

# Estructures de Dades i Algorismes

## Exercicis sobre arxius i *streams*

Dr. Enric Sesa i Nogueras



**TecnoCampus**  
Escola Superior  
Politécnica



## 1. Passar a majúscules (d'arxiu a "pantalla")

En el projecte que se us subministra, hi ha el programa, incomplet, `toUpper_Console`. El propòsit d'aquest programa és llegir el contingut d'un arxiu de text i mostrar-ne per pantalla el text original i el resultat de convertir-lo a "tot majúscules", línia per línia. La vostra feina consisteix en completar-lo.

```
import java.io.*;
import Keyboard.*;

public class ToUpper_Console {

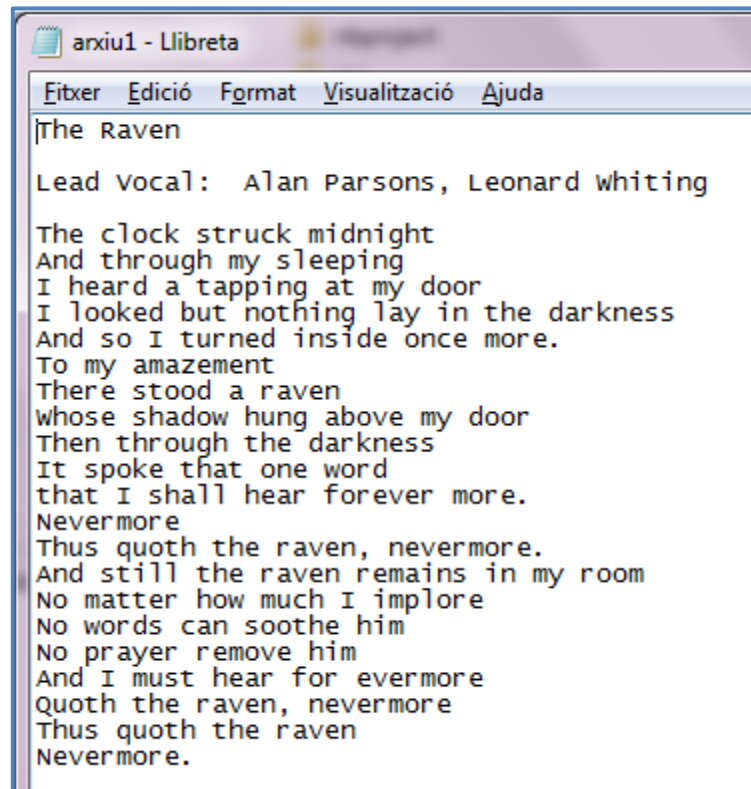
    public static void main (String [] args) {
        String filename;
        File file;
        BufferedReader entrada = null;
        String linia;

        System.out.print("Nom de l'arxiu (sense extensio): ");
        filename = Keyboard.readString();
        filename = filename+".txt";

        // vincular i obrir
        file = new File(filename);
        try {
            /* COMPLETAR */
        }
        catch (FileNotFoundException e) {
            /* COMPLETAR */
        }

        // llegir i mostrar en majúscules
        System.out.println();
        System.out.println("-----");
        try {
            /* COMPLETAR */
        }
        catch (IOException e) {
            /* COMPLETAR */
        }
        System.out.println("-----");
    }
}
```

A la carpeta del projecte, hi teniu l'arxiu `arxiu1.txt`, que podeu usar per tal de provar el funcionament del vostre desenvolupament.



