**Data Structure:**

Data structure is way of organizing a data in computer so that it can be used efficiently.

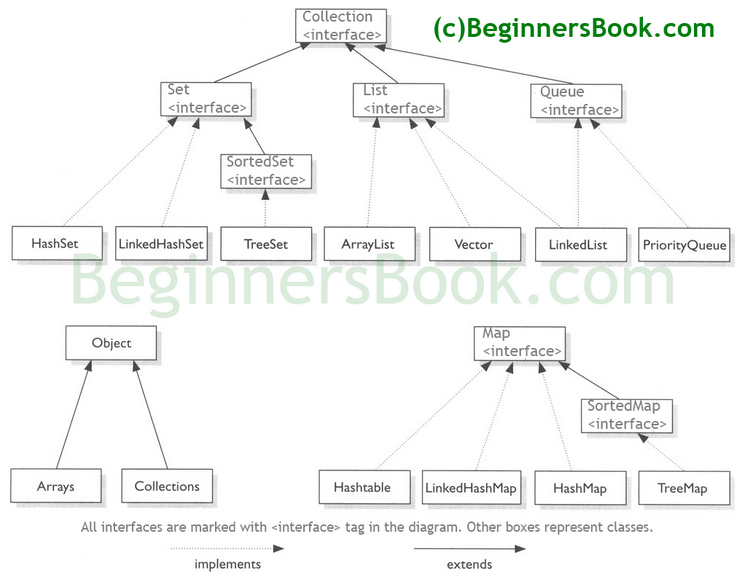
Two types of data structures are linear and non-linear.

* In Linear data structure data items are organized sequentially one after the other. Elements in data structure are traversed one after the other and only one element can be accessed one at a time.
* In non linear data items are not organized sequentially is called non linear. Data items could be connected to more than one element. All the elements in data structure cannot be traversed in single run.

**Collections in JAVA:**

* Collections framework is collection of interfaces and classes which helps in storing and processing data efficiently.

Collections framework Hierarchy:



**List Interface:**

* List is ordered collection of data.
* It may contain duplicate elements.
* Elements can be inserted or accessed by their position based on zero-based ind**ex.**

**Array List:**

* Array list class implements List interface.
* Array list is resizable array implementation of List interface.

**Array Vs ArrayList in java**

* Array is fixed size whereas array list is dynamic sized array in java
* Array is basic functionality provided by java , array List is part of collection of framework and it has set of methods to access the elements and modify them.
* Array can contain both primitive data types and objects of class depending on array definition.

However, array list can contain only objects of class it does not supports the primitive data types.

Note: When we do arrayList.add(1), it converts the primitive int data type to integer object.

* Since members of array list are always objects, content of array list is always reference to objects at different memory location. Therefore array list actual objects are never stored at contiguous memory location.

In case of primitive array elements are stored at contiguous locations, in case of objects similar to array list.

**Linked List:**

* It is linear data structure where the elements are not stored in contiguous memory and every element is separate object with data part and address part.
* Elements are linked using address part.
* Disadvantage of linked list is nodes cannot be accessed directly instead we need to start from the head and follow through link to reach the required the node.

Array list Versus Linked List:

* Array list is implemented with concept of dynamic array. However linked array list is implemented with concept of doubly linked list.
* Insertions are easy and fast in linked list, array list needs to update its index if element is inserted anywhere except at the end of array.
* Removal is faster is linked list.

Reason: Removal requires change in the pointer location in neighbor nodes of node which is going to be removed. In array list all the elements need to be shifted to fill out the space created by removing element.

* Linked list has more memory overhead than array list as each node needs to hold both data and address.
* Random access is not allowed in linked list.
* Array list search operation is fast compared to linked list.

Reason is as array list maintains the index based system for its elements which makes it faster for searching elements.

Similarities between array list and linked list are as follows,

1. Both ArrayList and LinkedList are implementation of List interface.
2. They both maintain the elements insertion order which means while displaying ArrayList and LinkedList elements the result set would be having the same order in which the elements got inserted into the List.
3. Both these classes are non-synchronized and can be made synchronized explicitly by using [Collections.synchronizedList](https://docs.oracle.com/javase/6/docs/api/java/util/Collections.html#synchronizedList(java.util.List)) method.
4. The iterator and list Iterator returned by these classes are fail-fast (if list is structurally modified at any time after the iterator is created, in any way except through the iterator’s own remove or add methods, the iterator will throw a [ConcurrentModificationException](https://docs.oracle.com/javase/6/docs/api/java/util/ConcurrentModificationException.html)).

