Why do we need collection Framework in java ?

To store multiple objects or group of objects together we can generally use arrays.

But arrays has some limitations

Limitations of Array : The size of the array is fixed, we cannot reduce or increase dynamically during the execution of the program. 1. 2. Array is a collection of homogeneous elements.

1. removing an element from an array.

2. adding the element in between the array etc…

3. Array manipulation such as : Requires complex logic to solve.

Therefore, to avoid the limitations of the array we can store the group of objects or elements using different data structures such as : 1. List 2. Set 3. Queue

Def : Collection Framework is set of classes and interfaces (hierarchies), which provides mechanism to store group of objects (elements) together. It also provides mechanism to perform actions such as :

1. create and add an element

2. access the elements

3. remove / delete the elements

4. search elements

5. update elements

6. sort the elements

Collection interface :

1. Collection is an interface defined in java.util. Collection interface provides the mechanism to store group of objects(elements) together. 2. All the elements in the collection are stored in the form of Objects. (i.e. only Non-primitive is allowed)

4. which helps the programmer to perform the following task.

a. add an element into the collection

b. search an element in the collection

c. remove an element from the collection

d. access the elements present in the collection

add an element

1. add(Object)

2. addAll(Collection)

search an element

1.contains(Object)

2.containsAll(Collection)

remove element

1.remove(Object)

2.removeAll(Collection)

3.retainAll(Collection)

4.clear()

Access elements Iterator

1.iterator( ) we can access elements with the help of for each loop.

2. Miscellaneous

1.size()

2.isEmpty()

3. toArray()

4.hashCode()

5.equals()

List interface:

1. List is an interface defined in java.util package.
2. List is a sub interface of Collection interface. Therefore, it has all the methods inherited from Collection interface.
3. Methods Of List Interface :

add an element

1. add(Object)

2. addAll(Collection)

3.add(int index , Object)

4.addAll(int index , Collection)

search an element

1.contains(Object)

2.containsAll(Collection)

3.indexOf(Object)

remove element

1.remove(Object)

2.removeAll(Collection)

3.retainAll(Collection)

4.clear()

5.remove(int index)

Access elements Iterator

1.iterator( )

2.listIterator()

3.get(int index) ========== Note : we can access with the help of for each loop

Miscellaneous

1.size()

2.isEmpty()

3. toArray()

4.hashCode()

5.equals()

What is list ? What is it Characteristics ?

1. List is a collection of objects. List is an ordered collection of Objects. (It maintains the insertion order of elements)

2. List has indexing, therefore, we can insert, remove or access elements with the help of index. 3. List can have duplicate objects.

ArrayList :

1. It is a Concrete implementing class of List interface.

2. The characteristics of ArrayList is same as List.

3. ArrayList is defined in java.util package.

Note: 1. All the abstract methods of List and Collection interface is implemented.

2. We can create an instance of ArrayList

Note : toString method is Overridden such that it returns a view of all the elements present in the array list.

The elements in the ArrayList can be accessed in the following ways :

1. get method 2. iterator 3. ListIterator 4. for each loop

1. get(index) : 1. get method is used to access the elements present in the ArrayList. 2. it accepts index as an argument. 3. It returns the Object type.

2. To access the elements of ArrayList using iterator.

1. iterator() is method which belongs to Iterable Interface.

iterator() method creates an Iterator type object and returns the reference. 2. iterator() method returns the reference in Iterator (interface) type

java.util.Iterator : Iterator interface has 2 important methods to perform iteration.

hasNext() : it checks whether there is an element to be accessed from the collection, if present it returns true, else it returns false. 1. next() : it is used to access the element, and moves the cursor to the next element. The return type of next() is Object.

NoSuchElementException : In the Iterator, when we try to access an element using next() method and if there is no element present, we get NoSuchElementException.

Disadvantage Of Iterator :

1. We can iterate only once.

2. We cannot access the elements in reverse order.

listIterator() :
listIterator() is a method that belongs to List interface, listIterator method creates an object of ListIterator type.

1. return type of listIterator() is ListIterator interface.

ListIterator:

ListIterator is a sub interface (child) of Iterator interface, it is defined in java.util package.

Methods of ListIterator :

hasNext() to check whether next element to be accessed is present or not.

next() it gives the element, and moves the cursor forward.

hasPrevious() to check whether previous element to be accessed is present or not.

previous() it gives the previous element pointed by the cursor, and moves the cursor backward.

remove(Object) it removes the current object pointed, from the collection, which is under iteration.

for each loop :
syntax :

```
for ( type variable : reference of collection/Array )
{
// statements  }
```

```
for ( Object i : ls )
{System.out.println( i ) ;  }
```

Set :

Set is an interface, it is a sub interface of Collection, therefore all the methods of Collection is inherited.

1. Set is an interface, it is a sub interface of Collection, therefore all the methods of Collection is inherited.

2. Set is defined in java.util package.

Characteristics Of Set :

1. it is an unordered collection of elements. (the order of insertion is lost)

2. Set does not allow duplicate elements. Set does not have indexing. Therefore, we cannot access, insert or remove based on index.

We can access the elements of set only by using:

i. iterator()
ii. for each loop

Concrete Implementing Classes Of Set :

1. HashSet :
2. it is a Concrete implementing class of Set interface. It has all the methods inherited from Collection interface. Characteristics :
   1. It is unordered.
   2. No Duplicates
   3. No index

   Methods:
   To add an element
   1. add(Object)
   2. addAll(Collection)
   TO search an Element
   1.contains(Object)
   2.containsAll(Collection)
   To remove an element
   1.remove(Object o)
   2.removeAll(Collection c)
   3.retainAll(Collection c)
   4.clear ()
   To access the elements
   1.iterator()
   2.using for each loop stmt
   Misc. 1.isEmpty() 2.size() 3.toArray() 4.equals() 5.hashCode()

   2. TreeSet :
   1. It is a concrete implementing class of Set interface.
   2. TreeSet will also have all the methods of Collection interface.
   Characteristics :
   1. The elements will be sorted by default. The elements to be added in treeset must be Comparable type, else we get ClassCastException.
   2. All the elements in TreeSet should be of same type(Homogeneous), if not we get ClassCastException.
   4. Duplicate elements not allowed.

5. No indexing, therefore we cannot add, remove elements using index.

To add an element

1. add(Object)
2. addAll(Collection)

TO search an Element

1.contains(Object)
2.containsAll(Collection)

To remove an element

1.remove(Object o
2.removeAll(Collection c)
3.retainAll(Collection c)
4.clear ()

To access the elements

1.iterator()
2.using for each loop stmt

Misc. 1.isEmpty() 2.size() 3.toArray() 4.equals() 5.hashCode()