

## LECTURA-ESCRITURA DE FITXERS EN JAVA

Un programa pot tenir oberts un nombre indeterminat de fitxers (tot i que el sistema operatiu pot establir limitacions en aquest sentit). La **Lectura-Escriptura** de fitxers la interpretem com llegir d'un o més fitxers d'origen, tractar la informació recollida i desar el resultat en un o més fitxers de destí (usualment, llegirem d'un fitxer i escriurem en un altre). Això ho farem mentre mantenim oberts tots els fitxers involucrats.

El que hem de fer en aquests cassos és obrir primer el fitxer o fitxers d'origen. En cas de que l'obertura d'algun o alguns dels fitxers d'origen fracassi (fitxer inexistent per exemple), el programa no pot continuar, de forma que ja no provarem de crear (o obrir per afegir al final) el fitxer o els fitxers de destí. En aquest context, anirem obrint els successius fitxers en ordre, i dintre del **try** d'obertura d'un fitxer, obrirem el següent, i així successivament (sempre i quan tots els fitxers sigui necessaris, si no és així, ens podem plantejar altres estratègies). Així, si no fos possible obrir o crear algun dels fitxers, es provocarà una excepció en aquest punt, i el nostre programa no progressarà.

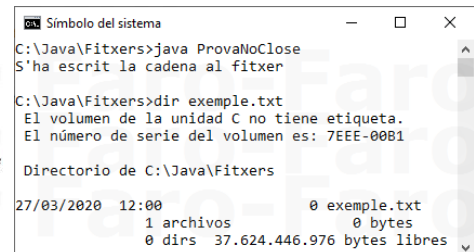
Aquest document, ens ha de servir també per repassar la lectura i escriptura de fitxers, i fins i tot per ampliar els coneixements que hem adquirit fins ara respecte a la gestió de fitxers de **java**, de forma que farem un repàs ràpid al que s'ha vist en documenta anteriors.

### Tancament dels fitxers

Respecte del tancament dels fitxers, amb el que sabem fins ara, podem seguir tres estratègies:

- No tancar el fitxer. Tot i que no es recomana, si només hem obert un fitxer per lectura, i el nostre programa ja acaba, podríem prescindir de tancar el fitxer, ja que el sistema operatiu ho farà per nosaltres. Això només és opcional en el cas de la lectura de fitxers, en el cas de l'escriptura, hem de saber que l'última informació escrita pels nostre programa, i que roman als buffers, no es porta al fitxer (**flush**) fins que aquest es tanca, de forma que la perdrem si no el tanquem dintre del nostre programa (el sistema operatiu no buida els buffers, només tanca el fitxer perquè aquest no quedi corrupte al disc). Constatau-lo amb el següent programa:

```
1 import java.io.*; // Importem les classes que necessitem
2 public class ProvaNoClose {
3     // Exemple de fitxer que escrivim i no tanquem
4     public static void main(String[] args) {
5         try {
6             String s = "Cadena que escrivim al fitxer de text";
7             PrintWriter pw = new PrintWriter("exemple.txt");
8             pw.println(s); // Escrivim la cadena al fitxer.
9             System.out.println("S'ha escrit la cadena al fitxer");
10        } catch (Exception e) {
11            System.out.println("ERROR: " + e.getMessage());
12        }
13    }
14 }
```



- Tancar el fitxer només en el cas d'execució exitosa (ho hem vist en alguns exemples). Si es produeix algun error en temps d'execució, dintre del bloc **catch** no tanquem el fitxer. Donat que la majoria de les operacions amb fitxers haurien de tenir èxit, deixem al sistema operatiu la responsabilitat de tancar els fitxers que haguem obert únicament en el cas de que es produeixi un error al seu tractament. Com que usualment, el programa acabarà llavors, i el sistema operatiu tanca tots els fitxers oberts per la nostra aplicació, optar per aquesta via no és tan greu com el cas anterior, ja que fins i tot en el cas de l'escriptura, si s'ha produït un error al tractament del fitxer, que els últims bytes escrits al buffer no quedin reflectits finalment al fitxer pot ser, en qualsevol cas, un problema menor.
- Assegurar-nos de tancar el fitxer (si aquest ha estat obert), fins i tot si s'ha produït una excepció. Aquesta és la forma més coherent i robusta de procedir, i hem vist com fer-ho en varis dels exemples dels documents anteriors (tanquem el fitxer al **finally**).

