¿Para qué se utilizan las siguientes clases?

File: Representa de forma abstracta la ruta de un archivo o directorio en el disco. No lee ni escribe contenido por sí misma; sirve para crear, borrar, renombrar o comprobar la existencia y propiedades de un archivo.

FileReader: Se utiliza para leer flujos de caracteres (texto) desde un archivo de texto en el disco, carácter por carácter o en bloques de caracteres.

BufferedReader: Se usa para leer texto de manera eficiente envolviendo otro lector (como FileReader). Almacena los caracteres en un búfer interno para evitar acceder al disco constantemente.

FileWriter: Se utiliza para escribir flujos de caracteres (texto) en un archivo de texto en el disco, carácter por carácter o en bloques.

BufferedWriter: Se usa para escribir texto de manera eficiente envolviendo otro escritor (como FileWriter). Acumula los caracteres en un búfer interno y los escribe en el disco de una sola vez cuando el búfer se llena.

PrintWriter: Es una clase de alto nivel que se utiliza para escribir representaciones formateadas de objetos en texto plano. Ofrece métodos muy cómodos como print() y println(), idénticos a los que usas con System.out.

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

FileReader lee directamente del disco duro, lee carácter a carácter y tiene una eficiencia baja
BufferedReader lee de un buffer en memoria RAM, lee lineas completas y tiene una eficiencia alta

FileReader realiza una petición de lectura al sistema operativo (acceso al disco) cada vez que se le pide un carácter. El acceso al almacenamiento físico es una operación muy lenta. En cambio, BufferedReader lee un gran bloque de caracteres del disco de una sola vez y los guarda en la memoria RAM (búfer). Las siguientes lecturas se realizan directamente desde la memoria RAM.

¿Qué diferencia hay entre FileWriter, BufferedWriter y PrintWriter?

- FileWriter: Es la herramienta básica de bajo nivel para escribir texto en disco. Escribe directamente carácter por carácter, por lo que no es óptima para grandes volúmenes de datos.

- BufferedWriter: Aporta eficiencia de rendimiento mediante el uso de un búfer intermedio en memoria RAM, reduciendo el número de accesos físicos al disco.

- PrintWriter: Aporta comodidad y formato. No mejora el rendimiento por sí sola (a menos que envuelva a un BufferedWriter), pero permite formatear texto fácilmente usando printf(), print() y println(), convirtiendo automáticamente números y objetos a texto legible.

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 lanza IOException en casi todos sus métodos de escritura, lo que te obliga a gestionar el error obligatoriamente mediante un bloque try-catch o propagarlo con throws.

PrintWriter no lanza IOException en sus métodos de impresión (print, println). En su lugar, gestiona los errores internamente de forma silenciosa. Si ocurre un fallo durante la escritura, la clase activa una bandera interna de error. Para saber si algo falló, debes invocar manualmente su método checkError(), el cual devolverá true si ocurrió un problema.

Para escribir al final de un fichero existente (modo append o añadir), debes usar FileWriter (o pasar un FileWriter configurado dentro de las otras clases). Esto se logra utilizando su constructor especial que recibe un booleano:

```
FileWriter fw = new FileWriter("archivo.txt", true);
// El parámetro 'true' activa el modo añadir
PrintWriter pw = new PrintWriter(new BufferedWriter(fw));
pw.println("Texto al final del archivo");
```