Visitor pattern

The visitor pattern allows you to define new operations for an object structure without changing the object structure.

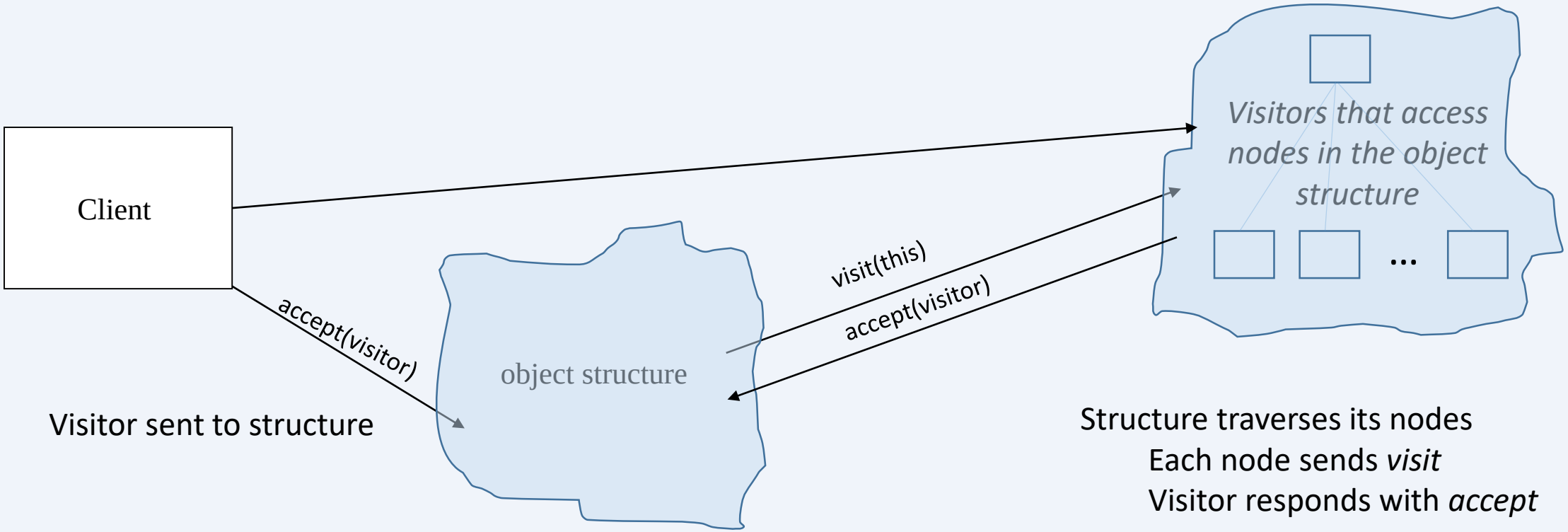
- Although … some method infrastructure is needed

How:

- Define two hierarchies: one for the object structure and one for the visitors
- In the visitor hierarchy create a subclass for each new operation

- A node to be visited receives an *accept(visitor)* message
- The node sends a *visit(this)* message to a visitor

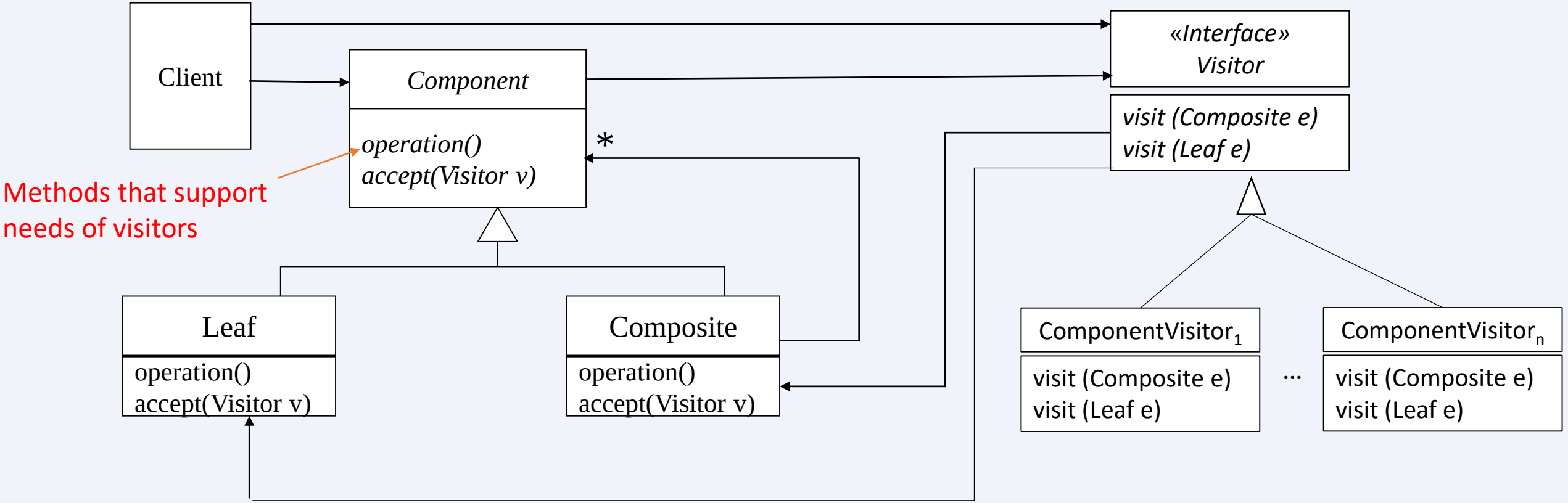
Behaviour in the Visitor pattern



Visitor sent to structure

Structure traverses its nodes
Each node sends *visit*
Visitor responds with *accept*

