DOC5: Programació estructurada - Accés a fitxers simples. BINARIS, TXT i CSV

Índex:

1- Introducció.

2- Accés a dades binaris.

3- Accés a dades .txt.

4- Accés a dades .csv

5- Treballar amb matrius .txt i .csv

6- Eliminar fitxers binaris, .txt i .csv

1- Introducció

L'accés a fitxers en programació permet llegir i escriure informació en dispositius d'emmagatzematge com el disc dur. En Java, es poden gestionar fitxers de text (.txt, .csv) o binari (.bin) mitjançant classes com FileReader, BufferedReader, FileWriter, DataOutputStream o RandomAccessFile. Aquest mecanisme és essencial per emmagatzemar dades de manera persistent i permet operar amb grans volums d'informació sense carregar-ho tot a la memòria.

L'accés a fitxers s'utilitza principalment per guardar dades localment en un disc o dispositiu d'emmagatzematge. Aquestes dades poden ser persistents (es mantenen després de tancar el programa) o temporals (es creen durant l'execució i s'eliminen després). Si es volen fitxers temporals, es poden crear en directoris del sistema com /tmp/ en Linux o C:\Users\Usuari\AppData\Local\Temp\ en Windows, utilitzant File.createTempFile()

IMPORTANT!: Si no es fica la ruta on volem guardar el fitxer es crea a la mateixa carpeta on hi ha el codi. Tot i això, als nostres programes locals és millor sempre ficar la ruta absoluta!!!

Els tipus de fitxers que treballarem en aquesta Resultat d'Aprenentatge són:

- Fitxers .bin (binari): Emmagatzemen dades en format binari, ocupant menys espai i sent més ràpids d'accedir, però no llegibles manualment. Són útils per dades estructurades i objectes.
- Fitxers .txt (text pla): Contenen informació en format llegible per humans, amb cada línia representant una dada o registre. Són fàcils d'editar però menys eficients en espai i velocitat.
- Fitxers .csv (valors separats per comes): Un tipus de .txt on les dades estan separades per comes, permetent estructurar informació en taules. És ideal per bases de dades i compatibilitat amb Excel.

Quan fer servir CSV (.csv)?

- Quan vols guardar dades estructurades en format de taula.
- Quan vols importar/exportar dades en Excel o Google Sheets.
- Quan cada línia representa un registre (per exemple, dades d'usuaris, productes, etc.).

Quan fer servir .txt?

- Quan vols guardar dades senzilles sense estructurar.
- Quan no necessites separadors i només guardes una dada per línia.

Quan fer servir .bin?

- Quan necessites més rapidesa i eficiència en l'emmagatzematge.
- Quan treballes amb objectes complexos i accés directe a posicions concretes.

Important!! Quan treballes amb fitxers (`.bin`, `.txt`, `.csv`), no es pot esborrar una línia concreta directament, ja que Java no permet modificar un fitxer en calent. Les opcions habituals són:

1. Esborrar tot el fitxer i tornar a escriure només les dades que vols conservar.
2. Llegir les dades en memòria, eliminar el que no necessites i sobreescriure el fitxer.

2- Treballar amb fitxers binaris.

1. Guardar dades a un fitxer binari

Per crear un fitxer binari en Java de manera senzilla, sense estructures com `try-catch`, podem utilitzar la classe `FileOutputStream`. Aquesta classe permet escriure bytes en un fitxer. Si volem escriure dades primitives com enters o dobles, podem combinar-la amb `DataOutputStream`.

Exemple pràctic per a guardar integers.

```
import java.io.FileOutputStream;
import java.io.DataOutputStream;

public class CrearFitxerBinari {
  public static void main(String[] args) {
    // Definim el nom del fitxer
    String nomFitxer = "dades.bin";
    // Creem un array d'enters per guardar al fitxer
    int[] numeros = {10, 20, 30, 40, 50};
    // Obrim el fitxer en mode d'escriptura binària
    FileOutputStream fitxer; //Inicialitzem la instància per a obrir fitxers.
    DataOutputStream dades; //Inicialitzem la instància per a treballar amb dades
    fitxer = new FileOutputStream(nomFitxer); //Creem i obrim el fitxer
```

```
            dades = new DataOutputStream(fitxer); // Obrim el fitxer per a treballar amb dades
        natives com int, String, Boolean etc..

            // Guardem cada número a l'arxiu en format binari. Utilitzem un for each
            for (int num : numeros) {
                dades.writeInt(num);
            }
            // Tanquem el flux de dades
            dades.close();
            fitxer.close();
        }
    }
```

