

**Universidad de las Fuerzas Armadas - ESPE**

Departamento de Ciencias de la Computación

Carrera de Ingeniería de Software

Análisis y Diseño de Software - NRC:22426

**Trabajo:**

U2T2 - Patrón de diseño Tubería y filtro.

**Grupo:** 4

**Integrantes:**

Diego Casignia

Anthony Villarreal

Javier Ramos

**Profesora**: Ing. Jenny Ruiz

1. **Definición del patrón de diseño (Tubería y filtro).**

El patrón Tubería y Filtro organiza el procesamiento de datos en una secuencia de componentes, donde:

* **Filtros:** Son los componentes que procesan los datos de entrada, transformando o filtrando para producir una salida. Cada filtro realiza una tarea específica y es independiente de los demás.
* **Tuberías:** Son los conectores que transportan los datos de un filtro al siguiente, actuando como un canal de comunicación.
* **Flujo de datos:** Los datos fluyen de manera unidireccional desde una fuente inicial, pasando por los filtros a través de las tuberías, hasta llegar a un destino final.

**Características principales:**

* **Modularidad:** Cada filtro es independiente y puede ser reutilizado o reemplazado sin afectar al resto del sistema.
* **Escalabilidad:** Se pueden agregar o quitar filtros según las necesidades.
* **Paralelismo:** Los filtros pueden ejecutarse en paralelo si no dependen entre sí, mejorando el rendimiento.
* **Reusabilidad:** Los filtros pueden usarse en diferentes pipelines.

**Cuándo usarlo:**

* Cuando se necesita procesar datos en etapas secuenciales.
* Cuando los datos deben transformarse o filtrarse de manera modular.
* En sistemas que manejan flujos de datos, como procesamiento de texto, imágenes o streams de datos.

1. **Aplicación en la industria.**

El patrón Tubería y Filtro es ampliamente utilizado en diversas industrias debido a su capacidad para manejar flujos de datos de manera estructurada y eficiente. Por ejemplo:

* Procesamiento de datos en tiempo real:
  + Sistemas de streaming: Plataformas como Apache Kafka o Flink usan este patrón para procesar grandes volúmenes de datos en etapas (lectura, transformación, agregación, escritura).
  + Análisis de logs: En sistemas de monitoreo, los logs se procesan en etapas para filtrar, transformar y generar alertas (ejemplo: Splunk).
* Procesamiento de señales e imágenes:
  + En aplicaciones de procesamiento de imágenes, como Photoshop o herramientas de edición de video, los filtros aplican transformaciones (ajuste de brillo, desenfoque, etc.) en secuencia.
  + En telecomunicaciones, las señales se procesan a través de filtros para eliminar ruido o codificar datos.
* Compiladores:
  + Los compiladores dividen el proceso de compilación en etapas (análisis léxico, sintáctico, generación de código, optimización), donde cada etapa actúa como un filtro.
* Sistemas de integración empresarial:
  + En sistemas ETL (Extract, Transform, Load), los datos se extraen de una fuente, se transforman en varias etapas (limpieza, enriquecimiento) y se cargan en un destino, como una base de datos.