Apart d'aquestes tres formes de procedir, hi ha una quarta, la més recomanable de fet, i que es pot utilitzar a partir de **java 7**. Es comentarà a continuació.

## Autotancament dels fitxers

Com ja hem vist, de forma general, l'última informació escrita al buffer del fitxer no s'escriu en aquest fins que no executem una ordre **flush** (buidar el buffers) o tanquem el fitxer. Aquest problema esdevé més greu contra major és el buffer evidentment (per defecte, és d'un clúster, com ja es va comentar), i ens suggereix que el tancament ordenat dels fitxers hauria de ser una prioritat.

Donat que el tancament dels fitxers després de fer-los servir esdevé completament necessari, a partir de **java 7** es pot fer servir la instrucció **try-with-resources**. Un recurs (**resource**) és un objecte que necessita ser tancat després de fer-lo servir. Utilitzant aquesta instrucció, ens assegurem que aquests objectes (al nostre cas fitxers) seran tancats al final de la instrucció **try**, sense que nosaltres haguem de fer res més. Qualsevol objecte que implementi la interfície **java.lang.AutoCloseable**, entre ells els que implementen la interfície **java.io.Closeable** poden ser utilitzats d'aquesta forma.

A la pàgina: <https://docs.oracle.com/javase/8/docs/api/java/io/PrintWriter.html>

Veiem que la classe **PrintWriter** (i la resta que utilitzem) implementa aquestes interfícies:

### Class PrintWriter

```
java.lang.Object
java.io.Writer
java.io.PrintWriter
```

All Implemented Interfaces:

Closeable, Flushable, Appendable, AutoCloseable

L'exemple que hem vist abans s'escriuria d'aquesta forma:

```
1 import java.io.*; // Importem les classes que necessitem
2 public class ProvaNoClose {
3     // Exemple de fitxer que escrivim i no tanquem
4     public static void main(String[] args) {
5         try (PrintWriter pw = new PrintWriter("exemple.txt")) {
6             String s = "Cadena que escrivim al fitxer de text";
7             pw.println(s); // Escrivim la cadena al fitxer.
8             System.out.println("S'ha escrit la cadena al fitxer");
9         } catch (Exception e) {
10            System.out.println("ERROR: " + e.getMessage());
11        }
12    }
13 }
```

```
Símbolo del sistema
C:\Java\Fitxers>java ProvaNoClose
S'ha escrit la cadena al fitxer

C:\Java\Fitxers>dir exemple.txt
El volumen de la unidad C no tiene etiqueta.
El número de serie del volumen es: 7EEE-00B1

Directorio de C:\Java\Fitxers

27/03/2020  13:37                39 exemple.txt
               1 archivos                39 bytes
                  0 dirs 37.591.601.152 bytes libres
```

Funcionament:

- A continuació de la instrucció **try**, s'escriu el recurs entre parèntesis. Ja no és necessari escriure el bloc **finally** per tancar el fitxer, ni tan sols cal posar l'ordre **close()**.
- Si la instrucció **try** conté més d'un recurs, aquests aniran separats per **punt i coma**.

