## Why Collection Framework?

****Collections**** are nothing but group of objects stored in well defined manner. Earlier, Arrays are used to represent these group of objects. But, arrays are not re-sizable. size of the arrays are fixed. Size of the arrays can not be changed once they are defined. This causes lots of problem while handling group of objects. To overcome this drawback of arrays, ****Collection framework**** or simply collections are introduced in java from **JDK 1.2.**

**Collection Framework in java is a centralized and unified theme to store and manipulate the group of objects.** Java Collection Framework provides some pre-defined classes and interfaces to handle the group of objects. Using collection framework, you can store the objects as a ****list**** or as a ****set**** or as a ****queue**** or as a ****map**** and perform operations like adding an object or removing an object or sorting the objects without much hard work.

All classes and interfaces related to Collection Framework are placed in ****java.util**** package. ****java.util.Collection**** interface is at the top of class hierarchy of Collection Framework.

The entire collection framework is divided into four interfaces.

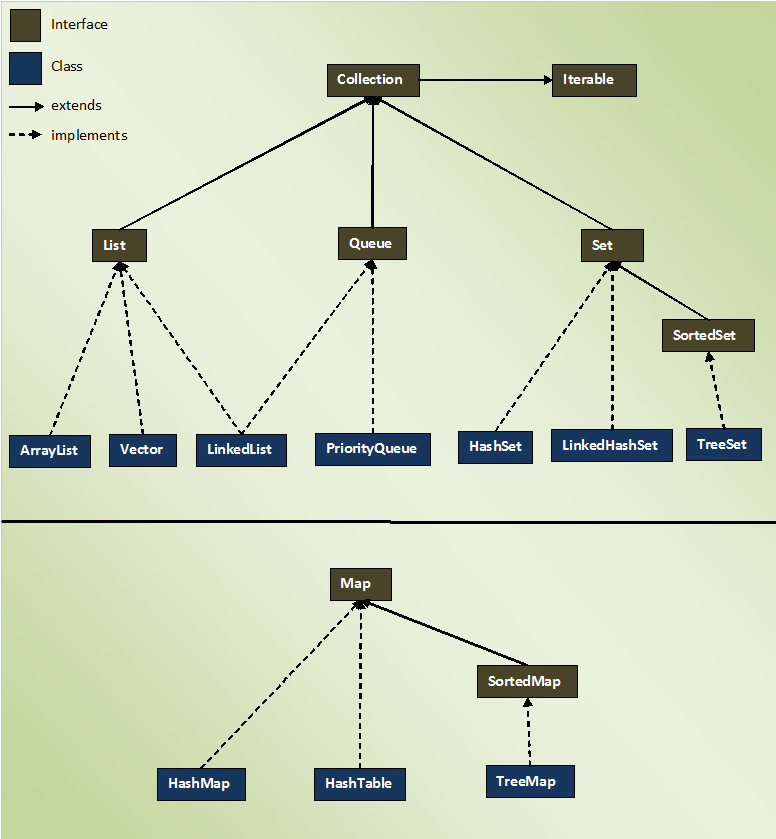
****1) List****  —> It handles sequential list of objects. ****ArrayList****, ****Vector**** and ****LinkedList**** classes implement this interface.

****2) Queue****  —> It handles the special group of objects in which elements are removed only from the head. ****LinkedList**** and ****PriorityQueue**** classes implement this interface.

****3) Set****  —> It handles the group of objects which must contain only unique elements. This interface is implemented by ****HashSet**** and ****LinkedHashSet**** classes and extended by ****SortedSet**** interface which in turn, is implemented by ****TreeSet****.

****4) Map****  —> This is the one interface in Collection Framework which is not inherited from *Collection* interface. It handles the group of objects as Key/Value pairs. It is implemented by ****HashMap**** and ****HashTable**** classes and extended by ****SortedMap**** interface which in turn is implemented by ****TreeMap****.

