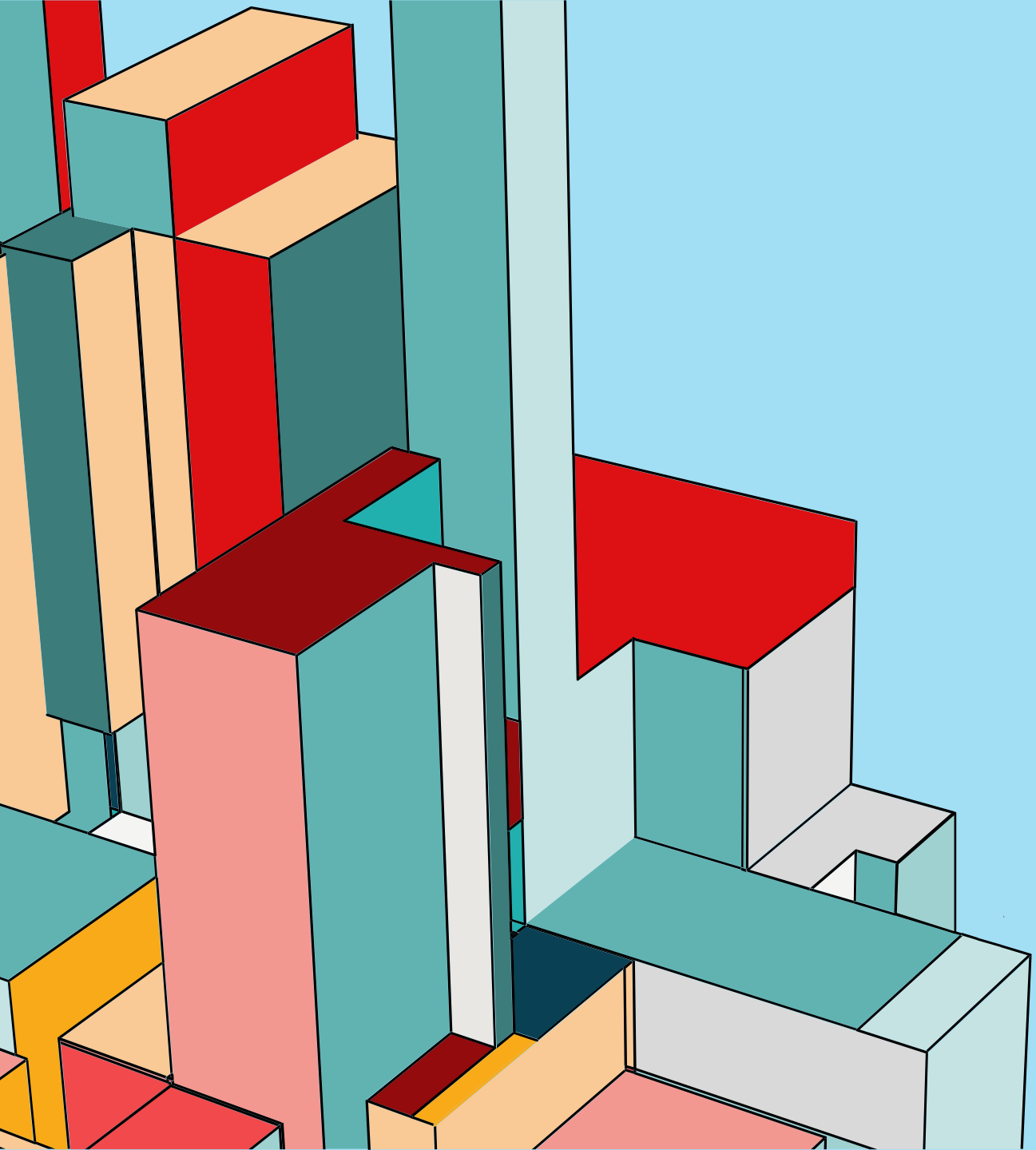


FACTORY METHOD

Santiago Andrés Benavides Coral
20232020036
Modelos de programación 020-86

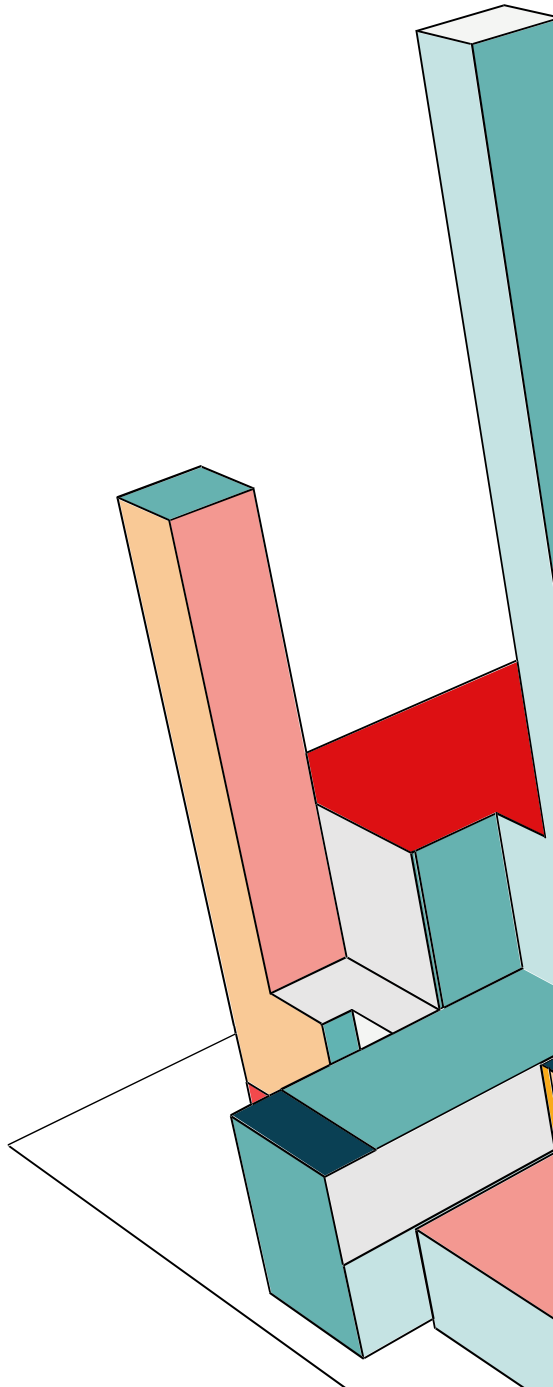
ESCENARIO PROBLEMA

Una empresa editorial recibe constantemente documentos en diferentes formatos: PDF, Word y Excel.

Estos documentos deben ser procesados para tareas internas como:

- Validación del contenido.
- Extracción de metadatos (título, autor, número de paginas/hojas).
- Generación de reportes

El sistema debe poder crear objetos de tipo Documento sin que el código principal dependa de clases concretas como:
PDFDocument, WordDocument o ExcelDocument.



ESCENARIO PROBLEMA

```
// Código con problemas de acoplamiento
if (tipo.equals("PDF")) {
    return new DocumentoPDF();
} else if (tipo.equals("WORD")) {
    return new DocumentoWord();
} else if (tipo.equals("EXCEL")) {
    return new DocumentoExcel();
}
```