## Resum de les classes de Java per llegir fitxers de text

### Class Reader

```
java.lang.Object
java.io.Reader
```

All Implemented Interfaces:

Closeable, AutoCloseable, Readable

Direct Known Subclasses:

BufferedReader, CharArrayReader, FilterReader, InputStreamReader, PipedReader, StringReader

### Class InputStreamReader

```
java.lang.Object
java.io.Reader
java.io.InputStreamReader
```

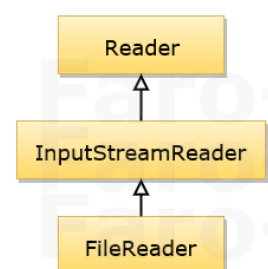
All Implemented Interfaces:

Closeable, AutoCloseable, Readable

Direct Known Subclasses:

FileReader

Com veiem a la dreta, la classe **Reader** (abstracta, la qual cosa vol dir que no podem instanciar-la (crear objectes d'aquesta classe)), és la classe pare de **InputStreamReader**, de la qual hereta **FileReader**, que és una de les classes que hem utilitzat per llegir fitxers de text. La classe **InputStreamReader** és un pont des dels **streams** (fluxos) de bytes als fluxos de caràcters. Aquesta classe, llegeix bytes i els codifica en caràcters a partir d'un joc de caràcters o **charset** (per defecte fa servir el de la plataforma, però podem especificar un altre).



## Lectura de fitxers de text en java

Tot i que es podria utilitzar la classe **InputStreamReader** per llegir fitxers directament, nosaltres no ho hem fet així:

```
Reader in = new InputStreamReader(new FileInputStream(file), encoding);
Reader in = new InputStreamReader(new FileInputStream(file)); // Platform's encoding
```

- **FileReader - Llegir fitxers de text, caràcter a caràcter i blocs de caràcters**

Nosaltres fem servir la classe **FileReader**, que hereta d'aquesta (on no podem especificar el joc de caràcters utilitzat) per llegir caràcters individuals i blocs de caràcters. Per incrementar el rendiment, podem fer servir **BufferedReader** amb el que canviem la grandària del buffer i ens permet també llegir línies del fitxer.

- **FileReader + BufferedReader - Llegir una o totes les línies d'un fitxer de text**

La classe **FileReader**, no permet directament llegir línies del fitxer, però si ho fem mitjançant **BufferedReader**, podrem fer-ho. Llegir línies del fitxer és més ineficient que llegir caràcters.

- **Scanner - Llegir int, double, boolean, etc... des de fitxers de text**

Fem servir la classe **Scanner** per llegir qualsevol cosa d'un fitxer de text. És una classe d'alt nivell que permet extreure la informació d'una cadena, d'un fitxer, de l'entrada estàndard, etc. Al contrari que a les anteriors, podem indicar també quina és la codificació dels caràcters que llegim. Com a contrapartida, el temps d'execució, serà considerablement superior al dels dos mètodes anteriors, i es poden provocar errors en temps d'execució si el format del fitxer no és l'esperat.

## Resum de les classes de Java per escriure fitxers de text

### Class Writer

java.lang.Object  
java.io.Writer

#### All Implemented Interfaces:

Closeable, Flushable, Appendable, AutoCloseable

#### Direct Known Subclasses:

BufferedWriter, CharArrayWriter, FilterWriter, OutputStreamWriter, PipedWriter, PrintWriter, StringWriter

### Class OutputStreamWriter

java.lang.Object  
java.io.Writer  
java.io.OutputStreamWriter

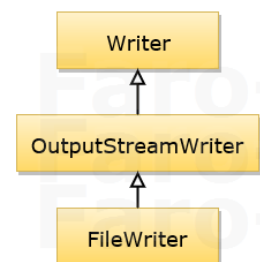
#### All Implemented Interfaces:

Closeable, Flushable, Appendable, AutoCloseable

#### Direct Known Subclasses:

FileWriter

Com veiem a la dreta, la classe **Writer** (abstracta, la qual cosa vol dir que no podem instanciar-la (crear objectes d'aquesta classe)), és la classe pare de **OutputStreamWriter**, de la qual hereta **FileWriter**, que és una de les classes que hem utilitzat per escriure fitxers de text. La classe **OutputStreamWriter** és un pont des dels **streams** (fluxos) de caràcters als fluxos de bytes. Aquesta classe, llegeix caràcters i els descodifica a bytes a partir d'un joc de caràcters o **charset** (per defecte fa servir el de la plataforma, però podem especificar un altre).



