**Java Collections Theory**

Array & ArrayList

1.Arrays have a fixed size, determined at the time of creation, and cannot be resized dynamically. In contrast, ArrayLists can dynamically grow or shrink in Arrays have a fixed size, determined at the time of creation, and cannot be resized dynamically. In contrast, ArrayLists can dynamically grow or shrink in size as elements are added or removed.

2. Type: Arrays can store both primitive types (like int, char, etc.) and objects. ArrayLists, on the other hand, can only store objects (reference types) and not primitive types directly. However, autoboxing and unboxing in Java allow you to store primitive types in an ArrayList indirectly by using their corresponding wrapper classes (e.g., Integer, Character, etc.).

3. Methods: Arrays in Java have limited methods and functionality compared to ArrayLists. ArrayLists offer a rich set of methods such as add, remove, get, size, isEmpty, contains, indexOf, and more, which provide convenient operations for working with collections of elements.

4. Memory management: Arrays are a fixed allocation of memory that is determined at compile-time. In contrast, ArrayLists automatically manage memory allocation and resizing internally. As elements are added or removed from an ArrayList. It changes its size in the form of n+n/2+1 .

Iterator & ListIterator

Iterator interface allows to travel in the foreward direction only. we have various methods like add(), next(), hasNext() , next() etc. Whereas ListIterator allows more flexible traversal of the List. Which means we can even make the iterator move in reverse order with the help of previous (), nextIndex (), previousIndex(), we can have indexbased access in ListIterator.

Unmodifiable List in Java

An unmodifiable list in Java refers to a list that cannot be modified after it is created. It is a read-only view of an underlying list, which means that any attempt to modify the unmodifiable list will result in an exception being thrown. In Java, you can create an unmodifiable list using the Collections.unmodifiableList()

ArrayList and LinkedList

ArrayList:

1. ArrayList internally uses a dynamic array to store the elements.
2. An ArrayList class can act as a list only because it implements List only.
3. 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.
4. ArrayList is better for storing and accessing data.
5. The memory location for the elements of an ArrayList is contiguous.
6. Generally, when an ArrayList is initialized, a default capacity of 10 is assigned to the ArrayList.
7. ArrayList is a resizable array.

LinkedList:

1. LinkedList internally uses a doubly linked list to store the elements.
2. LinkedList class can act as a list and queue both because it implements List and Deque interfaces.
3. Manipulation with LinkedList is faster than ArrayList because it uses a doubly linked list, so no bit shifting is required in memory.
4. LinkedList is better for manipulating data.
5. The location for the elements of a linked list is not contagious.
6. There is no case of default capacity in a LinkedList. In LinkedList, an empty list is created when a LinkedList is initialized.
7. LinkedList implements the doubly linked list of the list interface.

ArrayList and Vector

ArrayList:

1. ArrayList is not synchronized.
2. ArrayList increments 50% of the current array size if the number of elements exceeds its capacity.
3. ArrayList is not a legacy class. It is introduced in JDK 1.2.
4. ArrayList is fast because it is non-synchronized.
5. ArrayList uses the Iterator interface to traverse the elements.
6. ArrayList performance is high.
7. Multiple threads is allowed.

Vector:

1. Vector is synchronized.
2. Vector increments 100% means doubles the array size if the total number of elements exceeds its capacity.
3. Vector is a legacy class.
4. Vector is slow because it is synchronized, i.e., in a multithreading environment, it holds the other threads in a runnable or non-runnable state until the current thread releases the lock of the object.
5. A Vector can use the Iterator interface or Enumeration interface to traverse the elements.
6. Vector performance is low.
7. only one threads are allowed.