Composite

Patrón de diseño

- **Cibils, Juan Ignacio**
- **Dominguez, Nicolas**
- **Juarez, Luciano**

Intención

-Componer objetos en estructura de árbol para representar jerarquías parte-todo.

-Permitir a los clientes tratar objetos individuales y composiciones de objetos uniformemente.

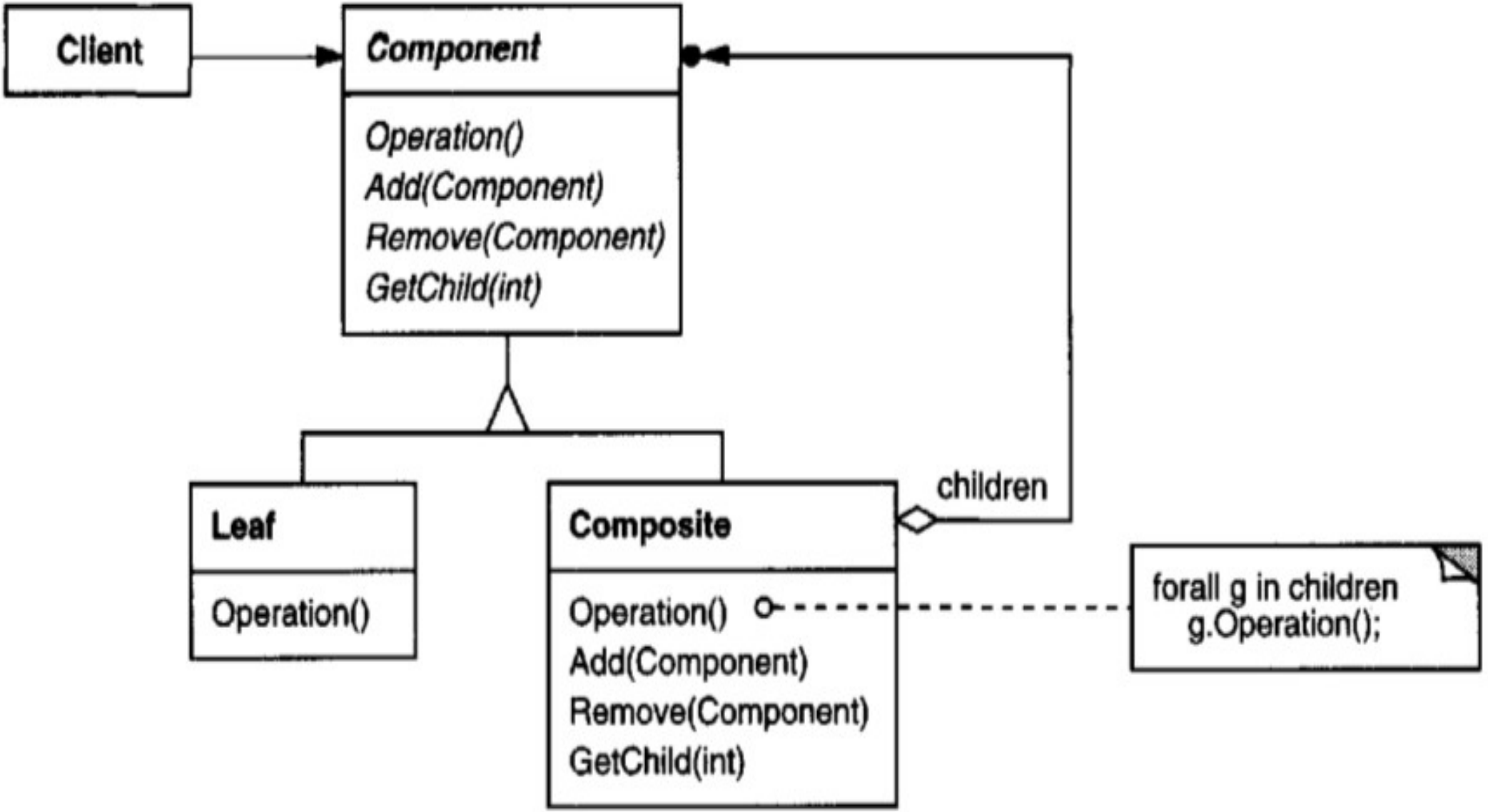
Motivación

En aplicaciones gráficas los usuarios pueden construir diagramas complejos a partir de componentes simples agrupandolos para formar otros mas grandes, que a su vez pueden estar agrupados para formar componentes todavıa mas grandes.

