TAREA3: “Creación de una pila (LIFO) y una cola (FIFO)”

Contexto: Las pilas y las colas son TADs (tipos abstractos de datos) con innumerables aplicaciones. Se usan en multitud de aplicaciones y procesos de nuestro ordenador:

- Pila de llamadas a métodos
- Gestión del historial de acciones
- Navegación Sistemas Operativos (round robin)

LIFO (PILA): Last In First Out: El último elemento apilado (push) es el primer elemento en ser desapilado(pop).

FIFO (COLA): First In First Out: El primer elemento apilado (push) es el primer elemento en ser desapilado(pop).

Métodos comunes:

- **push**. Añade un elemento nuevo a la pila.
- **pop**. Retira un elemento de la pila.
- **peek**. Devuelve el elemento que eliminaría pop, pero no en esta ocasión solo retorna el contenido no lo elimina.
- **size**. Cantidad de elementos en la pila.
- **isEmpty**. Nos indica si la pila está vacía.
- **isFull**. Nos indica si la pila ha alcanzado el número máximo de elementos de tenerlo.

Para implementar se proporciona los siguientes recursos:

Interfaz:

```
public interface TAD {

    public void push(String input) throws TADLleno,TADLLenado;
    public String pop() throws TADVacio, TADVaciado;
    public String peek() throws TADVacio;
    public int size();
    public boolean estaVacia();
    public boolean estaLlena();
    public void listar();

}
```

Excepciones:

- **TADLlenado()** : Enviará el mensaje: “TAD se ha llenado tras la última inserción". Extiende la clase RuntimeException.
- **TADVaciado()** : Enviará el mensaje: “TAD se ha vaciado tras la última retirada". Extiende la clase RuntimeException.
- **TADLleno()** : Enviará el mensaje: “TAD lleno. No es posible insertar más elementos". Extiende la clase Exception.
- **TADVacio()** : Enviará el mensaje: “TAD vacio. No es posible retirar más elementos". Extiende la clase Exception.

Uso de la interfaz:

- **push(String input) : void**. Insertamos un nuevo elemento a la colección (ArrayList) puede suceder las siguientes situaciones:
    - No es posible insertar porque la colección esta llena: lanzamos la excepción TADLleno.
    - Es posible inserta un nuevo elemento, tras lo cual si la cola se ha llenado y no se podrá insertar más elementos: lanzamos la excepción TADLlenado.
- **pop() : String.** Retiramos un elemento del ArrayList función del tipo de TAD. Puede suceder:
    - La pila está completamente vacía por lo que no hay ningún elemento que extraer: lanzamos la excepción TADVacio.
    - Extraemos un elemento y no retornamos, pero también detectamos que si tras esta acción la pila va a quedar completamente vacía: lanzamos TADVaciado.
- **peek() : String.** Devuelve un elemento del ArrayList en función del tipo de TAD pero no retiramos/quitamos de la colección. Puede suceder:
    - La pila está completamente vacía por lo que no hay ningún elemento que extraer: lanzamos la excepción TADVacio
- **size() : int**. Devuelve el número de elementos dentro de la colección actualmente.
- **estaVacia() : boolean**. Retorna true si la colección está vacía sino false.
- **estaLlena() : boolean**. Retorna true si la colección está completamente llena sino false.
- **listar (): void**. Saca por pantalla los elementos dentro de la colección en ese instante.

3.1. Crea una clase llamada "FIFO" que implementa la interfaz TAD conforme a la naturaleza propia de una cola FIFO trabajando con String como elemento a insertar y retirar de la cola, utilizamos un ArrayList (`private ArrayList<String>)`

A parte de los métodos a implementar introducidos por la interfaz, debemos codificar un constructor donde le pasamos el número máximo de elementos permitidos en la cola.

3.2. Crea una clase llamada "LIFO" que implementa la interfaz TAD conforme a la naturaleza propia de una pila LIFO trabajando con String como elemento a insertar y retirar de la cola. utilizamos un ArrayList (`private ArrayList<String>)`

A parte de los métodos a implementar introducidos por la interfaz, debemos codificar un constructor donde le pasamos el número máximo de elementos permitidos en la pila.

3.3. Crea una clase de pruebas "MainTAD" donde mediante polimorfismo de la clase "TAD" crearemos dos objetos uno que represente una cola (clase "FIFO") de 10 elementos y una pila (clase "LIFO") de 15 elementos.

Probar el correcto funcionamiento de cada método y en caso de producir las diversas excepciones mostrar un aviso notificando de dicha excepción mediante el terminal.