**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** |
| 1) ArrayList internally uses a **dynamic array** to store the elements. | 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. | Manipulation with LinkedList is **faster** than ArrayList because it uses a doubly linked list, so no bit shifting is required in memory. |
| 3) An ArrayList class can **act as a list** only because it implements List only. | LinkedList class can **act as a list and queue** both because it implements List and Deque interfaces. |
| 4) ArrayList is **better for storing and accessing** data. | LinkedList is **better for manipulating** data. |
| 5) The memory location for the elements of an ArrayList is contiguous. | The location for the elements of a linked list is not contagious. |
| 6) Generally, when an ArrayList is initialized, a default capacity of 10 is assigned to the ArrayList. | 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. | LinkedList implements the doubly linked list of the list interface. |

**3)** **What is the difference between Iterator and ListIterator?**

**Ans-**

| **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). |

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 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. |

**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 interface 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 (Red-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

8) What is a Map in Java?

Ans: A Map is a collection in Java that stores data as key-value pairs, where each key is unique

9) What are the commonly used implementations of Map in Java?

Ans: The commonly used implementations of Map in Java are HashMap, TreeMap, LinkedHashMap, and ConcurrentHashMap.

10. What is the difference between HashMap and TreeMap?

Ans: HashMap is an unordered collection that uses hashing to store the key-value pairs, while TreeMap is a

sorted collection that stores the key-value pairs in a sorted order based on the natural order of the keys or a

custom Comparator

11)How do you check if a key exists in a Map in Java?

Ans: we can check if a key exists in a Map in Java using the containsKey() method or the get() method. containsKey() method returns a boolean value indicating whether the Map contains the specified key, while the get() method returns the value associated with the specified key, or null if the key is not present in the Map