1. **Difference between LinkedList and ArrayList**
2. Both ArrayList and LinkedList are implementation of List interface.
3. They both maintain the elements insertion order which means while displaying ArrayList and LinkedList elements the result set would be having the same order in which the elements got inserted into the List.
4. Both these classes are non-synchronized and can be made synchronized explicitly by using [Collections.synchronizedList](https://docs.oracle.com/javase/6/docs/api/java/util/Collections.html" \l "synchronizedList(java.util.List)" \t "_blank) method.
5. **When to use LinkedList and when to use ArrayList?**
6. 1) As explained above the insert and remove operations give good performance (O(1)) in LinkedList compared to ArrayList(O(n)). Hence if there is a requirement of frequent addition and deletion in application then LinkedList is a best choice.
7. 2) Search (get method) operations are fast in Arraylist (O(1)) but not in LinkedList (O(n)) so If there are less add and remove operations and more search operations requirement, ArrayList would be your best bet.

Operations such as searching, sorting, insertion, manipulation, deletion, etc, can easily be performed by Java **Collections**.



**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 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](https://beginnersbook.com/2013/12/java-arraylist/), [LinkedList](https://beginnersbook.com/2013/12/linkedlist-in-java-with-example/) etc.

Set implementations: [HashSet](https://beginnersbook.com/2013/12/hashset-class-in-java-with-example/), [LinkedHashSet](https://beginnersbook.com/2013/12/linkedhashset-class-in-java-with-example/" \o "LinkedHashSet Class in Java with Example" \t "_blank), [TreeSet](https://beginnersbook.com/2013/12/treeset-class-in-java-with-example/" \o "TreeSet Class in Java with example" \t "_blank) etc.

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

5) [ListIterator](https://beginnersbook.com/2014/06/listiterator-in-java-with-examples/" \o "ListIterator in Java with examples" \t "_blank) 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](https://beginnersbook.com/2014/06/java-iterator-with-examples/) (It works with List too) to traverse a Set.

6) List interface has one legacy class called [Vector](https://beginnersbook.com/2013/12/vector-in-java/)whereas Set interface does not have any legacy class.