**Enumeration**

The Enumeration interface defines the methods by which you can enumerate (obtain one at a time) the elements in a collection of objects.

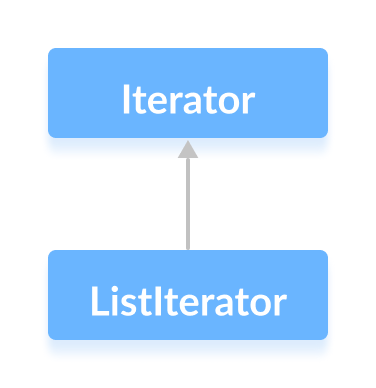
This legacy interface has been superceded by Iterator.

Although not deprecated, Enumeration is considered obsolete for new code.

However, it is used by several methods defined by the legacy classes such as Vector/

The methods declared by Enumeration are summarized in the following table −

|  |  |
| --- | --- |
| **Sr.No.** | **Method & Description** |
| 1 | **boolean hasMoreElements( )**  When implemented, it must return true while there are still more elements to extract, and false when all the elements have been enumerated. |
| 2 | **Object nextElement( )**  This returns the next object in the enumeration as a generic Object reference. |



| **Sr. No.** | **Key** | **Iterator** | **ListIterator** |
| --- | --- | --- | --- |
| 1 | Applicable | Iterator can be used to traverse **any collection** irrespective of the type of collection. | ListIterator can only be used to iterate **only List collection** implemented classes like arraylist,linkedlist etc. |
| 2 | Calling | Iterator object can be created by calling **iterator()** method of Collection interface. | The object of ListIterator can be created by calling **listIterator()** method present in List interface. |
| 3 | Data traverse | Data traverse in case of the iterator is possible only in **one direction** as Iterator can traverse in the **forward direction only** | List iterator could traverses both in **forward and backward** directions which makes data traverse in both directions. |
| 4 | Deletion | The **deletion** **of in between element is** **not allowed** in the case of the iterator.(remove() removes last element) | ListIterator can **replace** an element in list collection. |
| 5 | Addition | The **addition** of an element **is not allowed** in case of an iterator as it throws ConcurrentModificationException. | ListIterator **can add** an element in list collection any time easily. |
| 6 | Modification | **Modification** of an element is **not allowed** in case of an iterator as it throws ConcurrentModificationException. | ListIterator **can modify** an element in list collection any time easily by calling set() method. |
| 7 | Index of element | One **can't get the index** of the traversed element in collection while using an iterator. | ListIterator has methods like nextIndex() and previousIndex() to **obtain indexes** of elements at any time while traversing List. |

**Iterator**

**Methods of Iterator Interface**

|  |  |
| --- | --- |
| **Methods** | **Description** |
| forEachRemaining(Consumer<? super E>action) | Performs the given action on each of the element until and unless all the elements have been processed or unless an exception is thrown by the action. |
| hasNext() | * returns true if there exists an element in the list |
| next() | returns the next element of the list. |
| remove() | Removes the last element from the collection as provided by the iterator. |

**LISTITERATOR**

**Methods of ListIterator Interface**

|  |  |
| --- | --- |
| **Methods** | **Description** |
| nextIndex() | returns the index of the element that the next() method will return |
| hasNext() | * returns true if there exists an element in the list |
| next() | returns the next element of the list |
| previous() | returns the previous element of the list |
| previousIndex() | returns the index of the element that the previous() method will return |
| remove() | removes the element returned by either next() or previous() |
| set() | * replaces the element returned by either next() or previous() with the specified element |