ESCOLA TÈCNICA SUPERIOR D'ENGINYERIA
Universitat Rovira i Virgili

[DEIM]

Estructura de Dades
Pràctica 1: Part 1

Llista doblement encadenada
curs 2021-22

Estudiant: Marc Fonseca (GEI)

Professor/a: Adam Alvarado

Grup: L5

ÍNDEX

1. Llista Doblement Encadenada

1.1 Joc de Proves

```
LlistaDobleEncadenada<Ciutada> llista = new LlistaDobleEncadenada<Ciutada>();

int posicio = 2;

Ciutada ciutada1 = new Ciutada("Pablo", "Garcia", "47223427M");
Ciutada ciutada2 = new Ciutada("Marc", "Fontseca", "48017307T");
Ciutada ciutada3 = new Ciutada("Jesus", "Sepulveda", "35718295R");
Ciutada ciutada4 = new Ciutada("Oscar", "Perez", "34578924G");
Ciutada ciutada5 = new Ciutada("Marc", "Fonseca", "48017307T");
Ciutada ciutadaNoInserit = new Ciutada("jeje", "NoEstoy", "12345678F");

// 1. Inserim de manera normal
System.out.println("[Insercio Nromal]");

llista.inserir(ciutada1);
llista.inserir(ciutada2);
llista.inserir(ciutada3);
llista.inserir(ciutada4);

System.out.println("Numero de elements = " +llista.longitud());
for (Ciutada c:llista) {
        System.out.println("\t"+c+"\n");
}
```

```
[Insercio Nromal]
Numero de elements = 4
        Pablo Garcia amb DNI: 47223427M

        Marc Fontseca amb DNI: 48017307T

        Jesus Sepulveda amb DNI: 35718295R

        Pablo Garcia amb DNI: 47223427M
```

```
// 2. Inserim en una certa posicio
System.out.println("[Insercio Posicio]");

try {
	llista.inserir(posicio, ciutada5);
} catch (foraDeRang e) {
	System.out.println(e);
}

System.out.println("Numero de elements = " +llista.longitud());
for (Ciutada c:llista) {
	System.out.println("\t"+c+"\n");
}
```

```
[Insercio Posicio]
Numero de elements = 5
        Pablo Garcia amb DNI: 47223427M

        Marc Fontseca amb DNI: 48017307T

        Marc Fonseca amb DNI: 48017307T

        Jesus Sepulveda amb DNI: 35718295R

        Pablo Garcia amb DNI: 47223427M
```

```
// 3. Comparem valors diferents
System.out.println("[Comparacio diferents]");

try {
	if (llista.obtenir(1).compareTo(llista.obtenir(0)) != 0) {
	System.out.println("Primer i segon tenen diferent DNI\n");
}
	else
	{
		System.out.println("Segon i tercer tenen el mateix DNI\n");
	}
} catch (foraDeRang e) {
	System.out.println(e);
}
```

```
[Comparacio diferents]
Primer i segon tenen diferent DNI
```

```
// 4. Comparem valors iguals
System.out.println("[Comparacio iguals]");

try {
	if (llista.obtenir(1).compareTo(llista.obtenir(2)) == 0) {
		System.out.println("Primer i tercer tenen el mateix DNI\n");
	}
	else {
		System.out.println("Primer i tercer tenen diferent DNI\n");
	}
} catch (foraDeRang e) {
	System.out.println(e);
}
```

```
[Comparacio iguals]
Primer i tercer tenen el mateix DNI
```

```
// 5. Obtenim una posicio que no existeix
System.out.println("[Obtencio no existeix]");

try {
	llista.obtenir(5);
} catch (foraDeRang e) {
	System.out.println(e);
}
```

```
[Obtencio no existeix]
Excepcions.foraDeRang: ERROR : La posicio ha d'estar entre el seguent rang de valors: [0 - 4]
```

```
// 6. Esborrem l'element insertat previament
System.out.println("[Esborrat Existeix]");

try {
	llista.esborrar(posicio);
} catch (foraDeRang e) {
	System.out.println(e);
}

System.out.println("Numero de elements = " +llista.longitud());
for (Ciutada c:llista) {
	System.out.println("\t"+c+"\n");
}
```

```
[Esborrat Existeix]
Numero de elements = 4
        Pablo Garcia amb DNI: 47223427M

        Marc Fontseca amb DNI: 48017307T

        Jesus Sepulveda amb DNI: 35718295R

        Pablo Garcia amb DNI: 47223427M
```

```
// 7. Esborrem un element que no existeix
System.out.println("[Esborrat no Existeix]");

try {
	llista.esborrar(5);
} catch (foraDeRang e) {
	System.out.println(e);
}
```

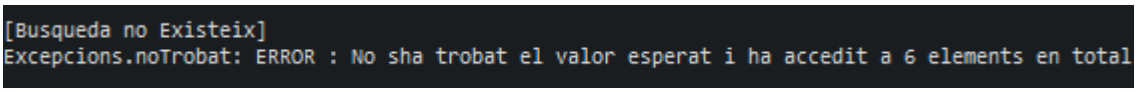
```
[Esborrat no Existeix]
Excepcions.foraDeRang: ERROR : La posicio ha d'estar entre el seguent rang de valors: [0 - 3]
```

```
// 8. Busquem un element que existeix
System.out.println("[Busqueda Existeix]");
try {
	System.out.println("Ha accedit a " +llista.buscar(ciutada1)+ " elements per trobar el ciutada\n");
} catch (noTrobat e) {
	System.out.println(e);
}
```

