DISEÑO DE SISTEMAS

Juan Carlos Molina Lozano

Docente

CONTENIDO

- Introducción
- Objetivos de la Clase
- Propósito de los Patrones Estructurales
- Características de los Patrones Estructurales
- Ventajas y Desventajas

OBJETIVOS DE LA CLASE

- Comprender qué son los patrones estructurales.
- Identificar y explicar sus características, ventajas y desventajas.
- Aplicar al menos dos patrones estructurales en el diseño orientado a objetos.
- Resolver un problema utilizando un patrón estructural apropiado.

INTRODUCCIÓN

En el diseño de sistemas complejos, no solo importa qué hace un componente, sino cómo se relaciona con los demás. A medida que los sistemas crecen, surge la necesidad de organizar, conectar, adaptar o extender estos componentes de forma clara, coherente y flexible. Aquí es donde entran en juego los Patrones Estructurales. Estos patrones nos ayudan a resolver problemas comunes de diseño relacionados con la estructura del sistema, como:

- Adaptar interfaces incompatibles entre clases.
- Organizar objetos que pueden contener otros objetos.
- Agregar funcionalidades dinámicamente sin modificar clases existentes.
- Controlar el acceso a ciertos objetos de manera segura o eficiente.

QUÉ SON LOS PATRONES ESTRUCTURALES?

Son patrones que se centran en cómo se componen y organizan las clases y objetos para formar estructuras más grandes y flexibles, respetando el principio de bajo acoplamiento.

Propósito:

Facilitar la adaptabilidad, escalabilidad y reutilización del código mediante estructuras estables y coherentes.

Principales Patrones Estructurales

Adapter (Adaptador)

Facade (Fachada)

Proxy (Representante)

Flyweight (Peso mosca, Peso ligero, Cache)

Decorator (Decorador)

Composite (Compuesto)

Bridge (Puente)

PATRONES ESTRUCTURALES

Adapter (Adaptador)

- Propósito:

Permitir que clases con interfaces incompatibles trabajen juntas.

- Ejemplo real:

Un cargador con adaptador europeo.

- Uso en software:

Integrar sistemas antiguos con nuevos módulos.

QUÉ SON LOS PATRONES ESTRUCTURALES

Decorator (Decorador)

- Propósito:

Agregar funcionalidades a un objeto en tiempo de ejecución sin modificar su clase.

- Ejemplo real:

Un café al que le agregas leche, azúcar, etc.

- Uso en software:

Añadir características dinámicamente (como log, seguridad).

QUÉ SON LOS PATRONES ESTRUCTURALES

Facade (Fachada)

- Propósito:

Proporcionar una interfaz simple para un sistema complejo.

- Ejemplo real:

Un control remoto con botones que simplifican muchas funciones.

- Uso en software:

Simplificar subsistemas complicados (como un sistema bancario o de pagos).

QUÉ SON LOS PATRONES ESTRUCTURALES

Composite (Compuesto)

- Propósito:

Permitir tratar objetos individuales y compuestos de la misma manera.

- Ejemplo real:

Carpetas que contienen archivos o más carpetas.

- Uso en software:

Interfaces gráficas, menús jerárquicos.

QUÉ SON LOS PATRONES ESTRUCTURALES

Proxy (Representante)

- Propósito:

Controlar el acceso a otro objeto (cargarlo, protegerlo o agregar lógica).

- Ejemplo real:

Un asistente que filtra visitas a un director.

- Uso en software:

Control de acceso remoto, cachés, logs, control de instancias.

QUÉ SON LOS PATRONES ESTRUCTURALES

Bridge (Puente)

- Propósito:

Bridge es un patrón de diseño estructural que te permite dividir una clase grande, o un grupo de clases estrechamente relacionadas, en dos jerarquías separadas (abstracción e implementación) que pueden desarrollarse independientemente la una de la otra..

- Aplicación:

Utiliza el patrón Bridge cuando quieras dividir y organizar una clase monolítica que tenga muchas variantes de una sola funcionalidad (por ejemplo, si la clase puede trabajar con diversos servidores de bases de datos).

- Ejemplo real:

Pensemos en un control remoto (abstracción) que puede funcionar con diferentes tipos de televisores (implementación). Puedes crear nuevos controles o nuevos televisores sin modificar los existentes.

QUÉ SON LOS PATRONES ESTRUCTURALES

Flyweight (Peso mosca, Peso ligero, Cache)

- Propósito

Usar compartición de objetos para soportar un gran número de objetos de forma eficiente en memoria.

- Idea clave:

En lugar de crear miles de objetos iguales, se reutilizan instancias compartidas. Solo se almacenan los datos intrínsecos (compartidos), y los datos extrínsecos (variables) se pasan desde fuera.

- Ejemplo real:

Un editor de texto donde cada letra (carácter) tiene formato. En lugar de crear un objeto para cada letra con su estilo, se comparten los estilos y se aplica según se necesite.

CARACTERÍSTICAS DE LOS PATRONES ESTRUCTURALES

- Enfocados en la composición de objetos y clases.
- Promueven la interfaz común entre elementos heterogéneos.
- Apoyan el principio de diseño abierto/cerrado.
- Fomentan el uso de interfaces y abstracciones.

VENTAJAS Y DESVENTAJAS

Ventajas

- ☑ Permiten la reutilización del código existente.
- ☑ Aumentan la escalabilidad y mantenimiento del sistema.
- ☑ Mejoran la separación de responsabilidades.

Desventajas

- ✗ Pueden aumentar la complejidad del diseño inicial.
- ✗ Exceso de abstracción si se usan incorrectamente.
- ✗ Mayor cantidad de clases e interfaces involucradas.

REFERENCIAS

Refactoring Guru

Patrones estructurales

https://refactoring.guru/es/design-patterns/structural-patterns

PREGUNTAS