

# **ESTRUCTURES DE DADES**

## **-LLISTA DOBLEMENT ENCADENADA-**

Alumne: Carlos Martínez

Professor: Albert Solé

## Classes utilitzades

- Ciutadà: conté els atributs i mètodes pròpis que necessitem per a poder tractar amb un ciutadà
- Ciutadalterator: classe necessaria per a poder utilitzar l'iterator
- LlistaDoble: llista doblement encadenada amb tots els seus atributs i mètodes propis
- Nodo: classe que conté l'enllaç al següent node, a l'anterior, i la dada que guardem al mateix node
- TADCiutada: conté la definició dels mètodes que tindrà la nostra llista doblement encadenada

## Aspectes a destacar

### Problemes trobats durant el desenvolupament

-Casting entre classes, ja que a vegades operem amb dades genèriques, a vegades tipus primitius o de tipus ciutadà.

-Mal enllaçament entre nodes: a vegades enllaçava l'element actual amb el següent però no el següent amb l'actual i aixó causava problemes.

-Error al treballar amb mètodes genèrics ja que a vegades alguns eren de tipus <T> i no <T extends Comparable<T>> .

### Creació de les excepcions pròpies

-ElementoNoEncontrado:

```
1 package Exceptions;
2
3 public class ElementoNoEncontrado extends Exception{
4     private static final long serialVersionUID = 1L;
5
6     public ElementoNoEncontrado(int n) {
7         super("Se han recorrido "+n+" posiciones pero no se ha encontrado el elemento");
8     }
9 }
10
```

-NoSePuede:

```
1 package Exceptions;
2
3 public class NoSePuede extends Exception{
4     private static final long serialVersionUID = 1L;
5     public NoSePuede(int posi) {
6         super("No se puede tratar el elemento en la posición: "+posi);
7     }
8 }
9
```

# Mètodes

## Creació de la llista

```
13  /**
14   * Creació de la llista doblement encadenada donada una mida
15   * @param nElems
16   */
17  public ListaDoble(int nElems) {
18      int i=0;
19      inicio=new Nodo(null);
20      this.nElems++;
21      Nodo<T>nodoAux=inicio;
22      while(i<nElems-1) {
23          nodoAux.siguiete=new Nodo(i);
24          nodoAux.siguiete.anterior=nodoAux;
25          nodoAux=nodoAux.siguiete;
26          this.nElems++;i++;
27      }
28  }
29  /**
30   * Inicialització del primer element de la llista encadenada
31   */
32  public void crear() {
33      inicio=new Nodo(null,null,null);
34  }
```

## Inserció d'un element al final

```
/**
 * Mètode per a inserir un element al final de la llista
 * @param data: dada a afegir
 */
public void Inserir(T data) {
    if(!empty()) {
        fin=new Nodo(null,fin,data);
        fin.anterior.siguiete=fin;
    }
    else {
        inicio=fin=new Nodo(data);
    }
    nElems++;
}
```

## Inserció d'un element donat un índex

```
59  /**
60   * Mètode per a inserir un element donada una posició
61   * @param posi: index on afegir
62   * @param data: dada a afegir
63   */
64  public void Inserir(int posi,T data) throws NoSePuede {
65      if(posi<0||posi>nElems)
66          throw new NoSePuede(posi);
67      Nodo<T>aux=new Nodo<T>(data);
68      if(inicio==null) {
69          inicio=aux;
70          fin=aux;
71      }
72      else if(posi==0) {
73          aux.siguiete=inicio;
74          inicio.anterior=aux;
75          inicio=aux;
76      }
77      else if(posi==nElems) {
78          aux.anterior=fin;
79          fin.siguiete=aux;
80          fin=aux;
81      }
82      else {
83          Nodo<T>nodoAux=inicio;
84          for(int i=1;i<posi;i++) {
85              nodoAux=nodoAux.siguiete;
86          }
87          aux.siguiete=nodoAux.siguiete;
88          nodoAux.siguiete=aux;
89          aux.anterior=nodoAux;
90          aux.siguiete.anterior=aux;
91      }
92      nElems++;
93  }
```

## Obtenció d'un element donat un índex

```
94  /**
95   * Mètode per a retornar l'element d'una posició donada en cas de ser possible
96   * @param posi: index de la posició
97   */
98  public T Obtener(int posi)throws NoSePuede {
99      if(posi<nElems&&posi>=0) {
100          int i=0;
101          Nodo aux=inicio;
102          while(i<posi) {
103              aux=aux.siguiete;
104              i++;
105          }
106          return (T)aux.data;
107      }
108      else {
109          throw new NoSePuede(posi);
110      }
111  }
112 }
```

