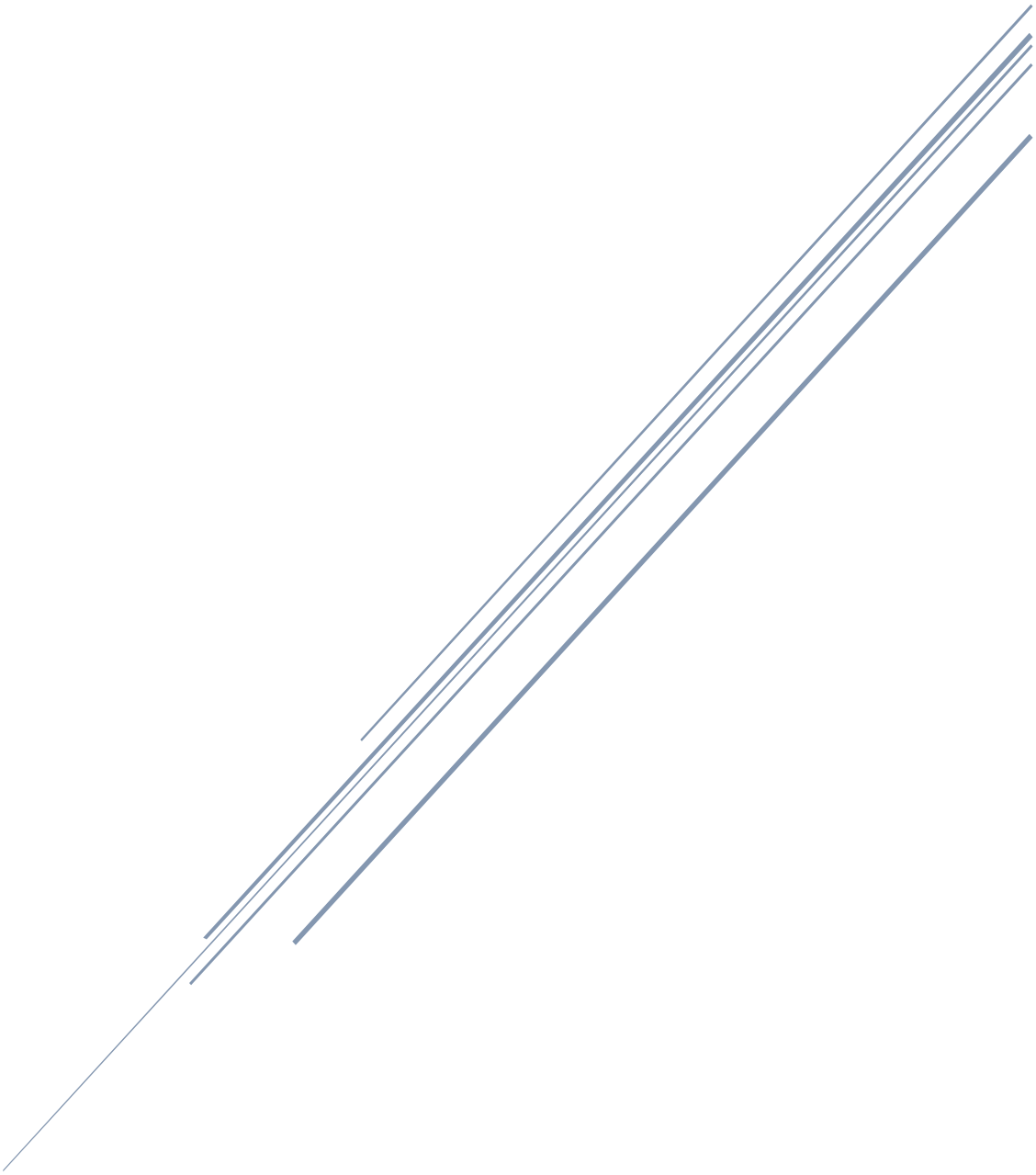
SOLID

DAGSS



ESEI
Óscar Lestón Casais

Índice

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

-Interactuar con el usuario para pedir nombres de ficheros

-Fichero de salida

-Fichero de entrada

-Transformar el contenido del fichero(en el caso inicial a XML)

-Llevar el flujo principal del programa

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

Utilizo SRP para identificar las clases mínimas que debo crear según la responsabilidad que tienen.

