



**UNIVERSIDAD NACIONAL  
AUTÓNOMA DE MÉXICO**



---

**FACULTAD DE ESTUDIOS SUPERIORES  
ARAGÓN**

Nombre del curso:  
**ESTRUCTURA DE DATOS**

Profesor:  
**Jesús Hernández Cabrera**

Tarea:  
**T5, LISTAS DOBLEMENTE LIGADAS**

Alumno:  
**Navarro Rocha Miguel Ángel**

Fecha:  
**06-09-24**

## Código

### Class doublelinkedlist:

```
package Tarea5Package;

public class DoubleLinkedList<T> {
    private NodoDoble<T> head;
    private NodoDoble<T> tail;
    private int tamanio;

    public DoubleLinkedList() {
        this.head = null;
        this.tail = null;
        this.tamanio = 0;
    }

    public boolean estaVacia() {
        return this.head == null && this.tail == null;
    }

    public int getTamanio() {
        return tamanio;
    }

    public void agregarAlInicio(T valor) {
        NodoDoble<T> nuevo = new NodoDoble<>(valor);
        if (this.estaVacia()) {
            this.head = nuevo;
            this.tail = nuevo;
        } else {
            nuevo.setSiguiente(this.head);
            this.head.setAnterior(nuevo);
            this.head = nuevo;
        }
        this.tamanio++;
    }

    public void agregarAlFinal(T valor) {
        NodoDoble<T> nuevo = new NodoDoble<>(valor);
        if (this.estaVacia()) {
            this.head = nuevo;
            this.tail = nuevo;
        } else {
            nuevo.setAnterior(this.tail);
            this.tail.setSiguiente(nuevo);
            this.tail = nuevo;
        }
        this.tamanio++;
    }

    public void agregarDespuesDe(T referencia, T valor) {
        NodoDoble<T> aux = this.head;
        while (aux != null && !aux.getData().equals(referencia)) {
            aux = aux.getSiguiente();
        }
        if (aux == null) {
            System.out.println("No existe la referencia!");
        } else {
            NodoDoble<T> nuevo = new NodoDoble<>(valor);
            nuevo.setAnterior(aux);
            nuevo.setSiguiente(aux.getSiguiente());
            aux.setSiguiente(nuevo);
            this.tamanio++;
        }
    }
}
```

```

        NodoDoble<T> siguiente = aux.getSiguiente();
        aux.setSiguiente(nuevo);
        nuevo.setAnterior(aux);
        if (siguiente != null) {
            nuevo.setSiguiente(siguiente);
            siguiente.setAnterior(nuevo);
        } else {
            this.tail = nuevo;
        }
        this.tamanio++;
    }
}

public T obtener(int posicion) {
    if (posicion < 1 || posicion > tamanio) {
        throw new IndexOutOfBoundsException("Posición inválida");
    }
    NodoDoble<T> aux = this.head;
    for (int i = 1; i < posicion; i++) { // Cambiado de 0 a 1
        aux = aux.getSiguiente();
    }
    return aux.getData();
}

public void eliminarElPrimero() {
    if (!this.estaVacia()) {
        if (this.head == this.tail) {
            this.head = null;
            this.tail = null;
        } else {
            this.head = this.head.getSiguiente();
            this.head.setAnterior(null);
        }
        this.tamanio--;
    }
}

public void eliminarElFinal() {
    if (!this.estaVacia()) {
        if (this.head == this.tail) {
            this.head = null;
            this.tail = null;
        } else {
            this.tail = this.tail.getAnterior();
            this.tail.setSiguiente(null);
        }
        this.tamanio--;
    }
}

public void eliminar(int posicion) {
    if (posicion < 1 || posicion > tamanio) { // Cambiado de 0 a 1
        throw new IndexOutOfBoundsException("Posición inválida");
    }
    if (posicion == 1) {
        eliminarElPrimero();
    } else if (posicion == tamanio) {
        eliminarElFinal();
    } else {
        NodoDoble<T> aux = this.head;

```

```

        for (int i = 1; i < posicion; i++) {
            aux = aux.getSiguiente();
        }
        NodoDoble<T> anterior = aux.getAnterior();
        NodoDoble<T> siguiente = aux.getSiguiente();
        anterior.setSiguiente(siguiente);
        siguiente.setAnterior(anterior);
        this.tamano--;
    }
}

public int buscar(T valor) {
    NodoDoble<T> aux = this.head;
    int posicion = 1; // Comienza en 1
    while (aux != null) {
        if (aux.getData().equals(valor)) {
            return posicion;
        }
        aux = aux.getSiguiente();
        posicion++;
    }
    return -1; // Osea que no fue encontrado
}

public void actualizar(T aBuscar, T valor) {
    NodoDoble<T> aux = this.head;
    while (aux != null) {
        if (aux.getData().equals(aBuscar)) {
            aux.setData(valor);
            return;
        }
        aux = aux.getSiguiente();
    }
    System.out.println("Elemento no encontrado");
}

public void transversal(int direccion) {
    if (direccion == 1) { // De derecha a izquierda
        NodoDoble<T> aux = this.tail;
        while (aux != null) {
            System.out.print(aux + " ");
            aux = aux.getAnterior();
        }
    } else { // De izquierda a derecha
        NodoDoble<T> aux = this.head;
        while (aux != null) {
            System.out.print(aux + " ");
            aux = aux.getSiguiente();
        }
    }
    System.out.println("");
}
}