**Beneficios en la industria:**

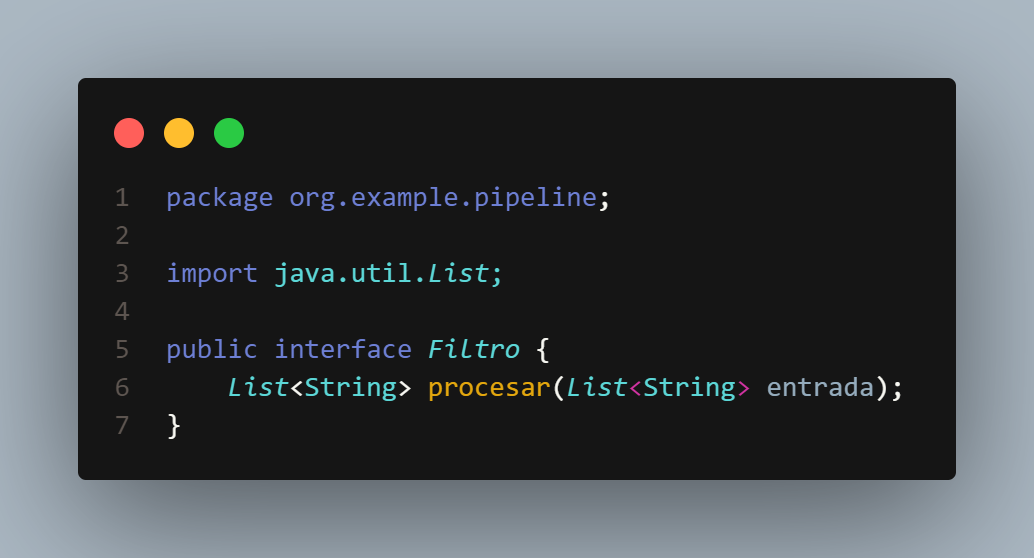
* Mejora la mantenibilidad al dividir el procesamiento en componentes independientes.
* Facilita la integración de nuevas funcionalidades añadiendo filtros.
* Optimiza el uso de recursos al permitir procesamiento paralelo.

1. **Ejemplo con código fuente IDE OO.**

A continuación, se presenta un ejemplo en Java que implementa el patrón Tubería y Filtro para procesar