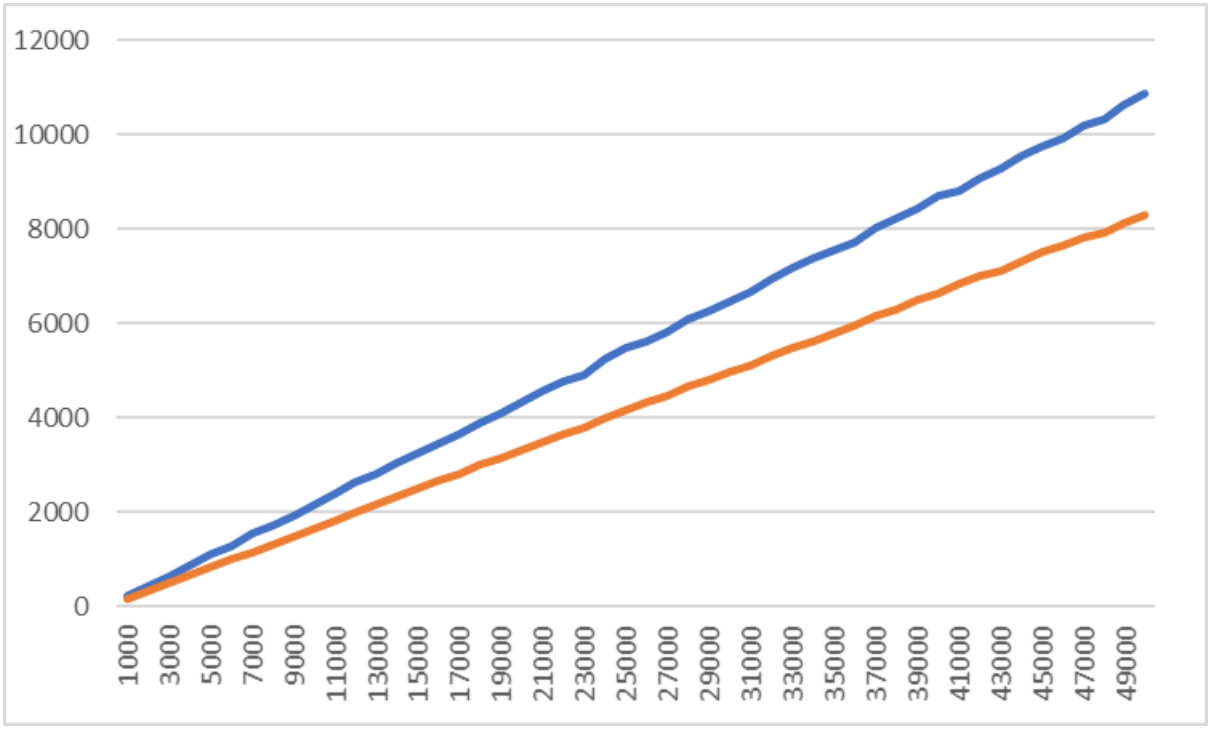
```
[Busqueda Existeix]
Ha accedit a 2 elements per trobar el ciutada
```

```
// 9. Busquem un element que no existeix
System.out.println("[Busqueda no Existeix]");

try {
	System.out.println("Ha accedit a " +llista.buscar(ciutadaNoInserit)+ " elements per
trobar el ciutada\n");
} catch (noTrobat e) {
	System.out.println(e);
}
```



1.2 Cost Computacional



1.3 Codi Font

TAD Generics

```
public interface TADGenerics<T> {

	/**
	 * Metode per inserir un element al final de la taula
	 * @param data - valor generalitzat
	 */
	public void inserir(T data);

	/**
	 * Metode per inserir un elemnt en una posicio de la taula
	 * @param posicio - posicio de la taula
	 * @param data - valor generalitzat
	 * @throws foraDeRang - error valor fora de rang
	 */
	public void inserir(int posicio, T data) throws foraDeRang;

	/**
	 * Metode per obtenir el valor duna posicio
	 * @param posicio - posicio de la taula
	 * @return data - valor generalitzat
	 * @throws foraDeRang - error valor fora de rang
	 */
	T obtenir(int posicio) throws foraDeRang;

	/**
	 * Metode per obtenir la longitud de la taula
	 * @return longitud de la taula
	 */
	public int longitud();

	/**
	 * Metode per esborrar una posicio de la taula
	 * @param posicio - posicio de la taula
	 * @throws foraDeRang - error valor fora de rang
	 */
	public void esborrar(int posicio) throws foraDeRang;

	/**
	 * Metode per buscar un valor en la taula
	 * @param data - valor generalitzat
	 * @return posicio de la taula
	 * @throws noTrobat - error valor no trobat
	 */
	public int buscar (T data) throws noTrobat;
}
```

LlistaDobleEncadenada

```
public interface TADGenerics<T> {

	/**
	 * Metode per inserir un element al final de la taula
	 * @param data - valor generalitzat
	 */
	public void inserir(T data);

	/**
	 * Metode per inserir un elemnt en una posicio de la taula
	 * @param posicio - posicio de la taula
	 * @param data - valor generalitzat
	 * @throws foraDeRang - error valor fora de rang
	 */
	public void inserir(int posicio, T data) throws foraDeRang;

	/**
	 * Metode per obtenir el valor duna posicio
	 * @param posicio - posicio de la taula
	 * @return data - valor generalitzat
	 * @throws foraDeRang - error valor fora de rang
	 */
	T obtenir(int posicio) throws foraDeRang;

	/**
	 * Metode per obtenir la longitud de la taula
	 * @return longitud de la taula
	 */
	public int longitud();

	/**
	 * Metode per esborrar una posicio de la taula
	 * @param posicio - posicio de la taula
	 * @throws foraDeRang - error valor fora de rang
	 */
	public void esborrar(int posicio) throws foraDeRang;

	/**
	 * Metode per buscar un valor en la taula
	 * @param data - valor generalitzat
	 * @return posicio de la taula
	 * @throws noTrobat - error valor no trobat
	 */
	public int buscar (T data) throws noTrobat;
}
```

