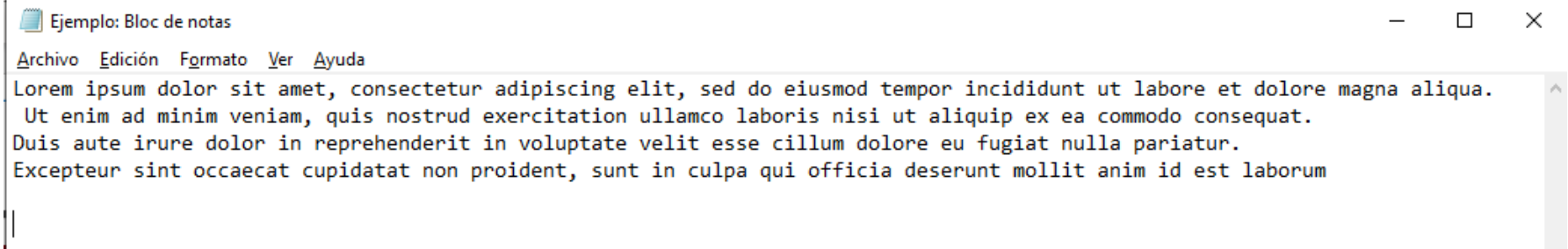
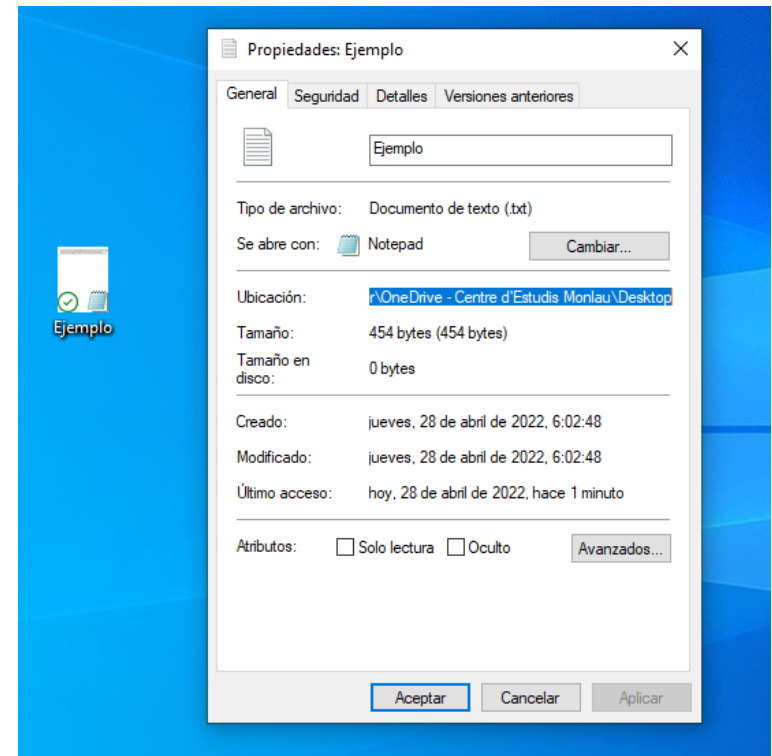


M₃-UF₃ –FICHEROS

- Leer ficheros:
 - Usando Scanner
 - Usando BufferedReader
 - Otras formas de leer ficheros
- Escribir ficheros usando BufferedWriter
 - Otras formas de escribir ficheros
- Otras operaciones con ficheros

Leer ficheros Usando Scanner

- Partimos de un fichero existente
- Podemos usar todos los métodos de Scanner (next(), nextInt(), ...) en función de la estructura y el contenido del fichero que queramos leer



Leer ficheros Usando Scanner

```
public class LecturaConScanner02 {  
    public static void main(String[] args) throws FileNotFoundException {  
        // En vez de "\", "\\\" para que Java no lo confunda con un carácter de escape, un código NO imprimible como "\n"  
        String fileName = "C:\\Users\\user\\OneDrive - Centre d'Estudis Monlau\\Desktop\\Ejemplo.txt";  
        //Creamos un objeto File, que será la representación en Java de nuestro fichero  
        File fichero = new File(fileName); //Para trabajar con la clase File necesitamos hacer import java.io.File  
  
        //Cuando indicamos que vamos a leer un fichero A, si A no existe se produce un error, una excepción (Exception)  
        //Para controlar el error, ponemos la sentencia que puede ir mal dentro de un try - catch para capturar la excepción  
        // o propagar el error hacia un nivel superior. En este caso como estamos en el main, se mostrará el error por pantalla.  
  
        Scanner in = new Scanner(fichero);  
  
        //Leemos el fichero línea a línea hasta leerlas todas  
        while (in.hasNextLine()) { //Mientras queden líneas por leer  
            String line = in.nextLine();  
            System.out.println(line);  
        }  
  
        in.close(); //Cuando trabajamos con ficheros hace falta cerrarlo  
    }  
}
```

Es importante cerrar los ficheros después de tratarlos para evitar problemas de falta de memoria, etc (si manejamos muchos ficheros abiertos a la vez).

Leer ficheros Usando Scanner

```
public class LecturaConScanner02 {
    public static void main(String[] args) throws FileNotFoundException {
        // En vez de "\", "\\\" para que Java no lo confunda con un carácter de escape, un código NO imprimible como "\n"
        String fileName = "C:\\\\Users\\user\\OneDrive - Centre d'Estudis Monlau\\Desktop\\Ejemplo.txt";
        //Creamos un objeto File, que será la representación en Java de nuestro fichero
        File fichero = new File(fileName); //Para trabajar con la clase File necesitamos hacer import java.io.File

        //Cuando indicamos que vamos a leer un fichero A, si A no existe se produce un error, una excepción (Exception)
        //Para controlar el error, ponemos la sentencia que puede ir mal dentro de un try - catch para capturar la excepción
        // o propagar el error hacia un nivel superior. En este caso como estamos en el main, se mostrará el error por pantalla

        Scanner in = new Scanner(fichero);

        //Leemos el fichero línea a línea hasta leerlas todas
        while (in.hasNextLine()) { //Mientras queden líneas por leer
            String line = in.nextLine();
            System.out.println(line);
        }

        in.close(); //Cuando trabajamos con ficheros hace falta cerrarlo
    }
}
```

ficheros.LecturaConScanner02 > main > fileName >

out - JavaApplication1 (run) x Analyzer

```
run:
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

BUILD SUCCESSFUL (total time: 8 seconds)
```

