Difference Between LinkedList and ArrayList

An ArrayList is a resizable array that grows as additional elements are added. A LinkedList is a doubly-linked list/queue implementation. This means that ArrayList internally contains an array of values and a counter variable to know the current size at any point. If an element is added, the size is increased. If an element is removed, the size is decreased. LinkedList doesn't have an array but a double-ended queue of mutually-connected elements instead. The first element *points to* the second one, which *points to* the third one, and so forth. Since this is a *doubly*-linked list, each element also points to its predecessor. The fifth element, for example, points both to the fourth element and the sixth element. ArrayList contains a single array for data storage. LinkedList needs a custom data structure. This custom data structure is a Node. It is a small internal class that serves as a wrapper around each element.

Here are some notable differences between the two:

1. **ArrayList internally uses a dynamic array to store the elements while LinkedList internally uses a doubly linked list to store the elements.**
2. **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 while Manipulation with LinkedList is faster than ArrayList because it uses a doubly linked list, so no bit shifting is required in memory.**
3. **An ArrayList class can act as a list only because it implements List only while a LinkedList class can act as a list and queue both because it implements List and Deque interfaces.**
4. **The memory location for the elements of an ArrayList is contiguous while the location for the elements of a linked list is not contagious.**

5. **Generally, when an ArrayList is initialized, a default capacity of 10 is assigned to the ArrayList while there is no case of default capacity in a LinkedList. In LinkedList, an empty list is created when a LinkedList is initialized.**

ArrayList provides constant time for the search operation, so it is better to use ArrayList if searching is a more frequent operation than add and remove operation. The LinkedList provides constant time for add and removes operations. So it is better to use LinkedList for manipulation. ArrayList has O(1) time complexity to access elements via the get and set methods, and LinkedList has O(n/2) time complexity to access the elements.

The performance difference between ArrayList and LinkedList for various operations

1) Search: The ArrayList search operation is pretty fast compared to the LinkedList search operation. get(int index) in ArrayList gives the performance of O(1) while LinkedList performance is O(n). ArrayList maintains index based system for its elements as it uses an array data structure implicitly which makes it faster to search an element in the list. On the other side, LinkedList implements a doubly linked list which requires the traversal through all the elements for searching an element.

2) Deletion: LinkedList remove operation gives O(1) performance while ArrayList gives variable performance: O(n) in worst case (while removing first element) and O(1) in best case (While removing last element). LinkedList element deletion is faster compared to ArrayList, LinkedList's each element maintains two pointers (addresses) which point to both neighbor elements in the list. Hence removal only requires

a change in the pointer location in the two neighbor nodes (elements) of the node which is going to be removed. While In ArrayList all the elements need to be shifted to fill out the space created by the removed element.

3) Inserts Performance: LinkedList add method gives O(1) performance while ArrayList gives O(n) in worst case. This is because every time you add an element, Java ensures that it can fit the element so it grows the ArrayList. If the ArrayList grows faster, there will be a lot of array copying taking place. In the worst case, the array must be resized and copied.

Note: As explained above the insert and remove operations give a good performance (O(1)) in LinkedList compared to ArrayList(O(n)). Hence if there are frequent addition and deletion in an application then LinkedList is the best choice. Search (get method) operations are fast in Arraylist (O(1)) but not in LinkedList (O(n)) so If there are fewer add and remove operations and more search operations requirements, ArrayList would be your best bet.