Diseño de Sistema: Lista con Acceso O(1) al Final

Diagrama de la nueva estructura

Antes solo tenía una referencia a la cabeza, lo que obligaba a recorrer toda la lista para insertar al final. La mejora fue agregar una referencia extra llamada cola, que siempre apunta al último nodo.

Diagrama ASCII

cabeza → [10 | *] → [20 | *] → [30 | null]

↑

cola

Lista vacía

cabeza → null

cola → null Un

solo nodo

cabeza → [10 | null]

↑

Cola

Código modificado con referencia a cola

Clase Nodo

```
class Nodo{
    int dato;
    Nodo siguiente;

    public Nodo(int dato){
        this.dato = dato;
        this.siguiente = null;
```

```
    }
}
ListaEnlazada mejorada public
class ListaEnlazada {

    private Nodo cabeza;
    private Nodo cola; private
    int tamaño;

    public ListaEnlazada(){
        cabeza = null;
        cola = null;
        tamaño = 0;
    }

    // Insertar al inicio
    public void insertarInicio(int dato){
        Nodo nuevo = new Nodo(dato);

        if(cabeza == null){
            cabeza = nuevo; cola
            = nuevo;
        }else{
            nuevo.siguiente = cabeza;
            cabeza = nuevo;
        }
```

```
    tamaño++;
}


// Ahora es O(1)
public void insertarFinal(int dato){
    Nodo nuevo = new Nodo(dato);

    if(cabeza == null){
        cabeza = nuevo; cola
        = nuevo;
    }else{
        cola.siguiente = nuevo; cola
        = nuevo;
    }

    tamaño++;
}

// Eliminar inicio
public void eliminarInicio(){
```

```
        if(cabeza == null){
            return;
        }

        cabeza = cabeza.siguiente;

        if(cabeza == null){ cola
            = null;
        }

        tamaño--;
    }

    // Buscar
    public boolean buscar(int dato){ Nodo
        actual = cabeza;

        while(actual != null){
            if(actual.dato == dato){
                return true;
            }
            actual = actual.siguiente;

           return false;
    }
    public int getTamaño(){
        return tamaño;
    }
}
```

Análisis de complejidad

Operación	Antes	Después	Motivo
insertarInicio	O(1)	O(1)	No cambió
insertarFinal	O(n)	O(1)	Ya no se recorre la lista
eliminarInicio	O(1)	O(1)	Solo se mueve la cabeza
buscar	O(n)	O(n)	Requiere recorrido

Mejora clave:

Se sacrificó un poco más de memoria (una referencia extra), pero se ganó mucho rendimiento en inserciones al final.

Trade-off clásico de ingeniería: más memoria → menos tiempo de ejecución.

Pruebas que demuestran que insertarFinal() es O(1)

Prueba lógica

Si la lista tiene 1 elemento o 1 millón:

Pasos que realiza el método:

Crear nodo

Apuntar cola.siguiente al nuevo nodo

Mover cola

Siempre son los mismos pasos → tiempo constante.

Prueba de uso

```
public static void main(String[] args){

    ListaEnlazada lista = new ListaEnlazada();


    lista.insertarFinal(10);
    lista.insertarFinal(20);
    lista.insertarFinal(30);
    lista.insertarFinal(40);
```

```
    System.out.println("Tamaño: " + lista.getTamaño());
    System.out.println("Buscar 30: " + lista.buscar(30));
}
```

Resultado esperado:

Tamaño: 4

Buscar 30: true

Sin importar cuánto crezca la lista, insertar al final mantiene el mismo costo. Conclusión del diseño

Agregar la referencia cola fue una mejora simple pero muy poderosa. Me ayudó a entender que muchas optimizaciones no requieren estructuras más complejas, sino guardar información estratégica.

Principal aprendizaje:

Diseñar estructuras de datos no es solo hacer que funcionen, sino pensar cómo escalarán cuando el sistema crezca.