### Explain various interfaces used in Collection framework?

Collection framework implements various interfaces, Collection interface and Map interface (java.util.Map) are the mainly used interfaces of Java Collection Framework. List of interfaces of Collection Framework is given below:

**1. Collection interface:** Collection (java.util.Collection) is the primary interface, and every collection must implement this interface.

**Syntax:**

1. **public** **interface** Collection<E>**extends** Iterable

Where <E> represents that this interface is of Generic type

**2. List interface:**List interface extends the Collection interface, and it is an ordered collection of objects. It contains duplicate elements. It also allows random access of elements.

**Syntax:**

1. **public** **interface** List<E> **extends** Collection<E>

**3. Set interface:** Set (java.util.Set) interface is a collection which cannot contain duplicate elements. It can only include inherited methods of Collection interface

**Syntax:**

1. **public** **interface** Set<E> **extends** Collection<E>

**Queue interface:**Queue (java.util.Queue) interface defines queue data structure, which stores the elements in the form FIFO (first in first out).

**Syntax:**

1. **public** **interface** Queue<E> **extends** Collection<E>

**4. Dequeue interface:** it is a double-ended-queue. It allows the insertion and removal of elements from both ends. It implants the properties of both Stack and queue so it can perform LIFO (Last in first out) stack and FIFO (first in first out) queue, operations.

**Syntax:**

1. **public** **interface** Dequeue<E> **extends** Queue<E>

**5. Map interface:**A Map (java.util.Map) represents a key, value pair storage of elements. Map interface does not implement the Collection interface. It can only contain a unique key but can have duplicate elements. There are two interfaces which implement Map in java that are Map interface and Sorted Map.

### What is the difference between Iterator and ListIterator?

Iterator traverses the elements in the forward direction only whereas ListIterator traverses the elements into forward and backward direction.

|  |  |  |
| --- | --- | --- |
| **No.** | **Iterator** | **ListIterator** |
| 1) | The Iterator traverses the elements in the forward direction only. | ListIterator traverses the elements in backward and forward directions both. |
| 2) | The Iterator can be used in List, Set, and Queue. | ListIterator can be used in List only. |
| 3) | The Iterator can only perform remove operation while traversing the collection. | ListIterator can perform ?add,? ?remove,? and ?set? operation while traversing the collection. |

### 7) What is the difference between Iterator and Enumeration?

|  |  |  |
| --- | --- | --- |
| **No.** | **Iterator** | **Enumeration** |
| 1) | The Iterator can traverse legacy and non-legacy elements. | Enumeration can traverse only legacy elements. |
| 2) | The Iterator is fail-fast. | Enumeration is not fail-fast. |
| 3) | The Iterator is slower than Enumeration. | Enumeration is faster than Iterator. |
| 4) | The Iterator can perform remove operation while traversing the collection. | The Enumeration can perform only traverse operation on the collection. |

What is the difference between List and Set?

The List and Set both extend the collection interface. However, there are some differences between the both 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 Set interface does not have any legacy class.
* The List interface can allow n number of null values whereas Set interface only allows a single null value

### difference between HashSet and TreeSet

| **Sr. No.** | **Key** | **Hash Set** | **Tree Set** |
| --- | --- | --- | --- |
| 1 | Implementation | Hash set is implemented using HashTable | The tree set is implemented using a tree structure. |
| 2 | Null Object | HashSet allows a null object | The tree set does not allow the null object. It throws the null pointer exception. |
| 3 | Methods | Hash set use equals method to compare two objects | Tree set use compare method for comparing two objects. |
| 4 | Heterogeneous object | Hash set doesn't now allow a heterogeneous object | Tree set allows a heterogeneous object |
| 5 | Ordering | HashSet does not maintain any order | TreeSet maintains an object in sorted order |

What is the difference between HashSet and HashMap?

The differences between the HashSet and HashMap are listed below.

* HashSet contains only values whereas HashMap includes the entry (key, value). HashSet can be iterated, but HashMap needs to convert into Set to be iterated.
* HashSet implements Set interface whereas HashMap implements the Map interface
* HashSet cannot have any duplicate value whereas HashMap can contain duplicate values with unique keys.
* HashSet contains the only single number of null value whereas HashMap can hold a single null key with n number of null values.

What is the difference between HashMap and TreeMap?

The differences between the HashMap and TreeMap are given below.

* HashMap maintains no order, but TreeMap maintains ascending order.
* HashMap is implemented by hash table whereas TreeMap is implemented by a Tree structure.
* HashMap can be sorted by Key or value whereas TreeMap can be sorted by Key.
* HashMap may contain a null key with multiple null values whereas TreeMap cannot hold a null key but can have multiple null values

### What is the difference between HashMap and Hashtable?

|  |  |  |
| --- | --- | --- |
| **No.** | **HashMap** | **Hashtable** |
| 1) | HashMap is not synchronized. | Hashtable is synchronized. |
| 2) | HashMap can contain one null key and multiple null values. | Hashtable cannot contain any null key or null value. |
| 3) | HashMap is not ?thread-safe,? so it is useful for non-threaded applications. | Hashtable is thread-safe, and it can be shared between various threads. |
| 4) | 4) HashMap inherits the AbstractMap class | Hashtable inherits the Dictionary class. |

### What is the difference between Comparable and Comparator?

|  |  |  |
| --- | --- | --- |
| **No.** | **Comparable** | **Comparator** |
| 1) | Comparable provides only one sort of sequence. | The Comparator provides multiple sorts of sequences. |
| 2) | It provides one method named compareTo(). | It provides one method named compare(). |
| 3) | It is found in java.lang package. | It is located in java.util package. |
| 4) | If we implement the Comparable interface, The actual class is modified. | The actual class is not changed. |

### What do you understand by BlockingQueue?

BlockingQueue is an interface which extends the Queue interface. It provides concurrency in the operations like retrieval, insertion, deletion. While retrieval of any element, it waits for the queue to be non-empty. While storing the elements, it waits for the available space. BlockingQueue cannot contain null elements, and implementation of BlockingQueue is thread-safe.

### How to synchronize List, Set and Map elements?

Yes, Collections class provides methods to make List, Set or Map elements as synchronized

What is the advantage of the generic collection?

There are three main advantages of using the generic collection.

* If we use the generic class, we don't need typecasting.
* It is type-safe and checked at compile time.
* Generic confirms the stability of the code by making it bug detectable at compile time.