```
class Nodo {
  constructor(dato) {
  this.dato = dato;
    // Guarda el valor del nodo

    this.siguiente = null;
    // Apunta al siguiente nodo.
    // Inicia en null porque aún no está enlazado.
  }
}

// Clase ListaEnlazada
// Controla toda la estructura
class ListaEnlazada {

  constructor() { this.cabeza
    = null;
    // Primer nodo de la lista.
    // Si es null, la lista está vacía.

    this.tamaño = 0;
    // Lleva el conteo de nodos.
```

```
}

// Verifica si la lista está vacía
estaVacia() {
  return this.cabeza === null;
}

// Inserta un nodo al inicio
// Lógica:
// 1. Crear nodo
// 2. Apuntarlo hacia la cabeza actual
// 3. Mover la cabeza al nuevo nodo
// Complejidad: O(1)
insertarInicio(valor) {
  const nuevoNodo = new Nodo(valor);

  nuevoNodo.siguiente = this.cabeza;
  // El nuevo nodo apunta al que era primero

  this.cabeza = nuevoNodo;
  // Ahora el nuevo nodo es la cabeza

  this.tamaño++;
}
```

```
// Inserta un nodo al final
// Lógica:
// - Si la lista está vacía → el nodo es la cabeza
// - Si no → recorrer hasta el último nodo
// Complejidad: O(n)
insertarFinal(valor) {

  const nuevoNodo = new Nodo(valor);

  // Caso especial: lista vacía if
  (this.estaVacia()) { this.cabeza
  = nuevoNodo;
  } else {

    let actual = this.cabeza;

    // Recorre hasta encontrar el último nodo
    while (actual.siguiente !== null) {
      actual = actual.siguiente;
    }

    // El último nodo apunta al nuevo
    actual.siguiente = nuevoNodo;
  }
```

```
  this.tamaño++;
}

// Elimina el primer nodo
// Lógica:
// Solo mover la cabeza al siguiente nodo
// Caso especial porque no hay nodo anterior
// Complejidad: O(1)
eliminarInicio() {

  if (this.estaVacia()) { console.log("La
    lista está vacía"); return null;
  }

  const valorEliminado = this.cabeza.dato;

  this.cabeza = this.cabeza.siguiente;
  // La cabeza ahora es el segundo nodo

  this.tamaño--;

  return valorEliminado;
}
```

```
// Busca un valor en la lista
// Lógica:
// Recorrer nodo por nodo comparando datos
// Complejidad: O(n)
buscar(valor) {

  let actual = this.cabeza;

  while (actual !== null) {

    if (actual.dato === valor) { return
      true;
    }

    actual = actual.siguiente;
  }

  return false;
}

// Imprime la lista
// Lógica:
// Recorrer la lista y construir una cadena
imprimir() {
```

```
    let actual = this.cabeza;
    let resultado = "";

    while (actual !== null) {
      resultado += `[${actual.dato}] -> `;
      actual = actual.siguiente;
    }

    resultado += "null";

    console.log(resultado);
  }
}


// Programa de prueba
// Demuestra TODAS las operaciones

const lista = new ListaEnlazada();

// Insertar elementos
lista.insertarInicio(7);
lista.insertarInicio(3);
lista.insertarFinal(15);
lista.insertarFinal(23);
lista.insertarFinal(42);
```

```
// Mostrar
lista
lista.impri
mir();
// [3] -> [7] -> [15] -> [23] -> [42] -> null


// Buscar elementos
console.log("¿Existe 15?: ", lista.buscar(15)); // true
console.log("¿Existe 99?: ", lista.buscar(99)); // false


// Eliminar inicio
console.log("Elemento eliminado:", lista.eliminarInicio());

lista.imprimir();
// [7] -> [15] -> [23] -> [42] -> null
```