## Esriptura de fitxers de text en java

Tot i que es podria fer servir la classe **OutputStreamWriter** per escriure fitxers de text, nosaltres no ho hem fet així. Més informació a:

<https://docs.oracle.com/javase/8/docs/api/java/io/OutputStreamWriter.html>

- **FileWriter - Esriptura caràcter a caràcter (també permet escriure cadenes)**

Fa servir el sistema de codificació per defecte de la plataforma per escriure en un fitxer un únic caràcter o be un vector de caràcters, o una cadena. Per incrementar el rendiment, utilitzem aquesta classe amb la classe **BufferedWriter** amb la que canviem la grandària del buffer i ens permet també escriure caràcters o línies al fitxer.

- **FileWriter + BufferedWriter - Escriptura caràcter a caràcter o línia a línia**  
Igual que al cas anterior, els caràcters i línies s'escriuen amb el sistema de codificació per defecte de la plataforma, no el podem especificar.
- **PrintWriter – Escriure qualsevol tipus de dades en un fitxer de text**  
**PrintWriter**, igual que **PrintStream** és una classe d'alt nivell que permet escriure qualsevol tipus de dades en un fitxer de text (int, double, String, boolean), inclosos tots els tipus que permeten les classes anteriors, i a més, permet indicar el sistema de codificació o **charset** del fitxer. Fem servir **PrintWriter + BufferedWriter + FileWriter** per optimitzar la velocitat d'execució.

## Lectura i escriptura de fitxers binaris en java

La lectura/escriptura de fitxers binaris no necessita cap tipus de codificació com al cas dels fitxers de text. Existeixen dues classes abstractes (**InputStream** i **OutputStream**) que no podem instanciar, però de les quals hereten les que farem servir.



Sense entrar en detalls, direm que les classes principals que podem fer servir en java per llegir i escriure fitxers binaris són:

- **FileInputStream i FileOutputStream** - Per llegir i escriure un o més bytes en mode cru. Serien l'equivalent a les classes **InputStreamReader** i **OutputStreamWriter** però pels fitxers binaris.
- **DataInputStream i DataOutputStream** - Fan servir les classes anteriors per llegir o escriure tipus de dades primitius de java (int, long, double, boolean, etc).
- **ObjectInputStream i ObjectOutputStream** - Per llegir i escriure tipus de dades primitius de java (int, long, double, boolean, etc) i també objectes. Fan servir **FileInputStream** i **FileOutputStream** per convertir-los a bytes. Els objectes han de ser serialitzables. Farem servir aquestes classes per escriure i llegir els nostres vectors d'objectes en disc.
- **BufferedInputStream i BufferedOutputStream** - Aquestes classes, combinades amb les anteriors, les utilitzem per incrementar la velocitat d'accés als fitxers amb buffers. Son l'equivalent pels fitxers binaris de les classes **BufferedReader** i **BufferedWriter** que utilitzem als fitxers de text.

## Lectura i escriptura de fitxers d'accés directe

L'accés als fitxers mitjançant totes les classes anteriors tenen un denominador comú, tots els accessos són seqüencials, es a dir, la informació es llegeix i s'escriu al fitxer seqüencialment. La classe que proporciona **java** per a l'accés als fitxers de forma directa o aleatòria (a qualsevol posició del fitxer directament) és:

- **RandomAccessFile** - Permet obrir un fitxer per a lectura o per a lectura-escriptura, moure l'apuntador del fitxer a qualsevol posició dintre d'aquest i llegir i/o escriure bytes individuals, vectors de bytes i qualsevol dels tipus primitius del llenguatge a partir de la posició on es troba l'apuntador al fitxer. Evidentment, no podem inserir en un fitxer com faríem en un editor de text (desplaçant el contingut previ del fitxer), sempre que escrivim en una posició intermèdia del fitxer, reescriurem la informació present en aquella ubicació, i el contingut previ allà emmagatzemat es perdrà.

Si tenim un fitxer on hem emmagatzemat objectes del nostre programa, i volem accedir a un d'aquests objectes per posició (sense accedir a la resta dels objectes que es troben al davant al fitxer), haurem de saber la grandària dels objectes i posicionar l'apuntador al fitxer al començament de l'objecte desitjat.



