PETI



MARIA CAMILA PRECIADO ROJAS

ROBINSON FERNANDO RODRIGUEZ APONTE



UNIVERSIDAD PEDAGÓGICA Y TECNOLÓGICA DE COLOMBIA

FACULTAD DE INGENIERÍA

ESCUELA DE SISTEMAS Y COMPUTACION

TUNJA, BOYACÁ

2017

**DECORATOR**

Es un patrón estructural. La utilidad principal del patrón Decorator, es la de dotar de funcionalidades dinámicamente a objetos mediante composición. Es decir, se va a decorar los objetos para darles más funcionalidad de la que tienen en un principio.

Esto es algo verdaderamente útil cuándo se quiere evitar jerarquías de clases complejas. La herencia es una herramienta poderosa, pero puede hacer que el diseño sea mucho menos extensible.

Este patrón se debe utilizar cuando:

* Hay una necesidad de extender la funcionalidad de una clase, pero no hay razones para extenderlo a través de la herencia.
* Se quiere agregar o quitar dinámicamente la funcionalidad de un objeto.

Ventajas

* Se puede añadir responsabilidades a un objeto de forma progresiva y dinámica. Más flexibilidad que con la herencia.

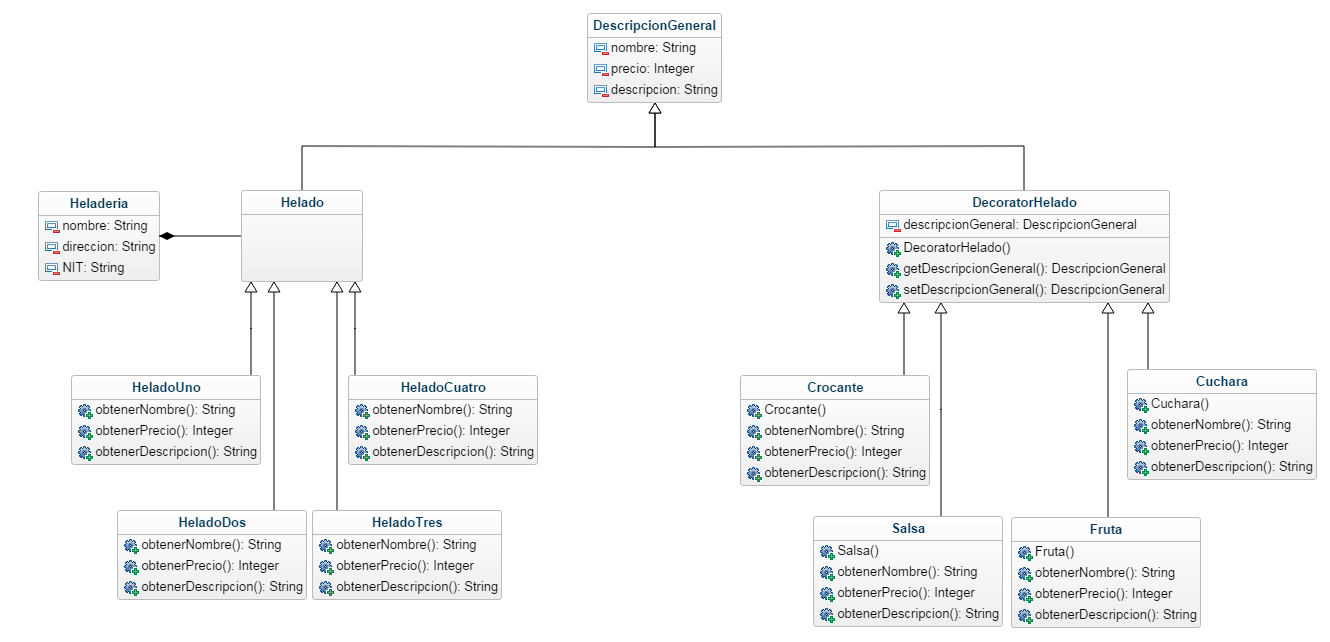
Desventajas

El objeto Decorator no es exactamente igual que la clase que está decorando, por lo que se debe tener cuidado. Además, se puede encontrar con un diseño de clases muy pequeñas, pero en gran cantidad.

**Ejemplo**

Se va a trabajar con una heladería, la cual tiene parámetros como: Nombre, Dirección y NIT, esta cuenta con 4 tipos de helados, con su respectivo nombre, precio y descripción, a su vez contara con una clase para el patrón decorator la cual tendrá una descripción general de los helados, también cuenta con los diferentes agregados como lo son: crocantes, salsas, fruta o cuchara

**Diagrama de clases**



**El código java se encuentra anexado como Decorator.jar**