## ArrayList vs Vector

|  |  |
| --- | --- |
| **ArrayList** | **Vector** |
| ArrayList is**not synchronized**, which means multiple threads can work on arrayList at the same time. | Vector is **synchronized**, which means only one thread at a time can access the code. |
| increments 50% of current array size if number of elements exceeds from its capacity. | increments 100% means doubles the array size if total number of elements exceeds than its capacity |
| It is not a legacy class, it is introduced in JDK 1.2. | It is a legacy class. |
| It is fast because it is non-synchronized. | It is slow because it is synchronized i.e. in multithreading environment, it will hold the other threads in runnable or non-runnable state until current thread releases the lock of object. |
| It uses Iterator interface to traverse the elements. | It uses Enumeration interface to traverse the elements. But it can use Iterator also |

*\* ArrayList is preferable when there is no specific requirement to use vector.*

## HashSet vs TreeSet

|  |  |
| --- | --- |
| **HashSet** | **TreeSet** |
| For operations like search, insert and delete. It takes constant time for these operations on average. HashSet is faster than TreeSet. HashSet is Implemented using a [hash table](https://www.geeksforgeeks.org/hashing-set-1-introduction/). | For operations like search, insert and delete. It takes constant time for these operations on average. HashSet is faster than TreeSet. HashSet is Implemented using a [hash table](https://www.geeksforgeeks.org/hashing-set-1-introduction/). |
| Elements in HashSet are not ordered. | TreeSet maintains objects in Sorted order defined by either Comparable or Comparator method in Java. TreeSet elements are sorted in ascending order by default. It offers several methods to deal with the ordered set like first(), last(), headSet(), tailSet(), etc. |
| HashSet allows null object.  HashSet uses [equals()](https://www.geeksforgeeks.org/overriding-equals-method-in-java/) method to compare two object in Set and for detecting duplicates. | TreeSet doesn’t allow null Object and throw NullPointerException, Why, because TreeSet uses compareTo() method to compare keys and compareTo() will throw java.lang.NullPointerException. |

## HashSet vs SortedSet

## **HashSet:**

The HashSet class implements the Set interface, backed by a hash table which is actually a HashMap instance.

Few important features of HashSet are:

* Implements [Set Interface](https://www.geeksforgeeks.org/set-in-java/).
* Underlying data structure for HashSet is hashtable.
* As it implements the Set Interface, duplicate values are not allowed.
* Objects that you insert in HashSet are not guaranteed to be inserted in same order. Objects are inserted based on their hash code.
* NULL elements are allowed in HashSet.
* HashSet also implements Searlizable and Cloneable interfaces.

## **SortedSet:**

SortedSet is an interface in collection framework. This interface extends Set and provides a total ordering of its elements. Exampled class that implements this interface is TreeSet.

All elements of a SortedSet must implement the Comparable interface (or be accepted by the specified Comparator) and all such elements must be mutually comparable (i.e, Mutually Comparable simply means that two objects accept each other as the argument to their compareTo method)

## Array vs List

## **Arrays in Java:**

* In Java all arrays are dynamically allocated.(discussed below)
* Since arrays are objects in Java, we can find their length using member length. This is different from C/C++ where we find length using sizeof.
* A Java array variable can also be declared like other variables with [] after the data type.
* The variables in the array are ordered and each have an index beginning from 0.
* Java array can be also be used as a static field, a local variable or a method parameter.
* The **size** of an array must be specified by an int value and not long or short.
* The direct superclass of an array type is Object.

## **List in Java:**

* The list is a child interface of Collection.
* It is an ordered collection of objects in which duplicate values can be stored.
* List preserves the insertion order, it allows positional access and insertion of elements.
* List Interface is implemented by ArrayList, LinkedList, Vector and Stack classes.
* List is an interface, and the instances of List can be created in many ways.
* After the introduction of Generics in Java 1.5, it is possible to restrict the type of object that can be stored in the List.
* List Interface extends Collection

## List vs Set

|  |  |
| --- | --- |
| **List** | **Set** |
| List is an ordered sequence of elements | Set is a distinct list of elements which is unordered |
| The user of this interface has precise control over where in the list each element is inserted. | A collection that contains no duplicate elements. |
| The user can access elements by their integer index (position in the list), and search for elements in the list. | Sets contain no pair of elements e1 and e2 such that e1.equals(e2), and at most one null element |
| Most of the List implementations (ArrayList,Vector) **implement RandomAccess** interface which is a marker interface for faster access | List uses one special Iterator called ListIterator **which supports iteration in both directions**. |
| List uses one special Iterator called ListIterator **which supports iteration in both directions**. | Set uses Iterator which supports only 1 way iteration |
| List takes less memory than set. | HashSet takes **5.5 times more memory** than ArrayList |

## NavigableSet vs NavigableMap

|  |  |
| --- | --- |
| **NavigableSet** | **NavigableMap** |
| NavigableSet represents a navigable set in Java Collection Framework and  inherits from the SortedSet interface | NavigableMap is an extension of Sorted Map |
| It behaves like a SortedSet with the exception that we have navigation methods available in addition to the sorting mechanisms of the SortedSet. | All of these methods are designed for locating, not traversing entries. |