**What is difference between List and Set in Java** is a very [popular Java collection interview questions](http://javarevisited.blogspot.com/2011/11/collection-interview-questions-answers.html) and an important fundamental concept to remember while using Collections class in Java. Both List and Set are two of most important Collection classes Java Program use along with various Map implementation. Basic feature of List and Set are abstracted in List and Set interface in Java and then various implementation of List and Set adds specific feature on top of that e.g. [ArrayList in Java](http://javarevisited.blogspot.com/2011/05/example-of-arraylist-in-java-tutorial.html) is a List implementation backed by Array while LinkedList is another List implementation which works like linked list data-structure. In this Java tutorial we will see some fundamental difference between List and Set collections. Since List and Set are generified with introduction of [Generics in Java5](http://javarevisited.blogspot.com/2011/09/generics-java-example-tutorial.html) these difference also application to List and Set.

This article is in continuation of my earlier post on Collection e.g. [Difference between HashMap vs HashSet](http://javarevisited.blogspot.com/2011/09/difference-hashmap-vs-hashset-java.html), [Difference between HashMap and Hashtable](http://javarevisited.blogspot.com/2012/01/java-hashtable-example-tutorial-code.html) and [Difference between Concurrent Collection and Synchronized Collection](http://javarevisited.blogspot.com/2011/04/difference-between-concurrenthashmap.html). If you haven't read them already you may find them useful.

List vs Set in Java

[Difference between Set and List in Java](http://javarevisited.blogspot.com/2012/01/convert-arraylist-to-set-java-example.html)here are few note worthy *differences between List and Set in Java*. Remember that both of them are used to store objects and provides convenient API to insert, remove and retrieve elements, along with to support Iteration over collection.

1) Fundamental difference between List and Set in Java is **allowing duplicate elements**. List in Java allows duplicates while **Set doesn't allow any duplicate**. If you insert duplicate in Set it will replace the older value. Any implementation of Set in Java will only contains unique elements.

2) Another significant difference between List and Set in Java is order. **List is an Ordered Collection** while Set is an unordered Collection. List maintains **insertion order of elements**, means any element which is inserted before will go on lower index than any element which is inserted after. Set in Java doesn't  maintain any order. Though Set provide another alternative called SortedSet which can store Set elements in specific Sorting order defined by [Comparable and Comparator](http://javarevisited.blogspot.com/2011/06/comparator-and-comparable-in-java.html) methods of Objects stored in Set.

3) Set uses [equals() method](http://javarevisited.blogspot.com/2011/02/how-to-write-equals-method-in-java.html) to check uniqueness of elements stored in Set, while SortedSet uses [compareTo() method](http://javarevisited.blogspot.com/2011/11/how-to-override-compareto-method-in.html) to implement natural sorting order of elements. In order for an element to behave properly in Set and SortedSet, [equals and compareTo must be consistent](http://javarevisited.blogspot.com/2011/11/how-to-override-compareto-method-in.html) to each other.

4) Popular implementation of List interface in Java includes ArrayList, Vector and LinkedList. While popular implementation of Set interface includes HashSet, TreeSet and LinkedHashSet.

When to use List and Set in Java

Another good follow-up question is "**when do you use List and Set in Java**" , which can also be answered based on properties of List and Set we have learn here.These *difference between Set and List* also teaches us when to use Set and when to prefer List. its pretty clear that if you need to maintain insertion order or object and you collection can contain duplicates than List is a way to go. On the other hand if your requirement is to maintain unique collection without any duplicates than Set is the way to go.

Important point to note is that both List and Set are derived from Collection Interface. In short **main difference between List and Set in Java is that List is an ordered collection** which allows duplicates while Set is an unordered collection which doesn't allow duplicates.