El resultat de l'execució amb aquest arxiu hauria de ser quelcom com ara:

```
Nom de l'arxiu (sense extensio): arxiu1

-----
The Raven ==> THE RAVEN
==>
Lead Vocal: Alan Parsons, Leonard Whiting ==> LEAD VOCAL: ALAN PARSONS, LEONARD WHITING
==>
The clock struck midnight ==> THE CLOCK STRUCK MIDNIGHT
And through my sleeping ==> AND THROUGH MY SLEEPING
I heard a tapping at my door ==> I HEARD A TAPPING AT MY DOOR
I looked but nothing lay in the darkness ==> I LOOKED BUT NOTHING LAY IN THE DARKNESS
And so I turned inside once more. ==> AND SO I TURNED INSIDE ONCE MORE.
To my amazement ==> TO MY AMAZEMENT
There stood a raven ==> THERE STOOD A RAVEN
Whose shadow hung above my door ==> WHOSE SHADOW HUNG ABOVE MY DOOR
Then through the darkness ==> THEN THROUGH THE DARKNESS
It spoke that one word ==> IT SPOKE THAT ONE WORD
that I shall hear forever more. ==> THAT I SHALL HEAR FOREVER MORE.
Nevermore ==> NEVERMORE
Thus quoth the raven, nevermore. ==> THUS QUOTH THE RAVEN, NEVERMORE.
And still the raven remains in my room ==> AND STILL THE RAVEN REMAINS IN MY ROOM
No matter how much I implore ==> NO MATTER HOW MUCH I IMPLYRE
No words can soothe him ==> NO WORDS CAN SOOTHE HIM
No prayer remove him ==> NO PRAYER REMOVE HIM
And I must hear for evermore ==> AND I MUST HEAR FOR EVERMORE
Quoth the raven, nevermore ==> QUOTH THE RAVEN, NEVERMORE
Thus quoth the raven ==> THUS QUOTH THE RAVEN
Nevermore. ==> NEVERMORE.
-----
```

## 2. Passar a majúscules (d'arxiu a arxiu)

En el mateix projecte de l'exercici anterior, hi trobareu el programa incomplet `toUpper_OutputFile`, el propòsit del qual és llegir un arxiu de text i convertir-ne el contingut a "tot majúscules" deixant el resultat en un altre arxiu.

```
public class ToUpper_OutputFile {

    public static void main (String [] args) {
        String filename;
        File inputFile, outputFile; // un fitxer per l'entrada i un per la sortida
        BufferedReader entrada = null;
        BufferedWriter sortida = null;
        String linia;

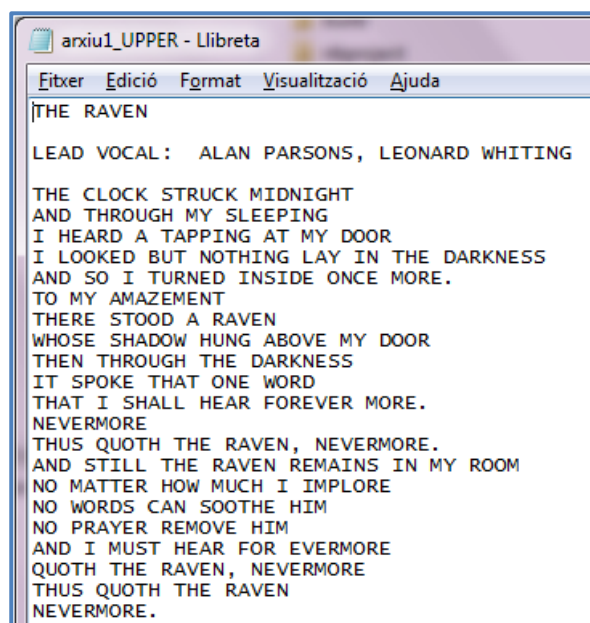
        System.out.print("Nom de l'arxiu (sense extensio): ");
        filename = Keyboard.readString();

        // vincular i obrir tant l'entrada com la sortida
        inputFile = new File(filename+".txt");
        outputFile = new File(filename+"_UPPER.txt");

        /* COMPLETAR */

        // llegir en un i escriure en l'altre
        try {
            /* COMPLETAR */
            sortida.close();
        }
        catch (IOException e) {
            /* COMPLETAR */
        }
        System.out.println(">>>Arxiu processat. Resultat a "+outputFile.getName());
    }
}
```

Per exemple, si l'arxiu a transformar fos el mateix `arxiu1` de l'exercici anterior, llavors el resultat hauria de ser un arxiu amb el següent contingut



### 3. Longitud mitjana de les línies

Disposeu del programa incomplet `InfoLínies` que té per propòsit llegir un arxiu de text per mostrar-ne les línies junt amb la seva mida i calcular-ne la mida mitjana (en caràcters).