## Proposta de programa de lectura-escriptura de fitxers

A continuació es proposa fer un programa que, a partir d'un fitxer que manté valors enters positius, un per línia, crearà altres tres fitxers:

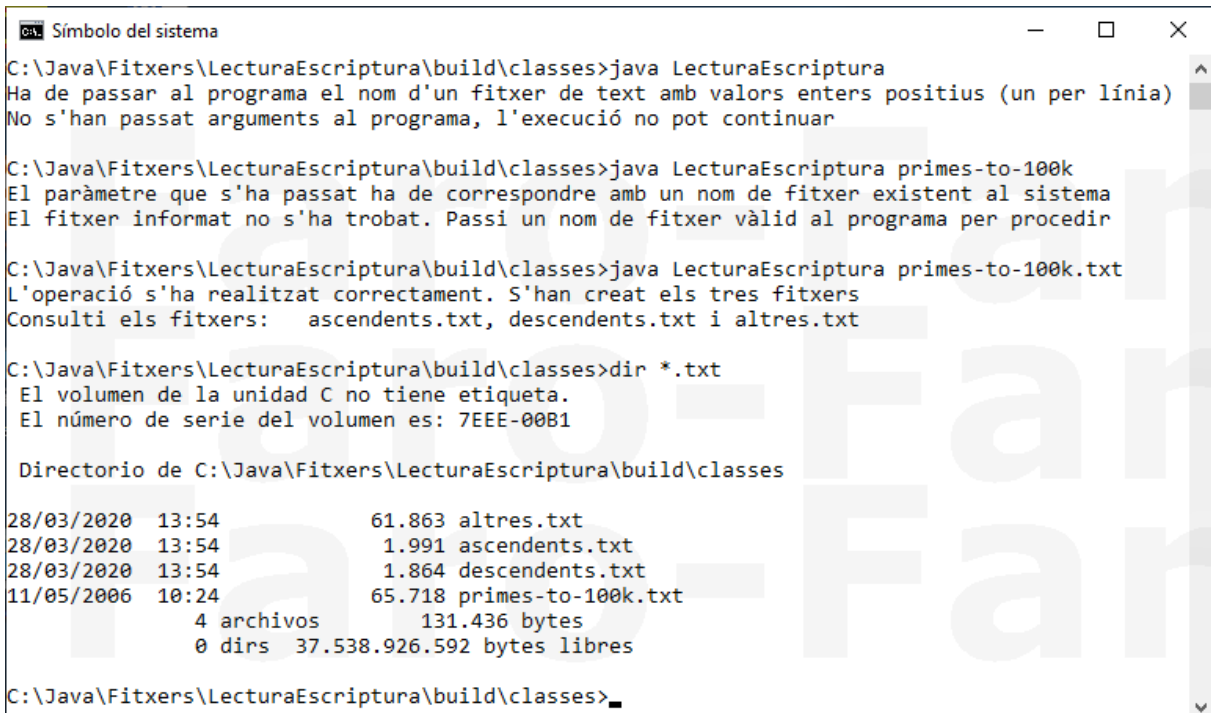
- **ascendents.txt** - Fitxer on escriurem els nombres del fitxer original (un per línia) les xifres dels quals son creixents, es a dir, que el codi ASCII de cadascuna de les xifres del nombre és superior o igual a la xifra anterior d'aquest.
- **descendents.txt** - Fitxer on escriurem els nombres del fitxer original (un per línia) les xifres dels quals son decreixents, es a dir, que el codi ASCII de cadascuna de les xifres del nombre és inferior o igual a la xifra anterior d'aquest.
- **altres.txt** - La resta de valors del fitxer original que no s'engloben a les dues categories anteriors.

