The Composite Pattern

Motivating Example

Douglas C. Schmidt
• Recognize how the *Composite* pattern can be applied to make the expression tree object structure more uniform & extensible.

Learning Objectives in This Lesson

- Composite_Binary_Node
- Composite_Unary_Node
- Leaf_Node
- Composite_Add_Node
- Composite_Subtract_Node
- Composite_Multiply_Node
- Composite_Divide_Node
- Component_Node
- Composite_Negate_Node
Motivating the Need for the Composite Pattern in the Expression Tree App

Douglas C. Schmidt
Learning Objectives in This Lesson

- Recognize how the *Bridge* pattern can be applied to make the expression tree structure easier to access & evolve transparently.
- Understand the structure & functionality of the *Bridge* pattern.
Structure & Functionality of the Bridge Pattern

Douglas C. Schmidt
Intent

- Separate an abstraction from its implementation(s) so the two can vary independently

See [en.wikipedia.org/wiki/Bridge_pattern](en.wikipedia.org/wiki/Bridge_pattern)
Applicability

- When the abstraction & extensible implementation(s) can vary independently

\[ \text{e.g., the} \ \textit{Expression\_Tree} \ \text{service can be refined without affecting clients} \]
The Bridge Pattern

Structure & Functionality

Douglas C. Schmidt
Bridge

GoF Object Structural

Structure & participants

Client

Abstraction

operation() 

Imp

Implementor

operationImp()

RevisedAbstraction

ConcretImplementorA

operationImp()

ConcretImplementorB

operationImp();

imp.operationImp();
Bridge

GoF Object Structural

Applicability

- When the abstraction & extensible implementation(s) can vary independently
- When there’s a need to change implementor hierarchies at design-time or runtime without breaking client code

*e.g., the Component_Node implementor hierarchy can change without affecting clients.*
Applicability

- When the abstraction & extensible implementation(s) can vary independently
- When there’s a need to change implementor hierarchies at design-time or runtime without breaking client code

*e.g., the Component_Node implementor hierarchy can change without affecting clients.*
Structure & participants

**Abstraction**
- operation()

**Implementor**
- operationImp()

**RefinedAbstraction**

**ConcreateImplementorA**
- operationImp()

**ConcreateImplementorB**
- operationImp()
Structure & participants

Expression_Tree

Abstraction
  operation()
  imp

Implementor
  operationImp()

RefinedAbstraction

ConcreteImplementorA
  operationImp()

ConcreteImplementorB
  operationImp()
Bridge GoF Object Structural

Structure & participants

Client

Abstraction

operation()

Implementor

operationImp()

imp.operationImp();

imp

RefinedAbstraction

ConcreateImplementorA

operationImp()

ConcreateImplementorB

operationImp()

Leaf_Node, Composite_Add_Node, Composite_Subtract_Node, etc.
**Structure & participants**

Bridge

GoF Object Structural

```
Bridge
GoF Object Structural
Structure & participants

Client

Abstraction
operation()  

Implementor
operationImp()

Imp.
operationImp();

RefinedAbstraction

ConcreteImplementorA
operationImp()

ConcreteImplementorB
operationImp()

Synchronized_Expression_Tree,
Instrumented_Expression_Tree, etc.
```