CENTRE ESPECÍFIC
D´EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

UNIDAD 11
FICHEROS
EJERCICIOS (A)

PROGRAMACIÓN
CFGS DAW

Autor:
Lionel Tarazón Alcocer
lionel.tarazon@ceedcv.es

2019/2020

Licencia

UD11. FICHEROS

EJERCICIOS (A) - GESTIÓN DE FICHEROS

Para probar estos ejercicios utilizar el archivo “Documentos.zip”. Descárgalo del aula virtual y descomprímelo en la carpeta de cada proyecto que crees.

Ejercicio A1 - Mostrar información de ficheros

Implementa un programa que pida al usuario introducir por teclado una ruta del sistema de archivos (por ejemplo, “C:/Windows” o “Documentos”) y muestre información sobre dicha ruta (ver función más abajo). El proceso se repetirá una y otra vez hasta que el usuario introduzca una ruta vacía (tecla intro). Deberá manejar las posibles excepciones.

Necesitarás crear la función **void muestraInfoRuta(File ruta)** que dada una ruta de tipo File haga lo siguiente:

- Si es un archivo, mostrará por pantalla el nombre del archivo.
- Si es un directorio, mostrará por pantalla la lista de directorios y archivos que contiene (sus nombres). Deberá mostrar primero los directorios y luego los archivos.
- En cualquier caso, añade delante del nombre la etiqueta [*] o [A] para indicar si es un directorio o un archivo respectivamente.
- Si el path no existe lanzará un FileNotFoundException.

Ejercicio A2 - Mostrar información de ficheros (v2)

Partiendo de una copia del programa anterior, modifica la función **muestraInfoRuta**:

- En el caso de un directorio, mostrará la lista de directorios y archivos en orden alfabético. Es decir, primero los directorios en orden alfabético y luego los archivos en orden alfabético. Te será útil Arrays.sort().
- Añade un segundo argumento ‘boolean info’ que cuando sea ‘true’ mostrará, junto a la información de cada directorio o archivo, su tamaño en bytes y la fecha de la última modificación. Cuando ‘info’ sea ‘false’ mostrará la información como en el ejercicio anterior.

Ejercicio A3 - Renombrando directorios y ficheros

Implementa un programa que haga lo siguiente:

- Cambiar el nombre de la carpeta ‘Documentos’ a ‘DOCS’, el de la carpeta ‘Fotografias’ a ‘FOTOS’ y el de la carpeta ‘Libros’ a ‘LECTURAS’
- Cambiar el nombre de todos los archivos de las carpetas FOTOS y LECTURAS quitándole la extensión. Por ejemplo, ‘astronauta.jpg’ pasará a llamarse ‘astronauta’.

Ejercicio A4 - Creando (y moviendo) carpetas

Implementa un programa que cree, dentro de ‘Documentos’, dos nuevas carpetas: ‘Mis Cosas’ y ‘Alfabeto’. Mueve las carpetas ‘Fotografias’ y ‘Libros’ dentro de ‘Mis Cosas’. Luego crea dentro de ‘Alfabeto’ una carpeta por cada letra del alfabeto (en mayúsculas): ‘A’, ‘B’, ‘C’... ‘Z’. Te serán de ayuda los códigos numéricos ASCII: https://elcodigoascii.com.ar

Ejercicio A5 - Borrando archivos

Implementa un programa con una función **boolean borraTodo(File f)** que borre f: Si no existe lanzará una excepción. Si es un archivo lo borrará. Si es un directorio, borrará primero sus archivos y luego el propio directorio (recuerda que para poder borrar un directorio debe estar vacío). Devolverá 'true' si pudo borrar el 'File f' ('false' si no fué posible).

Prueba la función borrando las carpetas: 'Documentos/Fotografias', 'Documentos/Libros' y 'Documentos' (es decir, tres llamadas a la función, en ese orden).

***Super extra challenge**: Esta función, tal y como está definida, no borrará las subcarpetas que estén dentro de una carpeta (para ello habría que borrar primero el contenido de dichas subcarpetas). ¿Se te ocurre cómo podría hacerse? Inténtalo si te animas ;-)*

CASO PRÁCTICO A - MiniTerminal & MiniFileManager

Implementa un programa que funcione como una pequeña terminal Linux, con algunos comandos (simplificados) que permitan al usuario realizar distintas operaciones de gestión de archivos. Los comandos que el usuario podrá ejecutar son:

- **pwd**: Muestra cual es la carpeta actual.
- **cd <DIR>**: Cambia la carpeta actual a 'DIR'. Con .. cambia a la carpeta superior.
- **ls**: Muestra la lista de directorios y archivos de la carpeta actual (primero directorios, luego archivos, ambos ordenados alfabéticamente).
- **ll**: Como ls pero muestra también el tamaño y la fecha de última modificación.
- **mkdir <DIR>**: Crea el directorio 'DIR' en la carpeta actual.
- **rm <FILE>**: Borra 'FILE'. Si es una carpeta, borrará primero sus archivos y luego la carpeta. Si tiene subcarpetas, las dejará intactas y mostrará un aviso al usuario.
- **mv <FILE1> <FILE2>**: Mueve o renombra 'FILE1' a 'FILE2'.
- **help**: Muestra una breve ayuda con los comandos disponibles.
- **exit**: Termina el programa.

Clase MiniTerminal: Clase principal (con función main) que se encargará de interactuar con el usuario e interpretar los comandos (qué comando se pide, argumentos, etc.). Utilizará la segunda clase para realizar las operaciones de gestión de archivos. Manejará todas las posibles excepciones.

Clase MiniFileManager: Tendrá los atributos y métodos que necesites, para realizar las distintas operaciones relacionadas con la gestión de archivos. Necesitarás al menos un método por cada operación. Se lanzará una excepción si se produce un error o la operación solicitada no es posible. Algunos ejemplos que podrías implementar:

- **String getPWD()**: Devuelve una cadena de texto con la carpeta actual.
- **boolean changeDir(String dir)**: Cambia la carpeta actual a *dir*. Devuelve 'true' si fue posible.
- **void printList(boolean info)**: Muestra una lista con los directorios y archivos de la carpeta actual. Si info es 'true' mostrará también su tamaño y fecha de modificación.
- etc.