## Esborrat d'un element donada una posició

```
114- /**
115  * Mètode per a esborrar un node, utilitzat al mètode 'Esborrar(int posi)'
116  * @param elem: node a eliminar de la llista
117  */
118- public void eliminarNodo(Nodo elem) {
119     if(inicio==null||elem==null)
120         return;
121     if(inicio==elem)
122         inicio=elem.siguiete;
123
124     //Ant<->Actual<->Siguiete
125     //Ant<->Siguiete
126     if(elem.siguiete!=null)
127         elem.siguiete.anterior=elem.anterior;
128
129     if(elem.anterior!=null)
130         elem.anterior.siguiete=elem.siguiete;
131 }
132- /**
133  * Mètode per a esborrar un element donada una posició
134  * @param posi: índex de l'element que es vol esborrar
135  */
136- public void Esborrar(int posi) {
137     if(posi>0 && posi<nElems) {
138         Nodo aux=inicio;
139         int i;
140         for(i=1;aux!=null&&i<posi;i++) {
141             aux=aux.siguiete;
142         }
143         if(aux==null)
144             return;
145
146         eliminarNodo(aux);
147         nElems--;
148     }
149 }
150
151 }
```

## Obtenció de la longitud de la llista

```
200- /**
201  * Mètode que retorna la longitud de la llista actual
202  */
203- public int Longitud() {
204     return nElems;
205 }
```














## Joc de proves

Primer de tot he creat una llista amb 4 elements per no h'aver d'introduir-los manualment i li passo al mètode on es criden els mètodes.

```
14      ListaDoble <Ciutada>listita=new ListaDoble<Ciutada>();
15      Ciutada ciudadano=new Ciutada("Carlos","Martínez","49424598J");
16      Ciutada ciudadano2=new Ciutada("Genis","Martínez","49422343K");
17      Ciutada ciudadano3=new Ciutada("David","Martí","77726323A");
18      Ciutada ciudadano4=new Ciutada("Albert","Solé","49424598Z");
19      listita.Inserir(ciudadano);
20      listita.Inserir(ciudadano2);
21      listita.Inserir(ciudadano3);
22      listita.Inserir(ciudadano4);
23      programaPrinc(listita);
```

### Crear llista

▼  llista	ListaDoble<T> (id=39)
▼  fin	Nodo<T> (id=47)
>  anterior	Nodo<T> (id=49)
> ▲ data	Ciutada (id=44)
○ siguiente	null
▼  inicio	Nodo<T> (id=48)
○ anterior	null
> ▲ data	Ciutada (id=41)
▼  siguiente	Nodo<T> (id=46)
>  anterior	Nodo<T> (id=48)
> ▲ data	Ciutada (id=42)
▼  siguiente	Nodo<T> (id=49)
>  anterior	Nodo<T> (id=46)
> ▲ data	Ciutada (id=43)
▼  siguiente	Nodo<T> (id=47)
>  anterior	Nodo<T> (id=49)
> ▲ data	Ciutada (id=44)
○ siguiente	null
 nElems	4

### Imprimir llista

```
BENVINGUT/UDA AL PROGRAMA:
1- Crear llista doblement enllaçada
2- Inserir un element al final de la llista
3- Inserir un element a una posició desitjada
4- Obtener element d'una posició escollida
5- Obtener longitud de la llista enllaçada
6- Esborrar element d'una posició escollida
7- Comprovar si existeix un element a la llista
8- Mostrar el contingut de la llista
0- SORTIR
R: 8
Ciutada [nom=Carlos, cognom=Martínez, DNI=49424598J]
Ciutada [nom=Genis, cognom=Martínez, DNI=49422343K]
Ciutada [nom=David, cognom=Martí, DNI=77726323A]
Ciutada [nom=Albert, cognom=Solé, DNI=49424598Z]
```

## Inserir al final

```
1- Crear llista doblement enllaçada
2- Inserir un element al final de la llista
3- Inserir un element a una posició desitjada
4- Obtenir element d'una posició escollida
5- Obtenir longitud de la llista enllaçada
6- Esborrar element d'una posició escollida
7- Comprovar si existeix un element a la llista
8- Mostrar el contingut de la llista
0- SORTIR
R: 2
Introdueix el nom i cognom (separats per un espai) del Ciutadà que vols afegir
Marc
Index 1 out of bounds for length 1
Elon Musk definit per defecte
Introdueix el DNI del ciutadà a afegir
46451548P
|
```

▼ ⓘ llista	ListaDoble<T> (id=39)
▼ ■ fin	Nodo<T> (id=78)
➤ ● anterior	Nodo<T> (id=45)
▼ ▲ data	Ciutada (id=77)
➤ ■ cognom	"Musk" (id=75)
➤ ■ DNI	"46451548P" (id=76)
➤ ■ nom	"Elon" (id=74)
● siguiente	null

