What is an Array?

An array is a data structure that allows us to store multiple values of the same type in a single variable.

Arrays are particularly useful when we need to manage a collection of items, such as a list of numbers or strings.

- **Fixed Size:** Once you create an array, its size is fixed and cannot be changed. We need to specify the size when we declare the array.
- **Homogeneous Elements**: All elements in an array must be of the same data type (e.g., all integers, all strings).
- **Indexed Access:** We can access element using an index, with the first element at index 0, the second at index 1, and so on.
- **Memory Allocation:** Arrays are allocated in contiguous memory locations, which allows for efficient access to elements.

Drawback of Array-:

- The size of the array is always fixed i.e we cannot modify the size of the array dynamically during usage of application.
- Inside arrays we can store only similar types of objects, we cannot store any other type of element.
- Inside array the manipulation like adding an element, removing an element or searching an elements requires combine logic.

Q. What is collection framework?

Collection framework is set of class and interfaces which provides mechanism to store group of objects (elements).

It also provides mechanism to perform action such as CRUD operations.

- Create and add an element
- Access the elements
- Remove/Delete the elements
- Search elements
- Update elements
- Sort the elements

Collection interface-:

Collection is an interface defined in java.util

It provides the mechanism to store group of objects together.

All the elements in the collection are stored in the form of objects.

Only non-primitive data is allowed. We cannot store primitive values inside collection.

It provides the abstract methods to the programmer to perform task like

- Add an element into the collection
- Search an element
- Remove an element
- Access the elements present in the collection
- It includes various interfaces (like List, Set, Queue) and their implementations (like ArrayList, HashSet, LinkedList, etc.)
- In Selenium, collections are used to manage dynamic sets of web elements, handle multiple items, and store test data.

Methods of collection

1. **Create -:** add(Object), addAll(Collection)
2. **Searching -:** contains(Object), containsAll(collection)
3. **Remove/Delete -:** remove(Object), removeAll(collection), retainAll(collection), clear().
4. **Access-:** Iterator, For each loop

List Interface-:

- List is a sub-interface of collection interface. Therefor all the methods inherited from collection.
- It follows order of insertion.
- It allows duplicates.
- Inside List we can store any type of elements
- It has indexing which help us to perform actions on elements using index.

Methods in List-:

- **Create -:** add(int index, Object), addAll(int index, Collection)
- **Search-:** contains(Object), conatinsAll(Collection), indexOf(Object).
- **Remove-:** remove(index, Object), removeAll(Collection), retainAll(Collection), clear(), remove(int index)

- **Access-:** get(int index), Iterator(), ListIterator()

  We can access with the help of For each loop.

  Set(index, Object o)-: using for replace element.

- **Miscellaneous-:** size(), isEmpty(), toArray(), hashCode(), equals()

Q. Differences between ArrayList and LinkedList?

- **ArrayList** is based on a dynamic array.
- Provides fast access to elements by index allowing direct access to elements.
- Adding or removing elements can be slower because it requires shifting elements to keep them in order.
- Requires less memory as it only stores data in each position of the array.
- It is better for storing and accessing data.

Methods of ArrayList-: add(Object), add(index, Object), addAll(Collection), size(), remove(index), removeAll(Collection), get(index), set(index, Object), contains(Object), isEmpty(), Collections.sort(Collection), Collections.suffle(Collection).

- **LinkedList** is based on a doubly linked list.
- Slower when accessing elements by index, as it must start from the beginning or end and move through each link to reach the desired position.
- Adding or removing elements at the beginning or end is easy.
- It is using more memory per element, as each item also stores links to neighboring elements.
- It is preferred for frequent insertions/deletions at the start or end, especially in queue implementations.
- **Methods of LinkedList-:** add(Object), add(index, Object), addAll(Collection), remove(Object), remove(index), removeAll(Collection), get(index), set(index, Object),
- addFirst(Object), addLast(Object), removeFirst(Object), removeLast(Object), getFirst(), getLast().

Set Interface-:

Set is a sub interface of collection interface. Therefore all the methods of collection interface will be present inside set interface.

- Set is present inside java.util package.
- It is does not maintain the order of insertion.
- It does not allow duplicates.
- It does not have indexing therefor we cannot access, insert or remove using index.

We can access the elements inside set by

I. Using iterator()
II. For each loop

There are two concrete implementing classes of set.

HashSet, LinkedHashSet and TreeSet

HashSet-: Class

- It is does not maintain the order of insertion.
- It does not allow duplicates.
- It does not have indexing
- We can store heterogeneous data.
- It support null.
- We go for HashSet for searching.
- The initial size is 16 element.
- It is having Load factor/ratio

Note-: As soon as 0.75% of the locations are filled then the new object will be created and all the elements will move to the new object and the new element also added.

Methods of HashSet-:

add(Object), addAll(Collection), remove(value) -: bez it doesn't support index.

removeAll(Collection), contains(Object), containsAll(Collection), isEmpty().

Here we can check Union, Intersection, Difference, Subset.

LinkedHashSet-: class

- Duplicates are not allowed.
- Insertion order is allowed.
- It is having HashTable + LinkedList class
- Intial size is 16

LinkedHashSet lhs = new LinkedHashSet();

TreeSet-:

It is also a concrete implementing class of set.

TreeSet will have all the methods of collection interface.