Leer ficheros Usando Scanner

Ponemos un nombre de fichero inexistente

```
public class LecturaConScanner02 {  
    public static void main(String[] args) throws FileNotFoundException {  
        // En vez de "\", "\\\" para que Java no lo confunda con un carácter de escape, un código NO imprimible como "\n"  
        String fileName = "C:\\Users\\user\\OneDrive - Centre d'Estudis Monlau\\Desktop\\Error.txt";  
        //Creamos un objeto File, que será la representación en Java de nuestro fichero  
        File fichero = new File(fileName); //Para trabajar con la clase File necesitamos hacer import java.io.File  
  
        //Cuando indicamos que vamos a leer un fichero A, si A no existe se produce un error, una excepción (Exception)  
        //Para controlar el error, ponemos la sentencia que puede ir mal dentro de un try - catch para capturar la excepción  
        // o propagar el error hacia un nivel superior. En este caso como estamos en el main, se mostrará el error por pantalla  
  
        Scanner in = new Scanner(fichero);  
  
        //Leemos el fichero línea a línea hasta leerlas todas  
        while (in.hasNextLine()) { //Mientras queden líneas por leer  
            String line = in.nextLine();  
            System.out.println(line);  
        }  
  
        in.close(); //Cuando trabajamos con ficheros hace falta cerrarlo  
    }  
}
```

ficheros.LecturaConScanner02 >

out - JavaApplication1 (run) x Analyzer

run:
Exception in thread "main" java.io.FileNotFoundException: C:\Users\user\OneDrive - Centre d'Estudis Monlau\Desktop\Error.txt (El sistema no puede acceder al archivo)
at java.base/java.io.FileInputStream.open0(Native Method)
at java.base/java.io.FileInputStream.open(FileInputStream.java:211)
at java.base/java.io.FileInputStream.<init>(FileInputStream.java:153)
at java.base/java.util.Scanner.<init>(Scanner.java:639)
at ficheros.LecturaConScanner02.main(LecturaConScanner02.java:27)
C:\Users\user\AppData\Local\NetBeans\Cache\12.6\executor-snippets\run.xml:111: The following error occurred while executing this line:

Error no controlado. Se imprime la StackTrace

Leer ficheros Usando Scanner

Ponemos un nombre de fichero inexistente

```
public static void main(String[] args) {  
    // En vez de "\", "\\\" para que Java no lo confunda con un carácter de escape, un código NO imprimible como "\n"  
    String fileName = "C:\\Users\\user\\OneDrive - Centre d'Estudis Monlau\\Desktop\\Error.txt";  
    //Creamos un objeto File, que será la representación en Java de nuestro fichero  
    File fichero = new File(fileName); //Para trabajar con la clase File necesitamos hacer import java.io.File  
  
    //Cuando indicamos que vamos a leer un fichero A, si A no existe se produce un error, una excepción (Exception)  
    //Para controlar el error, ponemos la sentencia que puede ir mal dentro de un try - catch para capturar la excepción  
    // y poder dar el error o repararlo de forma controlada:  
    try {  
        Scanner in = new Scanner(fichero);  
        while (in.hasNextLine()) {  
            String line = in.nextLine();  
            System.out.println(line);  
        }  
    } catch (FileNotFoundException ex) {  
        System.out.println("El fichero " + fichero.getName() + " no existe"); //Si fichero no existe daremos este mensaje  
    }  
}
```

ficheros.LecturaConScanner01

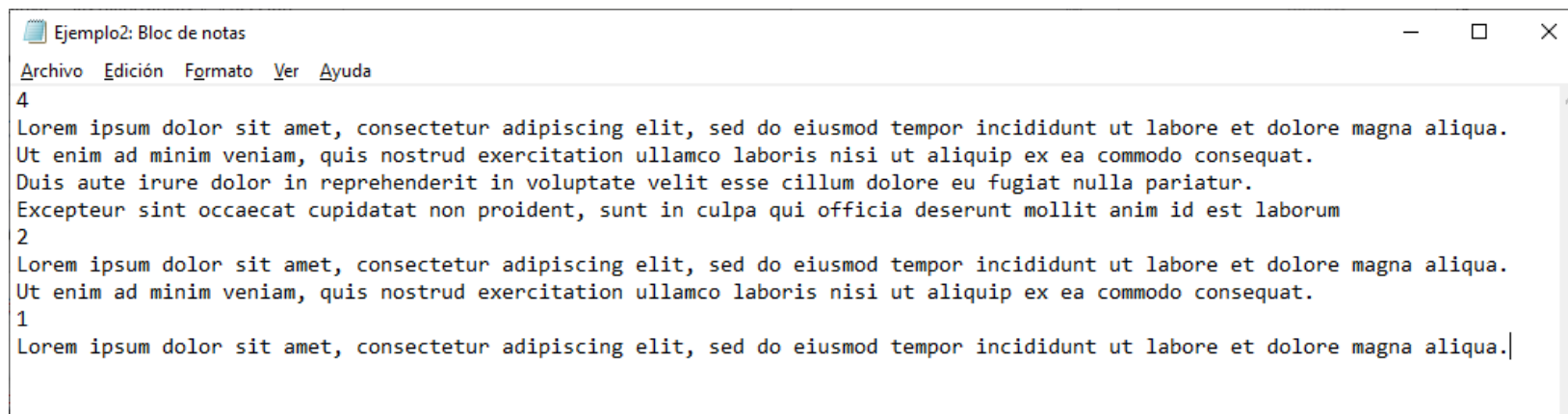
out - JavaApplication1 (run) x Analyzer

```
run:  
El fichero Error.txt no existe  
BUILD SUCCESSFUL (total time: 1 second)
```

Error controlado. Si el programa lo va usar un usuario final preferirá ver un mensaje personalizado que la pila de llamadas a métodos.

Leer ficheros Usando Scanner

- Cambiamos el contenido del fichero.
- Los números indican las líneas que vienen a continuación



```
Ejemplo2: Bloc de notas
Archivo Edición Formato Ver Ayuda
4
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum
2
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
1
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
```

- La línea 1 contiene: 4
- La línea 2 contiene: Lorem ipsum dolor sit amet ...
- La línea 3 contiene: Ut enim ad minim veniam, ...

Leer ficheros Usando Scanner

```
Scanner in = new Scanner(fichero);

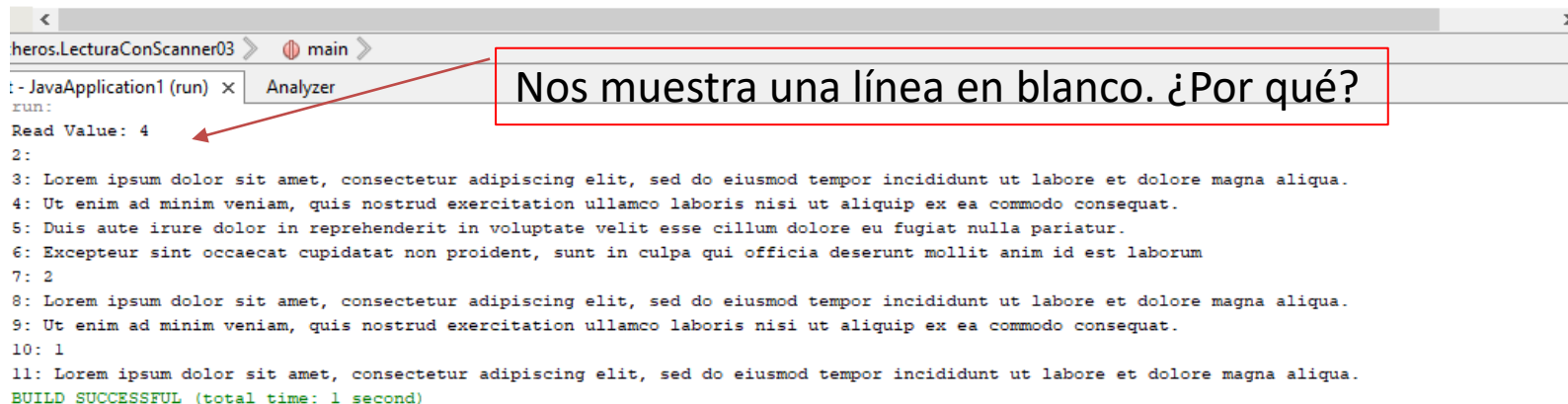
int value = in.nextInt();
System.out.println("Read Value: " + value);

int countLine = 2; //Inicializamos el contador de líneas a 2 porque previamente hemos hecho una lectura (la del número)

//Leemos el fichero línea a línea hasta leerlas todas
while (in.hasNextLine()) { //Mientras queden líneas por leer
    String line = in.nextLine();
    System.out.println(countLine + ": " + line);
    countLine++;
}

in.close(); //Cuando trabajamos con ficheros hace falta cerrarlo
```