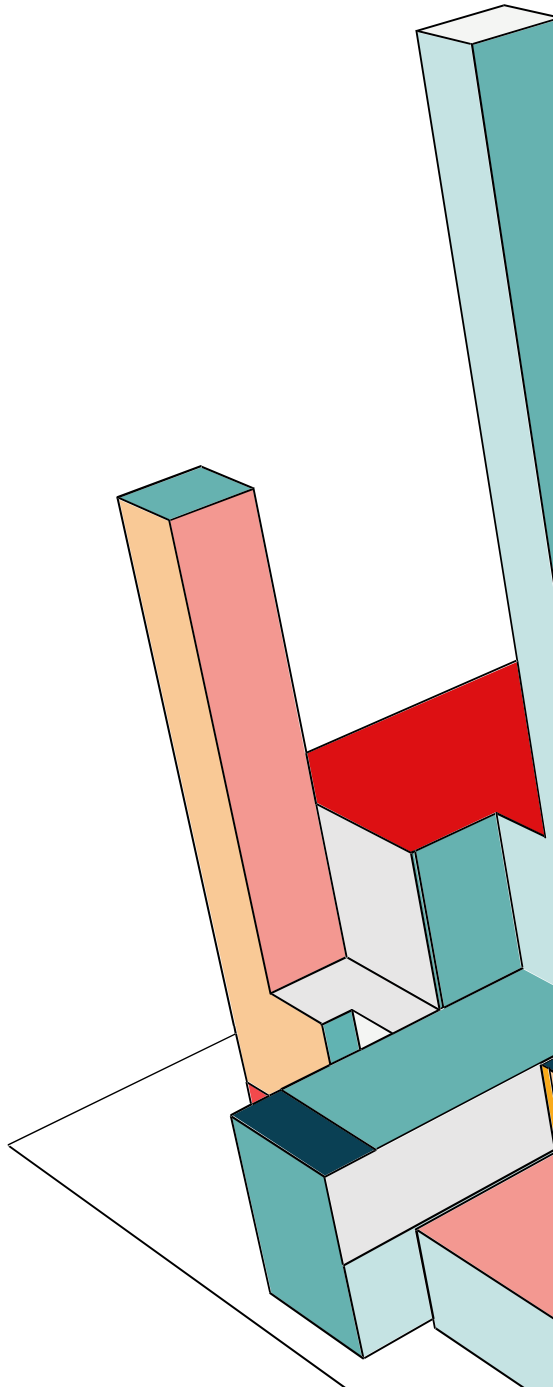
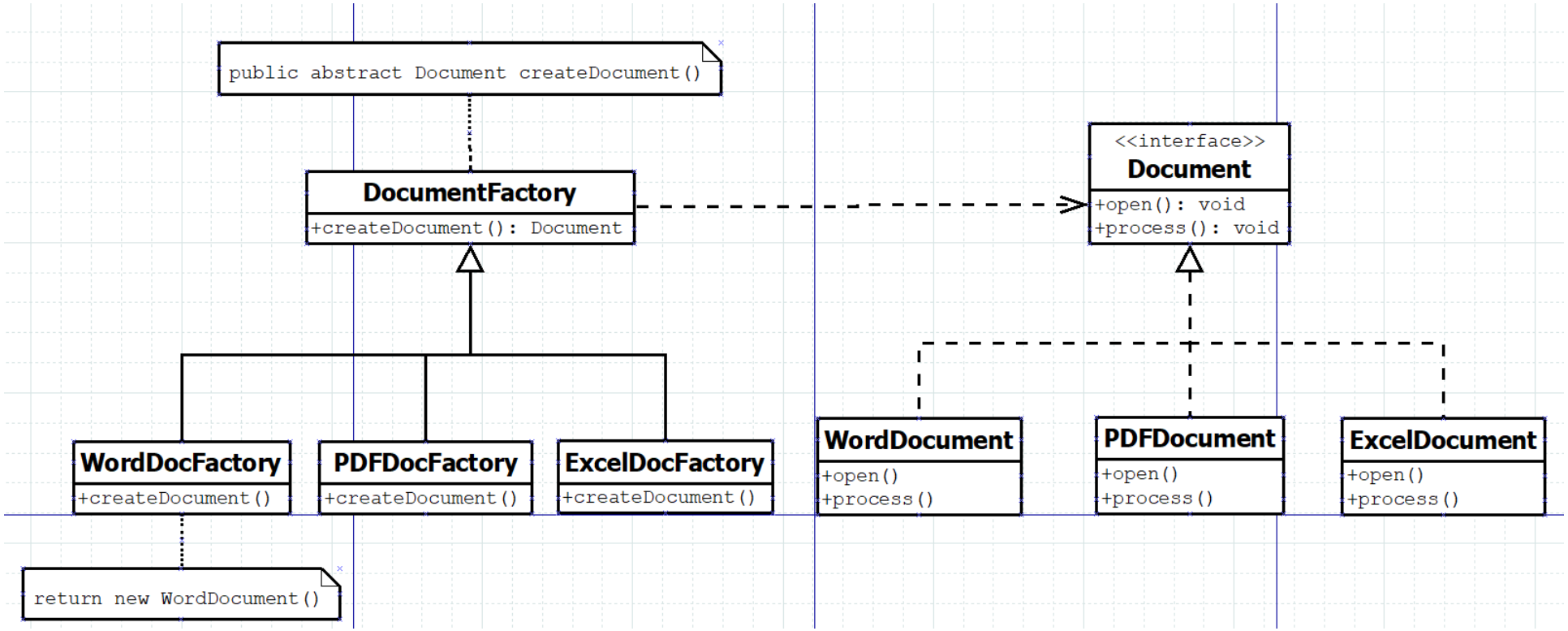
