**Collections in Java**

The Collection in Java is a framework that provides an architecture to store and manipulate the group of objects.

Java Collections can achieve all the operations that you perform on a data such as searching, sorting, insertion, manipulation, and deletion.

Java Collection means a single unit of objects. **Java Collection framework provides many interfaces (Set, List, Queue, Deque) and classes (ArrayList, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet).**

**What is Collection in Java**

A Collection represents a single unit of objects, i.e., a group.

**What is a framework in Java**

-It provides readymade architecture.

- It represents a set of classes and interfaces.

- It is optional.

**What is Collection framework**

The Collection framework represents a unified architecture for storing and manipulating a group of objects.

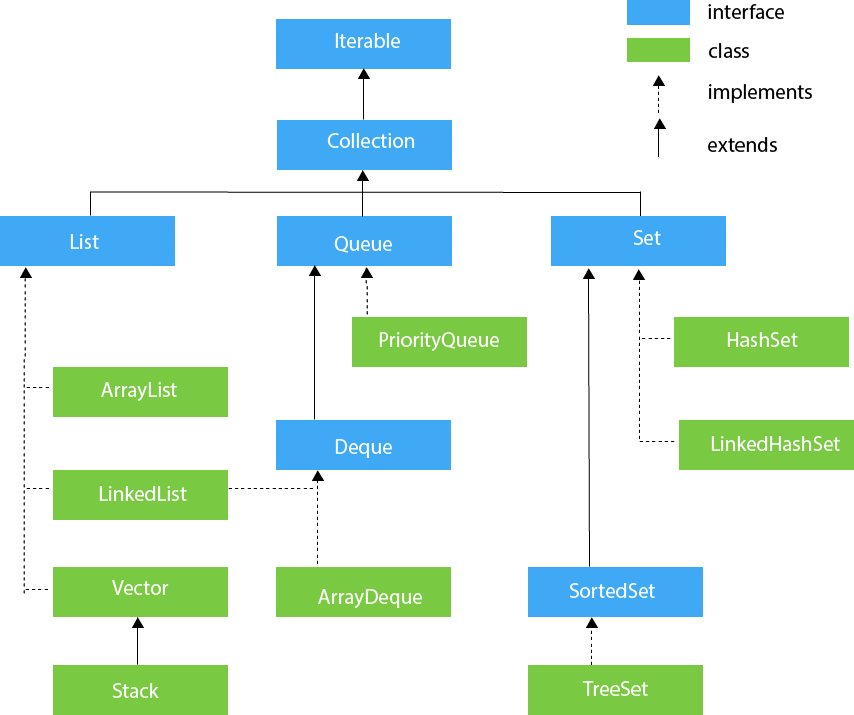
It has:

Interfaces and its implementations, i.e., classes

Algorithm

**Hierarchy of Collection Framework**

Let us see the hierarchy of Collection framework. The java.util package contains all the classes and interfaces for the Collection framework.



**Java Collection Interface**

The Collection interface is the root interface of the collections framework hierarchy.

Java does not provide direct implementations of the Collection interface but provides implementations of its subinterfaces like List, Set, and Queue.

**Note-People often get confused between the collections framework and Collection Interface.**

The Collection interface is the root interface of the collections framework. The framework includes other interfaces as well: Map and Iterator. These interfaces may also have subinterfaces.

**Short diagram to remember all interface altogether**



**Java List interface**

In Java, the List interface is an ordered collection that allows us to store and access elements sequentially.

**List interface is the child interface of Collection interface**. It inhibits a list type data structure in which we can store the ordered collection of objects. It can have duplicate values.

List interface is implemented by the classes ArrayList, LinkedList, Vector, and Stack.

To instantiate the List interface, we must use :

List <data-type> list1= new ArrayList();

List <data-type> list2 = new LinkedList();

List <data-type> list3 = new Vector();

List <data-type> list4 = new Stack();

There are various methods in List interface that can be used to insert, delete, and access the elements from the list.

The classes that implement the List interface are given below.

**Classes that Implement List**

Since List is an interface, we cannot create objects from it.

In order to use functionalities of the List interface, we can use these classes:

**ArrayList**

**LinkedList**

**Vector**

**Stack**

**Methods of List**

The List interface includes all the methods of the Collection interface. Its because Collection is a super interface of List.

Some of the commonly used methods of the Collection interface that's also available in the List interface are:

**add()** - adds an element to a list

**addAll()** - adds all elements of one list to another

**get()** - helps to randomly access elements from lists

**iterator()** - returns iterator object that can be used to sequentially access elements of lists

**set()** - changes elements of lists

**remove()** - removes an element from the list

**removeAll()** - removes all the elements from the list

**clear()** - removes all the elements from the list (more efficient than removeAll())

**size()** - returns the length of lists

**toArray()** - converts a list into an array

**contains()** - returns true if a list contains specified element.

**ArrayList**

The ArrayList class implements the List interface. It uses a dynamic array to store the duplicate element of different data types. The ArrayList class maintains the insertion order and is non-synchronized. The elements stored in the ArrayList class can be randomly accessed. Consider the following example.

**A)display list using Iterator**

import java.util.\*;

**class TestJavaCollection1**

{

public static void main(String args[])

{

ArrayList<String> list=new ArrayList<String>();

//Creating arraylist

list.add("Ravi");//Adding object in arraylist

list.add("Vijay");

list.add("Ravi");

list.add("Ajay");

//Traversing list through Iterator

Iterator itr=list.iterator();

while(itr.hasNext()){

System.out.println(itr.next());

}

}

}

**Output**:

Ravi

Vijay

Ravi

Ajay

**B)display list without iterator**

import java.util.List;

import java.util.ArrayList;

**class Main**

{

public static void main(String[] args)

{

**// Creating list using the ArrayList class**

List<Integer> numbers = new ArrayList<>();

**// Add elements to the list**

numbers.add(1);

numbers.add(2);

numbers.add(3);

System.out.println("List: " + numbers);

**// Access element from the list**

int number = numbers.get(2);

System.out.println("Accessed Element: " + number);

**// Remove element from the list**

int removedNumber = numbers.remove(1);

System.out.println("Removed Element: " + removedNumber);

}

}

**Java Iterator Interface**

The Iterator interface of the Java collections framework allows us to access elements of a collection. It has a subinterface ListIterator.

The Listiterator interface extends the Java Iterator interface.

All the Java collections include an iterator() method. This method returns an instance of iterator used to iterate over elements of collections.

**Methods of Iterator**

The Iterator interface provides 4 methods that can be used to perform various operations on elements of collections.

**hasNext()** - returns true if there exists an element in the collection

**next()** - returns the next element of the collection

**remove()** - removes the last element returned by the next()

**forEachRemaining()** - performs the specified action for each remaining element of the collection