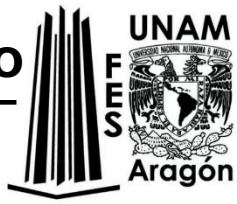
UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO



FACULTAD DE ESTUDIOS SUPERIORES ARAGON

TAREA 1

P R E S E N T A

Alexis Hernández Zamudio

APROFESOR

Jesús Hernández Cabrera

Gpo:1158

URL del repositorio:

https://github.com/TyrBalder1439/Estructura-de-Datos-

Ciudad Nezahualcóyotl, EDOMEX. 5 de septiembre del 2024

NodoDoble.java × DoubleLinkedList.java probandoListadoDoble.java

```java
class NodoDoble<T> {
    private T data;
    private NodoDoble<T> siguiente;
    private NodoDoble<T> anterior;

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

    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 data.toString();
    }
}
```

NodoDoble.java | DoubleLinkedList.java | probandoListadoDoble.java

gadas C:\Users\Alexis\Documents\FES ARAGON\Tercer_Semestre\Estructura_de_datos\Tarea_5\ListasDob

```java
    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 {
            this.head.setAnterior(nuevo);
            nuevo.setSiguiente(this.head);
            this.head = nuevo;
        }
        this.tamanio++;
    }

    public void agregarAlFinal(T valor) {
        NodoDoble<T> nuevo = new NodoDoble<>(valor);
        if (this.estaVacia()) {
```

```java
            this.head = nuevo;
            this.tail = nuevo;
        } else {
            this.tail.setSiguiente(nuevo);
            nuevo.setAnterior(this.tail);
            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.setSiguiente(aux.getSiguiente());
            nuevo.setAnterior(aux);
            if (aux.getSiguiente() != null) {
                aux.getSiguiente().setAnterior(nuevo);
            } else {
                tail = nuevo;  // Si es el último nodo, actualizamos tail
            }
            aux.setSiguiente(nuevo);
            this.tamanio++;
        }
    }
    public T obtener(int posicion) {
        if (posicion < 0 || posicion >= tamanio) {
            throw new IndexOutOfBoundsException("Posición inválida");
        }
        NodoDoble<T> aux = this.head;
        for (int i = 0; i < posicion; i++) {
            aux = aux.getSiguiente();
        }
        return aux.getData();
    }

    public void eliminarElPrimero() {
        if (!estaVacia()) {
            if (this.head == this.tail) {  // Solo un elemento
                this.head = this.tail = null;
            } else {
                this.head = this.head.getSiguiente();
                this.head.setAnterior(null);
            }
            this.tamanio--;
        }
    }
    public void eliminarElFinal() {
        if (!estaVacia()) {
            if (this.head == this.tail) {  // Solo un elemento
                this.head = this.tail = null;
            } else {
                this.tail = this.tail.getAnterior();
```

```
            this.tail.setSiguiente(null);
        }
        this.tamanio--;
    }
}
public void eliminar(int posicion) {
    if (posicion < 0 || posicion >= tamanio) {
        throw new IndexOutOfBoundsException("Posición inválida");
    }
    if (posicion == 0) {
        eliminarElPrimero();
    } else if (posicion == tamanio - 1) {
        eliminarElFinal();
    } else {
        NodoDoble<T> aux = this.head;
        for (int i = 0; i < posicion; i++) {
            aux = aux.getSiguiente();
        }
        aux.getAnterior().setSiguiente(aux.getSiguiente());
        aux.getSiguiente().setAnterior(aux.getAnterior());
        this.tamanio--;
    }
}
public int buscar(T valor) {
    NodoDoble<T> aux = this.head;
    int posicion = 0;
    while (aux != null) {
        if (aux.getData().equals(valor)) {
            return posicion;
        }
            aux = aux.getSiguiente();
            posicion++;
        }
        return -1;
    }
    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();
        }
    }
    public void transversal(int direccion) {
        if (direccion == 1) {
            NodoDoble<T> aux = this.tail;
            while (aux != null) {
                System.out.print(aux + " ");
                aux = aux.getAnterior();
            }
        } else {
            NodoDoble<T> aux = this.head;
            while (aux != null) {
                System.out.print(aux + " ");
                aux = aux.getSiguiente();
            }
        }
        System.out.println("");
```

NodoDoble.java | DoubleLinkedList.java | probandoListadoDoble.java

```java
public class probandoListadoDoble {
    public static void main(String[] args) {
        DoubleLinkedList<Integer> lista = new DoubleLinkedList<>();

        lista.agregarAlInicio(50);

        lista.agregarAlFinal(60);
        lista.agregarAlFinal(65);
        lista.agregarAlFinal(70);
        lista.agregarAlFinal(80);
        lista.agregarAlFinal(90);
        System.out.print("Contenido de la lista: ");
        lista.transversal(0);
        lista.eliminar(2);
        System.out.print("Contenido después de eliminar la posición 2: ");
        lista.transversal(0);
        lista.actualizar(lista.obtener(3), 88);
        System.out.print("actualizar el cuarto elemento a 88: ");
        lista.transversal(0);
        int posicion = lista.buscar(80);
        System.out.println("El valor 80 se encuentra en la posición: " + posicion);
    }
}
```

```
Contenido de la lista: 50 60 65 70 80 90
Contenido después de eliminar la posición 2: 50 60 70 80 90
actualizar el cuarto elemento a 88: 50 60 70 88 90
El valor 80 se encuentra en la posición: -1

Process finished with exit code 0
```