una cadena de texto. El objetivo es tomar una frase, eliminar palabras cortas, convertir en mayúsculas y agregar un prefijo a cada palabra. El código usa clases para modelar los filtros y tuberías, siguiendo un enfoque orientado a objetos.

**Clases:**

1. **Filtro.java:**



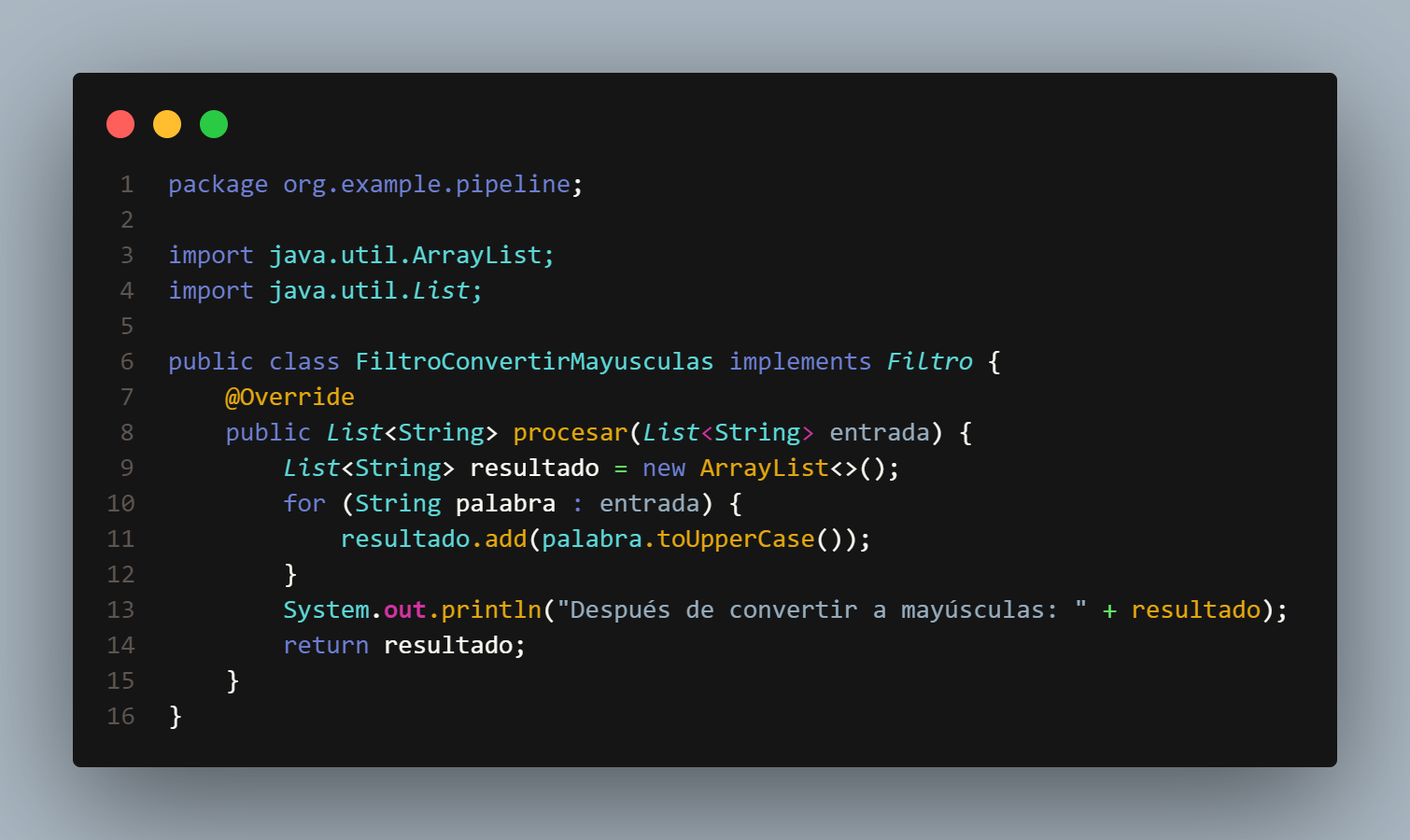
* Interfaz Filtro: Define el contrato para todos los filtros, asegurando que cada uno implemente el método procesar.