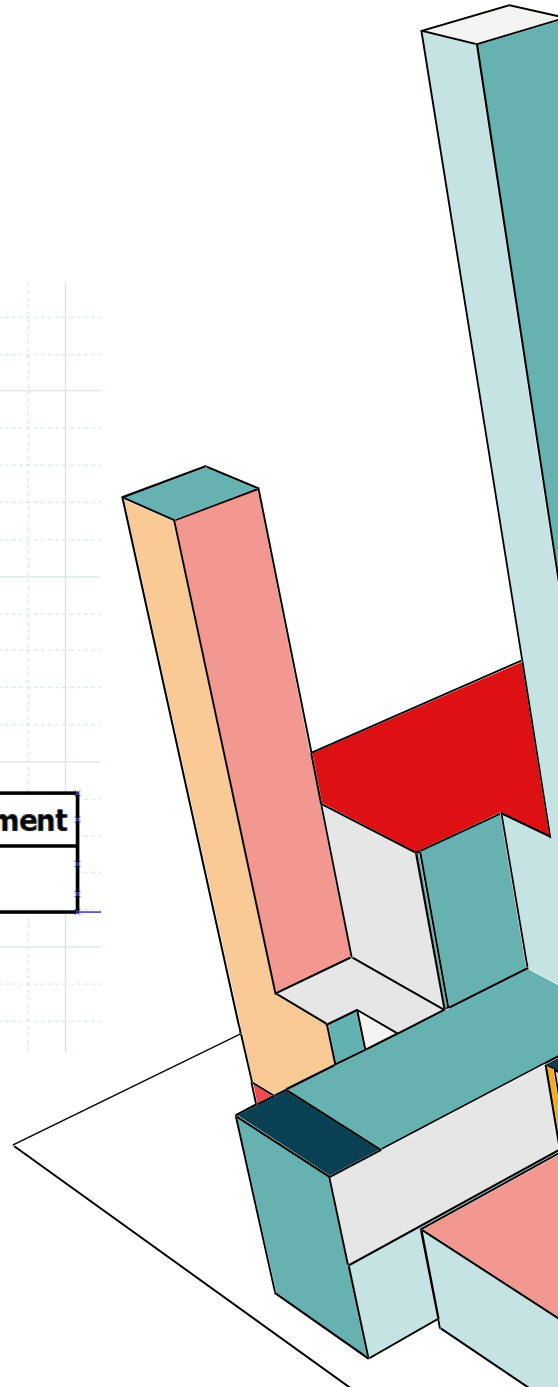


