

Ejercicio 01.02 - Escribir en ficheros de texto

El programa desarrollado tiene como finalidad trabajar con ficheros de texto en Java, tanto en la parte de escritura como en la de lectura. Para ello, se utiliza la clase `BufferedWriter` que permite escribir en el fichero de manera controlada. En este caso, el programa crea un archivo llamado *output.txt* en el que se van guardando los números impares comprendidos entre el 1 y el 10. El bucle `for` recorre todos los números del rango, comprobando si cada uno de ellos es impar con la condición `i % 2 != 0`. Si se cumple esta condición, se escribe el número en el fichero, seguido de un salto de línea. Una vez finalizado el recorrido, se añade manualmente una última línea con el texto *Fin del fichero*, que indica al usuario el cierre lógico del contenido.

```
import java.io.*;
import java.nio.file.*;
import java.util.List;

public class Main {
    public static void main(String[] args) {
        String fileName = "output.txt";

        // Escribir en el fichero
        try (BufferedWriter writer = new BufferedWriter(new FileWriter(fileName))) {
            for (int i = 1; i <= 10; i++) {
                if (i % 2 != 0) {
                    writer.write(str: i + "\n");
                }
            }
            writer.write(str: "Fin del fichero\n");
        } catch (IOException e) {
            System.err.println("Error al escribir en el fichero: " + e.getMessage());
        }
    }
}
```

Una vez escrito el archivo, el programa continúa con el proceso de lectura. Para ello se emplea la clase `BufferedReader`, que permite recorrer línea a línea el contenido escrito en el fichero. Mediante un bucle `while`, se van leyendo todas las líneas hasta que el método `readLine()` devuelve `null`, lo que significa que se ha llegado al final del archivo. Cada línea leída se muestra por pantalla, lo que permite comprobar que el contenido se ha escrito correctamente y que el fichero está accesible.

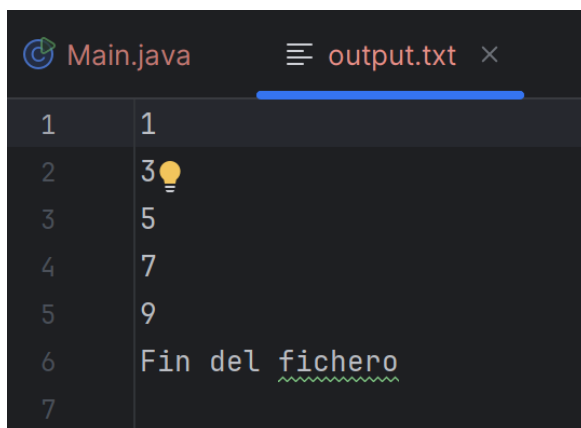
Finalmente, se incluye una segunda forma de leer el fichero, esta vez utilizando la clase `Files` y su método `readAllLines`. Esta alternativa carga todo el contenido en una lista de cadenas, lo que facilita mostrarlo de una sola vez. Posteriormente, se recorre la lista mediante una expresión `lambda` para imprimir cada línea por pantalla. De este modo, se pueden comparar dos formas distintas de realizar la lectura de un fichero: una línea a línea y otra cargando todo el contenido de golpe.

```
// Leer el fichero y mostrarlo en pantalla
try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {
    String line;
    while ((line = reader.readLine()) != null) {
        System.out.println(line);
    }
} catch (IOException e) {
    System.err.println("Error al leer el fichero: " + e.getMessage());
}

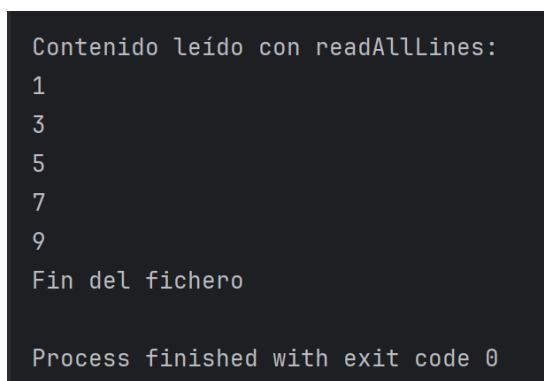
// Leer todo de golpe con readAllLines
try {
    List<String> lines = Files.readAllLines(Paths.get(fileName));
    System.out.println("\nContenido leído con readAllLines:");
    lines.forEach(System.out::println);
} catch (IOException e) {
    System.err.println("Error al leer el fichero con readAllLines: " + e.getMessage());
}
}
```

En conclusión, este programa no solo cumple con la tarea de escribir y leer números impares en un archivo, sino que también ejemplifica distintas formas de trabajar con ficheros en Java.

Aquí podemos ver cómo se ha creado el fichero con el contenido mencionado anteriormente, y seguidamente al ejecutar el programa cómo se puede leer su contenido por pantalla.



Linea	Contenido
1	1
2	3
3	5
4	7
5	9
6	Fin del fichero
7	



```
Contenido leído con readAllLines:
1
3
5
7
9
Fin del fichero

Process finished with exit code 0
```