

Programador Certificado

JAVA 2

Curso práctico

3ª EDICIÓN



Antonio J. Martín Sierra



Pueden descargarse archivos con todas las prácticas desarrolladas en el libro.

Alfaomega  Ra-Ma®

ACCESO AL DISCO

En capítulos anteriores hemos visto cómo los programas Java pueden almacenar diferentes tipos de información en arrays y colecciones. Sin embargo, estos datos permanecen en la memoria RAM del ordenador, lo que significa que en el momento en que la aplicación finalice, toda esta información desaparecerá.

Durante este capítulo veremos cómo almacenar información de forma permanente en ficheros de disco, así como su posterior recuperación.

Java dispone de un gran número de clases incluidas en el paquete `java.io` específicamente creadas para tratar con ficheros de distintos tipos y manipular información en diferentes formatos.

Durante este capítulo estudiaremos algunas de estas clases, centrando nuestro análisis en el almacenamiento y recuperación de información en disco en forma de tipos básicos, texto y objetos.

6.1 INFORMACIÓN SOBRE FICHEROS Y DIRECTORIOS. LA CLASE `FILE`

Los ficheros o archivos y directorios del disco se representan de forma lógica en las aplicaciones Java como objetos de la clase `File`.

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.

6.1.1 Creación de un objeto File

Existen diversas formas de crear un objeto File en función de cómo se indique la localización del fichero o directorio que va a representar. Por ejemplo, el constructor *File(String path)* permite construir un objeto File a partir de su dirección absoluta o relativa al directorio actual:

```
File f = new File ("datos.txt");  
File f2 = new File("misubdirectorio");
```

La creación del objeto no implica que exista el fichero o directorio indicado en la ruta. Si éste no existe, las anteriores instrucciones no provocarán ninguna excepción, aunque tampoco será creado de forma implícita.

Para crear físicamente el fichero o directorio indicado en el constructor de File habrá que recurrir a los siguientes métodos de la clase:

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

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

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

La llamada a *createNewFile()* puede provocar una excepción *IOException* que habrá que capturar.

Si un objeto File hace referencia a un fichero no existente y no es creado de forma explícita invocando a *createNewFile()*, la creación del mismo se llevará a cabo implícitamente cuando se vaya a utilizar el objeto File para construir un objeto *Writer* u *OutputStream*, a fin de realizar una operación de escritura sobre el fichero.

- boolean **mkdir()**. Crea el directorio cuyo nombre ha sido especificado en el constructor. Si el directorio no existía y se ha podido crear, el método devolverá *true*, de lo contrario el resultado será *false*.

Por ejemplo, para crear físicamente el subdirectorio referenciado por el objeto `File f2` sería:

```
f2.mkdir();
```

La clase `File` también proporciona un constructor que permite crear un objeto `File` asociado a un fichero, indicando a través de otro objeto `File` el directorio donde se encuentra dicho fichero:

File (File dir, String nombre_fichero)

En este caso, el directorio especificado en *dir* debe existir, de lo contrario se producirá una excepción al intentar utilizar el fichero para realizar cualquier operación sobre el mismo.

El siguiente bloque de instrucciones crearía un fichero llamado `info.txt` en el interior del directorio *datos*:

```
File f=new File("datos");
f.mkdir(); //si datos no existiese y esta instrucción
          //no estuviera, la llamada a createNewFile()
          //provocaría una excepción.
File fichero=new File(f,"info.txt");
fichero.createNewFile();
```

6.1.2 Información sobre un fichero/directorio

Una vez creado el objeto `File` asociado al fichero o directorio, podemos obtener información del mismo aplicándole los siguientes métodos de la clase `File`:

- boolean **canRead()**. Indica si se puede o no leer el fichero.
- boolean **canWrite()**. Indica si se puede o no escribir en el fichero.
- boolean **exists()**. Indica si existe o no el fichero o directorio indicado en la ruta.
- boolean **isFile()**. Indica si el objeto `File` hace referencia o no a un fichero.
- boolean **isDirectory()**. Indica si el objeto `File` hace referencia o no a un directorio.

- String **getName()**. Devuelve el nombre del fichero sin el path.
- String **getAbsolutePath()**. Devuelve el path absoluto completo.

6.1.3 Eliminación y renombrado

Para eliminar físicamente un fichero o directorio la clase File proporciona el método *delete()* con el siguiente formato:

- boolean **delete()**. Elimina el fichero o directorio especificado por el objeto. Si el elemento ha podido ser eliminado el método devolverá *true*, si no devolverá *false*. **Un directorio sólo podrá ser eliminado si está vacío**; si no lo está, la llamada a *delete()* sobre el objeto File devolverá *false*, aunque no provocará ninguna excepción.