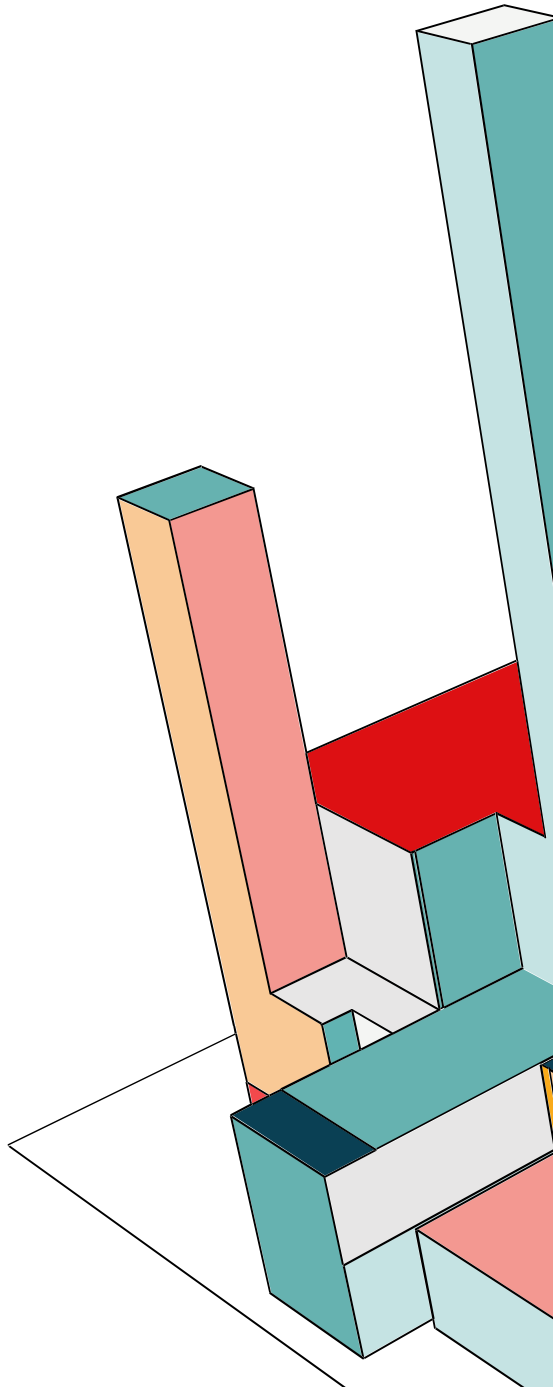
DIAGRAMA UML DE LA SOLUCIÓN VS PATRÓN





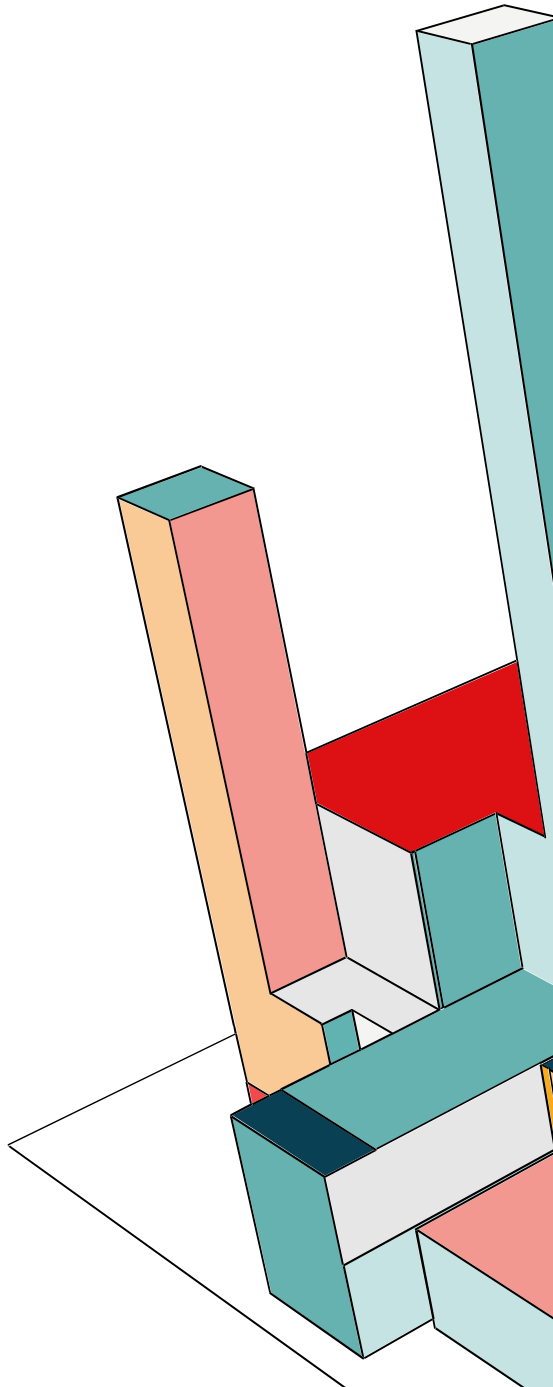
USOS CONOCIDOS DEL PATRÓN

1. Frameworks como Spring: Usan fábricas para crear beans de diferentes tipos → Cuando un bean es solicitado, el framework aplica el patrón para decidir qué implementación devolver.
   Fuente: https://docs.spring.io/spring-framework/docs/6.0.3/reference/html/core.html
   Fuente: https://docs.spring.io/spring-framework/reference/core/beans/introduction.html
2. APIs de conexión (JDBC): *DriverManager.getConnection(url)* devuelve la implementación correcta de *Connection* para la base de datos especificada en la URL.
   Fuente: https://docs.oracle.com/javase/8/docs/api/java/sql/DriverManager.html#getConnection-java.lang.String-



USOS CONOCIDOS DEL PATRÓN

1. Colecciones Java: *collection.iterator()* devuelve la implementación específica de *Iterator* para ese tipo de colección (ArrayList, LinkedList, etc.).
   Fuente:
   https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html#iterator--



¡MUCHAS GRACIAS!