



**Universidad Tecnológica  
del Norte de Guanajuato**  
Organismo Público Descentralizado del Gobierno del Estado  
“Educación y progreso para la vida”

## TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIÓN

Licenciatura en Ingeniería en Tecnologías de la Información e Innovación Digital  
(DSM)

### ESTRUCTURA DE DATOS

#### UNIDAD II

R1

Grupo: GTID0141

Alumno: Avalos Melado Rey Gerardo

No.Control: 1224100509

Docente: Gabriel Barron Rodriguez

**Lista doblemente  
enlazada**

## Ricardo León Medrano

### Práctica 2 - Listas doblemente enlazadas

#### 1- Conceptos fundamentales

clase Nodo <T>

```
public class Nodo <T> {
```

```
private T dato;
```

```
private Nodo <T> siguiente;
```

```
private Nodo <T> anterior;
```

```
public Nodo (T dato) {
```

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

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

```
this.anterior = 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; }
```

```
public Nodo <T> getAnterior () { return anterior; }
```

```
public void setAnterior (Nodo <T> anterior) { this.anterior = anterior; }
```

3

#### 2- Clase ListaDoble <T>

```
public class ListaDoble <T> {
```

```
private Nodo <T> cabecera;
```

```
private Nodo <T> coda;
```

```
public ListaDoble () {
```

```
this.cabecera = null;
```

```
this.coda = null;
```

3

3. Operaciones clave Agregar al inicio  
3.1. Public void agregarAlInicio(T dato) {  
    Nodo < T > nuevo = new Nodo < T > (dato);

    if (cabecera == null) {

        cabecera = nuevo;

        cola = nuevo;

    } else {

        nuevo.setAnterior(cabecera);

        cabecera.setSiguiente(nuevo);

        cabecera = nuevo;

    }

3.2 Eliminar al final

public void eliminarAlFinal() {

    if (cola != null) {

        if (cabecera == cola) {

            cabecera = null;

            cola = null;

        } else {

            cola.setAnterior(null);

            cola.setSiguiente(null);

        }

    } else {

        System.out.println("La lista està vacia");

    }

}

### 3.3 Imprimir hacia adelante

```
public void imprimirLista() {  
    Nodo <--> actual = cabecera;  
    System.out.println("[" + lista.adelante + "]");  
    while (actual != null) {  
        System.out.println(actual.getData() + " ->");  
        actual = actual.getSiguiente();  
    }  
    System.out.println("null");  
}
```

### 3.4 Imprimir hacia atrás

```
public void imprimirListaAtras() {  
    Nodo <--> actual = cola;  
    System.out.println("[" + lista.cabecera + "]");  
    while (actual != null) {  
        System.out.println(actual.getData() + " ->");  
        actual = actual.getAnterior();  
    }  
    System.out.println("null");  
}
```

### 4. Caso Práctico: Gestión de marca de mensajes.

```
public class Main {  
    public static void main (String [] args) {  
        ListaDoble<String> mensajes = new ListaDoble<>();  
  
        msj.agregarAlFinal ("enca grifo");  
        msj.agregarAlFinal ("Auto de gaso");  
        msj.agregarAlFinal ("Resumen Dime");  
    }  
}
```

`` imprimir cliente  
System.out.println ("mensaje Cliente");  
msg\_imprimirLista();  
`` imprimir cajas  
System.out.println ("mensaje Cajas");  
msg\_imprimirLista();  
`` procesar el mensaje más antiguo  
System.out.println ("Procesando mensaje más antiguo");  
msg\_clonarMmail();  
`` imprimir cliente cliente  
System.out.println ("mensajes despues de eliminar");  
msg\_imprimirLista();

3  
5