**Guía de Implementación y Uso: Lista Enlazada en Java**

**1. Cómo Ejecutar el Código**

**Requisitos**

* Java JDK 8 o superior instalado.
* Editor de código (Eclipse, IntelliJ, VSCode) o terminal.

**Pasos**

1. **Crear los archivos**:
   * Node.java (contiene la clase Node).
   * LinkedList.java (contiene la clase LinkedList con los métodos).
   * Main.java (contiene el menú interactivo).
2. **Compilar y ejecutar**:

bash

Copy

javac Node.java LinkedList.java Main.java

java Main

*En un IDE, simplemente ejecuta Main.java.*

1. **Interactuar con el menú**:
   * Se mostrarán opciones para manipular la lista.
   * Ingresa números según las operaciones deseadas.

**2. Propósito de Cada Clase y Método**

**Clase Node**

* **Propósito**: Representa un nodo de la lista enlazada.
* **Atributos**:
  + int data: Almacena el valor (ej: dígitos de un DPI).
  + Node next: Referencia al siguiente nodo.

**Clase LinkedList**

| **Método** | **Propósito** | **Complejidad** |
| --- | --- | --- |
| add(int data) | Agrega un nodo al final. | O(n) |
| addFirst(int data) | Agrega un nodo al inicio. | O(1) |
| addMiddle(int data, int pos) | Inserta un nodo en posición específica. | O(n) |
| remove(int data) | Elimina la primera ocurrencia del valor. | O(n) |
| printList() | Imprime todos los valores de la lista. | O(n) |
| reverse() | Invierte el orden de la lista. | O(n) |
| contains(int value) | Verifica si un valor existe. | O(n) |

**3. Ejemplos de Entradas y Salidas**

**Ejemplo 1: Operaciones Básicas**

**Entradas (menú interactivo)**:

1. Agregar 2, 3, 6 (al final).
2. Agregar 1 (al inicio).
3. Insertar 9 en posición 2.
4. Eliminar 3.
5. Revertir lista.

**Salida**:

Copy

Lista inicial: 2 3 6

Después de addFirst(1): 1 2 3 6

Después de addMiddle(9, 2): 1 2 9 3 6

Después de remove(3): 1 2 9 6

Después de reverse(): 6 9 2 1

**Ejemplo 2: Búsqueda con contains()**

**Entrada**:

java

Copy

LinkedList list = new LinkedList();

list.add(2); list.add(3); list.add(6);

boolean existe = list.contains(3); // true

boolean noExiste = list.contains(5); // false

**Salida**:

Copy

¿El 3 está en la lista? true

¿El 5 está en la lista? false

**Ejemplo 3: Gestión de Memoria**

**Proceso**:

1. Crear nodos: [2|•] → [3|•] → [6|•]
2. Eliminar 3: [2|•] → [6|•]
   * El nodo [3] queda sin referencias y es eliminado por el *Garbage Collector*.

**4. Diagrama de Flujo**

mermaid

Copy

graph TD

A[Inicio] --> B[Crear lista]

B --> C{Menú}

C --> |Opción 1| D[addFirst]

C --> |Opción 2| E[add]

C --> |Opción 3| F[addMiddle]

C --> |Opción 4| G[remove]

C --> |Opción 5| H[reverse]

C --> |Opción 6| I[contains]

C --> |Opción 0| J[Salir]

**5. Notas Adicionales**

* **Caso de error**: Si se ingresa una posición inválida en addMiddle(), muestra:

Copy

Posición inválida

* **Eficiencia**:
  + Operaciones en extremos (addFirst, remove en head) son O(1).
  + Operaciones que recorren la lista (add, contains) son O(n).