

TAD: Tipos Abstractos de Datos

1. Implementar una función en Java utilizando la clase **LinkedList** que cree una lista de N números enteros aleatorios y la retorne como resultado. N deber ser un parámetro de la función. Luego utilizarla y mostrar la lista generada.
Hint: Investigar como se utiliza la clase `java.util.Random`, para generar numeros aleatorios.
2. Implementar una función en Java que dada una lista de N números enteros y un número, retorne verdadero si el valor recibido como parámetro se encuentra en la lista, falso en caso contrario.
3. Investigar en que consiste el método de ordenamiento denominado **BubbleSort** u ordenamiento **Burbuja**. Describe paso a paso como ordenaria este método el siguiente arreglo: `[7, 9, 3, 1]`.
4. Implementa en Java una función que implemente el método **BubbleSort** sobre una lista de enteros utilizando la clase **ArrayList**.
5. Definir una interfaz genérica **Lista** con las siguientes operaciones:
 - `agregar(T item)`
 - `eliminar(T item)`
 - `buscar (T item)`
 - `tamaño()`

Crear dos clases que implementen esta interfaz:

- (a) **ListaEnlazada** : La cual implemente el TAD Lista utilizando memoria dinámica.
 - (b) **ListaArreglo** : La cual implemente el TAD Lista utilizando arreglos (memoria estática).
6. Importar la clase **Stack** de Java (`import java.util.Stack`) y utilizarla para implementar un programa que dada una secuencia de caracteres, que representa una expresión matemática, verifique si la misma tiene los paréntesis correctamente balanceados.
 - `"(3+4) * 8"` está balanceada.
 - `"((3 + 4) * 8 "` no está balanceada.

- ") 3 + 4 (* 8 " no está balanceada, ya que si bien cada paréntesis de apertura tiene un paréntesis de cierre, no están en el orden correcto.

7. Definir una interfaz genérica **Pila**, con sus operaciones características.

Para unificar notación sugerimos que utilicen los siguientes nombres:

- apilar(T item)
- desapilar()
- tope()
- esVacia()
- vaciar()

8. Crear dos clases que implementen la interfaz Pila, definida anteriormente en 7, utilizando las clases **ListaEnlazada** y **ListaArreglo** implementadas en el ejercicio 5.

9. Reimplementar el ejercicio 6 utilizando cada una de las clases de Pilas implementadas en el ejercicio 7. Verificar que el programa funciona cómo se espera.

10. Definir una interfaz **Cola genérica** con las operaciones características de este TAD. Crear dos clases que implementen esta interfaz, utilizando las clases **ListaEnlazada** y **ListaArreglo** implementadas en el ejercicio 5.

Para unificar notación sugerimos que utilicen los siguientes nombres:

- encolar(T item)
- desencolar()
- esVacia()
- vaciar()

11. Utilizando alguna de las implementaciones del TAD Cola, simule una cola de trabajos de impresión donde se agregan y procesan trabajos en el orden en que llegan. Para ello vamos a simplificar la representación de los trabajos a imprimir en un string que representará el nombre del archivo a procesar:

```
Cola<String> colaImpresion = new Cola<>();

colaImpresion.encolar("Trabajo1.txt");
colaImpresion.encolar("Trabajo2.pdf");
colaImpresion.encolar("Trabajo3.doc");
```

12. Averiguar cómo se implementan las colas con arreglos circulares. Qué ventajas presenta esta implementación?

13. Realizar una implementación en Java de una Cola Circular para enteros, con las siguientes operaciones:

- estaLlena()
- estaVacia()
- encolar(int elemento)
- desencolar()
- imprimir()