Composite

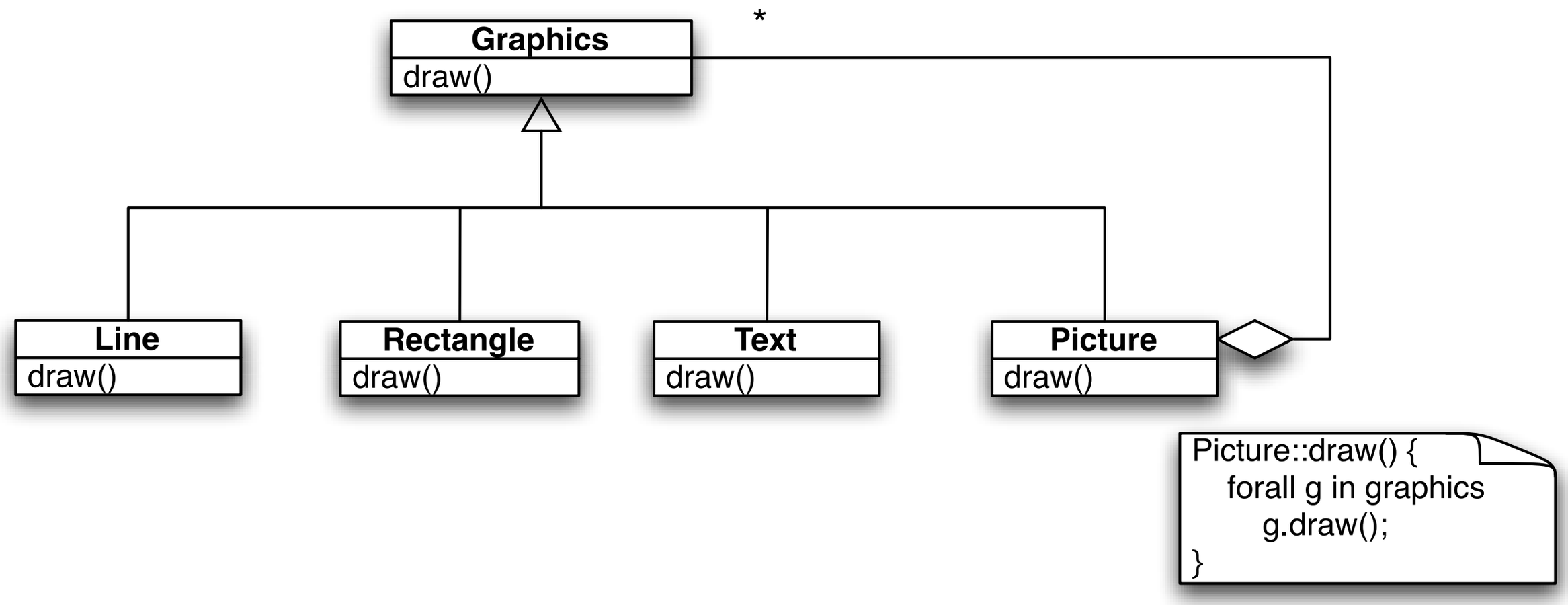
Objectif

- Composer des objets dans des arbres représentant des hiérarchies partie/tout (enfants/parent)
- Permettre aux clients de traiter de manière uniforme des objets ou des compositions d'objets

