Collection Framework

Why it is called as framework because it contain the collection of classes and interface that work together.

Need of collection?

1. Array are fixed in size, once it is created we cannot change its size based on requirements.
2. Array can hold homogenous data elements.
3. There is no readymade method support available in array.

To overcome this problem, we should go for collection.

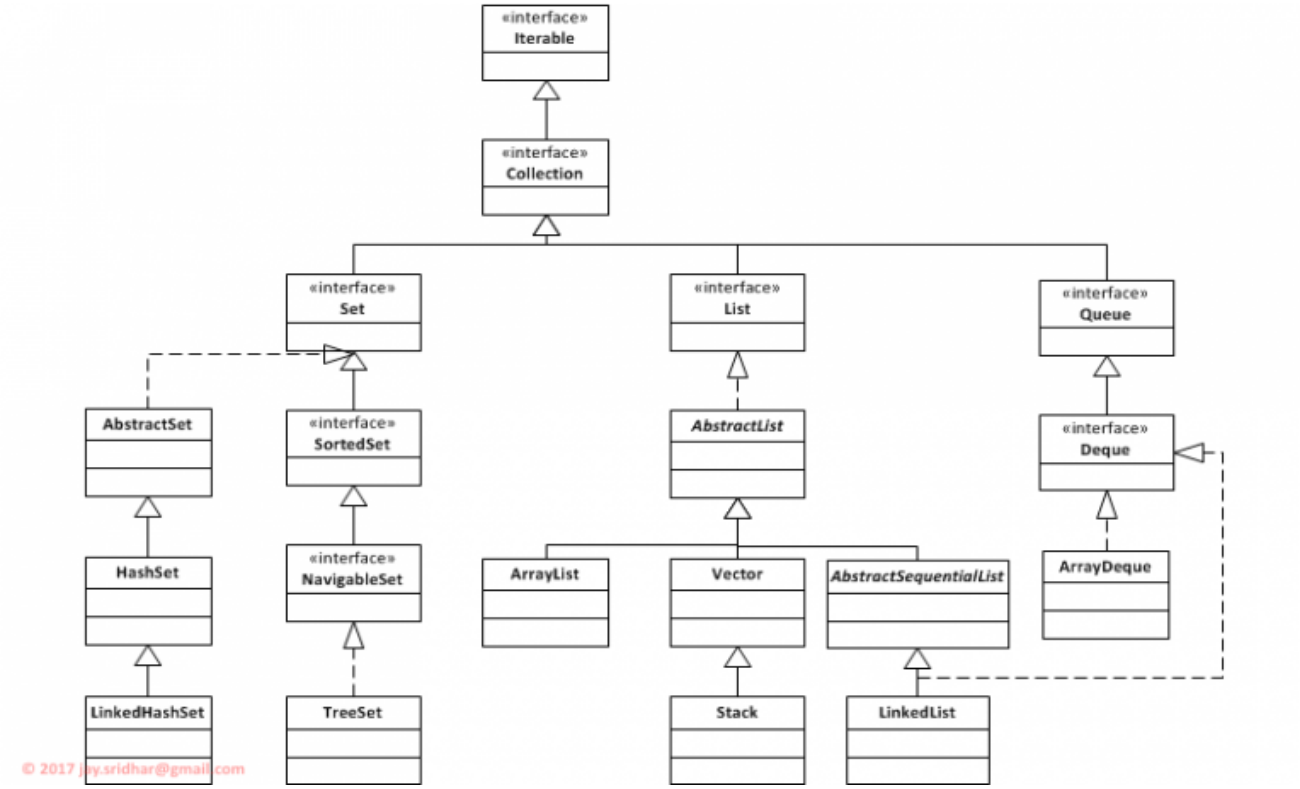
1. Collection are growable in nature, based on our requirement we can increase or decrease the size.
2. Collection can hold homogenous and heterogeneous elements.
3. Every collection class is implemented based on some data structure.
4. Hence readymade method support is available.

Difference between Array and Collection.

Array	Collection
Fixed in size	Growable in nature
Performance is good	Performance is not good
It hold homogenous data only.	It hold homogenous and heterogeneous type of data.
Readymade method support is not available	Readymade method support is available.
Array can hold both primitives data types and objects	It hold only objects not primitives data type.

Collection (I)-

If we want to represent the group of objects as single entity then we should go for collection.