Imprimimos el número de línea antes de su contenido



```
heros.LecturaConScanner03 > main >
:- JavaApplication1 (run) x Analyzer
run:
Read Value: 4
2:
3: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
4: Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
5: Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
6: Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum
7: 2
8: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
9: Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
10: 1
11: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
BUILD SUCCESSFUL (total time: 1 second)
```

Nos muestra una línea en blanco. ¿Por qué?

Leer ficheros Usando Scanner

```
Scanner in = new Scanner(fichero);

int value = in.nextInt();
System.out.println("Read Value: " + value);
in.nextLine();
int countLine = 2; //Inicializamos el contador de líneas por leer

//Leemos el fichero línea a línea
while (in.hasNextLine()) { //Mientras queden líneas por leer
    String line = in.nextLine();
    System.out.println(countLine + ": " + line);
    countLine++;
}

in.close(); //Cuando trabajamos con ficheros hace falta cerrarlo
}
```

El problema es que si usamos next(), nextInt(), nextDouble(), ... (cualquier cosa que no sea nextLine()), no estamos leyendo el “retorno de carro” (=“\n”) y eso queda pendiente de leer. Cuando hacemos el siguiente nextLine(), lo primero que nos encontramos es este carácter NO imprimible (invisible) y lo lee. Cuando lo “printamos” lo que aparece es una línea en blanco (un salto). Esto se soluciona añadiendo nextLine() para leerlo.

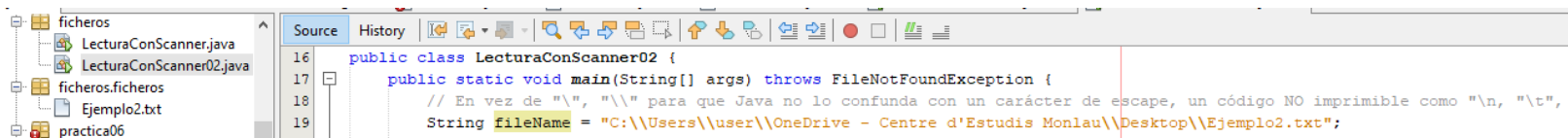
```
cheros.LecturaConScanner03 > main >
t - JavaApplication1 (run) x Analyzer

run:
Read Value: 4
2: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
3: Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
4: Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
5: Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum
6: 2
7: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
8: Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
9: 1
10: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
BUILD SUCCESSFUL (total time: 1 second)
```

Ahora la numeración de líneas es correcta

Leer ficheros Usando Scanner

- **¿QUÉ PASA SI MOVEMOS EL FICHERO DE SITIO?**



```
16 public class LecturaConScanner02 {  
17     public static void main(String[] args) throws FileNotFoundException {  
18         // En vez de "\", "\\" para que Java no lo confunda con un carácter de escape, un código NO imprimible como "\n", "\t",  
19         String fileName = "C:\\Users\\user\\OneDrive - Centre d'Estudis Monlau\\Desktop\\Ejemplo2.txt";  
20     }  
21 }
```

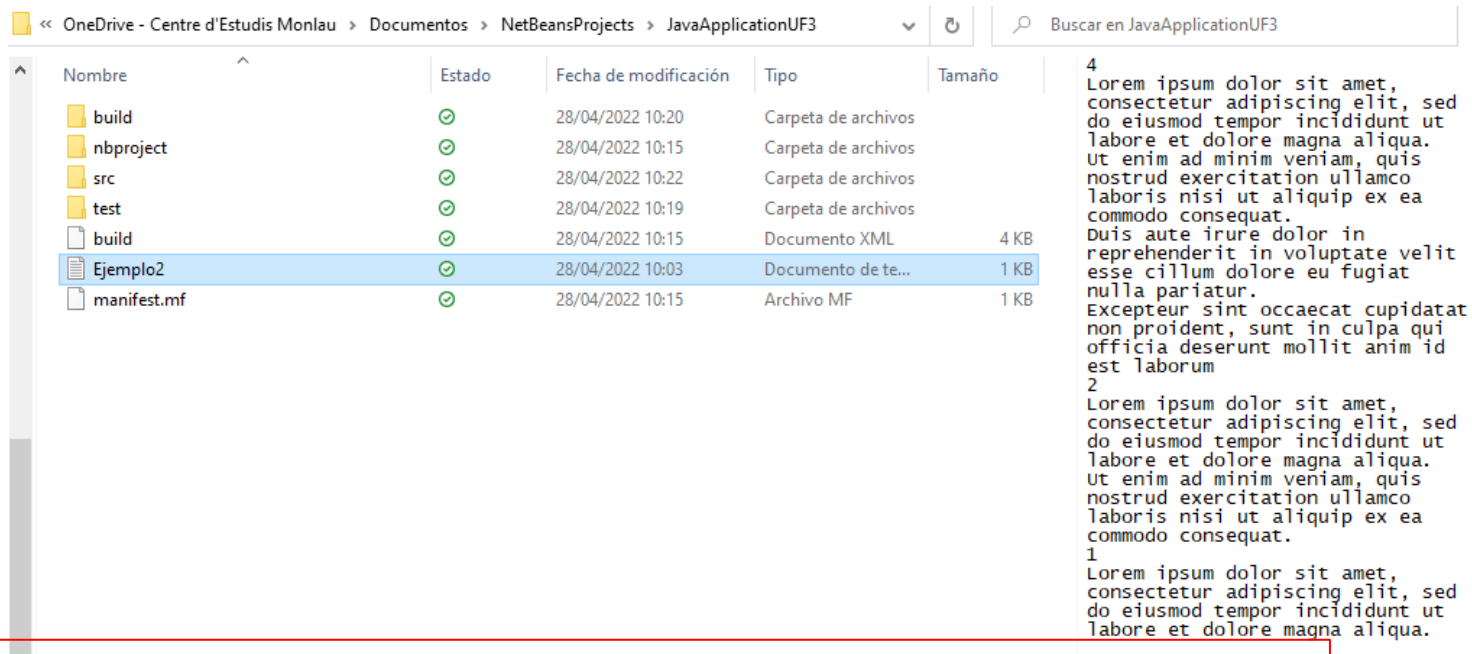
Si movemos los ficheros del Escritorio al package donde tenemos los .java, la ruta especificada será incorrecta y esto provocará un error.



```
Output - JavaApplication1 (run) x  
Exception in thread "main" java.io.FileNotFoundException: C:\Users\user\OneDrive - Centre d'Estudis Monlau\Desktop\Ejemplo2.txt (El sistema no puede encontrar el archivo especificado)  
    at java.io.FileInputStream.open0(Native Method)  
    at java.io.FileInputStream.open(FileInputStream.java:145)  
    at java.io.FileInputStream.<init>(FileInputStream.java:100)  
    at java.io.Scanner.<init>(Scanner.java:534)  
    at LecturaConScanner02.main(LecturaConScanner02.java:17)
```

