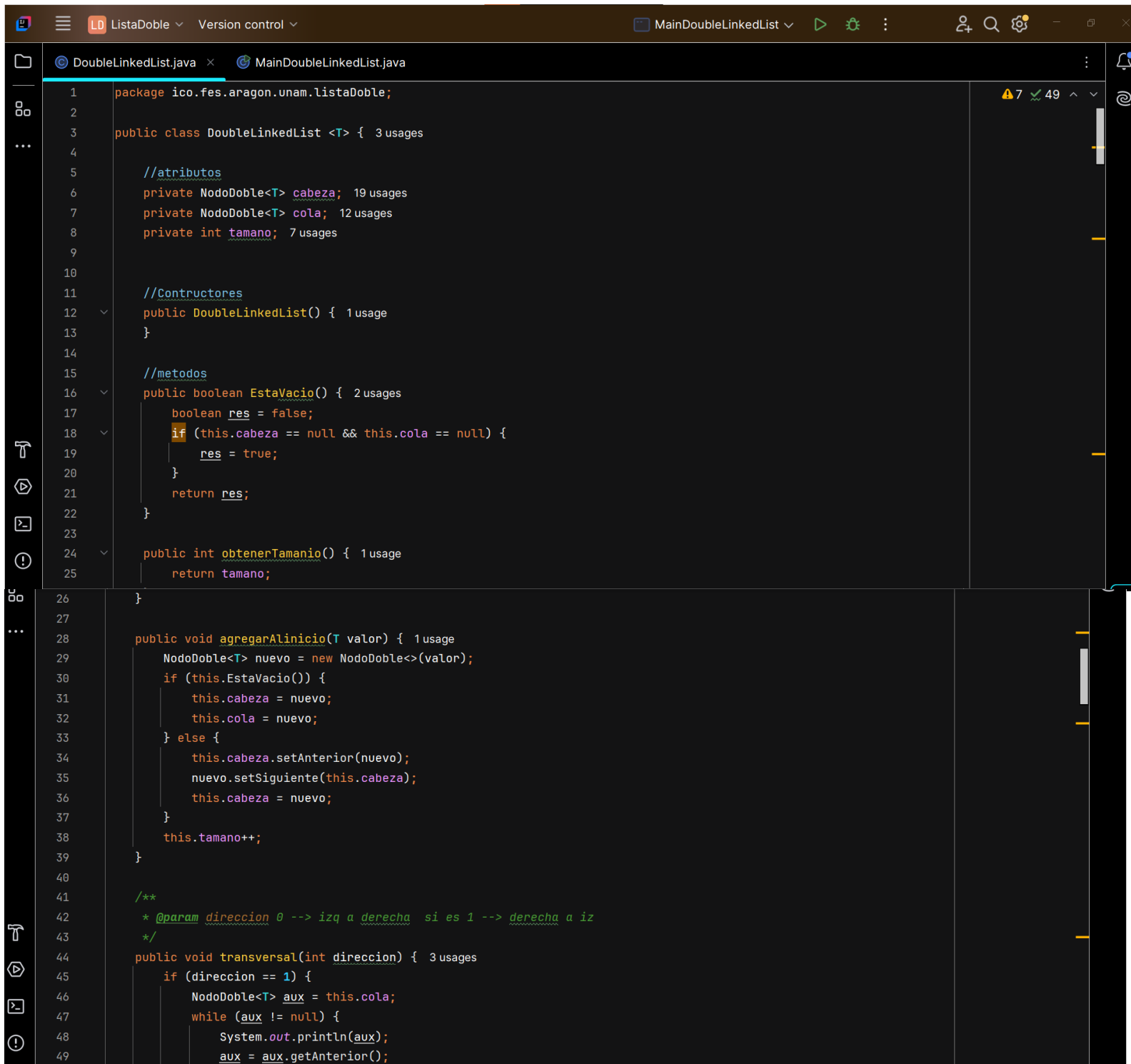


Código listo doblemente enlazada



```
1 package ico.fes.aragon.unam.listaDoble;
2
3 public class DoubleLinkedList <T> { 3 usages
4
5     //atributos
6     private NodoDoble<T> cabeza; 19 usages
7     private NodoDoble<T> cola; 12 usages
8     private int tamaño; 7 usages
9
10
11     //Constructores
12     public DoubleLinkedList() { 1 usage
13     }
14
15     //metodos
16     public boolean EstaVacio() { 2 usages
17         boolean res = false;
18         if (this.cabeza == null && this.colas == null) {
19             res = true;
20         }
21         return res;
22     }
23
24     public int obtenerTamano() { 1 usage
25         return tamaño;
26     }
27
28     public void agregarAlInicio(T valor) { 1 usage
29         NodoDoble<T> nuevo = new NodoDoble<>(valor);
30         if (this.EstaVacio()) {
31             this.cabeza = nuevo;
32             this.colas = nuevo;
33         } else {
34             this.cabeza.setAnterior(nuevo);
35             nuevo.setSiguiete(this.cabeza);
36             this.cabeza = nuevo;
37         }
38         this.tamaño++;
39     }
40
41     /**
42      * @param direccion 0 --> izq a derecha si es 1 --> derecha a iz
43      */
44     public void transversal(int direccion) { 3 usages
45         if (direccion == 1) {
46             NodoDoble<T> aux = this.colas;
47             while (aux != null) {
48                 System.out.println(aux);
49                 aux = aux.getAnterior();
50             }
51         } else {
52             NodoDoble<T> aux = this.cabeza;
53             while (aux != null) {
54                 System.out.println(aux);
55                 aux = aux.getSiguiente();
56             }
57         }
58     }
59 }
```

```

50     }
51     } else if(direccion == 0){
52         NodoDoble<T> aux = this.cabeza;
53         while (aux != null) {
54             System.out.println(aux);
55             aux = aux.getSiguiente();
56         }
57     }
58     System.out.println("");
59 }
60
61 public void agregarDespuesDe(T referencia, T valor) { no usages
62     NodoDoble<T> aux = this.cabeza;
63     while (aux != null) {
64         if (aux.getDato() == referencia) {
65             NodoDoble<T> nuevo = new NodoDoble<>(valor);