Per a guardar altres tipus de dades utilitzarem:

Guardar dades String: ***dades.writeUTF("Hola món");***

Guardar dades Boolean: ***dades.writeBoolean(true);***

Guardar dades Char: ***dades.writeChar('A');***

SI VOLEM GUARDAR DADES SENSE SOBREESCRIURE EL CONTINGUT CADA COP QUE OBRIM EL FITXER, HEM D'UTILITZAR—---->

```
FileOutputStream fitxerpuntuaciones = new FileOutputStream(fitxer, true);
```

- ***`true`*** *(segon paràmetre): Aquest paràmetre indica que s'obrirà el fitxer en mode append (afegir al final del fitxer en lloc de sobreescriure'l).*
- *Si es posa* ***`true`****, les noves dades s'afegiran al final del fitxer sense esborrar el contingut anterior.*
- *Si es posa* ***`false`*** *(o no s'especifica, perquè el valor per defecte és `false`), el fitxer es sobreescriurà cada vegada que s'obri.*

2. Llegir dades d'un fitxer binari en Java

Per llegir les dades d'un fitxer binari, fem servir `FileInputStream` i `DataInputStream`. Això ens permet recuperar les dades en el mateix ordre en què les hem escrit.

```
public class LlegirFitxerBinari {
```

```
  public static void main(String[] args) {
      String nomFitxer = "dades.bin";
      FileInputStream fitxer;
      DataInputStream dades;

      fitxer = new FileInputStream(nomFitxer);
      dades = new DataInputStream(fitxer);
      // Llegim un enter del fitxer i el guardem en una variable
      int numero = dades.readInt();
      System.out.println("Número llegit: " + numero);
      // Tanquem el fitxer
      dades.close();
      fitxer.close();
  }
}
```

Això llegiria el primer `int` del fitxer `dades.bin`.

Llegir i guardar dades en un array

Si coneixem la llarga exacta del elements que tenim al fitxer.Podem inicialitzar un array, i llavors anar cargant-los:

```
int[] numeros = new int[5];
for (int i = 0; i < numeros.length; i++) {
   numeros[i] = dades.readInt();
}
```

Si no sabem quants números hi ha, podem llegir fins al final:

```
ArrayList<Integer> llistaNumeros = new ArrayList<>();

while (dades.available() > 0) { // Llegim mentre hi hagi dades disponibles
  llistaNumeros.add(dades.readInt());
}
```

Llegir línia a línia (amb `String`)

Si hem guardat `Strings` amb `writeUTF()`, els podem llegir amb `readUTF()`:

```
while (dades.available() > 0) {
   String text = dades.readUTF();
```

```
    System.out.println("Text llegit: " + text);
}
```

Llegir una línia concreta

Els fitxers binaris **no tenen línies**, sinó una seqüència de bytes. No podem anar directament a una "línia", però sí a una **posició concreta** amb `RandomAccessFile`:

```
import java.io.RandomAccessFile;
public class LlegirPosicio {
  public static void main(String[] args) throws Exception {
      RandomAccessFile fitxer = new RandomAccessFile("dades.bin", "r");
      fitxer.seek(4); // Ens situem al segon número (cada int ocupa 4 bytes)
      int segonNumero = fitxer.readInt();
      System.out.println("Segon número: " + segonNumero);
      fitxer.close();
  }
}
```

Això funciona perquè sabem que cada `int` ocupa **4 bytes**, així que per llegir el **tercer número**, farem `seek(8)`.

Quant ocupen els diferents tipus de dades en un fitxer binari en Java?

