Patrón Composite

UAI
Universidad Abierta Interamericana

UAIOnline Ultra

Introducción a los patrones de diseño. Patrón Composite

- **OBJETIVOS**
  - Conocer las mejores prácticas en diseño de software
  - Aprender a estandarizar técnicas de implementación

Introducción a los patrones de diseño. Patrón Composite

- **HABILIDADES Y COMPETENCIAS QUE DESARROLLA LA ASIGNATURA**
  - Analizar un dominio para identificar problemas recurrentes de diseño utilizando los patrones de diseño de software.
  - Implementar código eficiente para lograr aplicaciones robustas, mantenibles y reutilizables utilizando el patrón de diseño Composite

Introducción a los patrones de diseño

UAI
Universidad Abierta Interamericana

UAIOnline Ultra

PATRONES

¿QUÉ ES UN PATRÓN?

- Nociones introducidas por el Arquitecto Christopher Alexander para describir diseños de arquitectura
- Descubrió más de 250 patrones, buscando la esencia de edificios, ciudades y lugares destacables.

PATRONES

¿QUÉ ES UN PATRÓN?

- “Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, así como la solución a ese problema, de tal modo que se pueda aplicar esta solución un millón de veces, sin hacer lo mismo dos veces”

PATRONES

¿QUÉ ES UN PATRÓN?

- “Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, así como la solución a ese problema, de tal modo que se pueda aplicar esta solución un millón de veces, sin hacer lo mismo dos veces”

PATRONES

¿QUÉ ES UN PATRÓN?

- Cada patrón representa una solución a un problema recurrente
- Los patrones pueden ser adaptados y combinados de diferentes maneras generando infinitas posibilidades

PATRONES DE DISEÑO EN IS

- Son descripciones de clases y objetos relacionados que están particularizados para resolver un problema de diseño general en un contexto determinado.
- Nomina, abstrae e identifica los aspectos clave de una estructura de diseño común, lo que los hace útiles para crear diseño OO reutilizables

PATRONES DE DISEÑO EN IS

- THE GANG OF FOUR
- Libro de 1995 donde se presentan los patrones de diseño
- *Design Patterns: Elements of Reusable Object –Oriented Software. (Gamma, Helm, Johnson, Vlissides)*

PATRONES DE DISEÑO EN IS

- THE GANG OF FOUR
- Se introdujeron 23 patrones divididos en 3 categorías
  - PATRONES DE CREACION
    - Centrados en el proceso de creación de objetos
  - PATRONES DE ESTRUCTURA
    - Enfocados en la composición estática y en las estructuras de clases y objetos
  - PATRONES DE COMPORTAMIENTO
    - Enfocados en la interacción dinámica

ELEMENTOS ESENCIALES DE UN PATRÓN

- **Nombre del Patrón** describe el problema junto con sus soluciones y consecuencias
- **Problema** a resolver y el contexto. Describe cuando usar el patrón.
- **Solución**, con el diseño de sus elementos, relaciones, responsabilidades y colaboraciones.
- **Consecuencias**, cuales son las ventajas e inconvenientes de aplicar el patrón.
- **Contexto**: Donde será posible aplicar el patrón

Patrón Composite

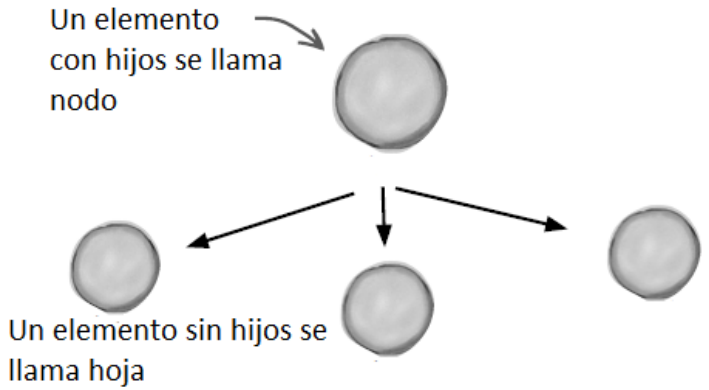
UAI
Universidad Abierta Interamericana

UAIOnline Ultra

PATRON COMPOSITE (COMPUESTO)

Propósito

- Compone objetos en estructuras de árbol para representar jerarquías de parte-todo.
- Permite que los clientes traten de manera uniforme a los objetos individuales y a los compuestos



