UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

ALUMNO:

PAZ MALDONADO CARLOS SAÚL

NOMBRE DEL PROFESOR:

HERNANDEZ CABRERA JESUS

NOMBRE DE LA MATERIA:

ESTRUCTURA DE DATOS

FECHA DE ENTREGA:

6 de septiembre del 2024

TAREA NO. 5

CAPTURAS DEL CÓDIGO

CLASE NodoDoble

```java
package unam.fesaragon.estructuradatos;

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

    // Constructores
    public NodoDoble() {
    }

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

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

    // Getters y Setters
    public T getDato() {
        return dato;
    }

    public void setDato(T dato) {
        this.dato = dato;
    }

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

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

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

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

Clase ListaDoblementeLigada

```
package unam.fesaragon.estructuradatos;

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

    // Constructor
    public ListaDoblementeLigada() {
        this.head = null;
        this.tail = null;
        this.tamanio = 0;
    }

    public ListaDoblementeLigada(NodoDoble<T> head,
NodoDoble<T> tail) {
        this.head = head;
        this.tail = tail;
    }

    // Comprobar si está vacía
    public boolean esta_vacia() {
        return this.tamanio == 0;
    }

    // Agregar al inicio de la lista
    public void agregar_al_inicio(T valor) {
        NodoDoble<T> nuevo = new NodoDoble<>(valor);
        if (esta_vacia()) {
            this.head = nuevo;
            this.tail = nuevo;
        } else {
            nuevo.setSiguiente(this.head);
            this.head.setAnterior(nuevo);
            this.head = nuevo;
        }
        tamanio++;
    }

    // Agregar al final de la lista
    public void agregar_al_final(T valor) {
        NodoDoble<T> nuevo = new NodoDoble<>(valor);
        if (esta_vacia()) {
            this.head = nuevo;
            this.tail = nuevo;
        } else {
            this.tail.setSiguiente(nuevo);
            nuevo.setAnterior(this.tail);
            this.tail = nuevo;
        }
        tamanio++;
```

```
    }

    // Agregar después de un nodo de referencia
    public void agregar_después_de(T referencia, T valor) {
        if (esta_vacia()) {
            System.out.println("La lista esta vacia y por
lo tanto no existe la referencia");
            return;
        }

        NodoDoble<T> aux = this.head;
        while (aux != null) {
            if (aux.getDato().equals(referencia)) {
                break;
            }
            aux = aux.getSiguiente();
        }

        if (aux == null) {
            System.out.println("El nodo de referencia no
existe");
            return;
        }

        NodoDoble<T> nuevo = new NodoDoble<>(valor);
        nuevo.setSiguiente(aux.getSiguiente());
        nuevo.setAnterior(aux);
        //Nodo de referencia, no es la ultima en la lista.
        if (aux.getSiguiente() != null) {
            aux.getSiguiente().setAnterior(nuevo);
        } else {
            this.tail = nuevo;
        }
        aux.setSiguiente(nuevo);
        this.tamanio++;
    }

    // Obtener el elemento en una posición específica
    public T obtener(int posicion) {
        //Condiciones
        if (esta_vacia()) {
            System.out.println("La lista está vacía");
            return null;
        }
        if (posicion > this.tamanio) {
            System.out.println("Posición fuera de rango");
            return null;
        }

