ESTRUCTURES DE DADES

-LLISTA DOBLEMENT ENCADENADA-

Alumne: Carlos Martínez

Professor: Albert Solé

Contenido

Classes utilitzades

- Ciutadà: conté els atributs i mètodes pròpis que necessitarem per a poder tractar amb un ciutadà
- CiutadaIterator: classe necessaria per a poder utilitzar l'iterator
- LlistaDoble: llista doblement encadenada amb tots els seus atributs i mètodes propis
- Nodo: clase que conté l'enllaç al següent node, a l'anterior, i la dada que guardem al mateix node
- TADCiutada: conté la definición 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 això 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:

```java
package Exceptions;

public class ElementoNoEncontrado extends Exception{
    private static final long serialVersionUID = 1L;

    public ElementoNoEncontrado(int n) {
        super("Se han recorrido "+n+" posiciones pero no se ha encontrado el elemento");
    }
}

```

-NoSePuede:

```java
package Exceptions;

public class NoSePuede extends Exception{
    private static final long serialVersionUID = 1L;
    public NoSePuede(int posi) {
        super("No se puede tratar el elemento en la posición: "+posi);
    }
}

```

Mètodes

Creació de la llista

```
3 usages  Carlos Martínez García-Villarrubia
10    public ListaDoble() { inicio=fin=null; }
13
```

Inserció d'un element al final

```
10 usages  1 override  Carlos Martínez García-Villarrubia
33  public void Inserir(T data) {
34      if(!empty()) {
35          fin=new Nodo<>( sig: null,fin,data);
36          fin.anterior.siguiente=fin;
37
38      }
39      else {
40          inicio=fin=new Nodo<T>(data);
41      }
42      nElems++;
43  }
```

Inserció d'un element donat un índex

```
1 usage  Carlos Martínez García-Villarrubia
49  public void Inserir(int posi,T data) throws NoSePuede {
50      if(posi<0||posi>nElems)
51          throw new NoSePuede(posi);
52      Nodo<T>aux=new Nodo<T>(data);
53      if(inicio==null) {
54          inicio=aux;
55          fin=aux;
56      }
57      else if(posi==0) {
58          aux.siguiente=inicio;
59          inicio.anterior=aux;
60          inicio=aux;
61      }
62      else if(posi==nElems) {
63          aux.anterior=fin;
64          fin.siguiente=aux;
65          fin=aux;
66      }
67      else {
68          Nodo<T>nodoAux=inicio;
69          for(int i=1;i<posi;i++) {
70              nodoAux=nodoAux.siguiente;
71          }
72          aux.siguiente=nodoAux.siguiente;
73          nodoAux.siguiente=aux;
74          aux.anterior=nodoAux;
75          aux.siguiente.anterior=aux;
76      }
77      nElems++;
78  }
```

Obtenció d'un element donat un índex

```
3 usages  Carlos Martínez García-Villarrubia + 1
public T Obtenir(int posi)throws NoSePuede {
    if(posi<nElems&&posi>=0) {
        int i=0;
        Nodo<T> aux=inicio;
        while(i<posi) {
            aux=aux.siguiente;
            i++;
        }
        return aux.data;
    }
    else {
        throw new NoSePuede(posi);
    }

}
```

Esborrat d'un element donada una posición

```
1 usage  Carlos Martínez García-Villarrubia
public void eliminarNodo(Nodo<T> elem) {
    if(inicio==null||elem==null)
        return;
    if(inicio==elem)
        inicio=elem.siguiente;

    //Ant<->Actual<->Siguiente
    //Ant<->Siguiente
    if(elem.siguiente!=null)
        elem.siguiente.anterior=elem.anterior;

    if(elem.anterior!=null)
        elem.anterior.siguiente=elem.siguiente;
}
/**
 * Mètode per a esborrar un element donada una posició
 * @param posi: index de l'element que es vol esborrar
 */
1 usage  Carlos Martínez García-Villarrubia + 1
public void Esborrar(int posi) {
    if(posi>=0 && posi<nElems) {
        Nodo<T> aux=inicio;
        int i;
        for(i=1;aux!=null&&i<posi;i++) {
            aux=aux.siguiente;
        }
        if(aux==null)
            return;

        eliminarNodo(aux);
        nElems--;

    }

}
```

