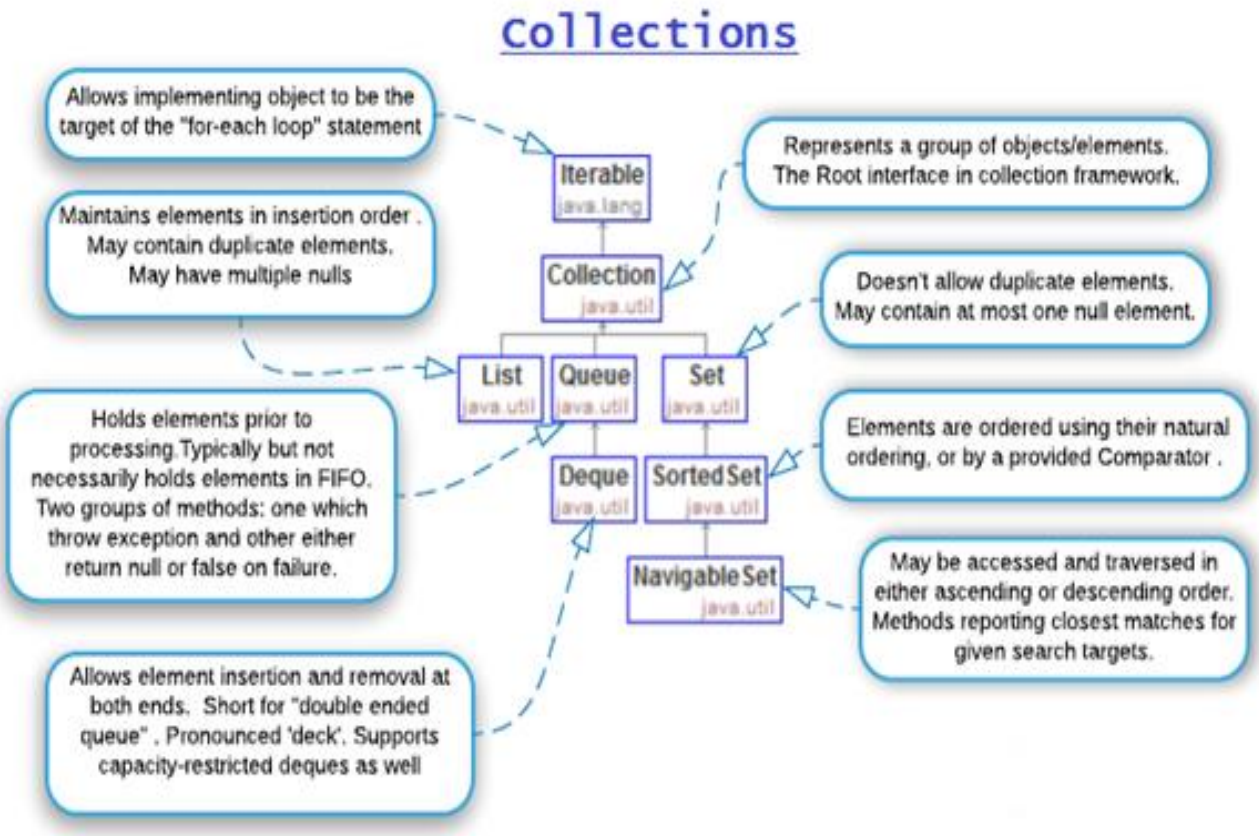
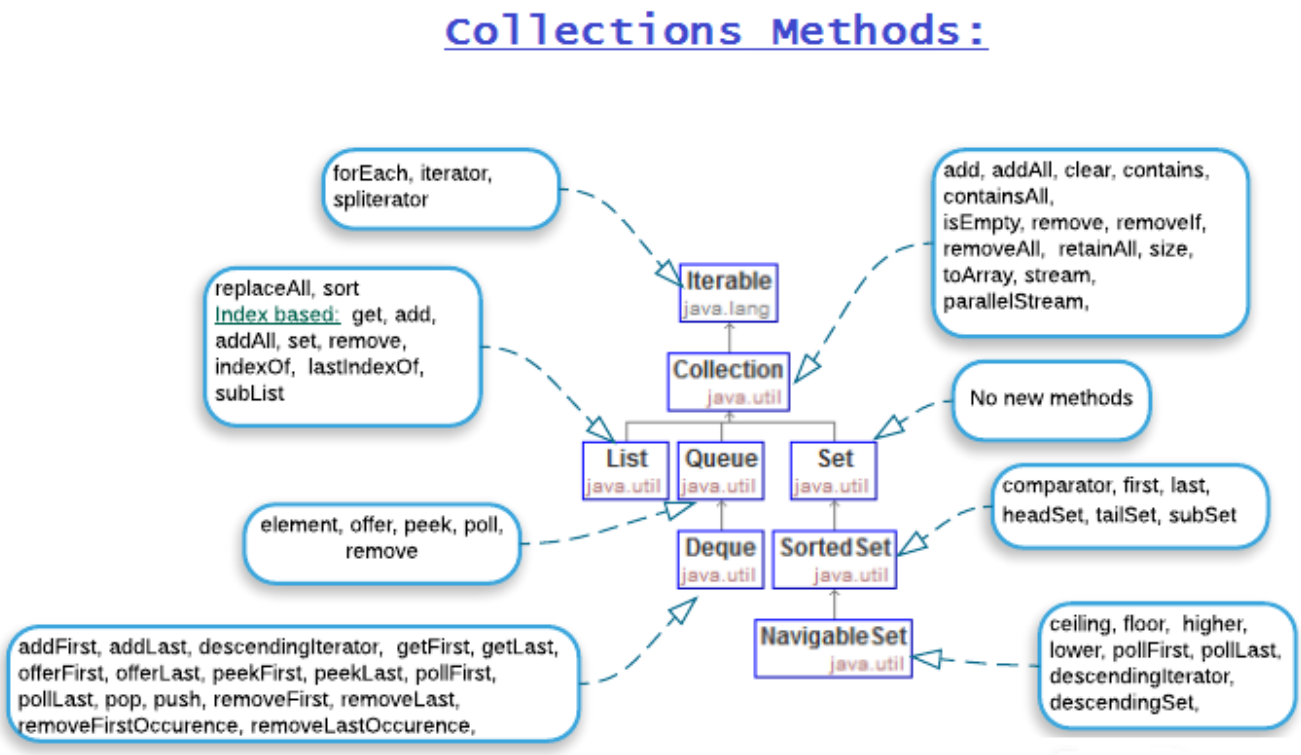
Java - Collection Interfaces and Implementations

