The Collection in Java is a framework that provides an architecture to store and manipulate the group of objects. Java Collections can achieve all the operations that we perform on a data such as searching, sorting, insertion, manipulation, and deletion.

Java Collection means a single unit of objects. Java Collection framework provides many interfaces (Set, List, Queue, Deque) and classes (ArrayList, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet).

**\*\*1. Collection Interface\*\***

Collection interface is at the root of the hierarchy. Collection interface provides all general purpose methods which all collections classes must support (or throw UnsupportedOperationException). It extends **\*\*Iterable\*\*** interface which adds support for iterating over collection elements using the â€œfor-each loopâ€� statement.

**\*\*2. List\*\***

Lists represents an **\*\*ordered collection\*\*** of elements. Using lists, we can access elements by their integer index (position in the list), and search for elements in the list. index start with 0, just like an array.

Some useful classes which implement List interface are â€“ **\*\*ArrayList\*\***, **\*\*CopyOnWriteArrayList\*\***, **\*\*LinkedList\*\***, **\*\*Stack\*\*** and **\*\*Vector\*\***.

**\*\*3. Set\*\***

Sets represents a collection of **\*\*sorted\*\*** elements. Sets do not allow the duplicate elements. Set interface does not provides no guarantee to return the elements in any predictable order; though some Set implementations store elements in their natural ordering and guarantee this order.

Some useful classes which implement Set interface are â€“ **\*\*ConcurrentSkipListSet\*\***, **\*\*CopyOnWriteArraySet\*\***, **\*\*EnumSet\*\***, **\*\*HashSet\*\***, **\*\*LinkedHashSet\*\*** and **\*\*TreeSet\*\***.

**\*\*4. Map\*\***

The Map interface enable us to store data in key-value pairs (keys should be immutable). A map cannot contain duplicate keys; each key can map to at most one value.

The Map interface provides three collection views, which allow a mapâ€™s contents to be viewed as a set of keys, collection of values, or set of key-value mappings. Some map implementations, like the TreeMap class, make specific guarantees as to their order; others, like the HashMap class, do not.

Some useful classes which implement Map interface are â€“ **\*\*ConcurrentHashMap\*\***, **\*\*ConcurrentSkipListMap\*\***, **\*\*EnumMap\*\***, **\*\*HashMap\*\***, **\*\*Hashtable\*\***, **\*\*IdentityHashMap\*\***, **\*\*LinkedHashMap\*\***, **\*\*Properties\*\***, **\*\*TreeMap\*\*** and **\*\*WeakHashMap\*\***.

**\*\*5. Stack\*\***

The Java Stack interface represents a classical stack data structure, where elements can be pushed to last-in-first-out (LIFO) stack of objects. In Stack we push an element to the top of the stack, and popped off from the top of the stack again later.

**\*\*6. Queue\*\***

A queue data structure is intended to hold the elements (put by producer threads) prior to processing by consumer thread(s). Besides basic Collection operations, queues provide additional insertion, extraction, and inspection operations.

Some useful classes which implement Map interface are â€“ **\*\*ArrayBlockingQueue, ArrayDeque, ConcurrentLinkedDeque, ConcurrentLinkedQueue, DelayQueue, LinkedBlockingDeque, LinkedBlockingQueue, LinkedList, LinkedTransferQueue, PriorityBlockingQueue, PriorityQueue and SynchronousQueue\*\***.

**\*\*7. Deque\*\***

A double ended queue (pronounced â€œdeckâ€œ) that supports element insertion and removal at both ends. When a deque is used as a queue, FIFO (First-In-First-Out) behavior results. When a deque is used as a stack, LIFO (Last-In-First-Out) behavior results.

Some common known classes implementing this interface are **\*\*ArrayDeque, ConcurrentLinkedDeque, LinkedBlockingDeque\*\*** and **\*\*LinkedList\*\***.

| |List |Set | Queue |Map |

|-----------|-------|-----------|----------|--------------------------------|

|Order |Yes |No |Yes |No |

|Duplicates |Yes |No |Yes |No (Allow duplicate values not keys)|

|Null Values|Yes |Single Null| Yes (LinkedList Queue). No (Priority Queue).|Single null key and many null values|

**\*\*The Java Collections Framework provides the following benefits:\*\***

\* Reduces programming effort

\* Increases program speed and quality

\* Allows interoperability among unrelated APIs

\* Reduces effort to learn and to use new APIs

\* Reduces effort to design new APIs

\* Fosters software reuse