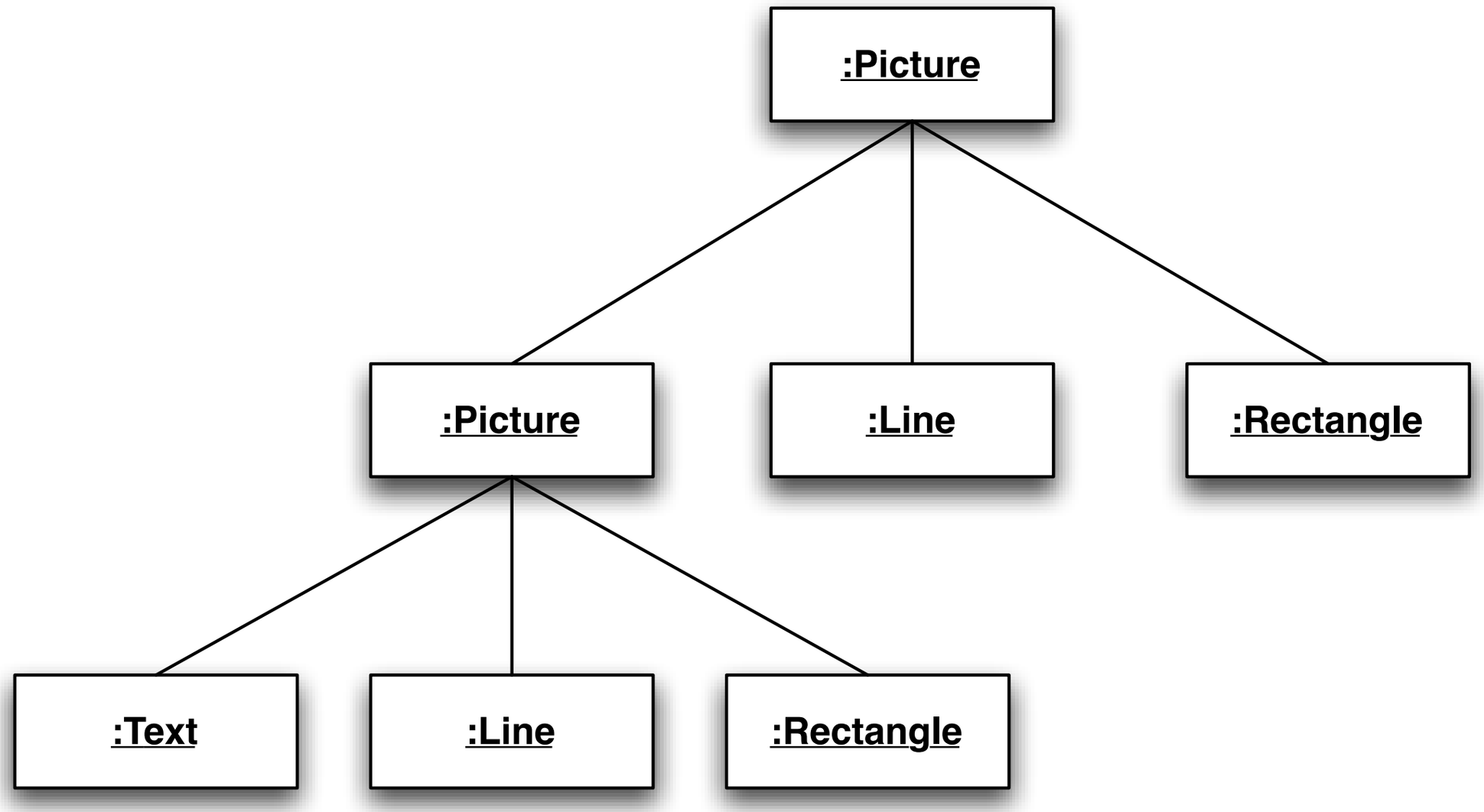
Motivation

- Parfois, certaines applications (e.g. éditeur graphique), ont besoin de traiter de la même manière, des objets composés d'autres objets et des objets individuels

