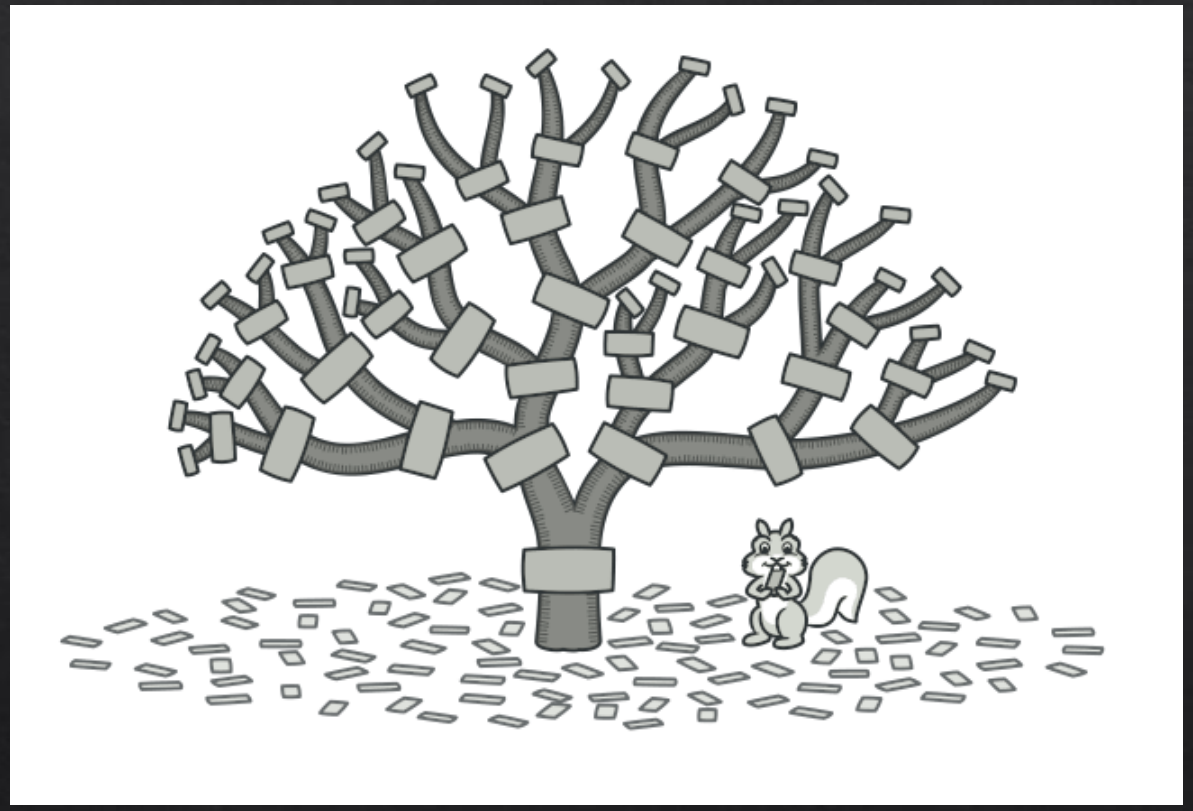
Patrones de diseño

Estructurales

COMPOSITE

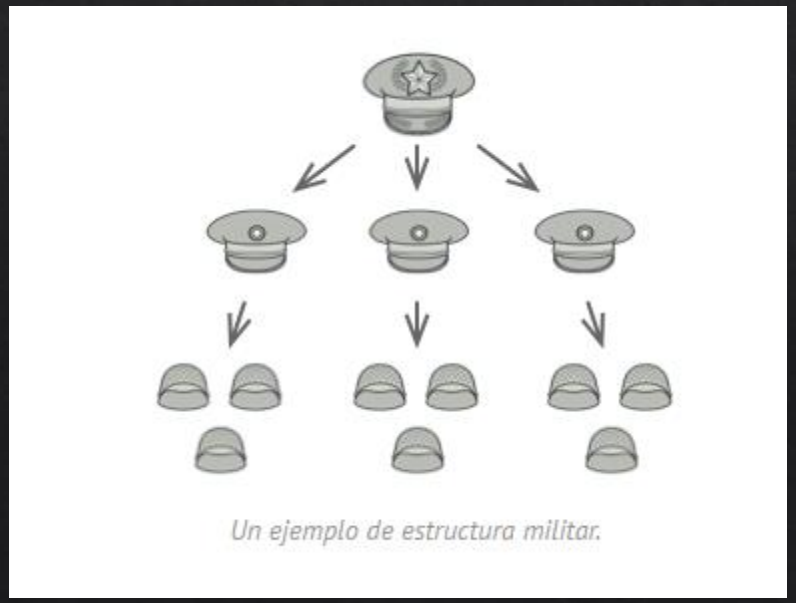


Es un patrón de diseño estructural que te permite componer objetos en estructuras de árbol y trabajar con esas estructuras como si fueran objetos individuales.

Nos sirve para construir estructuras complejas partiendo de otras estructuras mucho más simples, dicho de otra manera, podemos crear estructuras compuestas las cuales están conformadas por otras estructuras más pequeñas.

