Decorator (Decorador)

Patrones de diseño estructurales

B35473 Ana Teresa Quesada

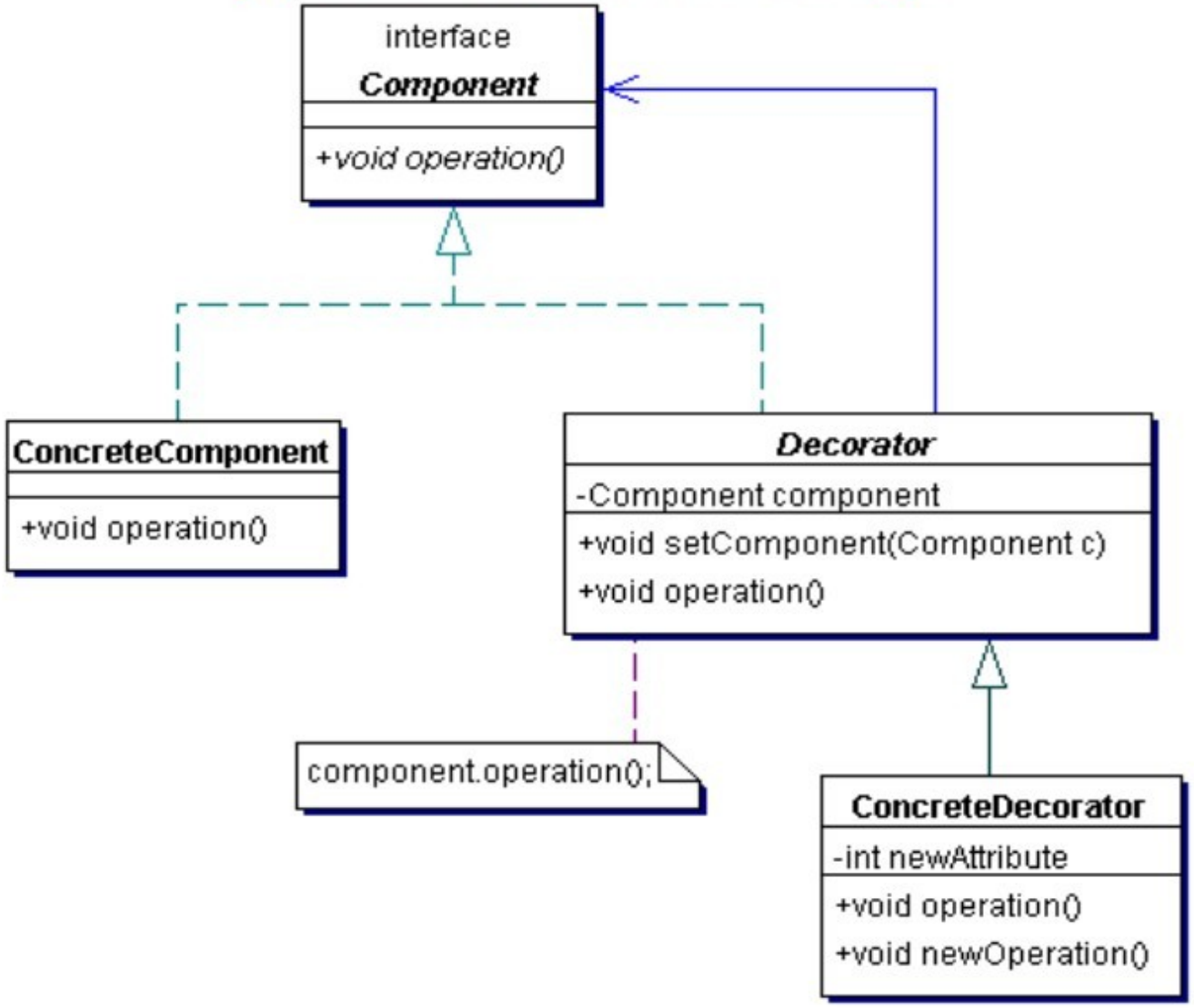
B57970 Frida Xirinachs

- Propiedades:
  - Tipo: estructural, objeto
  - Nivel: componente

- Propósito:

Proporcionar una forma flexible de introducir o eliminar funcionalidad de un componente sin modificar su apariencia externa o su función. Mediante la composición (para cambios en tiempo de ejecución).

Figure 3.6. Decorator class diagram



Implementación y sus agentes

- Component (Component): Componente que contiene el comportamiento genérico.
Clase abstracta o interfaz.

- ConcreteComponent (Componente Concreto): Objeto al que se le pueden agregar otras tareas.

- Decorator (Decorador): Comportamientos estándar esperados.

Mantiene referencia a un objeto Component

-> (objeto ConcreteObject u otro Decorator)

Clase abstracta o interfaz.

Misma referencia para cualquier propósito.

- Uno o más ConcreteDecorator (Decorador concreto): Deben soportar modificaciones (referencias a un componente, más la habilidad de insertar o eliminar la misma).

Pueden definir métodos adicionales y/o variables.

¿Como podemos reconocer el momento adecuado para su aplicación?

- Cambios dinámicos, sin restricciones de herencia.
- Introducir o eliminar capacidades a componentes en tiempo de ejecución.
- Características cambiantes, aplicadas dinámicamente, y combinadas sobre un componente.

¿Ventajas? ¿Desventajas?

Ajustar y aumentar comportamiento en ejecución.

Al escribir nuevas clases para añadir funcionalidad, la programación es más sencilla, y el comportamiento de un componente no cambia a pesar de ello.

Se vuelve complicado la verificación de código, depuración y capas, y la velocidad se reduce.

Bibliografía

Sthephen Stelting / Olav Maassen. (2003). Patrones de diseño aplicados a Java. Madrid: Pearson Education.