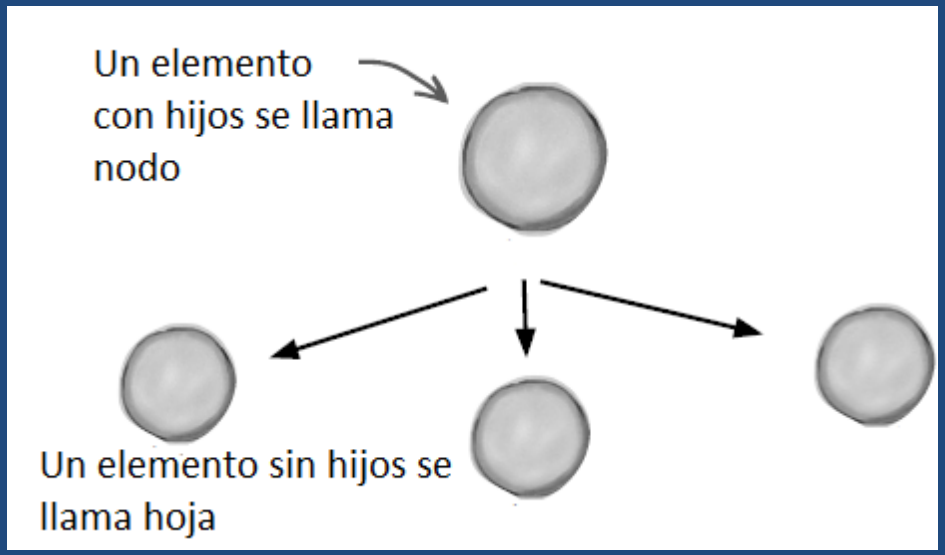
PATRON COMPOSITE (Compuesto)

- 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.



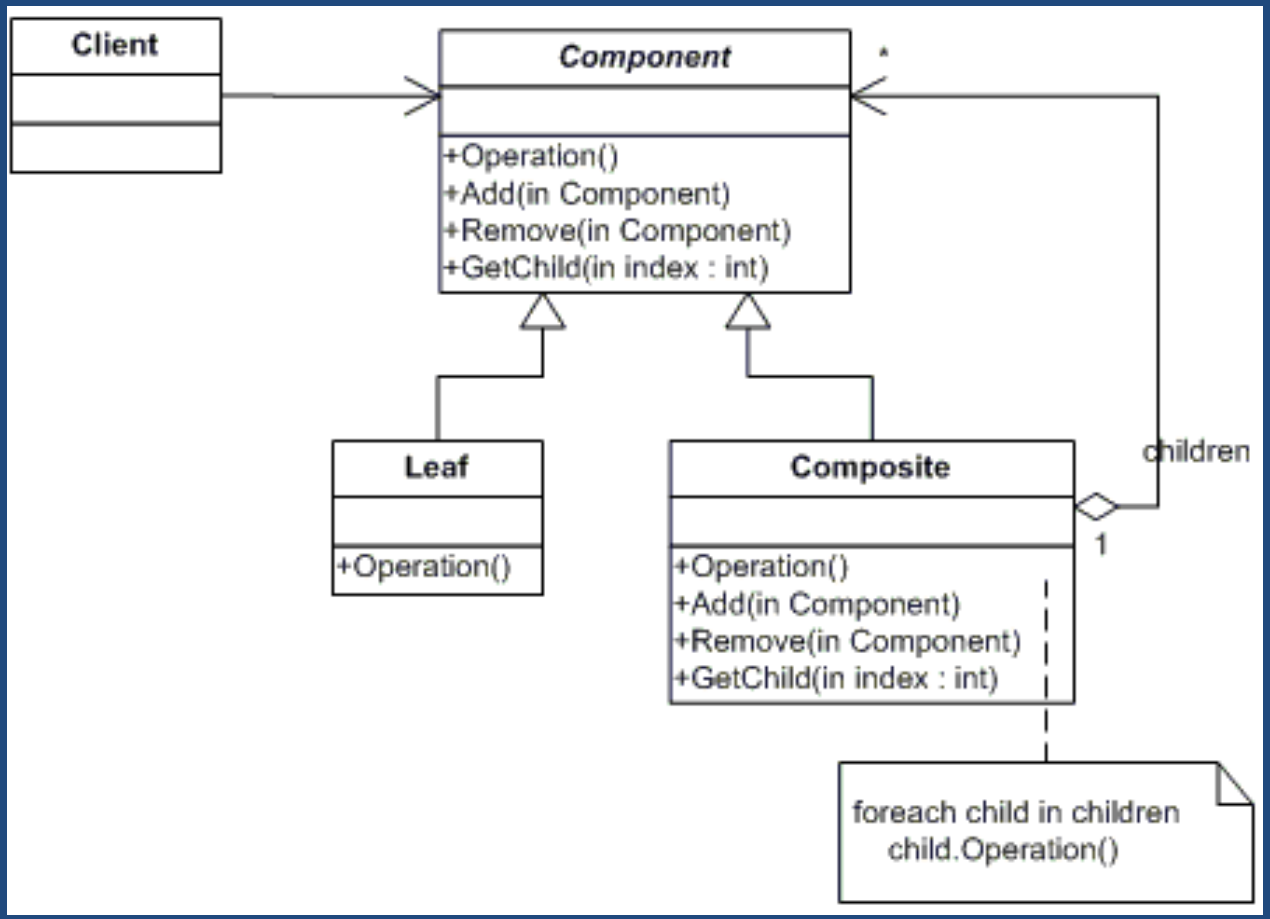
PATRON COMPOSITE (Compuesto)