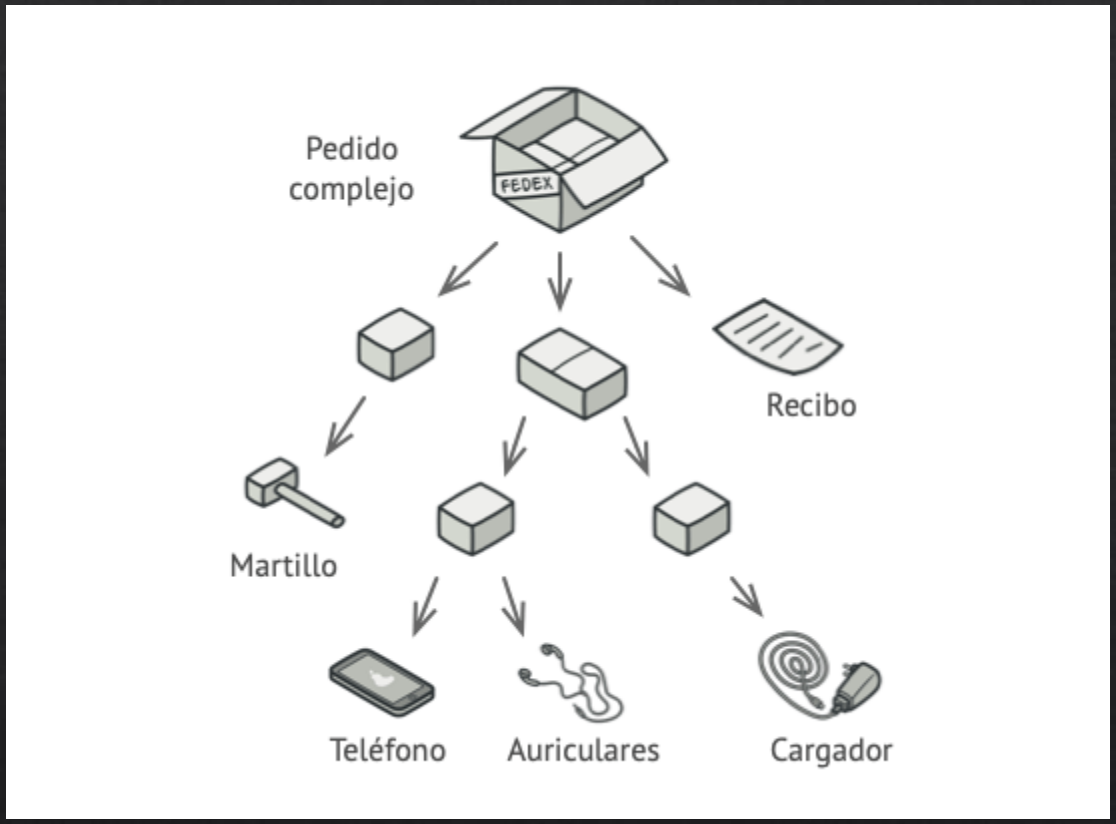
Ventajas y Desventajas

La gran ventaja de esta solución es que no tienes que preocuparte por las clases concretas de los objetos que componen el árbol. No tienes que saber si un objeto es un producto simple o una sofisticada caja. Puedes tratarlos a todos por igual a través de la interfaz común. Cuando invocas un método, los propios objetos pasan la solicitud a lo largo del árbol.

