Brais Sierra García

Ismael Sierra García

Xosé Antonio Silva González

Identifica las diferentes responsabilidades que existen en este programa:

- Pedir nombre de fichero de entrada:

- Pedir nombre de fichero de salida.

- Leer el fichero de entrada.

- Transformar el fichero de entrada en el formato de salida (en este caso XML).

- Escribir el documento XML en el fichero de salida.

Refactoriza la aplicación para que:

La interfaz de usuario pueda ser reutilizada en un proyecto que haga otra cosa dados dos ficheros (la interfaz se encarga de pedir los ficheros). ¿Qué patrón o patrones SOLID has empleado, cómo y por qué?

- Single Responsibility principle (SRC), este patrón especifica que cada clase debe tener una responsabilidad, en este caso la de leer dos archivos, uno de entrada y otro de salida. Mediante la creación de una UI (User Interface) con esta responsabilidad aumentas la cohesión de la clase.

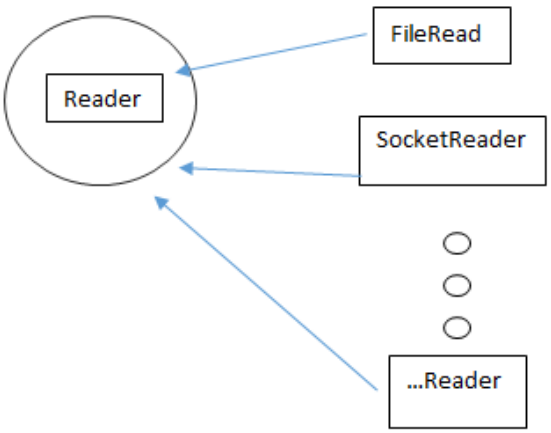
El origen de datos (ahora ficheros) pueda ser distinto y/o el destino de los datos también. ¿Qué patrón o patrones SOLID has empleado, cómo y por qué?

- Single Responsibility principle (SRC), con una responsabilidad por clase, ya que cada responsabilidad puede verse obligada a cambiar:

FileReader: lee el fichero de entrada desde el origen.

FileWriter: escribe el documento final en el fichero de salida.

- Open/Closed principle (OCP), al crear la interfaz Reader/Writer somos capaces de cambiar lo que hacen los módulos sin cambiar código existente, solo añadiendo nuevo. Pudiendo tener diferente origen/destino de los datos.



La transformación que se hace de la entrada pueda ser otra representación basada en texto cualquiera (no a XML). ¿Qué patrón o patrones SOLID has empleado, cómo y por qué?

- Single Responsibility principle (SRC), con una responsabilidad por clase, en este caso la transformación de los datos de entrada.

- Open/Closed principle (OCP), con la interfaz Transform que posibilita la aparición de diferentes formas de implementación de la transformación del documento de entrada a otro formato (no solo a XML).

1. Elabora una tabla para documentar las responsabilidades de las clases

Clase	Responsabilidad
UI	Pedir nombres de archivo de entrada/salida
FileReader	Leer fichero de entrada
FileWriter	Escribir documento transformado en fichero de salida
ToXML	Transformar fichero de entrada en formato deseado (XML)
Converter	Es la clase encargada de invocar los métodos de las interfaces Reader, Writter y Transform.

2. Elabora una tabla para documentar el principio OCP (Open Closed Principle) en tu código. Una clase que respeta el principio OCP, está cerrada para modificación, pero abierta para extensión, siempre centrándonos en una modificación futura concreta. Por lo tanto, la tabla debe contener estas columnas.

Clase	Modificación posible	Punto de extensión
FileReader	Añadir un nuevo origen de datos	Reader
FileWriter	Añadir un nuevo destino de datos	Writer
ToXML	Transformar datos a otro formato	Transformer

3. Modifica la aplicación para que la salida se produzca por pantalla y no a fichero. ¿Tuviste que cambiar código existente a mayores que el método *main*? Describe brevemente la modificación.

   Si. Añadimos una clase denominada ScreenWritter, que hereda los métodos de la interfaz Writter, a través de la cual tenemos un método write() diferente al de FileWritter, ya que nos imprime el resultado por pantalla.