DESIGN PATTERNS: DECORATORS

Es un patrón de diseño de tipo estructural; estos centran su atención en la composición de objetos; en cómo las entidades se relacionan entre sí, y cómo pueden ser usadas entre ellas mismas.

Diseño y Arquitectura de Software: Juan Carlos Sleiman Muñiz

Funcionalidad principal.

- La utilidad principal del patrón Decorator, es la de **dotar de funcionalidades dinámicamente a objetos mediante composición**. Es decir, vamos a *decorar* los objetos para darles más funcionalidad de la que tienen en un principio.
- Esto es algo verdaderamente útil cuándo queremos **evitar jerarquías de clases complejas**. La herencia es una herramienta poderosa, pero puede hacer que nuestro diseño sea mucho menos extensible.

EL PROBLEMA: DESCRIBE CUÁNDO APLICAR EL PATRÓN

- POR EJEMPLO, ES COMÚN AL MANEJAR RECURSOS COMO ARCHIVOS, COMUNICACIONES O BASES DE DATOS QUE EL ACCESO A LOS MISMOS PUEDA FALLAR POR DIVERSAS RAZONES QUE NO PODEMOS CONTROLAR TOTALMENTE, IDEALMENTE USAMOS EL CÓDIGO TRY-EXCEPT PARA HACERNOS CARGO, PERO SI TENEMOS MUCHAS FUNCIONES QUE VAN A NECESITAR EXACTAMENTE EL MISMO MANEJO DE ERROR

SOLUCIÓN:

- LO MEJOR ES ESCRIBIR UN DECORADOR QUE HAGA EL TRABAJO Y PUEDA RE-USARSE EN VEZ DE COPIAR-PEGAR VARIAS VECES LA MISMA RUTINA TRY-EXCEPT QUE SOLEMOS REALIZAR.

COSECUENCIAS: PROS Y CONTRAS DE APLICAR EL PATRÓN

- + PODEMOS AÑADIR RESPONSABILIDADES A UN OBJETO DE FORMA PROGRESIVA Y DINÁMICA. MÁS FLEXIBILIDAD QUE CON LA HERENCIA.
- + EVITA QUE LAS CLASES DE ARRIBA DE LA JERARQUÍA ESTÉN REPLETAS DE FUNCIONALIDADES. EN VEZ DE DEFINIR UNA CLASE COMPLEJA QUE TRATA DE DAR CABIDA A TODAS ELLAS, LA FUNCIONALIDAD SE LOGRA AÑADIENDO DECORADORES A UNA CLASE SIMPLE.
- - LA PRINCIPAL ES QUE EL OBJETO DECORATOR NO ES EXACTAMENTE IGUAL QUE LA CLASE QUE ESTÁ DECORANDO, POR LO QUE TENEMOS QUE TENER CUIDADO.
- - MUCHOS OBJETOS PEQUEÑOS. EL SISTEMA PUEDE SER MÁS DIFÍCIL DE APRENDER Y DE DEPURAR.