Llegirem el fitxer original línia a línia amb la classe **BufferedReader + FileReader**, de la forma **linia = br.readLine()** (en llegir una línia del fitxer, el caràcter de finalització o retorn de carro no queda a la cadena llegida **linia**), i escriurem també els fitxers de destí línia a línia a partir de les classes **PrintWriter + BufferedWriter + FileWriter** amb la instrucció de **PrintWriter** per escriure línies al fitxer: **fitxer.println(linia)**. Podríem llegir números en lloc de cadenes del fitxer, però llavors els hauríem de convertir a cadenes per saber si corresponen a una de les categories comentades.

El nom del fitxer original, que manté els valors enters positius que estem analitzant (un per línia), el passarem al programa com a paràmetre. En cas de no passar paràmetres al programa o que el paràmetre passat no es correspongui amb un fitxer existent, el programa haurà d'indicar l'error corresponent i sortir al sistema operatiu.

Els programa no comprovarà l'existència o no dels fitxers de destí, de forma que si aquests existeixen seran esborrats sense cap indicació al respecte per part del programa. Qualsevol error haurà de ser detectat e informat a l'usuari, i l'execució del programa s'interromprà convenientment (exemple, que algun dels fitxers de destí estigui obert en mode exclusiu per una altra aplicació).

## Exemple d'execució del programa



```
C:\Java\Fitxers\LecturaEscriptura\build\classes>java LecturaEscriptura
Ha de passar al programa el nom d'un fitxer de text amb valors enters positius (un per línia)
No s'han passat arguments al programa, l'execució no pot continuar

C:\Java\Fitxers\LecturaEscriptura\build\classes>java LecturaEscriptura primes-to-100k
El paràmetre que s'ha passat ha de correspondre amb un nom de fitxer existent al sistema
El fitxer informat no s'ha trobat. Passi un nom de fitxer vàlid al programa per procedir

C:\Java\Fitxers\LecturaEscriptura\build\classes>java LecturaEscriptura primes-to-100k.txt
L'operació s'ha realitzat correctament. S'han creat els tres fitxers
Consulti els fitxers:  ascendents.txt, descendents.txt i altres.txt

C:\Java\Fitxers\LecturaEscriptura\build\classes>dir *.txt
El volumen de la unidad C no tiene etiqueta.
El número de serie del volumen es: 7EEE-00B1

Directorio de C:\Java\Fitxers\LecturaEscriptura\build\classes

28/03/2020  13:54                61.863  altres.txt
28/03/2020  13:54                1.991  ascendents.txt
28/03/2020  13:54                1.864  descendents.txt
11/05/2006  10:24                65.718  primes-to-100k.txt
               4 archivos                131.436 bytes
               0 dirs  37.538.926.592 bytes libres

C:\Java\Fitxers\LecturaEscriptura\build\classes>
```

## Solució de l'exercici

Aquest programa, es prou general, i pretén resumir tot el que hem vist fins ara referent a la lectura i l'escriptura de fitxers de text.