NodeLlistaGenerica

```
public class NodeLlistaGenerica<T> {

	private T valor;
	private NodeLlistaGenerica<T> seguent;
	private NodeLlistaGenerica<T> anterior;

	/**
	 * Constructor
	 * @param valor
	 */
	public NodeLlistaGenerica (T valor) {
		this.valor = valor;
	}

	/**
	 * Getter valor
	 * @return valor
	 */
	public T getValor() {
		return valor;
	}

	/**
	 * Setter valor
	 * @param valor
	 */
	public void setValor(T valor) {
		this.valor = valor;
	}

	/**
	 * Getter seguent
	 * @return node seguent
	 */
	public NodeLlistaGenerica<T> getSeguent() {
		return seguent;
	}

	/**
	 * Setter seguent
	 * @param seguent - seguent node
	 */
	public void setSeguent(NodeLlistaGenerica<T> seguent) {
		this.seguent = seguent;
	}

	/**
```

```
	 * Getter anterior
	 * @return node anterior
	 */
	public NodeLlistaGenerica<T> getAnterior() {
		return anterior;
	}

	/**
	 * Setter anterior
	 * @param anterior - anterior node
	 */
	public void setAnterior(NodeLlistaGenerica<T> anterior) {
		this.anterior = anterior;
	}

	@Override
	public String toString() {
		return ""+valor;
	}

}
```

LlistaIterator

```
public class LlistaIterator<T extends Comparable<T>> implements Iterator<T> {
	private LlistaDobleEncadenada<T> llista;
	private int posicioIterator;

	/**
	 * Contructor de la llista iteradora
	 * @param ll - llista a iterar
	 */
	public LlistaIterator(LlistaDobleEncadenada<T> ll) {
		llista=new LlistaDobleEncadenada<T>();
		for (int i = 0; i < ll.longitud(); i++) {
			try {
				llista.inserir(ll.obtenir(i));
			} catch (foraDeRang e) {
				System.out.println(e);
			}

		}
		posicioIterator=0;	// ens preparem per a retornar els elements a partir de
la posicio 0
	}

	@Override
	public boolean hasNext() {
		return ((posicioIterator<llista.longitud()));
```

```
	}

	@Override
	public T next() {
		try {
			T aux=llista.obtenir(posicioIterator);
			posicioIterator++;
			return aux;
		} catch (foraDeRang e) {
			System.out.println(e);
		}
		T aux = null;
		return aux;
	}
}
```

Ciutada

```
public class Ciutada implements Comparable<Ciutada>{
	private String Nom, Cognom;
	private String DNI;

	/**
	 * Contructor Ciutada
	 * @param Nom - Nom del Ciutada
	 * @param Cognom - Cognom del Ciutada
	 * @param DNI - DNI del Ciutada
	 */
	public Ciutada (String Nom, String Cognom, String DNI){
		this.Nom = Nom;
		this.Cognom = Cognom;
		this.DNI = DNI;
	}

	@Override
	public int compareTo (Ciutada aux) {
		if (this.DNI.equals(aux.getDNI())) {
			return 0;
		}
		else {
			return 1;
		}
	}

	/**
	 * Getter nom
	 * @return nom del ciutada
	 */
	public String getNom() {
```

```
		return Nom;
	}

	/**
	 * Setter nom
	 * @param nom - nom del ciutada
	 */
	public void setNom(String nom) {
		Nom = nom;
	}

	/**
	 * Getter cognom
	 * @return cognom del ciutada
	 */
	public String getCognom() {
		return Cognom;
	}

	/**
	 * Setter cognom
	 * @param cognom - cognom del ciutada
	 */
	public void setCognom(String cognom) {
		Cognom = cognom;
	}

	/**
	 * gGetter dni
	 * @return dni del ciutada
	 */
	public String getDNI() {
		return DNI;
	}

	/**
	 * Setter dni
	 * @param dni - dni del ciutada
	 */
	public void setDNI(String dni) {
		DNI = dni;
	}

	@Override
	public String toString() {
		return Nom + " " + Cognom + " amb DNI: " + DNI;
	}
}
```