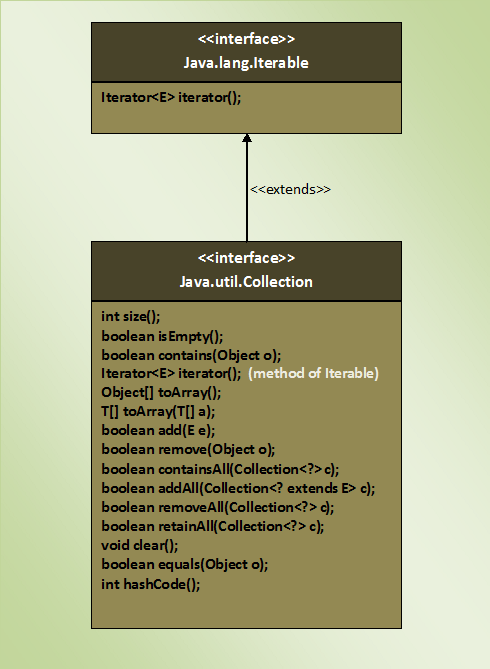
Three of above interfaces (*List*, *Queue* and *Set*) inherit from *Collection* interface. Although, *Map* is included in collection framework it does not inherit from *Collection* interface.



# [Collection Interface](https://javaconceptoftheday.com/collection-framework-collection-interface/)

****Collection interface**** is the root level interface in the collection framework. List, Queue and Set are all sub interfaces of Collection interface. JDK does not provide any direct implementations of this interface. But, JDK provides direct implementations of it’s sub interfaces.

Collection interface extends ****Iterable interface**** which is a member of java.lang package. Iterable interface has only one method called iterator(). It returns an Iterator object, using that object you can iterate over the elements of Collection. Here is the class diagram of Collection interface.



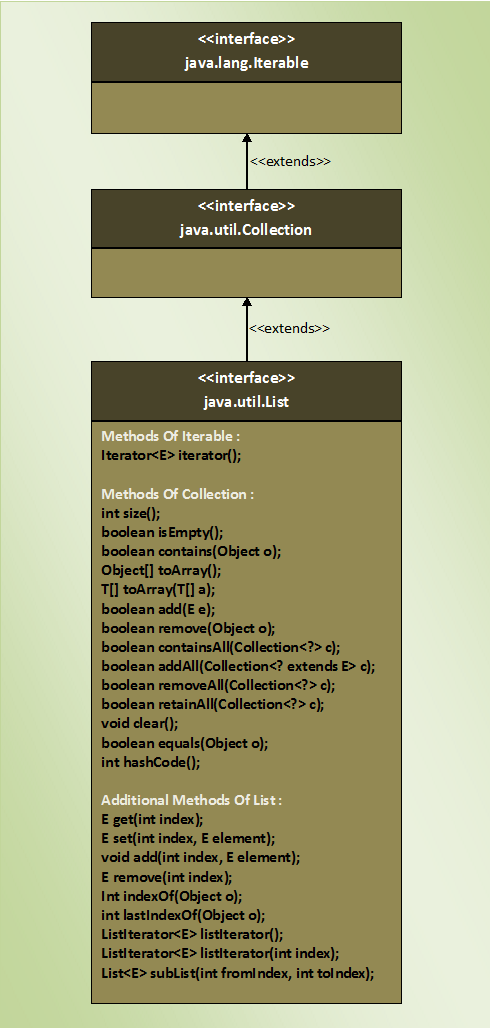
*equals()* and *hashcode()* methods in the *Collection* interface are not the methods of *java.lang.Object* class. Because, interfaces does not inherit from *Object* class. Only classes in java are inherited from *Object* class. Any classes implementing *Collection* interface must provide their own version of *equals()* and *hashcode()* methods or they can retain default version inherited from *Object* class.

# [List Interface](https://javaconceptoftheday.com/collection-framework-list-interface/)

****List**** ****Interface**** represents an ordered or sequential collection of objects. This interface has some methods which can be used to store and manipulate the ordered collection of objects. The classes which implement the List interface are called as ****Lists****. ArrayList, Vector and LinkedList are some examples of lists. You have the control over where to insert an element and from where to remove an element in the list.

Here are some properties of lists.

