various class implementations of the above collection methods:

```
import java.util.*;

public class CollectionsDemo {

   public static void main(String[] args) {
      List a1 = new ArrayList();
      a1.add("Zara");
      a1.add("Mahnaz");
      a1.add("Ayan");
      System.out.println(" ArrayList Elements");
      System.out.print("\t" + a1);

      List l1 = new LinkedList();
      l1.add("Zara");
      l1.add("Mahnaz");
      l1.add("Ayan");
      System.out.println();
      System.out.println(" LinkedList Elements");
      System.out.print("\t" + l1);

      Set s1 = new HashSet();
      s1.add("Zara");
      s1.add("Mahnaz");
      s1.add("Ayan");
      System.out.println();
      System.out.println(" Set Elements");
      System.out.print("\t" + s1);

      Map m1 = new HashMap();
      m1.put("Zara", "8");
      m1.put("Mahnaz", "31");
      m1.put("Ayan", "12");
      m1.put("Daisy", "14");
      System.out.println();
      System.out.println(" Map Elements");
```

```
      System.out.print("\t" + m1);
   }
}
```

This would produce the following result:

```
ArrayList Elements
        [Zara, Mahnaz, Ayan]
LinkedList Elements
        [Zara, Mahnaz, Ayan]
Set Elements
        [Zara, Mahnaz, Ayan]
Map Elements
        {Mahnaz=31, Ayan=12, Daisy=14, Zara=8}
```