- 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.

Contenedores que contienen elementos, cada uno de los cuales podría ser un contenedor

PATRON COMPOSITE (Compuesto)

- Estructura



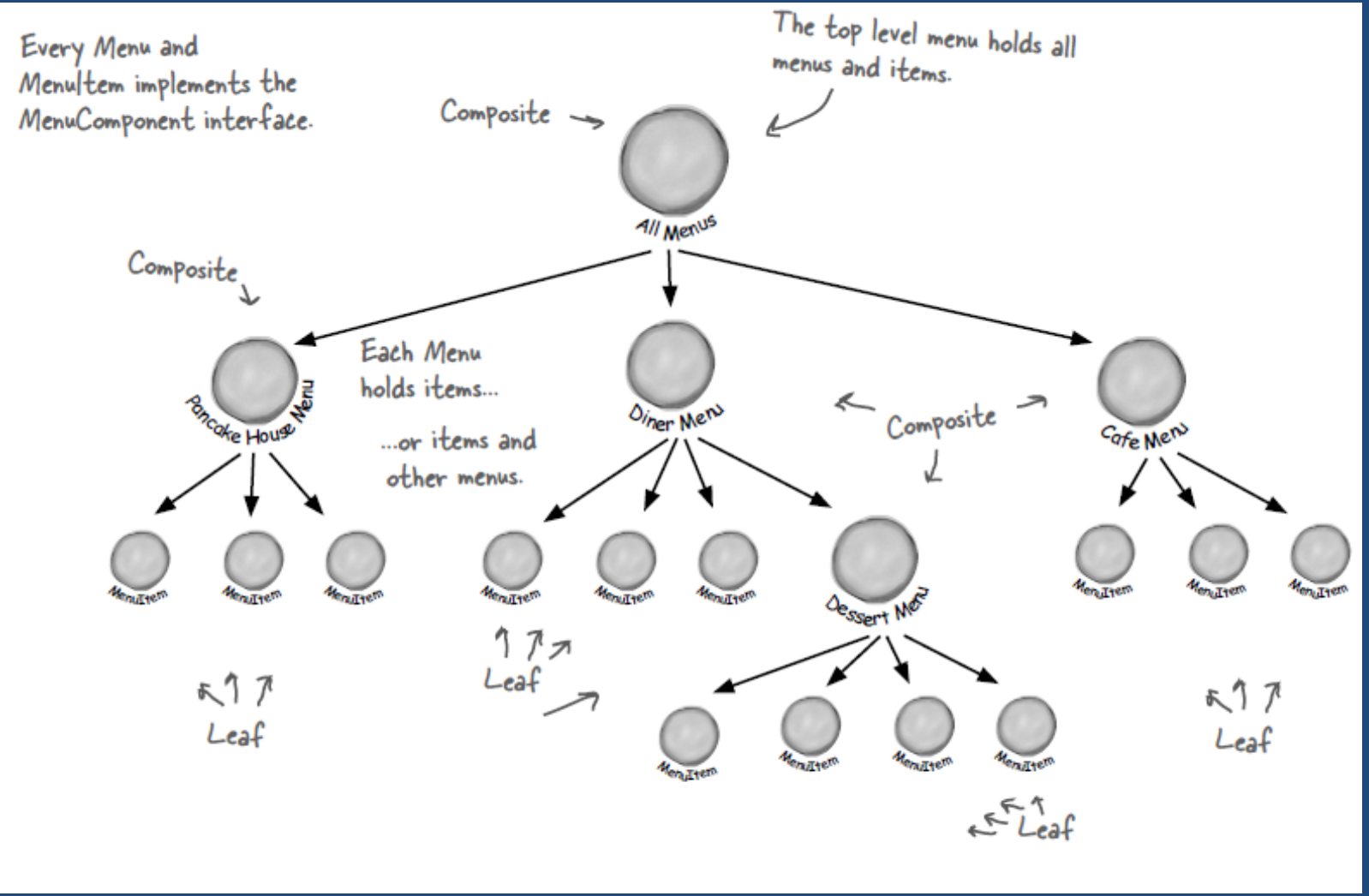
PATRON COMPOSITE (Compuesto)

- 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.