Utilizo OCP para identificar que clases cerrar y donde dejar una interfaz punto de expansión. En concreto toXML pasa a ser una clase cerrada a modificación con un punto de expansión en la interfaz que implementa, de forma análoga se actúa con la responsabilidad de "origen de ficheros".

Se utiliza el principio de sustitución de Liskov al permitir substituir las clases que llama el main por cualquiera de sus subclases o implementaciones.

Usamos el principio de segregación de interfaces haciendo que entrada y salida tengan interfaces distintas a pesar de poder ser consideradas la misma responsabilidad, si quieres cambiar la forma de salida y no la de entrada no debes tener que implementar un método ya implementado simplemente porque la interfaz te obligue

Al crear dependencias en interfaces y no en clases concretas estamos aplicando el principio de inversión de dependencias.

Elabora una tabla para documentar las responsabilidades de las clases

Paquete	Clase	Responsabilidad
Converterapp	ConverterApp	Flujo Principal Del programa
Converterapp	FormatterInterface	Punto de expansión para distintos formatos
Converterapp	GetterInputInterface	Punto de expansión para métodos de entrada
Converterapp	GetterOutputInterface	Punto de expansión para métodos de salida
Formatters	FormatterToXML	Escribir en la salida la entrada convertida a XML
IO.Input	GetterInputFileConsoleInput	Generar objetos File a partir de la entrada por consola
IO.Output	ShowOutputByConsole	Expulsar la salida por consola
IO.Output	GetterOutputConsoleInput	Expulsar la salida en el fichero que pide por consola

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	P.extensión
FormatterToXML	Convertir a otro formato que no sea XML	FormatterInterface
GetterInputFileConsoleInput	Entrada especificada de otra forma. Por ejemplo, por parámetros del main en vez de consola.	GetterInputInterface
GetterOutputConsoleInput Y ShowOutputByConsole	Salida especificada de otra forma. Por ejemplo, un correo electrónico.	GetterOutputInterface

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.

```java
public static void main(String args[]) {
    GetterInputFileInterface getterInputFile = new GetterInputFileConsoleInput();
    GetterOutputInterface getterOutputFile = new ShowOutputByConsole();

    Scanner scanner = getterInputFile.getInputFile();
    PrintStream out = getterOutputFile.getOutput();

    FormatterInterface formatter = new FormatterToXML();
    formatter.format(scanner, out);
    out.close();
}
```

```java
public static void main(String args[]) {
    GetterInputFileInterface getterInputFile = new GetterInputFileConsoleInput();
    GetterOutputInterface getterOutputFile = new GetterOutputConsoleInput();

    Scanner scanner = getterInputFile.getInputFile();
    PrintStream out = getterOutputFile.getOutput();

    FormatterInterface formatter = new FormatterToXML();
    formatter.format(scanner, out);
    out.close();
}
```

```java
public PrintStream getOutput() {
    System.out.println(x: "Output will be shown by console: ");
    PrintStream toret = new PrintStream(System.out);
    return toret;
}
```

```java
public PrintStream getOutput() {
    System.out.println(x: "output filename: ");
    File file = getFile();
    PrintStream toret = null;
    try {
        toret = new PrintStream(new FileOutputStream(file));
    } catch (FileNotFoundException e1) {
        System.err.println("the file " + file.getAbsolutePath()
                + " does cannot be created: " + e1.getMessage());
        System.exit(status: 1);
    }
    return toret;
}
```