Quan escrivim dades en format binari amb `DataOutputStream`, cada tipus de dada ocupa una quantitat fixa de bytes:

Tipus de dada	Bytes ocupats	Mètodes d'escriptura/lectura
`int`	4 bytes	`writeInt(int) / readInt()`
`double`	8 bytes	`writeDouble(double) / readDouble()`
`float`	4 bytes	`writeFloat(float) / readFloat()`
`long`	8 bytes	`writeLong(long) / readLong()`
`short`	2 bytes	`writeShort(short) / readShort()`
`byte`	1 byte	`writeByte(byte) / readByte()`
`char`	2 bytes	`writeChar(char) / readChar()`
`boolean`	1 byte	`writeBoolean(boolean) / readBoolean()`
`String`	**Variable**	`writeUTF(String) / readUTF()`

6. Bones pràctiques en fitxers binaris

1. **Tanca sempre els fluxos** (`close()`) per evitar bloquejos de fitxers.
2. **Escriu i llegeix en el mateix ordre** per evitar errors en el format de dades.
3. **Fes servir `available()` per evitar llegir més del que hi ha**.
4. **Usa `RandomAccessFile` si necessites accedir a posicions concretes**.
5. **No barregis text i binari** en el mateix fitxer si no és necessari.
6. Sí estem treballant **sense objectes** i volem facilitar la lectura i escriptura, **és millor que cada fitxer guardi només un tipus de dada** (per exemple, un fitxer per `int`, un altre per `String`, etc.). Això permet llegir les dades seqüencialment sense preocupar de l'ordre o dels diferents tamanys dels tipus de dades.

3- Treballar amb fitxers .TXT

1. Escriure dades a un fitxer TXT.

Escriure un fitxer `.txt` amb un sol tipus de dada (només `int`, `String`, etc.) Si només guardem nombres en un fitxer `.txt`, és millor escriure cada dada en una línia.

Guardar només `int` en un `.txt`

```
import java.io.FileWriter;
import java.io.PrintWriter;
public class EscriureTxt {
   public static void main(String[] args) {
       FileWriter fitxer;
       PrintWriter dades;

       fitxer = new FileWriter("numeros.txt");
       dades = new PrintWriter(fitxer);
       // Guardem només enters (un per línia)
       dades.println(10);
       dades.println(20);
       dades.println(30);
       // Tanquem el fitxer
       dades.close();
```

```
        fitxer.close();
    }
}
```

2. Llegir el fitxer .txt i guardar-ho en un int[]

Per a llegir un fitxer txt és necessari recorrer linea a linea a un bucle i cargar-ho a un array.

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.util.ArrayList;
public class LlegirTxt {
  public static void main(String[] args) {
     FileReader fitxer;
     BufferedReader dades;
     fitxer = new FileReader("numeros.txt");
     dades = new BufferedReader(fitxer);
     // Llegim cada línia i la guardem en una llista dinàmica
     ArrayList<Integer> llista = new ArrayList<>();
     String linia;

     while ((linia = dades.readLine()) != null) {
        llista.add(Integer.parseInt(linia));
     }
     // Convertim l'ArrayList en un array normal
    int[] numeros = new int[llista.size()]; // Creem un array de la mateixa mida que l'ArrayList
         for (int i = 0; i < llista.size(); i++) {
                  numeros[i] = llista.get(i); // Assignem cada element de l'ArrayList a l'array
         }
     // Mostrem les dades
     for (int num : numeros) {
        System.out.println("Número llegit: " + num);
     }
     // Tanquem el fitxer
     dades.close();
     fitxer.close();
  }
}
```

3. Fitxer .txt amb diferents tipus de dades (int, String, boolean)

Si volem barrejar diferents tipus de dades, escrivim cada dada separada per un separador (per exemple, `,` o `;`).

Exemple: Guardar `int`, `String` i `boolean` junts

```
import java.io.FileWriter;
```

```java
import java.io.PrintWriter;
public class EscriureTxtMixt {
   public static void main(String[] args) {
       FileWriter fitxer;
       PrintWriter dades;
       fitxer = new FileWriter("dades.txt");
       dades = new PrintWriter(fitxer);
       // Format: edat,nom,estat
       dades.println("25,Joan,true");
       dades.println("30,Maria,false");
       dades.println("22,Pere,true");
       // Tanquem el fitxer
       dades.close();
       fitxer.close();
   }
}
```

4. Llegir un `.txt` amb diferents tipus de dades

Aquí llegim el fitxer i separem les dades amb split(","):