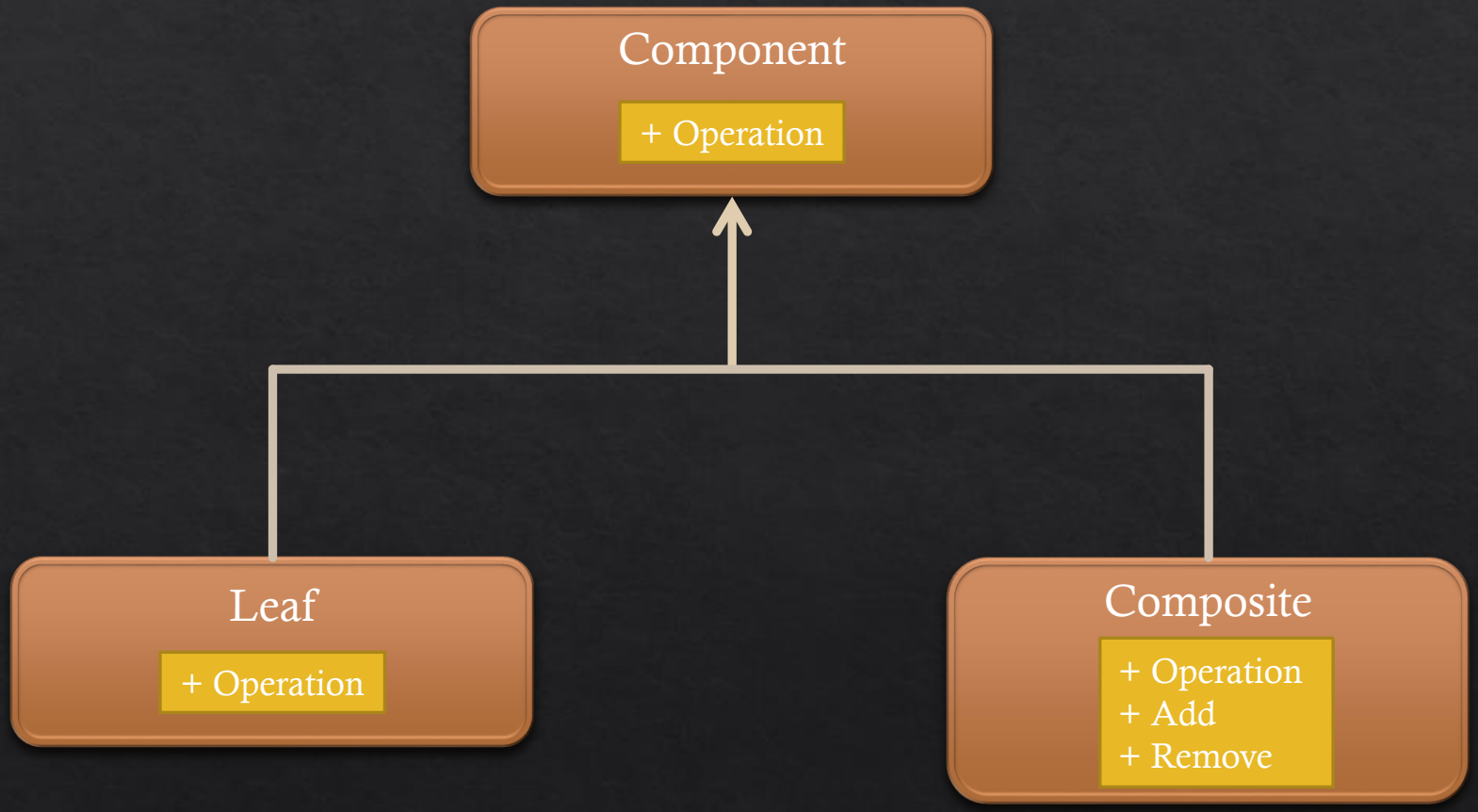


Un ejemplo de estructura militar.

El uso del patrón Composite sólo tiene sentido cuando el modelo central de tu aplicación puede representarse en forma de árbol.
Por ejemplo, imagina que tienes dos tipos de objetos: Productos y Cajas. Una Caja puede contener varios Productos así como cierto número de Cajas más pequeñas. Estas Cajas pequeñas también pueden contener algunos Productos o incluso Cajas más pequeñas, y así sucesivamente.