Podeu provar el funcionament del vostre desenvolupament processant els arxius `arxiu2.txt`, `arxiuLa.txt` i `buit.txt`. Els resultats esperats són, respectivament:

```
Console
<terminated> InfoLínies [Java Application] C:\Program Files\Java\jre1.8.0_231\bin\javaw.exe (6 jun. 2020 22:27:08)

Linia 1: 1 mida => 1
Linia 2: 12 mida => 2
Linia 3: 123 mida => 3
Linia 4: 1234 mida => 4
Linia 5: 12345 mida => 5
Linia 6: 123456 mida => 6
Linia 7: 1234567 mida => 7
Linia 8: 12345678 mida => 8
Linia 9: 123456789 mida => 9

L'arxiu arxiu2.txt té 9 línies
La mida mitjana de les línies és 5.0 caràcters
```

```
Console
<terminated> InfoLínies [Java Application] C:\Program Files\Java\jre1.8.0_231\bin\javaw.exe (6 jun. 2020 22:33:29)

Linia 1: La Maria, la Montse i les altres noies del barri mida => 48
Linia 2: van decidir que un viatge a Etiopia havia de passar mida => 51
Linia 3: per Lalibela. Pero el Llatzer va pensar que era mida => 47
Linia 4: una mala idea. mida => 15

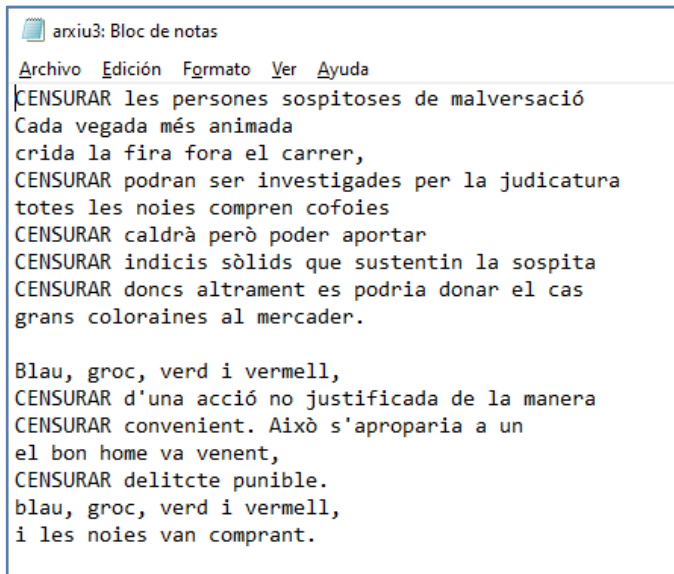
L'arxiu arxiuLA.txt té 4 línies
La mida mitjana de les línies és 40.25 caràcters
```

```
Console
<terminated> InfoLínies [Java Application] C:\Program Files\Java\jre1.8.0_231\bin\javaw.exe (6 jun. 2020 22:28:26)

L'arxiu buit.txt té 0 línies
Amb zero línies no és pot calcular cap mitjana
```

#### 4. “Censurar” un arxiu de text

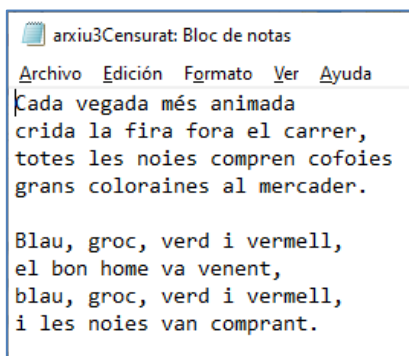
Disposeu del programa incomplet `Censurar` que té per propòsit llegir el contingut d'un arxiu de text per generar-ne un altre (també de text) que no contingui les línies del primer que comencin amb la cadena “CENSURAR”. Per exemple, el resultat de processar aquest arxiu (`arxiu3.txt`):



```
arxiu3: Bloc de notas
Archivo Edición Formato Ver Ayuda
CENSURAR les persones sospitoses de malversació
Cada vegada més animada
cria la fira fora el carrer,
CENSURAR podran ser investigades per la judicatura
totes les noies compren cofoies
CENSURAR caldrà però poder aportar
CENSURAR indicis sòlids que sustentin la sospita
CENSURAR doncs altrament es podria donar el cas
grans coloraines al mercader.

Blau, groc, verd i vermell,
CENSURAR d'una acció no justificada de la manera
CENSURAR convenient. Això s'aproparia a un
el bon home va venent,
CENSURAR delitcte punible.
blau, groc, verd i vermell,
i les noies van comprant.
```

