CLASES ASOCIADAS A LAS OPERACIONES DE GESTIÓN DE FICHEROS. LA CLASE FILE

La clase `File` desciende directamente de la clase `Object`. La clase `File` no se utiliza para transferir datos entre la aplicación y el disco, sino para obtener información sobre los ficheros y directorios de éste e incluso para la creación y eliminación de los mismos. La clase `File` representa un fichero (o directorio) del sistema de archivos, tiene múltiples métodos que nos van a permitir realizar todo tipo de operaciones con los ficheros.

Constructores:

`File(String ruta, String nombre)`: Crea un objeto de la clase que es un fichero. Se encuentra en la ruta que se indica en el primer argumento y cuyo nombre se pasa como segundo argumento.

```
a) File f = new File("directorio", "XXXXX.txt");
b) String directorio ="C:\\AccesoDatos\\Ejercicios\\UD01";
      File f = new File(directorio, "ejercicio01.txt");
```

`File(File ruta, String nombre)`: Crea un objeto de la clase que es un fichero. El `path` se indica en el objeto `File` que se pasa en el primer argumento y cuyo nombre se pasa como cadena en el segundo argumento. En este caso el directorio indicado en ruta deberá existir.

```
a) File f = new File(new File ("directorio"), "XXXXX.txt");
b) File dire = new File("directorio");
      File f = new File(dire, "ejercicio01.txt");
```

`File(String rutaAbsoluta)`: Crea un objeto de la clase que es un fichero. La ruta o nombre se pasa como argumento.

```
      File f = new File("C:\\directorio\\XXXXX.txt");
```

La creación de un objeto `File` no implica que exista el fichero o el directorio indicado en la ruta. Si éste no existe no se provoca una excepción, aunque tampoco será creado de forma implícita.

```
package ejemplos02ClaseFile;
import java.io.File;
import java.io.IOException;

/*
* Ejemplo que crea un objeto File con una ruta y un fichero
inexistentes
*/
public class EjemploFile01 {

      public static void main(String[] args)  {
            File f = new
File("C:\\Users\\34655\\Documents\\EjemplosFicheros1\\XXXXX111.txt");
```

```
            File f1 = new
File("C:\\Users\\34655\\Documents\\EjemplosFicheros1"+File.separator+"
XXX222.txt");

            System.out.println("Fin del programa EjemploFile01");

      }
}
```

Para crear el fichero utilizamos el método ***createNewFile()***, la ruta tiene que existir, en caso contrario salta una excepción.

```
package ejemplos02ClaseFile;

import java.io.File;
import java.io.IOException;

public class EjemploFile01B {
      public static void main(String[] args)  {
            File f = new
File("C:\\Users\\34655\\Documents\\EjemplosFicheros\\XXXXX111.txt");
            File f1 = new
File("C:\\Users\\34655\\Documents\\EjemplosFicheros"+File.separator+"X
XX222.txt");
            try {
                  f.createNewFile();
                  f1.createNewFile();
                  System.out.println(File.separator);

            } catch (IOException e) {
                  e.printStackTrace();
            }
            System.out.println("Fin del programa");
      }
}
```

El siguiente ejemplo muestra la lista de ficheros en el directorio actual. Se utiliza el método **list()** que devuelve un array de Strings con los nombres de los ficheros y directorios contenidos en el directorio asociado al objeto **File**. Para indicar que estamos en el directorio actual creamos un objeto **File** y le pasamos el parámetro **“.”**;

```
package ejemplos02ClaseFile;
/*
 * Ejemplo que muestra la lista de ficheros del directorio actual
 */
import java.io.File;
public class EjemploFile02 {
      public static void main(final String[] args) {
            System.out.println("Lista de ficheros del directorio
actual:");
```

```
            //muestra la lista del directorio actual
            File f = new File(".");
            // muestra la lista del directorio indicado en el path
      //    File f = new
File("C:\\Users\\34655\\Documents\\EjemplosFicheros");

            // muestra la lista del directorio indicado en el path
utilizando la constante separator
            String separator = File.separator;
      //    File f = new File("C:"+separator+"Users"+separator+
      "34655"+separator+"Documents"+separator+"EjemplosFicheros");
            String[] archivos = f.list();
            for(int i=0; i< archivos.length; i++){
                  System.out.println(archivos[i]);
            }
      }
}
```

