

## ESCRITURA DE FITXERS DE TEXT EN JAVA

L'escriptura d'un fitxer de text és molt més senzilla que la lectura, ja que tenim nosaltres el control absolut del que volem escriure, i en cap cas es produirà un error encara que escrivim les dades en un format incorrecte (circumstància de la qual ens adonarem o fent inspecció del fitxer o be quan fem una lectura del mateix). De fet, escriure en un fitxer, és més senzill que escriure a la pantalla, ja que la pantalla té dues dimensions, i el fitxer només una.

A l'hora d'escriure en un fitxer de text, podem procedir de diverses formes:

- **Creació** - Obrir un nou fitxer per escriure en ell a partir de zero. El fitxer no existeix prèviament, i es crea a la ubicació indicada. A partir de llavors, cada cop que escrivim al fitxer, de forma transparent a l'usuari, l'apuntador a la informació dintre del mateix s'actualitza per a que totes les escriptures siguin seqüencials. Finalment, quan tanquem el fitxer, tots els caràcters escrits romandran al fitxer.
- **Reescriptura** - Si en indicar el fitxer que volem crear, aquest existeix prèviament, l'esborra (el deixa amb longitud **0**, de forma que el contingut previ es perd). Aquest funcionament fa que es faci imprescindible afegir codi als nostres programes per evitar reescriure fitxers existents, si no fos aquesta la nostra intenció.
- **Afegir al final** (mode **Append**) - Aquest mode d'escriptura, preserva la informació present actualment en un fitxer existent, i ens permet afegir informació al final del mateix. Els fitxers de **logs** dels programes per exemple, podrien treballar d'aquesta forma si els **logs** els volem ordenats per data (de més antic a més recent).

### Comprovar si un fitxer existeix

Una possible forma de comprovar si un fitxer existeix abans d'obrir-lo per escriptura s'indica a continuació, on el programa és molt expeditiu, si el fitxer existeix simplement acaba la seva execució sense donar alternatives a l'usuari (hi ha d'altres formes de procedir evidentment). Cal comentar que també podríem comprovar que un fitxer existeix abans d'obrir-lo per lectura, però com que en intentar obrir-ho, si no existeix, es provoca una excepció que podem controlar, no s'havia introduït aquesta comprovació fins ara.

```
String sFitxer = "noufitxer.txt";
File fitxer = new File(sFitxer);
if (fitxer.exists()){
    System.out.println("El fitxer \"" + sFitxer + "\" ja existeix al disc");
    System.out.println("Esborri prèviament aquest fitxer abans de continuar");
    System.exit(-1);
}
```

L'execució d'aquest codi, amb un fitxer existent provocarà que el programa acabi amb el següent missatge:

```
C:\WINDOWS\SYSTEM32\cmd.exe
El fitxer "noufitxer.txt" ja existeix al disc
Esborri prèviament aquest fitxer abans de continuar
```

### Escriure en un fitxer de text caràcter a caràcter

Una de les possibles formes d'escriure en un fitxer de text és caràcter a caràcter. Per fer això, podem fer servir la classe **FileWriter**.

