



Universidad Nacional Autónoma
de México

Facultad de Estudios Superiores
Aragón

Tarea 5: DoubleLinkedList

Estructura de Datos

Prof. Hernández Cabrera Jesús

Grupo: 1306

Angeles Mermejo Octavio Emiliano

Código Fuente Nodo Doble:

```
1 package fes.aragon.clases;
2
3 public class NodoDoble <T>{
4     private T dato;
5     private NodoDoble<T> siguiente;
6     private NodoDoble<T> anterior;
7
8     public NodoDoble() {
9     }
10
11     public NodoDoble(T dato) {
12         this.dato = dato;
13     }
14
15     public NodoDoble(T dato, NodoDoble<T> siguiente, NodoDoble<T> anterior) {
16         this.dato = dato;
17         this.siguiente = siguiente;
18         this.anterior = anterior;
19     }
20
21     public T getDato() {
22         return dato;
23     }
24
25     public void setDato(T dato) {
26         this.dato = dato;
27     }
28
29     public NodoDoble<T> getSiguiente() {
30         return siguiente;
31     }
32
33     public void setSiguiente(NodoDoble<T> siguiente) {
34         this.siguiente = siguiente;
35     }
36
37     public NodoDoble<T> getAnterior() {
38         return anterior;
39     }
40
41     public void setAnterior(NodoDoble<T> anterior) {
42         this.anterior = anterior;
43     }
44
45     @Override
46     public String toString() {
47         return dato + "| -> |";
48     }
49 }
```

Código Fuente DoubleLinkedList:

```
1 package fes.aragon.clases;
2
3 3 usages
4 public class DoubleLinkedList<T> {
5     21 usages
6     private NodoDoble<T> head ;
7     14 usages
8     private NodoDoble<T> tail;
9     no usages
10    private int direccion;
11    10 usages
12    private int tamaño;
13
14    1 usage
15    public DoubleLinkedList() {
16    }
17
18    no usages
19    public boolean estaVacio() {
20        boolean resultado = false;
21        if(this.head == null && this.tail == null){
22            System.out.println("La lista esta vacia");
23            return true;
24        }
25        System.out.println("La lista no esta vacia");
26        return resultado;
27    }
28
29    no usages
30    public int getTamaño(){
31        return tamaño;
32    }
33
34    1 usage
35    public void agregarAlInicio (T valor){
36        NodoDoble<T> nuevo = new NodoDoble<>(valor);
37        NodoDoble<T> aux = this.head;
38
39        if(aux == null){
40            this.head = nuevo;
41            this.tail = nuevo;
42        }else {
43            this.head.setAnterior(nuevo);
44            nuevo.setSiguiente(this.head);
45            this.head = nuevo;
46        }
47        tamaño++;
48    }
49
50    5 usages
51    public void agregarAlFinal (T valor){
52        NodoDoble<T> nuevo = new NodoDoble<>(valor);
53        NodoDoble<T> aux = this.head;
54
55        if(aux == null){
56            this.head = nuevo;
57            this.tail = nuevo;
58        }else {
59            this.tail.setSiguiente(nuevo);
60            nuevo.setAnterior(this.tail);
61            this.tail = nuevo;
62        }
63        tamaño++;
64    }
65 }
```

```

54     }
55
56     no usages
57     public void agregarDespuesDe (T referencia , T valor){
58         NodoDoble<T> aux = this.head;
59         while (aux != null){
60             if(aux.getData().equals(referencia)){
61                 NodoDoble<T> nuevo = new NodoDoble<>(valor);
62                 nuevo.setSiguiente(nuevo.getSiguiente());
63                 nuevo.setAnterior(aux);
64                 if(aux.getSiguiente() != null){
65                     aux.getSiguiente().setAnterior(nuevo);
66                 }else{
67                     this.tail = nuevo;
68                 }
69                 aux.setSiguiente(nuevo);
70                 tamano++;
71                 return;
72             }
73             aux = aux.getSiguiente();
74         }
75         System.out.println("No se encontro la referencia");
76     }
77
78     no usages
79     public T obtener (int posicion){
80         if (posicion < 0 || posicion >= tamano){
81             System.out.println("fuera del rango");
82         }
83
84         NodoDoble<T> aux = this.head;
85         for (int i = 0; i < posicion; i++) {
86             aux = aux.getSiguiente();
87         }
88         return aux.getData();
89     }
90
91     1 usage
92     public void eliminarElPrimero() {
93         if (this.head == null) {
94             System.out.println("No hay elementos por eliminar");
95             return;
96         }
97
98         this.head = head.getSiguiente();
99         if (this.head != null) {
100             this.head.setAnterior(null);
101         } else {
102             this.tail = null;
103         }
104         tamano--;
105     }
106
107     1 usage
108     public void eliminarElFinal() {
109         if (this.tail == null) {
110             System.out.println("No hay elementos por eliminar");
111             return;
112         }
113
114         this.tail = tail.getAnterior();
115         if (this.tail != null) {
116             this.tail.setSiguiente(null);