```java
import java.io.BufferedReader;
import java.io.FileReader;
public class LlegirTxtMixt {
  public static void main(String[] args) {
     FileReader fitxer;
     BufferedReader dades;
     fitxer = new FileReader("dades.txt");
     dades = new BufferedReader(fitxer);
     String linia;

     while ((linia = dades.readLine()) != null) {
        String[] parts = linia.split(","); // Separem per comes
        int edat = Integer.parseInt(parts[0]);
        String nom = parts[1];
        boolean estat = Boolean.parseBoolean(parts[2]);
        System.out.println("Nom: " + nom + ", Edat: " + edat + ", Actiu: " + estat);
     }
     // Tanquem el fitxer
     dades.close();
     fitxer.close();
  }
}
```

5. Llegir una línia concreta

Amb fitxers `.txt`, podem anar línia per línia, però no saltar directament a una línia concreta com en els fitxers binaris.
L'única manera és recórrer el fitxer fins a la línia que ens interessa.

Exemple: Llegir la línia número

```java
import java.io.BufferedReader;
import java.io.FileReader;
public class LlegirLinia {
  public static void main(String[] args) {
      FileReader fitxer;
      BufferedReader dades;
      fitxer = new FileReader("dades.txt");
      dades = new BufferedReader(fitxer);
      int numeroLinia = 2; // Volem la segona línia
      int liniaActual = 1;
      String liniaDesitjada = null;
      String linia;
      while ((linia = dades.readLine()) != null) {
         if (liniaActual == numeroLinia) {
            liniaDesitjada = linia;
            break; // Aturem la lectura quan trobem la línia
         }
         liniaActual++;
      }
      System.out.println("Línia " + numeroLinia + ": " + liniaDesitjada);
      // Tanquem el fitxer
      dades.close();
      fitxer.close();
  }
}
```

4- Treballar amb .CSV

1. Escriure un fitxer `.csv` amb un sol tipus de dada (`int`)

Si només guardem nombres en un CSV, els podem escriure separats per comes en una única línia o una línia per número.

```java
import java.io.FileWriter;
import java.io.PrintWriter;
public class EscriureCSV {
  public static void main(String[] args) {
     FileWriter fitxer;
     PrintWriter dades;
     fitxer = new FileWriter("numeros.csv");
     dades = new PrintWriter(fitxer);
     // Escriure en una sola línia separats per comes
     dades.println("10,20,30,40,50");
     // O escriure cada número en una línia (si vols format més ordenat)
     // dades.println(10);
     // dades.println(20);
     // dades.println(30);
     // Tanquem el fitxer
     dades.close();
     fitxer.close();
  }
```

```
}
```

O ho podem fer amb un bucle.

```
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
public class GuardarArrayCSV {
  public static void main(String[] args) {
     int[] numeros = {10, 20, 30, 40, 50}; // Array d'enters
     try {
        FileWriter fitxer = new FileWriter("numeros.csv");
        PrintWriter dades = new PrintWriter(fitxer);
        // Escriure l'array sencer en una sola línia separada per comes
        for (int i = 0; i < numeros.length; i++) {
           if (i > 0) dades.print(","); // Afegim coma entre números
           dades.print(numeros[i]);
        }
        // Tanquem el fitxer
        dades.close();
        fitxer.close();
        System.out.println("Array guardat correctament a numeros.csv");
     } catch (IOException e) {
        e.printStackTrace();
     }
  }
}
```

2. Llegir un `.csv` i guardar-ho en un `int[]`

El mateix que amb .txt, recorrem el fitxer linea a linea,

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.util.ArrayList;
public class LlegirCSV {
   public static void main(String[] args) {
      FileReader fitxer;
      BufferedReader dades;
      fitxer = new FileReader("numeros.csv");
      dades = new BufferedReader(fitxer);
      String linia = dades.readLine(); // Llegim la primera línia
      String[] parts = linia.split(","); // Separem els nombres per comes
      ArrayList<Integer> llista = new ArrayList<>();
      for (String part : parts) {
         llista.add(Integer.parseInt(part.trim())); // Convertim a enter
      }
      // Convertim a array estàtic
     //1. Creem un array de la mateixa mida que l'ArrayList
        int[] numeros = new int[llista.size()];
        //2. Omplim l'array amb els valors de l'ArrayList
        for (int i = 0; i < llista.size(); i++) {
```

```
            numeros[i] = llista.get(i);
            }

        // Mostrem els números
        for (int num : numeros) {
            System.out.println("Número llegit: " + num);
        }
        // Tanquem el fitxer
        dades.close();
        fitxer.close();
    }
}
```

3. Fitxer `.csv` amb diferents tipus de dades (`int`, `String`, `boolean`)

Amb CSV, cada línia representa un registre, i cada valor està separat per comes.