66             nuevo.setSiguiente(aux.getSiguiente());
67             nuevo.setAnterior(aux);
68             if (aux.getSiguiente() != null) {
69                 aux.getSiguiente().setAnterior(nuevo);
70             } else {
71                 this cola = nuevo;
72             }

```

```

73             aux.setSiguiente(nuevo);
74             tamano++;
75             return;
76         }
77     }
78     aux = aux.getSiguiente();
79 }
80 }
81 System.out.println("No se encontro la referencia");
82 }
83
84 public void agregarAlfinal(T valor) { 5 usages
85     NodoDoble<T> nuevo = new NodoDoble<>(valor);
86     NodoDoble<T> aux = this.cabeza;
87     if (this.EstaVacio()) {
88         this.cabeza = nuevo;
89         this cola = nuevo;
90     } else {
91         while(aux != null){
92             if(aux.getSiguiente() == null){
93                 aux.setSiguiente(nuevo);
94                 nuevo.setAnterior(aux);
95                 this cola = nuevo;

```

```

95                 this cola = nuevo;
96                 break;
97             }
98             aux= aux.getSiguiente();
99         }
100     }
101     this.tamano++;
102 }
103
104 public int buscar(T valor) { 1usage
105     int indice = 0;
106     int contador = 0;
107     NodoDoble<T> aux = this.cabeza;
108     while (aux != null) {
109         ++contador;
110         if (aux.getDato() == valor) {
111             indice = contador;
112             break;
113         }
114         aux = aux.getSiguiente();
115     }
116     return indice;
117 }

```

```

118
119 public void eliminarElPrimero() { no usages
120     if(this.cabeza == null){
121         System.out.println("La lista esta vacia, no nada por eliminar");
122         return;
123     }
124     this.cabeza = cabeza.getSiguiente();
125     if(this.cabeza != null){
126         this.cabeza.setAnterior(null);
127     }else{
128         this cola = null;
129     }
130     tamano--;
131 }

```

```

132
133 public void eliminarElFinal() { no usages
134     if (this.cola == null){
135         System.out.println("Lista vacia, no hay elementos por borrar");
136         return;
137     }
138     this.cola = cola.getAnterior();
139     if(this.cola != null){
140         this.cola.setSiguiente(null);
141     }else{