Pedido
complejo
FEDEX
Recibo
Martillo
Teléfono
Auriculares
Cargador

Patrón de diseño



Component: Generalmente es una interface o clase abstracta la cual tiene las operaciones mínimas que serán utilizadas, este componente deberá ser extendido por los otros dos componentes Leaf y Composite.

Leaf: El leaf u hoja representa la parte más simple o pequeña de toda la estructura y este extiende o hereda de **Component.**

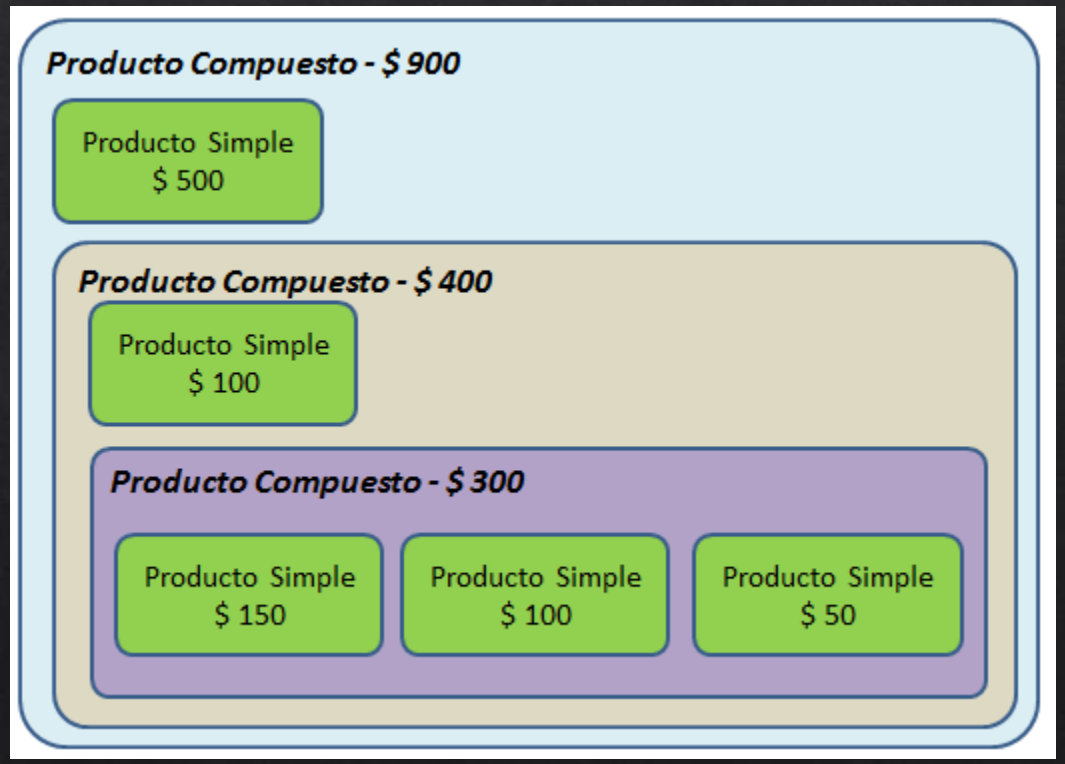
Composite: Es una estructura conformada por otros Composite y Leaf, los Composite tiene los métodos **add** y **remove** los cuales nos permiten agregar objetos de tipo **Component**, Sin embargo como hablamos anteriormente, el **Component** es por lo general un **Interface** o **Clase abstracta** por lo que podremos agregar objetos de tipo Composite o Leaf.

¿Cuándo se usa?

Debe usarse cuando los clientes necesitan ignorar la diferencia entre composiciones de objetos y objetos individuales.

Si los programadores descubren que están usando múltiples objetos de la misma manera, y a menudo tienen código idéntico para manejar cada uno de ellos, entonces composite es una buena opción, en esta situación es menos complejo tratar los primitivos y los compuestos como homogéneos.

La siguiente imagen muestra un ejemplo con un punto de venta en el que se ofrece a los clientes paquetes de ventas.
Leaf es representado por el producto simple.
Component es representado por productos compuestos.



Bibliografía

https://refactoring.guru/es/design-patterns/composite

https://www.oscarblancarteblog.com/2014/10/07/patron-de-diseno-composite/