Estructura de Datos
Amy Cárdenas Silva
GRupo: 1360
Carrera: ingenieria en computacion
Tarea 5, muestra de funcionamiento

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

      DoubleLinkedList<Integer> ll = new DoubleLinkedList<>();

      ll.agregarAlInicio(50);
      ll.agregarAlFinal(60);
      ll.agregarAlFinal(65);
      ll.agregarAlFinal(70);
      ll.agregarAlFinal(80);
      ll.agregarAlFinal(90);
      ll.transversal(0);
      ll.eliminar(2);
      ll.transversal(0);
      ll.actualizar(80,88);
      ll.transversal(0);
      ll.buscar(80);
      System.out.println(ll.buscar(88));
      //en la tarea pide encontrar el elemnto 80, el 4to elemento era antes 80
      //y fue actualizado a 88
      //no se si queria que actializaramos 90 (poscicion 4, quinto elemento)
      //o que encontrasemos 88
      //de todas maneras al buscar 80 que no existe solo imprimirá un mensaje de error
  }
}

import javax.swing.*;

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

```
private int tamanio;

public DoubleLinkedList() {
}

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

public int getTamanio() {
    NodoDoble auxiliar = this.head;
    int contador = 1;
    if(estaVacia() == true){
        return 0;
    }else {
        while (auxiliar.getSiguiente() != null) {
            auxiliar = auxiliar.getSiguiente();
            contador++;
        }
        return contador;
    }
    //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 (estaVacia() == true){
        this.head = nuevo;
        this.tail = nuevo;
    }else{
        nuevo.setAnterior(this.tail);
```

```
            this.tail.setSiguiente(nuevo);
            this.tail = nuevo;
        }


    }

    public void agregarDespuesDe(T referencia, T valor) {
        NodoDoble<T> aux = this.head;
        while (aux.getData() != 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);
            aux.getSiguiente().setAnterior(nuevo);
            aux.setSiguiente(nuevo);


        }

    }

    public NodoDoble<T> obtener (int pos){
        NodoDoble aux = this.head;
        if (pos >= tamanio){
            System.out.println("indice fuera de rango");
            return null;
        }
        if(estaVacia() == true){
            System.out.println("no hay elementos para buscar");
        }
        for (int i = 0; i < pos; i++) {
            aux = aux.getSiguiente();
        }

        return aux;
    }

    public void eliminarElPrimero (){

        if (getTamanio() == 1){
            this.head = null;
```

```
        this.tail = null;
    }else{
        this.head = this.head.getSiguiente();
        this.head.setAnterior(null);
    }

}

public void eliminarElFinal (){

    if (getTamanio() == 1){
        this.head = null;
        this.tail = null;
    }else{
        this.tail = this.tail.getAnterior();
        this.tail.setSiguiente(null);
    }

}

public void eliminar (int pos){

    NodoDoble aux = this.head;

    if (pos >= getTamanio()){
        System.out.println("indice fueera de rango");
        return;
    }

    for (int i = 0; i < pos-1; i++) {
        aux = aux.getSiguiente();
    }
    if(pos == 0){
        eliminarElPrimero();
    } else if (pos == getTamanio()-1) {
        eliminarElFinal();
    }else{
        aux.getSiguiente().getSiguiente().setAnterior(aux);
        aux.setSiguiente(aux.getSiguiente().getSiguiente());
    }
}

public int buscar (T valor){
    NodoDoble aux = this.head;
    int pos = 0;
    while(aux.getData() != valor){
        aux = aux.getSiguiente();
        pos++;
```

```
            if (aux.getSiguiente() == null && aux.getData() != valor){
                System.out.println("elemento inexistente");
                return 0;
            }
        }
        return pos;
    }

    public void actualizar (T aBuscar, T valor){

        NodoDoble aux = this.head;
        while (aux.getData() != aBuscar){
            if (aux.getSiguiente() == null && aux.getData() != valor){
                System.out.println("elemento inexistente");
                return;
            }
            aux = aux.getSiguiente();

        }
        aux.setData(valor);
        return;
    }

    /**
     * @param direccion 0 --> izq a derecha  si es 1 --> derecha a izq
     */
    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("");
    }



}

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 +" |-->";
    }
}
```

```
<--| 50 |--><--| 60 |--><--| 65 |--><--| 70 |--><--| 80 |--><--| 90 |-->
<--| 50 |--><--| 60 |--><--| 70 |--><--| 80 |--><--| 90 |-->
<--| 50 |--><--| 60 |--><--| 70 |--><--| 88 |--><--| 90 |-->
elemento inexistente
3
```