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'haver d'introduir-los manualment i li passo al mètode on es criden els mètodes de la classe *ListaDoble*.

```
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","494245987");
19          listita.Inserir(ciudadano);
20          listita.Inserir(ciudadano2);
21          listita.Inserir(ciudadano3);
22          listita.Inserir(ciudadano4);
23          programaPrinc(listita);
```

Crear llista

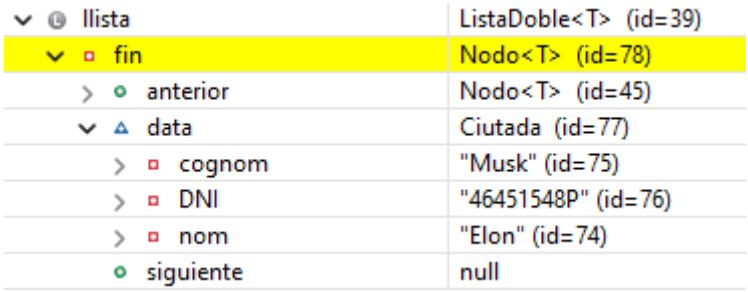
llista	ListaDoble<T> (id=39)
fin	Nodo<T> (id=47)
anterior	Nodo<T> (id=49)
data	Ciutada (id=44)
siguiente	null
inicio	Nodo<T> (id=48)
anterior	null
data	Ciutada (id=41)
siguiente	Nodo<T> (id=46)
anterior	Nodo<T> (id=48)
data	Ciutada (id=42)
siguiente	Nodo<T> (id=49)
anterior	Nodo<T> (id=46)
data	Ciutada (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- 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: 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
```



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- 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: 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- 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: 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- 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: 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)
```

