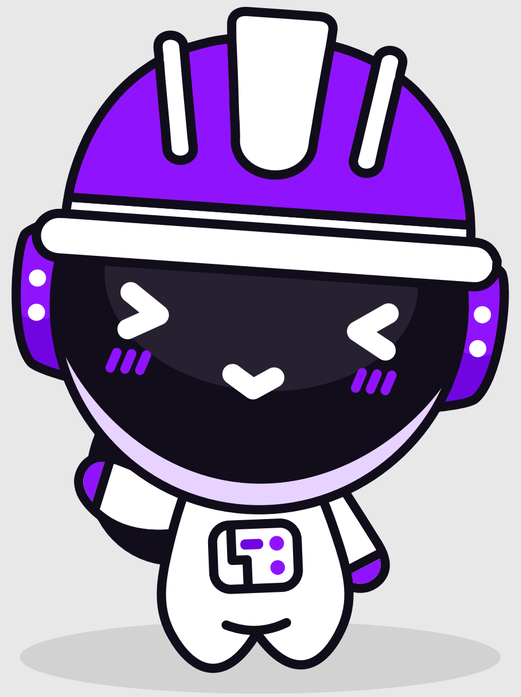
Patrones de diseño estructurales



Estos tipos de patrones se enfocan en cómo objetos y clases pueden ser usados en grandes estructuras sin comprometer la flexibilidad y eficiencia del sistema.

Los principales patrones de diseño estructurales son:

Adapter o Wrapper: este patrón tiene como objetivo permitir que 2 interfaces incompatibles trabajen en conjunto. De esta manera se crea un puente donde la data puede ser transformada a otra interfaz compatible pero sin cambiar el comportamiento.

Bridge: permite separar una clase grande o un conjunto de clases relacionadas en 2 jerarquías separadas (abstracción e implementación) que pueden ser desarrolladas de forma independiente.

Composite: permite componer objetos en estructuras de árboles y las utiliza como si fuera objetos individuales (incluso los objetos compuestos). Un ejemplo de uso es cuando tienes una jerarquía de objetos y una estructura de árboles donde se pueda aprovechar la recursividad.

Decorator: permite añadir nuevos comportamientos a los objetos al colocarlos dentro de una envoltura especial que contiene dichos métodos. Ejemplo de casos de uso es cuando se quiere añadir funcionalidades sin modificar la clase original o solo a determinadas instancias de la clase.

Flyweight: permite componer objetos en estructuras de árboles, que luego utiliza como si fueran objetos individuales (incluso si son objetos compuestos). Un ejemplo de uso es cuando se tiene una jerarquía de objetos y una estructura de árboles donde se puede aprovechar la recursividad.

Proxy: actúa como un reemplazo o marcador para otro objeto. Su función es controlar el acceso al objeto original, permitiendo realizar algo antes o después de que la solicitud llegue a ese objeto.

Explorar y aplicar estos patrones en tus proyectos te ayudará a escribir un código más limpio, reutilizable y organizado.

Links de referencia 1 Cursos relacionados Más recursos