Actividad 1.1. Investigación sobre clases del paquete java.io

1. ¿Para qué se utilizan estas clases?

- **File**: Representa la ruta y las propiedades de un archivo o directorio. No lee ni escribe contenido, solo sirve para saber si el archivo existe, crearlo, borrarlo o ver cuánto pesa.
- **FileReader**: Lee texto de un archivo en crudo. Lee directamente del disco, carácter por carácter.
- **BufferedReader**: Lee texto de forma eficiente. En lugar de ir al disco constantemente, carga un bloque gigante de datos en la memoria RAM (el "buffer") y lo va sirviendo. Además, incluye el método `readLine()` para leer línea a línea.
- **FileWriter**: Escribe texto (caracteres) directamente en un archivo en el disco.
- **BufferedWriter**: La versión optimizada del anterior. Acumula el texto en la RAM y lo vuelca todo de golpe al disco cuando el buffer se llena (o cuando se hace un `flush()`).
- **PrintWriter**: Se usa para escribir texto por consola (`print()` , `println()` , `printf()`), permitiendo escribir variables y objetos directamente como texto.

2. FileReader vs BufferedReader: Diferencias y eficiencia

BufferedReader es más eficiente. La diferencia principal radica en **cuántas veces interactúan con el disco duro**. Las operaciones de entrada/salida (I/O) en disco son de las tareas más lentas en programación.

FileReader va al disco a leer cada vez que se lo pides. Por el contrario, **BufferedReader** es inteligente: envuelve a un lector (como `FileReader`), va al disco una sola vez, se trae todo el texto a la memoria y lo va dando desde ahí. Minimizar los viajes al disco hace que el código vaya más rápido en comparación.

3. Diferencia entre FileWriter, BufferedWriter y PrintWriter

- **FileWriter**: escribe caracteres directamente al disco sin optimizaciones.
- **BufferedWriter**: agrupa las escrituras en la memoria para no machacar el disco. Es pura optimización de rendimiento.
- **PrintWriter**: Da herramientas de formateo (`println()`) para que no tengas que convertir tipos de datos (enteros, booleanos) a *Strings* a mano antes de guardarlos.

4. FileWriter vs PrintWriter: Excepciones y modo "Append"

Gestión de errores y excepciones:

La clase que lanza `IOException` es **FileWriter**.

PrintWriter sus métodos habituales (como `print` o `println`) no lanzan `IOException`. Para saber si algo ha fallado, tenemos que hacerlo manualmente llamando al método `checkError()`, que devolverá `true` si algo dió error durante la escritura.

Para añadir texto al final (Append):

Usaríamos la clase **FileWriter**. Esta respeta el texto existente a través de su constructor. Para añadir texto al final sin borrar lo que ya hay, basta con pasarle un `true` como segundo parámetro:

```
1   new FileWriter("fichero.txt", true);
```