PARTE 1. SOLID

DAGSS\_2

Currás Rodríguez, Alexandre

Vila Fernández, David

# Cuestiones

### ¿**Qué** principio o principios SOLID has empleado para cada uno de los dos objetivos anteriores, **cómo** los has empleado y **por qué**?

* **El origen de datos (ahora ficheros) pueda ser distinto y/o el destino de los datos también.**

Usamos el principio “Open/Closed Principle” creando las interfaces *Reader* y *Writer* que se encargan de obtener el origen y el destino de los datos, implementado una clase para según qué tipo de lectura o escritura.

El objetivo de esta decisión es poder añadir o eliminar distintos orígenes y/o destinos de datos sin modificar ninguna clase. De esta manera, si por ejemplo, quisiéramos leer los datos de una base de datos, valdría con crear una clase *BDReader* que implemente la interfaz *Reader*.

* **La transformación que se hace de la entrada pueda ser otra representación basada en texto cualquiera (no a XML).**

Usamos el principio “Open/Closed Principle” creando la interfaz *Transformer*, de manera que basta con crear una clase que implemente esta interfaz para cada tipo de transformación, al igual que lo hace la clase TABtoXMLTransformer.

De manera general, de modo que se aplica a ambos objetivos, también utilizamos el principio Single Responsibility Principle (SRP), asegurando coherencia en lo que hace cada clase, de manera que si se produce algún cambio en algún requisito, no afecte al resto de funcionalidades.

Finalmente, podemos destacar el uso del Dependency Inversion Principle (DIP), de forma que, en el método “main”, instanciamos las clases a utilizar. Es decir, las implementaciones de las interfaces que necesitamos para conseguir el objetivo de la ejecución del programa.

### Elabora una tabla para documentar las responsabilidades de las clases

|  |  |
| --- | --- |
| Clase | Responsabilidades |
| Converter | Dirigir el proceso de conversión |
| ConverterApp | Dirigir el programa |
| ConverterAppCLI | Pedir fichero entrada |
| ConverterAppCLI | Pedir fichero salida |
| ConverterAppCLI | Mostrar errores |
| FileReader | Mostrar error de fichero no existente |
| FileReader | Leer datos de fichero |
| Reader | Leer origen de datos |
| TABtoXMLTransformer | Convertir TAB a XML |
| Transformer | Transformar datos |
| FileWriter | Mostrar error de fichero no creable |
| FileWriter | Escribir datos en fichero |
| Writer | Escribir en destino de datos |

### Elabora una tabla para documentar el principio OCP en tu código

|  |  |  |
| --- | --- | --- |
| Clase | Modificación posible | Punto de extensión |
| FileReader | Leer de un origen distinto | Reader |
| FileWriter | Escribir en un destino distinto | Writer |
| TABtoXMLTransformer | Transformar otro tipo de datos | Transformer |

### 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.

No, solo hizo falta crear una nueva clase (*ConsoleWriter*) que implementara la interfaz *Write*.

La modificación consistió en cambiar la clase FileWriter que instanciaba el Writer por ConsoleWriter.