Collection Frameworks

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 you perform on a data such as searching, sorting, insertion, manipulation, and deletion.

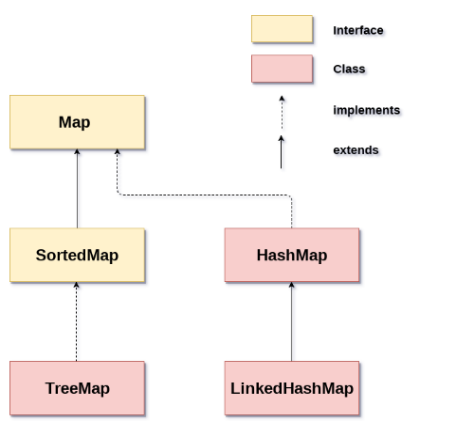
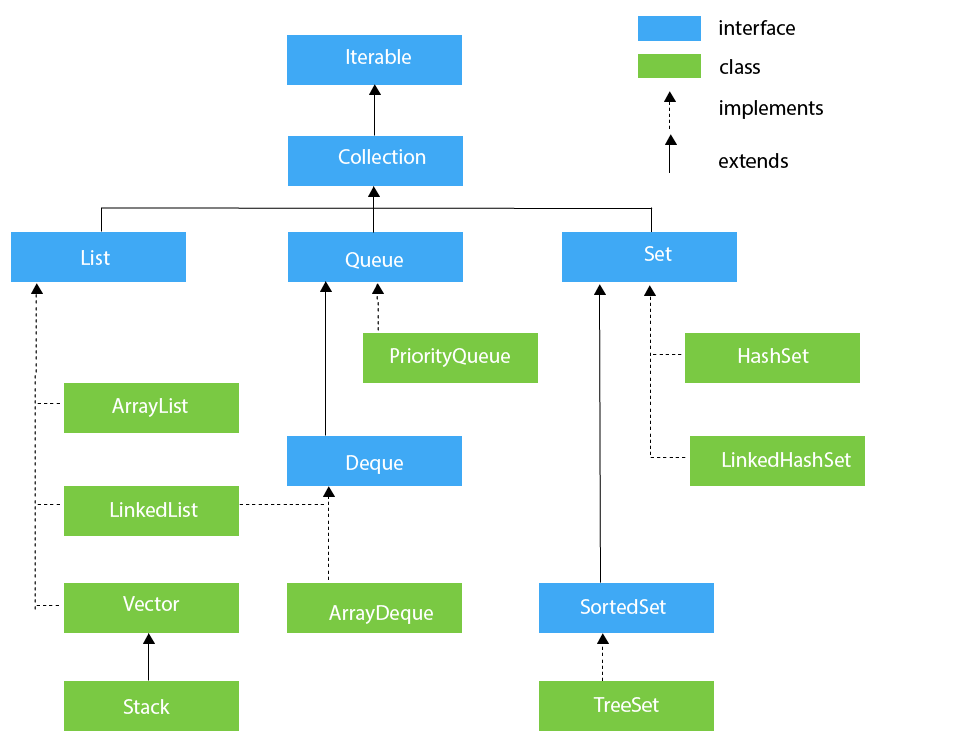
Java Collection means a single unit of objects. Java Collection framework provides many interfaces (Set, List, Queue, Deque) and classes ([ArrayList](https://www.javatpoint.com/java-arraylist)

, Vector, [LinkedList](https://www.javatpoint.com/java-linkedlist)

, [PriorityQueue](https://www.javatpoint.com/java-priorityqueue)

, HashSet, LinkedHashSet, TreeSet).

Collection frameworks Hierarchy



List Interface

List interface is the child interface of Collection interface. It inhibits a list type data structure in which we can store the ordered collection of objects. It can have duplicate values. List interface is implemented by the classes ArrayList, LinkedList, Vector, and Stack.

1. ArrayList
2. The ArrayList class implements the List interface.
3. It uses a dynamic array to store the duplicate element of different data types.
4. The ArrayList class maintains the insertion order and is non-synchronized.
5. The elements stored in the ArrayList class can be randomly accessed.
6. LinkedList
7. LinkedList implements the List interface.
8. It uses a doubly linked list internally to store the elements.
9. It can store the duplicate elements.
10. It maintains the insertion order and is not synchronized.
11. In LinkedList, the manipulation is fast because no shifting is required
12. Vector
13. Vector uses a dynamic array to store the data elements.
14. It is similar to ArrayList. However, it is synchronized and contains many methods that are not the part of Collection framework.
15. Stack
16. The stack is the subclass of Vector.
17. Stack implements the last-in-first-out (LIFO) data structure.
18. The stack contains all of the methods of Vector class and also provides its methods like boolean push(), boolean peek(), boolean push(object o), which defines its properties.

