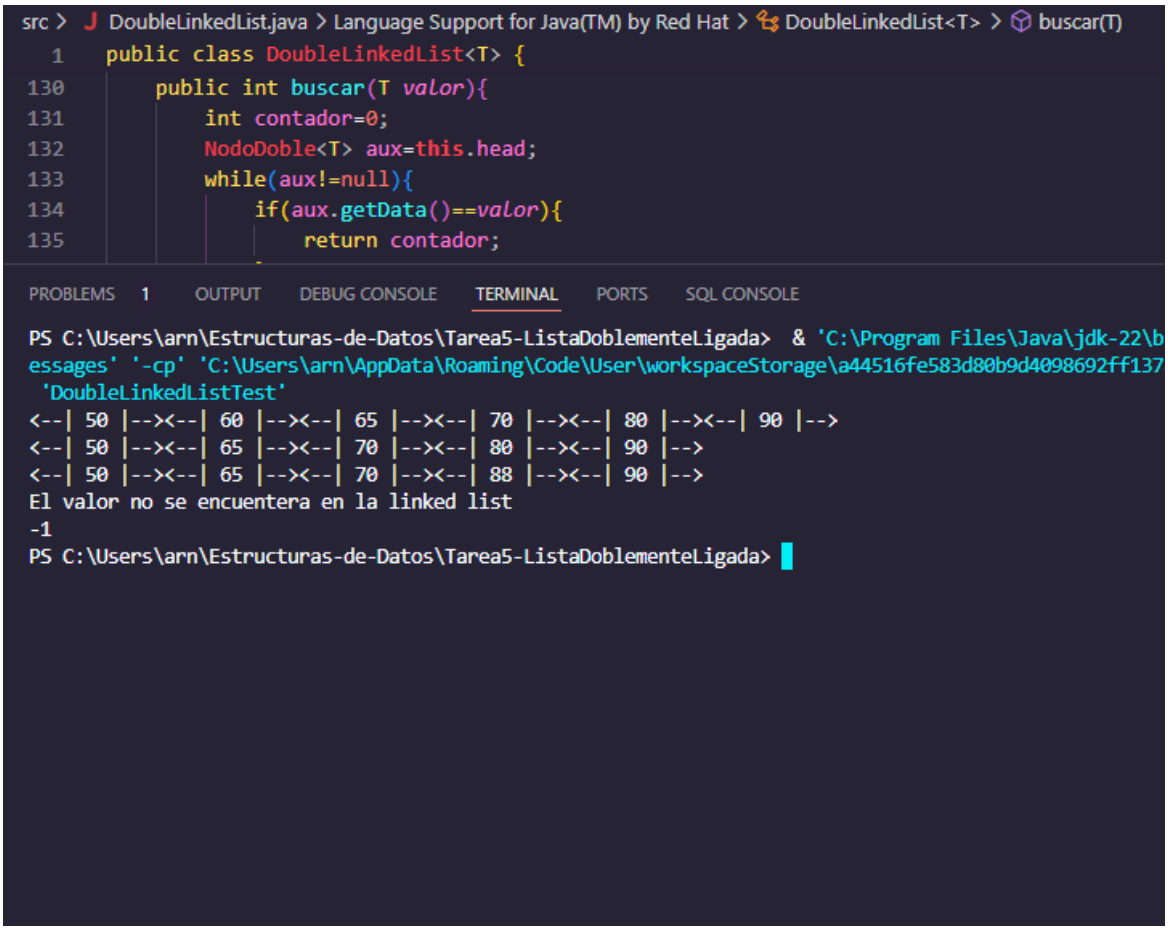
Hernández Rubio Aarón Isaí

Estructura de Datos

Grupo:1360(2025-1)

