|  |  |
| --- | --- |
| ***Array*** | ***Array list*** |
| *Single (or)Multi-dimension* | *Single dimension* |
| *Here we can’t revise the length of the array once constructed.* | *In ArrayList, we can revise the length of the array.* |
| *An array can hold primitives and objects both in Java.* | *ArrayList can only hold objects, not primitives.* |
| *Through the length keyword, we can determine the total size of an array.* | *Through the size() method, we can determine the size of an ArrayList.* |
| *Array is static.* | *ArrayList is dynamic and can be modified the size whenever needed.* |
| *It is faster than ArrayList due to its static behaviour.* | *It is slower as compared to the Array due to its dynamic behaviour.* |
| *For and for Each loop is used for iteration* | *Iterator is used for iteration* |

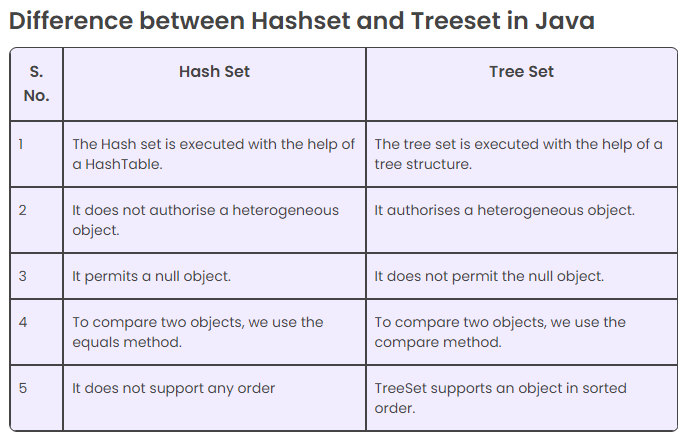
|  |  |
| --- | --- |
| ***Array list*** | ***Linked list*** |
| *ArrayList internally uses a****dynamic array****to store the elements.* | *LinkedList internally uses a****doubly linked list****to store the elements* |
| *Manipulation with ArrayList is****slow****because it internally uses an array. If any element is removed from the array, all the other elements are shifted in memory.* | *Manipulation with LinkedList is****faster****than ArrayList because it uses a doubly linked list, so no bit shifting is required in memory.* |
| *An ArrayList class can****act as a list****only because it implements List only.* | *LinkedList class can****act as a list and queue****both because it implements List and Deque interfaces.* |
| *ArrayList is****better for storing and accessing****data.* | *LinkedList is****better for manipulating****data.* |
| *Array list consumes less memory* | *Linkedlist consumes more memory tha arraylist because it stores next and previous reference* |

|  |  |
| --- | --- |
| ***Iterator*** | ***ListIterator*** |
| *It can traverse elements present in a Collection only in the forward direction.* | *It can traverse elements present in a Collection both in forward and backward directions.* |
| *It is used to traverse Map, List, and Set.* | *It is used only to traverse List. That is it can’t traverse Map and Set.* |
| *Iterator cannot add elements to a Collection..* | *ListIterator can add elements to the Collection.* |
| *An Iterator cannot modify the elements in the Collection.* | *A ListIterator can modify the elements in a Collection using a set().* |
| *An Iterator has no method to obtain an index of the element in a given Collection.* | *ListIterator can be used to obtain the index of an element in the Collection.* |
| *It has methods like hasNext(), next(), remove()* | *It has methods like add(E e), hasNext(), hasPrevious(), next(), nextIndex(), previous(), previousIndex(), remove(), set(E e).* |

|  |  |
| --- | --- |
| ***Vector*** | ***ArrayList*** |
| *Legacy class* | *Part of Collection framework* |
| *Synchronized* | *Not Synchronized* |
| *Performance is Slower* | *Performance is Faster* |
| *Single Thread – Thread Safe* | *Multi Thread – Not Thread safe* |
| *Enumerator is used for iteration* | *Iterator is used for iteration* |

|  |  |
| --- | --- |
| ***Vector*** | ***Array*** |
| *Dynamic in nature* | *Static in nature* |
| *Synchronized* | *Not Synchronized* |
| *Performance is Slower* | *Performance is Faster* |
| *Single Thread – Thread Safe* | *Multi Thread – Not Thread safe* |
| *Enumerator is used for iteration* | *For and for Each is used for iteration* |
| *Vector is dynamic and can be modified the size whenever needed.* | *Through the length keyword, we can determine the total size of an array.* |
| *It can only hold objects, not primitives.* | *An array can hold primitives and objects both in Java.* |