#### Class FileWriter

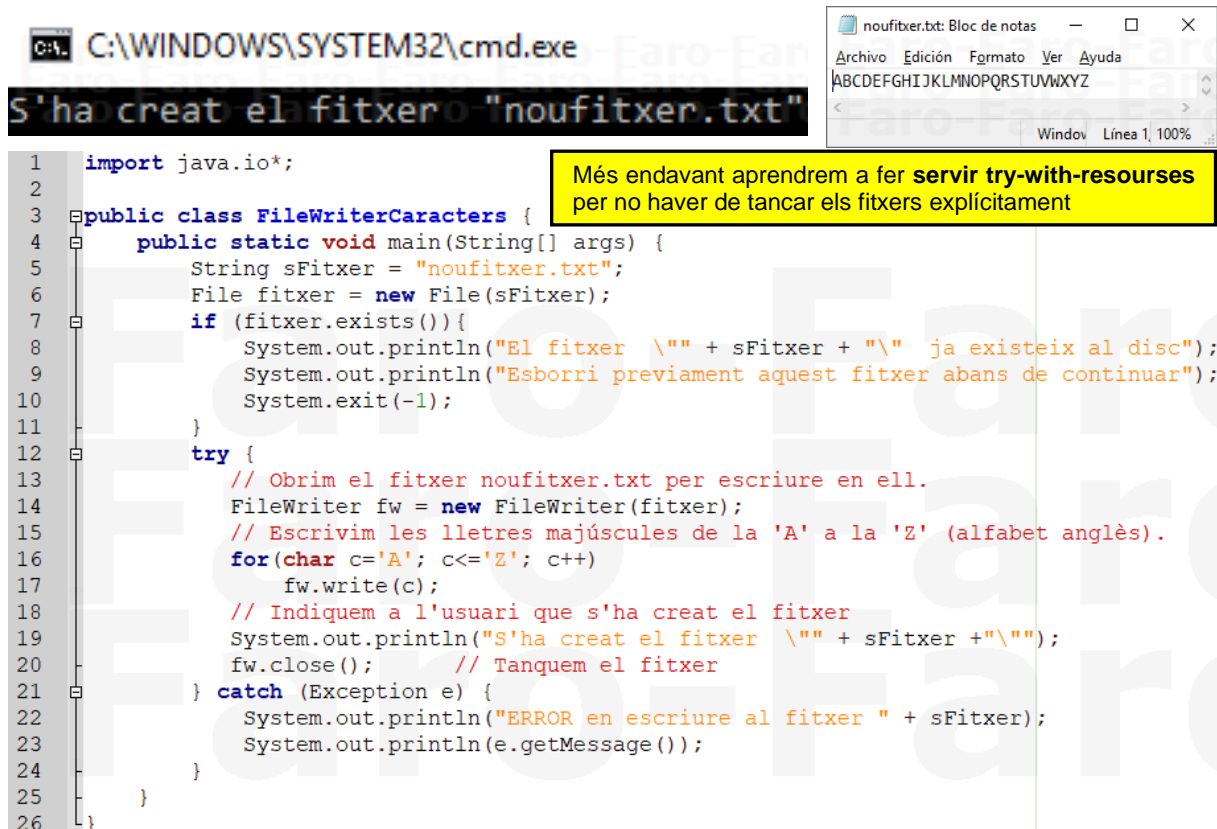
```
java.lang.Object
java.io.Writer
java.io.OutputStreamWriter
java.io.FileWriter
```

Constructor and Description	
<code>FileWriter(File file)</code>	Constructs a <code>FileWriter</code> object given a <code>File</code> object.
<code>FileWriter(File file, boolean append)</code>	Constructs a <code>FileWriter</code> object given a <code>File</code> object.
<code>FileWriter(FileDescriptor fd)</code>	Constructs a <code>FileWriter</code> object associated with a file descriptor.
<code>FileWriter(String fileName)</code>	Constructs a <code>FileWriter</code> object given a file name.
<code>FileWriter(String fileName, boolean append)</code>	Constructs a <code>FileWriter</code> object given a file name with a boolean indicating whether or not to append the data written.

Com a la lectura, tenim uns quants constructors que ens permeten escriure caràcters o línies, obrir el fitxer a partir del seu nom al disc, del seu descriptor o d'un objecte de la classe **File** creat prèviament. La creació d'aquest objecte s'ha vist a l'exemple anterior per comprovar si el fitxer existeix. Podem fer servir el mateix objecte per obrir posteriorment el fitxer.

També veiem als constructors que si volem posar la informació al final del fitxer (a continuació de la que ja existeix en aquest), afegim un paràmetre **append** de valor **true**.

A continuació, s'indica un exemple de programa que crea un fitxer de text i escriu dins les lletres de l'alfabet anglès (de la 'A' a la 'Z'). L'execució exitosa d'aquest codi, mostrarà el missatge que s'indica a l'esquerra, i a la dreta es mostra el fitxer creat pel programa:



```

1  import java.io.*;
2
3  public class FileWriterCaracters {
4      public static void main(String[] args) {
5          String sFitxer = "noufitxer.txt";
6          File fitxer = new File(sFitxer);
7          if (fitxer.exists()){
8              System.out.println("El fitxer \"" + sFitxer + "\" ja existeix al disc");
9              System.out.println("Esborri prèviament aquest fitxer abans de continuar");
10             System.exit(-1);
11         }
12         try {
13             // Obriem el fitxer noufitxer.txt per escriure en ell.
14             FileWriter fw = new FileWriter(fitxer);
15             // Escriuim les lletres majúscules de la 'A' a la 'Z' (alfabet anglès).
16             for(char c='A'; c<='Z'; c++){
17                 fw.write(c);
18             }
19             // Indiquem a l'usuari que s'ha creat el fitxer
20             System.out.println("S'ha creat el fitxer \"" + sFitxer + "\"");
21             fw.close(); // Tanquem el fitxer
22         } catch (Exception e) {
23             System.out.println("ERROR en escriure al fitxer " + sFitxer);
24             System.out.println(e.getMessage());
25         }
26     }
27 }