Leer ficheros Usando Scanner

- **SOLUCIÓN**



The screenshot shows a file explorer window with the following table of files:

Nombre	Estado	Fecha de modificación	Tipo	Tamaño
build	✓	28/04/2022 10:20	Carpeta de archivos	
nbproject	✓	28/04/2022 10:15	Carpeta de archivos	
src	✓	28/04/2022 10:22	Carpeta de archivos	
test	✓	28/04/2022 10:19	Carpeta de archivos	
build	✓	28/04/2022 10:15	Documento XML	4 KB
Ejemplo2	✓	28/04/2022 10:03	Documento de te...	1 KB
manifest.mf	✓	28/04/2022 10:15	Archivo MF	1 KB

The file 'Ejemplo2' is selected. The right pane shows the content of the file, which is a Lorem Ipsum text block.

Si tenemos el fichero en el directorio del proyecto. Si declaramos:
String fileName = "Ejemplo2.txt", Java lo encontrará aunque no hayamos especificado la ruta completa.

Leer ficheros Usando Scanner

- SOLUCIÓN**

```
public class LecturaConScanner02 {
    public static void main(String[] args) throws FileNotFoundException {
        // En vez de "\", "\\\" para que Java no lo confunda con un carácter de escape, un código NO imprimible como "\n", "\t", .
        //String fileName = "C:\\Users\\user\\OneDrive - Centre d'Estudis Monlau\\Desktop\\Ejemplo2.txt";

        //En vez de escribir la ruta entera del fichero. Sólo pondremos el nombre del archivo.
        //Si el fichero se encuentra en nuestro directorio de trabajo, el programa lo encontrará y funcionará.
        String fileName = "Ejemplo2.txt";
        //Creamos un objeto File, que será la representación en Java de nuestro fichero
        File fichero = new File(fileName); //Para trabajar con la clase File necesitamos hacer import java.io.File

        //Cuando indicamos que vamos a leer un fichero A, si A no existe se produce un error, una excepción (Exception)
        //Para controlar el error, ponemos la sentencia que puede ir mal dentro de un try - catch para capturar la excepción
        // o propagar el error hacia un nivel superior. En este caso como estamos en el main, se mostrará el error por pantalla.

        Scanner in = new Scanner(fichero);

        int value = in.nextInt();
    }
}
```

ficheros.LecturaConScanner02 > main > fileName >

put - JavaApplicationUF3 (run) x

```
run:
Read Value: 4
2: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
3: Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
4: Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
5: Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum
6: 2
7: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
8: Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
9: 1
10: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
BUILD SUCCESSFUL (total time: 0 seconds)
```

Funciona

Leer ficheros Usando Scanner

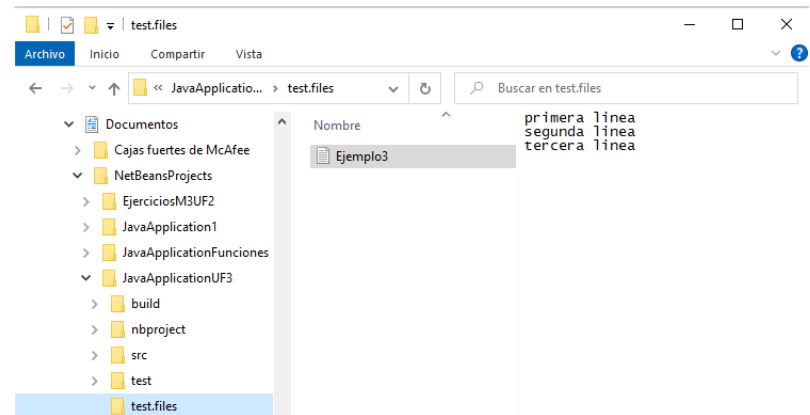
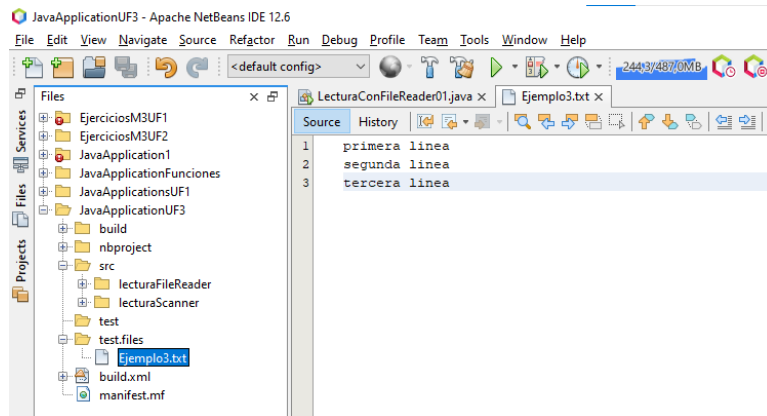
- **OTRAS SOLUCIONES:**

- `String fileName = "./Ejemplo2.txt";` // “.” indica el directorio actual
- `String fileName = System.getProperty("user.dir") + "/" + "Ejemplo2.txt";`
// `getProperty("user.dir")` devuelve el directorio de trabajo en el que está
// el programa

Leer ficheros Usando BufferedReader

- Es un mecanismo de lectura de ficheros algo más antiguo y engorroso que Scanner.
- Es importante conocerlo porque mucha gente aún usa versiones antiguas de Java que no soportan las nuevas características.
- Cuando usamos BufferedReader debemos trabajar también con otra clase: FileReader
- La diferencia entre FileReader y BufferedReader es que aunque con las 2 clases proporcionan métodos para leer:
 - FileReader lo hace carácter a carácter y BufferedReader permite leer líneas completas.
 - El final de fichero con FileReader se identifica con -1 mientras que BufferedeReader devuelve null
 - BufferedReader es más óptimo.