La siguiente declaración mostraría la lista de ficheros del directorio c:\db:

```
File f = new File("c:\\data\\db");
```

La siguiente declaración mostrará la lista de ficheros del directorio introducido desde la línea de comandos al ejecutar el programa.

```
String directorio = args[0];
System.out.println("Ficheros en el directorio " +directorio);
File f = new File(directorio);
```

Métodos

- Para crear físicamente el fichero o directorio indicado en el constructor de `File` utilizaremos los siguientes métodos

`boolean createNewFile().` Crea el fichero cuyo nombre se indica en el constructor. Devuelve **`true`** si se ha podido crear el fichero, mientras que el resultado será **`false`** si ya existe y por tanto no ha sido creado.

Por ejemplo, si el fichero anterior XXXXX.txt no existe se podría ejecutar la siguiente instrucción para crearlo:

```
f.createNewFile();
```

La llamada a este método puede provocar una excepción **`IOException`** que habrá que capturar.

Si un objeto `File` hace referencia a un fichero inexistente y no se crea de forma explícita con `createNewFile()`, el fichero se creará de forma implícita cuando se vaya a utilizar el objeto `Writer` o `OutputStream` para realizar una operación de escritura sobre el fichero.

`boolean mkdir().` Crea el directorio cuyo nombre se indica en el constructor. Devuelve **`true`** si se ha podido crear el directorio, mientras que el resultado será **`false`** si ya existe y por tanto no ha sido creado.

```
f.mkdir();
```

EJEMPLOS:

```
- File carpeta= new File("C:\\Ejemplos")
carpeta.mkdir();
```

Crea una carpeta ejemplos dentro de la unidad C.

```
- File archivo= new File(carpeta,"Ejemplo1.txt")
archivo.createNewFile();
```

Crea un fichero llamado ejemplo1 dentro de c:\Ejemplos.

```
- File archivo2 = new File("C:\\Ejemplos","Ejemplo2.txt")
archivo2.createNewFile();
```

Crea un fichero llamado ejemplo2 dentro de c:\Ejemplos.

```
- File archivo3= new File("C:\\Ejemplos\\Ejemplo2.txt")
archivo3.createNewFile();
```

Crea un fichero llamado ejemplo2 dentro de c:\Ejemplos.

```
- File carpeta= new File("Ejemplos")
```

Al no poner ruta crea la carpeta donde se está ejecutando la aplicación.

```
package ejemplos02ClaseFile;
import java.io.File;
/*
 * Ejemplo que crea la carpeta EjemplosFicheros1 en
C:\\Users\\34655\\Documents
 */
public class EjemploFile03B {
	public static void main(final String[] args) {
		//creamos el objeto File
		File f = new
File("C:\\Users\\34655\\Documents\\EjemplosFicheros1");
		//ejecutamos el método mkdir para crear la carpeta
		if(f.mkdir())
			System.out.println("Directorio creado OK");
		else
			System.out.println("La carpeta no se ha podido crear.
Ya existe");
	}
}
```

`boolean mkdirs()`: Crear una carpeta. Devuelve **`true`** si se puede crear. En el caso de que la carpeta este dentro de un path absoluto que no exista en el sistema, se crea de manera completa.

Ejemplo: Con mkdir C:\Nueva\Ejemplo solo crearía Nueva y con mkdirs nueva y ejemplo.

```
package ejemplos02ClaseFile;

import java.io.File;
/*
```

```
 * Ejemplo que crea las carpetas EjemplosFicheros3\\Nueva en
C:\\Users\\34655\\Documents
 */
public class EjemploFile03C {
	public static void main(final String[] args) {
		//creamos el objeto File
		File f = new
File("C:\\Users\\34655\\Documents\\EjemplosFicheros3\\Nueva");
		//ejecutamos el método mkdirs para crear la carpeta
		if(f.mkdirs())
			System.out.println("Directorio creado OK");
		else
			System.out.println("La carpeta no se ha podido crear.
Ya existe");
	}
}
```

- Para obtener información del objeto creado podemos utilizar los métodos de la clase `File`:

`boolean canRead()`: **`True`** si se puede hacer una operación de lectura en un fichero o carpeta.

`boolean canWrite()`: **`True`** si se puede hacer una operación de escritura en un fichero o carpeta.

