Composite

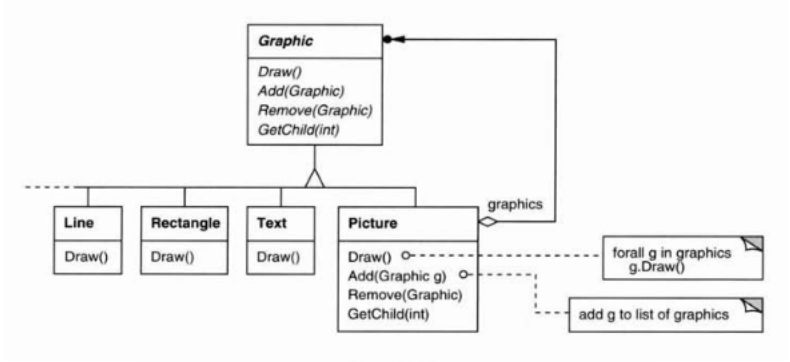
Baris Aktemur
CS 534| Ozyegin University

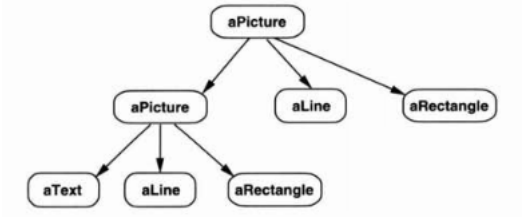
Contents are from “Design Patterns” by Gamma, Helm, Johnson, Vlissides

Composite

- Intent
  - Compose objects into tree structures to represent part-whole hierarchies. Composite lets clients treat individual objects and compositions of objects uniformly.
- The key to the Composite pattern is an abstract class that represents both primitives and their containers.

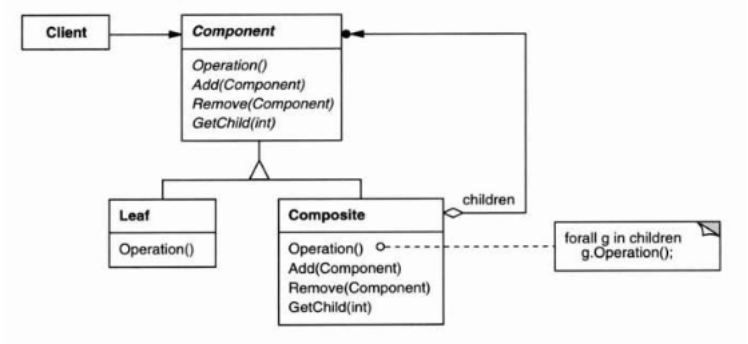
Composite





3

Structure



Consequences

- makes the client simple.
- makes it easier to add new kinds of components.
- can make your design overly general.
- maximal interface for the Composite.
  - child management operations (e.g. add, remove) can be pushed down to Composite, but then Leaf and Composite will have different interfaces
  - what is the default behavior for add and remove in Leaf?

Implementation

- Explicit parent references
- Sharing components
  - Problematic if keeping reference to parent: multiple parents
- Putting the child pointer in the base class incurs a space penalty for every leaf, even though a leaf never has children.
- It's usually best to make a Composite responsible for deleting its children when it's destroyed.