Collection Framework

Question 1: What is the collection framework in java?

Answer- It 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.

Question 2: What is the difference between ArrayList and LinkedList?

Answer –

ArrayList	LinkedList
1.ArrayList internally uses a dynamic array to store the elements.	1.LinkedList internally uses a doubly linked list to store the elements.
2.Manipulation with Arraylist is slow because it internally uses an array. If any element is removed from the array, all the other elements are shifted in memory.	2.manipulation with LinkedList is faster than ArrayList because it uses a doubly linkedlist, so no bit shifting is required in memory.
3.An arraylist class can act as a list only because it implements list only.	3.Linkedlist class can act as a list and queue both because it implements Lists and Dequeue interfaces.
4. ArrayList is better for storing and accessing data.	4. Linkedlist is better for manipulating data.
5.The memory location for the element of an ArrayList is contiguous.	5.The memory location for the elements of a linkedlist is not contiguous.
6.Generally, when an Arraylist is initialized , a default capacity of 10 is assigned to the ArrayList.	6.There is no case of default capacity in a linkedlist , In linkedlist , an empty list is created when a linkedlist is initialized.
7.To be precise, An ArrayList is a resizable array.	7.LinkedList implements the doubly linked list of the list interface.

Question 3: What is the difference between Iterator and ListIterator?

Iterator	ListIterator
Can traverse elements present in Collection only in the forward direction.	Can traverse elements present in Collection both in forward and backward directions.
Helps to traverse Map, List and Set.	Can only traverse List and not the other two.
Indexes cannot be obtained by using Iterator.	It has methods like nextIndex() and previousIndex() to obtain indexes of elements at any time while traversing List.
Cannot modify or replace elements present in Collection	We can modify or replace elements with the help of set(E e)
Cannot add elements and it throws ConcurrentModificationException.	Can easily add elements to a collection at any time.
Certain methods of Iterator are next(), remove() and hasNext().	Certain methods of ListIterator are next(), previous(), hasNext(), hasPrevious(), add(E e).

Question 4: What is the difference between Iterator and Enumeration?

Answer:

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 the element from the Collection during traversal).	Enumeration interface acts as a read only interface, one can not do any 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.

Question 5: What is the difference between List and Set?

Answer:

S.No	List	Set
1.	The list implementation allows us to add the same or duplicate elements.	The set implementation doesn't allow us to add the same or duplicate elements.
2.	The insertion order is maintained by the List.	It doesn't maintain the insertion order of elements.
3.	List allows us to add any number of null values.	Set allows us to add at least one null value in it.
4.	The List implementation classes are LinkedList and ArrayList.	The Set implementation classes are TreeSet, HashSet and LinkedHashSet.
5.	We can get the element of a specified index from the list using the get() method.	We cannot find the element from the Set based on the index because it doesn't provide any get method().
6.	It is used when we want to frequently access the elements by using the index.	It is used when we want to design a collection of distinct elements.
7.	The method of List interface listiterator() is used to iterate the List elements.	The iterator is used when we need to iterate the Set elements.

Question 6: What is the difference between HashSet and TreeSet?

Answer:

Parameters	HashSet	TreeSet
Ordering or Sorting	It does not provide a guarantee to sort the data.	It provides a guarantee to sort the data. The sorting depends on the supplied Comparator.
Null Objects	In HashSet, **only an element** can be null.	It does not allow null elements.
Comparison	It uses **hashCode()** or **equals()** method for comparison.	It uses **compare()** or **compareTo()** method for comparison.
Performance	It is **faster** than TreeSet.	It is **slower** in comparison to HashSet.
Implementation	Internally it uses **HashMap** to store its elements.	Internally it uses **TreeMap** to store its elements.
Data Structure	HashSet is backed up by a hash table.	TreeSet is backed up by a Red-black Tree.
Values Stored	It allows only **heterogeneous** value.	It allows only **homogeneous** value.

Question 7: What is the difference between Array and ArrayList?

Answer:

Basis	Array	ArrayList
Definition	An **array** is a dynamically-created object. It serves as a container that holds the constant number of values of the same type. It has a contiguous memory location.	The **ArrayList** is a class of Java **Collections** framework. It contains popular classes like **Vector, HashTable**, and **HashMap**.
Static/ Dynamic	Array is **static** in size.	ArrayList is **dynamic** in size.
Resizable	An array is a **fixed-length** data structure.	ArrayList is a **variable-length** data structure. It can be resized itself when needed.
Initialization	It is mandatory to provide the size of an array while initializing it directly or indirectly.	We can create an instance of ArrayList without specifying its size. Java creates ArrayList of default size.
Performance	It performs **fast** in comparison to ArrayList because of fixed size.	ArrayList is internally backed by the array in Java. The resize operation in ArrayList slows down the performance.
Primitive/ Generic type	An array can store both **objects** and **primitives** type.	We cannot store **primitive** type in ArrayList. It automatically converts primitive type to object.
Iterating Values	We use **for** loop or **for each** loop to iterate over an array.	We use an **iterator** to iterate over ArrayList.
Type-Safety	We cannot use generics along with array because it is not a convertible type of array.	ArrayList allows us to store only **generic/ type, that's why it is type-safe.**

Length	Array provides a **length** variable which denotes the length of an array.	ArrayList provides the **size()** method to determine the size of ArrayList.
Adding Elements	We can add elements in an array by using the **assignment** operator.	Java provides the **add()** method to add elements in the ArrayList.
Single/ Multi-Dimensional	Array can be **multi-dimensional**.	ArrayList is always **single-dimensional**.