**PRÁCTICA 14 | LISTAS SIMPLES LIGADAS**

# Introducción

Hemos dado un paso más en el tipo de estructuras de datos que se están manejando. Ahora toca trabajar con memoria dinámica y un correcto manejo de los apuntadores en el caso de este tipo de listas, para evitar el típico *NullPointerException.*

Voy a utilizar también una clase que codifiqué en las primeras prácticas que hice, espero no tener problemas con ella en el caso de que haya hecho algo mal.

Tengo ya una idea de cómo hacer el código, pero me limitaré a verificar de forma gráfica también aplicando un análisis, para asegurar mi hipótesis.

# Desarrollo

En esta práctica se creó una nueva clase de nombre **ListaOrdenada** que implementa la interfaz **Lista**, que fuecreada desde las primeras sesiones de este curso de Estructura de Datos. Como su nombre lo dice, la nueva clase almacena todo tipo de datos, pero de una forma ordenada (ascendente o descendente) utilizando memoria dinámica.

Procuré realizar un análisis un poco extenso para los métodos que pensé eran más complicados de codificar, tal es el caso de agregar y eliminar. En el caso, por ejemplo, del método invertir omití este paso y como resultado tuve que detenerme de nuevo a analizar un poco porque pensé que sería un poco más sencillo.

Hubo algunos otros métodos que sí resultaron bastante sencillos de programar, ya sea porque se apoyaban de algunos ya creados o porque su código no era tan extenso. Otro punto a favor para agilizar la realización de la práctica, fue que no son permitidos los valores repetidos así que, las funciones que tienen como objetivo contar el número de ocurrencias de un elemento en la lista, se limitaron a comprobar solamente si existía o no el valor a buscar.

Se utilizaron también recursos creados anteriormente, siendo **CriterioOrden** y **Comparador** piezas clave en el funcionamiento de **ListaOrdenada.**

A continuación, se indican los archivos creados o modificados, sus métodos y el paquete donde se encuentran. **Las funciones que aparecen subrayadas es porque están implementando un método de la interfaz “Lista”.**

## ListaOrdenada.java (paquete edlineal).

* public ListaOrdenada(CriterioOrden tipoOrdenamiento).
* public boolean llena().
* public boolean vacia().
* public boolean agregar(Object valor).
* private boolean agregarAscendente(Nodo nuevo).
* private boolean agregarDescendente(Nodo nuevo).
* public void imprimirR().
* public Object buscar(Object valor).
* public Object eliminar(Object valor).
* public boolean setElemento(Object valorV, Object valorN, int numOcurrencias).
* public Arreglo buscarOcurrencias(Object valor).
* public void vaciar().
* public boolean agregarArreglo(Arreglo arreglo2).
* public void invertir().
* public int contar(Object valor).
* public Object eliminarTope().
* public void imprimir().
* public void rellenar(Object valor, int cuantos).
* public Object getTope()
* public int getSize().
* private Pila getPilaElementos().
* public CriterioOrden getTipoOrdenamiento().

## ListaLigada.java (paquete edlineal).

* public boolean llena().

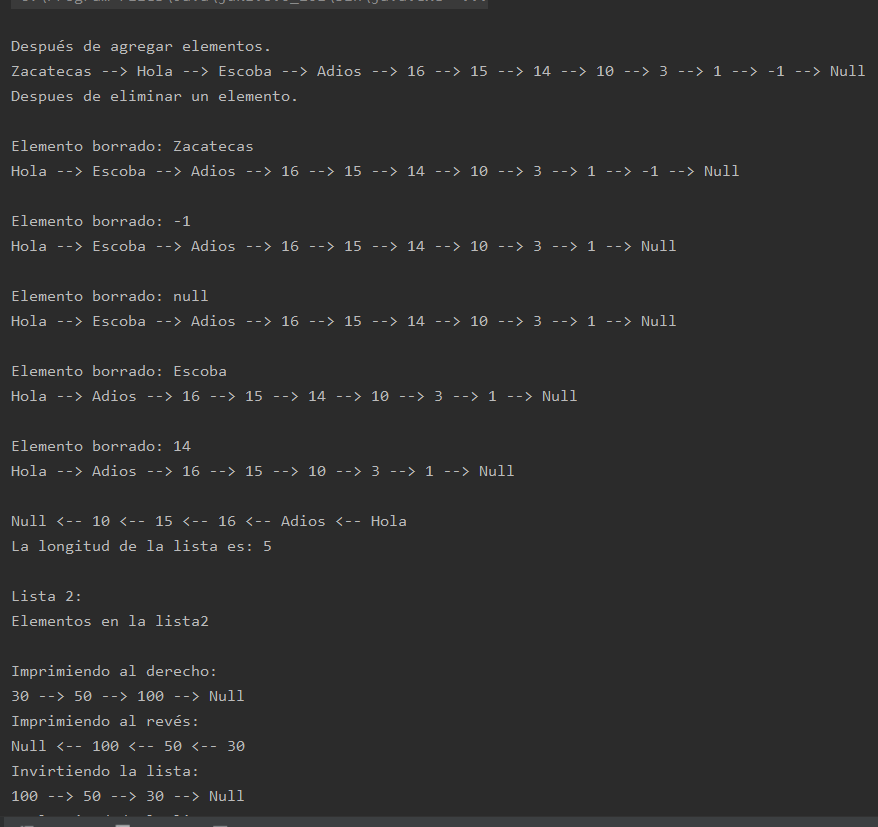
## PruebaListaOrdenada.java (paquete interfaces).

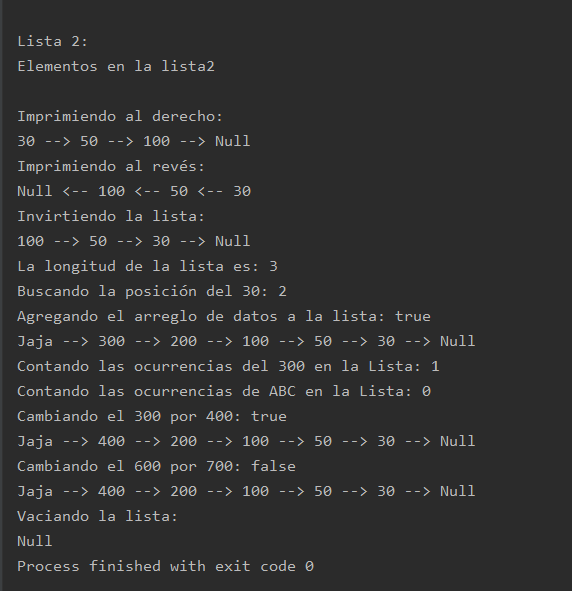
La clase anterior es donde se encuentran las pruebas de esta práctica.

**Para mayor información sobre los métodos y la clases creadas vaya a las siguientes ruta dentro del proyecto:**

* **./docs/edlineal/ListaOrdenada.html**
* **./docs/edlineal/ListaLigada.html**

# Captura de pantalla del programa funcionando





# Conclusiones

El manejar este tipo de listas se torna bastante confuso cuando se desea recorrerlas al momento de estar cambiando las variables a la liga derecha de cada Nodo. Los análisis me ayudaron bastante para evitar tanta confusión, en especial al agregar y eliminar elementos de la lista. Es interesante estar trabajando con memoria dinámica y saber cómo se maneja, que es muy diferente a como yo pensaba que funcionaba.

En esta práctica no tuve tantos problemas como en algunas anteriores, espero que para las siguientes pueda ocurrir la misma situación con sus correspondientes análisis que ayudan bastante.