Codi Font

```
package Data;
import java.util.*;

import Exceptions.*;
public class ListaDoble<T extends Comparable<T>> implements TADCiutada
<T>,Iterable<Ciutada>{
   protected Nodo<T> inicio,fin;
   private int nElems;
   private int posicioIterator;

   public ListaDoble() {
      inicio=fin=null;
   }


   /**
    * Inicialització del primer element de la llista encadenada
    */
   public void crear() {
      inicio=new Nodo(null,null,null);
   }
   /**
    * Mètode que comprova si la llista està plena o no
    * @return true if empty | false if not
    */
   public boolean empty() {

      return inicio==null;
   }
   /**
    * 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.siguiente=fin;

      }
      else {
         inicio=fin=new Nodo<T>(data);
      }
      nElems++;
   }
   /**
    * Mètode per a inserir un element donada una posició
    * @param posi: index on afegir
    * @param data: dada a afegir
    */
   public void Inserir(int posi,T data) throws NoSePuede {
      if(posi<0||posi>nElems)
         throw new NoSePuede(posi);
      Nodo<T>aux=new Nodo<T>(data);
      if(inicio==null) {
         inicio=aux;
         fin=aux;
      }
      else if(posi==0) {
         aux.siguiente=inicio;
```

```
            inicio.anterior=aux;
            inicio=aux;
        }
        else if(posi==nElems) {
            aux.anterior=fin;
            fin.siguiente=aux;
            fin=aux;
        }
        else {
            Nodo<T>nodoAux=inicio;
            for(int i=1;i<posi;i++) {
                nodoAux=nodoAux.siguiente;
            }
            aux.siguiente=nodoAux.siguiente;
            nodoAux.siguiente=aux;
            aux.anterior=nodoAux;
            aux.siguiente.anterior=aux;
        }
        nElems++;
    }
    /**
     * Mètode per a retornar l'element d'una posició donada en cas de
ser possible
     * @param posi: index de la posició
     */
    public T Obtenir(int posi)throws NoSePuede {
        if(posi<nElems&&posi>=0) {
            int i=0;
            Nodo<T> aux=inicio;
            while(i<posi) {
                aux=aux.siguiente;
                i++;
            }
            return aux.data;
        }
        else {
            throw new NoSePuede(posi);
        }

    }

    /**
     * Mètode per a esborrar un node, utilitzat al mètode 'Esborrar(int
posi)'
     * @param elem: node a eliminar de la llista
     */
    public void eliminarNodo(Nodo<T> elem) {
        if(inicio==null||elem==null)
            return;
        if(inicio==elem)
            inicio=elem.siguiente;

        //Ant<->Actual<->Siguiente
        //Ant<->Siguiente
        if(elem.siguiente!=null)
            elem.siguiente.anterior=elem.anterior;

        if(elem.anterior!=null)
            elem.anterior.siguiente=elem.siguiente;
    }
    /**
```

```
    * Mètode per a esborrar un element donada una posició
    * @param posi: índex de l'element que es vol esborrar
    */
   public void Esborrar(int posi) {
      if(posi>=0 && posi<nElems) {
         Nodo<T> aux=inicio;
         int i;
         for(i=1;aux!=null&&i<posi;i++) {
            aux=aux.siguiente;
         }
         if(aux==null)
            return;

         eliminarNodo(aux);
         nElems--;

      }

   }
   /**
    * Mètode per a comprovar si existeix a la llista un element passat
per paràmetre
    * @param dato: element que volem buscar a la llista
    */
   public int Buscar(T dato)throws ElementoNoEncontrado {
      int n=1,i=0;
      Nodo<T> aux=inicio;
      String temp;

      while(i<nElems) {
         if(aux.data.compareTo(dato)==0)
            return n;

         aux=aux.siguiente;
         n++;i++;
      }
      throw new ElementoNoEncontrado(i);

   }

   /**
    * Mètode de l'iterator
    */
   public Iterator<Ciutada>iterator(){
      return new CiutadaIterator<T>(this);
   }
   /**
    * Mètode per a duplicar llista actual
    * @return copia
    */
   public ListaDoble<T> copia() {
      return this;
   }
   /**
    * Mètode per a imprimir la llista, utilitzem el mètode iterator()
    */
   public void recorrer() {
      for (Ciutada ciutada : this) {
         System.out.println(ciutada);
      }
   }
```

```
    /**
     * Mètode que retorna la longitud de la llista actual
     */
    public int Longitud() {
        return nElems;
    }


}
```

```
package Programa;

import java.util.InputMismatchException;
import java.util.*;

import Data.*;
import Exceptions.ElementoNoEncontrado;
import Exceptions.NoSePuede;
public class main {
    private static Scanner scan;
    public static void main(String[] args) throws ElementoNoEncontrado,
NoSePuede {
        // TODO Auto-generated method stub
        ListaDobleOrdenada<Integer>listaux=new ListaDobleOrdenada<>();
        listaux.Inserir(12);
        listaux.Inserir(13);
        listaux.Inserir(14);
        listaux.Inserir(15);
        listaux.Inserir(16);
        Boolean bool=false;
        ListaDoble <Ciutada>listita=new ListaDoble<Ciutada>();
        Ciutada ciudadano=new Ciutada("Carlos","Martínez","49424598J");
        Ciutada ciudadano2=new Ciutada("Genis","Martínez","49422343K");
        Ciutada ciudadano3=new Ciutada("David","Martí","77726323A");
        Ciutada ciudadano4=new Ciutada("Albert","Solé","49424598J");
        listita.Inserir(ciudadano);
        listita.Inserir(ciudadano2);
        listita.Inserir(ciudadano3);
        listita.Inserir(ciudadano4);
        System.out.println(listita.Buscar(ciudadano4));
        programaPrinc(listita);

        listita.recorrer();

    }
    public static<T> void programaPrinc(ListaDoble<Ciutada> llista)
throws ElementoNoEncontrado, NoSePuede {
        int menOpt;
        //ListaDoble llista=null;

        do {
        mostrarMenu();
        scan=new Scanner(System.in);
        menOpt=llegirOpcio();
        switch(menOpt) {
        /**