Tarea 5: Lista Doblemente Ligada

6 de septiembre de 2024




```java
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() {
        return tamanio;
    }
```

```java
    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++;
    }

    /**
     * @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 void agregarDespuesDe(T referencia, T valor) {
        NodoDoble<T> aux = this.head;
       while(aux!=null){
        if(aux.getData()==referencia){
            break;
        }else{
            aux=aux.getSiguiente();
        }
       }
       NodoDoble<T> nuevo= new NodoDoble<>(valor);
       NodoDoble<T> siguiente=aux.getSiguiente();
       if(siguiente!=null){
```

```
            aux.setSiguiente(nuevo);
            nuevo.setSiguiente(siguiente);
            nuevo.setAnterior(aux);
            siguiente.setAnterior(nuevo);
        }else if(aux==this.head){
            this.agregarAlInicio(valor);
        }else{
            this.agregarAlFinal(valor);
        }

        this.tamanio++;

    }

    public void agregarAlFinal(T valor){
        NodoDoble<T> nuevo=new NodoDoble<>(valor,null,this.tail);
        this.tail.setSiguiente(nuevo);
        this.tail=nuevo;
        this.tamanio++;
    }

    public T obtener(int posicion){
        NodoDoble<T> aux=this.head;
        for(int i=0;i<=posicion;i++){
            aux=aux.getSiguiente();
        }
        return aux.getData();
    }

    public void eliminarElPrimero(){
        this.head=this.head.getSiguiente();
        this.head.setAnterior(null);
        this.tamanio--;
    }

    public void eliminarElFinal(){
        this.tail=this.tail.getAnterior();
        this.tail.getSiguiente().setAnterior(null);
        this.tail.setSiguiente(null);
        this.tamanio--;
    }

    public void eliminar(int posicion){

        if(posicion==0){
```

```
            this.eliminarElPrimero();
            return;
        }
        if (posicion==tamanio){
            this.eliminarElFinal();
            return;
        }

        NodoDoble<T> aux=this.head;
        for(int i=1;i<posicion;i++){
            aux=aux.getSiguiente();
        }
        aux.getAnterior().setSiguiente(aux.getSiguiente());
        aux.getSiguiente().setAnterior(aux.getAnterior());
        aux.setAnterior(null);
        aux.setSiguiente(null);

    }

    public int buscar(T valor){
        int contador=0;
        NodoDoble<T> aux=this.head;
        while(aux!=null){
            if(aux.getData()==valor){
                return contador;
            }
            aux=aux.getSiguiente();
        }
        System.out.println("El valor no se encuentera en la linked list");
        return -1;
    }

    /*

    ACTUALIZAR POR VALOR
    public void actualizar(T valorBuscar,T valor){
        NodoDoble<T> aux=this.head;

        while(aux!=null){
            if(aux.getData()==valorBuscar){
                aux.setData(valor);
                return;
            }else{
                aux=aux.getSiguiente();
            }
```

```
        }

        System.out.println("Valor no existente");
    }
   */

   //ACTUALIZAR POR INDEX//
   public void actualizar(int index,T valor){
        NodoDoble<T> aux=this.head;
        int contador=1;

        while(contador<index){
            contador++;
            aux=aux.getSiguiente();
        }
        if(contador>tamanio){
            System.out.println("Valor no existente");

        }else{
            aux.setData(valor);
        }


    }

}
```

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

```
public class DoubleLinkedListTest {
    public static void main(String[] args) {
        DoubleLinkedList<Integer> listaNumeros= new DoubleLinkedList<>();
        listaNumeros.agregarAlInicio(50);
        listaNumeros.agregarAlFinal(60);
        listaNumeros.agregarAlFinal(65);
        listaNumeros.agregarAlFinal(70);
        listaNumeros.agregarAlFinal(80);
        listaNumeros.agregarAlFinal(90);
        listaNumeros.transversal(0);
        listaNumeros.eliminar(2);
        listaNumeros.transversal(0);
        listaNumeros.actualizar(4,88);
        listaNumeros.transversal(0);
```

```
        System.out.println(listaNumeros.buscar(80));
    }
}
```