# **Collections Framework:**

**What is collections framework?**

A framework is set of classes and interfaces to build a functionality.

Java collections framework provides set of interfaces and classes for storing and manipulating collections. Collections framework contains classes and interfaces in java.util package and java.util.concurrent packages.

Advantages or benefits of Collections framework :

1) High performance

2) Using this framework we can create different types of collections

3) We can create our own collection and we can extend a collection.

4) Reduces programming effort.

5) Increases speed and quality: Collections framework provides high performance, implementations of useful data structures and algorithms.

**What is collection?**

A collection is a container which holds group of objects. Collection provides a way to manage objects easily. Collections manages group of objects as single unit. Examples include list of strings, integers etc.

Here are few basic operations we do on collections:

1) Adding objects to collection.

2) Removing or deleting objects from collection.

3) Retrieving object from collection.

4) Iterating collection.

**Explain about Collection interface in java?**

Collection is the fundamental and root interface in Collections framework. Collection extends Iterable interface and inherits iterator method which returns Iterator object.

Signature: public interface Collection extends Iterable { }

Methods in Collection interface:

|  |  |
| --- | --- |
| boolean add(E e); | Adds an element to the collection. Returns true if element is added. |
| boolean remove(Object o); | Removes an object from collection if that object is present in collection. Return true if matching object is removed from collection. |
| boolean addAll(Collection<? extends E> c); | Adds all the elements specified in the collection to this collection. Returns true if all elements are added. |
| boolean removeAll(Collection<?> c); | Removes all the elements from this collection that are specified in other collection. Returns true if all the elements are removed. |
| int size(); | Returns number of elements in collection. |
| boolean isEmpty(); | Checks whether collection contains elements or not. If no elements are present it returns false. |
| boolean contains(Object o); | Checks whether specified object is in collection or not. Return true if object is in collection. |
| Iterator<E> iterator(); | Used to iterator over collection. No guarantee on order of elements iterated. |
| boolean retainAll(Collection<?> c); | Removes all the elements which are not in specified collection. Returns only elements specified in collection removing other elements. |
| Object[] toArray(); | Returns an array of elements in collection. |

**What are the basic interfaces of Collections framework?**

The root interface is Iterable and then we have Collection framework. Then we have List, Set and Queue. These are the interfaces of Collections framework.

**Why Map interface doesn’t extend Collection interface?**

Maps are not collections even though they belong to the Collections framework. Maps provide functionality based on key value pairs. Collections are a group of elements. The way Maps and Collections work is different so Map interface doesn’t extend collection interface.

**What is iterable?**

Iterable is the interface which is implemented by Collection interface. It provides a method called iterator() which is used to iterate over a collection.

**What are the different ways to iterate over a list?**

We can iterate over a list in two ways.

1. Iterator
2. For – each loop

**List the interfaces which extends collection interface?**

1) List

2) Set

3) Queue

4) Deque

**List implementations of List Interface?**

1) ArrayList

2) Vector

3) LinkedList

**Explain List interface?**

List interface extends collection interface used to store sequence of elements in collection. We can even store duplicate elements in list. We can insert or access elements in list by using index as we do in arrays. List is an ordered collection.

Some of the operations we can perform on List:

1) Adding an element at specified index.

2) Removing an element at specified index.

3) To get the index of element List contains some specific methods apart from Collection interface methods.

**Explain methods specific to List interface?**

|  |  |
| --- | --- |
| boolean addAll(int index, Collection<? extends E>c); | This method inserts all the elements in specified collection to the list at specified position. |
| E get(int index); | This method returns an element at specified position in the list. |
| E set(int index, E element); | This method replaces the element at specified position in the list with the specified element. |
| void add(int index, E element); | This method inserts the specified element with the index specified. |
| E remove(int index); | This method removes the element at specified index and returns the element removed. |
| int indexOf(Object o); | indexOf() method returns the index of last occurrence of specified element. If there is no element in the list it removes the element. |
| ListIterator<E> listIterator(); | ListIterator<E> listIterator(); Returns a list iterator of elements in list. |
| List<E> subList(int fromIndex, int toIndex); | This method returns list of elements between indexes specified. |

**Explain about ArrayList?**

ArrayList is an ordered collection which extends List interface implements collections interface. We use ArrayList mainly when we need faster access and fast iteration of elements in list. We can insert nulls in to ArrayList. Arraylist is nothing but a growable array.

Advantages:

1) Faster and easier access.

2) Used for Random access of elements.

Drawbacks:

1) We cannot insert or delete elements from middle of list.

**Difference between Array and ArrayList?**

Arrays are used to store primitives or objects of same type or variables that are subclasses of same type.

ArrayList: It is an ordered collection which grows dynamically. In list we can insert nulls values and list allows duplicate elements.

|  |  |
| --- | --- |
| **Array** | **ArrayList** |
| While creating array we have to know the size. | But it is not required to know size while creating ArrayList, because arraylist grows dynamically. |
| To put an element in to array we use the following syntax: String array[] = newString[5];array[1] = “java”; We must know specific location to insert an element in to array. If we try to put element in index which is out of range we get arrayIndexOutOfBounds Exception. | We can add element to arraylist with following syntax: List<String> stringList = new ArrayList<String>();stringList.add(“java”); |
| Arrays are static | ArrayList is dynamic |
| We can store objects and primitives | We can store only primitives prior to 1.5 . From 1.5 we can store even objects also. |
| ) We have to manually write logic for inserting and removing elements. | Just a method call would add or remove elements from list. |
| Arrays are faster | Arraylist is slower. |
|  | Arraylist is implemented using arrays |

**What is vector?**

Vector is similar to arraylist used for random access.

Vector is a dynamic array like arraylist.

Size increases or decreases when elements are added and removed.

Vector is synchronized.

**Difference between arraylist and vector?**

Both ArrayList and vector grows dynamically.

The differences between arraylist and vector are:

1) Arraylist is not synchronized and vector is synchronized.

2) Performance wise it is recommended to use arraylist rather than vector because by default vector is synchronized which reduces performance if only one thread accesses it.

**Explain about Sets?**

A set is a collection which does not allow duplicates. Set internally implements equals() method which doesn’t allow duplicates. Adding a duplicate element to a set would be ignored. Set interface is implemented in java.util.set package.

Set interface does not have any additional methods. It has only collection methods. A set can contain atmost one null value. ArrayList is an ordered collection. In arraylists order remains same in which they are inserted. But coming to set it is an unordered collection.

Important operations that can be performed on set:

1) Adding an element to set.

2) Removing an element from set.

3) Check if an element exists in set.

4) Iterating through set.

**Implementations of Set interface?**

1) HashSet

2) Linked HashSet

3) TreeSet

**Explain about Map interface in java?**

A map is an association of key-value pairs. Both keys and values in map are objects.

Features of map:

1. Maps cannot have duplicate keys but can have duplicate value objects.
2. In order to get a value from a map, you need to know the key

Implementations of Map Interface:

1. HashMap
2. HashTable
3. TreeMap

**What is fail-fast system in java?**

When a problem occurs, a fail-fast system fails immediately. In java, we can find this with behaviour with iterators. If you call an iterator on a collection object on a collection object and other thread tries to modify collection object, then concurrent modification exception will be thrown. This is call fail-fast.

Ex: ArrayList, Vector, HashMap

**What is fail-safe system in java?**

Fail-Safe iterators don’t throw any exceptions if the collection is modified while iterating over it. Because they iterate on the clone of the collection and not on the actual collection.

Ex: ConcurrentHashMap