```

```

142         this.cabeza = null;
143     }
144     tamano--;
145 }

```

```

146
147
148 public void eliminarPosicion(int posicion){ 1 usage
149     int contador= 0;
150     int indice = posicion;
151     NodoDoble<T> aux = this.cabeza;
152     NodoDoble<T> copia;
153     while (aux != null){
154         contador++;
155         if (contador == indice){
156             copia = aux.getAnterior();
157             copia.setSiguiente(aux.getSiguiente());
158             aux.getSiguiente().setAnterior(copia);
159             return;
160         }
161         aux= aux.getSiguiente();
162     }
163     System.out.println("Posicion no encontrada");
164     tamano--;

```

```

165 }
166
167 public void actualizar(T referencia, T valor){ 1 usage
168     NodoDoble<T> aux = this.cabeza;
169     while(aux != null){
170         if(aux.getDatos() == referencia){
171             aux.setDatos(valor);
172             return;
173         }
174         aux = aux.getSiguiente();
175     }
176     System.out.println("No se encontro el dato");
177 }

```

```

178
179 public T obtener(int posicion){ no usages
180     NodoDoble<T> aux = this.cabeza;
181     T dato = null;
182     int contador = 0;
183     while (aux != null){
184         ++contador;
185         if(posicion != 0 && posicion<obtenerTamano()){
186             if (posicion == contador){
187                 dato =aux.getDatos();
188             }

```

```

177
178
179 public T obtener(int posicion){ no usages
180     NodoDoble<T> aux = this.cabeza;
181     T dato = null;
182     int contador = 0;
183     while (aux != null){
184         ++contador;
185         if(posicion != 0 && posicion<obtenerTamano()){
186             if (posicion == contador){
187                 dato =aux.getDato();
188             }
189         }
190         aux = aux.getSiguiente();
191     }
192     return dato;
193 }
194 }
195
196
197
198
199
200

```

Código main

```

1 package ico.fes.aragon.unam.principal;
2
3 import ico.fes.aragon.unam.listaDoble.DoubleLinkedList;
4 import ico.fes.aragon.unam.listaDoble.NodoDoble;
5
6 public class MainDoubleLinkedList {
7     public static void main(String[] args) {
8
9         System.out.println("Creando lista de tipo integer");
10        DoubleLinkedList<Integer> numeros = new DoubleLinkedList<>();
11
12        // añadiendo el 50 al inicio
13        numeros.agregarAlinicio(valor: 50);
14        //añadiendo al final
15        numeros.agregarAlfinal(valor: 60);
16        numeros.agregarAlfinal(valor: 65);
17        numeros.agregarAlfinal(valor: 70);
18        numeros.agregarAlfinal(valor: 80);
19        numeros.agregarAlfinal(valor: 90);
20        System.out.println("imprimiendo contenido");
21        numeros.transversal(direccion: 0);
22        System.out.println("eliminar el de la posicion 2");
23        numeros.eliminarPosicion(2);
24        System.out.println("volver a imprimir");
25        numeros.transversal(direccion: 0);
26        System.out.println("actualizar el cuarto elemento a 88");
27        numeros.actualizar(referencia: 80, valor: 88);
28        System.out.println("volver a imprimir");
29        numeros.transversal(direccion: 0);
30        System.out.println("buscar el valor 88, e imprimir en que pocicion se encuentra");
31        System.out.println("El valor 88 tiene la posicion: "+numeros.buscar(valor: 88));
32
33
34
35 }
36
37 }
38

```

Ejecución main

```
"C:\Program Files\Java\jdk-22\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.2.0.1\lib\idea_rt.jar=-
Creando lista de tipo integer
imprimiendo contenido
<-----| 50 |----->
<-----| 60 |----->
<-----| 65 |----->
<-----| 70 |----->
<-----| 80 |----->
<-----| 90 |----->

eliminar el de la posicion 2
volver a imprimir
<-----| 50 |----->
<-----| 65 |----->
<-----| 70 |----->
<-----| 80 |----->
<-----| 90 |----->

actualizar el cuarto elemento a 88
volver a imprimir
<-----| 50 |----->
<-----| 65 |----->
<-----| 70 |----->
<-----| 88 |----->
<-----| 90 |----->

buscar el valor 88, e imprimir en que pocicion se encuentra
El valor 88 tiene la posicion: 4

Process finished with exit code 0
```