Com que aquest és un programa relativament complex, s'indica una possible solució. Evidentment, aquesta no és l'única solució, i podria no ser tampoc la millor.

```

1  /*
2  * Programa exemple per llegir i escriure fitxers en java. Llegirem del fitxer
3  * primes-to-100k.txt, que manté els primers fins al 100.000 (un per linia) i
4  * crearem tres fitxers de primers, ascendent.txt, descendent.txt i altres.txt.
5  * Aquests fitxers mantindran 10 primers per linia.
6  */
7
8  /**
9   * @author José Javier Faro
10  */
11  import java.io.*;          // Importem totes les classes
12
13  public class LecturaEsScriptura {
14      private static boolean isAscendent(String primer){
15          if (primer.length() < 2)
16              return false;
17          for (int n=1; n<primer.length(); n++)
18              if (primer.charAt(n-1) > primer.charAt(n))
19                  return false;
20          return true;
21      }
22
23      private static boolean isDescendent(String primer){
24          if (primer.length() < 2)
25              return false;
26          for (int n=1; n<primer.length(); n++)
27              if (primer.charAt(n-1) < primer.charAt(n))
28                  return false;
29          return true;
30      }
31
32      /**
33       * @param args the command line arguments
34       */
35      public static void main(String[] args) {
36          // El programa espera que li passem com a paràmetre el nom del fitxer de
37          // primers a separar. Si no li passem, no pot continuar
38          File fitxer = null;          // Fitxer de primers que llegirem
39          boolean error = false;      // Indica si s'ha produït un error
40          if (args.length == 1){      // Li hem passat el nom del fitxer a analitzar
41              try{
42                  fitxer = new File(args[0]);
43                  if (!fitxer.exists()){
44                      System.out.println("El paràmetre que s'ha passat ha de correspondre amb un nom de fitxer existent al sistema");
45                      System.out.println("El fitxer informat no s'ha trobat. Passi un nom de fitxer vàlid al programa per procedir");
46                      error = true;
47                  }
48              }catch (Exception e){
49                  System.out.println("Error accedint al sistema de fitxers");
50                  System.out.println(e.getMessage());
51                  error = true;        // No s'ha pogut accedir al sistema de fitxers?
52              }
53              // Podem continuar si no hi ha hagut errors
54              if (!error){
55                  String linia;        // Linia llegida del fitxer
56                  // Crearem els fitxers de destí a la mateixa carpeta que l'origen
57
58                  // S'ha passat al programa el nom d'un fitxer existent. Podria no tenir la sintàxis adequada
59                  try (BufferedReader br = new BufferedReader(new FileReader(fitxer))){
60                      String path = fitxer.getParent();
61                      if (path == null)
62                          path = "";
63                      else
64                          path += "\\";
65                      String ascendent = path + "ascendents.txt";
66                      String descendent = path + "descendents.txt";
67                      String altre = path + "altres.txt";
68                      File ascendents = new File(ascendent);
69                      File descendents = new File(descendent);
70                      File altres = new File(altre);
71                      // No comprovem si els fitxers de destí existeixen, els reescriurem sense preguntar
72                      try (PrintWriter asc = new PrintWriter(new BufferedWriter(new FileWriter(ascendents)))){
73                          try (PrintWriter des = new PrintWriter(new BufferedWriter(new FileWriter(descendents)))){
74                              try (PrintWriter alt = new PrintWriter(new BufferedWriter(new FileWriter(altres)))){
75                                  // Si estem aquí, és que tots els fitxers s'han pogut obrir normalment
76                                  while ((linia = br.readLine()) != null)
77                                      if (isAscendent(linia))
78                                          asc.println(linia);
79                                      else if (isDescendent(linia))
80                                          des.println(linia);
81                                      else

```

```

82         alt.println(linia);
83         System.out.println("L'operació s'ha realitzat correctament. S'han creat els tres fitxers");
84         System.out.println("Consulti els fitxers: ascendents.txt, descendents.txt i altres.txt");
85     } catch (Exception e) {
86         System.out.println("Error indeterminat en escriure al fitxer " + altre);
87         System.out.println("L'error reportat pel sistema és: " + e.getMessage());
88     }
89     } catch (Exception e) {
90         System.out.println("Error indeterminat en escriure al fitxer " + descend);
91         System.out.println("L'error reportat pel sistema és: " + e.getMessage());
92     }
93     } catch (Exception e) {
94         System.out.println("Error indeterminat en escriure al fitxer " + ascend);
95         System.out.println("L'error reportat pel sistema és: " + e.getMessage());
96     }
97     } catch (Exception e) {
98         System.out.println("Error indeterminat en llegir del fitxer " + fitxer);
99         System.out.println("L'error reportat pel sistema és: " + e.getMessage());
100    }
101    }
102    }
103    else {
104        System.out.println("Ha de passar al programa el nom d'un fitxer de text amb valors enters positius (un per línia)");
105        System.out.println("No s'han passat arguments al programa, l'execució no pot continuar");
106    }
107    }
108    }