Leer ficheros Usando BufferedReader

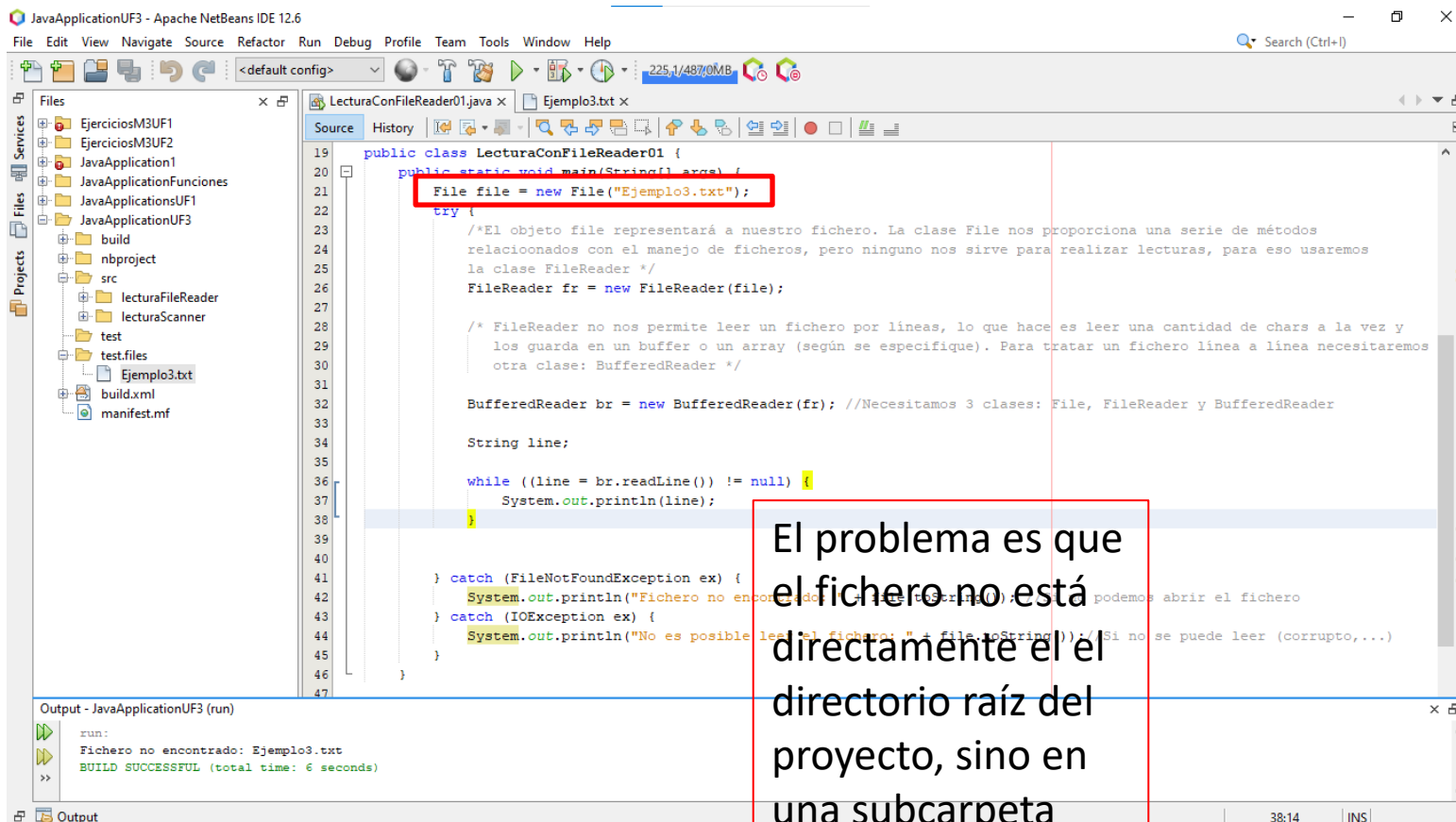


Creamos el fichero Ejemplo3.txt en el directorio del proyecto

Leer ficheros Usando BufferedReader

```
public class LecturaConFileReader01 {  
    public static void main(String[] args) {  
        File file = new File("Ejemplo3.txt");  
        try {  
            /*El objeto file representará a nuestro fichero. La clase File nos proporciona una serie de métodos  
            relacionados con el manejo de ficheros, pero ninguno nos sirve para realizar lecturas, para eso usaremos  
            la clase FileReader */  
            FileReader fr = new FileReader(file);  
  
            /* FileReader no nos permite leer un fichero por líneas, lo que hace es leer una cantidad de chars a la vez y  
            los guarda en un buffer o un array (según se especifique). Para tratar un fichero línea a línea necesitaremos  
            otra clase: BufferedReader */  
  
            BufferedReader br = new BufferedReader(fr); //Necesitamos 3 clases: File, FileReader y BufferedReader  
  
            String line;  
  
            while ((line = br.readLine()) != null) {  
                System.out.println(line);  
            }  
  
        } catch (FileNotFoundException ex) {  
            System.out.println("Fichero no encontrado: " + file.toString()); //Si no podemos abrir el fichero  
        } catch (IOException ex) {  
            System.out.println("No es posible leer el fichero: " + file.toString()); //Si no se puede leer (corrupto,...)  
        }  
    }  
}  
  
import java.io.BufferedReader;  
import java.io.File;  
import java.io.FileNotFoundException;  
import java.io.FileReader;  
import java.io.IOException;
```

Leer ficheros Usando BufferedReader



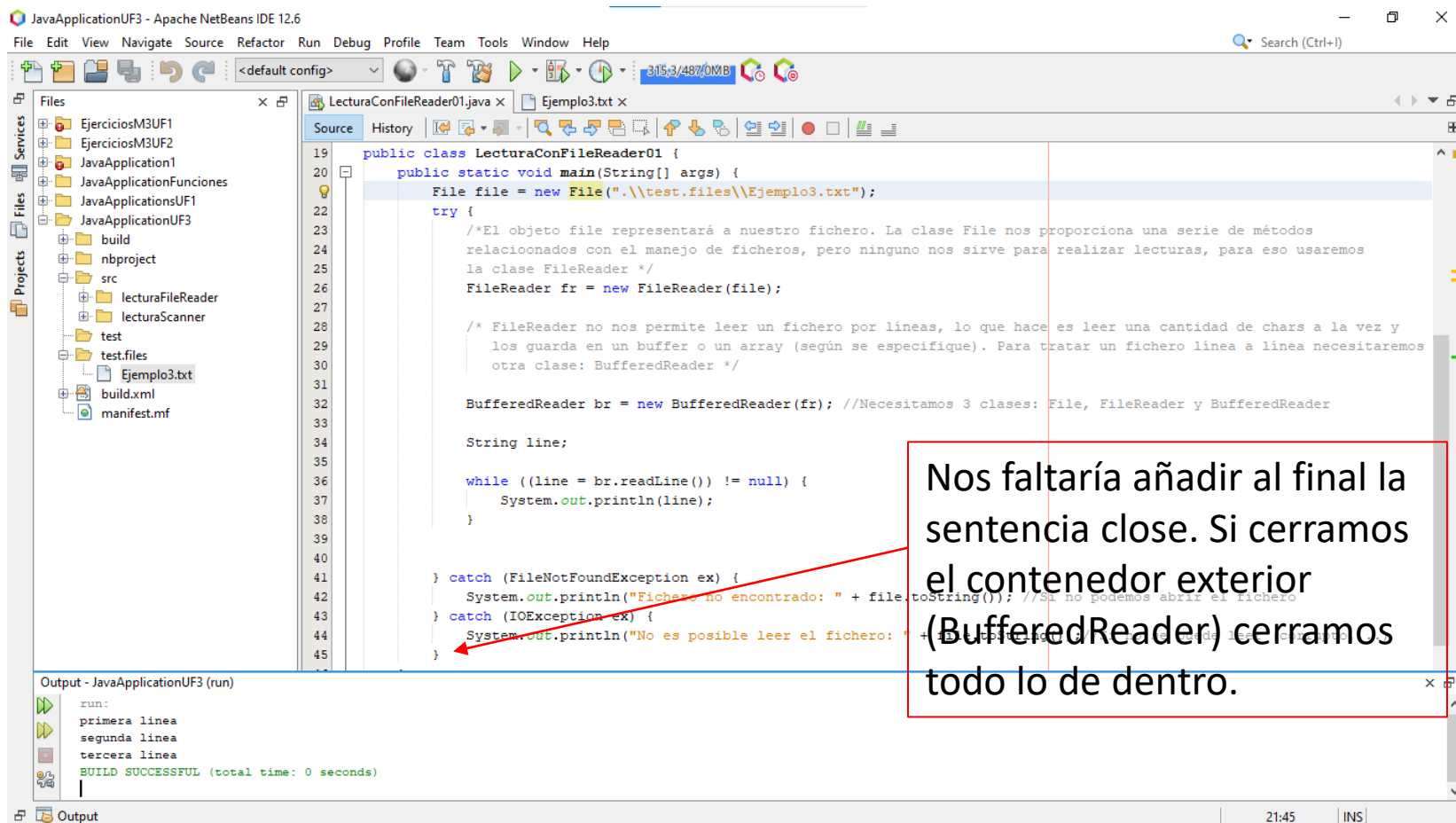
```
19 public class LecturaConFileReader01 {
20     public static void main(String[] args) {
21         File file = new File("Ejemplo3.txt");
22         try {
23             /*El objeto file representará a nuestro fichero. La clase File nos proporciona una serie de métodos
24             relacionados con el manejo de ficheros, pero ninguno nos sirve para realizar lecturas, para eso usaremos
25             la clase FileReader */
26             FileReader fr = new FileReader(file);
27
28             /* FileReader no nos permite leer un fichero por líneas, lo que hace es leer una cantidad de chars a la vez y
29             los guarda en un buffer o un array (según se especifique). Para tratar un fichero línea a línea necesitaremos
30             otra clase: BufferedReader */
31
32             BufferedReader br = new BufferedReader(fr); //Necesitamos 3 clases: File, FileReader y BufferedReader
33
34             String line;
35
36             while ((line = br.readLine()) != null) {
37                 System.out.println(line);
38             }
39
40         } catch (FileNotFoundException ex) {
41             System.out.println("Fichero no encontrado: " + ex.getMessage());
42         } catch (IOException ex) {
43             System.out.println("No es posible leer el fichero: " + file.toString());
44         }
45     }
46 }
```