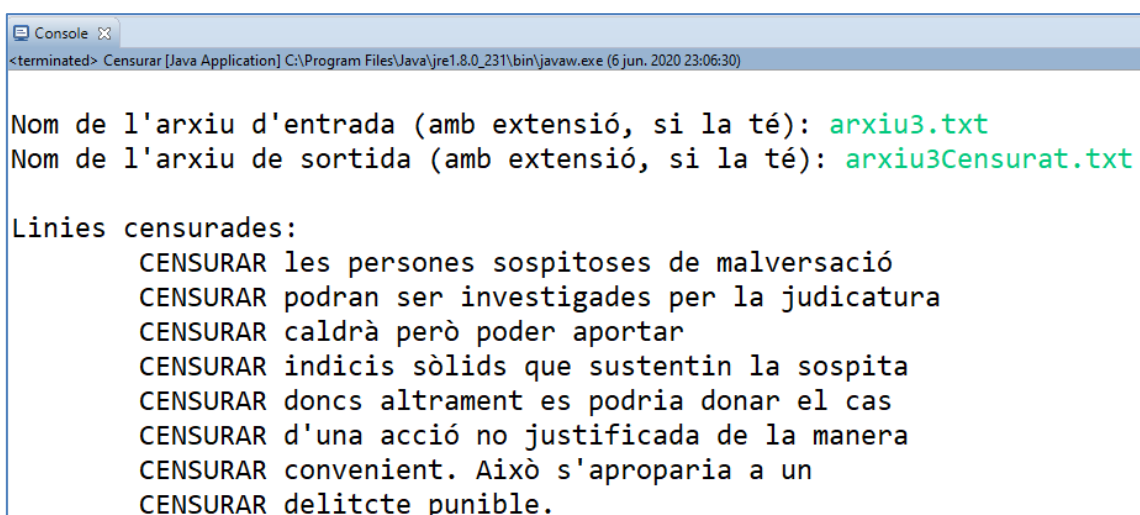
Seria aquest altre arxiu:



```
arxiu3Censurat: Bloc de notas
Archivo Edición Formato Ver Ayuda
Cada vegada més animada
cria la fira fora el carrer,
totes les noies compren cofoies
grans coloraines al mercader.

Blau, groc, verd i vermell,
el bon home va venent,
blau, groc, verd i vermell,
i les noies van comprant.
```

El programa també mostra, per pantalla, les línies censurades



```
Console
<terminated> Censurar [Java Application] C:\Program Files\Java\jre1.8.0_231\bin\javaw.exe (6 jun. 2020 23:06:30)

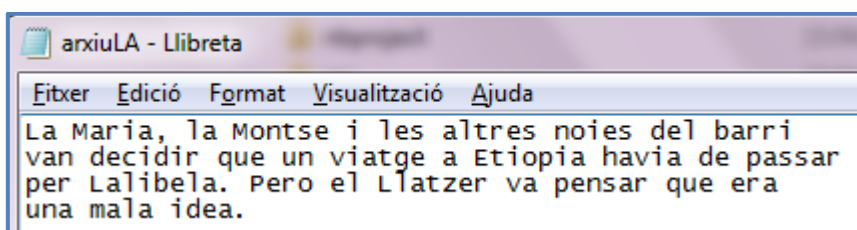
Nom de l'arxiu d'entrada (amb extensió, si la té): arxiu3.txt
Nom de l'arxiu de sortida (amb extensió, si la té): arxiu3Censurat.txt

Linies censurades:
    CENSURAR les persones sospitoses de malversació
    CENSURAR podran ser investigades per la judicatura
    CENSURAR caldrà però poder aportar
    CENSURAR indicis sòlids que sustentin la sospita
    CENSURAR doncs altrament es podria donar el cas
    CENSURAR d'una acció no justificada de la manera
    CENSURAR convenient. Això s'aproparia a un
    CENSURAR delitcte punible.
```

## 5. Comptar subcadena LA

Disposau del programa incomplet `Comptar_LA` que té per propòsit comptar el número de vegades que la cadena LA apareix en un arxiu de text. Aquesta cadena pot aparèixer en qualsevol posició i en qualsevol de les seves variants (La, LA, la, ...)