```

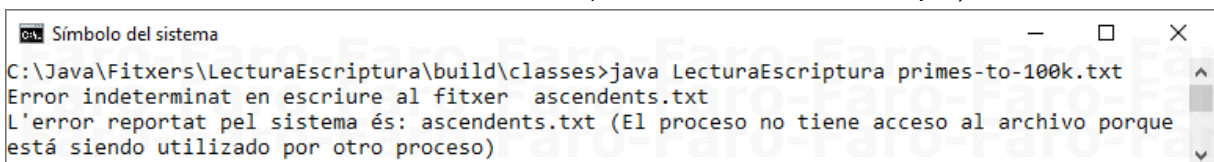
## Comentaris al programa

S'ha fet servir la funció **getParent()** de la classe **File** per deixar els fitxers de destí al mateix directori que el fitxer d'origen. Aquesta funció retorna el directori on es troba el fitxer, però si aquest està al directori per defecte, retorna **null**. Les cadenes no es poden comparar amb **==** com ja sabem, però un objecte sí que s'ha de comparar amb **null** amb **==** (que és el que fem al programa per determinar si s'ha especificat un directori al nom del fitxer d'origen).

Hem programat les funcions **isAscendent(cad)** i **isDescendent(cad)** que retornen **true** si **cad** conté respectivament una cadena ascendent o descendent. Estem passant cadenes a les funcions, de forma que qualsevol fitxer, encara que no sigui numèric, podria ser analitzat amb aquest programa sense que es provoqués cap error. Si es vol comprovar que els fitxers presenten el format esperat, s'hauria d'afegir el codi necessari. En aquest cas, caldrà tenir present quines són les accions que volem prendre si trobem alguna dada no vàlida al fitxer d'origen. Si generem una excepció i sortim del programa, els fitxers de destí s'hauran generat parcialment (també els podríem esborrar en cas d'error). Una altra possibilitat seria ignorar la línia que conté l'error i continuar endavant amb la resta de les línies del fitxer.

## Detecció d'errors

Si obrim amb el **Word** un dels fitxers (**ascendents.txt** a l'exemple)

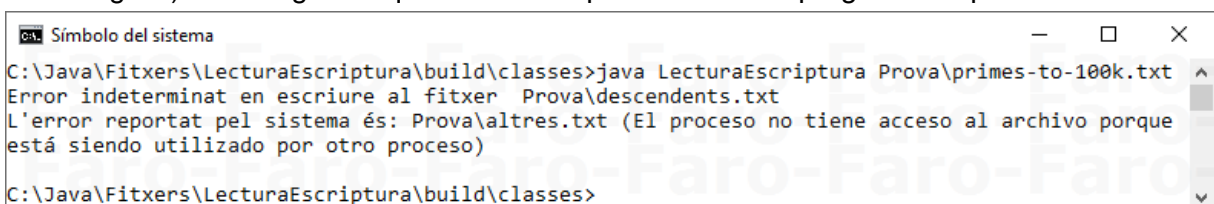


```

C:\Java\Fitxers\LecturaEscriptura\build\classes>java LecturaEscriptura primes-to-100k.txt
Error indeterminat en escriure al fitxer ascendents.txt
L'error reportat pel sistema és: ascendents.txt (El proceso no tiene acceso al archivo porque
está siendo utilizado por otro proceso)

```

L'error es produeix en crear el fitxer, i com és el primer que es crea, la resta dels fitxers, en cas de no existir, no es crearan. Si enlloc d'aquest obrim amb el **Word** el fitxer **altres.txt** (fixeu-vos que ara el fitxer original es troba al directori **Prova**, que està dintre del directori on està el programa **LecturaEscriptura.class**), llavors els altres dos fitxers (**ascendents.txt** i **descendents.txt**) es crearan i quedaran en aquest directori (queden sempre on es troba el fitxer original) amb longitud **0** quan s'interrompi l'execució del programa en provocar-se l'error.



```

C:\Java\Fitxers\LecturaEscriptura\build\classes>java LecturaEscriptura Prova\primes-to-100k.txt
Error indeterminat en escriure al fitxer Prova\descendents.txt
L'error reportat pel sistema és: Prova\altres.txt (El proceso no tiene acceso al archivo porque
está siendo utilizado por otro proceso)

C:\Java\Fitxers\LecturaEscriptura\build\classes>

```