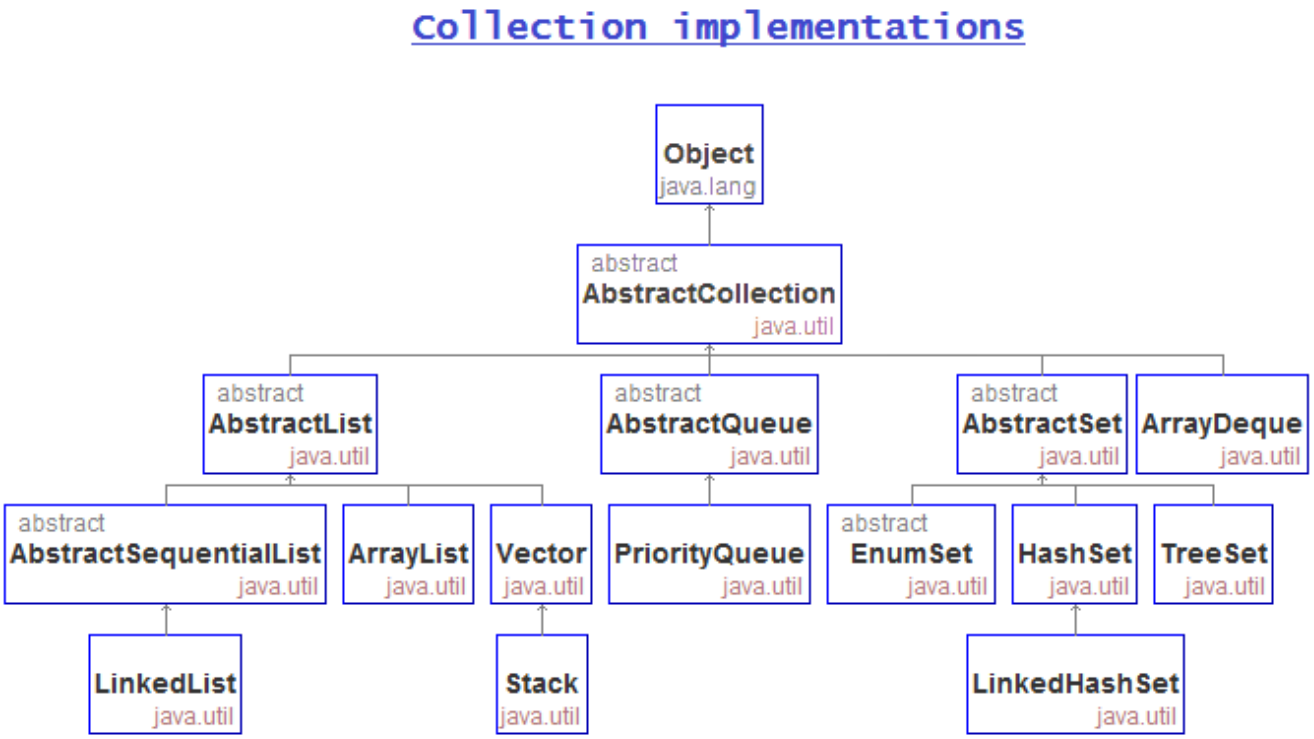
COLLECTION INTERFACES -



COLLECTION OPERATIONS :



