Ejercicio 01.11

El programa tiene como objetivo principal copiar el contenido de un fichero a otro, utilizando la lectura y escritura de datos en bloque. Para ello, se ha empleado Java y las clases proporcionadas por la API java.nio.file, que permiten un manejo eficiente de ficheros mediante rutas y operaciones de entrada/salida.

En primer lugar, el programa solicita al usuario los nombres de los ficheros de origen y destino mediante un objeto Scanner. Esta interacción permite que el usuario especifique qué fichero desea copiar y el nombre del fichero resultante, proporcionando flexibilidad frente a rutas fijas en el código.

```
Rename usages
public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        try {
            // Solicitar el nombre del fichero de origen
            System.out.print("Introduce el nombre del fichero de origen: ");
            String sourceFileName = scanner.nextLine();
            Path sourcePath = Paths.get(sourceFileName);

            // Solicitar el nombre del fichero de destino
            System.out.print("Introduce el nombre del fichero de destino: ");
            String destinationFileName = scanner.nextLine();
            Path destinationPath = Paths.get(destinationFileName);
```

Una vez obtenidos los nombres de los ficheros, se crean objetos Path a partir de las rutas proporcionadas. La clase Path representa de manera abstracta la ubicación de un fichero en el sistema de archivos y permite trabajar con él de forma más segura y moderna que las clases tradicionales de Java como File. Esta abstracción facilita la lectura y escritura de datos de manera directa y eficiente.

El núcleo del programa consiste en leer todo el contenido del fichero de origen de una sola vez mediante Files.readAllBytes(). Este método devuelve un array de bytes que representa exactamente todos los datos del fichero. La elección de esta estrategia garantiza que se capture el contenido completo, independientemente del tipo de fichero, ya sea texto o binario, y evita la necesidad de recorrer el fichero línea por línea o en bloques más pequeños.

A continuación, el array de bytes obtenido se escribe en el fichero de destino mediante Files.write(). Este método sobrescribe automáticamente cualquier contenido existente en el fichero de destino o crea uno nuevo si no existía previamente. De esta manera, el programa asegura que la copia sea exacta y completa, manteniendo el tamaño y la estructura original del fichero de origen.

```
// Leer todo el contenido del fichero de origen
byte[] fileContent = Files.readAllBytes(sourcePath);

// Escribir el contenido en el fichero de destino
Files.write(destinationPath, fileContent);
```

Para manejar posibles errores, como la inexistencia del fichero de origen o problemas de permisos, el código incluye un bloque try-catch. Este bloque captura las excepciones IOException y muestra un mensaje de error descriptivo en la consola, evitando que el programa falle abruptamente. Finalmente, en el bloque finally, se cierra el objeto Scanner para liberar recursos y evitar posibles fugas de memoria.

```
            System.out.println("El fichero se ha copiado correctamente.");
        } catch (IOException e) {
            System.err.println("Error al copiar el fichero: " + e.getMessage());
        } finally {
            scanner.close();
        }
    }
}
```

En resumen, el programa implementa un método sencillo pero efectivo para copiar ficheros enteros en Java. Su estructura es clara, segura y flexible, permitiendo al usuario definir los nombres de los ficheros y garantizando que el contenido se transfiera completamente de manera eficiente. La utilización de Files.readAllBytes() y Files.write() simplifica el proceso, evitando bucles innecesarios y asegurando la integridad de los datos.