`canNewFile()`: Permite crear en el sistema un fichero o carpeta que inicialmente está vacío.

`canRead()`: Devuelve **`true`** si el fichero se puede leer.

`canWrite()`: Devuelve **`true`** si el fichero se puede escribir.

`boolean delete()`: Se utiliza para eliminar un fichero o carpeta. Devuelve **`true`** si se puede eliminar.

`deleteOnExit()`: Permite eliminar un fichero o carpeta al finalizar la aplicación.

`boolean equals()`: **`True`** si son iguales los ficheros que se comparan.

`boolean exists()`: **`True`** si el fichero o carpeta existe.

`getAbsolutePath()`: Devuelve una cadena indicando la ruta completa donde se encuentra el archivo o carpeta.

`getPath()`: Devuelve una cadena indicando la ruta relativa donde se encuentra el archivo o carpeta.

`getName()`: Indica el nombre del fichero o carpeta.

`getParent()`: Devuelve el nombre del directorio padre, o **`null`** si no existe

`isDirectory()`: Devuelve **`true`** cuando el objeto `File` es una carpeta.

`isFile()`: Devuelve **`true`** cuando el objeto `File` es un fichero.

`isHidden()`: Devuelve **`true`** cuando el objeto File está oculto.

length(): Devuelve la longitud del fichero en bytes.

String []list(): Devuelve una array de tipo **String** con el nombre de todos los archivos y carpetas que haya dentro de otra carpeta.

String []list(filtro): Devuelve una array de tipo **String** con el nombre de todos los archivos y carpetas con un determinado filtro (por ejemplo "*.java").

renameTo(): Permite cambiar el nombre de un fichero o carpeta por el del objeto que se pasa como argumento. Devuelve **true** si lo ha cambiado.

setReadOnly(): Para establecer un fichero o carpeta de solo lectura.

Constantes:

pathSeparator: Devuelve un **String** indicando el carácter que se utiliza dentro del sistema para separar cada una de las cadenas de la variable path. En Windows es ";".

pathSeparatorChar: Idem del anterior pero lo devuelve como carácter.

separator: Devuelve un **String** indicando el carácter que se utiliza dentro del sistema para establecer la ruta a los recursos. En Windows es "\".

separatorChar: Idem del anterior pero lo devuelve como carácter.

Ejemplo que muestra los ficheros pdf de la carpeta C:\\Users\\34655\\Documents\\Ciclos Formativos\\CSDAMultiplataforma

```
package ejemplos02ClaseFile;

import java.io.File;
/*
 * En el siguiente ejemplo mostramos todos los ficheros pdf que tenemos en la  * carpeta CSDAMultiplataforma dentro de CiclosFormativos en MI disco C.
 */
public class EjemploFile03 {
	public static void main(final String[] args) {
		//separador alamacena el tipo de separador utilizado en la plataforma, en windows \
		String separador= File.separator;

		//En carpeta almaceno el path de la carpeta que quiero mirar sus subelementos
		File carpeta= new File("C:" +separador+ "Users" +separador +"34655"+separador+"Documents"+separador+"Ciclos Formativos"+
		separador+"CSDAMultiplataforma");

	//	File carpeta= new File("C:\\Users\\34655\\Documents\\Ciclos Formativos\\CSDAMultiplataforma");

		System.out.println("Carpeta: " +carpeta);
```

```
            //En elementos almaceno la matriz con todos los nombres de
los archivos
            // y carpetas dentro_de la carpeta que le indiqué
            String[] elementos = carpeta.list();

            System.out.println("elementos" +elementos);

            //Mostramos el número de carpetas dentro de la que le dije
            System.out.println("Los ficheros .pdf son:\n");
            for (int i=0; i<elementos.length; i++){
                  if (elementos[i].endsWith("pdf") ||
elementos[i].endsWith("pdf")){
                        System.out.println("\t"
+carpeta.getAbsolutePath() +separador +elementos[i]);
                  } // fin del if
            } // fin de for
      } // fin del main
} // fin de la clase
```

