Clase 2 - Adapter

Patrones de diseño

Los patrones de diseño son soluciones reutilizables a problemas comunes en el diseño de software. Son guias o estructuras que ayudan a organizar el codigo de manera mas eficiente y mantenible.
Los patrones no son solamente buenas soluciones, sino debe ser una solucion que permita reutilizarla, mantenerla y debe estar organizada.

Un patrón de diseño es una forma de programar siguiendo un conjunto de reglas que promueven buenas prácticas y que pueden ser reutilizadas en distintos proyectos.

✅ Son **soluciones probadas** a problemas comunes en el desarrollo de software.
✅ Mejoran la **reutilización**, **mantenimiento** y **organización** del código.

Clases singleton: Una clase singleton es una clase que solo permite una unica instancia en todo el programa y proporciona un metodo global para acceder a ella.

🔷 ¿Para qué sirve?

Se usa cuando **necesitas una única instancia de un objeto** en toda la aplicación, como:
✅ Conexión a una base de datos.
✅ Registro de logs.
✅ Administrador de configuraciones.

📌 **Beneficios:**
✓ Evita crear múltiples objetos innecesarios.
✓ Asegura que hay una única instancia centralizada.
✓ Ahorra memoria y recursos en la aplicación.

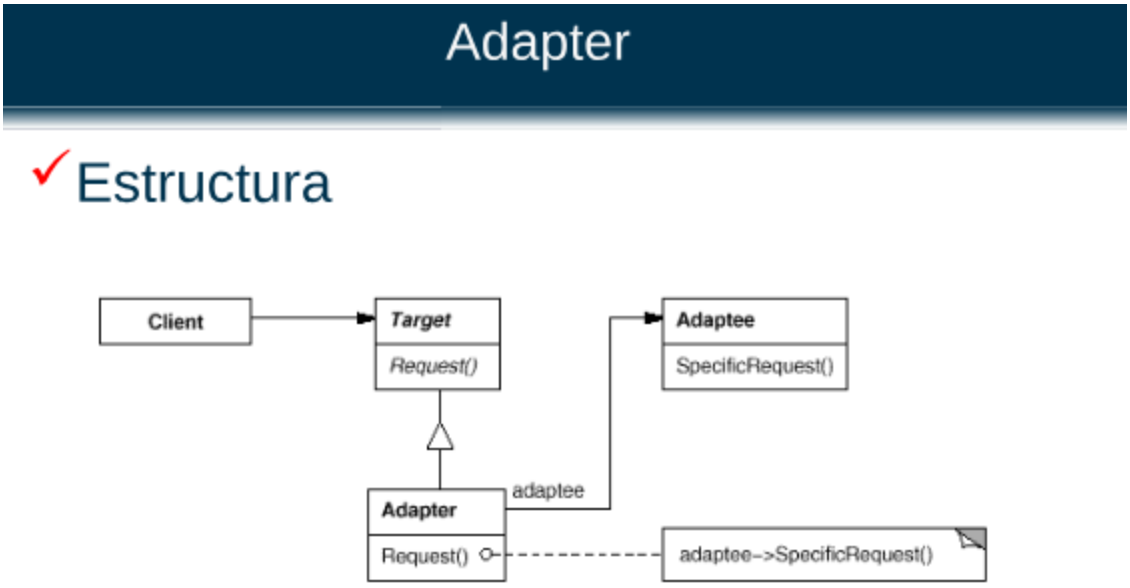
📌 **Posibles problemas:**
⚠ Puede generar problemas en entornos multihilo si no se maneja correctamente.
⚠ Puede romper el principio de responsabilidad única si se usa mal.

Adapter (adaptador):

Es un patron de diseño estructural que permite que dos clases con interfaces incompatibles trabajen juntas sin necesidad de modificar su codigo. Actua como traductor entre dos sistemas diferentes. Se usan cuando tenemos una clase que no puede cambiar, pero que necesitamos que funcione con otra clase que espera una interfaz distinta.

Definicion de la catedra:

Intencion: Convertir la interfaz de una clase en otra que el cliente espera. El adapter permite que ciertas clases trabajen en conjunto cuando no podrian tener interfaces incompatibles



Participantes del diagrama adapter
Target (Figura): Define la interfaz especifica que usa el cliente
Cliente (Editor): Colabora con objetos que satisfacen la interfaz de target
Adaptee (3DFigure): Define una interfaz que precisa ser adaptada
Adapter (3DAdapter): Adapta la interfaz del adaptee a la interfaz de Target

Metodo Template

Es un patron de diseño de comportamiento que define la estructura de un algoritmo en un metodo de una clase base, dejando algunos pasos sin implementar para que las subclases los definan. Este paton permite que la logica general de un algoritmo permanezca en la clase base, mientras que las partes especificas puedan ser implementadas por las subclases.

🔷 Definición

Un **Método Template** es un método en una **clase abstracta** que **define el flujo de un algoritmo**, permitiendo que las subclases personalicen ciertos pasos sin alterar la estructura general.

Conclusion propia: Separa,descompone el algoritmo en pasos bien definidos, donde cada funcionalidad sea independiente. Esto fomenta la reutilizacion, la separacion de responsabilidades y un codigo mas mantenible.

📌 **Principio clave → "Definir el esqueleto de un algoritmo y delegar los detalles en subclases"**

```
// Método template: define los pasos del algoritmo
public final void prepararBebida() {
    hervirAgua();
    agregarIngredientePrincipal();
    servir();
    agregarExtras();
}
```

✓ Structure