1. **FiltroAgregarPrefijo.java**



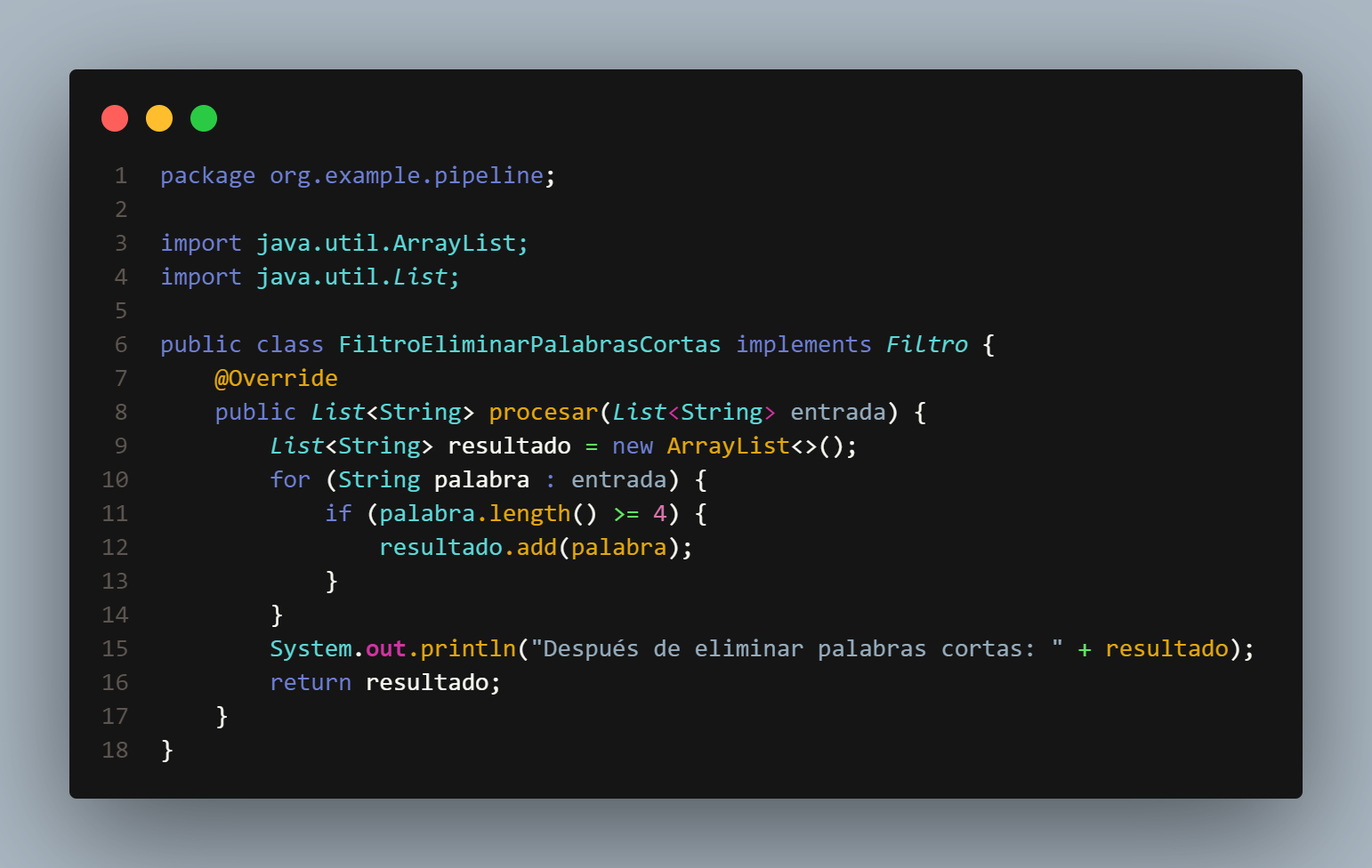
* FiltroAgregarPrefijo: Agrega un prefijo a cada palabra.