2. Taula Hash

2.1 Joc de Proves

```
TaulaHash<Ciutada, String> taulaHash = new TaulaHash<>(10);

Ciutada ciutada1 = new Ciutada("Pablo", "Garcia", "47223427M");
Ciutada ciutada2 = new Ciutada("Marc", "Fontseca", "48017307T");
Ciutada ciutada3 = new Ciutada("Jesus", "Sepulveda", "35718295R");
Ciutada ciutada4 = new Ciutada("Oscar", "Perez", "34578924G");
Ciutada ciutada5 = new Ciutada("Ramon", "Rey", "3541157H");
Ciutada ciutada6 = new Ciutada("Angel", "Rey", "48010773A");
Ciutada ciutada7 = new Ciutada("Font", "Vella", "54797147G");
Ciutada ciutada8 = new Ciutada("Benito", "Camela", "23456781E");
Ciutada ciutada9 = new Ciutada("Raquel", "Hernandez", "23415627Y");    Ciutada
ciutada2Repetit = new Ciutada("Marc", "Fonseca", "48017307T");
Ciutada ciutadaNoInserit = new Ciutada("jeje", "NoEstoy", "12345678F");

// 1. Inserim 8 elements a la taula
System.out.println("[Insercio Normal]\n");
taulaHash.inserir(ciutada1.getDNI(), ciutada1);
taulaHash.inserir(ciutada2.getDNI(), ciutada2);
taulaHash.inserir(ciutada3.getDNI(), ciutada3);
taulaHash.inserir(ciutada4.getDNI(), ciutada4);
taulaHash.inserir(ciutada5.getDNI(), ciutada5);
taulaHash.inserir(ciutada6.getDNI(), ciutada6);
taulaHash.inserir(ciutada7.getDNI(), ciutada7);
taulaHash.inserir(ciutada8.getDNI(), ciutada8);
System.out.println("Numero de elements: " + taulaHash.mida() + "\n");
System.out.println(taulaHash.toString());
```

```
[Insercio Normal]

Numero de elements: 8

Posició de la taula: 4 -> [valor = null, key = null] seguent = [valor = Pablo Garcia amb DNI: 47223427M, key = 47223427M] seguent = [valor = Benito Camela amb DNI: 23456781E, key = 23456781E] seguent = null

Posició de la taula: 5 -> [valor = null, key = null] seguent = [valor = Marc Fontseca amb DNI: 48017307T, key = 48017307T] seguent = [valor = Ramon Rey amb DNI: 3541157H, key = 3541157H] seguent = [valor = Font Vella a

Posició de la taula: 8 -> [valor = null, key = null] seguent = [valor = Oscar Perez amb DNI: 34578924G, key = 34578924G] seguent = null

Posició de la taula: 9 -> [valor = null, key = null] seguent = [valor = Jesus Sepulveda amb DNI: 35718295R, key = 35718295R] seguent = [valor = Angel Rey amb DNI: 48010773A, key = 48010773A] seguent = null
```

```
// 2. Inserim l'element numero 8 per tal de redimensionar-la
System.out.println("[Insercio Redimensionada]\n");
taulaHash.inserir(ciutada9.getDNI(), ciutada9);
System.out.println("Numero de elements: " + taulaHash.mida() + "\n");
System.out.println(taulaHash.toString());
```

```
[Insercio Redimensionada]

Numero de elements: 9

Posició de la taula: 3 -> [valor = null, key = null] seguent = [valor = Raquel Hernandez amb DNI: 23415627Y, key = 23415627Y] seguent = null

Posició de la taula: 5 -> [valor = null, key = null] seguent = [valor = Font Vella amb DNI: 54797147G, key = 54797147G] seguent = null

Posició de la taula: 9 -> [valor = null, key = null] seguent = [valor = Jesus Sepulveda amb DNI: 35718295R, key = 35718295R] seguent = null

Posició de la taula: 14 -> [valor = null, key = null] seguent = [valor = Pablo Garcia amb DNI: 47223427M, key = 47223427M] seguent = [valor = Benito Camela amb DNI: 23456781E, key = 23456781E] seguent = null

Posició de la taula: 15 -> [valor = null, key = null] seguent = [valor = Marc Fontseca amb DNI: 48017307T, key = 48017307T] seguent = [valor = Ramon Rey amb DNI: 3541157H, key = 3541157H] seguent = null

Posició de la taula: 18 -> [valor = null, key = null] seguent = [valor = Oscar Perez amb DNI: 34578924G, key = 34578924G] seguent = null

Posició de la taula: 19 -> [valor = null, key = null] seguent = [valor = Angel Rey amb DNI: 48010773A, key = 48010773A] seguent = null
```

```
// 3. Inserim un element repetit, per modificar el seu valor
System.out.println("[Insercio Repetit]\n");
taulaHash.inserir(ciutada2Repetit.getDNI(), ciutada2Repetit);

System.out.println("Numero de elements: " + taulaHash.mida() + "\n");
System.out.println(taulaHash.toString());
```

```
[Insercio Repetit]
Numero de elements: 9
Posició de la taula: 3 -> [valor = null, key = null] seguent = [valor = Raquel Hernandez amb DNI: 23415627Y, key = 23415627Y] seguent = null
Posició de la taula: 5 -> [valor = null, key = null] seguent = [valor = Font Vella amb DNI: 54797147G, key = 54797147G] seguent = null
Posició de la taula: 9 -> [valor = null, key = null] seguent = [valor = Jesus Sepulveda amb DNI: 35718295R, key = 35718295R] seguent = null
Posició de la taula: 14 -> [valor = null, key = null] seguent = [valor = Pablo Garcia amb DNI: 47223427M, key = 47223427M] seguent = [valor = Benito Camela amb DNI: 23456781E, key = 23456781E] seguent = null
Posició de la taula: 15 -> [valor = null, key = null] seguent = [valor = Marc Fonseca amb DNI: 48017307T, key = 48017307T] seguent = [valor = Ramon Rey amb DNI: 3541157H, key = 3541157H] seguent = null
Posició de la taula: 18 -> [valor = null, key = null] seguent = [valor = Oscar Perez amb DNI: 34578924G, key = 34578924G] seguent = null
Posició de la taula: 19 -> [valor = null, key = null] seguent = [valor = Angel Rey amb DNI: 48010773A, key = 48010773A] seguent = null
```

```
// 4. Obtenim un valor de la taula
System.out.println("[Obtenció Inserit]");
try {
	System.out.println(taulaHash.obtenir(ciutada2.getDNI()) + "\n");
} catch (noTrobat e) {
	System.out.println(e);
}
```

```
[Obtenció Inserit]
Marc Fonseca amb DNI: 48017307T
```

```
// 5. Obtenim un valor que no esta a la Taula
System.out.println("[Obtenció no Inserit]");
try {
	System.out.println(taulaHash.obtenir(ciutadaNoInserit.getDNI()) + "\n");
} catch (noTrobat e) {
	System.out.println(e);
}
```

```
[Obtenció no Inserit]
Excepcions.noTrobat: ERROR : No sha trobat el valor esperat
```

```
// 6. Busquem un element que esta en la primera colisio