Output - JavaApplicationUF3 (run)

```
run:
Fichero no encontrado: Ejemplo3.txt
BUILD SUCCESSFUL (total time: 6 seconds)
```

El problema es que el fichero no está directamente en el directorio raíz del proyecto, sino en una subcarpeta

Leer ficheros Usando BufferedReader



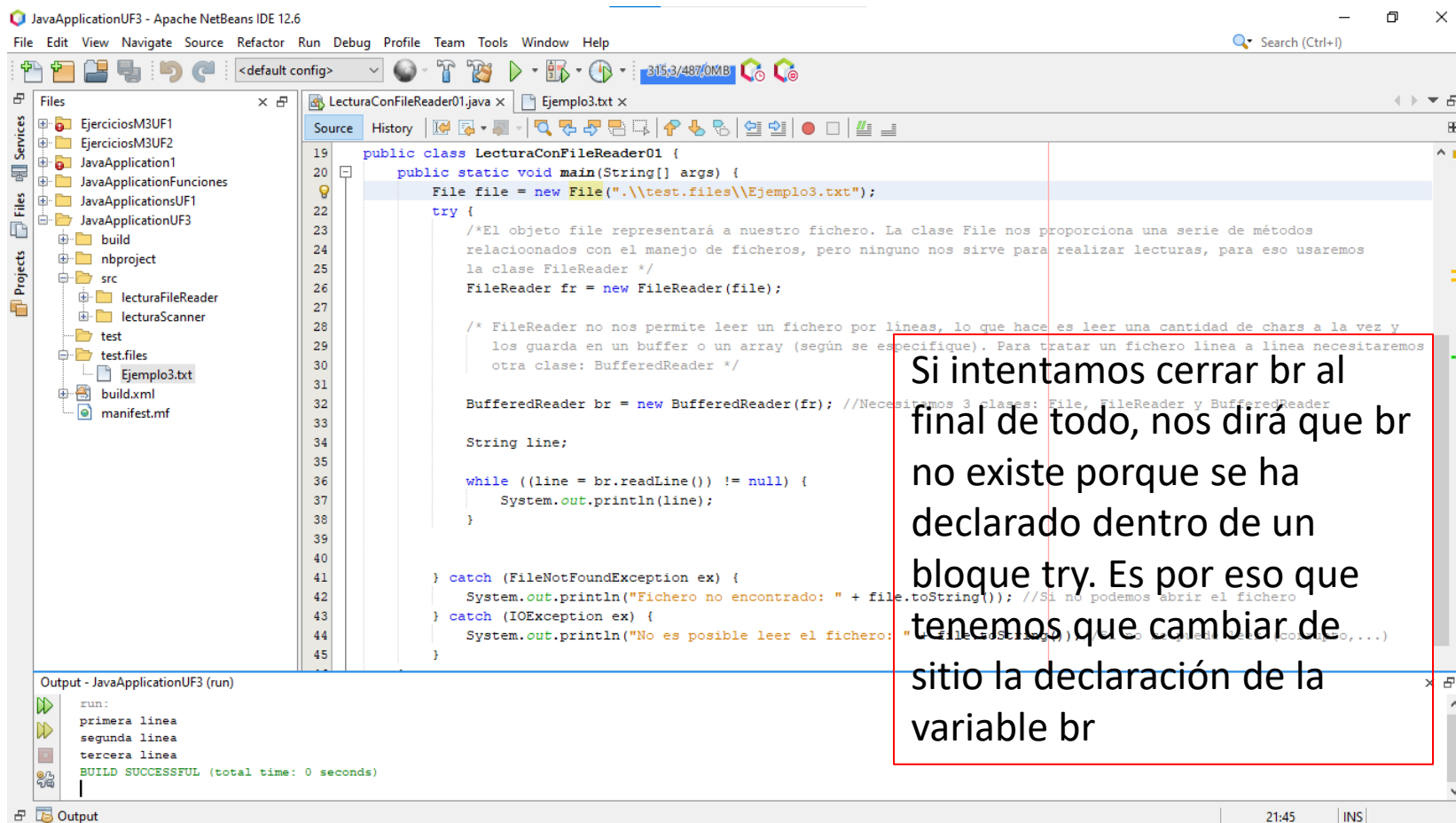
```
19 public class LecturaConFileReader01 {
20     public static void main(String[] args) {
21         File file = new File(".\\test.files\\Ejemplo3.txt");
22         try {
23             /*El objeto file representará a nuestro fichero. La clase File nos proporciona una serie de métodos
24             relacionados con el manejo de ficheros, pero ninguno nos sirve para realizar lecturas, para eso usaremos
25             la clase FileReader */
26             FileReader fr = new FileReader(file);
27
28             /* FileReader no nos permite leer un fichero por líneas, lo que hace es leer una cantidad de chars a la vez y
29             los guarda en un buffer o un array (según se especifique). Para tratar un fichero línea a línea necesitaremos
30             otra clase: BufferedReader */
31
32             BufferedReader br = new BufferedReader(fr); //Necesitamos 3 clases: File, FileReader y BufferedReader
33
34             String line;
35
36             while ((line = br.readLine()) != null) {
37                 System.out.println(line);
38             }
39
40         } catch (FileNotFoundException ex) {
41             System.out.println("Fichero no encontrado: " + file.toString()); //Si no podemos abrir el fichero
42         } catch (IOException ex) {
43             System.out.println("No es posible leer el fichero: " + ex.getMessage());
44         }
45     }
46 }
```

Nos faltaría añadir al final la sentencia close. Si cerramos el contenedor exterior (BufferedReader) cerramos todo lo de dentro.

