Miguel Lozano Arvizu GTID141

02

Implementacion de una lista Doblemente Enlazada con Nodos en Java.

2. Estructura de Clases en Java. - Implementar Tipo Genericos (<T>) y Encapsulamiento.

- 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; // Nuevo
  }
    // Setter and Getter.
}
```

- Clase ListaDoble <T>

```
public class ListaDoble <T> {
  private Nodo <T> cabeza;
  private Nodo <T> cola;

  public ListaDoble (){
    this.cabeza = null;
    this.cola = null; // Nuevo
  }
      // Operaciones.
}
```

3. Operaciones clave - Actualizar ambas referencias.

3.1 Agregar al inicio:

```
public void agregarInicio(T dato) {
    Nodo<T> nuevo = new Nodo<>(dato);
    if (cabeza == null) {
        cabeza = nuevo;
        cola = nuevo;
    } else {
        nuevo.setSiguiente(cabeza);
        cabeza.setAnterior(Nuevo);
        cabeza = Nuevo;
    }
```

3.2 - Eliminar al final

```
public void eliminarFinal() {
    if (cola = null) {
        System.out.println("Lista vaciaa :c");
        return;
    }
    if (cabeza == cola) {
        cabeza = null;
        cola = null;
    } else {
        cola = cola.getAnterior();
        cola = cola.getSiguiente(null);
    }
}
```

3.3 Recorrer / Imprimir hacia atras.

```
public void imprimirHaciaAtras () {
   Nodo <T> actual = cola;
     System.out.println("Lista (Atrás)");
while (actual != null) {
     System.out.println (actual.getDato() + " ___ ");
      actual = actual.getanterior ();
  }
    System.out.println ("null");
}
```

4.. Caso Practico: Gestion de una Cola de Mensajes.

- 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 getDato (){
    return dato;
}
  public void setDato (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;
  }
}
```

— Clase ListaDoble<T> —

```
public class ListaDoble<T>{
    private Nodo<T> cabeza;
    private Nodo<T> cola;

public ListaDoble(){
    this.cabeza = null;
    this.cola = null;
}

// Agregar Inicio

public void agregarAlInicio(T dato){
    Nodo<T> nuevo = new Nodo<>(dato);
    if(cabeza == null){
        cabeza = nuevo;
        cola = nuevo;
    }else{
```

```
nuevo.setSiguiente(cabeza);
cabeza.setAnterior(nuevo);
cabeza = nuevo;
}
}
// Eliminar al final
public void eliminarAlFinal (){
if (cola == null){
System.out.println ("La lista esta vacia");
return;
}
if (cabeza == cola){
cabeza = null;
cola = null;
} else {
cola = cola.getAnterior ();
cola.setSiguiente (null);
}
}
// Imprimir hacia adelante
public void imprimirLista (){
Nodo<T> actual = cabeza;
System.out.println("Lista (Adelante):");
while (actual != null){
System.out.print(actual.getDato() + " _ _ ");
actual = actual.getSiguiente ();
}
System.out.println("null");
}
```

```
// Imprime hacia atras.
public void imprimirHaciaAtras(){
    Nodo<T> actual = cola;
    System.out.print("Lista (Atras): ");
while (actual != null){
    System.out.print(actual.getDato() + " <- ");
    actual = actual.getAnterior();
  }
  System.out.println("null");
 }
}
```

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

ListaDoble<String> mensajes = new ListaDoble<>();

mensajes.agregarAlInicio("Error Critico");
mensajes.agregarAlInicio("Aviso de Pago");
mensajes.agregarAlInicio("Resumen Diario");

System.out.println("Estado Inicial");
mensajes.imprimirLista();
mensajes.imprimirHaciaAtras();
System.out.println("\nProcesando mensaje antiguo...");
mensajes.eliminarAlFinal();
System.out.println("\nEstado despues de eliminar");
mensajes.imprimirLista();
}
}
```