System.out.println("\n[Busqueda Primera Colisio]");
try {
System.out.println("S'han accedit " + taulaHash.buscar(ciutada9.getDNI()) + " element/s.\n");
} catch (noTrobat e) {
	System.out.println(e);
}
```

```
[Busqueda Primera Colisio]
S'han accedit 1 element/s.
```

```
// 7. Busquem un element que no esta en la primera colisio
System.out.println("[Busqueda no Primera Colisio]");
try {
System.out.println("S'han accedit " + taulaHash.buscar(ciutada8.getDNI()) + " element/s.\n");
} catch (noTrobat e) {
        System.out.println(e);
}
```

```
[Busqueda no Primera Colisio]
S'han accedit 2 element/s.
```

```
// 8. Busquem un element que no esta a la taula
System.out.println("[Busqueda no Existeix]");
try {
        System.out.println("S'han accedit " + taulaHash.buscar(ciutadaNoInserit.getDNI()) +
" " element/s.\n");
} catch (noTrobat e) {
        System.out.println(e);
}
```

```
[Busqueda no Existeix]
Excepcions.noTrobat: ERROR : No sha trobat el valor esperat i ha accedit a 1 elements en total
```

```
// 9. Esborrem un element que esta en la primera colisio
System.out.println("\n[Borrat Primera Colisio]");
try {
        taulaHash.esborrar(ciutada9.getDNI());
} catch (noTrobat e) {
        System.out.println(e);
}

System.out.println("Numero de elements: " + taulaHash.mida() + "\n");
System.out.println(taulaHash.toString());
```

```
[Borrat Primera Colisio]
Numero de elements: 8

Posició de la taula: 3 -> [valor = null, key = null] seguent = null

Posició de la taula: 5 -> [valor = null, key = null] seguent = [valor = Font Vella amb DNI: 54797147G, key = 54797147G] seguent = null

Posició de la taula: 9 -> [valor = null, key = null] seguent = [valor = Jesus Sepulveda amb DNI: 35718295R, key = 35718295R] seguent = null

Posició de la taula: 14 -> [valor = null, key = null] seguent = [valor = Pablo Garcia amb DNI: 47223427M, key = 47223427M] seguent = [valor = Benito Camela amb DNI: 23456781E, key = 23456781E] seguent = null

Posició de la taula: 15 -> [valor = null, key = null] seguent = [valor = Marc Fonseca amb DNI: 48017307T, key = 48017307T] seguent = [valor = Ramon Rey amb DNI: 3541157H, key = 3541157H] seguent = null

Posició de la taula: 18 -> [valor = null, key = null] seguent = [valor = Oscar Perez amb DNI: 34578924G, key = 34578924G] seguent = null

Posició de la taula: 19 -> [valor = null, key = null] seguent = [valor = Angel Rey amb DNI: 48010773A, key = 48010773A] seguent = null
```

```
// 10. Esborrem un element que no esta en la primera colisio
System.out.println("[Borrat no Primera Colisio]");
try {
	taulaHash.esborrar(ciutada8.getDNI());
} catch (noTrobat e) {
	System.out.println(e);
}

System.out.println("Numero de elements: " + taulaHash.mida() + "\n");
System.out.println(taulaHash.toString());
```

```
[Borrat no Primera Colisio]
Numero de elements: 7

Posició de la taula: 3 -> [valor = null, key = null] seguent = null

Posició de la taula: 5 -> [valor = null, key = null] seguent = [valor = Font Vella amb DNI: 54797147G, key = 54797147G] seguent = null

Posició de la taula: 9 -> [valor = null, key = null] seguent = [valor = Jesus Sepulveda amb DNI: 35718295R, key = 35718295R] seguent = null

Posició de la taula: 14 -> [valor = null, key = null] seguent = [valor = Pablo Garcia amb DNI: 47223427M, key = 47223427M] seguent = null

Posició de la taula: 15 -> [valor = null, key = null] seguent = [valor = Marc Fonseca amb DNI: 48017307T, key = 48017307T] seguent = [valor = Ramon Rey amb DNI: 3541157H, key = 3541157H] seguent = null

Posició de la taula: 18 -> [valor = null, key = null] seguent = [valor = Oscar Perez amb DNI: 34578924G, key = 34578924G] seguent = null

Posició de la taula: 19 -> [valor = null, key = null] seguent = [valor = Angel Rey amb DNI: 48010773A, key = 48010773A] seguent = null
```

```
// 11. Esborrem un element que no esta a la taula
System.out.println("[Borrat no Existeix]");
try {
	taulaHash.esborrar(ciutadaNoInserit.getDNI());
} catch (noTrobat e) {
	System.out.println(e);
}

System.out.println("Numero de elements: " + taulaHash.mida() + "\n");
System.out.println(taulaHash.toString());
```

```
[Borrat no Existeix]
Excepcions.noTrobat: ERROR : No sha trobat el valor esperat
Numero de elements: 7

Posició de la taula: 3 -> [valor = null, key = null] seguent = null

Posició de la taula: 5 -> [valor = null, key = null] seguent = [valor = Font Vella amb DNI: 54797147G, key = 54797147G] seguent = null

Posició de la taula: 9 -> [valor = null, key = null] seguent = [valor = Jesus Sepulveda amb DNI: 35718295R, key = 35718295R] seguent = null

Posició de la taula: 14 -> [valor = null, key = null] seguent = [valor = Pablo Garcia amb DNI: 47223427M, key = 47223427M] seguent = null

Posició de la taula: 15 -> [valor = null, key = null] seguent = [valor = Marc Fonseca amb DNI: 48017307T, key = 48017307T] seguent = [valor = Ramon Rey amb DNI: 3541157H, key = 3541157H] seguent = null

Posició de la taula: 18 -> [valor = null, key = null] seguent = [valor = Oscar Perez amb DNI: 34578924G, key = 34578924G] seguent = null

Posició de la taula: 19 -> [valor = null, key = null] seguent = [valor = Angel Rey amb DNI: 48010773A, key = 48010773A] seguent = null
```

```
// 12. Llistar valors de la taula
System.out.println("[Llistar Valors]");
LlistaDobleEncadenada<Ciutada> valors = new LlistaDobleEncadenada<Ciutada>();

valors = taulaHash.obtenirValors();

System.out.println("Numero de elements: " + valors.longitud());
for(Ciutada c:valors) {
	System.out.println("\t"+ c);
}
```

```
[Llistar Valors]
Numero de elements: 7
	Font Vella amb DNI: 54797147G
	Jesus Sepulveda amb DNI: 35718295R
	Pablo Garcia amb DNI: 47223427M
	Marc Fonseca amb DNI: 48017307T
	Ramon Rey amb DNI: 3541157H
	Oscar Perez amb DNI: 34578924G
	Angel Rey amb DNI: 48010773A
```

