**TRABAJO PROGRAMACIÓN:**

**ListIterator**

**En programación de computadoras, un iterador se refiere al objeto que permite al programador recorrer un contenedor, (una colección de elementos) particularmente listas.**

An iterator for lists that allows the programmer to traverse the list in either direction, modify the list during iteration, and obtain the iterator's current position in the list. A ListIterator has no current element; its cursor position always lies between the element that would be returned by a call to previous() and the element that would be returned by a call to next(). An iterator for a list of length n has n+1 possible cursor positions, as illustrated by the carets (^) below:

ListIterator es un iterador para listas que permite recorrer la lista en cualquier dirección, modificar la lista durante la iteración y obtener la posición actual del iterador en la lista. Un ListIterator no tiene ningún elemento actual; la posición de su cursor siempre se encuentra entre el elemento que sería devuelto por una llamada a previous() y el elemento que sería devuelto por una llamada a next(). Un iterador para una lista de longitud n tiene n+1 posibles posiciones del cursor, como lo ilustran los signos de intercalación (^) a continuación:

**Element(0) Element(1) Element(2) ... Element(n-1)**

**cursor positions: ^ ^ ^ ^ ^**

Un ListIterator es un iterador Java que nos permite recorrer una lista de elementos en varias direcciones, bien hacia delante o bien hacía atrás. Para recorrer una lista con ListIterator lo primero que tenemos que hacer es crear una lista. En este caso vamos a crear un ArrayList.

ArrayList al = new ArrayList();

// Añadir elementos a un ArrayList

al.add("Victor");

al.add("Luis");

al.add("Elena");

Lo siguiente para recorrer una lista con ListIterator será obtener el puntero al iterador, esto lo conseguimos mediante el método .listIterator(). El puntero se quedará sobre un elemento ListIterator.

ListIterator it = al.listIterator();

Ahora que hemos obtenido el puntero podemos recorrer una lista con ListIterator desplazando el puntero hacia delante o moviendo el puntero hacía atrás. Esta segunda opción será muy útil para mostrar los elementos en orden inverso de la lista.

Podemos recorrer una lista con ListIterator hacia delante que seguirá el mismo mecanismo que cuando explicábamos cómo iterar un ArrayList.

De esta forma moveremos el iterador mediante el método .next() y controlaremos el final de la lista mediante el método .hasNext().

**while(it.hasNext()){**

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

**}**

Pero podemos recorrer una lista con ListIterator en orden inverso. En este caso vamos a utilizar el método .previous().

**while(it.hasPrevious()){**

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

**}**

Sabremos si hemos llegado al principio de la lista cuando el método .hasPrevious() nos devuelva false.

**INFORMACIÓN IMPORTANTE**

In Java, ListIterator is an interface in Collection API. It extends Iterator interface. To support Forward and Backward Direction iteration and CRUD operations, it has the following methods. **We can use this Iterator for all List implemented** classes like ArrayList, CopyOnWriteArrayList, LinkedList, Stack, Vector, etc.

En Java, ListIterator es una interfaz en Collection API. Amplía la interfaz Iterator. Para admitir iteraciones en dirección hacia adelante y hacia atrás y operaciones CRUD, tiene los siguientes métodos. Podemos usar este Iterador para todas las clases implementadas en Lista como ArrayList, CopyOnWriteArrayList, LinkedList, Stack, Vector, etc.

**ERRORES**

listIterator para gestionar un Set ->

Información ListIterator: <https://www.digitalocean.com/community/tutorials/java-listiterator>

Nathan S:

<https://www.youtube.com/watch?v=quRRkdsY13U>

Programándonos

<https://www.youtube.com/watch?v=Nbp7ANBXF7E>