Si lo que queremos es renombrar un fichero o directorio existente, utilizaríamos el siguiente método:

- boolean **renameTo(File nuevo)**. Renombra el fichero o directorio, asignándole el nombre del objeto File especificado. La llamada a este método devolverá *true* si se ha podido renombrar el elemento, mientras que el resultado será *false* si esto no ha sido posible. En el caso de un directorio, **no es necesario que esté vacío para poderlo renombrar**.

El siguiente ejemplo renombra un fichero existente:

```
File f=new File("fichero.txt");
f.createNewFile(); //para poderlo renombrar
                  //debe existir
File f2=new File("nuevo_nombre.txt");
f.renameTo(f2); //renombra "fichero.txt" a
               //"nuevo_nombre.txt"
```

6.2 LECTURA DE UN FICHERO DE TEXTO

Muchas de las operaciones realizadas con ficheros en un programa tienen que ver con la lectura de ficheros de texto.

Para recuperar cadenas de caracteres de un fichero, el paquete *java.io* proporciona dos clases: **FileReader** y **BufferedReader**. A continuación veremos

cómo se utilizan estas clases, examinando los dos pasos que hay que seguir para la recuperación de cadenas de caracteres de un fichero.

6.2.1 Creación de un objeto `FileReader`

Para poder recuperar información de un fichero de texto, es necesario primeramente crear un objeto `FileReader` asociado al mismo. Un objeto `FileReader` representa un fichero de texto "abierto" para la lectura de datos. Este objeto es capaz de adaptar la información recuperada del fichero a las características de una aplicación Java, transformando los bytes almacenados en el mismo en caracteres unicode.

Se puede construir un objeto `FileReader` a partir de un objeto `File` existente o bien proporcionando directamente la ruta del fichero, a partir de los siguientes constructores:

`FileReader (String path)`

`FileReader (File fichero)`

La siguiente instrucción crearía un objeto `FileReader` que haga referencia al fichero `datos.txt`:

```
File f = new File("datos.txt");  
FileReader fr = new FileReader(f);
```

La clase `FileReader` proporciona el método `Read()` para la lectura de la información almacenada en un fichero, resultando su uso bastante engorroso ya que los caracteres son recuperados como tipo `byte`, debiendo ser posteriormente convertidos a `String`. Es por ello que resulta más cómodo recurrir a la clase `BufferedReader` para realizar esta operación, utilizando el objeto `FileReader` como puente para crear un objeto de este tipo.

6.2.2 Creación de un objeto `BufferedReader`

Una vez creado el objeto `FileReader`, el segundo paso consiste en crear un `BufferedReader`, para lo cual utilizaremos el constructor:

`BufferedReader (Reader entrada)`

Ya vimos en el capítulo 3 cómo utilizar esta clase para leer cadenas de caracteres desde el teclado. Para leer de un fichero el procedimiento es el mismo,

sólo que en este caso el objeto Reader que hay que proporcionar al constructor será el objeto FileReader asociado al fichero:

```
BufferedReader bf=new BufferedReader(fr);
```

Una vez creado el objeto, puede utilizarse el método *readLine()* para leer líneas de texto del fichero de forma similar a como se leían del teclado. En el caso de un fichero, y dado que éste puede estar formado por más de una línea, será necesario utilizar un bucle *while* para recuperar todas las líneas de texto del mismo de forma secuencial.

El siguiente programa muestra por pantalla el contenido completo de un fichero llamado datos.txt:

```
import java.io.*;
public class Lectura {
    public static void main(String[] args)
        throws IOException{
        File f = new File("datos.txt");
        if(f.exists()){
            FileReader fr=new FileReader("datos.txt");
            BufferedReader bf=new BufferedReader(fr);
            String cad;
            while((cad=bf.readLine())!=null){
                System.out.println(cad);
            }
        }
        else{
            System.out.println("El fichero "+
                               "no existe");
        }
    }
}
```

Según se desprende del programa anterior, el método *readLine()* apunta a la siguiente línea de texto después de recuperar la línea actual. Cuando no existan más líneas para leer, la llamada a *readLine()* devolverá *null*.

Si lo que se quiere es leer caracter a caracter en vez de línea a línea, deberíamos utilizar el método *read()* en lugar de *readLine()*. El método *read()*