UT9 Ej12: Preguntas

Nombre: Jose Cabrera Rojas
Ciclo: 1º DAW

Responder a las preguntas:

A la hora de eliminar elementos es mucho más eficiente un ArrayList que un LinkedList. Verdadero/Falso. Indica el motivo.

Verdadero. Esto se debe a que LinkedList no tiene una buena gestión para el acceso a elementos aleatorios.

Si tenemos una LinkedList lista de cadenas. Escribe las sentencias válidas.

- **¿Con qué dos métodos agregamos un elemento en la última posición?**

```
listaCadenas.add(listaCadenas.size(),"elemento");
listaCadenas.addLast("elemento");
listaCadenas.push("elemento");
```

- **¿Cómo agregamos un elemento en la tercera posición (tener en cuenta como se numera al primer elemento)?**

```
listaCadenas.add(2,"elemento");
```

- **¿Pon tres formas para obtener el primer elemento de la lista sin eliminarlo?**

```
listaCadenas.getFirst();
listaCadenas.get(0);
listaCadenas.element();
```

Las LinkedList están ordenadas, y por tanto el orden se corresponde con el orden en el que se agregan los elementos en la lista. (Verdadero/Falso)

Falso.

Con un Iterador solo puedo moverme hacia delante, pero no hacia atrás (verdadero/falso)

Verdadero.

Con un ListIterator, pega aquí una captura de los métodos que encontrarás en la API de java.

Methods

Modifier and Type	Method and Description
void	`add(E e)` Inserts the specified element into the list (optional operation).
boolean	`hasNext()` Returns `true` if this list iterator has more elements when traversing the list in the forward direction.
boolean	`hasPrevious()` Returns `true` if this list iterator has more elements when traversing the list in the reverse direction.
E	`next()` Returns the next element in the list and advances the cursor position.
int	`nextIndex()` Returns the index of the element that would be returned by a subsequent call to `next()`.
E	`previous()` Returns the previous element in the list and moves the cursor position backwards.
int	`previousIndex()` Returns the index of the element that would be returned by a subsequent call to `previous()`.
void	`remove()` Removes from the list the last element that was returned by `next()` or `previous()` (optional operation).
void	`set(E e)` Replaces the last element returned by `next()` or `previous()` with the specified element (optional operation).

¿Qué método de ListIterator me permite saber si hay algún elemento más hacia atrás? Y ¿Cuál para obtener el valor del elemento anterior?.

```
hasPrevious();
previous();
```

Dado el siguiente ejemplo del video. Modifícalo para que agregue a Miguel después de Sandra, utilizando un ListIterator.

```
public static void main(String[] args) {
  LinkedList<String> personas = new LinkedList<String>();
  personas.add("Pepe");
  personas.add("Sandra");
  personas.add("Ana");
  personas.add("Laura");
  System.out.println(personas.size());

  //Introducir aquí el código

  ListIterator lit = personas.ListIterator(2);

  lit.add("Miguel");

  for (String persona : personas) {
    System.out.println(persona);
  }
}
```