



Universidad Nacional Autónoma de  
México

Facultad de Estudios Superiores  
Aragón

Alumno: Leal Sanchez Bryan Uriel

Materia: Estructura de Datos

Grupo: 1360

Profesor: Hernández Cabrera Jesús

Listas Doblemente Ligadas

## Código

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

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

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

    public int getTamaño() {
        return tamaño;
    }

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

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

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

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

```

```

        this.tamano--;
    }
}

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

public void eliminar(int posicion) {
    if (posicion < 0 || posicion >= tamano) {
        throw new IndexOutOfBoundsException("Posición fuera de rango.");
    }
    if (posicion == 0) {
        eliminarElPrimero();
    } else if (posicion == tamano - 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());
        tamano--;
    }
}

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; // No 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();
    }
}

```

```

/**
 * @param direccion 0 --> izquierda a derecha, si es 1 --> derecha a izquierda
 */
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 static 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();
    }
}

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

    // Agregar al inicio el 50
    lista.agregarAlInicio(50);

    // Agregar al final 60, 65, 70, 80, 90
    lista.agregarAlFinal(60);
    lista.agregarAlFinal(65);
    lista.agregarAlFinal(70);
    lista.agregarAlFinal(80);
}

```

```

        lista.agregarAlFinal(90);

        // Imprimir el contenido
        System.out.println("Contenido de la lista:");
        lista.transversal(0);

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

        // Imprimir el contenido nuevamente
        System.out.println("Contenido después de eliminar el elemento en la posición 2:");
        lista.transversal(0);

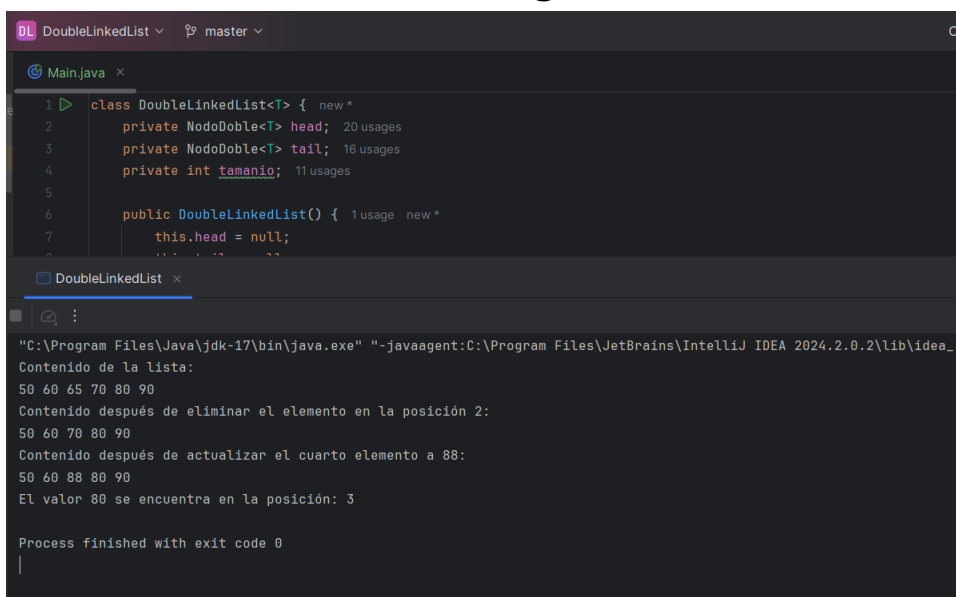
        // Actualizar el cuarto elemento a 88
        lista.actualizar(70, 88);

        // Imprimir el contenido nuevamente
        System.out.println("Contenido después de actualizar el cuarto elemento a 88:");
        lista.transversal(0);

        // Buscar el valor 80
        int posicion = lista.buscar(80);
        System.out.println("El valor 80 se encuentra en la posición: " + posicion);
    }
}

```

## Código Corrido



The screenshot shows an IDE with two panels. The top panel displays the source code for `DoubleLinkedList` and `Main.java`. The bottom panel shows the output of the program execution.

```

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

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

// Main.java
class Main {
    public static void main(String[] args) {
        DoubleLinkedList<Integer> lista = new DoubleLinkedList<>();
        lista.agregarAlFinal(90);
        System.out.println("Contenido de la lista:");
        lista.transversal(0);
        lista.eliminar(2);
        System.out.println("Contenido después de eliminar el elemento en la posición 2:");
        lista.transversal(0);
        lista.actualizar(70, 88);
        System.out.println("Contenido después de 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);
    }
}

```

Output:

```

"C:\Program Files\Java\jdk-17\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.0.2\lib\idea_
Contenido de la lista:
50 60 65 70 80 90
Contenido después de eliminar el elemento en la posición 2:
50 60 70 80 90
Contenido después de actualizar el cuarto elemento a 88:
50 60 88 80 90
El valor 80 se encuentra en la posición: 3

Process finished with exit code 0

```