```

Més endavant aprendrem a fer servir **try-with-resources** per no haver de tancar els fitxers explícitament

## Esriptura d'un fitxer caràcter a caràcter amb **BufferedWriter**

Igual que al cas de la lectura, podem escriure el fitxer utilitzant un buffer intermedi que incrementarà la velocitat d'accés per a fitxers grans (en fitxers petits, de menys d'un clúster de grandària, no notarem cap diferència). Això ho farem amb la classe **BufferedWriter**.

Amb **BufferedWriter**, cada vegada que fem una escriptura al fitxer, el programa guardarà temporalment les dades fins que tingui prou quantitat per fer una escriptura eficient. Aquesta forma de treballar fa els accessos a disc més eficients i el programa anirà més ràpid. La diferència es notarà més com més gran sigui el fitxer que volem escriure.

Anem a fer un programa per escriure **102.400** caràcters aleatoris (de la 'A' a la 'Z' en un fitxer). Aquest programa el farem de dues formes, fent servir la classe **BufferedWriter** i sense fer-la servir. Com he comentat a la pràctica anterior, el portàtil que estic utilitzant, té un **SSD** (disc d'estat sòlid), amb una grandària del **cluster** de **1 KByte**. Per escriure aquest fitxer sense la classe **BufferedWriter**, necessitarem fer un total de **100** escriptures al disc.

La forma de fer servir la classe **BufferedWriter** amb un objecte de la classe **FileWriter** s'indica a continuació (fixeu-vos que enlloc de '\n' podem indicar '\r' també en **Windows**):

```

File fitxer = new File("C:/Java/Fitxers/fitxer.txt");
bw = new BufferedWriter(new FileWriter(fitxer));

```

Programa que escriu en un fitxer de text **2048** línies de **50** caràcters amb un contingut aleatori de lletres majúscules (de la 'A' a la 'Z'). Fan un total de **2048 x 50 = 102.400** caràcters al fitxer.

```

1 import java.io.*;
2
3 public class FileWriterFitxerGran {
4     public static void main(String[] args) {
5         long intervalo = System.currentTimeMillis(); // Per comptar el temps d'execució
6         FileWriter fw = null;
7         try {
8             fw = new FileWriter("C:/Java/Fitxers/fitxer.txt");
9             for (int lines = 0; lines < 2048; lines++){
10                 for (int columnes = 0; columnes < 50; columnes++){
11                     // Escriu lletres 'A' a 'Z' aleatòriament
12                     fw.write((char) (Math.random()*26+65));
13                     fw.write('\n'); // Escriu un retorn de carro
14                 }
15                 System.out.println("S'ha creat el fitxer");
16                 System.out.println("Temps: " + (System.currentTimeMillis()-intervalo)+ " ms" );
17             } catch (IOException ioe) {
18                 System.out.println(ioe.getMessage());
19             }
20             finally
21             {
22                 // Per tancar el fitxer encara que hagi hagut un error.
23                 try{
24                     if(fw!=null)
25                         fw.close();
26                 } catch (Exception ex) {
27                     System.out.println("Error en tancar el fitxer: " + ex);
28                 }
29             }
30         }
31     }
32 }