```

```

113     } else {
114         this.head = null;
115     }
116     tamano--;
117 }
118
119 1 usage
120 public void eliminar(int posicion) {
121     if (posicion < 0 || posicion >= tamano) {
122         System.out.println("Elemento fuera del rango");
123         return;
124     }
125
126     if (this.head == null) {
127         System.out.println("No hay elementos por eliminar");
128         return;
129     }
130
131     if (posicion == 0) {
132         eliminarElPrimero();
133     } else if (posicion == tamano - 1) {
134         eliminarElFinal();
135     } else {
136         NodoDoble<T> aux = this.head;
137         for (int i = 0; i < posicion; i++) {
138             aux = aux.getSiguiente();
139         }
140         aux.getAnterior().setSiguiente(aux.getSiguiente());
141         if (aux.getSiguiente() != null) {
142             aux.getSiguiente().setAnterior(aux.getAnterior());
143         }
144         tamano--;
145     }
146
147 1 usage
148 public int buscar(T valor){
149     NodoDoble<T> aux = this.head;
150     int indice = 1;
151     while (aux != null){
152         if(aux.getDato().equals(valor)){
153             return indice;
154         }
155         aux=aux.getSiguiente();
156         indice++;
157     }
158     System.out.println("No se encontro el elemento");
159     return -1;
160
161 1 usage
162 public void actualizar (T aBuscar , T valor){
163     NodoDoble<T> aux = this.head;
164     while (aux != null) {
165         if (aux.getDato().equals(aBuscar)) {
166             aux.setDato(valor);
167             return;
168         }
169         aux = aux.getSiguiente();
170     }
171     System.out.println("No se encontro el elemento");
172 }

```

```

173
3 usages
174 public void transversal(int direccion) {
175     if (direccion == 0) {
176         NodoDoble<T> aux = this.head;
177         while (aux != null) {
178             System.out.println(aux.getDatos());
179             aux = aux.getSiguiente();
180         }
181     } else {
182         NodoDoble<T> aux = this.tail;
183         while (aux != null) {
184             System.out.println(aux.getDatos());
185             aux = aux.getAnterior();
186         }
187     }
188 }
189 }

```

Código Fuente Main:

```

1 package fes.aragon.inicio;
2
3 import fes.aragon.clases.DoubleLinkedList;
4
5 public class Main {
6     public static void main(String[] args) {
7
8         DoubleLinkedList<Integer> listaObjects = new DoubleLinkedList<>();
9         listaObjects.agregarAlInicio( valor: 50);
10        listaObjects.agregarAlFinal( valor: 60);
11        listaObjects.agregarAlFinal( valor: 65);
12        listaObjects.agregarAlFinal( valor: 70);
13        listaObjects.agregarAlFinal( valor: 80);
14        listaObjects.agregarAlFinal( valor: 90);
15
16        System.out.println("Lista Principal");
17        listaObjects.transversal( direccion: 0);
18        System.out.println(" ");
19
20        System.out.println("Lista actualizada a : Eliminar elemento de la posicion 2");
21        listaObjects.eliminar( posicion: 1);
22        listaObjects.transversal( direccion: 0);
23
24        System.out.println();
25        System.out.println("Lista Actualizada a: Actualizar el cuarto elemento a 88");
26        listaObjects.actualizar( aBuscar: 80, valor: 88);
27        listaObjects.transversal( direccion: 0);
28
29        System.out.println();
30        System.out.println("Busca el valor de 80 dentro de la lista: ");
31        listaObjects.buscar( valor: 80);
32    }
33 }

```

Ejecución del Programa:

```
/opt/java/jdk-17.0.2/bin/java -javaagent:/opt/java/idea-IC-233.14015.106/lib/idea_rt.jar=43203:/opt/java/idea-IC-233.14015.106/bin -Dfile.encoding=UTF-8
Lista Principal
50
60
65
70
80
90

Lista actualizada a : Eliminar elemento de la posicion 2
50
65
70
80
90

Lista Actualizada a: Actualizar el cuarto elemento a 88
50
65
70
88
90

Busca el valor de 80 dentro de la lista:
No se encontro el elemento

Process finished with exit code 0
```