Read more: <http://javarevisited.blogspot.com/2012/04/difference-between-list-and-set-in-java.html#ixzz3RF7H4Ch4>

LinkedList and [ArrayList](http://javarevisited.blogspot.com/2011/05/example-of-arraylist-in-java-tutorial.html) both implement List Interface but how they work internally is where the differences lies. Main **difference between ArrayList and LinkedLis**t is that ArrayList is implemented using re sizable array while LinkedList is implemented using doubly LinkedList. ArrayList is more popular among Java programmer than LinkedList as there are few scenarios on which LinkedList is a suitable collection than ArrayList. In this article we will see some *differences between LinkedList and ArrayList* and try to find out **when and where to use LinkedList over ArrayList**.

## LinkedList vs ArrayList in Java

[Difference between LinkedList and ArrayList in Java](http://javarevisited.blogspot.com/2011/10/google-dart-program-example-tutorial.html)All the differences between LinkedList and ArrayList has there root on difference between [Array](http://javarevisited.blogspot.com/2012/01/anonymous-array-example-java-create.html) and LinkedList data-structure. If you are familiar with Array and LinkedList data structure you will most likely derive following differences between them:

1) Since Array is an index based data-structure searching or getting element from Array with index is pretty fast. Array provides O(1) performance for get(index) method but remove is costly in ArrayList as you need to rearrange all elements. On the Other hand LinkedList doesn't provide Random or index based access and you need to iterate over linked list to retrieve any element which is of order O(n).

2) Insertions  are easy and fast in LinkedList as compared to ArrayList because there is no risk of resizing array

and copying content to new array if array gets full which makes adding into ArrayList of O(n) in worst case, while adding is O(1) operation in LinkedList in Java. ArrayList also needs to update its index if you insert something anywhere except at the end of array.

3) Removal is like insertions better in LinkedList than ArrayList.

4) LinkedList has more memory overhead than ArrayList because in ArrayList each index only holds actual object (data) but in case of LinkedList each node holds both data and address of next  and previous node.

## When to use LinkedList and ArrayList in Java

As I said LinkedList is not as popular as ArrayList but still there are situation where a LinkedList is better choice than ArrayList in Java. Use LinkedList in Java if:

1) Your application can live without Random access. Because if you need nth element in LinkedList you need to first traverse up to nth element O(n) and than you get data from that node.

2) Your application is more insertion and deletion driver and you insert or remove more than retrieval. Since insertion or

removal doesn't involve resizing its much faster than ArrayList.

That’s all on **difference between ArrayList and LinkedList in Java**. Use ArrayList in Java for all there situation where you need a **non-synchronized index based access**. ArrayList is fast and easy to use, just try to minimize array resizing by constructing arraylist with proper initial size.

Read more: <http://javarevisited.blogspot.com/2012/02/difference-between-linkedlist-vs.html#ixzz3RF7jUfGX>

**Difference between HashSet and TreeSet**   
  
**1. Ordering :** HashSet stores the object in random order . There is no guarantee that the element we  inserted first in the HashSet  will be printed first in the output . For example  

**import** **java.util.HashSet**;

**public** **class** **HashSetExample** {

**public** **static** **void** **main**(String[] args) {

HashSet<String> obj1= **new** HashSet<String>();

obj1.add("Alive");

obj1.add("is");

obj1.add("Awesome");

System.out.println(obj1);

}

}

**OUTPUT :** **[is, Awesome, Alive]**

Elements are sorted according to the natural ordering of its elements in TreeSet. If the objects can not   
be sorted in natural order than use [compareTo() method to sort the elements](http://javahungry.blogspot.com/2013/08/difference-between-comparable-and.html) of TreeSet object .

**import** **java.util.TreeSet**;

**public** **class** **TreeSetExample** {

**public** **static** **void** **main**(String[] args) {

TreeSet<String> obj1= **new** TreeSet<String>();

obj1.add("Alive");

obj1.add("is");

obj1.add("Awesome");

System.out.println(obj1);

}

}

**OUTPUT : [Alive, Awesome, is]**

**2. Null value :**   HashSet can store null object while TreeSet does not allow null object. If one try to store null object in TreeSet object , it will throw Null Pointer Exception.  
  
