Patrón Decorador

Juan Francisco González Junior

Universidad de la Cuenca del Plata

Características

El patrón **Decorador** es un patrón de diseño estructural que permite añadir dinámicamente nuevos comportamientos a objetos colocándolos dentro de objetos especiales que los envuelven conocidos como “wrappers”. Según este patrón, cualquier objeto puede complementarse con un comportamiento deseado sin influir en las funcionalidades de otros objetos de la misma clase. El componente de *software* que hay que ampliar se “decora” con una o más **clases decoradoras**, que lo envuelven completamente, siguiendo el patrón de diseño Decorador. Cada decorador es del mismo tipo que el **componente al que envuelve** y, por lo tanto, tiene la **misma interfaz**. De esta manera, las llamadas de método entrantes pueden delegarse fácilmente al componente adjunto mientras lleva a cabo una funcionalidad. Las llamadas también pueden procesarse dentro del decorador.

Ventajas

Componente: Define la interfaz para los objetos que pueden tener responsabilidades añadidas.

Componente Concreto: Define un objeto al cual se le pueden agregar responsabilidades adicionales.

Decorador: Mantiene una referencia al componente asociado. Implementa la interfaz de la superclase Componente delegando en el componente asociado.

Decorador Concreto: Añade responsabilidades al componente.

Ventajas

Alto grado de flexibilidad.

Ampliación de las funciones de las clases sin herencia.

Funcionalidades optimizadas para los recursos

Desventajas

Alto número de objetos/módulos.

Alta complejidad del software (especialmente la interfaz Decorador).

Dificultad de testeo y depuración elevada

Referencias

https://refactoring.guru/es/design-patterns/decorator
https://nithinbekal.com/posts/ruby-decorators/
https://codigofacilito.com/articulos/presenters-en-rails-con-decorators-que-por-que-y-como