

7º 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 estructural: Composite	
Intención	permite componer objetos en estructuras de árbol y trabajar con esas estructuras como si fueran objetos individuales
Soluciona qué problema	Composite sólo tiene sentido cuando el modelo central de tu aplicación puede representarse en forma de árbol
Solución propuesta	El patrón Composite sugiere que trabajes con Productos y Cajas a través de una interfaz común. 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.
Diagrama de clases	<pre> classDiagram class Component { <<interface>> +doThis() } class Leaf { +doThis() } class Composite { -elements +addElement() +doThis() } Component < -- Leaf Component < -- Composite Composite o-- "elements" : -elements Composite ..> Note : // Container functionality\n// for each element\nelements[i].doThis(); </pre>

Diagrama de secuencias	<div><p>Composite pattern – Diagram of sequence</p><pre>sequenceDiagram participant Client participant CompositeA participant CompositeB participant LeafA participant LeafB participant LeafC Client->>CompositeA: doAction() activate CompositeA CompositeA->>CompositeB: doAction() activate CompositeB CompositeB->>LeafA: doAction() activate LeafA LeafA-->>CompositeB: return deactivate LeafA CompositeB->>LeafB: doAction() activate LeafB LeafB-->>CompositeB: return deactivate LeafB CompositeB->>LeafC: doAction() activate LeafC LeafC-->>CompositeB: return deactivate LeafC CompositeB-->>CompositeA: return deactivate CompositeB CompositeA-->>Client: return deactivate CompositeA</pre></div>
Participantes	Interfaz de componente, hojas y composiciones.
Aplicabilidad	Se usa para representar jerarquías reales con el uso de árboles
Consecuencias	<ul style="list-style-type: none">• Puede resultar difícil proporcionar una interfaz común para clases cuya funcionalidad difiere demasiado. En algunos casos, se tendrá que generalizar en exceso la interfaz componente, provocando que sea más difícil de comprender.

2. Referencias

- “Composite - Patrones estructurales.” *Refactoring.Guru*, <https://refactoring.guru/es/design-patterns/composite>. Accessed 19 October 2023.
- “Composite Design Pattern.” *SourceMaking*, https://sourcemaking.com/design_patterns/composite. Accessed 19 October 2023.