```

Si no fem servir **BufferedWriter**, l'execució serà més lenta

C:\WINDOWS\SYSTEM32\cmd.exe  
S'ha creat el fitxer  
Temps: 63 ms

## Prova del mateix programa amb **BufferedWriter**

```

1 import java.io.*;
2
3 public class FileWriterambBufferedWriter {
4     public static void main(String[] args) {
5         long intervalo = System.currentTimeMillis(); // Per comptar el temps d'execució
6         BufferedWriter bw = null; // Creem el BufferedWriter
7         try {
8             File fitxer = new File("C:/Java/Fitxers/fitxer.txt");
9             bw = new BufferedWriter(new FileWriter(fitxer));
10             for (int lines = 0; lines < 2048; lines++){
11                 for (int columnes = 0; columnes < 50; columnes++){
12                     // Escriu lletres 'A' a 'Z' aleatòriament
13                     bw.write((int) (Math.random()*26+65));
14                     bw.newLine(); // Escriu un retorn de carro
15                 }
16                 System.out.println("S'ha creat el fitxer");
17                 System.out.println("Temps: " + (System.currentTimeMillis()-intervalo)+ " ms" );
18             } catch (IOException ioe) {
19                 System.out.println(ioe.getMessage());
20             }
21             finally
22             {
23                 // Per tancar el fitxer encara que hagi hagut un error.
24                 try{
25                     if(bw!=null)
26                         bw.close();
27                 } catch (Exception ex) {
28                     System.out.println("Error en tancar el fitxer: " + ex);
29                 }
30             }
31         }
32     }
33 }

```

Amb **BufferedWriter**, l'execució és més ràpida

C:\WINDOWS\SYSTEM32\cmd.exe  
S'ha creat el fitxer  
Temps: 31 ms

Fixeu-vos que el temps d'execució d'aquest segon codi és de **31 ms**. El mateix programa, realitzat amb la classe **BufferedWriter** triga la meitat de temps en executar-se. La diferència es faria més gran si el fitxer creat tingués més grandària.

## Escriure un fitxer de text línia a línia amb FileWriter

Els objectes de la classe **FileWriter** disposen del mètode **write(s)** per escriure cadenes de text en un fitxer. Així com els objectes de la classe **BufferedReader** disposen d'un mètode **readLine()** per llegir una línia del fitxer, la classe **BufferedWriter** permet escriure una cadena amb el mètode **write**.

A continuació, s'indica un programa per escriure línies de text en un fitxer utilitzant la classe **FileWriter**. Es pot apreciar també la creació de cadenes amb la instrucció **format**, que ja es va comentar a la UF1. El fitxer que es vol construir s'indica a la dreta. Es pot apreciar que a cada línia indiquem la línia del fitxer (%5d, un enter en un camp de **cinc**, precedit d'espais), i un valor aleatori (%8.3f valor en coma flotant, en un camp de **vuit** precedit d'espais i amb **tres** decimals). L'execució del programa mostra el missatge:

C:\WINDOWS\SYSTEM32\cmd.exe

Temps: 1343 ms  
S'ha escrit el fitxer correctament

Archivo	Edición	Formato	Ver	Ayuda
Línia 0:	Valor:	845,954		
Línia 1:	Valor:	71,138		
Línia 2:	Valor:	329,757		
Línia 3:	Valor:	80,101		
Línia 4:	Valor:	857,430		
Línia 5:	Valor:	142,293		
Línia 6:	Valor:	884,545		
Línia 7:	Valor:	752,708		
Línia 8:	Valor:	453,690		
Línia 9:	Valor:	905,862		
Línia 10:	Valor:	680,325		
Línia 11:	Valor:	970,229		

```

1 import java.io.File;
2 import java.io.FileWriter;
3 import java.io.IOException;
4
5 public class EscriureLiniesFileWriter {
6     public static void main(String[] args) {
7         try {
8             long temps = System.currentTimeMillis(); // Comptem el temps d'execució
9             // No comprovem si el fitxer existeix. Es reescriurà.
10            FileWriter fw = new FileWriter(new File("filewriter.txt"));
11            for (int n=0; n<50000; n++){
12                fw.write(String.format("Línia %5d: ",n));
13                fw.write(String.format("      Valor: %8.3f",1000*Math.random()));
14                fw.write(System.lineSeparator()); //Nova línia
15            }
16            fw.close(); // No tanquem el fitxer en cas d'error
17            System.out.println("Temps: " + (System.currentTimeMillis()-temps)+ " ms" );
18            System.out.println("S'ha escrit el fitxer correctament");
19        } catch (IOException ex) {
20            System.out.println(ex.getMessage());
21        }
22    }
23 }