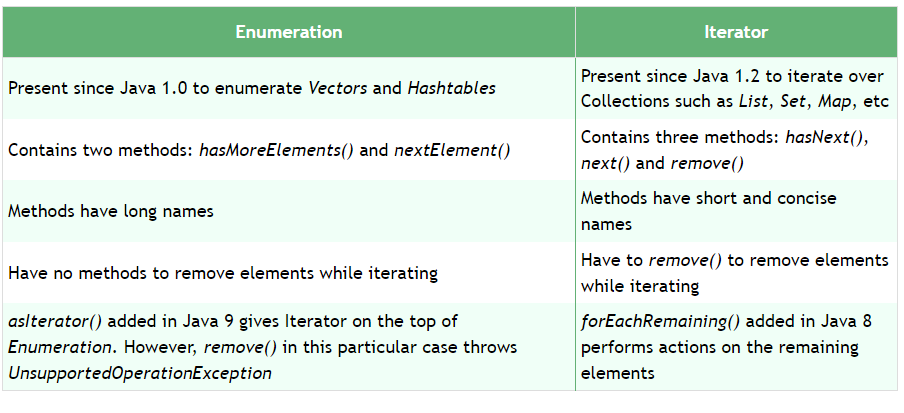
|  |  |
| --- | --- |
| ***HashSet*** | ***LinkedHashSet*** |
| *It uses a Hashtable to store the elements.* | *It uses a HashTable and doubly linked list to store and maintain the insertion order of the elements.* |
| *It does not provide any insertion order. We can not predict the order of elements.* | *It provides an insertion order; we can predict the order of elements.* |
| *It requires less memory.* | *It requires more memory than HashSet.* |
| *It provides slightly faster performance than LinkedHashSet* | *It provides low performance than HashSet* |

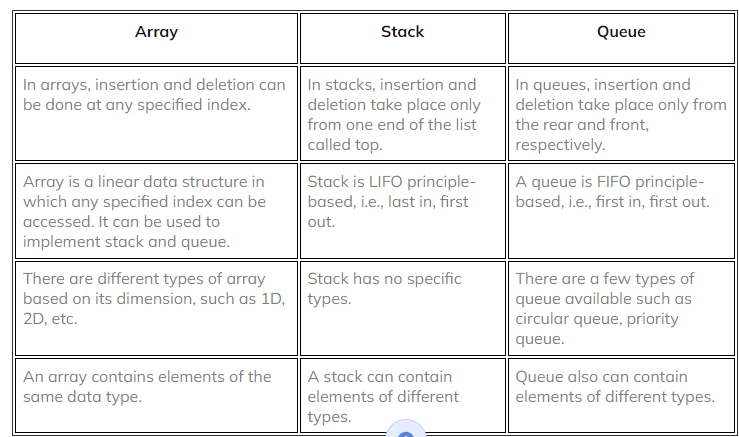


|  |  |  |
| --- | --- | --- |
| ***List*** | ***Set*** | ***Map*** |
| *The list interface allows duplicate elements* | *Set does not allow duplicate elements.* | *The map does not allow duplicate elements* |
| *The list maintains insertion order.* | *Set do not maintain any insertion order.* | *The map also does not maintain any insertion order.* |
| *We can add any number of null values.* | *But in set almost only one null value.* | *The map allows a single null key at most and any number of null values.* |
| *List implementation classes are*[*Array List*](https://www.geeksforgeeks.org/arraylist-in-java/)*, [LinkedList](https://www.geeksforgeeks.org/linked-list-in-java/).* | *Set implementation classes are [HashSet](https://www.geeksforgeeks.org/hashset-in-java/), [LinkedHashSet](https://www.geeksforgeeks.org/linkedhashset-in-java-with-examples/), and [TreeSet](https://www.geeksforgeeks.org/treeset-in-java-with-examples/).* | *Map implementation classes are [HashMap](https://www.geeksforgeeks.org/java-util-hashmap-in-java/), [HashTable](https://www.geeksforgeeks.org/hashtable-in-java/), [TreeMap](https://www.geeksforgeeks.org/treemap-in-java/), [ConcurrentHashMap](https://www.geeksforgeeks.org/concurrenthashmap-in-java/), and [LinkedHashMap](https://www.geeksforgeeks.org/linkedhashmap-class-java-examples/).* |
| *The list provides get() method to get the element at a specified index.* | *Set does not provide get method to get the elements at a specified index* | *The map does not  provide get method to get the elements at a specified index* |
| *If you need to access the elements frequently by using the index then we can use the list* | *If you want to create a collection of unique elements then we can use set* | *If you want to store the data in the form of key/value pair then we can use the map.* |
| *To traverse the list elements by using Listlterator.* | *Iterator can be used traverse the set elements* | *Through keyset, value, and entry set.* |

|  |  |
| --- | --- |
| ***Queue*** | ***Deque*** |
| *Single directional Queue* | *Bi-Directional Queue* |
| *It supports methods for adding,removing and inspecting data from one end of the queue. Eg-add,poll* | *It supports methods for adding,removing and inspecting data from both end of the queue. Eg-add,addFirst,addLast,poll,pollFirst,pollLast* |

***Additional info***

******

******