1. **FiltroConvertirMayusculas.java**



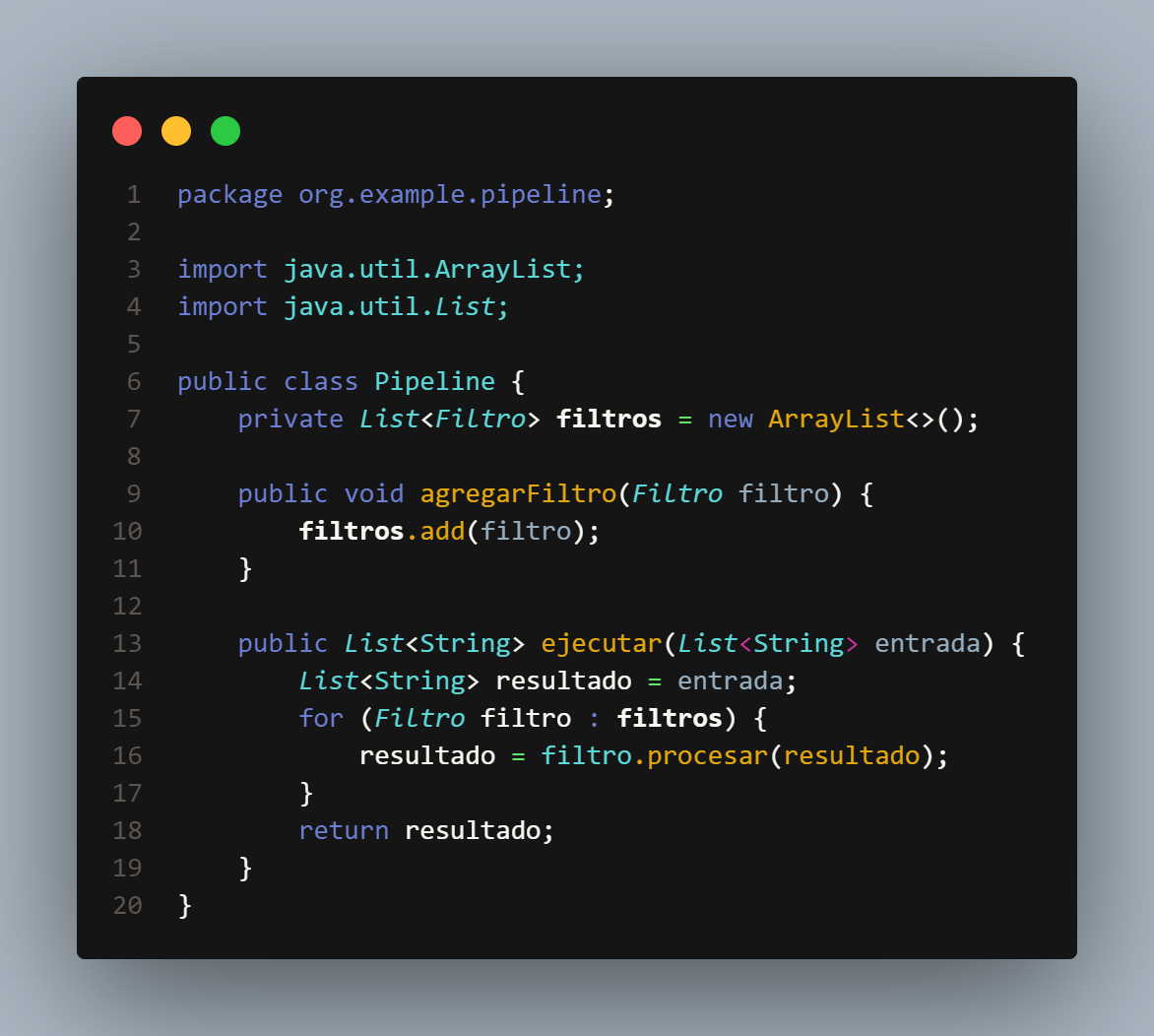
* FiltroConvertirMayusculas: Convierte las palabras a mayúsculas.

1. **FiltroEliminarPalabrasCortas.java**



* FiltroEliminarPalabrasCortas: Elimina palabras con menos de 4 caracteres.

1. **Pipeline.java**



* Clase Pipeline: Gestiona la secuencia de filtros y actúa como la tubería, ejecutando cada filtro en orden.

