**Java Collections**

> An array is an indexed collection of fixed no. of homogenous data elements.

> The main advantage of Array is we can represent multiple values by using single variable so that readability of the code will be improved.

> Limitations of Array are

a) Fixed in size i.e. once we create an array, there is no chance of increasing/decreasing the size based on our requirement. Due to this, to use Array’s concept, compulsory we should know the size in advance which may not possible always.

b) Array can hold only homogenous datatype elements. (Though we can use **Object** type array)

c) Array concept is not implemented based on some standard data Structure & hence readymade method support is not available i.e. for every requirement, we have to write the code explicitly which increases complexity of programming.

> To overcome above limitations of Array, we should go for Collection concept.

> Advantages of Collection are

a) Collections are growable in nature i.e. based on our requirement we can increase/decrease the size.

b) Collections can hold both homogenous & heterogeneous elements.

c) Every collection class is implemented based on some standard data Structure hence for every requirement, readymade method support is available.

> Difference b/w Array & Collection

|  |  |  |
| --- | --- | --- |
| SNo | **Array** | **Collection** |
| 1. | Fixed in size. | Growable in nature. |
| 2. | With respect to memory, Arrays are not recommended to use. | With respect to memory, Collections are recommended to use. |
| 3. | With respect to performance, Arrays are recommended to use. | With respect to performance, Collections are not recommended to use. |
| 4. | Arrays can hold only homogenous datatype elements. | Collections can hold both homo & heterogeneous datatype elements. |
| 5. | No underlying data structure hence no readymade methods are available. | Every collection class has underlying data structure hence readymade methods are available. |
| 6. | Arrays can hold both primitive & objects. | Collections can hold only object types but not primitive. |

**Collections Framework**

> A collections framework is a unified architecture for representing & manipulating collections.

> It contains several interfaces, classes & algorithms which can be used to represent as group of individual objects as a single entity or as a key value pair & perform useful computations.

**Note**: Apart from the Java Collections Framework, the best-known examples of collections framework is the C++ Standard template Library (STL)

> 10 key interfaces of Collection Framework

1. Collection

2. List

3. Set

4. SortedSet

5. NavigableSet

6. Queue

7. Deque

8. Map

9. SortedMap

10. NavigableMap

**1. Collection (Interface)**

> In general, Collection interface is considered as root interface of Collections framework.

> If we want to represent a group of individual Objects as a single entity then we should go for Collection.

> Collection interface defines the most common methods which are applicable for any Collection Object.

> No concrete class can directly implement Collection interface.

**Note**: **Collections** is also an utility class present in **java.util** package containing several methods for Collection objects

like searching, sorting etc.

**2. List (Interface) [Order Collection or Sequence]**

> It is child interface of Collection interface.

> If we want to represent a group of individual objects as a single entity where duplicates are allowed & insertion order must be preserved then we should go for list.

List (I) (1.2 V)

Collection (I) (1.2 V)

1.0 V (legacy classes)

**3. Set (Interface)**

> It is the child interface of Collection interface.

> If we want to represent a group of individual objects as a single entity where duplicates are not allowed & insertion order not required to be preserved then we should go for Set interface.

**4. SortedSet (Interface)**

> It is the child interface of Set interface.

> If we want to represent a group of individual objects as a single entity where duplicates are not allowed & all objects should be inserted according to some sorting order then we should go for SortedSet.

**5. NavigableSet (Interface)**

> It is the child interface of SortedSet interface.

> It contains several methods for Navigation purposes.

Set (I) (1.2 V)

NavigableSet (I) (1.6 V)

SortedSet (I) (1.2 V)

Collection (I) (1.2 V)

> Difference b/w List & Set

|  |  |  |
| --- | --- | --- |
| SNo | **List** | **Set** |
| 1. | Duplicates are allowed. | Duplicates are not allowed. |
| 2. | Insertion order preserved. | Insertion order not preserved. |

6. Queue

> It is the child interface of Collection interface.

> If we want to represent a group of individual objects prior to processing then we should go for Queue.