COLLECTION IMPLEMENTATIONS -

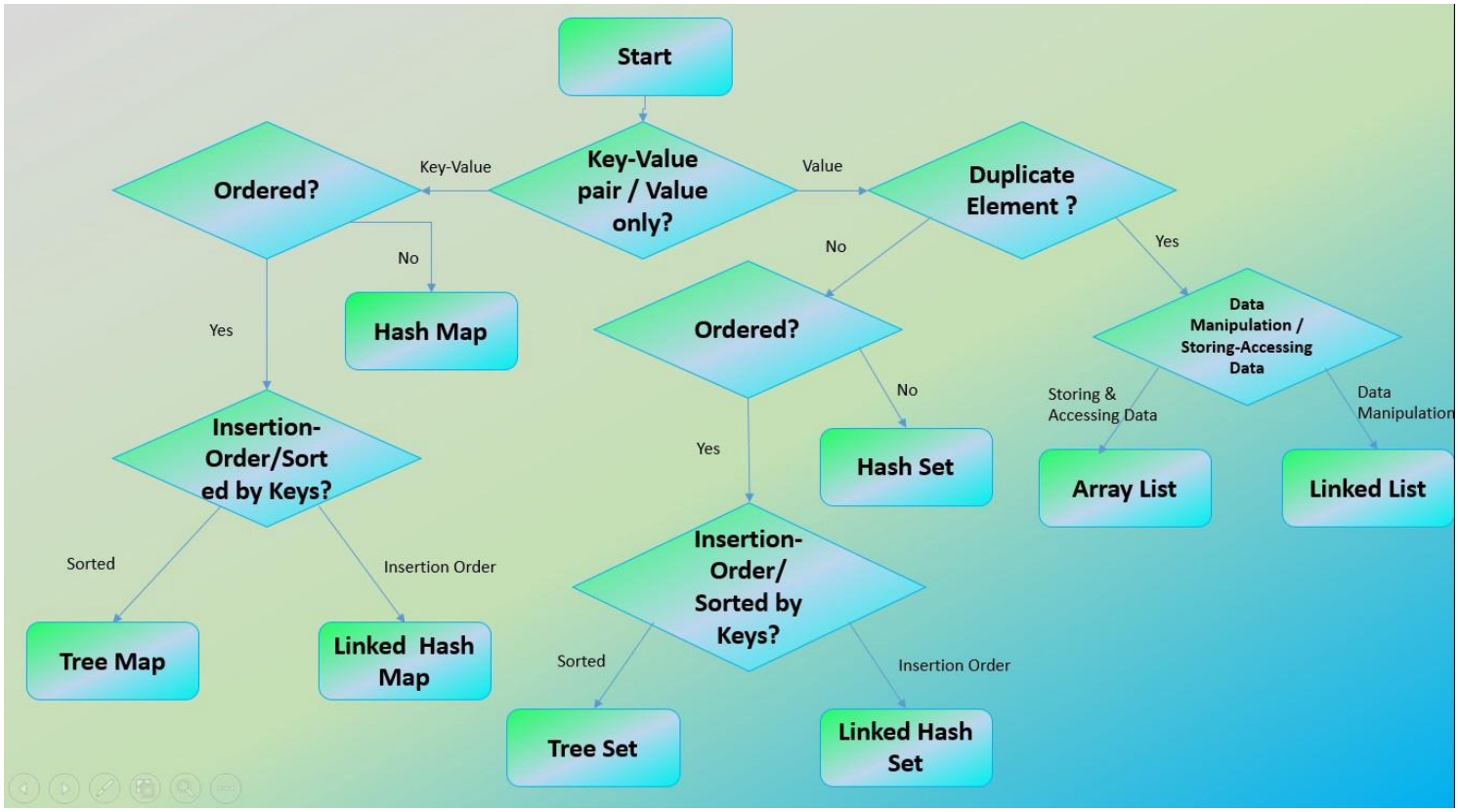


Impl	ADT	Data Structure	Performance (Big O notation)
ArrayList (sync)	List	Array of objects. A new array is created and populated whenever elements are added beyond the current length (capacity) of the underlying array.	**add(E element)** **method:** O(1) amortized. That is, adding n elements within capacity: constant time O(1). Adding an element beyond capacity: O(n) times. It's better to specify initial capacity at construction if known. **remove(int index)**: O(n - index), removing last is O(1). **All other operations including get(int index)** run in linear time O(1) . The constant factor of O(1) is low compared to that for the LinkedList implementation.

LinkedList (sync)	List, Deque	Doubly-linked list. Each element has memory addresses of the previous and next item used internally.	**get(int index), remove(int index)**: O(n) **add(E element) and others**: Constant time O(1).
Vector (sync) (Legacy)	List	Array of objects. Similar to ArrayList	Similar to ArrayList but slower because of synchronization.
Stack extends Vector (sync) (Legacy)	List	Array of objects. LIFO (Last in first out). It provides addition methods **empty(), peek(), pop(), push(E e) and search(Object o)**	Similar to Vector/ArrayList but slower because of synchronisation.
HashSet (sync)	Set	Backed by HashMap (a Hash table data structure). Elements of the set are populated as key of the HashMap. Allows at most one null.	**add, remove, contains, size**: O(1) **Iteration**: O(n + capacity). Better don't set initial capacity (size of backing hasMap) too high or load factor too low if iteration is frequently used.