**3. Performance :** HashSet take constant time performance for the basic operations like add, remove contains and  size.While TreeSet guarantees log(n) time cost for the basic operations (add,remove,contains).  
  
**4. Speed :** HashSet is much faster than TreeSet,as performance time of HashSet is constant against the log time of TreeSet for most operations (add,remove ,contains and size) . Iteration performance of HashSet mainly depends on the load factor and initial capacity parameters.   
  
**5. Internal implementation :**  As we have already discussed [How hashset internally works in java](http://javahungry.blogspot.com/2013/08/how-sets-are-implemented-internally-in.html) thus, in one line HashSet are internally backed by hashmap. While TreeSet is backed by a  Navigable  TreeMap.

[](http://3.bp.blogspot.com/-xeMj5q7Sz2U/UxhFsyz4YbI/AAAAAAAAARg/JPfDsm45i5k/s1600/Difference+between+HashSet+and+TreeSet+in+Java+with+Example.jpg)

**6. Functionality :**    TreeSet is rich in functionality as compare to HashSet. Functions like pollFirst(),pollLast(),first(),last(),ceiling(),lower() etc. makes TreeSet easier to use than HashSet.

**7. Comparision :** HashSet uses equals() method for comparison in java while TreeSet uses compareTo() method for maintaining ordering .  
  
  
  
  
**To whom priority is given TreeSet comparator or Comparable.compareTo() .**  
  
Suppose there are elements in TreeSet which can be naturally sorted by the TreeSet , but we also added our own sorting method by implementing Comparable interface compareTo() method .  
Then to whom priority is given.  
  
  
Answer to the above question is that the Comparator passed into the TreeSet constructor has been given priority.  
According to [Oracle Java docs](http://docs.oracle.com/javase/6/docs/api/java/util/TreeSet.html#TreeSet%28java.util.Comparator%29)   
  
public TreeSet(Comparator comparator)  
  
Constructs a new, empty tree set, sorted according to the specified comparator.  
  
   *Parameters:*  
     
   *comparator* - the comparator that will be used to order this set. *If null*, the natural ordering of the elements will be used.  
  
  
  
  
**Similarities Between HashSet and TreeSet**  
  
**1. Unique Elements :**   Since HashSet and TreeSet both implements Set interface . Both are allowed to store only unique elements in their objects. Thus there can never be any duplicate elements inside the HashSet and TreeSet objects.  
  
**2.** **Not Thread Safe :** HashSet and TreeSet both are not synchronized or not thread safe.HashSet and TreeSet, both implementations are not synchronized. If multiple threads access a hash set/ tree set concurrently, and at least one of the threads modifies the set, it must be synchronized externally.  
  
**3. Clone() method copy technique:**  Both HashSet and TreeSet uses shallow copy technique to create a clone of  their objects .  
  
**4. Fail-fast Iterators :**  The iterators returned by this class's  method are fail-fast: if the set is modified at any time after the iterator is  created, in any way except through the iterator's own  remove method, the iterator will throw a  ConcurrentModificationException.  Thus, in the face of concurrent modification, the iterator fails quickly and cleanly, rather than risking arbitrary, non-deterministic behavior at   an undetermined time in the future.  
  
  
**When to prefer TreeSet over HashSet**  
  
1.  Sorted unique elements are required instead of unique elements.The sorted list given by TreeSet is always in ascending order.  
  
2.   TreeSet has greater locality than HashSet.  
  
If two entries  are near by in the order , then TreeSet places them near each other in data structure and hence in memory, while HashSet spreads the entries all over memory  regardless of the keys they are associated to.   
       
As we know Data reads from the hard drive takes much more latency time than data read from the cache or memory. In case data needs to be read from hard drive than prefer TreeSet as it has greater locality than HashSet.  
  
3. TreeSet uses Red- Black tree algorithm underneath to sort out the elements. When one need to perform read/write operations frequently , then TreeSet is a good choice.  
  
Thats it for the difference between HashSet and TreeSet , if you have any doubts then please mention in the comments.