**COLLECTIONS INTERFACE HIERARCHY**

Collection

List

Queue

Set

ArrayList

LinkedList

Vector

Stack

PriorityQueue

SortedSet

HashSet

LinkedHashSet

TreeSet

The interfaces and classes for the Collection interface is in **java.util** package. It is the interface which is implemented by all the classes in the collection framework.

Collection interface has mainly three interfaces that implement it. These interfaces are:

1. List ----------------------------------------------- Interface
   1. ArrayList --------------------------- Class
   2. LinkedList --------------------------- Class
   3. Vector --------------------------- Class
      1. Stack --------------------------- Class
2. Queue ----------------------------------------------- Interface
3. PriorityQueue --------------------------- Class
4. Set ----------------------------------------------- Interface
5. HashSet --------------------------- Class
6. LinkedHashSet --------------------------- Class
7. SortedSet --------------------------- Interface
8. TreeSet--------------------------- Class

**List**

It is just similar to a list type data structure in which we can store the ordered collection of objects. It can have duplicate values.

**ArrayList**

It uses dynamic array to store the duplicate element of different datatypes. It also maintains the insertion order and is non-synchronized. The elements stored can be randomly accessed.

**LinkedList**

It uses a doubly linked list internally to store the elements. It can store the duplicate elements and maintains the insertion order. Manipulation is fast as no shifting is required.

**Vector**

It is similar to ArrayList but is synchronized and contains many methods that are not the part of Collection framework. Like: addElement(), clone() etc.

**Stack**

It is the subclass of Vector. It implements the LIFO data structure i.e. Satck.

**Queue**

It maintains the FIFO. It can be defined as an ordered list that is used to hold the elements which are about to be processed.

**PriorityQueue**

It implements the Queue interface. It holds the elements which are to be processed by their priorities. It doesn’t allow null value to be stored in the queue.

**Set**

It represents the unordered se of elements which doesn’t allow us to store the duplicate items. It can store at most one null value.

**HashSet**

It represents the collection that uses a hash table for storage. Hashing is used to store the elements in it. It contains unique items but insertion order is not maintained.

**LinkedHashSet**

It extends the HashSet class and implements the Set interface. It also contains unique elements. It maintain the insertion order and permits null elements.

**SortedSet**

It is the alternate of Set interface that provides a total ordering on its elements. The elements of the SortedSet are arranged in the ascending order.

**TreeSet**

It implements the Set interface that uses a tree for storage. Like Hashset, TreeSet also contains unique elements. The access and retrieval time of TreeSet is quite fast. The elements in it is stored in ascending order.