Práctica 2 – Programación de Servicios y Procesos (PSP)

Alberto agredano orellana

Introducción

Esta práctica tiene como objetivo que el alumnado aprenda a desarrollar aplicaciones que trabajen con entrada y salida estándar, además de comprender cómo se comunican los procesos mediante tuberías en un sistema operativo.

El trabajo consiste en implementar tres programas en Java: lectorTexto, filtraLineas y contadorPalabras. Cada aplicación resuelve un problema concreto y en conjunto permiten realizar un flujo de procesamiento encadenado.

Requisitos previos

Para poder realizar esta práctica, fue necesario disponer de:
- Un entorno de desarrollo Java (JDK 17 en mi caso).
- Un IDE (utilicé IntelliJ IDEA, aunque también se podría usar NetBeans o Eclipse).
- Maven como herramienta de gestión de dependencias y empaquetado.
- Conocimientos básicos de Java, entrada/salida con ficheros y consola.
- Un sistema operativo con soporte para tuberías en consola (Linux, macOS o Windows con PowerShell).

Metodología de trabajo

La práctica se dividió en varias fases:

1. Configuración del proyecto en IntelliJ IDEA con Maven.
2. Implementación de la aplicación lectorTexto, encargada de leer el fichero de entrada.
3. Implementación de filtraLineas, que filtra las líneas recibidas por la entrada estándar.
4. Implementación de contadorPalabras, que cuenta el número total de palabras.
5. Pruebas individuales de cada aplicación.
6. Prueba final integrando las tres aplicaciones mediante el operador de tubería '|'.

1. Aplicación lectorTexto

El primer paso fue desarrollar lectorTexto. Esta aplicación lee el archivo entrada.txt línea por línea y muestra su contenido por consola.

El motivo de usar un BufferedReader es que permite una lectura eficiente línea a línea. Además, se incluyó un control de excepciones para capturar posibles errores, como la ausencia del archivo.

2. Aplicación filtraLineas

Después implementé filtraLineas, que recibe texto desde la entrada estándar. La decisión de usar entrada estándar permite que esta aplicación pueda funcionar de forma aislada o encadenada con otras mediante tuberías. El criterio de filtrado fue mostrar solo las líneas con más de 20 caracteres, tal y como pedía el enunciado.

3. Aplicación contadorPalabras

Por último, desarrollé contadorPalabras, que cuenta cuántas palabras hay en el texto recibido por la entrada estándar. Se definió como palabra cualquier secuencia separada por espacios. El algoritmo divide las líneas en fragmentos usando split(' ') y acumula el total. La razón de hacerlo así es que resulta simple y efectivo para la práctica, aunque en un entorno real se podrían usar expresiones regulares más complejas.

Manual de uso y pruebas

Cada aplicación puede ejecutarse de manera independiente, pero la parte más interesante es cuando se combinan en una tubería.

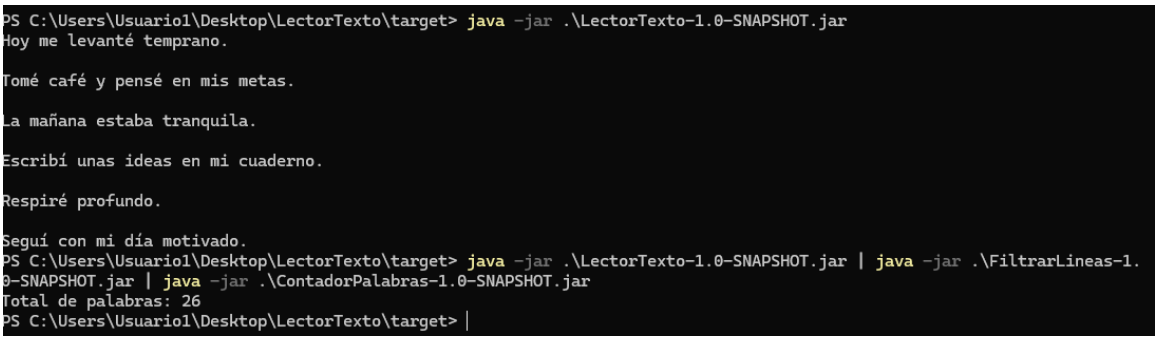
Ejecución independiente:
java -jar LectorTexto-1.0-SNAPSHOT.jar
java -jar FiltrarLineas-1.0-SNAPSHOT.jar
java -jar ContadorPalabras-1.0-SNAPSHOT.jar

Ejecución encadenada:
java -jar LectorTexto-1.0-SNAPSHOT.jar | java -jar FiltrarLineas-1.0-SNAPSHOT.jar | java -jar ContadorPalabras-1.0-SNAPSHOT.jar

En este caso:
- lectorTexto lee el archivo entrada.txt
- filtraLineas filtra las líneas con más de 20 caracteres

Alberto agredano orellana

- contadorPalabras cuenta cuántas palabras hay en el texto resultante



Ejemplo con entrada.txt:

Hoy me levanté temprano.

Tomé café y pensé en mis metas.

La mañana estaba tranquila.

Escribí unas ideas en mi cuaderno.

Respiré profundo.

Seguí con mi día motivado.

Al ejecutar la tubería completa, se obtiene como resultado el total de palabras de las líneas filtradas.

Conclusiones

La práctica me permitió comprender mejor el uso de entrada y salida estándar en Java, así como la importancia de diseñar aplicaciones que puedan comunicarse mediante tuberías.

Además, reforcé el manejo de ficheros, la captura de errores y la forma de documentar correctamente un proyecto.

Este tipo de ejercicios ayudan a desarrollar buenas prácticas en la programación y a preparar el terreno para aplicaciones más complejas.

Alberto agredano orellana