Los Patrones de Diseño

¿Qué son?

Los patrones de diseño son soluciones generales, reutilizables y aplicables a distintos problemas de diseño de software. Su propósito es identificar problemas del sistema y proporcionar soluciones apropiadas a problemas generales ya resueltos, gracias al trabajo de desarrolladores anteriores que encontraron soluciones a través de prueba y error.

Es aconsejable emplear patrones de diseño ya que te ayudan a estar seguro de la validez de tu código, debido a que son soluciones que funcionan y han sido probados por muchísimos desarrolladores siendo menos propensos a errores.

Existen 3 categorías de patrones de diseño principales:

- **Patrones creacionales**
- **Patrones estructurales**
- **Patrones de comportamiento**

Patrones Creacionales

Los patrones de creación proporcionan diversos mecanismos de creación de objetos, que aumentan la **flexibilidad y la reutilización del código existente de una manera adecuada a la situación**. Esto le da al programa más flexibilidad para decidir qué objetos deben crearse para un caso de uso dado.

Corresponden a patrones de diseño de software que solucionan problemas de creación de instancias. Nos ayudan a encapsular y abstraer dicha creación:

Abstract Factory

En este patrón, una interfaz crea conjuntos o familias de objetos relacionados sin especificar el nombre de la clase.

Prototype

Permite copiar objetos existentes sin hacer que su código dependa de sus clases. Se utiliza para restringir las operaciones de memoria / base de datos manteniendo la modificación al mínimo utilizando copias de objetos.

Patrones Estructurales

Facilitan soluciones y estándares eficientes con respecto a las composiciones de clase y las estructuras de objetos. El concepto de herencia se utiliza para componer interfaces y definir formas de componer objetos para obtener nuevas funcionalidades.

Son los patrones de diseño software que solucionan problemas de composición (agregación) de clases y objetos:

Bridge

En este patrón hay una alteración estructural en las clases principales y de implementador de interfaz sin tener ningún efecto entre ellas. Estas dos clases pueden desarrollarse de manera independiente y solo se conectan utilizando una interfaz como puente.

Adapter

Se utiliza para vincular dos interfaces que no son compatibles y utilizan sus funcionalidades. El adaptador permite que las clases trabajen juntas de otra manera que no podrían al ser interfaces incompatibles.

Patrones de comportamiento

El patrón de comportamiento se ocupa de la **comunicación entre objetos de clase.** Se utilizan para detectar la presencia de patrones de comunicación ya presentes y pueden manipular estos patrones.

Estos patrones de diseño están específicamente relacionados con la comunicación entre objetos:

Command

Convierte una solicitud en un objeto independiente que contiene toda la información sobre la solicitud. Esta transformación permite parametrizar métodos con diferentes solicitudes, retrasar o poner en cola la ejecución de una solicitud y respaldar operaciones que no se pueden deshacer.

Template method

Se usa con componentes que tienen similitud donde se puede implementar una plantilla del código para probar ambos componentes. El código se puede cambiar con pequeñas modificaciones.