ListIterator<E> interface.

ListIterator<E> allows

Traverse in Either direction

Modify list during iteration

Get iterator's current position in list.

IMPORTANT: ListIterator<E> extends Iterator<E> interface.

As mentioned in JavaDoc ListIterator does not have current element; the cursor position always lies between element that would be returned call to previous() and element that would be returned by call to next().

So in List<E> n elements there are n+1 iterator positions

element(0) element(1) element(2) . . . element(n-1)

Current ^ ^ ^ ^ ^

position:

ListIterator is usually created by corresponding container class by invoking listIterator().

|  |  |
| --- | --- |
| **Method** | **Description** |
| **boolean** hasNext(); | Returns true if there are more elements while list is being traversed in forward direction. |
| **boolean** hasPrevious(); | Returns true if there are more elements when list is being traversed backwards. |
| E previous(); | Returns the previous element in iteration while being traversed in backward direction. It will throw NoSuchElementException if there are no more elements. |
| E next(); | Returns the next element in iteration while being traversed in forward direction. It will throw NoSuchElementException if there are no more elements. |
| **int** nextIndex(); | Returns the index of element that will be returned by next() |
| **int** previousIndex(); | Returns the index of element that will be returned by previous() |
| **void** remove(); | removes the element from the list which was returned by next() or previous() |
| **void** set(E e); | Replaces the last element that was returned by next() or previous(). |
| **void** add(E e); | Inserts element e in the list |

Below is the code that uses ListIterator<E>.

**package** iterator;

**import** java.util.ArrayList;

**import** java.util.List;

**import** java.util.ListIterator;

**public** **class** ListIteratorImpl {

**public** **static** **void** main(String[] args) {

List<String> list = **new** ArrayList<String>();

list.add("Java");

list.add("C#");

list.add("MySQL");

list.add("Java");

list.add("MongoDB");

list.add("MySQL");

list.add("Oracle");

//[Java, C#, MySQL, Java, MongoDB, MySQL, Oracle]

System.***out***.println("List is -> "+list);

*iterateByListIterator*(list);

}

**static** **void** iterateByListIterator(List<String> list) {

ListIterator<String> listItr = list.listIterator();

//Below prints C# 1, MySQL 2, MongoDB 3, MySQL 4, Oracle 5

System.***out***.print("Using next() List is -> ");

**while** (listItr.hasNext()) {

String str = listItr.next();

**if** (str.equalsIgnoreCase("java")) {

// remove all entries that are "Java".

listItr.remove();

**continue**;

}

System.***out***.print(str+" "+listItr.nextIndex());

**if** (listItr.hasNext()) {

System.***out***.print(", ");

}

}

System.***out***.println();

/\*\*

\* Now let us use set(E e).

\* Set "MySSQL" to "Java".

\* And traverse using hasPrevious as currently we are at the end of list

\* \*/

System.***out***.print("Using previous() List is -> ");

**while** (listItr.hasPrevious()) {

String str = listItr.previous();

**if** (str.equalsIgnoreCase("MySQL")) {

/\*\*

\* replace all entries that are "MySQL"

\* to "Java"

\*/

listItr.set("Java");

listItr.next();

**continue**;

}

//Prints Oracle, Java, MongoDB, Java, C#

System.***out***.print(str);

**if** (listItr.hasPrevious()) {

System.***out***.print(", ");

}

}

/\*\*

\* adding new element at the beginning of list as

\* cursor is currently pointing before first element.

\*/

listItr.add("First Element");

System.***out***.println();

System.***out***.println("List is "+list);

}

}

Below is the output:

List is -> [Java, C#, MySQL, Java, MongoDB, MySQL, Oracle]

Using next() List is -> C# 1, MySQL 2, MongoDB 3, MySQL 4, Oracle 5

Using previous() List is -> Oracle, Java, MongoDB, Java, C#

List is [First Element, C#, Java, MongoDB, Java, Oracle]