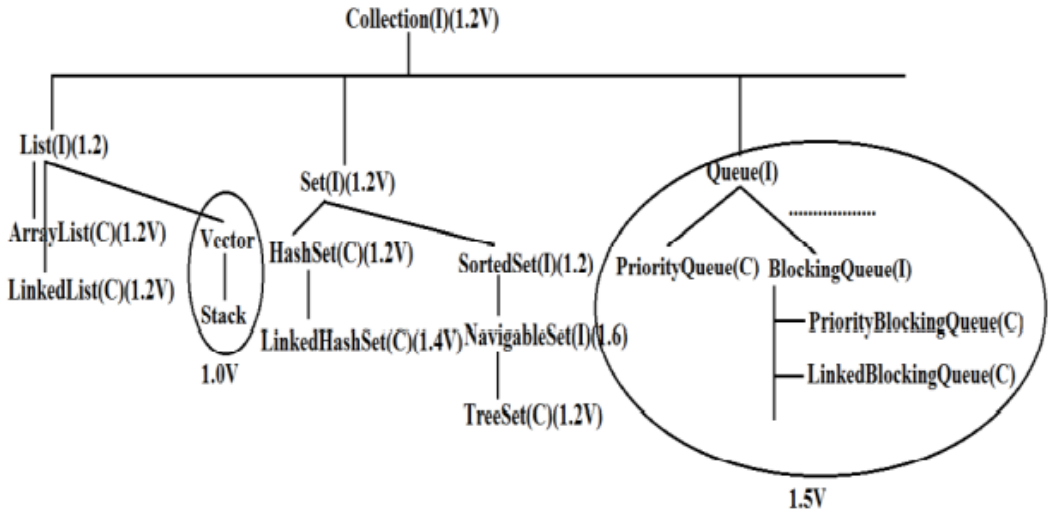


Diagram:

Collection Framework-

It defines several classes and interface which can be used to represent group of objects as single entity.

There are 9 key interfaces of collection framework.

1. Collection
2. List
3. Set
4. SortedSet
5. NavigableSet
6. Queue
7. Map
8. SortedMap
9. NavigableMap

1. Collection-
    1. If we want to represent group of individual objects as single entity then we should go for collection.

2. Collection interface defines the most common method which are applicable for any collection objects.
3. In general, Collection interface is considered as root interface of collection framework.

Note- There is no concrete class which implements collection interface directly.

Note:

- There is No Concrete Class which implements Collection Interface Directly.
- There is No Direct Method in Collection Interface to get Objects.
- Every Collection class implements Serializable and Cloneable Interfaces.

List

1. It is child interface of collection.
2. It is present in Java.util package.
3. If we want to represent group of individual objects as single entity where duplicates are allowed and insertion order is preserved then we should go for list.
4. We can differentiate by using index.
5. We can preserve insertion order by using index. Hence index plays important role. It will get the same sequence of element while retrieving the elements.
6. It has three subclasses in java such as- ArrayList LinkedList Vector, etc.

List of methods in List Interface :

1) void add(int index, Object o)
2) booleanaddAll(int index, Collection c)
3) Object get(int index)
4) Object remove(int index)
5) Object set(int index, Object new):To Replace the Element Present at

specified Index with provided Object and Returns Old Object.
6) int indexOf(Object o):Returns Index of 1st Occurrence of 'o'
7) int lastIndexOf(Object o)
8) ListIteratorlist Iterator();

1. ArrayList
    1. The ArrayList extends the AbstractList and implements the List interface.
    2. Duplicates are allowed.
    3. Insertion order is preserved.
    4. Heterogenous objects are allowed.
    5. Null insertion is possible.(we can add n number of null values in arraylist.

    Constructor-

    There are four types of constructor as below.

    1. ArrayList al= new ArrayList();

    Create the empty array list with default initial capacity 10. Once array list reaches its max capacity then new array list will be created with its new capacity.

    Formula
    Till Java 1.7 below formula was used :
    newCapacity=Current capacity * 3/2 +1;
    So current size is 10 then it will calculate like
    10*3/2+1,
    30/2+1
    15+1
    16
    New size will be created 16.
    Next time it will become 25
    For Java 1.7 onwards, formula being used is :
    newCapacity=(CurrentCapacity)+(CurrentCapacity >> 1)
    For ex. As initial capacity is 10 so,
    newCapacity = 10+(10>>1)=15

New size will be created 15.
Next time it will become 22

2. ArrayList al= new ArrayList(int initial_capacity);
   Here we are passing the int type value.

3. ArrayList al= new ArrayList(Collection c);
   Here we are passing the objects. This Constructor Meant for Inter Conversion between Collection Objects

Example-

```
import java.util.ArrayList;

public class ArrayListDemo {

	public static void main(String[] args) {

		ArrayList al= new ArrayList();
		al.add(50);
		al.add("ram");
		al.add(10);
		al.remove(1); //passing the index value here, ram will be removed from list.
		al.add(null);
		al.add(null);
		System.out.println(al);
	}
}
```

Output-
[50, ram, 10, null, null]

Note-

1. ArrayList is the best choice if our frequent operation is retrival

operation because it implements the random-access interface (Marker Interface). Hence, we can access the required object directly.

2. ArrayList is the worst choice if our frequent operation is insertion or deletion (because several shift operations are required for this).

How to get the synchronized version of ArrayList ?

By default, all the methods in array list are non-synchronized but we can get the synchronized version of arraylist by using collection class synchronized list method.

Example – ArrayList al= new ArrayList(); //this is non-synchronized
List list=Collections.synchronizedList(al); // this is synchronized