```

## Escriure un fitxer de text línia a línia amb BufferedWriter

El mateix exemple amb **BufferedWriter**. L'increment a la velocitat d'execució és gairebé inapreciable en el cas de l'escriptura, tot i que el fitxer resultant és major de **1,5 MBytes**.

```

1 import java.io.*;
2
3 public class EscriureLiniesBufferedWriter {
4     public static void main(String[] args) {
5         try {
6             long temps = System.currentTimeMillis(); // Comptem el temps d'execució
7             // No comprovem si el fitxer existeix. Es reescriurà.
8             BufferedWriter bf = new BufferedWriter(new FileWriter("filewriter.txt"));
9             for (int n=0; n<50000; n++){
10                bf.write(String.format("Línia %5d: ",n));
11                bf.write(String.format("      Valor: %8.3f",1000*Math.random()));
12                bf.newLine(); // Nova línia
13            }
14            bf.close(); // No tanquem el fitxer en cas d'error
15            System.out.println("Temps: " + (System.currentTimeMillis()-temps)+ " ms" );
16            System.out.println("S'ha escrit el fitxer correctament");
17        } catch (IOException ex) {
18            System.out.println(ex.getMessage());
19        }
20    }
21 }

```

C:\WINDOWS\SYSTEM32\cmd.exe

Temps: 1328 ms  
S'ha escrit el fitxer correctament



## Crear un fitxer temporal en java

En moltes circumstàncies és necessari la creació de fitxers que emmagatzemaran dades de forma provisional, i que finalment seran esborrats un cop hagin realitzat la seva funció. Aquest tipus de fitxers els creem per emmagatzemar resultats temporals que posteriorment ja no necessitem. Creem el fitxer, treballem amb ell i després quan ha deixat de tenir utilitat per a nosaltres l'esborrem. Com escollim el nom i la ubicació d'aquest fitxer?. Evidentment, una possibilitat seria donar-li qualsevol nom, però hi ha solucions més elegants.

Java ens ofereix un conjunt extens de mètodes en la classe **File**, i entre ells trobem **createTempFile** que ens permet crear un fitxer temporal, en el directori que l'indiquem, o bé al directori temporal del sistema.

```
static File createTempFile(String prefix, String suffix)
    Creates an empty file in the default temporary-file directory, using the given prefix and suffix to
    generate its name.

static File createTempFile(String prefix, String suffix, File directory)
    Creates a new empty file in the specified directory, using the given prefix and suffix strings to
    generate its name.
```

Al java, podem saber quin és el directori temporal del sistema utilitzant la classe **System** que entre d'altre coses ens permet accedir a les propietats del sistema. La que ens interessa és **java.io.tmpdir**, i per accedir-hi utilitzem el mètode **getProperty()**. A continuació s'indica com escriure el directori temporal des d'un programa java:

```
System.out.println("Directorio temporal del sistema: " + System.getProperty("java.io.tmpdir"));
```

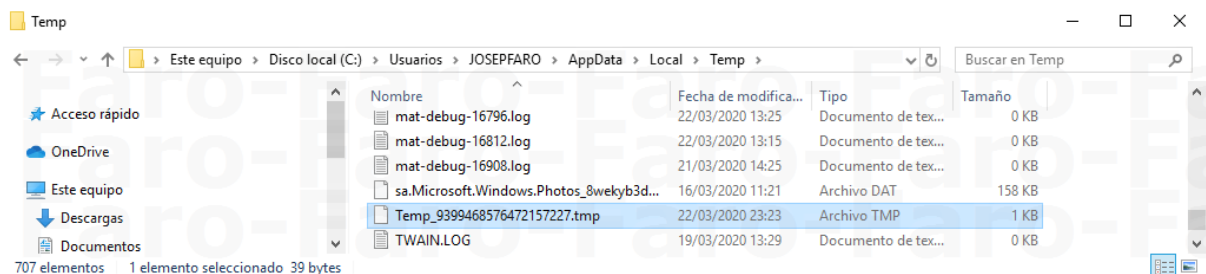
C:\WINDOWS\SYSTEM32\cmd.exe

Directorio temporal del sistema: C:\Users\JOSEPFARO\AppData\Local\Temp\

El següent programa, crea un fitxer temporal al directori temporal del sistema i escriu una frase en aquest utilitzant l'últim mètode vist (**FileWriter** + **BufferedWriter**).

```
1 import java.io.*;
2 public class FitxerTemporal {
3     public static void main(String[] args) {
4         BufferedWriter bw = null;
5         try{
6             System.out.println("Directorio temporal: " + System.getProperty("java.io.tmpdir"));
7             File tempFile = File.createTempFile("Temp_", null);
8             bw = new BufferedWriter(new FileWriter(tempFile));
9             bw.write("Escribim aquest text al fitxer temporal");
10            bw.close(); // Tanquem el fitxer. En cas d'error no el tanquem
11        } catch (Exception e) { // Qualsevol error
12            System.out.println(e.getMessage()); // Escribim el missatge d'error
13        }
14    }
15 }
```

El fitxer temporal creat tindrà el contingut que hem escrit, una grandària d'un clúster (1K en aquest sistema) i un nom que començarà per **Temp\_** seguit d'un número aleatori, i l'extensió **tmp**.



