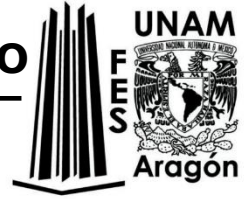




UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

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

```
1  class NodoDoble<T> {
2      private T data;
3      private NodoDoble<T> siguiente;
4      private NodoDoble<T> anterior;
5
6      public NodoDoble(T data) {
7          this.data = data;
8          this.siguiente = null;
9          this.anterior = null;
10     }
11
12     public T getData() {
13         return data;
14     }
15
16     public void setData(T data) {
17         this.data = data;
18     }
19
20     public NodoDoble<T> getSiguiente() {
21         return siguiente;
22     }
23
24     public void setSiguiente(NodoDoble<T> siguiente) {
25         this.siguiente = siguiente;
26     }
27
28     public NodoDoble<T> getAnterior() {
29         return anterior;
30     }
31
32     public void setAnterior(NodoDoble<T> anterior) {
33         this.anterior = anterior;
34     }
35
36     @Override
37     public String toString() {
38         return data.toString();
39     }
40 }
41
```

gadas C:\Users\Alexis\Documents\FES ARAGON\Tercer\_Semestre\Estructura\_de\_datos\Tarea\_5>ListasDob

```

2     private NodoDoble<T> head;
3     private NodoDoble<T> tail;
4     private int tamanio;
5
6     public DoubleLinkedList() {
7         this.head = null;
8         this.tail = null;
9         this.tamanio = 0;
10    }
11    public boolean estaVacia() {
12        return this.head == null && this.tail == null;
13    }
14    public int getTamanio() {
15        return tamanio;
16    }
17    public void agregarAlInicio(T valor) {
18        NodoDoble<T> nuevo = new NodoDoble<>(valor);
19        if (this.estaVacia()) {
20            this.head = nuevo;
21            this.tail = nuevo;
22        } else {
23            this.head.setAnterior(nuevo);
24            nuevo.setSiguierte(this.head);
25            this.head = nuevo;
26        }
27        this.tamanio++;
28    }
29
30    public void agregarAlFinal(T valor) {
31        NodoDoble<T> nuevo = new NodoDoble<>(valor);
32        if (this.estaVacia()) {

```

```

33         this.head = nuevo;
34         this.tail = nuevo;
35     } else {
36         this.tail.setSiguiente(nuevo);
37         nuevo.setAnterior(this.tail);
38         this.tail = nuevo;
39     }
40     this.tamanio++;
41 }
42 public void agregarDespuesDe(T referencia, T valor) {
43     NodoDoble<T> aux = this.head;
44     while (aux != null && !aux.getData().equals(referencia)) {
45         aux = aux.getSiguiente();
46     }
47     if (aux == null) {
48         System.out.println("No existe la referencia!!!");
49     } else {
50         NodoDoble<T> nuevo = new NodoDoble<>(valor);
51         nuevo.setSiguiente(aux.getSiguiente());
52         nuevo.setAnterior(aux);
53         if (aux.getSiguiente() != null) {
54             aux.getSiguiente().setAnterior(nuevo);
55         } else {
56             tail = nuevo; // Si es el último nodo, actualizamos tail
57         }
58         aux.setSiguiente(nuevo);
59         this.tamanio++;
60     }
61 }
62 public T obtener(int posicion) {
63     if (posicion < 0 || posicion >= tamanio) {
64         throw new IndexOutOfBoundsException("Posición inválida");
65     }
66     NodoDoble<T> aux = this.head;
67     for (int i = 0; i < posicion; i++) {
68         aux = aux.getSiguiente();
69     }
70     return aux.getData();
71 }
72
73 public void eliminarElPrimero() {
74     if (!estaVacia()) {
75         if (this.head == this.tail) { // Solo un elemento
76             this.head = this.tail = null;
77         } else {
78             this.head = this.head.getSiguiente();
79             this.head.setAnterior(null);
80         }
81         this.tamanio--;
82     }
83 }
84 public void eliminarElFinal() {
85     if (!estaVacia()) {
86         if (this.head == this.tail) { // Solo un elemento
87             this.head = this.tail = null;
88         } else {
89             this.tail = this.tail.getAnterior();

```

```

90         this.tail.setSiguiente(null);
91     }
92     this.tamanio--;
93 }
94 }
95 public void eliminar(int posicion) {
96     if (posicion < 0 || posicion >= tamanio) {
97         throw new IndexOutOfBoundsException("Posición inválida");
98     }
99     if (posicion == 0) {
100         eliminarElPrimero();
101     } else if (posicion == tamanio - 1) {
102         eliminarElFinal();
103     } else {
104         NodoDoble<T> aux = this.head;
105         for (int i = 0; i < posicion; i++) {
106             aux = aux.getSiguiente();
107         }
108         aux.getAnterior().setSiguiente(aux.getSiguiente());
109         aux.getSiguiente().setAnterior(aux.getAnterior());
110         this.tamanio--;
111     }
112 }
113 public int buscar(T valor) {
114     NodoDoble<T> aux = this.head;
115     int posicion = 0;
116     while (aux != null) {
117         if (aux.getData().equals(valor)) {
118             return posicion;
119         }
120         aux = aux.getSiguiente();
121         posicion++;
122     }
123     return -1;
124 }
125 public void actualizar(T aBuscar, T valor) {
126     NodoDoble<T> aux = this.head;
127     while (aux != null) {
128         if (aux.getData().equals(aBuscar)) {
129             aux.setData(valor);
130             return;
131         }
132         aux = aux.getSiguiente();
133     }
134 }
135 public void transversal(int direccion) {
136     if (direccion == 1) {
137         NodoDoble<T> aux = this.tail;
138         while (aux != null) {
139             System.out.print(aux + " ");
140             aux = aux.getAnterior();
141         }
142     } else {
143         NodoDoble<T> aux = this.head;
144         while (aux != null) {
145             System.out.print(aux + " ");
146             aux = aux.getSiguiente();
147         }
148     }
149     System.out.println("");

```

```
© NodoDoble.java × © DoubleLinkedList.java © probandoListadoDoble.java ×
1 public class probandoListadoDoble {
2     public static void main(String[] args) {
3         DoubleLinkedList<Integer> lista = new DoubleLinkedList<>();
4
5         lista.agregarAlInicio(50);
6
7         lista.agregarAlFinal(60);
8         lista.agregarAlFinal(65);
9         lista.agregarAlFinal(70);
10        lista.agregarAlFinal(80);
11        lista.agregarAlFinal(90);
12        System.out.print("Contenido de la lista: ");
13        lista.transversal(0);
14        lista.eliminar(2);
15        System.out.print("Contenido después de eliminar la posición 2: ");
16        lista.transversal(0);
17        lista.actualizar(lista.obtener(3), 88);
18        System.out.print("actualizar el cuarto elemento a 88: ");
19        lista.transversal(0);
20        int posicion = lista.buscar(80);
21        System.out.println("El valor 80 se encuentra en la posición: " + posicion);
22    }
23 }
24
```

```
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
```