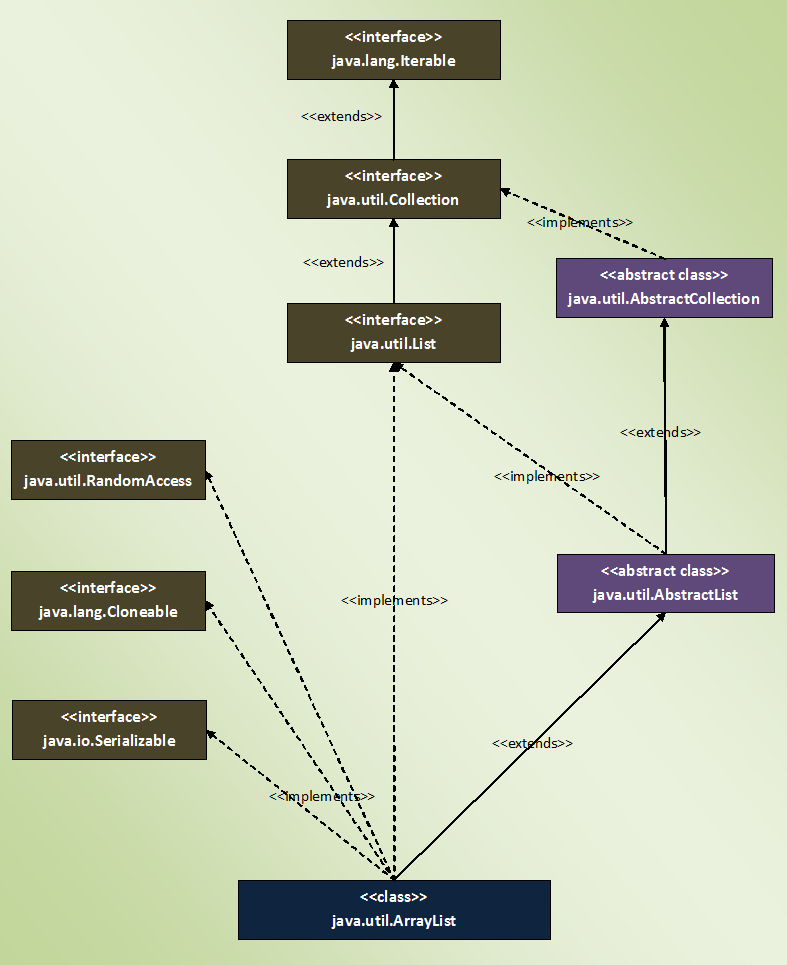
1. Elements of the lists are ordered using Zero based index.
2. You can access the elements of lists using an integer index.
3. Elements can be inserted at a specific position using integer index. Any pre-existing elements at or beyond that position are shifted right.
4. Elements can be removed from a specific position. The elements beyond that position are shifted left.
5. A list may contain duplicate elements.A list may contain multiple null elements.



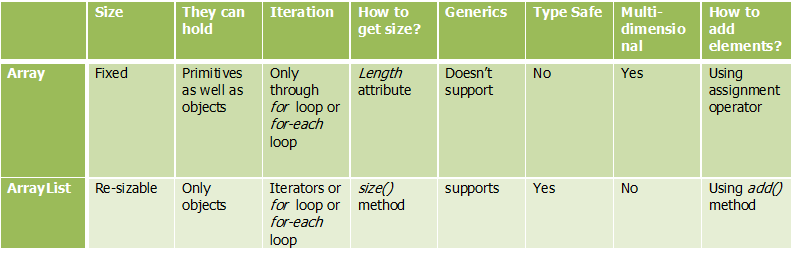
# [The ArrayList Class](https://javaconceptoftheday.com/collection-framework-arraylist-class/)

****ArrayList****, in simple terms, can be defined as **re-sizable array**. ArrayList is same like normal array but it can grow and shrink dynamically to hold any number of elements. **ArrayList is a sequential collection of objects which increases or decreases in size as we add or delete the elements.**

* In ArrayList, elements are positioned according to ****Zero-based index****. That means, elements are inserted from index 0.
* ****Default initial capacity** of an ArrayList is 10**. This capacity increases automatically as we add more elements to arraylist. You can also specify initial capacity of an ArrayList while creating it.
* ArrayList class implements ****List interface**** and extends ****AbstractList****. It also implements 3 marker interfaces – ****RandomAccess****, ****Cloneable**** and ****Serializable****. Here is hierarchy diagram of ArrayList class.
* When you insert an element in the middle of the ArrayList, the elements at the right side of that position are shifted one position right and when you delete an element, they will be shifted one position left. This feature of the ArrayList causes some performance issues as shifting of elements is time consuming if ArrayList has lots of elements.



