# Collections

**Intro**

* **Collection** is the root of the collection hierarchy. A collection represents a group of objects known as its elements. The Java platform doesn’t provide any direct implementations of this interface.
* **Set** is a collection that cannot contain duplicate elements. This interface models the mathematical set abstraction and is used to represent sets, such as the deck of cards.
* **List** is an ordered collection and can contain duplicate elements. You can access any element from it’s index. List is more like array with dynamic length.
* A **Map** is an object that maps keys to values. A map cannot contain duplicate keys: Each key can map to at most one value.
* Collections define three static variables: EMPTY\_SET, EMPTY\_LIST, and EMPTY\_MAP. All are immutable



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Interfaces** | **Hash table Implementations** | **Resizable array Implementations** | **Tree Implementations** | **Linked list Implementations** | **Hash table + Linked list Implementations** |
| Set | HashSet |  | TreeSet |  | LinkedHashSet |
| List |  | ArrayList |  | LinkedList |  |
| Queue |  |  |  |  |  |
| Deque |  | ArrayDeque |  | LinkedList |  |
| Map | HashMap |  | TreeMap |  | LinkedHashMap |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Collection** | **Ordering** | **Random Access** | **Key-Value** | **Duplicate Elements** | **Null Element** | **Thread Safety** |
| ArrayList | Yes | Yes | No | Yes | Yes | No |
| LinkedList | Yes | No | No | Yes | Yes | No |
| HashSet | No | No | No | No | Yes | No |
| TreeSet | Yes | No | No | No | No | No |
| HashMap | No | Yes | Yes | No | Yes | No |
| TreeMap | Yes | Yes | Yes | No | No | No |
| Vector | Yes | Yes | No | Yes | Yes | Yes |
| Hashtable | No | Yes | Yes | No | No | Yes |
| Properties | No | Yes | Yes | No | No | Yes |
| Stack | Yes | No | No | Yes | Yes | Yes |
| CopyOnWriteArrayList | Yes | Yes | No | Yes | Yes | Yes |
| ConcurrentHashMap | No | Yes | Yes | No | No | Yes |
| CopyOnWriteArraySet | No | No | No | No | Yes | Yes |

|  |  |
| --- | --- |
| **Sr.No.** | **Class & Description** |
| 1 | **AbstractCollection**  Implements most of the Collection interface. |
| 2 | **AbstractList**  Extends AbstractCollection and implements most of the List interface. |
| 3 | **AbstractSequentialList**  Extends AbstractList for use by a collection that uses sequential rather than random access of its elements. |
| 4 | [**LinkedList**](https://www.tutorialspoint.com/java/java_linkedlist_class.htm)  Implements a linked list by extending AbstractSequentialList. |
| 5 | [**ArrayList**](https://www.tutorialspoint.com/java/java_arraylist_class.htm)  Implements a dynamic array by extending AbstractList. |
| 6 | **AbstractSet**  Extends AbstractCollection and implements most of the Set interface. |
| 7 | [**HashSet**](https://www.tutorialspoint.com/java/java_hashset_class.htm)  Extends AbstractSet for use with a hash table. |
| 8 | [**LinkedHashSet**](https://www.tutorialspoint.com/java/java_linkedhashset_class.htm)  Extends HashSet to allow insertion-order iterations. |
| 9 | [**TreeSet**](https://www.tutorialspoint.com/java/java_treeset_class.htm)  Implements a set stored in a tree. Extends AbstractSet. |
| 10 | **AbstractMap**  Implements most of the Map interface. |
| 11 | [**HashMap**](https://www.tutorialspoint.com/java/java_hashmap_class.htm)  Extends AbstractMap to use a hash table. |
| 12 | [**TreeMap**](https://www.tutorialspoint.com/java/java_treemap_class.htm)  Extends AbstractMap to use a tree. |
| 13 | [**WeakHashMap**](https://www.tutorialspoint.com/java/java_weakhashmap_class.htm)  Extends AbstractMap to use a hash table with weak keys. |
| 14 | [**LinkedHashMap**](https://www.tutorialspoint.com/java/java_linkedhashmap_class.htm)  Extends HashMap to allow insertion-order iterations. |
| 15 | [**IdentityHashMap**](https://www.tutorialspoint.com/java/java_identityhashmap_class.htm)  Extends AbstractMap and uses reference equality when comparing documents. |