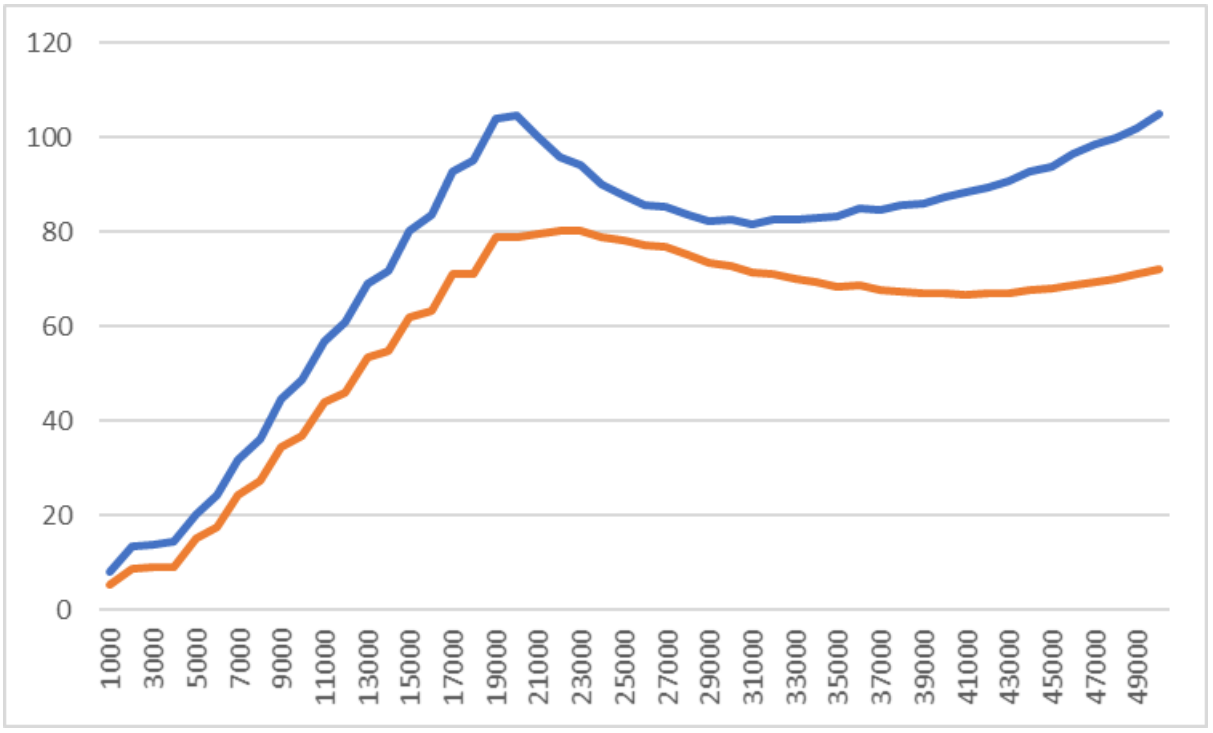
```
// 13. Llistar claus de la taula
System.out.println("\n[Llistar Claus]");
LlistaDobleEncadenada<String> claus = new LlistaDobleEncadenada<String>();

claus = taulaHash.obtenirClaus();

System.out.println("Numero de elements: " + claus.longitud());
for(String c:claus) {
	System.out.println("\t"+ c);
}
```

```
[Llistar Claus]
Numero de elements: 7
	54797147G
	35718295R
	47223427M
	48017307T
	3541157H
	34578924G
	48010773A
```