         * 1-Creació de la llista doblement encadenada
         */
```

```
      case 1:
         llista=new ListaDoble<>();
         break;
      /**
       * 2-Demanem a l'usuari un Ciutadà i l'afegim a la llista
       */
      case 2:
         String name=new String();
         String lastName=new String();
         String id=new String();
         if(llista==null)
            System.out.println("ERR: la llista no s'ha creat encara");
         else {
         System.out.println("Introdueix el nom i cognom (separats per
un espai) del Ciutadà que vols afegir");
         scan.nextLine();
         String str=scan.nextLine();
         try {
            name=str.split(" ")[0];
            lastName=str.split(" ")[1];
         }catch(ArrayIndexOutOfBoundsException e) {
            System.out.println(e.getMessage());
            System.out.println("Elon Musk definit per defecte");
            name="Elon";
            lastName="Musk";
         }
         System.out.println("Introdueix el DNI del ciutadà a afegir");
         id=scan.next();
         llista.Inserir(new Ciutada(name,lastName,id));
         }
         break;
      /**
       * 3-Demanem una posició i un usuari i l'afegim a la llista en
cas de ser possible
       */
      case 3:
         if(llista==null)
            System.out.println("ERR: la llista no s'ha creat encara");
         else {
         //scan=new Scanner(System.in);
         System.out.println("Introdueix la posició on vols inserir
l'element");
         int posi=scan.nextInt();
         System.out.println("Introdueix l'element que vols afegir");
         System.out.println("LLISTA ABANS DE PROVAR INSERCIÓ");
         llista.recorrer();
         llista.Inserir(posi,readCiutada());}
         System.out.println();
         System.out.println("LLISTA DESPRÉS DE PROVAR INSERCIÓ");
         llista.recorrer();
         break;
      /**
       * 4-Demanem una posició i retornem el seu element en cas de ser
possible
       */
      case 4:
         System.out.println("Introdueix la posició desitjada per a
obtenir el seu element: ");
         int ind=scan.nextInt();

         System.out.println(llista.Obtenir(ind));
```

```
            break;
         /**
          * 5-Retornem la longitud de la llista doblement encadenada
          */
         case 5:
            System.out.println("La longitud de la llista doblement
encadenada es "+llista.Longitud());
            break;
         /**
          * 6-Demanem una posició i esborrem l'element que es troba en
ella en cas de ser possible
          */
         case 6:
            int toDelete=0;
            int length=llista.Longitud();
            System.out.println("Introdueix l'index de la posició que vols
eliminar");
            try {
            toDelete=scan.nextInt();
            }catch(NumberFormatException e) {
               System.out.println(e.getMessage());
            }
            System.out.println("LLISTA ABANS DE PROVAR ELIMINACIÓ");
            llista.recorrer();
            System.out.println();
            llista.Esborrar(toDelete);
            if(llista.Longitud()!=length)
               System.out.println("S'ha esborrat l'element
correctament");
            else
               System.out.println("No s'ha esborrat cap element");
            System.out.println("LLISTA DESPRÉS DE PROVAR ELIMINACIÓ");
            llista.recorrer();
            break;
         /**
          * 7-Demanem un número de DNI i intentem trobar si hi ha alguna
coincidència
          */
         case 7:
            System.out.println("Introdueix l'element que vols buscar: ");
            //scan=new Scanner(System.in);
            String temp=scan.next();
            int n=llista.Buscar(new Ciutada("X","Y",temp));
            System.out.println("S'han mirat "+n+ " elements fins a trobar
el que buscaves");
            break;
         case 8:
            llista.recorrer();
            break;
         default:

            break;
         }
         System.out.println();
         }while(menOpt!=0);
      }


      public static void mostrarMenu() {
         System.out.println("BENVINGUT/UDA AL PROGRAMA: ");
```

```
      System.out.println("1- Crear llista doblement enllaçada");
      System.out.println("2- Inserir un element al final de la
llista");
      System.out.println("3- Inserir un element a una posició
desitjada");
      System.out.println("4- Obtenir element d'una posició
escollida");
      System.out.println("5- Obtenir longitud de la llista
enllaçada");
      System.out.println("6- Esborrar element d'una posició
escollida");
      System.out.println("7- Comprovar si existeix un element a la
llista");
      System.out.println("8- Mostrar el contingut de la llista");
      System.out.println("0- SORTIR");
      System.out.printf("R: ");
   }

   /**
    * Demanem la opció del menú que vol executar l'usuari
    * @return num d'opció
    */
   public static int llegirOpcio() {
      scan=new Scanner(System.in);
      int resp=0;
      do {
      try {
         resp=scan.nextInt();
      }catch(NumberFormatException e) {
         resp=1;
         System.out.println("Format de número incorrecte, opció 1
assignada per defecte");
      }catch(InputMismatchException e) {
         System.out.println(e.getMessage());
      }
      }while(resp<0||resp>8);
      return resp;
   }
   /**
    * Llegim dades d'un nou usuari que crearem i retornarem
    * @return new Ciutadà(nom,cognom,DNI);
    */
   public static Ciutada readCiutada() {
      Ciutada aux=null;
      String nom,cognom,DNI=new String();
      System.out.println("Introdueix el nom del ciutadà");
      nom=scan.next();
      System.out.println("Introdueix el cognom del ciutadà");
      cognom=scan.next();
      System.out.println("Introdueix el DNI del ciutadà");
      DNI=scan.next();
      aux=new Ciutada(nom,cognom,DNI);
      return aux;
   }


}
```