Podeu provar el funcionament del vostre desenvolupament processant l'arxiu `arxiuLA.txt` que té el següent contingut:



Aquest arxiu conté 6 aparicions de la cadena LA, per la qual cosa s'espera un resultat com:

```
nom de l'arxiu TXT (sense extensió): arxiuLA
la cadena LA apareix 6 vegades
```

La dificultat de resoldre aquest exercici no recau tant en el processament de l'arxiu en si mateix com en el poder detectar les diferents aparicions de LA dins d'una cadena de text. Per aquesta raó se us suggereix que completeu un procediment especialitzat en resoldre aquesta qüestió:

```
private static int processarLinia (String linia) {
    // aquest procediment compta el número de vegades que apareix
    // la cadena LA, en qualsevol de les seves variants, a la línia de
    // text donada com a paràmetre
    int pos, resultat;

    linia = linia.toUpperCase();

    resultat = 0;
    pos = linia.indexOf("LA");

    /* COMPLETAR */
    // us pot ser útil una iteració en què es fa ús del mètode
    // indexOf en la seva versió de dos paràmetres...

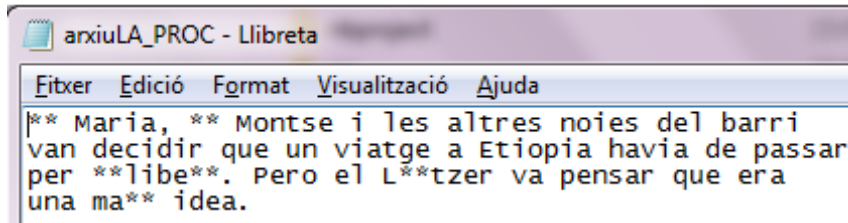
    return resultat;
}
```



## 6. Substituir les cadenes LA

Disposeu d'una versió incompleta del programa `Susbstituir_LA`. El propòsit d'aquest programa és processar un arxiu de text, generant-ne una nova versió en què cada aparició de la cadena LA ha estat substituïda pels caràcters `**`.

Per exemple, quan processa l'arxiu `arxiuLA.txt` (el mateix de l'exercici anterior) el resultat hauria de ser un arxiu amb un contingut com el següent:



La dificultat d'aquest exercici recau en fer la substitució de les cadenes LA per cadenes `**`. De nou se us suggereix que completeu un procediment especialitzat en aquesta qüestió:

```
private static String processarLinia (String linia) {  
    // genera una versió de la línia en què LA ha estat substituït per **  
  
    int pos;  
    String versioMaj;  
    StringBuffer sb;  
  
    versioMaj = linia.toUpperCase();  
    sb = new StringBuffer(linia);  
  
    pos = versioMaj.indexOf("LA");  
  
    /* COMPLETAR */  
  
    return sb.toString();  
}
```

Els objectes de la classe `StringBuffer` són en molts sentits equivalents als objectes de la classe `String` però es diferencien d'aquests darrers perquè ofereixen la possibilitat de fer canvis en la cadena de caràcters. La classe està definida a `java.lang`:

java.lang

**Class StringBuffer**

[java.lang.Object](#)  
└─ [java.lang.StringBuffer](#)

All Implemented Interfaces:  
[Serializable](#), [Appendable](#), [CharSequence](#)

```
public final class StringBuffer  
extends Object  
implements Serializable, CharSequence
```

A thread-safe, mutable sequence of characters. A string buffer is like a [String](#), but can be modified. At any point in time it contains some particular sequence of characters, but the length and content of the sequence can be changed through certain method calls.