2.2 Cost Computacional



2.3 Codi Font

TADHash

```
public interface TADHash<T extends Comparable<T>, K extends Comparable<K>> {
	/**
	 * Metode per inserir element en una posicio al final
	 * @param key - "posicio de la taula"
	 * @param data - valor
	 */
	public void inserir(K key, T data);

	/**
	 * Metode per obtenir un valor d'una certa posicio
	 * @param key - "posicio de la taula"
	 * @return data - valor
	 * @throws noTrobat - error valor no trobat
	 */
	public T obtenir(K key) throws noTrobat;

	/**
	 * Metode per buscar un element d'una certa posicio
	 * @param key - "posicio de la taula"
	 * @return data - valor
	 * @throws noTrobat - error valor no trobat
	 */
	public int buscar (K key) throws noTrobat;

	/**
	 * Metode per obtenir el numero d'elements en la taula
	 * @return nElems
	 */
	public int mida();

	/**
	 * Metode per esborrar un element d'una certa posicio
	 * @param key - "posicio de la taula"
	 * @throws noTrobat - error valor no trobat
	 */
	public void esborrar(K key) throws noTrobat;

	/**
	 * Metode per llistar tots els valors de la taula
	 * @return llista de valors
	 */
	public LlistaDobleEncadenada<T> obtenirValors();

	/**
```

```
	 *  Metode per llistar totes les claus de la taula
	 * @return llista de claus
	 */
	public LlistaDobleEncadenada<K> obtenirClaus();

	/**
	 * Metode per obtenir el factor de carrega de la taula
	 * @return factor de carrega de la taula
	 */
	public float ObtenirFactorDeCàrrega();

}
```

TaulaHash

```
public class TaulaHash<T extends Comparable<T>, K extends Comparable <K>>
implements TADHash<T, K>{
	private int nElems;
	private NodeHashGeneric<T, K>[] taula;

	/**
	 * Funcio de Hash
	 * @param key - "posicio de la taula"
	 * @param taula - taula de Hash
	 * @return - posicio de la taula
	 */
	private int codeHash (K key, NodeHashGeneric<T, K>[] taula) {
		int contador = 0;
		for (int i = 0; i < key.toString().length(); i++) {
			contador = (contador + (2^i) * key.toString().charAt(i)) % taula.length;
		}
		return contador;
	}
	// 1. Constructor per inicialitzar la taula
	/**
	 * Constructor
	 * @param dim - tamany de la taula
	 */
	@SuppressWarnings("unchecked")
	public TaulaHash (int dim ) {
		taula = new NodeHashGeneric[dim];
		nElems = 0;
	}

	// 2. Funció per tal d'inserir un element a la taula de Hash
	//    Si l'element ja existia, actualitza el seu valor
	//    L'operació llença una excepció en cas que no es pugui inserir
	//    Si el factor de càrrega es superior a 0.75, haurem de redimensionar la taula
```

```
	@SuppressWarnings("unchecked")
	public void inserir(K key, T data) {
		int posicio, posicioAux;
		boolean trobat = false;
		NodeHashGeneric<T,K> auxNode, nodeAntic;

		if (ObtenirFactorDeCàrrega() > 0.75) {
			NodeHashGeneric<T,K>[] taulaAux = new
NodeHashGeneric[taula.length * 2];

			for (int i = 0; i < taula.length; i++) {
				if (taula[i] != null) {
					nodeAntic = taula[i];
					while (nodeAntic.getSeguent() != null) {
						posicioAux =
codeHash(nodeAntic.getSeguent().getKey(), taulaAux);

						auxNode = taulaAux[posicioAux];

						if (auxNode == null) {
							auxNode = new
NodeHashGeneric<T,K>();
							taulaAux[posicioAux] = auxNode;
						}
						else {
							while (auxNode.getSeguent() != null &&
!trobat) {
								if
(auxNode.getSeguent().getKey().compareTo(key) == 0) {

auxNode.getSeguent().setValor(data);
									trobat = true;
								}
								auxNode =
auxNode.getSeguent();
							}
						}
						if (!trobat) {
							auxNode.setSeguent(new
NodeHashGeneric<T, K> (nodeAntic.getSeguent().getValor(),
nodeAntic.getSeguent().getKey()));
						}

						nodeAntic = nodeAntic.getSeguent();
					}
				}
			}
```

```
                posicio = codeHash(key, taulaAux);
                auxNode = taulaAux[posicio];

                if (auxNode == null) {
                        auxNode = new NodeHashGeneric<T,K>();
                        taulaAux[posicio] = auxNode;
                }
                else {
                        trobat = false;
                        while (auxNode.getSeguent() != null && !trobat) {
                                if (auxNode.getSeguent().getKey().compareTo(key) ==
0) {
                                        auxNode.getSeguent().setValor(data);
                                        trobat = true;
                                }
                                auxNode = auxNode.getSeguent();
                        }
                }
                if (!trobat) {
                        auxNode.setSeguent(new NodeHashGeneric<T, K> (data,
key));
                        nElems++;
                }
                taula = taulaAux;
        }
        else {
                posicio = codeHash(key, taula);
                auxNode = taula[posicio];

                if (auxNode == null) {
                        auxNode = new NodeHashGeneric<T,K>();
                        taula[posicio] = auxNode;
                }
                else {
                        while (auxNode.getSeguent() != null && !trobat) {
                                if (auxNode.getSeguent().getKey().compareTo(key) ==
0) {
                                        auxNode.getSeguent().setValor(data);
                                        trobat = true;
                                }
                                auxNode = auxNode.getSeguent();
                        }
                }
                if (!trobat) {
                        auxNode.setSeguent(new NodeHashGeneric<T, K> (data,
key));
                        nElems++;
                }
```

```
            }
        }

        // 3. Funció que retorna l'element que té la clau K
        //    L'operació llença una excepció en cas que no es pugui obtenir
        public T obtenir(K key) throws noTrobat {
            int posicio = codeHash(key, taula);
            NodeHashGeneric<T, K> auxNode = taula[posicio];

            if (auxNode != null) {
                while (auxNode.getSeguent() != null) {
                    if (auxNode.getSeguent().getKey().compareTo(key) == 0) {
                        return auxNode.getSeguent().getValor();
                    }
                    auxNode = auxNode.getSeguent();
                }
            }
            throw new noTrobat();
        }

        // 4. Funció que comprova si un element està a la taula
        //    La funció retorna el cost de l'operació. Nombre d'elements que s'hagin accedit
per tal
        //    de comprovar si l'element existeix o no
        //    L'operació llença una excepció en cas que l'element no s'hagi trobat. La mateixa
        //    excepció contindrà informació del nombre d'elements que s'han accedit per
comprovar
        //    si l'element buscat existeix o no
        public int buscar (K key) throws noTrobat{
            int posicio = codeHash(key, taula);
            NodeHashGeneric<T, K> auxNode = taula[posicio];

            int contador = 1;

            if (auxNode != null) {
                while (auxNode.getSeguent() != null) {
                    if (auxNode.getSeguent().getKey().compareTo(key) == 0) {
                        return contador;
                    }
                    contador++;
                    auxNode = auxNode.getSeguent();
                }
            }
            throw new noTrobat(contador);
        }

        // 5. Retorna el nombre d'elements que conté la taula en aquest moment
        public int mida() {
```

```
		return nElems;
	}

	// 6. Funció per tal d'esborrar un element de la taula.
	//    L'operació llença una excepció en cas que l'element no s'hagi trobat
	public void esborrar(K key) throws noTrobat{
		int posicio = codeHash(key, taula);
		NodeHashGeneric<T, K> auxNode = taula[posicio];

		if (auxNode != null) {
			if (auxNode.getSeguent().getKey().compareTo(key) == 0) {
				nElems--;
				auxNode.setSeguent(auxNode.getSeguent().getSeguent());
				return;
			}
			else {
				auxNode = auxNode.getSeguent();

				while (auxNode.getSeguent() != null) {
					if (auxNode.getSeguent().getKey().compareTo(key) ==
0) {
						nElems--;

auxNode.setSeguent(auxNode.getSeguent().getSeguent());
						return;
					}
					auxNode = auxNode.getSeguent();
				}
			}
		}
		throw new noTrobat();
	}

	// 7. Retorna una llista amb tots els valors de la taula
	public LlistaDobleEncadenada<T> obtenirValors (){
		NodeHashGeneric<T, K> auxNode = null;;
		LlistaDobleEncadenada<T> llistaValors = new LlistaDobleEncadenada<T>();

		for (int i = 0; i < taula.length; i++) {
			if (taula[i] != null) {
				auxNode = taula[i];
				while (auxNode.getSeguent() != null) {
					llistaValors.inserir(auxNode.getSeguent().getValor());
					auxNode = auxNode.getSeguent();
				}
			}
		}
		return llistaValors;
```

```
        }

        // 8. Retorna una llista amb tots les claus de la taula
        public LlistaDobleEncadenada<K> obtenirClaus() {
                NodeHashGeneric<T, K> auxNode = null;;
                LlistaDobleEncadenada<K> llistaClaus = new LlistaDobleEncadenada<K>();

                for (int i = 0; i < taula.length; i++) {
                        if (taula[i] != null) {
                                auxNode = taula[i];
                                while (auxNode.getSeguent() != null) {
                                        llistaClaus.inserir(auxNode.getSeguent().getKey());
                                        auxNode = auxNode.getSeguent();
                                }
                        }
                }
                return llistaClaus;
        }

        // 9. Retorna el factor de càrrega actual (#elements/mida taula)
        public float ObtenirFactorDeCàrrega() {
                return (float)nElems/taula.length;
        }

        @Override
        public String toString() {
                String aux = "";
      for (int i = 0; i < taula.length; i++) {
         NodeHashGeneric<T, K> nodeAux = taula[i];
         if (nodeAux != null) {
            aux += "Posició de la taula: " + i + " -> " + nodeAux + "\n\n";
         }
      }
      return aux;
   }
}
```

