6° PRÁCTICA DE LABORATORIO

Presentado por:
Jesus Gabriel Parra Dugarte

Presentado a:
Ricardo Zambrano

Laboratorio de Ingeniería de Software II

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones

Programa de Ingeniería de sistemas

Popayán, Cauca

11/octubre/2023

1. Desarrollar Plantilla

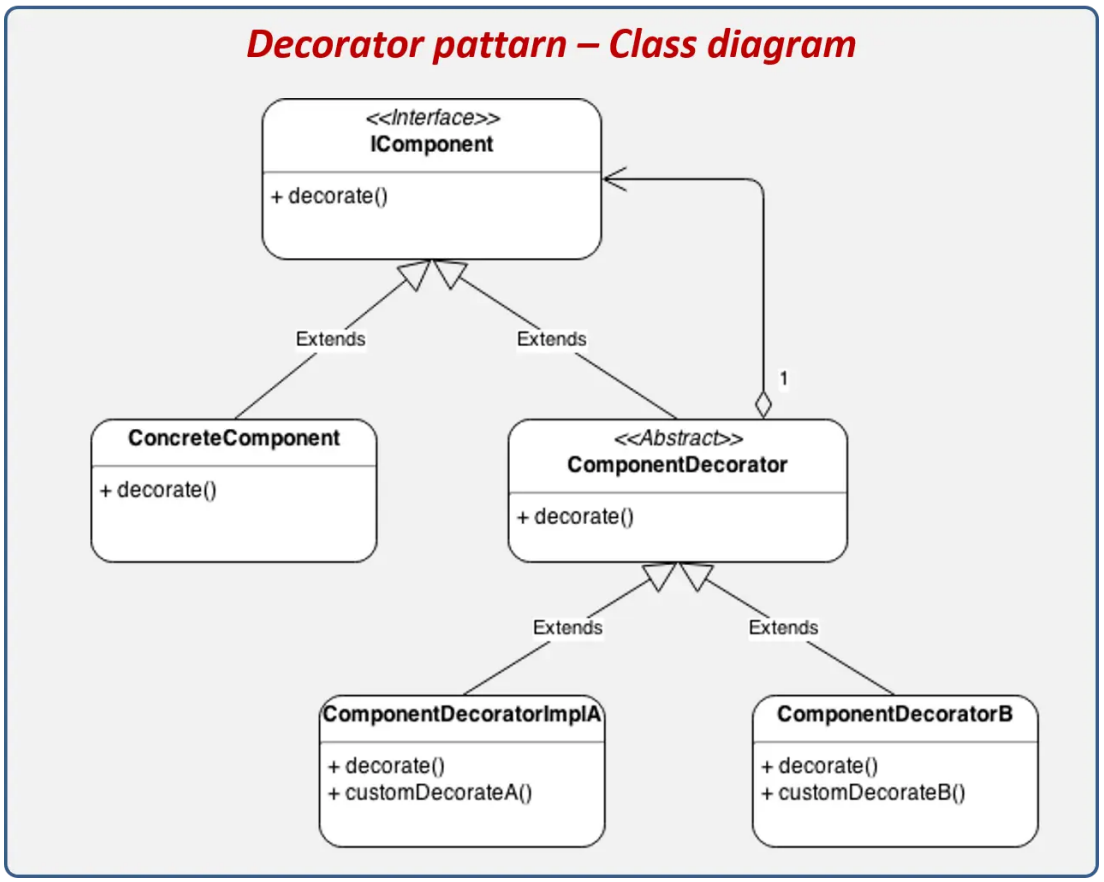
Patrón creacional: **Decorador**	
Intención	Agregar dinámicamente nuevas funcionalidades a un objeto. Es una alternativa flexible a la herencia.
Soluciona qué problema	Si se quiere añadir comportamientos o estados a objetos individuales durante la ejecución. La herencia no es recomendable porque es estática y aplica a la clase completa, además de eso sólo es posible especializar de una clase a la vez.
Solución propuesta	La solución propuesta por el patrón decorador, es hacer uso de la composición o agregación para efectuar cambios en tiempo de ejecución y delegar responsabilidades.
Diagrama de clases	

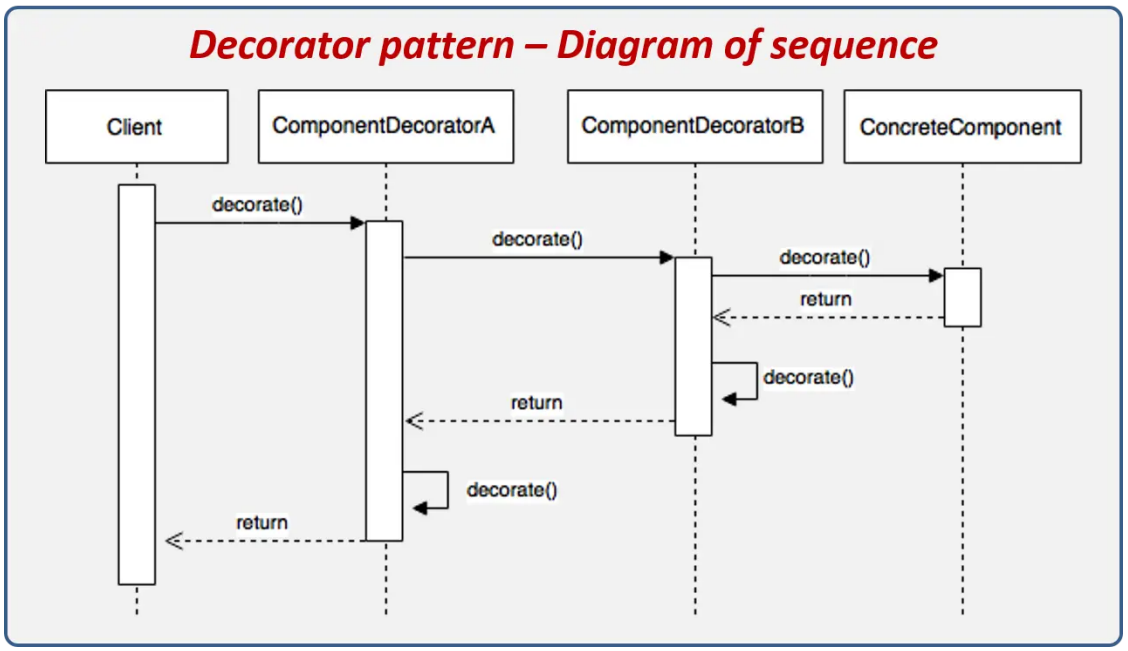
Diagrama de secuencias	
Participantes	La interfaz de componente, el componente concreto, la interfaz de decoradores y los decoradores concretos.
Aplicabilidad	Este patrón puede ser utilizado cuando se quiere añadir funcionalidades a un objeto en tiempo de ejecución sin descomponer el código ya creado.
Consecuencias	• Permite añadir responsabilidades a un objeto, no métodos a la interfaz de un objeto. La interfaz presentada permanece constante conforme las capas son especificadas. • También se puede decir que la identidad del objeto núcleo ha sido “ocultada” por los objetos envoltorios. Acceder directamente al objeto central se complica

Figura 1.1 (Hawaiiano con la camiseta de millonarios)

3. Referencias

- “Decorator Design Pattern.” *SourceMaking*, https://sourcemaking.com/design_patterns/decorator. Accessed 11 October 2023.
- “Decorator - Patrones estructurales.” *Refactoring.Guru*, https://refactoring.guru/es/design-patterns/decorator. Accessed 11 October 2023.