Trabajo 1 – Patrones y diseño de software

Objetivo: Demostrar la aplicación de principios SOLID y patrones creaciones.

**Punto 1  
  
Parte A:  
.** Definición del problema que se desean abordar (25%)

a) Describir uno o varios problemas donde se puedan implementar tres patrones creacionales distintos, justificando porqué la aplicación del patrón da solución a los problemas planteados.

Los problemas pueden estar relacionados o ser independientes entre sí, la temática es libre. (Se esperan en total tres patrones creacionales justificados).

**Solución Parte A:**

1. **Object pool:** <https://thepowerups-learning.com/patrones-de-diseno-object-pool/>

En los juegos tenemos la problemática de que instanciar o crear los objetos es una operación pesada y requiere de memoria dinámica.

Si se empieza a crear y destruir objetos, es posible que empecemos a sufrir problemas de fragmentación de memoria.

Con el **object pool** se busca reciclar los objetos, es decir se crea una vez y luego se reutilizan apagándolos cuando no sean necesarios.

En este caso el patrón es útil, puesto que no se esta creando y destruyendo en memoria con cada objeto nuevo, si no que se reutilizara los objetos disponibles preexistentes, reduciendo la instanciación de los objetos y mejorando su proceso

**Parte B. Diagrama UML de Clases (25%)**

Para cada uno de los problemas definidos en el punto 1, elaborar el diagrama UML donde se evidencie la estructura del patrón de diseño, explicando la correspondencia con los participantes.

Es importante que los nombres de las clases se adapten al dominio, es decir, no dejar nombres de clases genéricas o igual a como se presentan en el material de estudio.

Se esperan tres diagramas diferentes o un único diagrama que implemente los tres patrones seleccionados.

**Solución parte B:**

1. **Object pool**

Interfaz de usuario gráfica, Aplicación, Sitio web

El contenido generado por IA puede ser incorrecto.

**Parte C. Implementación de los patrones diseño (25%).**

Implementar en el lenguaje de su elección las clases y los objetos que hacen parte de los tres patrones creacionales, evidenciando un correcto uso de estos y siguiendo principios SOLID.

No es necesario implementar funcionalidades, es suficiente con mostrar mensajes por consola donde se evidencie el funcionamiento del patrón.

**Punto 2 (25%).**

Se requiere modelar un sistema para construir documentos en formato PDF, asegurando la flexibilidad en la creación de los documentos entre lo que se incluye la generación de documento a partir de plantillas.

La aplicación deberá permitir al usuario seleccionar el tipo de formato de origen con el que desea trabajar (por ejemplo, XLSX, DOCX, o XML) y posteriormente generar documentos complejos de manera programática, incluyendo adición de múltiples columnas, tablas, títulos, estilos, entre otros.

**Requisitos:**

1. El sistema debe permitir seleccionar el tipo de documento con el que se desea trabajar. El sistema debe funcionar independiente del formato.
2. Los documentos deben ser construidos de manera flexible y permitir agregar columnas, tablas, enunciados, estilos, entre otros, en tiempo de ejecución.
3. La solución debe ser fácilmente extensible para soportar nuevos formatos en el futuro sin modificar el código existente.
4. El código debe estar organizado de forma que el cliente de la aplicación no necesite conocer los detalles de lectura y escritura de los formatos, ni de la generación del código fuente.

Entregables:

* Diagrama de Clases de UML de la solución en el que se evidencie el uso de al menos dos patrones.

Interfaz de usuario gráfica, Aplicación, Teams

El contenido generado por IA puede ser incorrecto.

* Explicación de cómo se abordaría este problema, justificando la elección del diseño.

Teniendo en cuenta los requisitos iniciales planteados en el problema, se toma la decisión técnica de implementar dos patrones en el diseño de la solución. Las razones para la implementación de estos se dejan a continuación plasmada.

El **Factory Method** es una elección natural para este tipo de problema, ya que permite que la aplicación sea flexible respecto a los formatos de entrada (XLSX, DOCX, XML, etc.). sin tener que modificar el código cuando se agregan nuevos formatos.

El **Builder** es ideal para construir objetos complejos de manera flexible, sin exponer la complejidad de su construcción al cliente. En este caso, se utiliza para construir el documento PDF de manera modular y ordenada:

Se cumple el principio de **open/closed** de SOLID, ya que, para añadir nuevos formatos o nuevas formas de construir un documento, se extiende el sistema con nuevas clases sin alterar las existes. Así mismo se cumple con principio de single responsability.

El diseño adoptado combina patrones de diseño orientados a objetos que favorecen la **flexibilidad**, la **extensibilidad** y la **separación de responsabilidades**. Los patrones **Factory Method** y **Builder** permiten que el sistema sea fácilmente adaptable a futuros cambios, como la adición de nuevos formatos o la personalización de la construcción de documentos, sin afectar el código existente. La solución está diseñada para ser simple de entender y fácil de mantener a largo plazo, cumpliendo con los requisitos planteados en el problema.

* Código fuente de la aplicación donde se evidencie la interacción de los patrones, no es necesario implementar las funcionalidades.

**Plazo de entrega:**

Sábado 22 de marzo.

El trabajo deberá ser enviado a través de Teams, pueden usar repositorios como github, gitlab o Azure Repos para el código, en el documento deberá aparecer el enlace al repositorio, el repositorio deberá ser público. El trabajo será presentado en horario de clase.

El documento debe contener el nombre de los integrantes del grupo. Máximo 3 personas por grupo, los grupos son de conformación libre.