A. Investigación sobre clases del paquete java.io

¿Para qué se utilizan las siguientes clases?

File: Representa de forma abstracta las rutas de archivos y directorios en el sistema de ficheros. Permite consultar metadatos (tamaño, permisos, existencia), crear o borrar ficheros o directorios, pero no lee ni escribe datos directamente en ellos.

FileReader: Flujo de caracteres orientado a la lectura de datos desde un archivo de texto de manera nativa (utilizando la codificación por defecto del sistema). Lee caracteres uno a uno o en pequeños búferes de arrays.

BufferedReader: Envuelve a otro lector (como FileReader) añadiendo un búfer de memoria intermedia. Esto reduce drásticamente los accesos físicos al disco duro y permite leer líneas completas de texto de manera eficiente mediante el método readLine().

FileWriter: Flujo de caracteres diseñado específicamente para escribir datos de texto directamente en un fichero. Puede sobrescribir el archivo o abrirlo en modo adjuntar (append) si se le indica en el constructor.

BufferedWriter: Envuelve a un escritor (como FileWriter) proporcionando un búfer de almacenamiento temporal para las escrituras, mejorando el rendimiento al minimizar las operaciones de E/S físicas y facilitando el salto de línea con newLine().

PrintWriter: Clase de apoyo que permite imprimir representaciones formateadas de objetos de tipos de datos primitivos y objetos (print(), println(), printf()). Facilita enormemente la escritura estructurada de texto.

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

La diferencia principal radica en que FileReader lee directamente del almacenamiento físico carácter por carácter (o en arrays pequeños), mientras que BufferedReader almacena un bloque de datos en un búfer de memoria RAM al leer.

Es más eficiente BufferedReader porque agrupa las lecturas físicas del disco, reduciendo el número de operaciones de E/S de bajo nivel (que son costosas en términos de rendimiento). Además, aporta la comodidad del método readLine().

¿Qué diferencia hay entre FileWriter, BufferedWriter y PrintWriter?

FileWriter escribe flujos de caracteres directamente al fichero de forma básica.

BufferedWriter añade un búfer en memoria para acelerar las operaciones repetitivas de escritura.

PrintWriter ofrece métodos de formato avanzados (println, printf), maneja de forma transparente la propagación o gestión de errores de E/S y no lanza directamente IOException en sus métodos de impresión.

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 obliga a capturar o declarar explícitamente excepciones de tipo IOException

PrintWriter no lanza excepciones IOException directamente en sus métodos de escritura (print/println); en su lugar, captura internamente los errores y marca un indicador interno de fallo que puede consultarse mediante el método checkError().

Para añadir texto al final de un fichero de forma sencilla, se suele usar la combinación de ambos o FileWriter pasando un parámetro boolean a true en su constructor (new FileWriter("fichero.txt", true))