Composite Pattern

Chapter Content

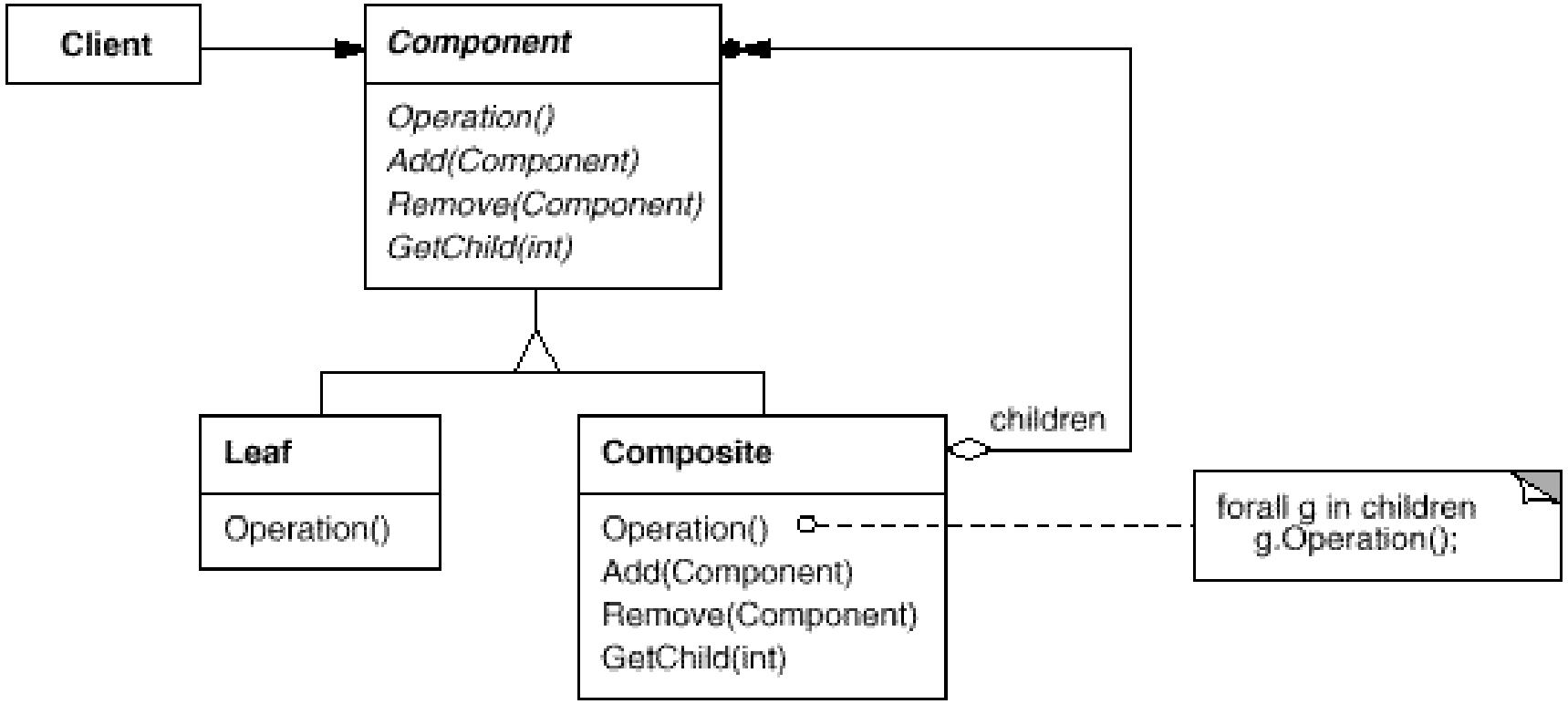
InterBit
Training & Consulting Ltd.

- The Composite Pattern – Overview
- Composite Pattern UML Diagram
- Composite Example
- Leaf representation
- Technical considerations
- Exercise

The Composite Pattern – Overview

- **Recursive tree structures**
- A component may be:
  - Simple (leaf).
  - Or composed of a collection of other components (which may be composite too).
- Define a common base-class for both simple & complex components.
  - so that users can refer to them uniformly (e.g draw a shape, be it simple or composite).

Composite Pattern UML Diagram



Component may be simple, or composed of other components (which, recursively, may be composite too)

Note: class "Leaf" is optional (see discussion later).

Composite Example

- XML DOM documents.
- Swing components.
- Tree representing arithmetical expression (see the Interpreter pattern).
- File/directory structure.

Leaf representation

InterBit Training & Consulting Ltd.

Two approaches to leaf representation:

1. Define a special "Leaf" class (as in the diagram).
2. Omit class Leaf , and use a single class
   (so Leaf is just a component that happens to have 0 sons).
   - The 2nd approach allows leaves to dynamically become composite (by adding sons).
   - java.awt took the 1st approach, while javax.swing takes the 2nd (so a JButton may hold a smaller JButton inside.
   - XML parsers take the 1st approach.

Technical considerations

- **Pointer to parent:**
  - Children may hold pointers to their parent. This increases memory requirements, but may speed up traversal algorithms.
- **Child iteration:**
  - A node may allow iteration over its children using an *iterator* rather than *getChild(int index)*.