9.- Manejo de ficheros y carpetas

1. La clase File.

Un objeto de la clase File es una representación abstracta de la ruta y el nombre de un ficheros o carpetas. Cuando creamos en Java un objeto de la clase File en representación de un fichero o carpeta concretos, no creamos el fichero al que se representa. Es decir, el objeto File representa al fichero o carpeta de disco, pero no es el fichero o carpeta de disco.

La clase File dispone de métodos que permiten realizar determinadas operaciones sobre el fichero o carpeta al que representa. Podríamos, por ejemplo, crear un objeto de tipo File que represente a *c:\datos\libros.txt* y, a través de ese objeto File, realizar consultas relativas al fichero libros.txt, como su tamaño, permisos, etc, o realizar operaciones sobre él: borrarlo, renombrarlo, …

2. Constructores:

La clase File tiene varios constructores, que permiten referirse, de varias formas, al fichero o carpeta que queremos representar:

public File (String ruta)	Crea el objeto File a partir del nombre del fichero o carpeta indicado. El nombre incluirá la ruta que hay que seguir para llegar hasta el fichero o carpeta.
public File (String ruta, String nombre)	Permite indicar de forma separada la ruta del fichero y su nombre
public File (File ruta, String nombre)	Permite indicar de forma separada la ruta del fichero y su nombre. En este caso la ruta está representada por otro objeto File.
public File (URI uri)	Crea el objeto File a partir de un objeto URI (Uniform Resource Identifier). Un URI permite representar un elemento siguiendo una sintaxis concreta, un estándar.

3. Métodos.

Aquí exponemos algunos métodos interesantes. Hay otros que puedes consultar en la documentación de Java

Relacionados con el nombre del fichero	
String getName()	Devuelve el nombre del fichero o directorio al que representa el objeto. (Solo el nombre, sin la ruta)
String getPath()	Devuelve la ruta completa del fichero o directorio (incluyendo el nombre). La ruta obtenida es dependiente del sistema, es decir, contendrá el carácter de separación de directorios que esté establecido por defecto. Este separador está definido en la constante pública public static final String **separator**
String getAbsolutePath()	Devuelve la ruta absoluta del fichero o directorio.
String getParent()	Devuelve la ruta del directorio en que se encuentra el fichero o directorio representado. Devuelve null si no hay directorio padre.

Para hacer comprobaciones	
boolean exists() Boolean canWrite() Boolean canRead() Boolean isFile() Boolean isDirectory()	Permiten averiguar, respectivamente, si el fichero existe, si se puede escribir en el, si se puede leer de él, si se trata de un fichero o si se trata de un directorio
Obtener información de un fichero	
long length	Devuelve el tamaño en bytes del fichero. El **resultado es indefinido si se consulta sobre un directorio** o una unidad.
long lastModified	Devuelve la fecha de la última modificación del fichero. Devuelve el **número de milisegundos** transcurridos desde el 1 de enero de 1970
Para trabajar con directorios	
Boolean mkdir()	Crea el directorio al cual representa el objeto File.
Boolean mkdirs()	Crea el directorio al cual representa el objeto File, incluyendo todos aquellos que sean necesarios y no existan.
String[] list()	Devuelve un array de Strings con los nombres de los ficheros y directorios que contiene el directorio al que representa el objeto File.
String[] list(FileNameFilter filtro)	Devuelve un array de Strings con los nombres de los ficheros y directorios que contiene el directorio al que representa el objeto File y que cumplen con determinado filtro.
public File[] listFiles()	Devuelve un array de objetos File que representan a los ficheros y carpetas contenidos en el directorio al que se refiere el objeto File.
Para hacer cambios	
Boolean renameTo(File nuevoNombre)	Permite renombrar un fichero. Hay que tener en cuenta que la operación puede fracasar por muchas razones, y que será dependiente del sistema: Que no se pueda mover el fichero de un lugar a otro, que ya exista un fichero que coincide con el nuevo, etc. El método devuelve true solo si la operación se ha realizado con éxito. Existe un método move en la clase Files para mover ficheros de una forma independiente del sistema.
Boolean delete()	Elimina el fichero o la carpeta a la que representa el objeto File. Si se trata de una carpeta tendrá que estar vacía. Devuelve true si la operación tiene éxito.
Boolean createNewFile()	Crea un fichero vacío. Devuelve true si la operación se realiza con éxito.
File createTempFile(String prefijo, String sufijo)	Crea un fichero vacío en la carpeta de ficheros temporales. El nombre llevará el prefijo y sufijo indicados. Devuelve el objeto File que representa al nuevo fichero.

4. Ejemplo: Mostrar información del contenido de una carpeta.

```
package _02Ejemplos;
import java.io.*;
importjava.util.*;

publicclass _01InformacionCarpeta {
    publicstaticvoid main(String[] args) {

        Scanner tec = new Scanner(System.in);
        System.out.println("Introduce ruta absoluta de una carpeta");
        StringnombreCarpeta = tec.nextLine();
        //Creamosobjeto File pararepresentar a lacarpeta
        File car = new File(nombreCarpeta);
        //Comprobamossiexiste
        if (car.exists()){
            //¿Esunacarpeta?
            if(car.isDirectory()){
                if(car.canRead()) System.out.println("Lecturapermitida");
                else System.out.println("Lectura no permitida");

                if(car.canWrite()) System.out.println("Escriturapermitida");
                else System.out.println("Escritura no permitida");

                if(car.isHidden()) System.out.println("Carpetaoculta");
                else System.out.println("Carpeta visible");

                System.out.println("---- Contenido de la carpeta ----");
                File[] contenido = car.listFiles();
                for(File f: contenido){
                    System.out.println(f.getName());
                }
            }
            else System.out.println(car.getAbsolutePath() + "No esunacarpeta" );
        } else System.out.println("No existe la carpeta" + car.getAbsolutePath());
    }
}
```