Classe FILE – Gestió de Fitxers

La peça més bàsica per a poder operar amb arxius, independentment del seu tipus, en un programa Java és la **classe File**. S'ha d'importar la llibreria:

```
import java.io.File;
```

- S'utilitza per a gestionar el sistma d'arxius en Java.
- **NO representa el contingut dels arxius, sinó una ruta a l'arxiu.** ÉS una classe que fa referència al la RUTA o localització de l'arxiu.
- Com es tracta d'una ruta pot representar tant **arxius, com carpetes o directoris.**
- Al utilitzar una classe per representar rutes, **aconseguim independència sobre el Sistema Operatiu**, i com este represente les rutes.
- Per tant, quan creem un **objecte** de la classe FILE, estarà vinculat a la ruta d'un arxiu durant tota la seua existència, i no podrà ser reutilitzat, per tant**, si necessitem treballar amb noves rutes de fitxers, haurem de crear nous objectes de la classe FILE.**

CONSTRUCTORS

Per crear un objecte de la classe FILE, podem utilitzar 3 constructors diferents:

- **File (String *directori_i_fitxer*) :** indicant amb un únic paràmetre tant el directori com l'arxiu, es a dir l'arxiu amb la seua ruta.

```
File fitxer1 = new File("/home/usuari/Exemples/fitxer1.txt");
File fitxer1 = new File ("C:\\Exemples\\fitxer1.txt");
```

Per refereciar un directori s'utilitza la mateixa tècnica:

```
File directori = new File ("/home/usuari/Exemples");
```

Les rutes anteriors són **absolutes**, ja que comencen des de l'arrel. Si no la posem absoluta (començant per /) serà relativa i començarà en el directori actiu.

- **File (String *directori*, String *fitxer*) :** amb dos paràmetres, indicant el directori on està el fitxer (amb ruta), i el nom del fitxer (sense la ruta)

```
File fitxer2 = new File ( "/home/usuari/Exemples", "fitxer2.txt");
```

Farà referència a un arxiu amb el nom com el segon paràmetre col·locat en el directori referit en el primer paràmetre. Observa com el segon paràmetre podria ser també un directori, i per tant seria una referència a un subdirectori d'un directori referenciat en el primer paràmetre.

- **File (File *directori*, String *fitxer*) :** En este cas, el directori és un objecte FILE creat anteriorment.

```
File fitxer3 = new File (directori, "fitxer3.txt");
```

En els exemples anteriors hem posat directament les rutes. Però els programadors de Java han de fer un esforç per independitzar les aplicacions implementades de les plataformes on s'executaran.
Per tant, haurem d'anar amb cura, fent servir tècniques que eviten escriure les rutes directament en el codi. Per això encara que ara a el principi utilitzarem dels 3 constructors, en el futur hauríem d'utilitzar massivament el tercer, ja que com veieu la manera d'especificar la ruta de localització de el fitxer, és per mitjà d'un altre File.

La classe File encapsula pràcticament tota la funcionalitat necessària per gestionar un sistema d'arxius organitzat en arbre de directoris. És una gestió completa que inclou:

- Funcions de manipulació i consulta de la pròpia estructura jeràrquica (creació, eliminació, obtenció de la ubicació, etc. d'arxius o directoris)
- Funcions de manipulació i consulta de les característiques particulars dels elements (noms, mida o capacitat, etc.)
- Funcions de manipulació i consulta d'atributs específics de cada sistema operatiu, com ara els permisos d'escriptura, d'execució, atributs d'ocultació. Només funcionarà si el sistema operatiu amfitrió suporta també la funcionalitat d'aquests atributs.

Exemples

Exemple 1.- Programa que mostra la llista d'arxius i directoris del directori actual (utilitzem "." que val per a tots els sistemes) .
Per recórrer la llista utilitzem el mètode **list()** de la classe **File**

```
import java.io.File;

public class Exemple1 {
    public static void main(String[] args) {

        //Obri el directori actual '.'

        File f = new File(".");
        System.out.println("Llistat de fitxers i directoris del directori actual");
        System.out.println("----------------------------------------------------");

        //recorre la llista de fitxers
        for (String e : f.list()){
            System.out.println(e);
        }
    }
}
```

```
Llistat de fitxers i directoris del directori actual
----------------------------------------------------
.classpath
.project
.settings
bin
dades.txt
datos.txt
src
```

Exemple 2.- Si volem traure el contingut d'un directori que es demana per teclat:

```
import java.io.File;
import java.util.Scanner;

public class Exemple2
{
    public static void main(String[] args) {
        Scanner entrada = new Scanner(System.in);
        String directori;

        System.out.println("Introduce un directorio:");
        directori=entrada.nextLine();

        File f = new File(directori);
        System.out.println("Llistat de fitxers i directoris del directori: " + directori);
        System.out.println("------------------------------------------------------------");
        for (String e : f.list()) {
            System.out.println(e);
        }
    }
}
```

Mètodes de la Classe FILE

Per a obtindré el nom o la ruta	
String **getName()**	Torna el nom del fitxer o directori (relacionat amb la ruta).
String **getPath()**	Torna la ruta relativa
String **getAbsolutePath()**	Torna la ruta absoluta

Per accedir al directori pare o als subdirectoris (fitxers i directoris)	
String[] **list()**	Torna un array **d'Strings** amb tots els elements continguts en el **FILE** (noms d'arxius i directoris «dins» d'eixa ruta).
File[] **listFiles()**	Torna un array de **FILES** amb tots els elements continguts en el **FILE** (arxius i directoris «dins» d'eixa ruta).
String **getParent()**	Torna el nom (String) del directori pare. Si no existeix per ser l'arrel torna **NULL**
File **getParentFile()**	Torna el directori pare com un **FILE.** Si no existeix per ser l'arrel torna **NULL**

Per comprovar l'existència i característiques	
boolean **exists()**	Torna **true** si el fitxer o directori (relacionat amb la ruta) existeix
boolean **isDirectory()**	Torna **true** si el file relacionat amb la ruta és un directori
boolean **isFile()**	Torna **true** si el file relacionat amb la ruta és un arxiu
boolean **isHidden()**	Torna **true** si el file relacionat amb la ruta és un arxiu ocult
long **length()**	Torna el tamany en **bytes** del fitxer
long **lastModified()**	Torna la **data de modificació** de l'arxiu

Creació i eliminació	
boolean **delete()**	Esborra l'arxiu o directori. Si és un direcxtori, el directori ha d'estar buit per poder ser eliminat
boolean **mkdir()**	Crea un directori amb el nom indicat en la creació del FILE. Ha d'existir el directori pare.
boolean **renameTo(File** desti**)**	Canvia el nom de l'arxiu; «destí» és el nou camí o path per al fitxer reanomenat