The Composite Pattern

Structure & Functionality

Douglas C. Schmidt
Learning Objectives in This Lesson

- Recognize how the *Composite* pattern can be applied to make the expression tree more uniform & extensible.
- Understand the structure & functionality of the *Composite* pattern.
Structure & Functionality of the Composite Pattern
**Intent**

- Treat individual objects & multiple, recursively-composed objects uniformly

See [en.wikipedia.org/wiki/Composite_pattern](http://en.wikipedia.org/wiki/Composite_pattern)
Applicability

- Objects must be composed recursively

\[ e.g., \text{Composite\_Binary\_Node} \text{ contains other types of nodes.} \]
Applicability

- Objects must be composed recursively
- And no distinction between individual & composed elements

*e.g., Leaf_Nodes & Composite Nodes all share the same API.*
Applicability

- Objects must be composed recursively
- *And* no distinction between individual & composed elements
- *And* objects in structure can be treated uniformly

*e.g., Leaf Nodes & Composite *Nodes are (largely) treated the same by operations on a tree.*

See upcoming lessons on “The Iterator Pattern” & “The Visitor Pattern.”
The diagram illustrates the Composite pattern from the GoF (Gang of Four) design patterns. It shows the structure and participants of the Composite pattern:

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

The diagram depicts the relationships between these components, with arrows indicating the flow of data and interactions.
Composite

GoF Object Structural

Structure & participants

Client

Component

Operation()
Add(Component)
Remove(Component)
GetChild(int)

Leaf

Operation()

Composite

Operation()
Add(Component)
Remove(Component)
GetChild(int)

forall g in children
g.Operation();

Some program (also Bridge “abstraction”)

See upcoming lessons on the Bridge pattern.
**Structure & participants**

- **Client**
- **Component**
  - Operation()
  - Add(Component)
  - Remove(Component)
  - GetChild(int)
- **Leaf**
  - Operation()
- **Composite**
  - Operation()
  - Add(Component)
  - Remove(Component)
  - GetChild(int)

**GoF Object Structural**

forall g in children
g.Operation();
Structure & participants

**Client**

**Component**
- Operation()
- Add(Component)
- Remove(Component)
- GetChild(int)

**Leaf**
- Operation()

**Composite**

**Composite_Unary_Node**, **Composite_Binary_Node**, **Composite_Add_Node**, etc.

forall g in children
g.Operation();