Output - JavaApplicationUF3 (run)

```
run:
primera linea
segunda linea
tercera linea
BUILD SUCCESSFUL (total time: 0 seconds)
```

Leer ficheros Usando BufferedReader



JavaApplicationUF3 - Apache NetBeans IDE 12.6

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (Ctrl+I)

Files

- EjerciciosM3UF1
- EjerciciosM3UF2
- JavaApplication1
- JavaApplicationFunciones
- JavaApplicationsUF1
- JavaApplicationUF3
 - build
 - nbproject
 - src
 - lecturaFileReader
 - lecturaScanner
 - test
 - test.files
 - Ejemplo3.txt
 - build.xml
 - manifest.mf

Source

```
19 public class LecturaConFileReader01 {
20     public static void main(String[] args) {
21         File file = new File(".\\test.files\\Ejemplo3.txt");
22         try {
23             /*El objeto file representará a nuestro fichero. La clase File nos proporciona una serie de métodos
24             relacionados con el manejo de ficheros, pero ninguno nos sirve para realizar lecturas, para eso usaremos
25             la clase FileReader */
26             FileReader fr = new FileReader(file);
27
28             /* FileReader no nos permite leer un fichero por líneas, lo que hace es leer una cantidad de chars a la vez y
29             los guarda en un buffer o un array (según se especifique). Para tratar un fichero línea a línea necesitaremos
30             otra clase: BufferedReader */
31
32             BufferedReader br = new BufferedReader(fr); //Necesitamos 3 clases: File, FileReader y BufferedReader
33
34             String line;
35
36             while ((line = br.readLine()) != null) {
37                 System.out.println(line);
38             }
39
40
41         } catch (FileNotFoundException ex) {
42             System.out.println("Fichero no encontrado: " + file.toString()); //Si no podemos abrir el fichero
43         } catch (IOException ex) {
44             System.out.println("No es posible leer el fichero: " + file.toString()); //Si no podemos leer el fichero
45         }
46     }
47 }
```

Output - JavaApplicationUF3 (run)

```
run:
primera linea
segunda linea
tercera linea
BUILD SUCCESSFUL (total time: 0 seconds)
```

Output

21:45 INS

Si intentamos cerrar br al final de todo, nos dirá que br no existe porque se ha declarado dentro de un bloque try. Es por eso que tenemos que cambiar de sitio la declaración de la variable br

Leer ficheros Usando BufferedReader

```
LecturaConFileReader01.java x Ejemplo3.txt x
Source History
23 try {
24     /*El objeto file representará a nuestro fichero. La clase File nos proporciona una serie de métodos
25     relacionados con el manejo de ficheros, pero ninguno nos sirve para realizar lecturas, para eso usaremos
26     la clase FileReader */
27     FileReader fr = new FileReader(file);
28
29     /* FileReader no nos permite leer un fichero por líneas, lo que hace es leer una cantidad de chars a la vez y
30     los guarda en un buffer o un array (según se especifique). Para tratar un fichero línea a línea necesitaremos
31     otra clase: BufferedReader */
32     br = new BufferedReader(fr); //Necesitamos 3 clases: File, FileReader y BufferedReader
33     String line;
34     while ((line = br.readLine()) != null) {
35         System.out.println(line);
36     }
37 } catch (FileNotFoundException ex) {
38     System.out.println("Fichero no encontrado: " + file.toString()); //Si no podemos abrir el fichero
39 } catch (IOException ex) {
40     System.out.println("No es posible leer el fichero: " + file.toString()); //Si no se puede leer (corrupto,...)
41 }
42
43 finally{ //El código contenido en finally se ejecutará siempre (tanto si hay errores como si no)
44     try{
45         br.close();
46     } catch (IOException e){
47         System.out.println("No es posible cerrar el fichero: " + file.toString());
48     }
49     catch (NullPointerException ex){
50         /*La excepción NullPointerException sólo se daría si falla algo antes de que se haga br = new BufferedeReader
51         Si pasa esto br se queda sin valor (a null) por lo que al intentar hacer br.close dará NullPointerException
52         Si se da esto es porque probablemente el fichero nunca se ha abierto. En cualquier caso no estamos obligados
53         a dar siempre un mensaje informativo del error. en este caso no lo hacemos.*/
```

Leer ficheros Usando BufferedReader

- Normalmente nos irá bien tratar los ficheros por líneas, pero se nos podría ocurrir guardarnos todo el contenido del fichero en un String usando la operación de concatenar, o el operador '+', pero no sería una buena idea.
- Cada vez que creamos un String, se genera un objeto invariable, es decir, su contenido no se puede cambiar. Cuando usamos el operador '+' para añadir algo, lo que estamos haciendo es crear un nuevo String con el nuevo valor.
- Usar muchas veces el operador '+' sobre Strings muy largos es muy ineficiente (consume mucha memoria)
- Si la eficiencia es un factor a tener en cuenta en nuestro programa, debermos usar StringBuilder que nos permite modificar su contenido sin crear nuevas instancias (nuevos objetos StringBuilder).

```
StringBuilder objectOfStringBuilder = new StringBuilder("Carmen ");  
objectOfStringBuilder.append("Quintas");
```

Leer ficheros Usando BufferedReader

- La gestión de los errores es bastante engorrosa, pero no siempre tenemos que gestionarlos.
- Si las excepciones se producen dentro de un método/función probablemente lo que tendremos que hacer es propagar el error a las capas superiores, es decir, hacer un throw de las posibles excepciones que se puedan dar, para que estas sean tratadas desde donde se está llamando al método.

```
public static void leerFichero(String fichero)  
throws FileNotFoundException, IOException {  
    //1-abrir fichero:  
        File file =new File (fichero);  
        FileReader fr=new FileReader(file);  
        BufferedReader br=new BufferedReader(fr);  
    //2-Leer fichero  
        String line;  
        while( (line = br.readLine()) !=null ) {  
            System.out.println(line);  
        } //fin de while  
    //3-Cerrar fichero  
        fr.close;  
}
```

Otras formas de leer ficheros:

- **Leer el fichero entero usando Scanner (sin bucles)**

```
public class ReadingEntireFileWithoutLoop
{
    public static void main(String[] args)
        throws FileNotFoundException
    {
        File file = new File("C:\\Users\\quintasc\\Desktop\\test.txt");
        Scanner sc = new Scanner(file);

        // we just need to use \\Z as delimiter
        sc.useDelimiter("\\\\Z");

        System.out.println(sc.next());
    }
}
```


Otras formas de leer ficheros:

- **Leer el fichero entero en una lista**

```
// Java program to illustrate reading data from file
// using nio.File
import java.util.*;
import java.nio.charset.StandardCharsets;
import java.nio.file.*;
import java.io.*;
public class ReadFileIntoList
{
    public static List<String> readFileInList(String fileName)
    {
        List<String> lines = Collections.emptyList();
        try{
            lines =
                Files.readAllLines(Paths.get(fileName), StandardCharsets.UTF_8);
        }
        catch (IOException e){
            // do something
            e.printStackTrace();
        }
        return lines;
    }
}
```

