

# Collection Framework

1. What is the Collection framework in Java

Ans1. The **Collection Framework** in Java is a unified architecture that provides a set of interfaces and classes for storing, manipulating, and processing groups of objects efficiently. It is part of the `java.util` package and includes various data structures such as lists, sets, queues, and maps.

## Key Components of Java Collection Framework

The Collection Framework consists of:

1. **Interfaces:** Define abstract data types (e.g., `Collection`, `List`, `Set`, `Queue`, `Map`).
  2. **Classes:** Concrete implementations of these interfaces (e.g., `ArrayList`, `HashSet`, `LinkedList`, `HashMap`).
  3. **Algorithms:** Utility methods for sorting, searching, shuffling, etc. (e.g., `Collections.sort()`).
-

## Hierarchy of Java Collections Framework

### 1. Collection Interface (Root)

- **List** (Ordered, allows duplicates)
  - **ArrayList**
  - **LinkedList**
  - **Vector** (Legacy)
  - **Stack**
- **Set** (Unordered, no duplicates)
  - **HashSet**
  - **LinkedHashSet**
  - **TreeSet**
- **Queue** (FIFO, used for queues)
  - **PriorityQueue**
  - **ArrayDeque**

### 2. Map Interface (Key-Value Pairs)

- **HashMap**
- **LinkedHashMap**
- **TreeMap**
- **Hashtable** (Legacy)
- **ConcurrentHashMap** (Thread-safe)

### 2. What is the difference between ArrayList and LinkedList

Ans2. Both **ArrayList** and **LinkedList** are implementation of **List interface in Java**. Both classes are non-synchronized. But there are certain differences as well.

Following are the important differences between ArrayList and LinkedList method.

Sr	Key	ArrayList	LinkedList
.	No		
.			
1	Internal Implementation	ArrayList internally uses a dynamic array to store its elements.	LinkedList uses Doubly Linked List to store its elements.
2	Manipulation	ArrayList is slow as array manipulation is slower.	LinkedList is faster being node based as not much bit shifting required.

3	Implementa tion	ArrayList implements only List.	LinkedList implements List as well as Queue. It can acts as a queue as well.
4	Access	ArrayList is faster in storing and accessing data.	LinkedList is faster in manipulation of data.

Example Code

```
import java.util.*;
```

```
public class ListExample {
```

```
public static void main(String[] args) {
```

```
    // ArrayList Example
```

```
    List<Integer> arrayList = new ArrayList<>();
```

```
    arrayList.add(10);
```

```
    arrayList.add(20);
```

```
    arrayList.add(30);
```

```
    System.out.println("ArrayList: " + arrayList); // [10, 20, 30]
```

```
// LinkedList Example
```

```
List<Integer> linkedList = new LinkedList<>();
```

```
linkedList.add(10);
```

```
linkedList.add(20);
```

```
linkedList.add(30);
```

```
System.out.println("LinkedList: " + linkedList); // [10, 20, 30]
```

```
}
```

```
}
```

3. What is the difference between Iterator and ListIterator

### Ans3. Difference Between Iterator and ListIterator in Java with Example

In Java, iterators provide a way to traverse collections and perform operations on their elements. The Java Collections Framework offers two common iterator types: Iterator and ListIterator. While both iterators serve a similar purpose, they have some important differences that make each suitable for specific scenarios. In this section, we will discuss the distinctions between Iterator and ListIterator in Java with examples to illustrate their usage.

#### Iterator

The Iterator interface is part of the Java Collections Framework and provides a

generic way to traverse collections such as lists, sets, and queues. Here are the key characteristics of an Iterator:

- Forward-only traversal: An Iterator allows sequential access to elements in a collection in a forward direction.
- Read and remove operations: It offers methods like `next()` to retrieve the next element and `remove()` to remove the last element returned by the `next()` method.
- Limited functionality: Unlike `ListIterator`, the `Iterator` interface does not support bidirectional traversal or modification of elements during iteration.

### Example

Let's consider a simple example to demonstrate the usage of an Iterator:

```
List<String> fruits = new ArrayList<>();
fruits.add("Apple");
fruits.add("Banana");
fruits.add("Orange");
Iterator<String> iterator = fruits.iterator();
while (iterator.hasNext()) {
    String fruit = iterator.next();
}
```

## ListIterator

The `ListIterator` interface extends the `Iterator` interface and provides additional functionality specifically designed for lists. Here are the key characteristics of a `ListIterator`:

**Bidirectional traversal:** Unlike `Iterator`, `ListIterator` allows traversing the list in both forward and backward directions.