**Set:**

* It is an interface which extends collection. Set is a collection that cannot contain the duplicate elements.
* It also adds a stronger contract on the behavior of equals and hash code operations, allowing set instances to be compared meaningfully.

**Hash Set:**

* It implements the set interface and it does not allow the duplicate elements.
* Underlying data structure is Hash map.
* It does not maintain the insertion order.
* It allows the Null values.
* Implements searlizable and clone able interfaces.
* Hash set uses the hash map internally to store the inserted elements, the values inserted is acts as key and java use a constant variable as its value of map.

**Tree Set:**

* It implements the sorted Set interface; it does not maintain insertion order.
* Objects in Tree set are sorted in ascending order.
* It throws classCastException if heterogeneous object is inserted.
* Tree set can be synchronized by synchronizing on some object which naturally encapsulates the set. Or it can be synchronized using Collections.synchronizedSet();
* Insertion of null into a TreeSet throws [NullPointerException](https://www.geeksforgeeks.org/null-pointer-exception-in-java/) because while insertion of null, it gets compared to the existing elements and null cannot be compared to any value.
* Secondly, if insertion of an object that cannot be compared with the elements existing in the set a ClassCastException is thrown.

Constructors of tree se are as below,

**TreeSet t = new TreeSet();**

**TreeSet t = new TreeSet(Comparator comp);**

**TreeSet t = new TreeSet(Collection col);**

**TreeSet t = new TreeSet(SortedSet s);**

**Hash Set versus Tree Set:**

* First major difference between HashSet and TreeSet is performance. HashSet is faster than TreeSet and should be preferred choice if sorting of element is not required.
* Second difference between HashSet and TreeSet is that HashSet allows null object but 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.
* Another significant difference between HashSet and TreeSet is that , HashSet is backed by HashMap while TreeSet is backed by TreeMap in Java.
* One more difference between HashSet and TreeSet which is worth remembering is that HashSet uses equals() method to compare two object in Set and for detecting duplicates while TreeSet uses compareTo() method for same purpose. if equals() and compareTo() are not consistent, i.e. for two equal object equals should return true while compareTo() should return zero, than it will break contract of Set interface and will allow duplicates in Set implementations like TreeSet
* Now most important difference between HashSet and TreeSet is ordering. HashSet doesn't guaranteed any order while TreeSet maintains objects in Sorted order defined by either Comparable or Comparator method in Java.

The ordering maintained in each collections is as follows;

* HashSet - undefined.
* HashMap - undefined
* LinkedHashSet - insertion order
* LinkedHashMap - insertion order of keys (by default), or 'access order'
* ArrayList - insertion order.
* LinkedList - insertion order.
* TreeSet - ascending order, according to Comparable / Comparator.

**List Vs Set**

* List is an ordered collection it maintains the insertion order, which means upon displaying the list content it will display the elements in the same order in which they got inserted into the list.
* Set is an unordered collection, it doesn’t maintain any order. There are few implementations of Set which maintains the order such as LinkedHashSet (It maintains the elements in insertion order).
* List allows duplicates while Set doesn’t allow duplicate elements. All the elements of a Set should be unique if you try to insert the duplicate element in Set it would replace the existing value.
* List implementations: [ArrayList](https://beginnersbook.com/2013/12/java-arraylist/), [LinkedList](https://beginnersbook.com/2013/12/linkedlist-in-java-with-example/) etc.
* Set implementations: [HashSet](https://beginnersbook.com/2013/12/hashset-class-in-java-with-example/), [LinkedHashSet](https://beginnersbook.com/2013/12/linkedhashset-class-in-java-with-example/), [TreeSet](https://beginnersbook.com/2013/12/treeset-class-in-java-with-example/) etc.
* List allows any number of null values. Set can have only a single null value at most.
* [ListIterator](https://beginnersbook.com/2014/06/listiterator-in-java-with-examples/) can be used to traverse a List in both the directions(forward and backward) However it can not be used to traverse a Set. We can use [Iterator](https://beginnersbook.com/2014/06/java-iterator-with-examples/) (It works with List too) to traverse a Set.
* List interface has one legacy class called [Vector](https://beginnersbook.com/2013/12/vector-in-java/) whereas Set interface does not have any legacy class.

