PIPERS AND FILTERS

Universidad del Cauca

Vigilada Mineducación

Ingeniería de Software II

Presentado por:

Isabella Solarte Sandoval

Daniela Riascos Urrego

Andres Felipe Ocampo Chaguendo

Andrés Felipe Herrera Artunduaga

Carolina Solarte Lopez

Daniel Santiago Muñoz Rodríguez

Docente:

Julio Ariel Hurtado Alegria

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones

Ingeniería de Sistemas

Popayán, Diciembre de 2022

Índice

Introducción

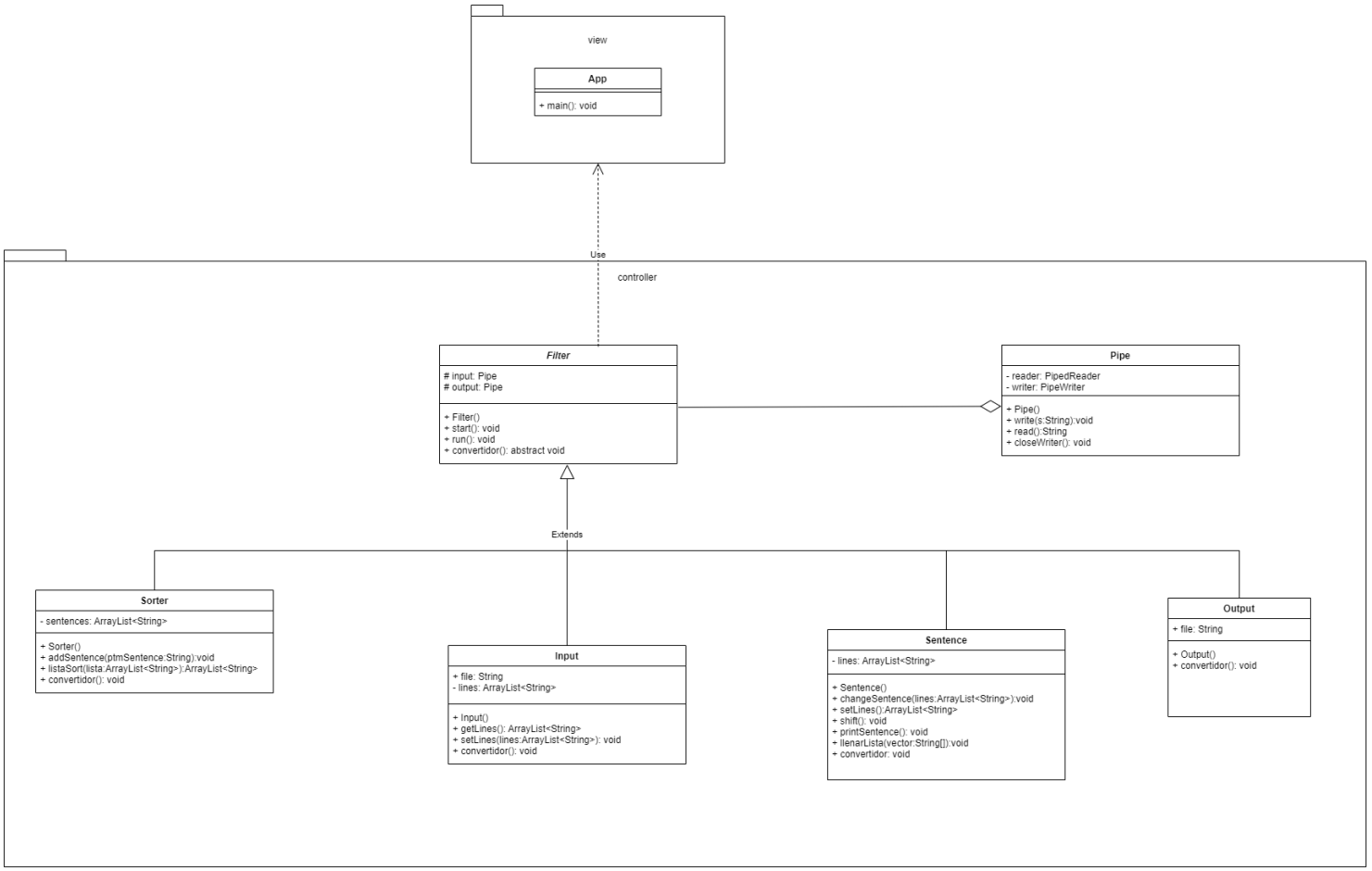
Los patrones arquitectónicos son considerados como plantillas para arquitecturas de software específicas, que determinan las propiedades estructurales de una aplicación y tienen un impacto en la arquitectura de subsistemas.
El patrón de tuberías y filtros consiste en ir transformando un flujo de datos en un proceso comprendido por varias fases secuenciales, siendo la entrada de cada una la salida de la anterior, este patrón satisface la necesidad de que muchos tipos de transformaciones ocurren repetidamente en la práctica,y por tanto es deseable crearlos como partes independientes y reutilizables.
Con la ayuda de este patrón se va a dar solución a la problemática KWIC (KeyWord In Context) y se analizará la solución respecto a las cualidades del software.