Normalment, aquests fitxers temporals no tenen sentit que quedin al nostre sistema un cop acaba la seva execució el programa que els ha creat. Per esborrar el fitxer temporal creat en finalitzar el nostre programa, hem d'afegir al programa la sentència que s'indica:

```
File tempFile = File.createTempFile("Temp_", null);
tempFile.deleteOnExit();
```

## Escriure fitxers de text amb **PrintWriter** i **PrintStream**

**PrintWriter** és una classe que ens ajuda a escriure text a la sortida estàndard, així com a fitxers, cadenes o fluxos (**streams**). Implementa tots els mètodes de la classe **PrintStream**. A diferència d'aquesta, els seus mètodes d'escriptura no es buiden en escriure cada nova línia i per tant executa més ràpid. Totes dues classes tenen mètodes que ens permeten escriure un enter, un long, un número en coma flotant, i en general de qualsevol dels tipus de dades del llenguatge, incloent-hi objectes i booleans. Podríem dir, que són classes d'alt nivell, que contemplen totes les possibilitats de les classes anteriors.

### Class **PrintWriter**

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

### Class **PrintStream**

```
java.lang.Object
java.io.OutputStream
java.io.FilterOutputStream
java.io.PrintStream
```

S'indiquen a continuació, dos senzills programes que fan servir aquestes dues classes per escriure fitxers, s'ha comptabilitzat el temps que els programes triguen en executar-se:

```
1 import java.io.*;
2
3 public class EscriureLiniesPrintWriter {
4     public static void main(String[] args) {
5         try {
6             long temps = System.currentTimeMillis(); // Comptem el temps d'execució
7             // No comprovem si el fitxer existeix. Es reescriurà.
8             PrintWriter pw = new PrintWriter("printwriter.txt");
9             for (int n=0; n<50000; n++){
10                 pw.print(String.format("Línia %5d: ",n));
11                 pw.print(String.format("    Valor: %8.3f",1000*Math.random()));
12                 pw.println(); //Nova línia
13             }
14             pw.close(); // No tanquem el fitxer en cas d'error
15             System.out.println("Temps: "+ (System.currentTimeMillis()-temps)+ " ms" );
16             System.out.println("S'ha escrit el fitxer correctament");
17         } catch (IOException ex) {
18             System.out.println(ex.getMessage());
19         }
20     }
21 }
```

C:\WINDOWS\SYSTEM32\cmd.exe  
Temps: 1360 ms  
S'ha escrit el fitxer correctament

```
1 import java.io.*;
2
3 public class EscriureLiniesPrintStream {
4     public static void main(String[] args) {
5         try {
6             long temps = System.currentTimeMillis(); // Comptem el temps d'execució
7             // No comprovem si el fitxer existeix. Es reescriurà.
8             PrintStream ps= new PrintStream("printstream.txt");
9             for (int n=0; n<50000; n++){
10                 ps.print(String.format("Línia %5d: ",n));
11                 ps.print(String.format("    Valor: %8.3f",1000*Math.random()));
12                 ps.println(); //Nova línia
13             }
14             ps.close(); // No tanquem el fitxer en cas d'error
15             System.out.println("Temps: "+ (System.currentTimeMillis()-temps)+ " ms" );
16             System.out.println("S'ha escrit el fitxer correctament");
17         } catch (IOException ex) {
18             System.out.println(ex.getMessage());
19         }
20     }
21 }
```

C:\WINDOWS\SYSTEM32\cmd.exe  
Temps: 3048 ms  
S'ha escrit el fitxer correctament

Com es pot apreciar el programa realitzat a partir de la classe **PrintStream** ha trigat molt més en executar-se. El motiu és que amb aquesta classe, després d'escriure un retorn de carro o invocar el mètode **println()**, el buffer es buida cap al fitxer. En els dos programes anteriors, podem buidar el buffer en qualsevol moment invocant al mètode **flush()**.