Que Interface

Queue interface maintains the first-in-first-out order. It can be defined as an ordered list that is used to hold the elements which are about to be processed. There are various classes like PriorityQueue, Deque(Interface), and ArrayDeque which implements the Queue interface

1. PriorityQue
2. The PriorityQueue class implements the Queue interface.
3. It holds the elements or objects which are to be processed by their priorities.
4. PriorityQueue doesn't allow null values to be stored in the queue.
5. Deque Interface (DoubleEnded Que)
6. Deque interface extends the Queue interface.
7. In Deque, we can remove and add the elements from both the side.
8. Deque stands for a double-ended queue which enables us to perform the operations at both the ends.
9. ArrayDeque
10. ArrayDeque class implements the Deque interface.
11. It facilitates us to use the Deque. Unlike queue, we can add or delete the elements from both the ends.
12. ArrayDeque is faster than ArrayList and Stack and has no capacity restrictions.

Set Interface

Set Interface in Java is present in java.util package. It extends the Collection interface. It represents the unordered set of elements which doesn't allow us to store the duplicate items. We can store at most one null value in Set. Set is implemented by HashSet, LinkedHashSet, and TreeSet.

1. HashSet
2. HashSet class implements Set Interface.
3. It represents the collection that uses a hash table for storage.
4. Hashing is used to store the elements in the HashSet.
5. It contains unique items.
6. LinkedHashSet
7. LinkedHashSet class represents the LinkedList implementation of Set Interface.
8. It extends the HashSet class and implements Set interface.
9. Like HashSet, It also contains unique elements.
10. It maintains the insertion order and permits null elements.
11. Sorted Set Interface
12. SortedSet is the alternate of Set interface that provides a total ordering on its elements.
13. The elements of the SortedSet are arranged in the increasing (ascending) order. The SortedSet provides the additional methods that inhibit the natural ordering of the elements.
14. The SortedSet can be instantiated as

SortedSet<data-type> set = new TreeSet();

1. TreeSet
2. TreeSet class implements the Set interface that uses a tree for storage.
3. Like HashSet, TreeSet also contains unique elements. However, the access and retrieval time of TreeSet is quite fast.
4. The elements in TreeSet stored in ascending order

Map Interface

There are two interfaces for implementing Map in java: Map and SortedMap, and three classes: HashMap, LinkedHashMap, and TreeMap. The hierarchy of Java Map is given below

A Map doesn't allow duplicate keys, but you can have duplicate values. HashMap and LinkedHashMap allow null keys and values, but TreeMap doesn't allow any null key or value

***Map.Entry Interface :*** *Entry is the subinterface of Map. So we will be accessed it by Map.Entry name. It returns a collection-view of the map, whose elements are of this class. It provides methods to get key and value.*

1. HashMap
2. Java HashMap contains values based on the key.
3. Java HashMap contains only unique keys.
4. Java HashMap may have one null key and multiple null values.
5. Java HashMap is non synchronized.
6. Java HashMap maintains no order.
7. The initial default capacity of Java HashMap class is 16 with a load factor of 0.75.

# Working of HashMap in Java

Hashing -> is the process of converting an object into an integer value. The integer value helps in indexing and faster searches.

1. LinkedHashMap
2. Java LinkedHashMap contains values based on the key.
3. Java LinkedHashMap contains unique elements.
4. Java LinkedHashMap may have one null key and multiple null values.
5. Java LinkedHashMap is non synchronized.
6. Java LinkedHashMap maintains insertion order.
7. The initial default capacity of Java HashMap class is 16 with a load factor of 0.75.
8. TreeMap
   1. Java TreeMap contains values based on the key.
   2. It implements the NavigableMap interface and extends AbstractMap class.
   3. Java TreeMap contains only unique elements.
   4. Java TreeMap cannot have a null key but can have multiple null values.
   5. Java TreeMap is non synchronized.
   6. Java TreeMap maintains ascending order.

Hashtable -