1️⃣

Actividad 1

Actividad1.1-Investigación_y_Tratamiento_fichero_txt.pdf

A. Investigación sobre clases del paquete java.io

1. ¿Para qué se utilizan las siguientes clases?

- File.-

Representa de forma abstracta la ruta de un archivo o directorio en el sistema, permitiendo crear, eliminar, renombrar o verificar propiedades físicas (tamaño, permisos), pero no lee ni escribe el contenido del archivo.

- FileReader.-

Permite leer flujos de caracteres (streams) desde un archivo físico, decodificando los bytes a caracteres de forma directa.

- BufferedReader.-

Lee texto de un flujo de entrada de caracteres y lo almacena en un búfer interno de memoria para optimizar la velocidad de las operaciones de lectura de líneas y grandes bloques de texto.

- FileWriter.-

Permite escribir flujos de caracteres directamente en un archivo físico del disco.

- BufferedWriter.-

Escribe texto en un flujo de salida de caracteres utilizando un búfer interno para agrupar los datos y minimizar la cantidad de operaciones de escritura físicas en el disco.

- PrintWriter.-

Proporciona métodos de alto nivel (`print`, `println`, `printf`) para escribir representaciones formateadas de objetos y variables primitivas en un flujo de salida de texto sin tener que convertir explícitamente los tipos de datos.

2. ¿Qué diferencia hay entre FileReader y BufferedReader? ¿Cuál es más eficiente en el tratamiento de información? ¿Por qué?
3. ¿Qué diferencia hay entre FileWriter, BufferedWriter y PrintWriter?
4. 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?

- `FileWriter` realiza la escritura a nivel básico, escribiendo los caracteres directamente en el archivo cada vez que se invoca un método.
- `BufferedWriter` envuelve a un escritor base (como `FileWriter`) y acumula los caracteres en memoria, escribiendo en el disco en bloques grandes de una sola vez, lo que mejora drásticamente el rendimiento en escrituras repetitivas.
- `PrintWriter` se centra en el formato y la conveniencia del programador. No está pensado principalmente para la eficiencia pura de I/O, sino para facilitar la escritura de variables y cadenas complejas mediante sus métodos de formateo, pudiendo envolver a las otras dos clases para combinar su formato con el almacenamiento en búfer.

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?

`FileWriter` es la clase que lanza excepciones de tipo `IOException` en casi todos sus constructores y métodos, obligándote a manejarlas con un bloque `try-catch` o propagarlas.

`PrintWriter`, por el contrario, no lanza excepciones `IOException` durante la ejecución de sus métodos de escritura. En su lugar, captura cualquier excepción de este tipo internamente y activa un indicador (flag) de error. Para verificar si algo falló, debes llamar manualmente al método `checkError()` que devolverá un valor booleano.

Para añadir texto al final de un fichero existente sin sobrescribirlo, usarías `FileWriter`. Debes instanciarlo utilizando su constructor que acepta un parámetro booleano `append` establecido en `true` (ejemplo: `new FileWriter("archivo.txt", true)`). Si deseas aprovechar las ventajas de formato de `PrintWriter` al añadir texto, puedes pasar esa misma instancia de `FileWriter` al constructor del `PrintWriter`.