

Alumne: Carlos Martínez

Professor: Albert Solé

**ESTRUCTURES DE DADES**

**-LLISTA DOBLEMENT ENCADENADA-**

Contenido

[Classes utilitzades 3](#_Toc106470759)

[Aspectes a destacar 3](#_Toc106470760)

[Problemes trobats durant el desenvolupament 3](#_Toc106470761)

[Creació de les excepcions pròpies 3](#_Toc106470762)

[Mètodes 4](#_Toc106470763)

[Creació de la llista 4](#_Toc106470764)

[Inserció d’un element al final 4](#_Toc106470765)

[Inserció d’un element donat un índex 4](#_Toc106470766)

[Obtenció d’un element donat un índex 5](#_Toc106470767)

[Esborrat d’un element donada una posición 5](#_Toc106470768)

[Obtenció de la longitud de la llista 6](#_Toc106470769)

[Joc de proves 6](#_Toc106470770)

[Crear llista 6](#_Toc106470771)

[Imprimir llista 7](#_Toc106470772)

[Inserir al final 7](#_Toc106470773)

[Inserir en una posició 8](#_Toc106470774)

[Buscar en una posición 8](#_Toc106470775)

[Buscar per DNI 9](#_Toc106470776)

[Codi Font 10](#_Toc106470777)

# 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 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:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

-NoSePuede:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

# Mètodes

## Creació de la llista

Texto

Descripción generada automáticamente

## Inserció d’un element al final

Texto

Descripción generada automáticamente

## Inserció d’un element donat un índex

Texto

Descripción generada automáticamente

## Obtenció d’un element donat un índex

Texto

Descripción generada automáticamente

## Esborrat d’un element donada una posición

Texto

Descripción generada automáticamente

## Obtenció de la longitud de la llista

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

# 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*.

Texto

Descripción generada automáticamente

## Crear llista

Tabla

Descripción generada automáticamente con confianza media

## Imprimir llista

Texto, Carta

Descripción generada automáticamente

## Inserir al final

Texto

Descripción generada automáticamente

Tabla

Descripción generada automáticamente

## Inserir en una posició

Texto

Descripción generada automáticamenteTexto, Carta

Descripción generada automáticamente

## Buscar en una posición

Texto, Carta

Descripción generada automáticamente

Texto

Descripción generada automáticamente

## Buscar per DNI

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente con confianza media

# 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;  
 }  
   
  
}