

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;

ok

}

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;

}

}

25-11-25

Estructura de datos

CyT1D141

Elsy Joselyn Godínez Juárez

• Clase ListaEnlazadas

```
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<T>(dato); // Crear NN
```

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

```
    cabeza = nuevoNodo;
```

```
}
```

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

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

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

```
        Cabeza = Cabeza.getSigiente();
```

```
} else {
```

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

```
}
```

```
    } Nodo<T> nuevo = new Nodo<T>(dato);
```

```
// 3.3 Imprimir lista
```

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

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

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

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

```
        actual = actual.getSigiente();
```

```
}
```

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

```
}
```

```
// seter y getters
```

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

```
    return siguiente;
```

deko°

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

NM



[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 apunte en la cabeza

nuevoNodo.setSiguiente(cabeza);

// Actualizar cabeza NN

cabeza = nuevoNodo;

}

3.2 Eliminar al inicio (eliminarAlInicio())

► Antes de eliminar

Cabeza



[10] → [20] → [30] → null

► Despues de eliminar al inicio

Cabeza



[20] → [30] → null

“Implementación en Java”

```
Public void eliminarAlInicio() {  
    if (cabeza != null) {
```

```
        Cabeza = Cabeza.getSigiente();
```

```
    } else {
```

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

```
}
```

```
}
```

3.3 Recorrer / Imprimir (imprimirLista())

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.getSigiente();
```

```
}
```

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

EndeKro°

4: Caso Práctico : Gestión de un Histórico de Transacciones

Public class Main {

 Public static void main (String [] args) {

 // inicializa la lista

 ListaEnlazada lista = new ListaEnlazada ();

 // Agregar transacciones al inicio

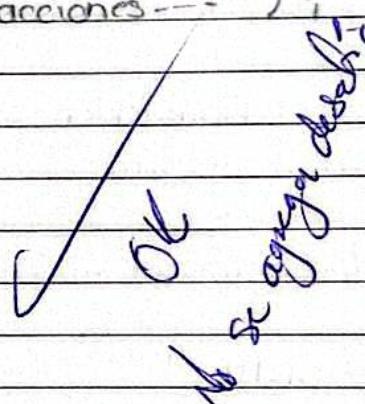
 System.out.print ("--- Agregando Transacciones --- ");

 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.ImprimirLista ();

 // Paso 4

 System.out.println ("\n--- Cancelando la Transacción más Reciente (75) --- ");

 lista.eliminarAlInicio ();

 // Paso 5

 System.out.println ("Después de eliminar la transacción más reciente");

 lista.ImprimirLista ();

 }

}

// Getter y Setter

```
public Nodo<T> getCabecera(){  
    return cabecera;  
}
```

del

```
public void setCabecera(Nodo<T> Cabecera){  
    this.cabecera = Cabecera;  
}
```

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

// Insertar con propiedad

```
mensaje.insertarConPropiedad("URGENTE: Servidor Caido");  
mensaje.insertarConPropiedad("Recordatorio de factura");  
mensaje.insertarConPropiedad("URGENTE: Base de Datos");
```

// Mostrar lista

```
mensaje.ImprimirLista();  
mensaje.ImprimirHaciaAtras();
```

// Eliminar mensajes que contenga "Pago";

```
System.out.println("In 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

no borrar
por la repeticion