PARTE 1: SOLID

Alejandro Gómez González

1) Identifica las diferentes responsabilidades que existen en este programa. Haz una lista con ellas.

1. Pedir fichero lectura de datos a usuario.
2. Leer de fichero.
3. Pedir fichero de escritura a usuario.
4. Escribir en fichero.
5. Transformar cadenas de texto formato de tabuladores a XML.
6. Manejo de excepciones.

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

Single-responsibility principle (SRP)

El principio SRP se ha aplicado porque se ha dividido la clase anterior que tenia muchas (en este caso todas) las responsabilidades del programa en la misma clase, ahora cada clase tiene su única responsabilidad. Las responsabilidades se han dividido de la siguiente forma:

Clase	Responsabilidad
UI	Pedir información a usuario
FileReader	Leer de fichero.
FileWriter	Escribir en fichero.
TABToXMLTransformer	Transformar cadenas de texto separadas por tabulador a formato XML.
ConverterApp	Manejo de excepciones.
TextTransformer	Maneja la lectura de datos y escritura en el medio correspondiente.

Open–closed principle (OCP)

El principio OCP se ha aplicado añadiendo las interfaces Writer, Reader y Transform. De modo que ahora el programa puede añadir nuevas formas de lectura, escritura y

transformación de datos siendo ahora abierta para extensión y cerrada para modificación.
Las clases que implementan estas interfaces son: FileWriter,FileReader y TextTransformer.

Liskov substitution principle (LSP)

Este principio se aplica ya que si se borramos las implementaciones de Reader, Writer y Tranform, estas interfaces serían capaces de sustituir a las implementaciones ya las clases concretas no añaden funcionalidad o requisitos con respecto a las interfaces.

Interface segregation principle (ISP)

El principio ISP se aplica para que cada cliente puede acceder a la interfaz que quiera sin depender de otras. En este caso, se han creado las interfaces FileReader y FileWriter, cada una independiente del resto, si se produjese un cambio en la interfaz el cambio en la totalidad del programa tiene un efecto mucho menor.

Dependency inversion principle (DIP)

El principio DIP se ha aplicado al crear la clase Tranform, la cual no depende de las clases FileReader y FileWriter , sino que utiliza sus interfaces para comunicarse con ellas.

3) 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
TextTransformer	Cambiar el origen de los datos	Interfaz Reader
TextTransformer	Cambiar el destino de los datos	Interfaz Writer
TextTransformer	Transformación distinta de TAB->XML	Interfaz Transform

4) 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.

Hay que crear la clase ScreenWriter y cambiar en el Transformer el FileWriter por la nueva clase.