Motivation - exemple



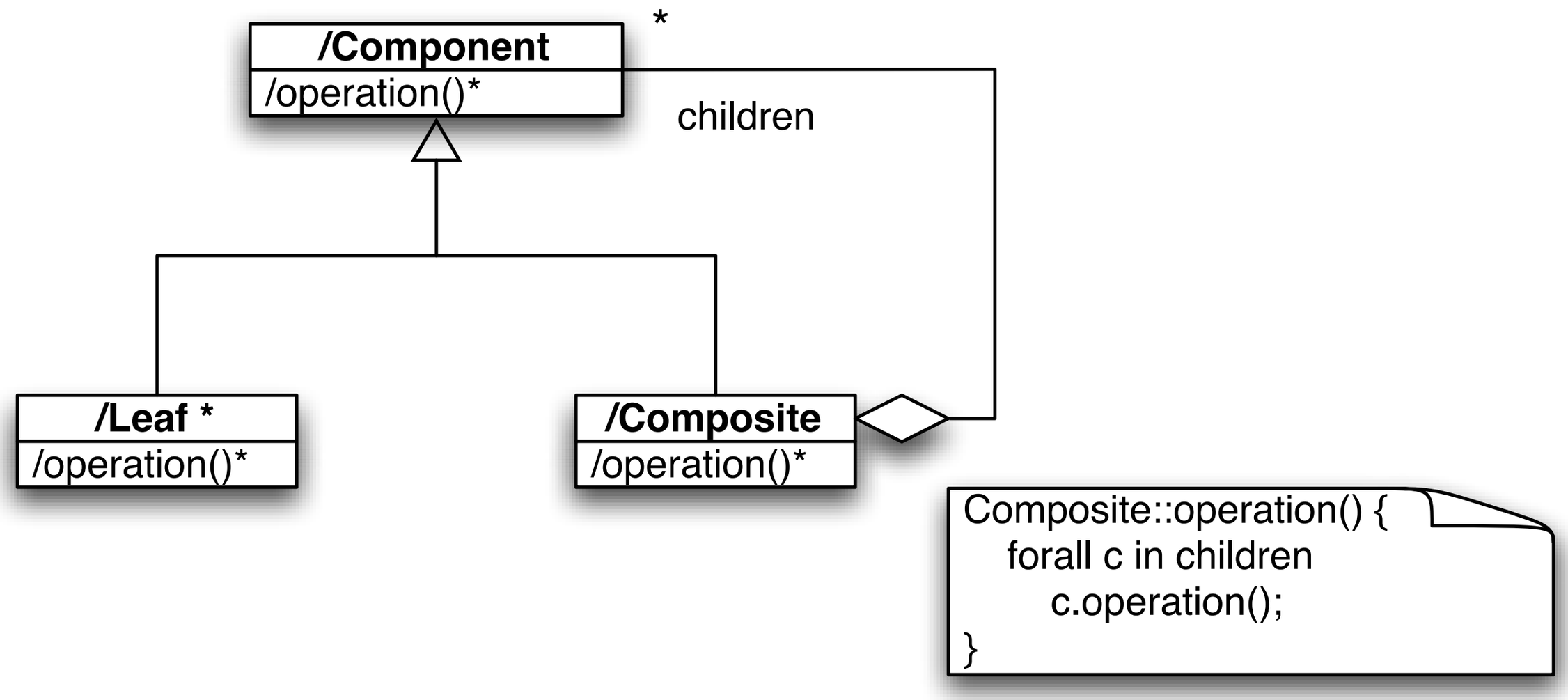
Motivation - exemple



Applicabilité

- Utiliser Composite quand:
  - On a besoin de représenter des structures hiérarchiques (tout/partie)
  - Les classes clientes n'ont pas besoin de savoir si un objet est atomique ou composé

Structure


/Component
/operation()*
*
children
/Leaf *
/operation()*
/Composite
/operation()*
Composite::operation() {
forall c in children
c.operation();
}

Conséquences

- Les clients traitent indifféremment des structures composite ou atomiques
- Les clients sont simplifiés
- Il est plus simple d'ajouter des nouveaux composants
- La solution est parfois trop généraliste (un composite accepte n'importe quel composant)

Compromis d'implémentation

- Référence explicite au parent
- Partage de composants
- Où spécifier la gestion de composants:
  - Composant?
  - Composite?
- Utilisation du Composite comme une mémoire tampon

Composite