Ejemplo que muestra información del fichero **prueba.pdf**

```
package ejemplos02ClaseFile;

import java.io.File;

//Ejemplo que muestra información del fichero prueba.pdf

public class EjemploFile04 {
      public static void main(final String[] args) {
            System.out.println("Información sobre el fichero\n");
            String separador= File.separator;
            //En carpeta almaceno el path de la carpeta que quiero mirar
sus subelementos
            File f= new
File("C:\\Users\\34655\\Documents\\EjemplosFicheros\\prueba.pdf");
            System.out.println("Nombre del fichero: " +f.getName());
            System.out.println("Ruta : " +f.getPath());
            System.out.println("Ruta absoluta : " +f.getAbsolutePath());
            System.out.println("Ruta anterior : " +f.getParent());
            System.out.println("Se puede escribir : " +f.canWrite());
            System.out.println("Se puede leer : " +f.canRead());
            System.out.println("Tamaño : " +f.length());
            System.out.println("Es un directorio : " +f.isDirectory());
            System.out.println("Es un fichero : " +f.isFile());
      }
}
```

Visualiza la siguiente información del fichero:

```
<terminated> EjemploFile04 (1) [Java Application] C:\Program Files\Java\jre1.8.0_261\bin\javaw.exe (27 sept. 2020 18:54:40)
Información sobre el fichero

Nombre del fichero: prueba.pdf
Ruta : C:\Users\34655\Documents\EjemplosFicheros\prueba.pdf
Ruta absoluta : C:\Users\34655\Documents\EjemplosFicheros\prueba.pdf
Ruta anterior : C:\Users\34655\Documents\EjemplosFicheros
Se puede escribir : true
Se puede leer : true
Tamaño : 257235
Es un directorio : false
Es un fichero : true
```

El siguiente ejemplo crea un directorio (de nombre NUEVODIR) en el directorio actual, a continuación crea dos ficheros vacíos en dicho directorio y uno de ellos lo renombre. En este caso para crear los ficheros se definen 2 parámetros en el objeto *File*: `File(File directorio, String nombreFichero)`, en el primero indicamos el directorio donde se creará el fichero y en el segundo indicamos el nombre del fichero:

```
package ejemplos02ClaseFile;

import java.io.File;
import java.io.IOException;

public class EjemploFile05 {

	public static void main(String[] args) {
		File directorio = new File("NUEVODIR"); // directorio que se crea a partir del actual
		File fichero1 = new File(directorio, "Fichero1.txt");
		File fichero2 = new File(directorio, "Fichero2.txt");

		directorio.mkdir(); // crea el directorio
		try{
			if(fichero1.createNewFile())
				System.out.println("Fichero1 creado correctamente");
			else
				System.out.println("Fichero1 no se ha podido crear");
			if(fichero2.createNewFile())
				System.out.println("Fichero2 creado correctamente");
			else
				System.out.println("Fichero2 no se ha podido crear");
		}catch(IOException ioe){
			ioe.printStackTrace();
		}
		System.out.println("Listado Inicial ficheros creados");
		String[] archivos = directorio.list();
		for(int i=0; i< archivos.length; i++){
```

```
                System.out.println(archivos[i]);
            }

            if(fichero1.renameTo(new File(directorio, "nuevoFichero")))
// renombtramos el fichero
                System.out.println("Fichero1 cambiado de nombre");
            else
                System.out.println("No se ha podido cambiar de
nombre");

      System.out.println("=========================================");
            System.out.println("Listado después de renombrar fichero
1");
            archivos = directorio.list();
            for(int i=0; i< archivos.length; i++){
                System.out.println(archivos[i]);
            }
            try{
                File fichero3 = new File("NUEVODIR\\Fichero3.txt");
                fichero3.createNewFile(); // crea Fichero3 en NUEVODIR
            }catch(IOException ioe){
                ioe.printStackTrace();
            }

            if(fichero2.delete())
                System.out.println("Fichero2 borrado correctamente");
            else
                System.out.println("Fichero2 no se ha podido borrar");


      System.out.println("=========================================");
            System.out.println("Listado después de borrar fichero 2");
            archivos = directorio.list();
            for(int i=0; i< archivos.length; i++){
                System.out.println(archivos[i]);
            }
      }
}
```

Como el método ***createNewFile()*** puede lanzar la excepción ***IOException***, por ello se utiliza el bloque ***try-catch***.

Para borrar un fichero o un directorio usamos el método delete(), para borrar un directorio deberá estar vacío.

```
      if(fichero2.delete())
                System.out.println("Fichero2 borrado correctamente");
            else
                System.out.println("Fichero2 no se ha podido borrar");
```