**Read, remove, replace, and add operations:** `ListIterator` supports all the operations of `Iterator`, along with methods like `hasPrevious()`, `previous()`, `set()`, and `add()`. These methods enable modifications to the list during iteration.

### Example

Consider the following example that demonstrates the usage of `ListIterator`:



```

List<String> fruits = new ArrayList<>();
fruits.add("Apple");
fruits.add("Banana");
fruits.add("Orange");
ListIterator<String> listIterator = fruits.listIterator();
while (listIterator.hasNext()) {
    String fruit = listIterator.next();
    System.out.println(fruit);
}
while (listIterator.hasPrevious()) {
    String fruit = listIterator.previous();
}

```

4. What is the difference between Iterator and Enumeration

Ans4.

Sr. No.	Key	Iterator	Enumeration
1	Basic	In Iterator, we can read and remove element while traversing element in the collections.	Using Enumeration, we can only read element during traversing element in the collections.
2.	Access	It can be used with any class of the collection framework.	It can be used only with legacy class of the collection framework such as a Vector and HashTable.

3.	Fail-Fast and Fail-Safe	Any changes in the collection, such as removing element from the collection during a thread is iterating collection then it throw concurrent modification exception.	Enumeration is Fail safe in nature. It doesn't throw concurrent modification exception
4.	Limitation	Only forward direction iterating is possible	Remove operations can not be performed using Enumeration.
5.	Methods	It has following methods – *hasNext() *next() *remove()	It has following methods – *hasMoreElements() *nextElement()

5. What is the difference between List and Set?

Ans5: The List and Set both extend the collection interface. However, there are some differences between the two which are listed below

The List can contain duplicate elements whereas Set includes unique items

The List is an ordered collection which maintains the insertion order whereas Set is an unordered collection which does not preserve the insertion order

The List interface contains a single legacy class which is Vector class whereas the Set interface does not have any legacy class

The List interface can allow a number of null values whereas Set interface only allows a single null value.

6. What is the difference between HashSet and TreeSet?

Ans6: Both HashSet and TreeSet are implementations of the Set interface in Java, but they have some differences in terms of their properties and usage

Ordering: HashSet is an unordered collection of elements, while TreeSet is a sorted set of elements based on their natural order or a custom comparator

Duplication: HashSet does not allow duplicate elements, while TreeSet does not allow duplicates as well

Implementation: HashSet is implemented using a hash table, while TreeSet is implemented using a self-balancing binary search tree (Red-Black tree)

Performance: HashSet has constant-time complexity  $O(1)$  for adding, removing, and testing the existence of an element, while TreeSet has a logarithmic-time complexity  $O(\log n)$  for these operations due to the self-balancing property

Memory usage: HashSet uses less memory than TreeSet because it only stores the elements, while TreeSet stores additional information for maintaining the order

Iteration: HashSet provides no guarantees regarding the order of iteration, while TreeSet guarantees the elements are iterated in sorted order

Usage: HashSet is suitable when ordering is not important, and fast access and membership tests are needed. TreeSet is suitable when elements need to be sorted or accessed in a specific order.

7. What is the difference between Array and ArrayList?

Ans7: Both arrays and ArrayLists are used to store collections of elements in Java, but they have some differences in terms of their properties and usage

**Type:** Arrays can store elements of primitive data types as well as objects, while ArrayList can only store objects

**Size:** The size of an array is fixed once it is created, while the size of an ArrayList can be dynamically increased or decreased by adding or removing elements

**Mutability:** Arrays are mutable, meaning that you can modify the elements in an array after it has been created. ArrayList is also mutable, but the only way to modify it is by adding, removing or modifying elements

**Performance:** Arrays have better performance than ArrayLists for certain operations, such as accessing elements by index, because they are implemented as a continuous block of memory. ArrayLists, on the other hand, use dynamic memory allocation and are implemented as a dynamic array, which may result in more memory overhead and slower performance for certain operations

**Methods:** Arrays have a limited set of methods compared to ArrayLists, which provides more methods for manipulating the collection, such as adding, removing, and sorting elements.