```

## Class NodoDoble

```
package Tarea5Package;

public class NodoDoble<T> {
    private T data;
    private NodoDoble<T> siguiente;
    private NodoDoble<T> anterior;

    public NodoDoble() {
    }

    public NodoDoble(T data) {
        this.data = data;
    }

    public NodoDoble(T data, NodoDoble<T> siguiente, NodoDoble<T> anterior) {
        this.data = data;
        this.siguiente = siguiente;
        this.anterior = anterior;
    }

    public T getData() {
        return data;
    }

    public void setData(T data) {
        this.data = data;
    }

    public NodoDoble<T> getSiguiente() {
        return siguiente;
    }

    public void setSiguiente(NodoDoble<T> siguiente) {
        this.siguiente = siguiente;
    }

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

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

    @Override
    public String toString() {
        return "<--| "+ this.data +" |-->";
    }
}
```

## Class Prueba

```
package Tarea5Package;

public class Prueba {
    public static void main(String[] args) {
        DoubleLinkedList<Integer> numeros = new DoubleLinkedList<>();

        numeros.agregarAlInicio(50);
        numeros.agregarAlFinal(60);
        numeros.agregarAlFinal(65);
        numeros.agregarAlFinal(70);
        numeros.agregarAlFinal(80);
        numeros.agregarAlFinal(90);

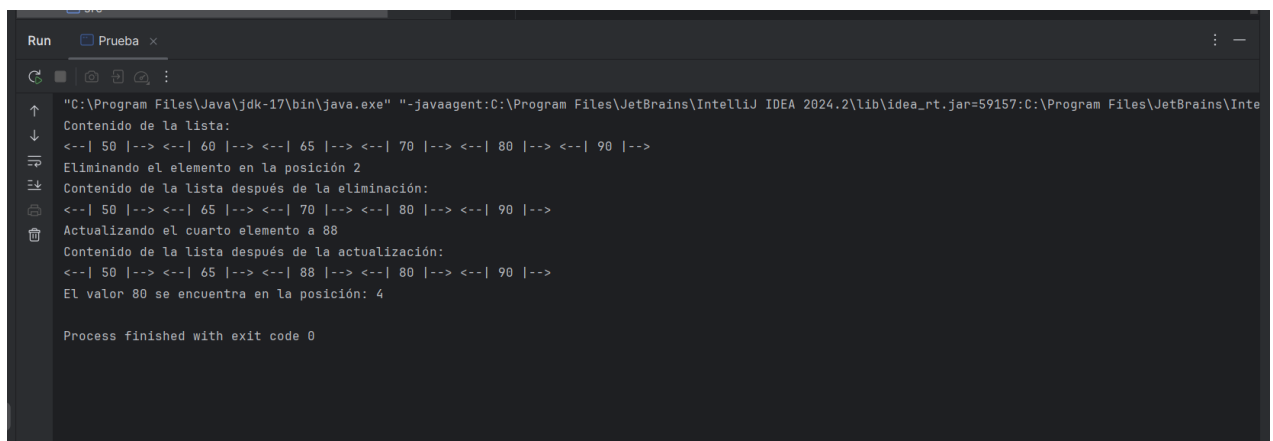
        System.out.println("Contenido de la lista:");
        numeros.transversal(0); // Imprime la lista de izquierda a derecha

        System.out.println("Eliminando el elemento en la posición 2");
        numeros.eliminar(2);
        System.out.println("Contenido de la lista después de la eliminación:");
        numeros.transversal(0);

        System.out.println("Actualizando el cuarto elemento a 88");
        numeros.actualizar(70, 88); // Actualiza el valor 70 a 88.
        System.out.println("Contenido de la lista después de la actualización:");
        numeros.transversal(0);

        int posicion = numeros.buscar(80); // Busca la posición del valor 80
        System.out.println("El valor 80 se encuentra en la posición: " + posicion);
    }
}
```

## Ejecución:



```
Run Prueba x
"C:\Program Files\Java\jdk-17\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2\lib\idea_rt.jar=59157:C:\Program Files\JetBrains\Inte
Contenido de la lista:
<--| 50 |--> <--| 60 |--> <--| 65 |--> <--| 70 |--> <--| 80 |--> <--| 90 |-->
Eliminando el elemento en la posición 2
Contenido de la lista después de la eliminación:
<--| 50 |--> <--| 65 |--> <--| 70 |--> <--| 80 |--> <--| 90 |-->
Actualizando el cuarto elemento a 88
Contenido de la lista después de la actualización:
<--| 50 |--> <--| 65 |--> <--| 88 |--> <--| 80 |--> <--| 90 |-->
El valor 80 se encuentra en la posición: 4

Process finished with exit code 0
```