Java Collection Framework

1.What is the Collection framework in Java?

Ans: Collection Framework is a combination of classes and interface,which is used to store and manipulate the data in the form of objects. It provides various classes such as ArrayList, Vector, Stack, and HashSet, etc. and interfaces such as List, Queue, Set, etc. for this purpose.

2. What is the difference between ArrayList and LinkedList?

Ans:

ArrayList	LinkedList
ArrayList uses a dynamic array to store the elements in it.	LinkedList uses a doubly linked list to store the elements in it.
ArrayList is not efficient for manipulation because too much is required.	LinkedList is efficient for manipulation.
ArrayList is better to store and fetch data.	LinkedList is better to manipulate data.
ArrayList provides random access.	LinkedList does not provide random access.
ArrayList takes less memory overhead as it stores only object.	LinkedList takes more memory overhead, as it stores the object as well as the address of that object.
Insertion operation is slow.	Insertion operation is fast.
Deletion operation is not very efficient.	Deletion operation is very efficient.
It is used to store only similar types of data.	It is used to store any types of data.
The memory is allocated at compile-time only.	The memory is allocated at run-time.
This class implements a List interface. Therefore, this acts as a list.	This class implements both the List interface and the Deque interface. Therefore, it can act as a list and a deque.

This is known as static memory allocation.	This is known as dynamic memory allocation.
Data access and storage is very efficient as it stores the elements according to the indexes.	Data access and storage is slow in LinkedList.

3. What is the difference between Iterator and ListIterator?

Ans: An Iterator is an interface in Java and we can traverse the elements of a list in a forward direction whereas a ListIterator is an interface that extends the Iterator interface and we can traverse the elements in both forward and backward directions.

An Iterator can be used in these collection types like List, Set, and Queue whereas ListIterator can be used in List collection only.

The important methods of Iterator interface are hasNext(), next() and remove() whereas important methods of ListIterator interface are add(), hasNext(), hasPrevious() and remove().

Indexes cannot be obtained by using Iterator. While ListIterator It has methods like nextIndex() and previousIndex() to obtain indexes of elements at any time while traversing List.

An Iterator Cannot add elements and it throws ConcurrentModificationException. While ListIterator Can easily add elements to a collection at any time.

4. What is the difference between Iterator and Enumeration?

Ans:

Iterator	**Enumeration**
Iterator is a universal cursor as it is applicable for all the collection classes.	Enumeration is not a universal cursor as it applies only to legacy classes.
Iterator has the remove() method.	Enumeration does not have the remove() method.
Iterator can do modifications (e.g using remove() method it removes	Enumeration interface acts as a read only interface, one can not do any

the element from the Collection during traversal).	modifications to Collection while traversing the elements of the Collection.
Iterator is not a legacy interface. Iterator can be used for the traversal of HashMap, LinkedList, ArrayList, HashSet, TreeMap, TreeSet .	Enumeration is a legacy interface which is used for traversing Vector, Hashtable.
It can be used with any class of the collection framework.	It can be used only with legacy class of the collection framework such as a Vector and HashTable.

It has following methods
*hasNext()
*next()
*remove()
	It has following methods
	*hasMoreElements()
	*nextElement()

5. What is the difference between List and Set?

Ans: The List and Set both extend the collection interface. However, there are some differences between the two which are listed below.

- The List can contain duplicate elements whereas Set includes unique items.
- The List is an ordered collection which maintains the insertion order whereas Set is an unordered collection which does not preserve the insertion order.
- The List interface contains a single legacy class which is Vector class whereas the Set interface does not have any legacy class.
- The List interface can allow a number of null values whereas Set interface only allows a single null value.

6. What is the difference between HashSet and TreeSet?

Ans: Both HashSet and TreeSet are implementations of the Set interfaces in Java, but they have some differences in terms of their properties and usage.

- **Ordering:** HashSet is an unordered collection of elements, while TreeSet is a sorted set of elements based on their natural order or a custom comparator.
- **Duplication:** HashSet does not allow duplicate elements, while TreeSet does not allow duplicates as well.
- **Implementation:** HashSet is implemented using a hash table, while TreeSet is implemented using a self-balancing binary search tree (Rep-Black tree).
- **Performance:** HashSet has constant-time complexity O(1) for adding, removing, and testing the existence of an element, while TreeSet has a logarithmic-time complexity O(log n) for these operations due to the self-balancing property.
- **Memory usage:** HashSet uses less memory than TreeSet because it only stores the elements, while TreeSet stores additional information for maintaining the order.
- **Iteration:** HashSet provides no guarantees regarding the order of iteration, while TreeSet guarantees the elements are iterated in sorted order.
- **Usage:** HashSet is suitable when ordering is not important, and fast access and membership tests are needed. TreeSet is suitable when elements need to be sorted or accessed in a specific order.

7. What is the difference between Array and ArrayList?

Ans: Both arrays and ArrayLists are used to store collections of elements in Java, but they have some differences in terms of their properties and usage:

- **Type:** Arrays can store elements of primitive data types as well as objects, while ArrayList can only store Objects.
- **Size:** The size of an array is fixed once it is created, while the size of an ArrayList can be dynamically increased or decreased by adding or removing elements.
- **Mutability:** Arrays are mutable, meaning that you can modify the elements in an array after it has been created. ArrayList is also mutable, but the only way to modify it is by adding, removing or modifying elements.

- **Performance:** Arrays have better performance than ArrayLists for certain operations, such as accessing elements by index, because they are implemented as a continuous block of memory. ArrayLists, on the other hand, use dynamic memory allocation and are implemented as a dynamic array, which may result in more memory overhead and slower performance for certain operations.
- **Methods:** Arrays have a limited set of methods compared to ArrayLists, which provides more methods for manipulating the collection, such as adding, removing, and sorting elements.
- **Initialization:** Arrays can be initialized with values at the time of creation, while ArrayList requires the use of methods to add elements to the collection.
- **Compatibility:** Arrays are compatible with traditional for-loops and can be easily passed to other methods, while ArrayList requires the use of a special for-each loop and may require more code to be passed to other methods.