Nombre:

< **Bicola (Deque)**>

1. DESCRIPCIÓN DEL TDA:

< Una **bicola** (o **deque**) es un tipo de dato abstracto que permite la inserción y eliminación de elementos tanto por el **frente** como por el **final**. Es una generalización de las pilas y colas, ya que combina las funcionalidades de ambas. Es útil en situaciones donde se necesita flexibilidad para manipular elementos en ambos extremos.>

2. INVARIANTE DE TDA:

< La bicola es una secuencia de elementos donde:

- Los elementos se insertan y eliminan siguiendo el principio **FIFO (First In, First Out)** o **LIFO (Last In, First Out)**, dependiendo de la operación.
- Los elementos pueden ser de cualquier tipo (en este caso, String).
- El tamaño de la bicola es dinámico y puede crecer o reducirse según las operaciones realizadas.

3. OPERACIONES:

Nombre de operación: addFirst

- **Descripción**: Agrega un elemento al frente de la bicola.
- **Descripción operacional**:
  - El elemento se inserta en la posición inicial de la bicola.
  - El resto de los elementos se desplazan hacia la derecha.
- **Precondición**: Ninguna (la bicola puede estar vacía o contener elementos).
- **Postcondición**: El tamaño de la bicola aumenta en 1, y el nuevo elemento es el primero.

Nombre de operación: addLast

- **Descripción**: Agrega un elemento al final de la bicola.
- **Descripción operacional**:
  - El elemento se inserta en la posición final de la bicola.
  - El resto de los elementos permanece en su lugar.
- **Precondición**: Ninguna (la bicola puede estar vacía o contener elementos).
- **Postcondición**: El tamaño de la bicola aumenta en 1, y el nuevo elemento es el último.

Nombre de operación: removeFirst

- **Descripción**: Elimina y devuelve el elemento del frente de la bicola.
- **Descripción operacional**:
  - El primer elemento se elimina de la bicola.
  - El segundo elemento (si existe) pasa a ser el nuevo frente.

- **Precondición**: La bicola no debe estar vacía.
- **Postcondición**: El tamaño de la bicola disminuye en 1, y el primer elemento es eliminado.

Nombre de operación: removeLast

- **Descripción**: Elimina y devuelve el elemento del final de la bicola.
- **Descripción operacional**:
  - El último elemento se elimina de la bicola.
  - El penúltimo elemento (si existe) pasa a ser el nuevo final.
- **Precondición**: La bicola no debe estar vacía.
- **Postcondición**: El tamaño de la bicola disminuye en 1, y el último elemento es eliminado.

Nombre de operación: peekFirst

- **Descripción**: Devuelve el elemento del frente sin eliminarlo.
- **Descripción operacional**:
  - Se accede al primer elemento de la bicola sin modificarla.
- **Precondición**: La bicola no debe estar vacía.
- **Postcondición**: La bicola permanece sin cambios.

Nombre de operación: peekLast

- **Descripción**: Devuelve el elemento del final sin eliminarlo.
- **Descripción operacional**:
  - Se accede al último elemento de la bicola sin modificarla.
- **Precondición**: La bicola no debe estar vacía.
- **Postcondición**: La bicola permanece sin cambios.

Nombre de operación: isEmpty

- **Descripción**: Verifica si la bicola está vacía.
- **Descripción operacional**:
  - Se comprueba si el tamaño de la bicola es 0.
- **Precondición**: Ninguna.
- **Postcondición**: Devuelve true si la bicola está vacía, false en caso contrario.

Nombre de operación: size

- **Descripción**: Devuelve el número de elementos en la bicola.
- **Descripción operacional**:
  - Se cuenta el número de elementos almacenados en la bicola.
- **Precondición**: Ninguna.
- **Postcondición**: Devuelve un entero que representa el tamaño de la bicola.