Una simple implementación de estas aplicaciones podría definir clases para gráficas primitivas y clases que actúan como contenedores de estas gráficas, lo cual lleva al problema de que el código que usan estas clases deben tratar los objetos primitivos y contenedores de manera diferente.

Como solución a este problema el patrón Composite propone una clase abstracta (Graphic) que represente tanto objetos primitivos como contenedores y sus operaciones. Donde las subclases (Line, Rectangle) definen objetos gráficos primitivos y la clase compuesta (Picture) define un conjunto de objetos gráficos

Estructura


Client
Component
Operation()
Add(Component)
Remove(Component)
GetChild(int)
Leaf
Operation()
Composite
Operation()
Add(Component)
Remove(Component)
GetChild(int)
children
forall g in children
g.Operation();

Participantes

Client:

>Manipula objetos en la composición a través de la interface Component.

Component (Graphic):

>Declara la interface para objetos en la composición
>Implementa el comportamiento de la interface común a todas las clases.
>Declara una interface para el acceso y manejo de sus componentes hijos.

Leaf(Text, Line, Rectangle):

>Representa objetos hoja en la composicion. No tiene hijos.
>Define el comportamiento de los objetos primitivos.

Composite(Picture):

>Define el comportamiento de los componentes que tienen hijos.
>Almacena componentes hijos.
>Implementa las op. relacionadas a los hijos de la interface Component.