Molts programes, com per exemple el **Word**, de **Microsoft**, obren els fitxers en mode exclusiu. Quan obro un fitxer per escriptura sense indicar que vull afegir al final (**append**), si aquest fitxer ja existeix, el contingut previ es perdrà. Als programes anteriors, no he afegit cap codi per detectar aquesta possibilitat. Si els fitxers que creen els dos programes anteriors, estiguessin oberts per una altra aplicació en mode exclusiu (com per exemple el Word), en executar-los, obtindré els dos missatges d'error que s'indiquen a continuació:

C:\WINDOWS\SYSTEM32\cmd.exe

printwriter.txt (El proceso no tiene acceso al archivo porque está siendo utilizado por otro proceso)

C:\WINDOWS\SYSTEM32\cmd.exe

printstream.txt (El proceso no tiene acceso al archivo porque está siendo utilizado por otro proceso)

A continuació, s'indica un exemple complet que fa servir la classe **PrintWriter**, a partir d'un objecte **BufferedWriter** que s'ha fet servir per incrementar la velocitat d'execució per accedir a un fitxer a partir d'un objecte de la classe **FileWriter**. També s'ha indicat com procedir si el fitxer existeix prèviament per no reescriure'l. En aquest cas, s'ha preferit demanar confirmació a l'usuari abans de continuar. S'ha modificat també el programa posant el tancament dins de la sentència **finally**, d'aquesta forma, quan es produeix un error en temps d'execució, intentarem igualment tancar el fitxer. També és convenient saber que el sistema operatiu s'encarrega de tancar tots els fitxers oberts per les aplicacions que han acabat la seva execució. Tot i això, sempre és millor que la nostra aplicació tanqui tots els fitxers que hagi obert abans de terminar, i no deixar en mans del sistema operatiu aquesta responsabilitat.

```

1  import java.io.*;
2  import java.util.Scanner;
3
4  public class EscriureLiniesPrintWriterFileWriter {
5      public static void main(String[] args) {
6          PrintWriter pw = null;
7          try {
8              Scanner sc = new Scanner(System.in);
9              // No comprovem si el fitxer existeix. Es reescriurà.
10             File fitxer = new File("filewriter_printwriter.txt");
11             char sino = 'N';
12             while (fitxer.exists() && sino!='S') {
13                 System.out.println("El fitxer " + fitxer + " ja existeix al directori");
14                 System.out.print("Desitja sobreescure aquest fitxer i perdre el contingut previ [S/N]: ");
15                 sino = sc.nextLine().toUpperCase().charAt(0);
16                 if (sino == 'N')
17                     System.exit(-1); // Acabem l'execució del programa
18             }
19             long temps = System.currentTimeMillis(); // Contem el temps d'execució
20             FileWriter fw = new FileWriter(fitxer);
21             BufferedWriter bw = new BufferedWriter(fw);
22             pw = new PrintWriter(bw);
23
24             for (int n=0; n<50000; n++){
25                 pw.print(String.format("Línia %5d: ",n));
26                 pw.print(String.format(" Valor: %8.3f",1000*Math.random()));
27                 pw.println(); //Nova línia
28             }
29             System.out.println("Temps: " + (System.currentTimeMillis()-temps)+ " ms");
30             if (!pw.checkError())
31                 System.out.println("S'ha escrit el fitxer correctament");
32             else
33                 System.out.println("Error indeterminat en la creació del fitxer");
34         } catch (IOException ex) {
35             System.out.println(ex.getMessage()); // Indiquem l'error
36         }
37         finally{
38             try{
39                 if (pw != null) // Controlem l'error en tancar el fitxer
40                     pw.close();
41             } catch (Exception e){
42                 System.out.println(e.getMessage()); // Indiquem l'error
43             }
44         }
45     }
46 }

```

El fitxer filewriter\_printwriter.txt ja existeix al directori.  
Desitja sobreescure aquest fitxer i perdre el contingut previ [S/N]: yo no se  
El fitxer filewriter\_printwriter.txt ja existeix al directori.  
Desitja sobreescure aquest fitxer i perdre el contingut previ [S/N]: s  
Temps: 1219 ms  
S'ha escrit el fitxer correctament

Com es pot apreciar, hi ha un lleuger increment a la velocitat d'execució

Fixeu-vos també a l'ús del mètode **checkError()** que retorna **true** si s'ha produït qualsevol error en l'escriptura del fitxer. Això és convenient per garantir que l'escriptura del fitxer s'ha realitzat correctament.