File: Representa una abstracción de las rutas de archivos y directorios del sistema de archivos. Se usa para consultar metadatos (comprobar si existe, tamaño, permisos) y realizar operaciones de gestión.

FileReader: Clase para leer ficheros de texto directamente. Lee los datos carácter a carácter o mediante arrays de caracteres de tamaño fijo.

BufferedReader: Lee texto de una secuencia de entrada almacenando en un buffer. Permite leer eficientemente caracteres, arrays y líneas completas de texto.

FileWriter: Clase básica de bajo nivel para ficheros de texto directamente en un archivo del disco.

BufferedWriter: Escribe texto en una secuencia de salida utilizando un búfer de memoria. Agrupa las operaciones de escritura antes de enviarlas al disco, reduciendo los accesos a los dispositivos de almacenamiento e independizando el salto de línea según el sistema operativo mediante newLine().

La diferencia principal radica en que **FileReader** realiza la lectura física directamente desde el archivo en disco carácter a carácter, mientras que **BufferedReader** lee y almacena un bloque grande de caracteres en una memoria, permitiendo además leer líneas completas mediante el método readLine(). Por este motivo, **BufferedReader** es significativamente más eficiente.

FileWriter es una clase básica de bajo nivel que escribe caracteres directamente en el archivo en disco provocando múltiples accesos físicos, mientras que **BufferedWriter** envuelve a otro flujo de escritura para acumular los datos en un búfer de memoria RAM antes de guardarlos de una sola vez e insertar saltos de línea adaptados al sistema operativo mediante **newLine()**. Por su parte, **PrintWriter** es una clase avanzada orientada al formato que ofrece métodos como **print(), println() y printf()**, lo que facilita la conversión automática y escritura de diferentes tipos de datos a texto.

La clase que lanza directamente excepciones comprobadas de tipo IOException es **FileWriter**, obligando a gestionarlas con un bloque try-catch o declararlas mediante. Por el contrario, PrintWriter captura e ignora internamente las excepciones IOException en sus métodos de impresión, por lo que la gestión de errores no usa excepciones sino su propio método **checkError()**, el cual devuelve true si ocurrió alguna falla.
Finalmente, para añadir o escribir texto al final de un fichero se utiliza **FileWriter** activando el parámetro booleano true en su constructor pudiendo encadenarse con **BufferedWriter y PrintWriter** para combinar la eficiencia de memoria y las facilidades de formateo.