## 7. Carregar i descarregar un diccionari de sinònims

Se us subministra un paquet que conté la implementació del ja conegut diccionari de sinònims. També se us subministra un punt d'entrada en execució anomenat `PexProva`. Aquest punt d'entrada en execució és igual al que ja hi havia en la versió anterior d'aquest problema, llevat que ara incorpora la possibilitat de “carregar” el diccionari des d'un arxiu extern i “descarregar-lo” a un arxiu extern

```
public static void main (String [] args) {

    int opcio; // opció del menú triada

    // crear el diccionari
    elMeuDiccionari = new DiccioAmbMap();

    // donem l'opció de carregar el diccionari
    carregarDiccionari(elMeuDiccionari);

    // iterar mentre l'opció seleccionada no sigui la d'abandonar el programa
    opcio = menu();
    while (opcio!=FINAL) {
        // processar l'opció seleccionada
        switch (opcio) {
            case BUSCAR: buscarSinonims(); break;
            case AFEGER: afegirSinonim(); break;
            case AFEGER_CJT: afegirConjunt(); break;
            case LLISTAR: llistarParaules(); break;
        }
        opcio = menu();
    }

    // donem l'opció de descarregar el diccionari
    descarregarDiccionari(elMeuDiccionari);
} // final de main
```

Els mètodes `carregarDiccionari` i `descarregarDiccionari` no implementen directament la càrrega o la descarrega sinó que invoquen un de tres mètodes especialitzats en càrrega i descàrrega, respectivament

```

private static void carregarDiccionari(DiccionariSinonims dic) {
    String filename;
    char resposta;

    System.out.println();
    do {
        System.out.print("Vols carregar el diccionari des d'un arxiu(S/N)? ");
        resposta = Keyboard.readString().toUpperCase().charAt(0);
    }
    while(resposta!='N' && resposta!='S');

    if (resposta=='S') {
        System.out.print("nom arxiu que conté el diccionari: ");
        filename = Keyboard.readString();

        // triar format de càrrega
        switch(modeCarrega) {
            case TEXT: carregarText (filename, dic); break;
            case BIN_NUM: carregarBinari_num (filename, dic);break;
            case BIN_MAR: carregarBinari_marca (filename, dic); break;
        }
    }
}

```

```

private static void descarregarDiccionari(DiccionariSinonims dic) {
    String filename;
    char resposta;

    System.out.println();
    do {
        System.out.print("Vols descarregar el diccionari en d'un arxiu(S/N)? ");
        resposta = Keyboard.readString().toUpperCase().charAt(0);
    }
    while(resposta!='N' && resposta!='S');

    if (resposta=='S') {
        System.out.print("nom de l'arxiu per desar diccionari: ");
        filename = Keyboard.readString();

        // triar format de descarrega
        switch(modeDescarrega) {
            case TEXT: descarregarText (filename, dic); break;
            case BIN_NUM: descarregarBinari_num (filename, dic);break;
            case BIN_MAR: descarregarBinari_marca (filename, dic); break;
        }
    }
}

```

#### A. Càrrega en forma TEXT

El mètode `descarregarText`, ja està implementat: descarrega el contingut del diccionari en un arxiu de format text. En canvi el mètode `carregarText` no ho està. La vostra feina en aquest cas consisteix en implementar aquest darrera mètode. Observeu el codi del mètode `descarregarText`:

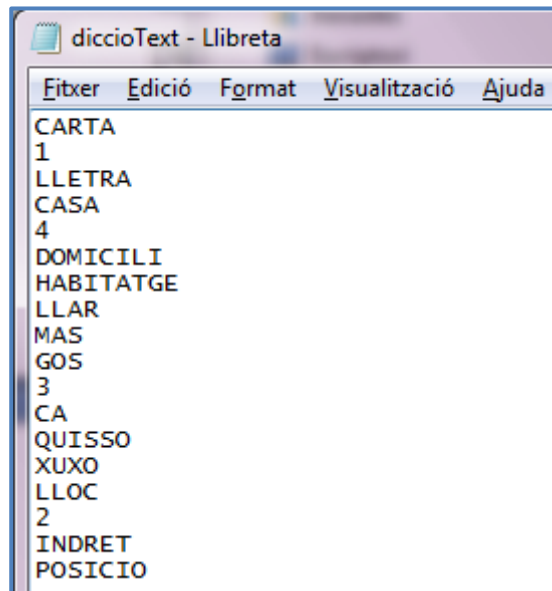
```
private static void descarregarText(String filename,
                                    DiccionariSinonims dic) {
    BufferedWriter out;
    String [] paraules;
    String [] sinonims;

    try {
        out = new BufferedWriter(new FileWriter(filename));

        paraules = dic.paraulesConegudes();
        for (String paraula : paraules) {
            // guardar la paraula
            out.write(paraula); out.newLine();
            // recuperar els seus sinonims
            sinonims = dic.recuperar(paraula);
            // escriure el número de sinònims i anar-los guardant
            out.write(String.valueOf(sinonims.length)); out.newLine();
            for (String sinonim : sinonims) {
                out.write(sinonim); out.newLine();
            }
        }
        // tancar stream
        out.close();
    }
    catch(IOException e) {
        System.out.println(e);
        System.exit(0);
    }
}
```

