TEMA 8. FICHEROS

1. Introducción a los Ficheros

Un fichero o archivo es un conjunto de bits almacenado en un dispositivo permanente, por ejemplo un disco duro. Existen dos tipos de ficheros

- **Ficheros de texto**: Almacenan caracteres alfanuméricos en un formato estándar (ASCII, UNICODE, UTF-8, etc).
- **Ficheros binarios**: Almacenan cualquier secuencia de dígitos binarios. Tienen la ventaja de que ocupan menos espacio en disco.

Para la gestión de ficheros se utiliza la clase **File**. Esta clase proporciona un conjunto de utilidades relacionadas con los ficheros que nos proporcionan información acerca de los mismos, y permiten borrar renombrar, crear o ver el contenido de un directorio, etc .

El constructor de la clase File que utilizaremos será el siguiente:

File(String directorioyfichero):

- En plataformas Linux:

```
File fichero=new File("/directorio/fichero.txt")
```

- En plataformas Windows :

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

Algunos de los métodos de la clase File son:

- isFile(): devueltre true si el objeto File corresponde a un fichero normal
- isDirectory(): devuelve true si el objeto File corresponde a un directorio
- getName(): devuelve el nombre del fichero o directorio
- getPath(): devuelve la ruta relativa
- getAbsolutePath(): devuelve la ruta absoluta
- canRead(): devuelve true si se puede leer del fichero
- canWrite(): devuelve true si se puede escribir en el fichero
- length(): devuelve el tamaño del fichero en bytes
- createNewFile(): crea un nuevo fichero, vacío, asociado a File si y solo si no existe un fichero con ese nombre
- mkdir(): crea un directorio con el nombre indicado en la creación del objeto File.
- delete(): borra el fichero o directorio asociado a File
- String[] list(): Devuelve un array de Strjngs con los nombres de los ficheros o directorios que cuelgan del directorio actual
- File[] listFiles(): Devuleve el array de objetos File con los fichero o directorios que cuelgan del directorio actual

Operaciones sobre ficheros

Independientemente del tipo de ficheros las operaciones básicas que se realizan sobre ellos son

- Apertura del fichero. Para que el programa pueda operar con un fichero la primera operación que tiene que realizar es la apertura del mismo. Se le puede indicar el nombre del fichero o el correspondiente objeto File.
- Cierre del fichero. Cuando no se vaya a utilizar un fichero debe cerrarse.
- Lectura de datos del fichero: Consiste en transferir información de fichero a memoria principal. En este caso el fichero se dice que el fichero se utiliza en modo lectura.
- Escritura de datos del fichero: Consiste en transferir en volcar información de memoria al fichero. En este caso el fichero se dice que se utiliza en modo escritura.

Tipos de acceso a ficheros

- Acceso secuencial:
  Para leer un determinado dato en el fichero hay que leer todos los datos anteriores a ese.

- Acceso directo:
  Es posible acceder a un determinado dato del fichero sin haber leído los anteriores a él.

Flujos o streams

Un flujo (stream) es la clase Java que se utiliza para abstraer la comunicación de información entre una fuente y un destino. Dicha información puede estar en un disco duro, en un dispositivo, en algún lugar de la red, etc. Cualquier programa que tenga que obtener información de una fuente necesita abrir un stream y leer de él (el stream en este caso será de entrada). Si necesita enviar información abrirá un stream y escribirá la información en él (se dice que en este caso el stream es de salida)

Se definen dos tipos de flujo

- Flujo de caracteres (16 bits): Realizan operaciones de E/S de caracteres, orientado a **ficheros de texto**. Viene gobernado por las clases **Reader** y **Writer**
- Flujo de bytes (8 bits): Realizan operaciones de E/S de bytes, orientado a **ficheros binarios**. Viene gobernado por las clases **InputStream** y **OutputStream**

Excepciones

Cuando trabajemos con ficheros debemos tratar las excepciones de la jerarquía IOException. Entre ellas está la FileNotFoundException.

2. Ficheros de texto

Para trabajar con los ficheros de texto usaremos las clases **FileReader** para leer caracteres de un fichero de texto y **FileWriter** para escribir caracteres en un fichero de texto.

Podemos trabajar sólo con estas dos clases pero lo normal es además de ellas usar otras clases que servirán como filtro. Las clases filtro sirven para aportar más funcionalidad a la lectura/escritura pudiendo escribir y leer con formato.

Existen varios tipos de filtros. Nosotros usaremos el filtro **BufferedReader** para **FileReader** y los filtros **PrintWriter** para **FileWriter**.

FICHEROS DE TEXTO		
	FLUJO	**FILTRO**
Lectura	`FileReader`	`BufferedReader`
Escritura	`FileWriter`	`PrintWriter`

Para crear un objeto **FileReader** o **FileWriter** hay que pasarle en el constructor, el objeto File o bien el nombre del fichero.

LECTURA

Si abrimos un fichero para lectura, el fichero debe existir. Si no existe se lanzará la excepción `FileNotFoundException.`

```
FileReader flujoLectura=new FileReader(nombreFichero);
```

ESCRITURA

Si el fichero no existe y se abre en escritura se creará.
Si el fichero existe y lo abrimos en escritura todo lo que tenía almacenado se borrará.

```
FileWriter flujoEscritura=new FileWriter (nombreFichero);
```

ESCRITURA EN MODO AÑADIR (APPEND)

Si el fichero no existe y se abre en escritura “modo append” se creará.
Si el fichero existe y lo abrimos en escritura “modo append” se escribirá al final del fichero.

```
FileWriter flujoEscritura=new FileWriter ( nombreFichero, true);
```

2.1. Leer de un fichero línea a línea

```
private static void imprimirFicheroPorLineas(String nombreFichero) {
	String linea;
	try
	{
		FileReader flujoLectura=new FileReader(nombreFichero);
		BufferedReader filtroLectura=new BufferedReader(flujoLectura);

		linea=filtroLectura.readLine();
		while ( linea!=null)
		{
			System.out.println(linea);
			linea=filtroLectura.readLine();
		}
		filtroLectura.close();
		flujoLectura.close();
	}
	catch(FileNotFoundException e){
		System.out.println("No existe el fichero " + nombreFichero);
	}
	catch(IOException e){
		System.out.println( e.getMessage());
	}

}
```

2.2. Escribir en un fichero por líneas

```
private static void escribirEnFicheroPorLineas(String nombre) {
	String cadena;
	try {
		FileWriter flujoEscritura=new FileWriter(nombre);
		PrintWriter filtroEscritura=new PrintWriter(flujoEscritura);
		for (int i = 1; i <= NUMERO_LINEAS; i++) {
			System.out.println("Introduce texto:");
			cadena=teclado.nextLine();
			filtroEscritura.println(cadena);
		}
		filtroEscritura.close();
		flujoEscritura.close();
	} catch (IOException e) {
		System.out.println(e.getMessage());
	}
}
```