Actividad

Actividad AE01

Ubicación

Tema 1

Objetivos

- Utilizar la clase File para la gestión de ficheros y directorios.
- Conocer los principales métodos de la clase File de Java.
- Manejar la API de Java.
- Entender el concepto de *streams* de entrada y salida en Java.
- Familiarizarse con el manejo de archivos en Java.
- Construir interfaces para aplicaciones de escritorio en Java.
- Utilizar *streams* para guardar objetos.

Temporalización

La duración prevista para esta actividad es de seis sesiones lectivas.

Instrucciones

Se solicita desarrollar en Java una aplicación con interfaz gráfica (ventana principal y emergentes) para gestionar archivos de texto plano.

El diseño de la interfaz, clases y métodos es libre.

Se pide que la aplicación tenga las siguientes funcionalidades:

- Dado un directorio que el usuario elija o indique por teclado, listar todos los archivos que contenga, ya sea en el mismo directorio o en cualquier subdirectorio que contenga. Para cada elemento que se muestre, se debe indicar la ruta, el nombre del archivo, la extensión, el tamaño y la fecha/hora de la última modificación. Puedes presentar la información de dos maneras (se puntuarán de manera diferente - ver rúbrica).

    - Opción de presentación básica:

```
dir\subdir1\subdir2\fichero.txt (1,1 KB - 09/09/2024 15:00:05)
dir\subdir1\subdir2\fichero.docx (67,1 KB – 09/09/2024 15:01:05)
dir\subdir1\fichero.pdf (126,6 KB – 09/09/2024 15:02:05)
dir\fichero1.java (1,1 KB – 09/09/2024 15:03:05)
dir\fichero2.java (3,0 KB – 09/09/2024 15:04:05)
dir\fichero.java (100,5 KB – 09/09/2024 15:05:05)
dir\fichero.ppt (2527,9 KB – 09/09/2024 15:06:05)
```

- Opción de presentación avanzada (árbol de directorios/ficheros, valdría esta presentación o similar, siempre y cuando se vea claramente la estructura del árbol de directorios/ficheros):

```
dir
|-- \subdir1
|   |-- \subdir2
|   |   |-- fichero.txt (1,1 KB – 09/09/2024 15:00:05)
|   |   |-- fichero.docx (67,1 KB – 09/09/2024 15:01:05)
|   |-- fichero.pdf (126,6 KB – 09/09/2024 15:02:05)
|-- fichero1.java (1,1 KB – 09/09/2024 15:03:05)
|-- fichero2.java (3,0 KB – 09/09/2024 15:04:05)
|-- fichero (100,5 KB – 09/09/2024 15:05:05)
|-- fichero.pptx (2527,9 KB – 09/09/2024 15:06:05)
```

NOTA: si un directorio o subdirectorio no contiene archivos, no es necesario mostrarlo en la lista o en el árbol. Los archivos del directorio raíz pueden aparecer al principio o al final del árbol. El orden de aparición no es un requisito, solo la estructura.

- Dada una cadena de texto por parte del usuario, la aplicación debe buscar y decir cuántas coincidencias hay de la cadena indicada (exactamente tal y como la ha introducido el usuario) en cada uno de los archivos del directorio. Si algún archivo no es accesible como texto plano (por ejemplo, un archivo de Word), se asume que devolverá 0 coincidencias. Por ejemplo, siguiendo el ejemplo anterior:

```
dir
|-- /subdir1
|   |-- /subdir2
|   |   |-- fichero.txt (1 coincidencias)
|   |   |-- fichero.docx (0 coincidencias)
|   |-- fichero.pdf (0 coincidencias)
|-- fichero1.java (0 coincidencias)
|-- fichero2.java (25 coincidencias)
|-- fichero (0 coincidencias)
|-- fichero.pptx (0 coincidencias)
```

- Amplía la funcionalidad anterior. Deberás permitir que se busque la cadena de texto respetando mayúsculas/minúsculas o sin respetarlas. Además, deberás permitir que se busque la cadena respetando los acentos o sin respetarlos. Por último, se valorará positivamente que la aplicación pueda acceder al contenido de archivos PDF y mostrar las coincidencias (investiga cómo hacerlo).

- Añade una funcionalidad a la anterior que dé la opción de reemplazar la cadena encontrada por otra cadena. De manera similar al punto anterior, indica cuántos reemplazos se han producido en cada uno de los archivos del directorio, mencionando también aquellos archivos que no han sido accesibles. Para esta funcionalidad, puedes asumir que los archivos PDF no son accesibles. Debes tener en cuenta que el contenido del archivo debe ser el mismo que el original antes de hacer el reemplazo, excepto por los elementos reemplazados. En lugar de sobrescribir los archivos, puedes crear otros nuevos, modificando el nombre (por ejemplo: archivo original -> archivo1.txt, archivo nuevo con el texto reemplazado -> MOD_archivo1.txt).

Evaluación

La actividad es obligatoria y se puede realizar individualmente o en parejas. Si se realiza en parejas, ambos alumnos deben hacer la entrega e indicarlo claramente como comentario en la entrega.
Para la evaluación se tendrá en cuenta el funcionamiento del programa, la codificación adecuada y la documentación del mismo. También se solicitará a los alumnos que presenten y expliquen en clase la aplicación que han desarrollado.

El código y la documentación se deberán entregar en Florida Oberta (archivo .zip) y también estar disponible en el GitHub del alumno. También se debe entregar un vídeo donde se muestre en detalle la ejecución de las funcionalidades y se explique el código desarrollado. **IMPORTANTE**: no se corregirá la actividad si no hay vídeo.

Recursos

Material del módulo (Florida Oberta).

Rúbrica

Información del directorio	(0) No implementado.	(1) Muestra solo algunos ficheros y/o con información incompleta.	(2) Muestra todos los ficheros con la información completa.	(3) Muestra todos los ficheros con la información completa y utilizando un árbol de directorios y ficheros bien estructurado.
Buscar en ficheros	(0) No implementado.	(0.5) No busca en todos los ficheros y no encuentra todas las coincidencias	(1.25) No busca en todos los ficheros o no encuentra todas las coincidencias.	(2) Busca en todos los ficheros, encuentra las coincidencias y gestiona correctamente los no accesibles.
Buscar en ficheros (ampliación)	(0) No implementado.	(0.5) No busca en todos los ficheros o no encuentra todas las coincidencias.	(1) Busca en todos los ficheros accesibles y encuentra todas las coincidencias.	(1.5) Busca en todos los ficheros accesibles (incluyendo PDFs) y encuentra todas las coincidencias.
Reemplazar contenido	(0) No implementado.	(0.5) No reemplaza en todos los ficheros que debería y no reemplaza todas las coincidencias.	(1) No reemplaza en todos los ficheros que debería o no reemplaza todas las coincidencias.	(1.5) Reemplaza correctamente todas las coincidencias.
Interfaz gráfica		(0) No implementada.	(0.5) La interfaz no es fácil de utilizar y/o tiene carencias estéticas importantes.	(1) La interfaz es sencilla, fácil de utilizar y estéticamente correcta.
Documentación		(0) No generada o no se han seguidos las especificaciones.	(0.25) Faltan métodos por documentar y/o la descripción de clases y métodos no es adecuada.	(0.5) La documentación está completa y detallada. El vídeo es explicativo.
Calidad del código		(0) No se ha seguido la guía de estilo o las especificaciones.	(0.25) El código presenta algunos errores de estilo y/o algunos comentarios innecesarios.	(0.5) El código es limpio, está bien estructurado y responde a la guía de estilo.