When to use ArrayList vs LinkedList in Java? [Answered]

When to use ArrayList or LinkedList in Java is one of the most popular Java interview questions and also asked as a difference between ArrayList and LinkedList. Earlier, I have shared c**ommon Java collections interview questions** and in this article, I will explain the difference between them. ArrayList and LinkedList are two popular concrete implementations of the List interface from Java's popular Collection framework. Being List implementation both `ArrayList` and `LinkedList` are ordered, the index-based and allows duplicate. Despite being from the same type of hierarchy there are a lot of differences between these two classes which makes them popular among Java interviewers.

The main difference between ArrayList vs LinkedList is that the former is backed by an array while the latter is based upon the linked list data structure, which makes the performance of `add()`, `remove()`, `contains()`, and `iterator()` different for both ArrayList and LinkedList.

The difference between ArrayList and LinkedList is also an important Java collection interview question, as much popular as Vector vs ArrayList or HashMap vs HashSet in Java. Sometimes this is also asked as for when to use LinkedList and when to use ArrayList in Java.

In this Java collection tutorial, we will compare LinkedList vs ArrayList on various parameters which will help us to decide when to use ArrayList over LinkedList in Java.

Btw, we will not focus on the array and linked list data structure much, which is subject to data structure and algorithm, we'll only focus on the Java implementations of these data structures which are ArrayList and LinkedList.

If you want to learn more about the array and linked list data structure itself, I suggest you check **Data Structures and Algorithms: Deep Dive Using Java** course by Tim Buchalaka on Udemy.

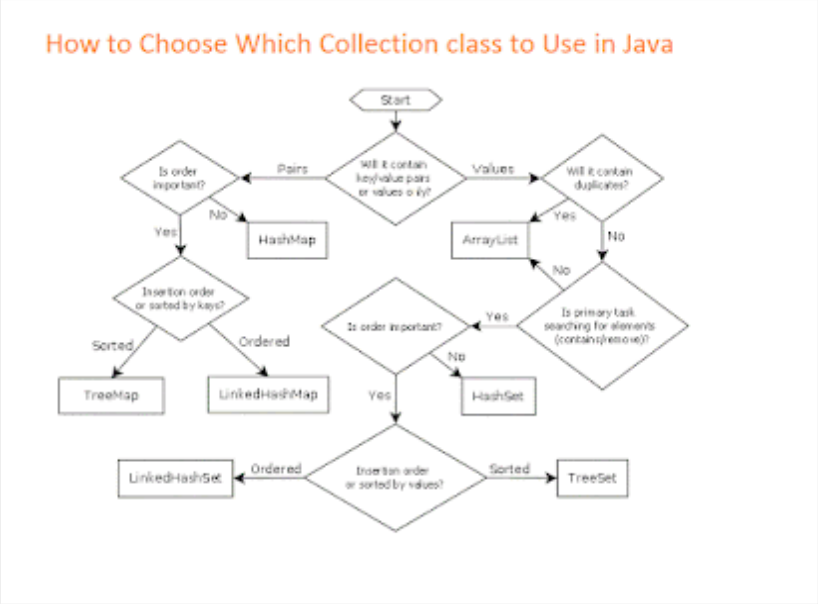
It explains essential data structure in Java programming language and most importantly teaches you when to use which data structure, a good refresher for those who are preparing for coding interviews too.

When to use ArrayList vs LinkedList in Java

Before comparing differences between ArrayList and LinkedList, let's see What is common between ArrayList and LinkedList in Java :

1) Both ArrayList and LinkedList are an implementation of the List interface, which means you can pass either ArrayList or LinkedList if a method accepts the `java.util.List` interface.

Btw, if you are new to Java's collections framework then I suggest you first go through **Java Fundamentals: Collections** by Richart Warburton. It's an online Java course on Pluralsight, which you can avail of free by signing their 10-day free trial. IMHO, it's worth going through that course to learn Java collections in the right way.



2) Both ArrayList and LinkedList are not synchronized, which means you can not share them between multiple threads without external synchronization. See here to know How to make ArrayList synchronized in Java.

3) ArrayList and LinkedList are ordered collection e.g. they maintain insertion order of elements i.e. the first element will be added to the first position.

4) ArrayList and LinkedList also allow duplicates and null, unlike any other List implementation e.g. Vector.

5) An iterator of both LinkedList and ArrayList are fail-fast which means they will throw `ConcurrentModificationException` if a collection is modified structurally once the Iterator is created. They are different than CopyOnWriteArrayList whose Iterator is fail-safe.

Difference between LinkedList and ArrayList in Java

Now let's see some differences between ArrayList and LinkedList and when to use ArrayList and LinkedList in Java.

1. Underlying Data Structure

The first difference between ArrayList and LinkedList comes with the fact that ArrayList is backed by Array while LinkedList is backed by LinkedList. This will lead to further differences in performance.

2. LinkedList implements Deque

Another difference between ArrayList and LinkedList is that apart from the `List` interface, `LinkedList` also implements the `Deque` interface, which provides first in first out operations for `add()` and `poll()` and several other Deque functions.

Also, LinkedList is implemented as a doubly-linked list and for index-based operation, navigation can happen from either end (see **Complete Java MasterClass**).

3. Adding elements in ArrayList

Adding an element in ArrayList is `O(1)` operation if it doesn't trigger re-size of Array, in

which case it becomes `O(log(n))`, On the other hand, appending an element in LinkedList is O(1) operation, as it doesn't require any navigation.

4. Removing an element from a position

In order to remove an element from a particular index e.g. by calling remove(index), ArrayList performs a copy operation which makes it close to O(n) while LinkedList needs to traverse to that point which also makes it O(n/2), as it can traverse from either direction based upon proximity.

5. Iterating over ArrayList or LinkedList

Iteration is the O(n) operation for both LinkedList and ArrayList where n is a number of an element.

6. Retrieving element from a position

The `get(index)` operation is `O(1)` in ArrayList while its `O(n/2)` in LinkedList, as it needs to traverse till that entry. Though, in Big O notation O(n/2) is just O(n) because we ignore constants there.