### Array Vs ArrayList In Java :



# [Advantages Of Using ArrayList Over Arrays](https://javaconceptoftheday.com/advantages-of-using-arraylist-over-arrays/)

what are the drawbacks of arrays:

* Arrays are of fixed length. You can not change the size of the arrays once they are created.
* You can not accommodate an extra element in an array after they are created.
* Memory is allocated to an array during it’s creation only, much before the actual elements are added to it.

some advantages of using ArrayList over arrays:

1. You can define ArrayList as ****re-sizable array****. Size of the ArrayList is not fixed. ArrayList can grow and shrink dynamically.
2. Elements can be inserted at or deleted from a particular position.
3. ArrayList class has many methods to manipulate the stored objects.
4. ArrayList class has methods to perform solo modifications ( add(), remove()… ), bulk modifications ( addAll(), removeAll(), retainAll()… ), searching( indexOf(), lasIndexOf() ) and iterations( iterator() ).
5. If generics are not used, ArrayList can hold any type of objects.
6. You can traverse an ArrayList in both the directions – forward and backward using ListIterator.
7. ArrayList can hold multiple null elements.
8. ArrayList can hold duplicate elements.

ArrayList<String> list = new ArrayList<String>();

list.add("ONE");

list.add("TWO");

list.add("THREE");

list.add("FOUR");

ListIterator iterator = list.listIterator();

System.out.println("Elements in forward direction");

while (iterator.hasNext()){

System.out.println(iterator.next());

}

System.out.println("Elements in backward direction");

while (iterator.hasPrevious()){

System.out.println(iterator.previous());

}

## Java ArrayList Programming Questions :

1. ****Explain the different ways of constructing an ArrayList?****ArrayList can be created in 3 ways.
2. ****ArrayList()**** —> It creates an empty ArrayList with initial capacity of 10.
3. ****ArrayList(int initialCapacity)**** —> It creates an empty ArrayList with supplied initial capacity.
4. ****ArrayList(Collection c)**** —> It creates an ArrayList containing the elements of the supplied collection.
5. ****How do you increase the current capacity of an ArrayList? ensureCapacity() method**** is used to increase the current capacity of an ArrayList. However, capacity of an ArrayList is automatically increased when we try to add more elements than the current capacity. To manually increase the current capacity, ensureCapacity() method is used.
6. **How do you decrease the current capacity of an ArrayList to the current size?**

****trimToSize() method**** is used to trim the capacity of arrayList to the current size of ArrayList. Developers use this method to minimize the storage area of an ArrayList.

1. ****How do you find the number of elements present in an ArrayList?****

Using ****size()**** method. size() method returns number of elements present in an ArrayList.

