

R1 Unidad 2

"Implementación de una Lista Simplemente Enlazada con Nodos en Java"

2: Corregir las siguientes Clases en Java
▶ Nodo y Lista Enlazada

• Clase Nodo

```
Public class Nodo <T> {
```

```
    Private T dato;
```

```
    Private Nodo <T> siguiente;
```

```
    Public Nodo(T dato) {
```

```
        this.dato = dato;
```

```
        this.siguiente = null;
```

```
    }
```

```
    Public T getData() {
```

```
        return dato;
```

```
    }
```

```
    Public void setData(T dato) {
```

```
        this.dato = dato;
```

```
    }
```

```
    Public Nodo <T> getSiguiente() {
```

```
        return siguiente;
```

```
    }
```

```
    Public void setSiguiente(Nodo <T> siguiente) {
```

```
        this.siguiente = siguiente;
```

```
    }
```

```
}
```

	Estructura de datos	CyT10141
25 / 11 / 25	Elsy Joselyn Godínez Juárez	

Clase ListaEnlazada

```
Public class ListaEnlazada<T> {
```

```
    Private Nodo<T> cabeza;
```

```
    // Constructor
```

```
    Public ListaEnlazada() {
```

```
        this.cabeza = null;
```

```
    }
```

```
    // 3.1 Agregar al inicio
```

```
    Public void agregarInicio(T dato) {
```

```
        Nodo<T> nuevoNodo = new Nodo<>(dato); // Crear NN
```

```
        nuevoNodo.setSiguiente(cabeza); // NN.siguiente
```

```
        cabeza = nuevoNodo;
```

```
    }
```

```
    // 3.2 Eliminar al inicio
```

```
    Public void eliminarAlInicio() {
```

```
        if (cabeza != null) {
```

```
            cabeza = cabeza.getSiguiente();
```

```
        } else {
```

```
            System.out.println("La lista está vacía. No se puede eliminar.");
```

```
        }
```

```
    } // 3.3 Imprimir lista
```

```
    Public void imprimirLista() {
```

```
        Nodo<T> actual = cabeza;
```

```
        While (actual != null) {
```

```
            System.out.print(actual.getDato() + " -> ");
```

```
            actual = actual.getSiguiente();
```

```
        }
```

```
        System.out.print("null");
```

```
    }
```

```
    // setter y getters
```

```
    Public Nodo getSiguiente() {
```

```
        return siguiente;
```

Buen por la estructura pero hace falta implementar setter y getters

3: Operaciones Clave con Representación Visual

Realiza el dibujo de manera visual en tu cuaderno

Cabeza



[5] → [10] → [20] → null

Crear NN

NN = [3] → null ?

NN.siguiente = Cabeza

NN



[3] → [5] → [10] → [20] → null

Cabeza = NN

Cabeza



[3] → [5] → [10] → [20] → null

Implementación en Java

```
Public void agregarAlInicio(int dato) {
```

```
// Crear el nuevo nodo (NN)
```

```
Nodo nuevoNodo = new Nodo(dato);
```

```
// NN.siguiente que apunta en la cabeza
```

```
nuevoNodo.setSiguiente(cabeza);
```

```
// Actualizar cabeza NN
```

```
cabeza = nuevoNodo;
```

```
}
```

3.2 Eliminar al inicio (eliminar Al inicio)

► Antes de eliminar

Cabeza



[10] → [20] → [30] → null

► Después de eliminar al inicio

Cabeza



[20] → [30] → null

"Implementación en Java"

```
Public void eliminarAlInicio() {
```

```
    if (Cabeza != null) {
```

```
        Cabeza = Cabeza.getSiguiete();
```

```
    } else {
```

```
        System.out.println("Error: la lista esta vacia, no puede eliminarse.");
```

```
    }
```

```
}
```

3.3 Recorrer / Imprimir (imprimir lista())

Cabeza



[10] → [20] → [30] → null

Recorrido: 10 → 20 → 30 → null

"Implementación en Java"

```
Public void imprimirLista() {
```

```
    Nodo actual = Cabeza;
```

```
    while (actual != null) {
```

```
        System.out.print(actual.getData() + " → ");
```

```
        actual = actual.getSiguiete();
```

```
    }
```

```
    System.out.print("null");
```

Elsy Joselyn Godínez Juárez
Estructura de datos.

C1110141

26

11

25

4= Caso Práctico: Gestión de un Historial de Transacciones

```
Public class Main {
```

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

```
        // inicializa la lista
```

```
        ListaEnlazada lista = new ListaEnlazada();
```

```
        // Agregar transacciones al inicio
```

```
        System.out.print("--- Agregando Transacciones ---");
```

```
        lista.agregarAlInicio(150);
```

```
        lista.agregarAlInicio(45);
```

```
        lista.agregarAlInicio(200);
```

```
        lista.agregarAlInicio(75);
```

```
        // Resultado
```

```
        // 75 → 200 → 45 → 150 → null
```

```
        // Paso 3 Imprimir
```

```
        System.out.println("Historial de transacciones después de agregar:");
```

```
        lista.imprimir(lista);
```

```
        // Paso 4
```

```
        System.out.println("\n--- Cancelando la Transaccion mas Reciente (75)");
```

```
        lista.eliminarAlInicio();
```

```
        // Paso 5
```

```
        System.out.println("Después de eliminar la transaccion mas reciente");
```

```
        lista.imprimir(lista);
```

```
    }
```

```
}
```

// Getter y Setter

```
public Nodo<T> getCabeza() {  
    return cabeza;
```

```
}
```

```
public void setCabeza (Nodo<T> cabeza) {  
    this.cabeza = cabeza;
```

```
}
```

```
System.out.println("In --- Desafio ---");
```

// Insertar con propiedad

```
mensaje.insertarConPropiedad("URGENTE: Servidor Caído");
```

```
mensaje.insertarConPropiedad("Recordatorio de fatura");
```

```
mensaje.insertarConPropiedad("URGENTE: Base de Datos");
```

// Mostrar lista

```
mensaje.imprimirLista();
```

```
mensaje.imprimirListaAtras();
```

// Eliminar mensaje que contenga Pago

```
System.out.println("Eliminando mensajes con palabra clave: Pago");
```

```
mensaje.eliminarPorPalabraClave("Pago");
```

```
mensaje.imprimirLista();
```

```
int total = mensaje.contarNodos();
```

```
System.out.println("total de mensajes pendientes: " + total);
```

Cabeza



[5] + [10] + [20] → null

Crear NN

NN = [3]

NN.Siguiente = Cabeza

NN



[3] + [5] + [10] + [20] → null

Cabeza = NN

Cabeza



[3] + [5] + [10] + [20] → null

Muy borroso por la representación visual