LinkedHashSet (sync)	Set	Backed by LinkedHashMap where elements of this LinkedHashSet are populated as key of the Map. Maintains elements in insertion order. Allows at most one null.	**add, remove, contains, size**: O(1) **Iteration**: O(n), slightly slow that of HashSet, due to maintaining the linked list.
TreeSet (sync)	NavigableSet	Backed by TreeMap (a red-black tree data structure). The elements of this set are populated as key of the Map. Doesn't permit null.	**add, remove, contains**: O(log n) **Iteration**: O(n) slower than HashSet.
EnumSet (sync)	Set	Bit vectors All of the elements must come from a single enum type.	**All methods**: O(1). Very efficient
PriorityQueue (sync)	Queue	Binary Heap Unbounded Elements are ordered to their natural ordering or by a provided Comparator.	**offer, poll, remove() and add**: O(log n) **remove(Object), contains(Object)** O(n) **peek, element, and size**: O(1)
ArrayDeque (sync)	Dequeue	Resizable-array (similar to ArrayList). Unbounded Nulls not permitted.	**remove, removeFirstOccurrence, removeLastOccurrence, contains, iterator.remove(), and the bulk**

			operations: O(n) **All other operations** O(1) amortized

Making a Decision in choosing a Collection :



DIFFERENCES :

	Insertion Order	Duplicate Elements	Sorting	Null Elements	Synchronized / Thread Safe
LIST					
Array List	Yes	Yes	No	Yes	No
Linked List	Yes	Yes	No	Yes	No
SET					
Hash Set	No	No	No	Yes	No
Linked Hash Set	Yes	No	No	Yes	No
Tree Set	No	No	Yes (Ascending)	No	No
MAP					
Hash Map	No	Unique Key & Duplicate Values	No	1 Null Key & Multiple Null Values	No
Linked Hash Map	Yes	Unique Key & Duplicate Values	No	1 Null Key & Multiple Null Values	No
Tree Map	No	Unique Key & Duplicate Values	Yes (Ascending Order of Keys)	No Null Key & Multiple Null Values	No