1. ****How do you find out whether the given ArrayList is empty or not? : isEmpty()**** method of ArrayList is used to check whether the given ArrayList is empty or not. This method returns true if an ArrayList contains no elements otherwise returns false.
2. ****How do you check whether the given element is present in an ArrayList or not?** Using **contains() method** of ArrayList, we can examine whether the ArrayList contains the given element or not. This method returns true if ArrayList has that element otherwise returns false.**
3. ****How do you get the position of a particular element in an ArrayList?** We can use **indexOf() and lastIndexOf() methods** to find out the position of a given element in an ArrayList. indexOf() method returns index of first occurrence of a specified element where as lastIndexOf() method returns index of last occurrence of a specified element in an ArrayList. If element is not found, they will return -1.**
4. ****How do you convert an ArrayList to Array?** Using **toArray() method** of ArrayList class. toArray() method returns an array containing all elements of the ArrayList. This method acts as a bridge between normal arrays and collection framework in java.** Object[] array = list.toArray();
5. ****How do you retrieve an element from a particular position of an ArrayList? get() method** returns an element from a specified position of an ArrayList. This method takes index of the element as an argument.** System.out.println(list.get(3));
6. ****How do you replace a particular element in an ArrayList with the given element? set()**** method replaces a particular element in an Arraylist with the given element. This method takes two arguments. One is the index of the element to be replaced and another one is the element to be placed at that position.list.set(1, 000);
7. ****How do you append an element at the end of an ArrayList or insert an element at a particular position? add()**** method appends an element at the end of an ArrayList. list.add("TWO"); list.add(3, "BBB");
8. **remove() method** which takes int type as an argument is used to remove an element from a particular position of an ArrayList.
9. ****remove(Object obj)**** method removes the first occurrence of the specified element ‘****obj****‘. If that element doesn’t exist, ArrayList will be unchanged.
10. ****clear()**** method removes all elements of an ArrayList. ArrayList will be empty after this method is executed.
11. ****How do you retrieve a portion of an ArrayList?**** Using ****subList()**** method of ArrayList, we can retrieve a portion of an ArrayList. subList() method returns a view of a portion of an ArrayList in the given range. The returned subList is backed by original ArrayList. That means any changes made to subList will be reflected in original ArrayList or Vice-Versa.
12. ****How do you insert more than one element at a particular position of an ArrayList?**** addAll() method which takes two arguments, one is index and another one is Collection type, can be used for this requirement.

## What is the difference between ArrayList and Vector Class?

## 1) Thread Safety

This is the main difference between ArrayList and Vector class. ArrayList class is not thread safety where as Vector class is thread safety. Vector class is a synchronized class. Only one thread can enter into Vector object at any moment of time during execution. Where as ArrayList class is not synchronized. Multiple threads can access ArrayList object simultaneously.

## 2) Performance

ArrayList has better performance compared to Vector. It is because, Vector class is synchronized. It makes the threads to wait for object lock to enter into vector object. Where as ArrayList class is not synchronized. Threads need not to wait for object lock to access ArrayList object. This makes ArrayList faster than the Vector class.

## 3) Capacity Increment

Whenever the size of the ArrayList exceeds it’s capacity, the capacity is increased by half of the current capacity. Where as in case of Vector, the capacity is increased by ****Capacity Increment**** passed while creating the Vector object. If Capacity increment is not passed, capacity will be doubled automatically when the size exceeds it’s capacity. In ArrayList, there is no provision to pass Capacity increment while creating it. It’s capacity is automatically increased by half of the current capacity whenever size exceeds capacity.

## 4) Size

You can manually change the current size of the vector. Vector class has a method called ****setSize().**** Using this method, you can change the current size of the vector. If the new size is greater than the current size, new slots will be filled with null elements and if the new size is smaller than the current size, extra elements will be discarded. But in case of ArrayList, you can’t change the current size manually. It doesn’t have methods which alter it’s size. The size of the ArrayList will be changed only when you add or delete it’s elements.

## 5) Traversing The Elements.

ArrayList elements can be traversed using Iterator, ListIterator and using either normal or advanced for loop. But, vector elements can be traversed using Enumeration also along with these methods. Vector class has a method called ****elements()**** which returns Enumeration object containing all elements of the vector. Where as ArrayList does not have such methods.

## 6) Searching The Elements.

In ArrayList, you have to start searching for a particular element from the beginning of an Arralist. But in the Vector, you can start searching for a particular element from a particular position in a vector. This makes the search operation in Vector faster than in ArrayList.