Problemática

Palabras claves en contexto (Problema KWIC) Un ejemplo clásico para ilustrar las diferencias entre los patrones de arquitectura es el problema KWIC (KeyWord In Context) introducido por Mary Shaw y David Garlan en 19961 . El problema de palabra clave en contexto toma como entrada una lista de líneas de texto compuesta de palabras. El resultado de la solución del problema debe ser todas las “rotaciones circulares” de las líneas ordenadas alfabéticamente. Una rotación circular se realiza eliminando repetidamente la última palabra y agregándola al principio de la línea.

UML

1. El diseño arquitectónico básico con el patrón arquitectónico asignado al equipo. Debe crearse un modelo según el tipo de vista UML correspondiente:



2. Implementación ejecutable en Java (o su lenguaje orientado a objetos favorito) con correspondencia a la arquitectura planteada.

3. Analizar las ventajas y desventajas del patrón para lograr las siguientes cualidades de la solución: modificabilidad, disponibilidad, desempeño y seguridad. Identifique al menos cuatro tácticas presentes en la solución para las cualidades que el patrón favorece(Drivers). En este punto apóyese en el libro “Arquitectura de Software en la Práctica”.

Ventajas

1. Nos brindó flexibilidad cambiando y reconfigurando los filtros.
2. Se puede reutilizar los componentes, por lo que nos ahorramos más líneas de código y tiempo para poder seguir con más funciones.
3. Nos permitió el procesamiento paralelo por tener el uso de hilos, y así resolver problemas en un tiempo más razonable.
4. El sistema puede beneficiarse de la distribución de los pasos de procesamiento entre diferentes servidores.

Desventajas

1. Hay pérdida de eficiencia, es bastante difícil implementar ciclos o bucles ya que la mayoría de su procesamiento se da de manera secuencial o en paralelo
2. Problemas de sincronización y de la ejecución misma de la arquitectura, no sirve para situaciones interactivas
3. La independencia de filtros puede haber duplicación de funciones si no están bien establecidos cada uno de los componentes y dificulta el manejo de errores.
4. Tener un gran número de filtros independientes puede agregar importantes cantidades de sobrecarga computacional.

Tácticas Presentes Del Patrón

1. Al usar los patrones modelo vista controlador y tuberías y filtros pudimos sacarle provecho a la comunicación entre ellos para acceder a la capa de almacenamiento de datos de manera más eficiente.
2. Al identificar que debemos hacer una estructura de sistemas que producen y procesan una secuencia de datos fue necesario hacer el uso del patrón tuberías y filtros
3. Resultó más cómodo hacer uso del patrón modelo vista controlador, patrón que facilita la modificación e incluso sustitución de cualquiera de ellos sin afectar al resto
4. Junto con el paquete java.io se logró realizar la tarea de poder controlar estos flujos con cierta abstracción sobre el origen de los mismos.

Conclusión

El patrón modelo vista controlador tiene una fácil organización, puesto que solo cuenta con tres componentes. Es un patrón que se puede adaptar a diferentes framework y del patrón tuberías y filtros se puede escalar fácilmente, los filtros no se conocen entre sí, sólo deben estar de acuerdo en el formato de los datos que reciben o que producen.

El patrón modelo vista controlador facilita la modificación e incluso sustitución de cualquiera de ellos sin afectar al resto. Todas estas virtudes contribuyen a simplificar el diseño de aplicaciones complejas que, de otra forma, resultan mucho más difíciles de abordar y mantener.