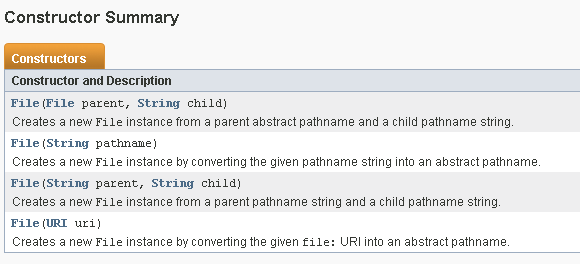
Clase File

La **clase File** representa un fichero o directorio de nuestro sistema de archivos. También nos es útil ya que puede pasarse como parámetro para las principales clases que manejan ficheros, como **FileReader, FileInputStream, FileWriter, etc.**

Podemos pasarle un String con la ruta del fichero, también hay otros constructores. Estos son:

[](http://www.discoduroderoer.es/wp-content/uploads/2013/07/Constructores-file.png)

Esta clase está en el paquete **java.io.**

A diferencia de otras clases que manejan ficheros, en la clase **File** no es necesario controlar excepciones en la mayoría de los métodos. Si hay alguna excepción a manejar lo indicaremos.

Estos son los métodos más conocidos:

| **NOMBRE** | **DESCRIPCIÓN** | **PARÁMETROS** | **DATO DEVUELTO** |
| --- | --- | --- | --- |
| exists | Indica si existe o no el fichero. | Ninguno. | boolean |
| isDirectory | Indica si el objeto File es un directorio. | Ninguno. | boolean |
| isFile | Indica si el objeto File es un fichero. | Ninguno. | boolean |
| isHidden | Indica si el objeto File está oculto. | Ninguno. | boolean |
| getAbsolutePath | Devuelve una cadena con la ruta absoluta del fichero o directorio. | Ninguno. | String |
| canRead | Indica si se puede leer. | Ninguno. | boolean |
| canWrite | Indica si se puede escribir. | Ninguno. | boolean |
| canExecute | Indica si se puede ejecutar. | Ninguno. | boolean |
| getName | Devuelve una cadena con el nombre del fichero o directorio. | Ninguno. | String |
| getParent | Devuelve una cadena con el directorio padre. | Ninguno. | String |
| listFiles | Devuelve un array de File con los directorios y ficheros hijos. | Ninguno. | Array de File |
| list | Devuelve un array de String con los directorios hijos. Solo funciona con directorios. | Ninguno. | Array de String |
| mkdir | Permite crear el directorio en la ruta indicada. Solo se creara si no existe. | Ninguno. | boolean |
| mkdirs | Permite crear el directorio en la ruta indicada, también crea los directorios intermedios. Solo se creara si no existe. | Ninguno. | boolean |
| createNewFile | Permite crear el fichero en la ruta indicada. Solo se creará si no existe. Debemos controlar la excepcion con IOException. | Ninguno. | boolean |

Os dejo un ejemplo, usando los métodos:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102 | import java.io.File;  import java.io.IOException;  public class EjemploFileApp {        public static void main(String[] args) throws IOException {            //Creamos objetos File          File fichero=new File("D:\\fich\_binario.ddr");          File fichero2=new File("D:\\fichero.txt");          File directorio=new File("D:\\prueba");          File directorio2=new File("D:\\directorio");                    //Creo los ficheros y directorios                  fichero.createNewFile();                  fichero2.createNewFile();                  directorio.mkdir();                  directorio2.mkdir();                    //Indica si existen los archivos          System.out.println("Existencia: ");          System.out.println("Fichero "+fichero.exists());          System.out.println("Directorio "+directorio.exists());            System.out.println("");            //Indica si son directorios          System.out.println("¿Son directorios?: ");          System.out.println("Fichero "+fichero.isDirectory());          System.out.println("Directorio "+directorio.isDirectory());            System.out.println("");            //Indica si son ficheros          System.out.println("¿Son ficheros?: ");          System.out.println("Fichero "+fichero.isFile());          System.out.println("Directorio "+directorio.isFile());            System.out.println("");            //Indica la ruta absoluta del fichero o directorio          System.out.println("Ruta absoluta: ");          System.out.println("Fichero "+fichero.getAbsolutePath());          System.out.println("Directorio "+directorio.getAbsolutePath());            System.out.println("");            //Indica si se puede leer          System.out.println("¿Se pueden leer?:");          System.out.println("Fichero "+fichero.canRead());          System.out.println("Directorio "+directorio.canRead());            System.out.println("");            //Indica si se puede escribir          System.out.println("¿Se pueden escribir?:");          System.out.println("Fichero "+fichero.canWrite());          System.out.println("Directorio "+directorio.canWrite());            System.out.println("");            //Indica si se puede ejecutar          System.out.println("¿Se pueden ejecutar?:");          System.out.println("Fichero "+fichero.canExecute());          System.out.println("Directorio "+directorio.canExecute());            System.out.println("");            //Indica el nombre sin rutas          System.out.println("Nombres sin rutas: ");          System.out.println("Fichero "+fichero.getName());          System.out.println("Directorio "+directorio.getName());            System.out.println("");            //Indica el nombre del directorio padre          System.out.println("Nombre del directorio padre: ");          System.out.println("Fichero "+fichero.getParent());          System.out.println("Directorio "+directorio.getParent());            System.out.println("");            //Guarda en un array de File los directorios hijos, solo con directorios          System.out.println("Nombre de los objetos File dentro de un array");          File lista[]=directorio.listFiles();          for(int i=0;i<lista.length;i++){              System.out.println(lista[i]);          }            System.out.println("");            //Guarda en un array de String los directorios hijos, solo con directorios          System.out.println("Nombre de los objetos String dentro de un array");          String listaString[]=directorio.list();          for(int i=0;i<listaString.length;i++){              System.out.println(listaString[i]);          }            System.out.println("");          }  } |

La clase File tiene un par de constantes muy útiles, que nos servirán si usamos nuestra aplicación en varios sistemas operativos. Nos permitirá mostrar el separador entre ficheros entre rutas de cada sistema operativo, por ejemplo, para Windows, se separan las rutas con; y los directorios con \. Los métodos son **pathSeparator** y **separator.**

Se invocan con el nombre de la clase, por ejemplo, **File.separator.**