NodeHashGeneric

```
public class NodeHashGeneric<T, K>{
	private T valor;
	private K key;
	private NodeHashGeneric<T, K> seguent;

	/**
	 * Constructor
	 */
	public NodeHashGeneric (){
		valor = null;
		key = null;
		seguent = null;
	}

	/**
	 * Constructor
	 * @param data - valor
	 * @param key - "posicio de la taula"
	 */
	public NodeHashGeneric (T data, K key){
		this.valor = data;
		this.key = key;
		seguent = null;
	}

	/**
	 * Getter valor
	 * @return valor
	 */
	public T getValor() {
		return valor;
	}

	/**
	 * Setter valor
	 * @param data - valor
	 */
	public void setValor(T data) {
		this.valor = data;
	}

	/**
	 * Getter clau
	 * @return "posicio de la taula"
	 */
	public K getKey() {
```

```
		return key;
	}

	/**
	 * Setter clau
	 * @param key - "posicio de la taula"
	 */
	public void setKey(K key) {
		this.key = key;
	}

	/**
	 * Getter seguent
	 * @return seguent node
	 */
	public NodeHashGeneric<T, K> getSeguent() {
		return seguent;
	}

	/**
	 * Setter seguent
	 * @param seguent - seguent node
	 */
	public void setSeguent(NodeHashGeneric<T, K> seguent) {
		this.seguent = seguent;
	}

	@Override
	public String toString() {
		return "[valor = " + valor + ", key = " + key + "] seguent = " + seguent;
	}


}
```