Actividad 1.1: Configuración entorno y Tratamiento fichero.txt

A. Investigación sobre clases del paquete java.io

¿Para qué se utilizan las siguientes clases?

- **File:** Representa de forma abstracta la ruta de un archivo o directorio en el sistema de archivos. No se usa para leer o escribir contenido, sino para verificar si existe, crear directorios, borrar archivos o ver sus propiedades.
- **FileReader:** Es una clase de conveniencia para leer flujos de caracteres desde archivos de texto. Lee los datos directamente del archivo, carácter por carácter o mediante arreglos de caracteres.
- **BufferedReader:** Lee texto de un flujo de entrada de caracteres, almacenando en búfer los caracteres para proporcionar una lectura eficiente de caracteres, arreglos y líneas (mediante el método readLine()).
- **FileWriter:** Es una clase de conveniencia para escribir flujos de caracteres en archivos de texto. Escribe directamente en el archivo subyacente carácter por carácter.
- **BufferedWriter:** Escribe texto en un flujo de salida de caracteres, almacenando en búfer los caracteres para proporcionar una escritura eficiente de caracteres, arreglos y cadenas.
- **PrintWriter:** Imprime representaciones formateadas de objetos en un flujo de salida de texto. Cuenta con métodos muy prácticos como print(), println() y printf().

¿Qué diferencia hay entre FileReader y BufferedReader? ¿Cuál es más eficiente en el tratamiento de información? ¿Por qué?

FileReader es una clase diseñada para leer flujos de caracteres directamente desde un archivo en el disco. Su principal característica es que cada solicitud de lectura que se le realiza se traduce en una operación directa al sistema de archivos subyacente, leyendo carácter por carácter o en pequeños arreglos.

BufferedReader, por otro lado, es una clase que envuelve a otro lector de caracteres (como FileReader) y añade una memoria intermedia llamada búfer. En lugar de ir al disco por cada carácter, lee un bloque grande de datos de una sola vez y los almacena en la memoria RAM, atendiendo las siguientes lecturas de forma inmediata desde allí. Además, cuenta con el método readLine() para capturar líneas de texto completas.

La clase más eficiente es BufferedReader. Esto se debe a que las operaciones de entrada/salida (I/O) a nivel de disco físico son extremadamente costosas en tiempo de ejecución. Al minimizar drásticamente el número de accesos directos al archivo y resolver la lectura desde la memoria intermedia, el rendimiento del programa aumenta de manera sustancial. Por esta razón, Oracle recomienda envolver siempre cualquier FileReader dentro de un BufferedReader.

¿Qué diferencia hay entre FileWriter, BufferedWriter y PrintWriter?

FileWriter es una herramienta para la escritura de texto. Se conecta de forma directa a un archivo físico y escribe flujos de caracteres en él. Al igual que FileReader, carece de

memoria intermedia, por lo que cada método de escritura (write) impacta de inmediato en el almacenamiento del sistema, lo que la hace lenta si se usa de manera intensiva.

BufferedWriter introduce eficiencia al proceso de escritura. Almacena los caracteres que se desean escribir en una memoria intermedia propia. A nivel de funcionalidades, añade el método newLine() para insertar saltos de línea de forma independiente del sistema operativo.

PrintWriter se enfoca por completo en la comodidad del desarrollador y en el formato de los datos. No añade eficiencia por sí misma (debe envolver a un BufferedWriter para ser eficiente), pero proporciona una amplia gama de métodos altamente prácticos como print(), println() y printf(). Esto permite escribir representaciones de texto formateadas de variables de cualquier tipo (enteros, booleanos, objetos complejos) de la misma manera que se hace en la consola con System.out.

Entre FileWriter y PrintWriter ¿Cuál de las dos clases lanza excepciones de tipo IOException?¿Cómo gestionará los errores la otra clase? ¿Cuál de las dos clases usarías para añadir/escribir texto al final de un fichero?

La clase que lanza excepciones de tipo IOException es FileWriter. Prácticamente todos sus métodos de construcción y escritura están declarados con throws IOException. Esto obliga estrictamente al programador a gestionar los errores en el código mediante bloques try-catch.

PrintWriter gestiona los errores de forma interna y silenciosa. Ninguno de sus métodos de impresión (print, println, etc.) lanza excepciones de tipo IOException. En su lugar, si ocurre un fallo durante la escritura, la clase captura el error internamente y activa una bandera de estado. El programador debe consultar explícitamente el método checkError() si desea verificar si la última operación falló.

Para añadir o escribir texto al final de un fichero (modo append), se debe usar FileWriter. Esta clase cuenta con un constructor específico que recibe un valor booleano (new FileWriter("archivo.txt", true)), el cual le indica al sistema operativo que no debe sobrescribir el archivo, sino posicionar el cursor al final del contenido existente. PrintWriter no dispone de un constructor nativo que reciba una ruta de archivo y permita activar el modo append directamente, por lo que siempre requerirá de un FileWriter configurado previamente para lograr este comportamiento.

Fuentes:

Documentación oficial de Oracle

https://docs.oracle.com/en/java/javase/21/docs/api/overview-tree.html
https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/io/FileReader.html
https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/io/BufferedReader.html
https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/io/FileWriter.html
https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/io/BufferedWriter.html
https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/io/PrintWriter.html