Exemple: Escriure `int`, `String` i `boolean` junts

```
import java.io.FileWriter;
import java.io.PrintWriter;
public class EscriureCSVMixt {
    public static void main(String[] args) {
        FileWriter fitxer;
        PrintWriter dades;
        fitxer = new FileWriter("dades.csv");
        dades = new PrintWriter(fitxer);
        // Primera línia com a capçalera (opcional)
        dades.println("Edat,Nom,Actiu");
        // Afegim dades (una persona per línia)
        dades.println("25,Joan,true");
        dades.println("30,Maria,false");
        dades.println("22,Pere,true");
        // Tanquem el fitxer
        dades.close();
        fitxer.close();
    }
}
```

4. Llegir un `.csv` amb diferents tipus de dades

Aquí llegim cada línia i separem les dades per comes (`split(",")`).

```
import java.io.BufferedReader;
import java.io.FileReader;
public class LlegirCSVMixt {
    public static void main(String[] args) {
```

```
        FileReader fitxer;
        BufferedReader dades;
        fitxer = new FileReader("dades.csv");
        dades = new BufferedReader(fitxer);
        String linia;
        dades.readLine(); // Saltem la primera línia (capçalera)
        while ((linia = dades.readLine()) != null) {
            String[] parts = linia.split(","); // Separem per comes
            int edat = Integer.parseInt(parts[0].trim());
            String nom = parts[1].trim();
//serveix per convertir un String (parts[2]) en un boolean (true o false).
            boolean actiu = Boolean.parseBoolean(parts[2].trim());
            System.out.println("Nom: " + nom + ", Edat: " + edat + ", Actiu: " + actiu);
        }
        // Tanquem el fitxer
        dades.close();
        fitxer.close();
    }
}
```

5. Llegir una línia concreta en un CSV

Igual que amb `.txt`, si volem llegir una línia específica d'un CSV, hem de recórrer el fitxer fins a trobar-la.

Exemple: Llegir la línia 2 d'un `.csv`

```
import java.io.BufferedReader;
import java.io.FileReader;
public class LlegirLiniaCSV {
    public static void main(String[] args) {
        FileReader fitxer;
        BufferedReader dades;
        fitxer = new FileReader("dades.csv");
        dades = new BufferedReader(fitxer);
        int numeroLinia = 2; // Volem la segona línia
        int liniaActual = 0;
        String liniaDesitjada = null;
        String linia;
        while ((linia = dades.readLine()) != null) {
            liniaActual++;
            if (liniaActual == numeroLinia) {
                liniaDesitjada = linia;
                break;
            }
        }
        System.out.println("Línia " + numeroLinia + ": " + liniaDesitjada);
        // Tanquem el fitxer
        dades.close();
        fitxer.close();
    }
}
```

5- Treballar amb matrius .txt i .csv

Quan treballem amb matrius en fitxers de text (.csv o .txt), el procés sempre segueix els mateixos passos fonamentals:

1. Crear i guardar la matriu en un fitxer

1. Definim una matriu (String[][], int[][], etc.).
2. Obrim un fitxer (.csv o .txt) en mode escriptura.
3. Recorrem la matriu fila per fila i guardem cada línia al fitxer.
4. Tanquem el fitxer.

2. Llegir la matriu des d'un fitxer

- Obrim el fitxer en mode lectura.
- Llegim cada línia i la guardem línea a línea.
- Si és un .csv, separem les dades amb split(",").
- Si és un .txt, podem utilitzar split("\t"), espais o altres separadors.
- Convertim les dades si és necessari (int, double, boolean...).
- Mostrem o utilitzem la matriu llegida.

CREAR UNA MATRIU D'ARRAYLIST.

Crearem un ArrayList que el seu tipus sigui un array dins. Per tant cada fila serà un array dins del Arraylist.

