Difference between List and Set in Java

BY CHAITANYA SINGH | FILED UNDER: JAVA COLLECTIONS

List and Set both are interfaces. They both extends Collection interface. In this post we are discussing the **differences between List and Set interfaces** in java.

List Vs Set

1) List is an ordered collection it maintains the insertion order, which means upon displaying the list content it will display the elements in the same order in which they got inserted into the list.

Set is an unordered collection, it doesn't maintain any order. There are few implementations of Set which maintains the order such as LinkedHashSet (It maintains the elements in insertion order).

2) List allows duplicates while Set doesn't allow duplicate elements. All the elements of a Set should be unique if you try to insert the duplicate element in Set it would replace the existing value.

3) List implementations: ArrayList, LinkedList etc.

Set implementations: HashSet, LinkedHashSet, TreeSet etc.

4) List allows any number of null values. Set can have only a single null value at most.

5) ListIterator can be used to traverse a List in both the directions(forward and backward) However it can not be used to traverse a Set. We can use Iterator (It works with List too) to traverse a Set.

6) List interface has one legacy class called Vector whereas Set interface does not have any legacy class.

When to use Set and When to use List?

The usage is purely depends on the requirement:

If the requirement is to have only unique values then Set is your best bet as any implementation of Set maintains unique values only.

If there is a need to maintain the insertion order irrespective of the duplicity then List is a best option. Both the implementations of List interface – ArrayList and LinkedList sorts the elements in their insertion order.

List Example

```
import java.util.List;
import java.util.ArrayList;
import java.util.LinkedList;

public class ListExample {

 public static void main(String[] args) {
   List<String> al = new ArrayList<String>();
   al.add("Chaitanya");
   al.add("Rahul");
   al.add("Ajeet");
   System.out.println("ArrayList Elements: ");
   System.out.print(al);

   List<String> ll = new LinkedList<String>();
   ll.add("Kevin");
   ll.add("Peter");
   ll.add("Kate");
   System.out.println("\nLinkedList Elements: ");
   System.out.print(ll);
 }
}
```

Output:

```
ArrayList Elements:
[Chaitanya, Rahul, Ajeet]
LinkedList Elements:
[Kevin, Peter, Kate]
```

Set Example

```
import java.util.Set;
import java.util.HashSet;
import java.util.TreeSet;

public class SetExample {

  public static void main(String args[]) {
    int count[] = {11, 22, 33, 44, 55};
    Set<Integer> hset = new HashSet<Integer>();
    try{
      for(int i = 0; i<4; i++){
         hset.add(count[i]);
      }
      System.out.println(hset);

      TreeSet<Integer> treeset = new TreeSet<Integer>(hset);
      System.out.println("The sorted list is:");
      System.out.println(treeset);
    }
    catch(Exception e){
        e.printStackTrace();
    }
  }
}
```

Output:

```
[33, 22, 11, 44]
The sorted list is:
[11, 22, 33, 44]
```