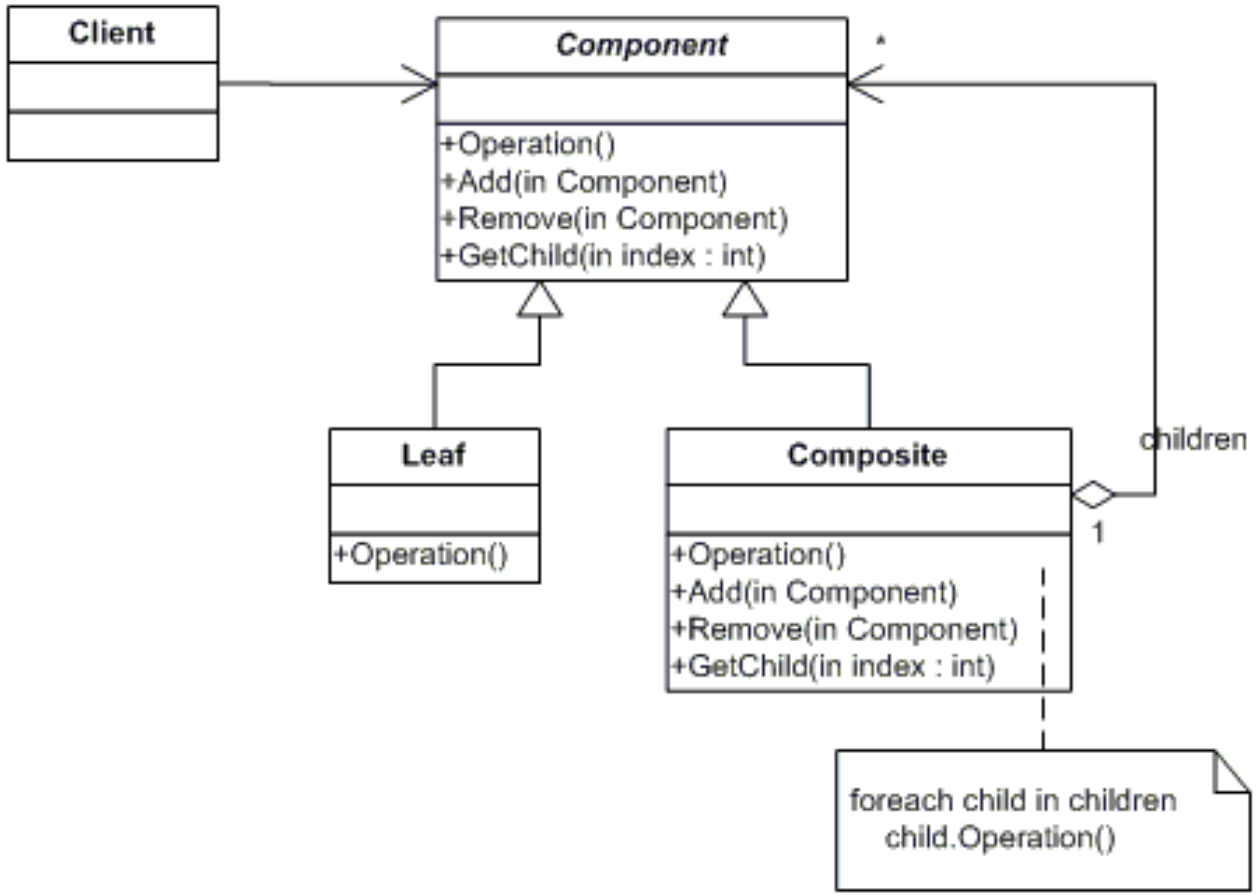
PATRON COMPOSITE (COMPUESTO)

Aplicabilidad

- Quiera representar jerarquías de objetos todo-parte.
- Quiere que los clientes sean capaces de obviar las diferencias entre composiciones de objetos y los objetos individuales. Los clientes tratarán a todos los objetos de la estructura compuesta de manera uniforme.

PATRON COMPOSITE (COMPUESTO)

- Estructura



PATRON COMPOSITE (COMPUESTO)

Consecuencias:

- Define jerarquías de clases formadas por objetos primitivos y compuestos.
- Simplifica al cliente. El cliente trata las estructuras uniformemente.
- Facilita añadir nuevos tipos de componentes.
- Puede hacer que un diseño sea demasiado general.

PATRON COMPOSITE (COMPUESTO)

Notas:

- Crear una interfaz que oficie de "mínimo denominador común" que haga que los contenedores y los contenidos puedan ser intercambiables
- Todas las clases Contenedor y Contenido implementan la interfaz.
- Métodos de manejos de hijos (Ej. *AddChild()*, *RemoveChild()*) deberían normalmente ser definidos en la clase *Composite*. Por desgracia, el deseo de tratar a los objetos *Leaf* y *Composite* de manera uniforme requiere que dichos métodos sean movidos a la clase abstracta *Component*.
- Las clases Contenedor aprovechan el polimorfismo para delegar en sus objetos Contenido.

Fin de la clase

UAI
Universidad Abierta Interamericana