ArrayList<String[]> matriu = new ArrayList<>();

ArrayList<Integer[]> matriu = new ArrayList<>();

1. Guardar una matriu a un fitxer.

Aquest exemple desa una **matriu de productes** (`Nom`, `Quantitat`, `Disponible`) en un fitxer `.csv` i després el llegeix.

```
import java.io.FileWriter;
import java.io.PrintWriter;
public class EscriureMatriuCSV {
  public static void main(String[] args) throws Exception {
    String[][] productes = {
      {"Laptop", "10", "true"},
      {"Raton", "50", "true"},
      {"Teclat", "0", "false"},
      {"Monitor", "5", "true"}
    };
    FileWriter fitxer = new FileWriter("productes.csv");
    PrintWriter dades = new PrintWriter(fitxer);
    // Escriure la capçalera
    dades.println("Nom,Quantitat,Disponible");
    // Escriure la matriu
     for (String[] fila : productes) {
        dades.println(String.join(",", fila)); // Uneix elements separats per comes → No cal reccorerar dos bucles, ja que podem guardar directament separant les comes.
      }
    dades.close();
    fitxer.close();
    System.out.println("Matriu guardada correctament a productes.csv");
  }
}
```

2. Llegir la matriu des d'un .csv

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.util.ArrayList;
public class LlegirMatriuCSV {
```

```
    public static void main(String[] args) throws Exception {
        BufferedReader lector = new BufferedReader(new FileReader("productes.csv"));
        ArrayList<String[]> matriu = new ArrayList<>();

        String linia;
        lector.readLine(); // Saltem la capçalera
        while ((linia = lector.readLine()) != null) {
            String[] parts = linia.split(","); // Separem per comes
            matriu.add(parts);
        }
        lector.close();
        // Mostrem la matriu llegida
        System.out.println("Productes llegits:");
        for (String[] fila : matriu) {
            System.out.println("Nom: " + fila[0] + ", Quantitat: " + fila[1] + ", Disponible: " +
fila[2]);
        }
    }
}
```

3. Escriure la matriu en un `.txt`

Aquest exemple desa una matriu en un .txt i després el llegeix.

```
import java.io.FileWriter;
import java.io.PrintWriter;
public class EscriureMatriuTXT {
    public static void main(String[] args) throws Exception {
        String[][] horaris = {
            {"Matemàtiques", "08:00", "Aula 101"},
            {"Història", "09:30", "Aula 102"},
            {"Física", "11:00", "Aula 103"},
            {"Anglès", "12:30", "Aula 104"}
        };
        FileWriter fitxer = new FileWriter("horaris.txt");
        PrintWriter dades = new PrintWriter(fitxer);
```

```
        // Escriure cada fila separada per tabuladors
        for (String[] fila : horaris) {
            dades.println(fila[0] + "\t" + fila[1] + "\t" + fila[2]);
        }
        dades.close();
        fitxer.close();
        System.out.println("Matriu guardada correctament a horaris.txt");
    }
}
```

4. Llegir la matriu des d'un .txt

Llegim un TXT i anem guardan les dades a una matriu.

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.util.ArrayList;
public class LlegirMatriuTXT {
    public static void main(String[] args) throws Exception {
        BufferedReader lector = new BufferedReader(new FileReader("horaris.txt"));
        ArrayList<String[]> matriu = new ArrayList<>();

        String linia;
        while ((linia = lector.readLine()) != null) {
            String[] parts = linia.split("\t"); // Separem per tabuladors
            matriu.add(parts);
        }
        lector.close();
        // Mostrem la matriu llegida
        System.out.println("Horaris llegits:");
        for (String[] fila : matriu) {
            System.out.println("Assignatura: " + fila[0] + ", Hora: " + fila[1] + ", Aula: " + fila[2]);
        }
    }
}
```

6. Eliminar fitxers binaris, .txt i .csv

Per a eliminar un fitxer sobretot hem de tenir tancades les 2 connexions que hem obert, ja que sinó el fitxer està obert i mai podem eliminar un fitxer que ja es troba obert.
A part, hem de asignar la ruta i el fitxer a una variable tipus File, aquests tipus FILE es connecta amb el sistema operatiu per a llençar l'ordre de SO.

Guardem el fitxer a la tipus FILE.

```
File fitxer = new File("dades.bin");
```

Fem els tancaments de connexions al fitxers. ELS DOS!!!!

```
dades.close(); // —--> La connexió per treballar les dades.
fileInput.close(); //—---> La connexió al fitxer.
```

ELIMINEM EL FITXER AMB CONTROL D'ERRORS. TOT AIXÒ IGUALMENT ES FA DINS DEL TRY CATCH!!!!!

```
// Ara eliminem el fitxer
if (fitxer.delete()) {
    System.out.println("El fitxer s'ha eliminat correctament.");
} else {
    System.out.println("No s'ha pogut eliminar el fitxer.");
}
```