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

```
        return aux.getDato();
    }

    // Eliminar el primer elemento
    public void eliminar_el_primero() {
        if (esta_vacia()) {
            System.out.println("La lista está vacía");
            return;
        }
        if (this.head == this.tail) { // Solo un elemento
            this.head = null;
            this.tail = null;
        } else {
            this.head = this.head.getSiguiente();
            this.head.setAnterior(null);
        }
        tamanio--;
    }

    // Eliminar el último elemento
    public void eliminar_el_final() {
        if (esta_vacia()) {
            System.out.println("La lista está vacía");
            return;
        }

        if (this.head == this.tail) { // Solo un elemento
            this.head = null;
            this.tail = null;
        } else {
            this.tail = this.tail.getAnterior();
            this.tail.setSiguiente(null);
        }
        tamanio--;
    }

    // Eliminar un elemento en una posición específica
    public void eliminar(int posicion) {
        if (esta_vacia()) {
            System.out.println("La lista está vacía.");
            return;
        }
        if (posicion > this.tamanio) {
            System.out.println("Posición fuera de rango");
            return;
        }

        if (posicion == 1) {
            eliminar_el_primero();
            return;
        }
        if (posicion == this.tamanio) {
```

```
            eliminar_el_final();
            return;
        }

        NodoDoble<T> aux = this.head;
        int i = 1;
        while (i < posicion) {
            aux = aux.getSiguiente();
            i++;
        }

        aux.getAnterior().setSiguiente(aux.getSiguiente());
        aux.getSiguiente().setAnterior(aux.getAnterior());
        this.tamanio--;
    }

    // Buscar un elemento y retornar su posición
    public int buscar(T valor) {
        if (esta_vacia()) {
            System.out.println("La lista está vacía");
            return 0;
        }

        NodoDoble<T> aux = this.head;
        int posicion = 1;
        while (aux != null) {
            if (aux.getDato().equals(valor)) {
                return posicion;
            }
            aux = aux.getSiguiente();
            posicion++;
        }
        System.out.println("No se encontro el elemento");
        return 0;
    }

    // Actualizar un elemento
    public void actualizar(T a_buscar, T valor) {
        if (esta_vacia()) {
            System.out.println("La lista está vacía.");
            return;
        }

        NodoDoble<T> aux = this.head;
        while (aux != null) {
            if (aux.getDato().equals(a_buscar)) {
                aux.setDato(valor);
                return;
            }
            aux = aux.getSiguiente();
        }
        System.out.println("El elemento valor o elemento no
esta en la lista");
```

```
    }

    // Recorrido transversal en una dirección específica
    // Verdadero es valor por defecto y con el falso es de
derecha a izquierda
    public void transversal(boolean porDefecto) {
        if (esta_vacia()) {
            System.out.println("La lista está vacía.");
            return;
        }

        if (porDefecto) {
            NodoDoble<T> aux = this.head;
            while (aux != null) {
                System.out.print("[" + aux.getDato() + "]
<--> ");
                aux = aux.getSiguiente();
            }
        } else {
            NodoDoble<T> aux = this.tail;
            while (aux != null) {
                System.out.print("[" + aux.getDato() + "]
<--> ");
                aux = aux.getAnterior();
            }
        }
        System.out.println("NULL");
    }
    //Sobrecargue el metodo para poner por default el
recorrido de izquierda a derecha
    public void transversal() {
        transversal(true);
    }

    //GETTERS Y SETTERS
    // Obtener el tamaño de la lista
    public int get_tamanio() {
        return this.tamanio;
    }
}
```

Clase Main

```java
package unam.fesaragon.estructuradatos;

public class Main {
    public static void main(String[] args) {
        // Crear una lista de Integer
        ListaDoblementeLigada<Integer> enteros = new
ListaDoblementeLigada<>();

        // Agregar elementos según el ejemplo
        // Agregar al inicio el 50
        enteros.agregar_al_inicio(50);
        // Agregar al final el 60
        enteros.agregar_al_final(60);
        // Agregar al final el 65
        enteros.agregar_al_final(65);
        // Agregar al final el 70
        enteros.agregar_al_final(70);
        // Agregar al final el 80
        enteros.agregar_al_final(80);
        // Agregar al final el 90
        enteros.agregar_al_final(90);

        // Imprimir el contenido
        System.out.println("Contenido de la lista:");
        enteros.transversal();

        // Eliminar el de la posición 2
        enteros.eliminar(2);

        // Volver a imprimir el contenido
        System.out.println("\nLista después de eliminar el
elemento en la posición 2:");
        enteros.transversal();

        // Actualizar el cuarto elemento a 88
        enteros.actualizar(enteros.obtener(4), 88);

        // Volver a imprimir el contenido
        System.out.println("\nLista después de actualizar
el cuarto elemento a 88:");
        enteros.transversal();

        // Buscar el valor 80 e imprimir su posición
        int posicion = enteros.buscar(80);
        System.out.println("\nEl valor 80 se encuentra en
la posición: " + posicion);

    }
}
```

CAPTURAS DE EJECUCIÓN

La sentencia “no se encontró el elemento” se presenta, ya que, al actualizar el cuarto elemento que guarda el número 80 este ya no existe, puesto que, es remplazado por el número 88.