Visitor applied to Composite

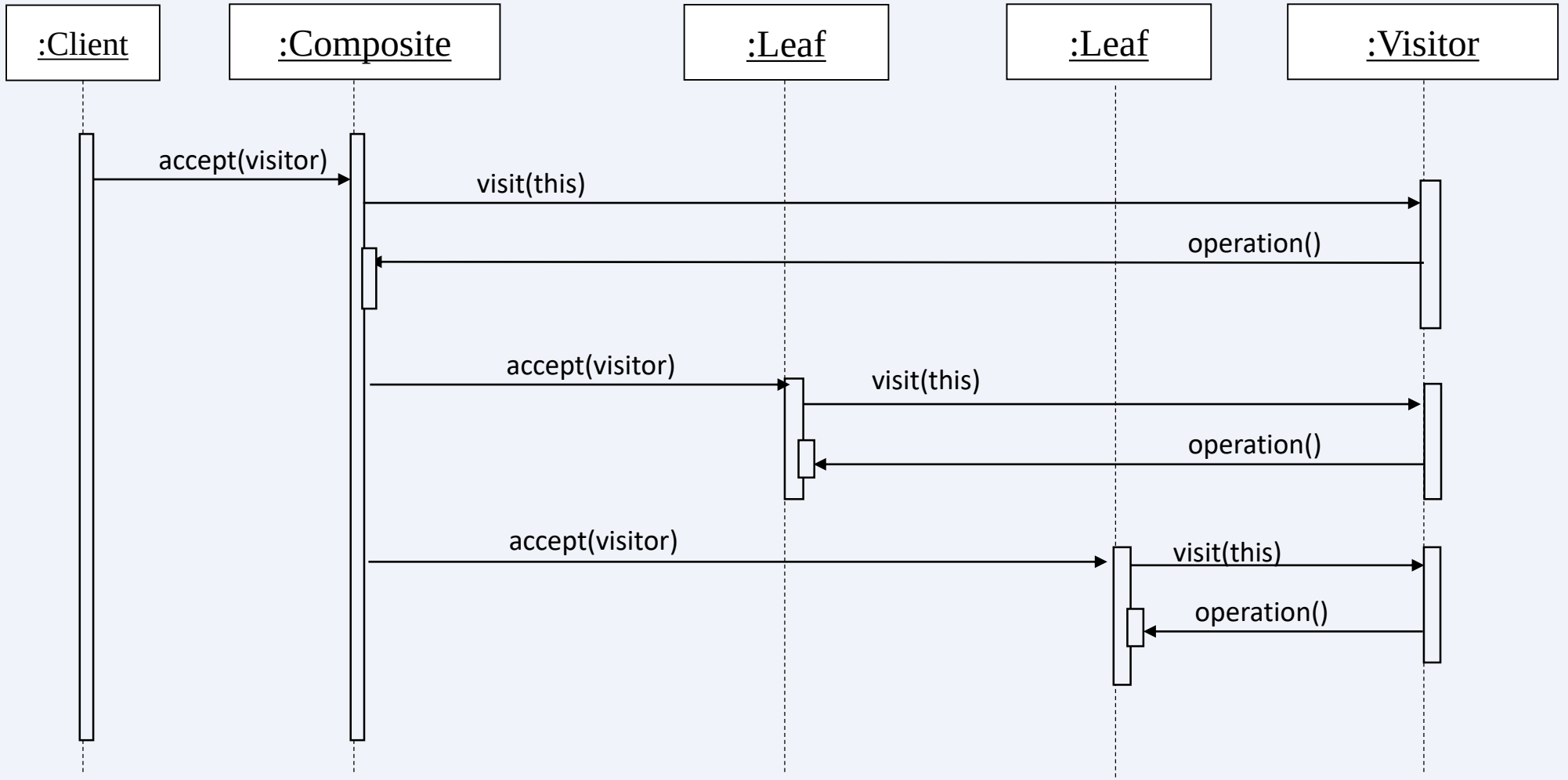


Component
Each component defines an *accept* method that takes a visitor as an argument

Visitor
Each visitor defines a *visit* method that takes a concrete component as an argument (typically one method for each component class)

Visitor applied to Composite

Suppose we have a component hierarchy that has one composite that has two leafs.
Basic outline of messages sent:



Example: Visitor applied to Menu Example

1. See 3913 web page

2. Wikipedia, java example

2. A useful webpage with more extensive example
http://www.ooxs.be/EN/java/design-patterns/Composite%20and%20Visitor.html#design-pattern-Visitor