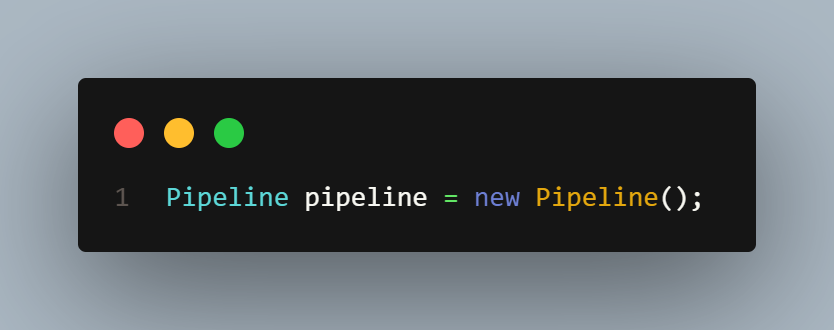
1. **PipelineTexto.java**



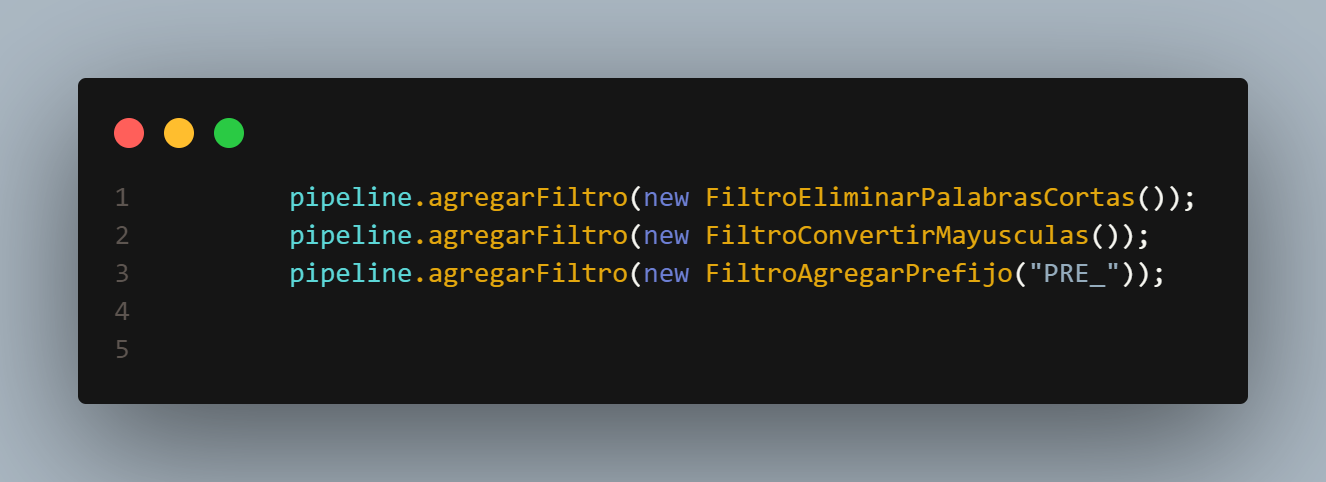
* Clase PipelineTexto: Contiene el método main, que configura la entrada, crea la tubería, agrega los filtros y ejecuta el procesamiento.



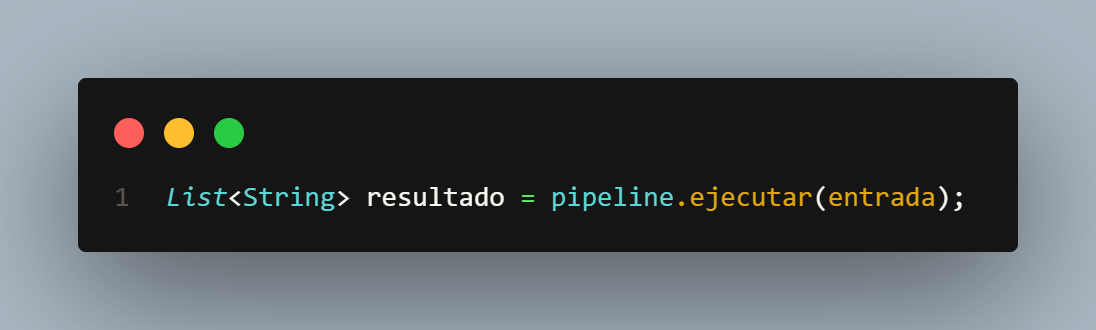
* + Se establece la fuente de **datos (source)**, que es el punto de partida del flujo en el patrón Tubería y Filtro.