Otras formas de leer ficheros:

- **Leer el fichero entero en una lista**

```
public static void main(String[] args)
{
    List l = readFileInList("C:\\Users\\pankaj\\quintasc\\test.java");

    Iterator<String> itr = l.iterator();
    while (itr.hasNext())
        System.out.println(itr.next());
}
```

Otras formas de leer ficheros:

- **Leer el fichero entero como un String**

```
package io;

import java.nio.file.*;;

public class ReadTextAsString {

    public static String readFileAsString(String fileName) throws Exception
    {
        String data = "";
        data = new String(Files.readAllBytes(Paths.get(fileName)));
        return data;
    }

    public static void main(String[] args) throws Exception
    {
        String data = readFileAsString("C:\\Users\\quintasc\\Desktop\\test.java");
        System.out.println(data);
    }
}
```

Escribir ficheros Usando BufferedWriter

- Cuando usamos BufferedWriter debemos trabajar también con otra clase: FileWriter
- La diferencia entre FileWriter y BufferedWriter es que aunque con las 2 clases proporcionan métodos para escribir:
 - FileWriter lo hace carácter a carácter y BufferedWriter permite escribir un salto de línea.
 - Después de escribir los caracteres almacenados en BufferedWriter, antes de cerrar el fichero, es recomendable usar el método flush() que lo que hace es forzar la escritura de los bytes almacenados en el buffer en la salida
 - BufferedWriter es más óptimo.

Escribir ficheros Usando BufferedWriter

```
7 import java.io.BufferedWriter;
8 import java.io.File;
9 import java.io.FileWriter;
10 import java.io.IOException;
11
12 /**
13  *
14  * @author user
15  */
16 public class EscrituraConFileWriter {
17
18     public static void main(String[] args) {
19         File file = new File(".\\test.files\\Ejemplo4.txt");
20
21         try (BufferedWriter bw = new BufferedWriter(new FileWriter(file))) {
22             bw.write("This is line one\n"); //Podemos añadir un salto de línea usando la secuencia de escape "\n"
23             bw.write("This is line two");
24             bw.newLine(); //Podemos añadir un salto de línea con newLine()
25             bw.write("Last line.");
26             bw.flush(); // Vacía los bytes almacenados en el buffer y fuerza su escritura en la salida prevista (fichero)
27         } catch (IOException e) {
28             System.out.println("Unable to write file " + file.toString());
29         }
30     }
31 }
```

Ejemplo4.txt x | EscrituraConFileWriter.java x |

Source History

```
1 This is line one
2 This is line two
3 Last line.
```

- Si el fichero no existe lo crea
- Si el fichero existe lo sobrescribe

Escribir ficheros Usando BufferedWriter

```
7 import java.io.BufferedWriter;
8 import java.io.File;
9 import java.io.FileWriter;
10 import java.io.IOException;
11
12 /**
13  *
14  * @author user
15  */
16 public class EscrituraConFileWriter {
17
18     public static void main(String[] args) {
19         File file = new File(".\\test.files\\Ejemplo4.txt");
20
21         try (BufferedWriter bw = new BufferedWriter(new FileWriter(file,true))) {
22             bw.write("This is line one\n"); //Podemos añadir un salto de línea usando la secuencia de escape "\n"
23             bw.write("This is line two");
24             bw.newLine(); //Podemos añadir un salto de línea con newLine()
25             bw.write("Last line.");
26             bw.flush(); // Vacía los bytes almacenados en el buffer y fuerza su escritura en la salida prevista (fichero)
27         } catch (IOException e) {
28             System.out.println("Unable to write file " + file.toString());
29         }
30     }
31 }
```

Ejemplo4.txt x EscrituraConFileWriter.java x

Source History

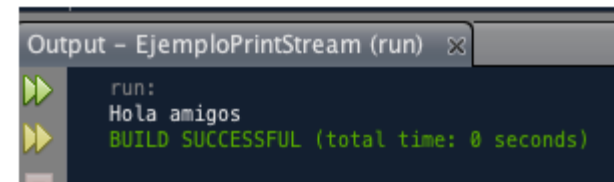
```
1 This is line one
2 This is line two
3 Last line.This is line one
4 This is line two
5 Last line.
```

- Si el fichero no existe lo crea
- Si el fichero existe añade el nuevo contenido por el final

Otras formas de escribir ficheros

- **Escribir un fichero usando PrintStream**

```
public class EjemploPrintStream {  
  
    public static void main(String[] args) throws FileNotFoundException {  
  
        PrintStream ejemplo1 = new PrintStream(System.out);  
        ejemplo1.println("Hola amigos");  
        ejemplo1.close();  
    }  
}
```

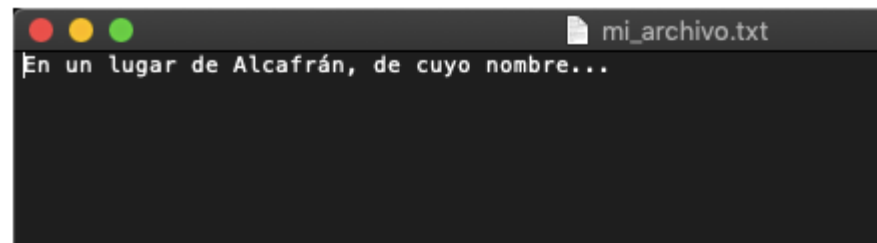


Resultado

Otras formas de escribir ficheros

- **Escribir un fichero usando PrintStream**

```
public class EjemploPrintStream {  
  
    public static void main(String[] args) throws FileNotFoundException {  
  
        try (PrintStream ejemplo2 = new PrintStream("/Users/carlosr/mi_archivo.txt")) {  
            ejemplo2.println("En un lugar de Alcafrán, de cuyo nombre...");  
            ejemplo2.close();  
        }  
    }  
}
```



Archivo creado

Otras operaciones con ficheros

- Crear una carpeta

```
public static void crearCarpeta(String nombre){  
  
    String path=System.getProperty("user.dir");  
    String separador=File.separator;  
    String rutaCarpeta= path+ separador+nombre;  
    File carpeta=new File(rutaCarpeta);  
    if(!carpeta.exists()){  
        carpeta.mkdir();  
    }  
}
```

Retorna el separador entre carpetas propio del sistema operativo en el que se está trabajando

Otras operaciones con ficheros

- **Borrar un fichero**

```
private static void borrarFichero(String nombreFichero)
{
    File fichero = new File(nombreFichero);
    fichero.delete();
}
```

Otras operaciones con ficheros

- Listar archivos

```
private static String showFiles(String rutaCarpeta)
throws IOException {

    File carpetaFormacion = new File(rutaCarpeta);
    String[] listaDocumentos = carpetaFormacion.list();
    //obtenemos el listado de todos los documentos dentro de la
    for(int i =0; i< listaDocumentos.length; i++){
        System.out.println(i+"-"+listaDocumentos[i]);
    }
    System.out.print("Seleccione el Numero de Fichero:");
    String textoTeclado= br.readLine();
    int numeroDocumento =Integer.parseInt(textoTeclado);
    String rutaDocumento = rutaCarpeta +
    File.separator+listaDocumentos[numeroDocumento];
    return rutaDocumento;
}
```