**Collections**

* Collections are grow able in nature. Based on our requirement we can increase or decrease the size.
* Collections can hold both homogeneous and heterogeneous element.
* Every collection class implemented based on some standard data structure. Hence for every requirement readymade method support is available.
* Being a programmer we are responsible to use methods and we are not responsible to implement those methods
* Collections can hold only object data types but not primitives.
* If we want represent group of individual object as single entity then we should go for collection.

**Collection Framework**

It contains several classes and interfaces to represent group of individual object as single entity.

**9 Key Interfaces**

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

**Collection (I)**

If we want represent a group of individual object as a single entity then we should go for collection. Collection defines most common methods which are applicable for any collection object. In general collection interface is considered as root interface of collection framework.

**Collections (C)**

Collections is an utility class present in java.util package. To define several utility methods for collection object like sorting, searching, etc.

**List(I)**

It is a child interface of collection. If we want represent a group of individual object as a single entity where duplicate are allowed and insertion order must be preserved then we should go for list.

**Collection -> List -> ArrayList -> LinkedList -> Vector -> Stack**

**Set (I)**

It is a child interface of collection. If we want represent a group of individual object as a single entity where duplicates are not allowed and insertion order not required then we should go for Set.

**SortedSet (I)**

It is a child interface of collection. If we want represent a group of individual object as a single entity where duplicates are not allowed and all objects should be inserted to some sorting order then we should go sorted set.

**NavigableSet (I)**

It is a child interface of collection. It contains several methods for navigation purposes.

**Collection -> Set -> SortedSet -> NavigableSet -> TreeSet**

**Queue (I)**

It is a child interface of collection. If we want represent a group of individual objects prior to processing then we should go for Queue. Usually Queue Follows FIFO order based on our requirement. We can implement our own priority order also.

Eg – mail Service

Collection -> Queue -> PriorityQueue -> BlockingingQueue -> …..  
 PriorityBlockingQueue  
 LinkedBlockingQueue

**Map**

Map is not child Interface of collection. If we want to represent a group of object as KEY, Value pair then we should go for Map. Both Key and value are Object only. Duplicate keys are not allowed. But values can be duplicated.

Dictionary (AC)  
Map -> HashMap -> WeakHashMap -> IdentityHashMap -> Hashtable  
 LinkedHashMap Properties

**SortedMap**

It is a child interface of collection. If we want represent a group of Key, Value pair according to some sorting order of key.

**NavigableMap**

It is a child interface of SortedMap and it defines several methods for Navigation Purposes.

Map -> SortedMap -> NavigableMap -> TreeMap