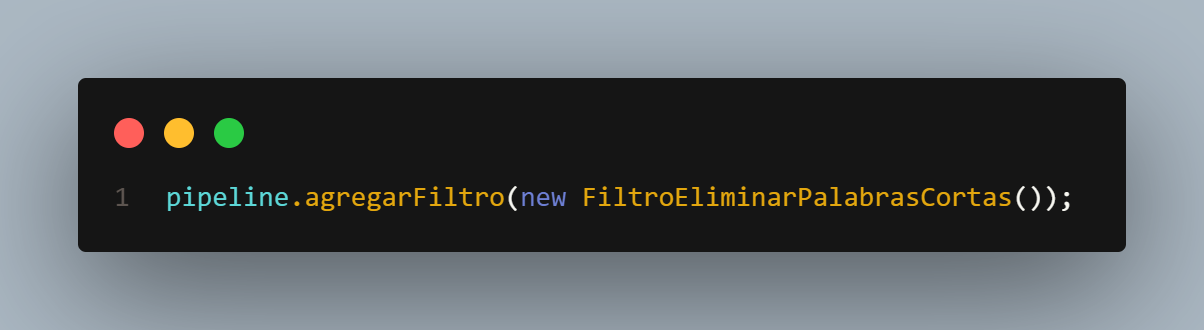
* + La clase Pipeline representa la **tubería**, que conectará los filtros y controlará el flujo de datos entre ellos.



* + Cada filtro es un módulo autónomo que realiza una transformación específica. El orden en que se agregan determina el orden de procesamiento, lo que refleja la **secuencia lineal** característica del patrón.



* + El método ejecutar de la clase Pipeline toma la lista de entrada y la pasa secuencialmente por cada filtro. El flujo de datos es el siguiente:



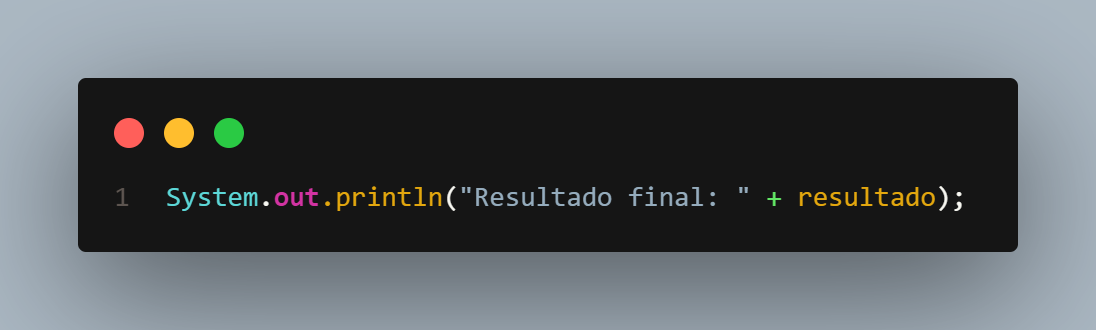
* + Este filtro transforma los datos, eliminando elementos según un criterio, y los pasa a la siguiente etapa.



* + Este filtro aplica una transformación simple y pasa los datos al siguiente filtro.

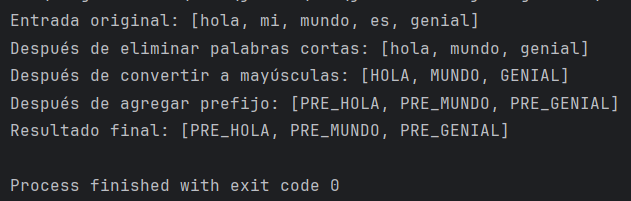


* + Este filtro realiza una última transformación, completando el flujo de datos.



* + Este es el **sumidero (sink)** del patrón, donde se recoge el resultado final después de que los datos han pasado por todas las etapas.

**Ejecución:**



**Por qué es un ejemplo del patrón:**

* Los filtros (FiltroEliminarPalabrasCortas, FiltroConvertirMayusculas, FiltroAgregarPrefijo) son independientes y realizan tareas específicas.
* La clase Pipeline actúa como la tubería, conectando los filtros y gestionando el flujo de datos.
* Los datos fluyen unidireccionalmente desde la entrada hasta la salida final.

**Ventajas de este diseño:**

* Cada filtro puede modificarse o reemplazarse sin afectar a los demás.
* Es fácil agregar nuevos filtros, como uno que invierta palabras o elimine vocales.
* El código es modular y reutilizable en otros contextos.