Árbol generado

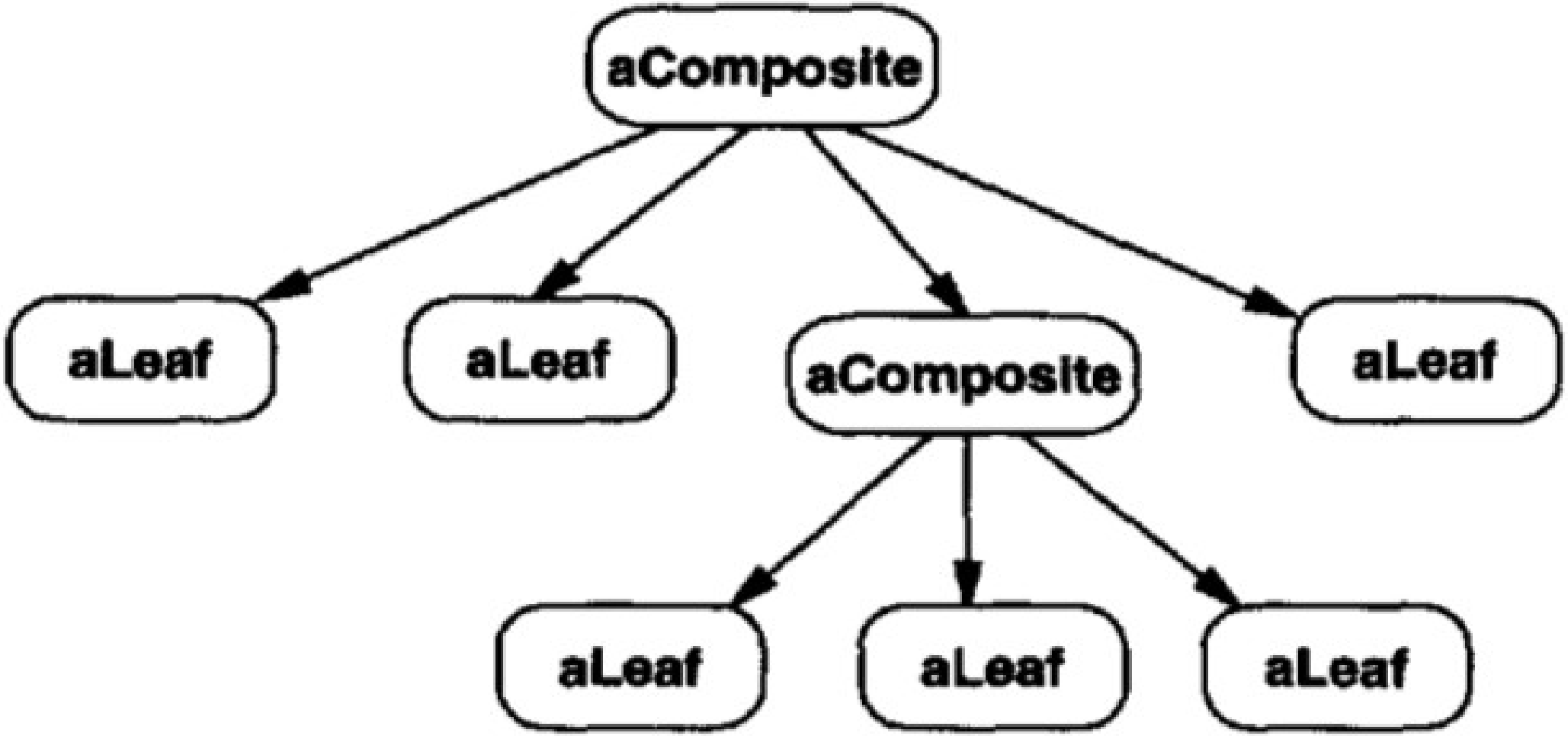
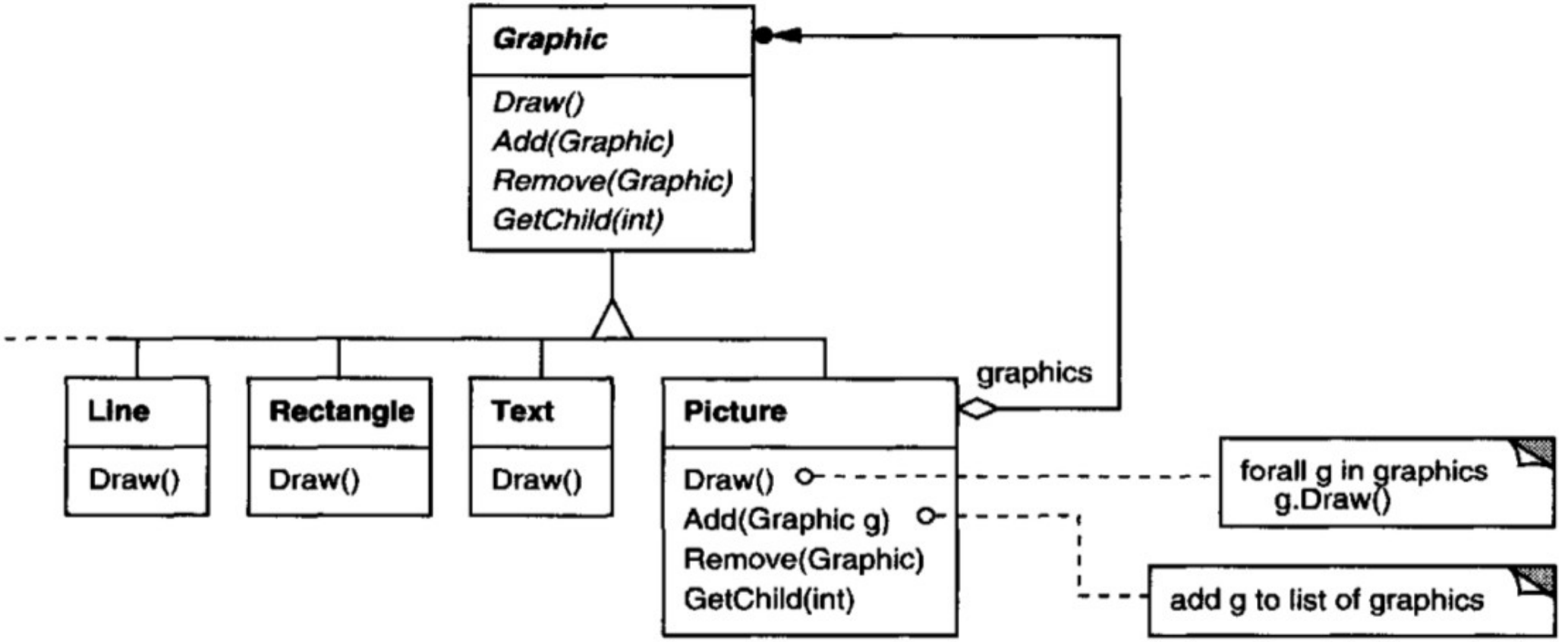


Diagrama de clases ejemplo



Aplicabilidad

El patron Composite se utiliza cuando:

>Se desea representar jerarquıas parte-todo de objetos.

>Se desea que los clientes ignoren la diferencia entre composiciones de objetos y objectos individuales.

Consecuencias

>Define jerarquías de clases formadas por componentes primitivos y compuestos

>Le permite al cliente tratar indistintamente a los objetos primitivos y compuestos, por lo general el cliente no sabe si trata con una hoja o un elemento compuesto. Simplificación de código.

>Hace al diseño demasiado general.

>Facilidad al añadir nuevos componentes. La desventaja de esto es que se hace mas difícil restringir los componentes de un objeto compuesto.