Top of Form

#### 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](https://www.javatpoint.com/java-arraylist), Vector, [LinkedList](https://www.javatpoint.com/java-linkedlist), [PriorityQueue](https://www.javatpoint.com/java-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:

1. Interfaces and its implementations, i.e., classes
2. Algorithm

#### Hierarchy of Collection Framework

Let us see the hierarchy of Collection framework. The **java.util** package contains all the [classes](https://www.javatpoint.com/object-and-class-in-java) and [interfaces](https://www.javatpoint.com/interface-in-java) for the Collection framework.

#### Hierarchy of Java Collection frameworkMethods of

#### ****Collection interface****

There are many methods declared in the Collection interface. They are as follows:

Bottom of Form

Top of Form

#### Iterator interface

Iterator interface provides the facility of iterating the elements in a forward direction only.

#### Methods of Iterator interface

There are only three methods in the Iterator interface. They are:

#### Iterable Interface

The Iterable interface is the root interface for all the collection classes. The Collection interface extends the Iterable interface and therefore all the subclasses of Collection interface also implement the Iterable interface.

It contains only one abstract method. i.e.,

1. Iterator<T> iterator()

It returns the iterator over the elements of type T.

#### Collection Interface

The Collection interface is the interface which is implemented by all the classes in the collection framework. It declares the methods that every collection will have. In other words, we can say that the Collection interface builds the foundation on which the collection framework depends.

Some of the methods of Collection interface are Boolean add ( Object obj), Boolean addAll ( Collection c), void clear(), etc. which are implemented by all the subclasses of Collection interface.

#### List Interface

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 :

1. List <data-type> list1= **new** ArrayList();
2. List <data-type> list2 = **new** LinkedList();
3. List <data-type> list3 = **new** Vector();
4. 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.

Bottom of Form

Top of Form

#### 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.

1. **import** java.util.\*;
2. **class** TestJavaCollection1{
3. **public** **static** **void** main(String args[]){
4. ArrayList<String> list=**new** ArrayList<String>();//Creating arraylist
5. list.add("Ravi");//Adding object in arraylist
6. list.add("Vijay");
7. list.add("Ravi");
8. list.add("Ajay");
9. //Traversing list through Iterator
10. Iterator itr=list.iterator();
11. **while**(itr.hasNext()){
12. System.out.println(itr.next());
13. }
14. }
15. }

Output:

Ravi

Vijay

Ravi

Ajay

#### LinkedList

LinkedList implements the Collection interface. It uses a doubly linked list internally to store the elements. It can store the duplicate elements. It maintains the insertion order and is not synchronized. In LinkedList, the manipulation is fast because no shifting is required.

Consider the following example.

1. **import** java.util.\*;
2. **public** **class** TestJavaCollection2{
3. **public** **static** **void** main(String args[]){
4. LinkedList<String> al=**new** LinkedList<String>();
5. al.add("Ravi");
6. al.add("Vijay");
7. al.add("Ravi");
8. al.add("Ajay");
9. Iterator<String> itr=al.iterator();
10. **while**(itr.hasNext()){
11. System.out.println(itr.next());
12. }
13. }
14. }

Output:

Ravi

Vijay

Ravi

Ajay

Bottom of Form

Top of Form

<https://youtu.be/VvDTro4Shzg>

**Reference Links:**

**ONLINE NOTES LINKS:**

https://www.javatpoint.com/collections-in-java

https://www.geeksforgeeks.org/collections-in-java-2/

https://beginnersbook.com/java-collections-tutorials/

https://www.tutorialspoint.com/java/java\_collections.htm

https://docs.oracle.com/javase/7/docs/api/java/util/Collection.html

https://docs.oracle.com/javase/8/docs/technotes/guides/collections/overview.html

https://www.edureka.co/blog/java-collections/

**VIDEO LINKS:**

https://youtu.be/VvDTro4Shzg

Bottom of Form