## Inserir en una posició

```
Introdueix la posició on vols inserir l'element
3
Introdueix l'element que vols afegir
LLISTA ABANS DE PROVAR INSERCIÓ
Ciutada [nom=Carlos, cognom=Martínez, DNI=49424598J]
Ciutada [nom=Genis, cognom=Martínez, DNI=49422343K]
Ciutada [nom=David, cognom=Martí, DNI=77726323A]
Ciutada [nom=Albert, cognom=Solé, DNI=49424598Z]
Ciutada [nom=Elon, cognom=Musk, DNI=46451548P]
Introdueix el nom del ciutadà
Nil
Introdueix el cognom del ciutadà
Monfort
Introdueix el DNI del ciutadà
92348320
|
LLISTA DESPRÉS DE PROVAR INSERCIÓ
Ciutada [nom=Carlos, cognom=Martínez, DNI=49424598J]
Ciutada [nom=Genis, cognom=Martínez, DNI=49422343K]
Ciutada [nom=David, cognom=Martí, DNI=77726323A]
Ciutada [nom=Nil, cognom=Monfort, DNI=92348320]
Ciutada [nom=Albert, cognom=Solé, DNI=49424598Z]
Ciutada [nom=Elon, cognom=Musk, DNI=46451548P]
Introdueix la posició on vols inserir l'element
-1
Introdueix l'element que vols afegir
LLISTA ABANS DE PROVAR INSERCIÓ
Ciutada [nom=Carlos, cognom=Martínez, DNI=49424598J]
Ciutada [nom=Genis, cognom=Martínez, DNI=49422343K]
Ciutada [nom=David, cognom=Martí, DNI=77726323A]
Ciutada [nom=Albert, cognom=Solé, DNI=49424598Z]
Introdueix el nom del ciutadà
carlos
Introdueix el cognom del ciutadà
adasda
Introdueix el DNI del ciutadà
12312141
Exception in thread "main" Exceptions.NoSePuede: No se puede tratar el elemento en la posición: -1
    at Data.ListaDoble.Inserir(ListaDoble.java:66)
    at Programa.main.programaPrinc(main.java:83)
    at Programa.main.main(main.java:23)
```

## Buscar en una posición

```
4- Obténir element d'una posició escollida
5- Obténir longitud de la llista enllaçada
6- Esborrar element d'una posició escollida
7- Comprovar si existeix un element a la llista
8- Mostrar el contingut de la llista
0- SORTIR
R: 4
Introdueix la posició desitjada per a obtenir el seu element:
3
Ciutada [nom=Albert, cognom=Solé, DNI=49424598Z]
```

BENVINGUT/UDA AL PROGRAMA:

```
1- Crear llista doblement enllaçada
2- Inserir un element al final de la llista
3- Inserir un element a una posició desitjada
4- Obténir element d'una posició escollida
5- Obténir longitud de la llista enllaçada
6- Esborrar element d'una posició escollida
7- Comprovar si existeix un element a la llista
8- Mostrar el contingut de la llista
0- SORTIR
```

R: 4

Introdueix la posició desitjada per a obtenir el seu element:

12

```
Exception in thread "main" Exceptions.NoSePuede: No se puede tratar el elemento en la posición: 12
    at Data.ListaDoble.Obtenir(ListaDoble.java:109)
    at Programa.main.programaPrinc(main.java:95)
    at Programa.main.main(main.java:23)
```

## Buscar per DNI

```
7- Comprovar si existeix un element a la llista
8- Mostrar el contingut de la llista
0- SORTIR
R: 7
Introdueix l'element que vols buscar:
49424598J
S'han mirat 1 elements fins a trobar el que buscaves
```

BENVINGUT/UDA AL PROGRAMA:

```
1- Crear llista doblement enllaçada
2- Inserir un element al final de la llista
3- Inserir un element a una posició desitjada
4- Obténir element d'una posició escollida
5- Obténir longitud de la llista enllaçada
6- Esborrar element d'una posició escollida
7- Comprovar si existeix un element a la llista
8- Mostrar el contingut de la llista
0- SORTIR
```

R: 7

Introdueix l'element que vols buscar:

98234U289p

```
Exception in thread "main" Exceptions.ElementoNoEncontrado: Se han recorrido 4 posiciones pero no se ha encontrado el elemento
    at Data.ListaDoble.Buscar(ListaDoble.java:173)
    at Programa.main.programaPrinc(main.java:134)
    at Programa.main.main(main.java:23)
```