Observeu que aquest mètode guarda el diccionari fent, per a cada paraula, el següent: primer escriu la paraula, després escriu un número enter que és la quantitat de sinònims d'aquella paraula i després escriu un a un els sinònims d'aquella paraula.

A continuació teniu un arxiu generat per aquest mètode:



Aquest mateix arxiu, anomenat `diccioText.txt` el trobareu a la carpeta del projecte. Podeu utilitzar-lo per a provar el vostre mètode de càrrega. També podeu crear, amb un editor de texts senzills, altres arxius per a carregar diccionaris de sinònims.

## B. Descàrrega en format BINARI (amb MARCA)

En aquest cas, el que se us subministra és el mètode de càrrega anomenat `carregarBinari_marca`. Aquest mètode carrega un diccionari a partir d'un arxiu binari que estructurat de la següent manera: cada paraula està escrita en format UTF (vegeu el codi) i immediatament després de cada paraula s'hi troben tots els seus sinònims, també en format UTF. La seqüència de sinònims de cada paraula s'acaba amb un sinònim fictici (que anomenarem `marca`) que és la cadena (de nou en format UTF) formada per tres asteriscs: `***`

```
private static void carregarBinari_marca(String filename,
                                         DiccionariSinonims dic) {
    DataInputStream in;
    String paraula;
    String sinonim;
    String marca = "***";

    try {
        in = new DataInputStream(
            new BufferedInputStream(
                new FileInputStream(filename)));

        try {
            while (true) {
                paraula = in.readUTF();
                // llegir el primer sinonim i iterar
                sinonim = in.readUTF();
                while (!sinonim.equals(marca)) {
                    dic.afegir(paraula, sinonim);
                    sinonim = in.readUTF();
                }
            }
        }
        catch (EOFException e) {
            // arribats a aquest punt podem tancar el canal
            in.close();
        }
    }
    catch (IOException e) {
        System.out.println(e);
        System.exit(0);
    }
}
```

Ara la vostra feina consisteix en codificar el corresponent mètode de descàrrega, anomenat `descarregarBinari_marca`. Si ho feu correctament, els arxius descarregats amb aquest mètode podran ser carregats amb `carregarBinari_marca`.

### C. Càrrega en format BINARI (amb número de sinònims)

En aquesta tercera i darrera part, se us subministra, igual que en la primera, un mètode de descarrega (anomenat `descarregarBinari_num`) i la vostra feina consisteix en codificar el corresponent mètode de càrrega

```
private static void descarregarBinari_num(String filename,
                                           DiccionariSinonims dic) {
    DataOutputStream out;
    String [] paraules;
    String [] sinonims;
    try {
        out = new DataOutputStream(new BufferedOutputStream
                                   (new FileOutputStream(filename)));

        paraules = dic.paraulesConegudes();
        for (String paraula : paraules) {
            // guardar la paraula
            out.writeUTF(paraula);
            // recuperar els seus sinonims
            sinonims = dic.recuperar(paraula);
            // escriure el número de sinònims que hi ha
            out.writeInt(sinonims.length);
            // anar escrivint els sinonims
            for (String sinonim : sinonims) {
                out.writeUTF(sinonim);
            }
        }

        // tancar stream
        out.close();
    }
    catch(IOException e) {
        System.out.println(e);
        System.exit(0);
    }
}
```

Observeu que aquesta forma de descàrrega s'assembla molt a la utilitzada en el primer apartat. Ara però, la quantitat de sinònims és escrita com un valor int i no pas com una cadena de caràcters que representa un número int.

Per tal que pugueu fer alguna petita prova d'aquest apartat, trobareu a la carpeta del projecte l'arxiu `diccioBinNum.dat` que està enregistrat el format de `descarregarBinari_num`.