- The elements will be sorted by default.
- The elements to be added in TreeSet must be of comparable type else we get classCastException.
- All the elements in TreeSet should be of same type else we get classCastException.
- Duplicates are not allowed
- No indexing therefor we cannot access the elements using indexing.

It has 2 constructor declared

- TreeSet()-: it create an empty TreeSet object.
- TreeSet(Collection c)-: It creates a TreeSet object and copies all the elements of collection into TreeSet.

Queue-: interface

Purpose of Queue-:

- If we have group of elements which are a prior to processing then we can go for Queue concept.
- It is follow FIFO concept whichever element is inserted first that element will be process.
- There are two ends, from Tail we will insert an element and from the Head end we will get the element of the queue.

It is having 2 class-: PriorityQueue and LinkedList

PriorityQueue-:

- Insertion order is allowed.
- Duplicates are allowed.
- Heterogeneous data are not allowed. Same type of data need to be pass.

Methods Of Queue-:

add(), offer()-:

- If add() is successful it will return true. If it is not successful it will return Exception.
- If offer()is successful it will return true. If it is not successful it will return false.

element(), peek() -: It will help to get Head element.

- If there is Head element then both method will return true.
- If the Queue is empty then the element() will return an Exception and peek() will return null.

remove(), poll()-:

- These are used to return the Header element and remove the element.
- If the queue is empty then remove() will throw an exception and poll() will return null.

MAP interface-:

- Here we can store the element as key and value pairs.
- Every keys are unique but we can add duplicate values in the Map.
- Each key and value we can call as one pair or one entry.
- We are having two class -: HashMap and HashTable

HashMap -: It is implemented with Map interface.

- Insertion order is not allowed.
- Duplicate keys are not allowed.
- Duplicate values are allowed.
- Null key allowed but only one time.
- Null value we can use multiple time.
- Underlying data structure in HashTable.

If we have more number of searching points then we should go with HashMap.

Methods of HashMap -:

I. put(key, value)
II. putAll(map)
III. containsKey(kay)-: return type is true/false
IV. containsValue(value)-: return type is true/false
V. get(key)-: return a value
VI. keyset()-: return type is Set, it will return all the Keys
VII. values()-: return type is Collection, it will return all the values
VIII. entrySet()-: return type is Collection, it will return all the Keys & Values
IX. clear()
X. remove()
XI. size()
XII. isEmpty()-: return type is true/false

HashTable-:

- Here data will be stored as key & value pair.
- Underlying data structure for HashMap & HashCode.

Difference between HashMap & HashTable

HashTable-:

- It is synchronize (At a time only one thread will works on the particular method and other threads should wait.)
- One thread allowed at a time.
- Thread safe.
- Performance is low.
- Null cannot accept.
- Insertion order not preserved.
- Default capacity is 11.
- Sort is not present.

HashMap-:

- It is non-synchronize.
- Multiple thread can be allowed at a time.
- Not thread safe.
- Performance is faster.
- Null can accept.

TreeMap

It is having two constructor

TreeMap() and TreeMap(Map)

Length

- It is a non-static variable of array.
- It gives us the size of array object that is total number of elements present inside array object.

Length()-:

It is a non-static method present inside String class which help us to get number of characters present inside a String.

Sorting -:

It is a process of arranging the elements either in ascending order or descending order.

There are 2 way to short

- With the help of Built-in

```
int [] a= { 20,50,80,60};
Array.sort(a);
```

- With the help of shorting algorithm-: Bubble sort, selection sort, Insertion sort, merge sort, Heap sort, quick sort.

Object Class-:

Object class is the super most class of all the classes, which is defined in java.lang package

The fully qualified name of object class is java.lang.Object. It has 11 non-static method.

1. public String toString()
2. public boolean equals(Object o)
3. public int hashCode()
4. protected Object clone() throws CloneNotSupportedException
5. protected void finalize()
6. final public void wait()

7. final public void wait(long)
8. final public void wait(long.int)
9. final public void notify()
10. final public void notifyAll()
11. final public class getClass()

equal (==) operator-:

- The equality operator is use to compare the address of objects.
- It cannot compare states of the objects.
- When we compare the reference with the help of equality operator if it is returning us true then it means both the reference variable are pointing to same object.
- If it is returning false then it means both the reference variable are pointing to two different object.

Equals()-:

- The equals() of object class compare the address of object which is of no use for the user.
- Therefor we will override equals() to compare state of object instead of address.

Difference between interator() and listIterator().

Iterator-:

- Using iterator() we can access the elements in only one direction.
- We can access the elements only once.
- We cannot add the elements while accessing.
- We cannot modify the element with the help of iterator().
- We cannot obtain index of any elements using iterator().
- Using iterator() we can access all the collection like list, set, queue.
- There are 3 methods present those are hasNext(), next(), remove().

listIteraor()

- using listIteraor() we can access the elements in forward and backword direction.
- We can access any number of times.
- We can add the elements with the help of add(Object o) present inside listIteraor()
- We can modify the elements with the help of set(object)
- We can obtain the index of elements using previous(index) and next(index)
- Using listIteraor() we can access only the members of list and its sub members.
- The methods are hasNext(), next(), remove(), hasPrevious(), previous(), nextIndex(), previousIndex(), set(Object), add(Object).