Código con Precondiciones y Postcondiciones

```
import java.util.Deque;
import java.util.LinkedList;
import java.util.Scanner;

public class Bicola {

    public static void main(String[] args) {
        Deque<String> deque = new LinkedList<>();
        Scanner scanner = new Scanner(System.in);
        String comando;

        System.out.println("Bienvenido al Sistema de Bicola (Deque)");
        System.out.println("Comandos disponibles:");
        System.out.println("  - ADD_FIRST [nombre]: Agrega un elemento al frente.");
        System.out.println("  - ADD_LAST [nombre]: Agrega un elemento al final.");
        System.out.println("  - REMOVE_FIRST: Elimina el elemento del frente.");
        System.out.println("  - REMOVE_LAST: Elimina el elemento del final.");
        System.out.println("  - PEEK_FIRST: Muestra el elemento del frente.");
        System.out.println("  - PEEK_LAST: Muestra el elemento del final.");
        System.out.println("  - MOSTRAR: Muestra todos los elementos.");
        System.out.println("  - SALIR: Finaliza el programa.");
        System.out.println("-------------------------------------------");

        while (true) {
            System.out.print("> ");
            comando = scanner.nextLine();

            if (comando.toUpperCase().startsWith("ADD_FIRST")) {
                // Precondición: Ninguna
                String elemento = comando.substring(10).trim();
                if (!elemento.isEmpty()) {
                    deque.addFirst(elemento);
                    System.out.println("Elemento agregado al frente: " + elemento);
```

```
            } else {
                System.out.println("Error: Debes ingresar un nombre después de 'ADD_FIRST'.");
            }
            // Postcondición: El tamaño de la bicola aumenta en 1, y el nuevo elemento es el primero.
        } else if (comando.toUpperCase().startsWith("ADD_LAST")) {
            // Precondición: Ninguna
            String elemento = comando.substring(9).trim();
            if (!elemento.isEmpty()) {
                deque.addLast(elemento);
                System.out.println("Elemento agregado al final: " + elemento);
            } else {
                System.out.println("Error: Debes ingresar un nombre después de 'ADD_LAST'.");
            }
            // Postcondición: El tamaño de la bicola aumenta en 1, y el nuevo elemento es el último.
        } else if (comando.equalsIgnoreCase("REMOVE_FIRST")) {
            // Precondición: La bicola no debe estar vacía.
            if (!deque.isEmpty()) {
                System.out.println("Elemento eliminado del frente: " + deque.removeFirst());
            } else {
                System.out.println("La bicola está vacía.");
            }
            // Postcondición: El tamaño de la bicola disminuye en 1, y el primer elemento es eliminado.
        } else if (comando.equalsIgnoreCase("REMOVE_LAST")) {
            // Precondición: La bicola no debe estar vacía.
            if (!deque.isEmpty()) {
                System.out.println("Elemento eliminado del final: " + deque.removeLast());
            } else {
                System.out.println("La bicola está vacía.");
            }
            // Postcondición: El tamaño de la bicola disminuye en 1, y el último elemento es eliminado.
```

```
        } else if (comando.equalsIgnoreCase("PEEK_FIRST")) {
            // Precondición: La bicola no debe estar vacía.
            if (!deque.isEmpty()) {
                System.out.println("Elemento en el frente: " + deque.peekFirst());
            } else {
                System.out.println("La bicola está vacía.");
            }
            // Postcondición: La bicola permanece sin cambios.
        } else if (comando.equalsIgnoreCase("PEEK_LAST")) {
            // Precondición: La bicola no debe estar vacía.
            if (!deque.isEmpty()) {
                System.out.println("Elemento en el final: " + deque.peekLast());
            } else {
                System.out.println("La bicola está vacía.");
            }
            // Postcondición: La bicola permanece sin cambios.
        } else if (comando.equalsIgnoreCase("MOSTRAR")) {
            // Precondición: Ninguna.
            if (!deque.isEmpty()) {
                System.out.println("Elementos en la bicola: " + deque);
            } else {
                System.out.println("La bicola está vacía.");
            }
            // Postcondición: Ninguna.
        } else if (comando.equalsIgnoreCase("SALIR")) {
            System.out.println("Saliendo del sistema. ¡Hasta luego!");
            scanner.close();
            return;
        } else {
            System.out.println("Comando no válido. Intenta de nuevo.");
        }
    }
  }
}
```