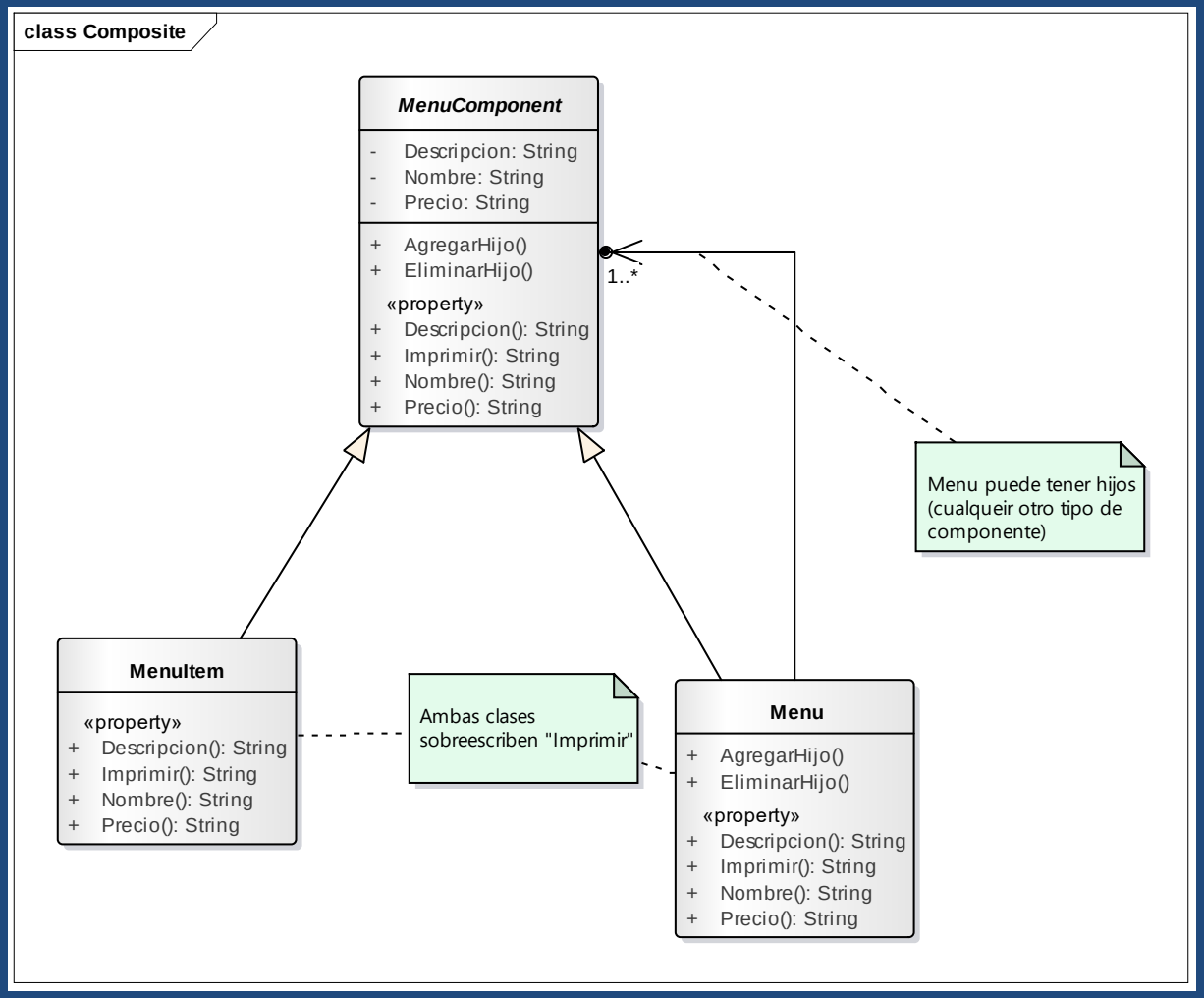
PATRON COMPOSITE - Ejemplo

- Nos piden desarrollar un sistema para manejar los diferentes menús en un restaurante, ya que dependiendo el momento del día el mismo cambia completamente.

PATRON COMPOSITE - Ejemplo



PATRON COMPOSITE - Ejemplo



NOTAS

- Crear una interfaz que oficie de "mínimo denominador común" que haga que los contenedores y los contenidos puedan ser intercambiables
- odas 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.