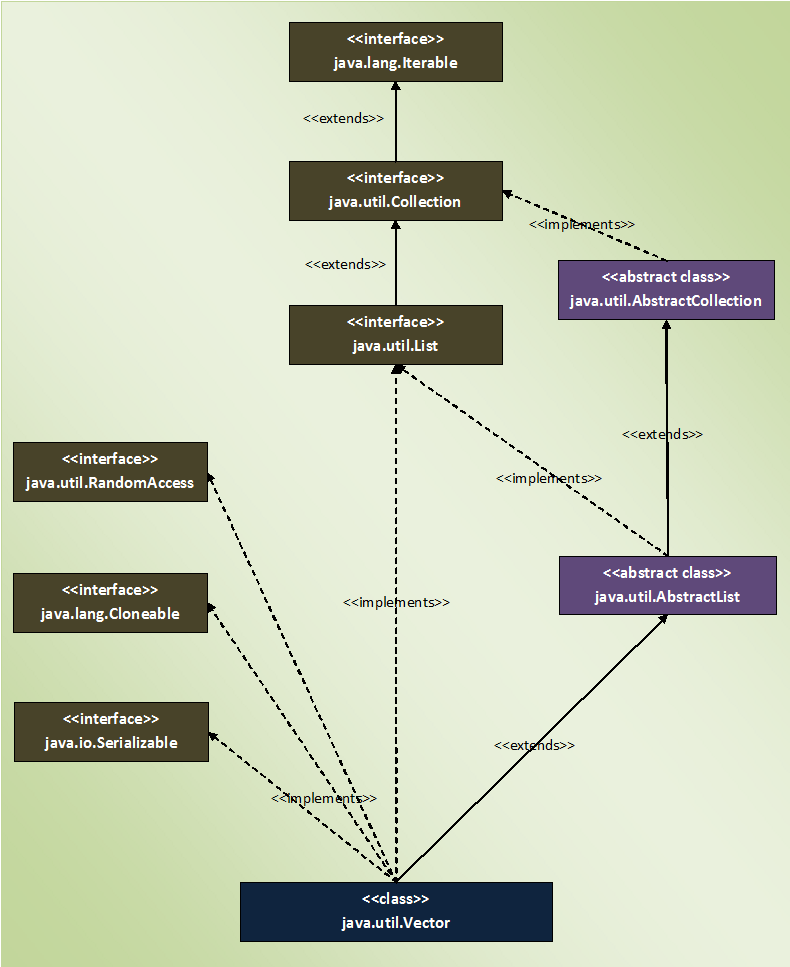
## 7) Legacy Code

Vector class is considered as Legacy code. Because, it exist in Java before the introduction of Collection Framework. Earlier it was not a part of Collections. Later it has been included in Collections. But, the older methods of vector class have been retained as it is.

# [The Vector Class](https://javaconceptoftheday.com/collection-framework-vector-class/)

****The Vector Class**** is also dynamically grow-able and shrink-able collection of objects like an [ArrayList](https://javaconceptoftheday.com/collection-framework-arraylist-class/" \o "Collection Framework – The ArrayList Class" \t "https://javaconceptoftheday.com/collection-framework-vector-class/_blank) class. But, the main difference between ArrayList and Vector is that ****Vector class is synchronized****. That means, only one thread can enter into vector object at any moment of time.

Vector class has same features as ArrayList. Vector class also extends ****AbstractList**** class and implements ****List interface****. It also implements 3 marker interfaces – ****RandomAccess****, ****Cloneable**** and ****Serializable****. Below is the hierarchy diagram of Vector class.



## Properties Of Vector Class :

* **Vector class is preferred over ArrayList class when you are developing a multi threaded application**. But, precautions need to be taken because vector may reduce the performance of your application as it is thread safety and only one thread is allowed to have object lock at any moment of time and remaining threads have to wait until a thread releases the object lock which is held by it. So, it is always recommended that if you don’t need thread safety environment, it is better to use ArrayList class than the Vector class.
* **Capacity Increment** : Capacity increment is an amount by which the capacity of the vector is automatically incremented whenever size of the vector exceeds it’s capacity. You can pass this capacity increment while creating a vector. If you don’t pass, capacity increment will be treated as zero and capacity of the vector will be doubled whenever size exceeds capacity.
* **Unlike an ArrayList, you can set the size of the Vector manually.** If the new size is greater than the current size, the new slots will be filled with null elements. If the new size is smaller than current size, then the extra elements will be discarded.