## 8. Save & Restore (A)

Disposau de les classes Part, Producte i d'un procediment anomenat saveONE que enregistra en un arxiu una col·lecció de productes.

```
public class Part {  
  
    public static final String MARCA = "CSxxx77DHRPE=EXX";  
  
    private String id, desc;  
  
    public Part (String id, String desc) {  
        this.id = id;  
        this.desc = desc;  
    }  
  
    public String getId() {return this.id;}  
    public String getDesc() {return this.desc;}  
}
```

```
public class Producte {  
  
    private String nom;  
    private double preu;  
    private Set<Part> parts;  
  
    public Producte (String nom, double preu, Set<Part> parts) {  
        this.nom = nom;  
        this.preu = preu;  
        this.parts = parts;  
    }  
  
    public String getNom() {return this.nom;}  
    public double getPreu() {return this.preu;}  
    public Set<Part> getParts() {return this.parts;}  
}
```

```
public static void saveONE (Collection<Producte> col, File file)  
    throws IOException {  
    // desa el contingut de la col·lecció donada en el primer paràmetre  
    // a l'arxiu especificat pel segon paràmetre  
  
    BufferedWriter buw;  
  
    buw = new BufferedWriter(new FileWriter(file));  
  
    for(Producte prod : col) {  
        buw.write(prod.getNom()); buw.newLine(); // desar nom  
        buw.write(Double.toString(prod.getPreu())); buw.newLine(); //desar preu  
        for (Part part : prod.getParts()){ // iterar sobre les parts  
            buw.write(part.getId()); buw.newLine(); // desar id de la part  
            buw.write(part.getDesc()); buw.newLine(); //desar desc. de la part  
        }  
        buw.write(Part.MARCA); buw.newLine(); // desar marca de fi de seq. de parts  
    }  
    buw.close();  
}
```



Completeu la codificació del procediment `restoreONE` que, a partir del contingut d'un arxiu generat pel procediment anterior, genera una col·lecció de productes.

```
public static Collection<Producte> restoreONE (File file)
    throws IOException {
    // recupera de l'arxiu especificat el contingut d'una col·lecció que
    // va ser desada amb el mètode save
    Collection<Producte> resultat = new ArrayList();
    Set<Part> parts;

    /* COMPLETAR */

    return resultat;
}
```

Per a provar el bon funcionament del vostre procediment `restore`, podeu crear-vos un programa que creï una col·lecció de productes, l'enregistri amb el procediment `save` i la recuperi amb `restore`.

## 9. Save & Restore (B)

Disposeu de les classes Part, Producte, les mateixes que en l'exercici precedent i dels procediments saveTWO i saveTHREE que enregistren en arxius una col·lecció de productes.

```
public static void saveTWO(Collection<Producte> col, File file) throws IOException
{
    DataOutputStream dos = new DataOutputStream(
        new BufferedOutputStream (new FileOutputStream (file)));

    for (Producte prod : col) {
        dos.writeUTF(prod.getNom());
        dos.writeDouble(prod.getPreu());
        dos.writeInt(prod.getParts().size());
        for (Part pa : prod.getParts()) {
            dos.writeUTF(pa.getId());
            dos.writeUTF(pa.getDesc());
        }
    }
    dos.writeUTF("END OF COLLECTION");
    dos.close();
}
```

```
public static void saveTHREE(Collection<Producte> col, File file)
                                throws IOException {

    DataOutputStream dos = new DataOutputStream(
        new BufferedOutputStream (new FileOutputStream (file)));

    for (Producte prod : col) {
        for (Part pa : prod.getParts()) {
            dos.writeUTF(pa.getId());
            dos.writeUTF(pa.getDesc());
        }
        dos.writeUTF(Part.MARCA);
        dos.writeUTF(prod.getNom());
        dos.writeDouble(prod.getPreu());
    }

    dos.close();
}
```

Escriviu els corresponents mètodes de recuperació

```
public static Collection<Producte> restoreTWO (File file)
                                throws IOException {

    /* COMPLETAR */
}
```

```
public static Collection<Producte> restoreTHREE (File file)
                                throws IOException {

    /* COMPLETAR */
}
```