The Visitor Pattern

Structure & Functionality

Douglas C. Schmidt
Learning Objectives in This Lesson

- Recognize how the Visitor pattern can be applied to enhance expression tree operation extensibility.
- Understand the Visitor pattern.

**Visitor** is one of the most complicated GoF patterns (along with State).
Structure & Functionality of the Visitor Pattern

Douglas C. Schmidt
**Intent**

- Centralize operations on an object structure so that they can vary independently, but still behave polymorphically

Operations:
- **Operation 1:** Print all values of nodes in tree
- **Operation 2:** Evaluate “yield” of nodes in tree
- **Operation 3:** Conduct semantic analysis of tree
- **Operation 4:** ...

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

- When classes involve many unrelated operations

Visitor

Operation 1: Print all values of nodes in tree

Operation 2: Evaluate “yield” of nodes in tree

Operation 3: Conduct semantic analysis of tree

Operation 4: ...

- 5
- 3
+ 4
Applicability

• When classes involve many unrelated operations
• Class relationships in structure rarely change, but operations on them *do* change

**Visitor**

- Operation 1: Print all values of nodes in tree
- Operation 2: Evaluate “yield” of nodes in tree
- Operation 3: Conduct semantic analysis of tree
- Operation 4: ...

**Stable**
Applicability

- When classes involve many unrelated operations
- Class relationships in structure rarely change, but operations on them do change
- Algorithms keep the state that’s updated during traversal

Visitor

GoF Object Behavioral

Operation 1: Print all values of nodes in tree

Operation 3: Conduct semantic analysis of tree

Operation 2: Evaluate “yield” of nodes in tree

Operation 4: ...

- 
+ 
5 
3 
4
Visitor

Structure & participants

- **Visitor**
  - `visitConcreteElement1ConcreteElement1`
  - `visitConcreteElement2ConcreteElement2`

- **ConcreteVisitor**
  - `visitConcreteElement1ConcreteElement1`
  - `visitConcreteElement2ConcreteElement2`

- **ObjectStructure**
  - `accept(Visitor)`

- **Element**
  - `accept(Visitor v)`
  - `v.visitConcreteElement1ConcreteElement1`
  - `v.visitConcreteElement2ConcreteElement2`

- **ConcreteElement1**
  - `accept(Visitor v)`

- **ConcreteElement2**
  - `accept(Visitor v)`
  - `v.visitConcreteElement1ConcreteElement1`
  - `v.visitConcreteElement2ConcreteElement2`
Visitor

Structure & participants

Visitor

Client

ObjectStructure

Element

accept(Visitor)

ConcreteElement1

accept(Visitor v)

ConcreteVisitor

visitConcreteElement1(ConcreteElement1)

visitConcreteElement2(ConcreteElement2)

ConcreteElement2

accept(Visitor v)

v.visitConcreteElement1(this)

v.visitConcreteElement2(this)
Visitor

GoF Object Behavioral

Structure & participants

Visitor

Client

ObjectStructure

Element

accept(Visitor)

ConcreteElement1

accept(Visitor v)

ConcreteElement2

v.visitConcreteElement1(this)

v.visitConcreteElement2(this)

ConcreteVisitor

visitConcreteElement1(ConcreteElement1)
visitConcreteElement2(ConcreteElement2)

Evaluation_Visitor, Print_Visitor, etc.
Visitor

Structure & participants

**Component_Node**

**ObjectStructure**

**Element**

- `accept(Visitor)`

**ConcreteElement1**

- `accept(Visitor v)`

**ConcreteElement2**

- `accept(Visitor v)`

**ConcreteVisitor**

- `visitConcreteElement1(ConcreteElement1)`
- `visitConcreteElement2(ConcreteElement2)`

**Visitor**

- `visitConcreteElement1(ConcreteElement1)`
- `visitConcreteElement2(ConcreteElement2)`

**Client**
Structure & participants

Visitor

- visitConcreteElement1(ConcreteElement1)
- visitConcreteElement2(ConcreteElement2)

ConcreteVisitor

- visitConcreteElement1(ConcreteElement1)
- visitConcreteElement2(ConcreteElement2)

ObjectStructure

Element

accept(Visitor)

Composite_Add_Node, Composite_Subtract_Node, Composite_Unary_Node, Leaf_Node, etc.

ConcreteElement1

accept(Visitor v)

ConcreteElement2

accept(Visitor v)

v.visitConcreteElement1(this)

v.visitConcreteElement2(this)
Visitor’s dual inheritance hierarchy + dynamic/static polymorphism is tricky.
Collaborations

This generic object interaction diagram doesn’t shed much light on Visitor!