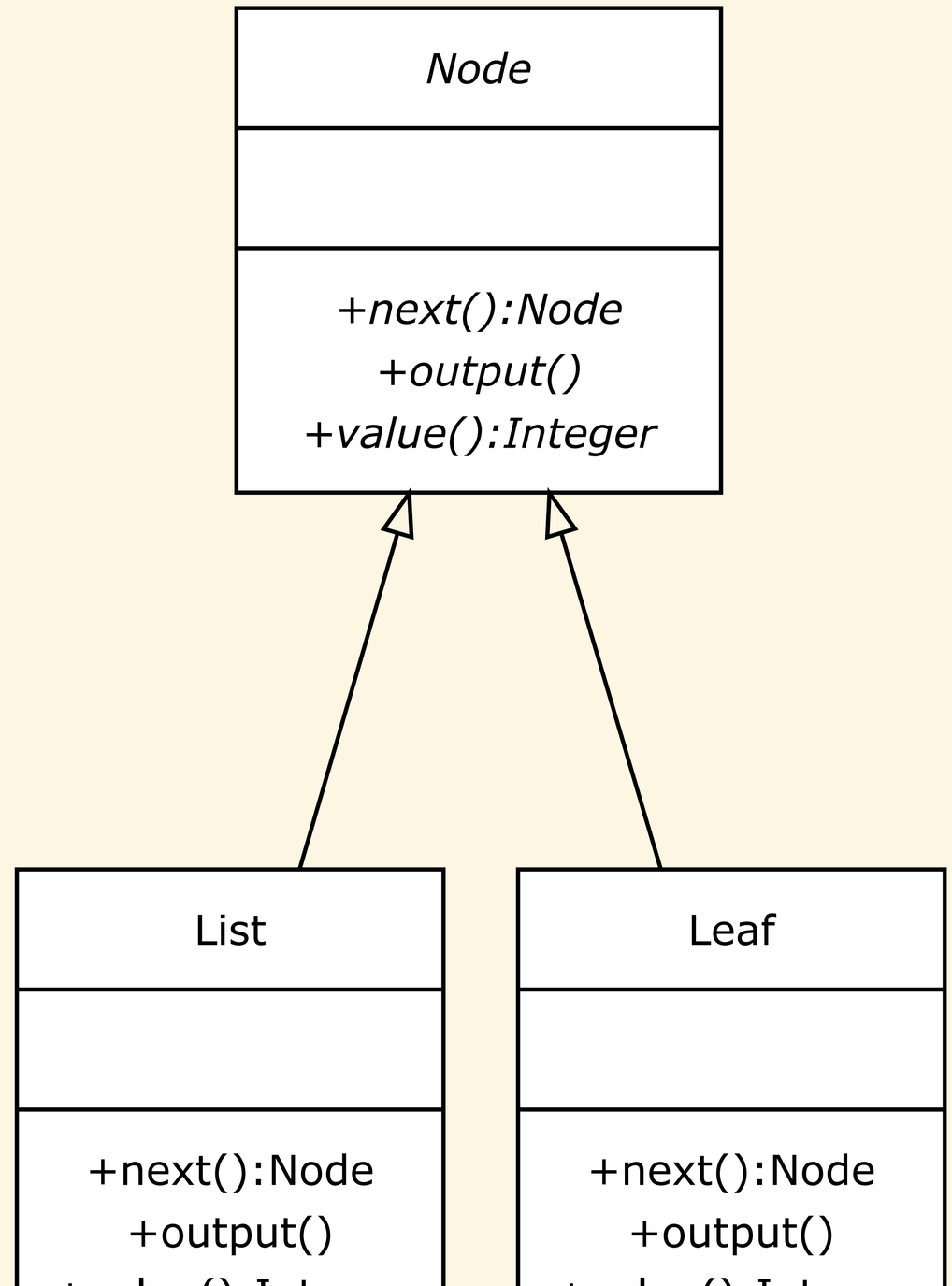
Object-Oriented Programming

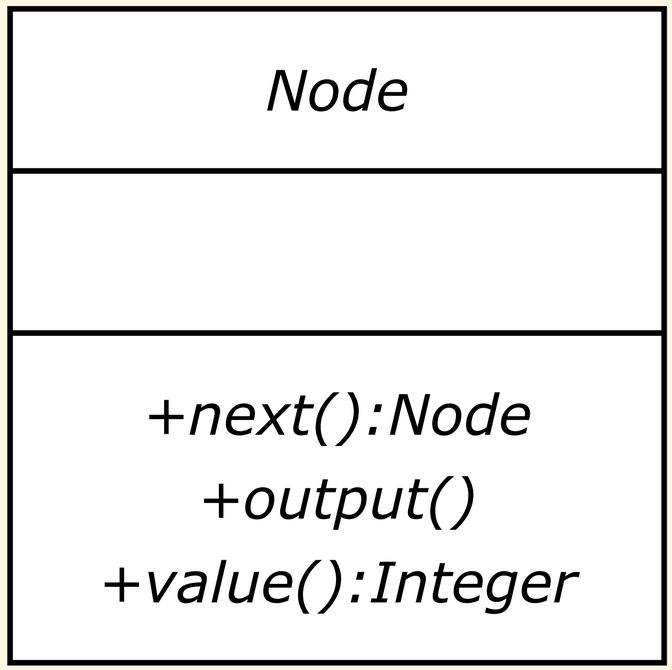
Design Pattern Visitor

Michael L. Collard, Ph.D.

Department of Computer Science, The University of Akron

OutputList


Node
+next():Node
+output()
+value():Integer
List
+next():Node
+output()
Leaf
+next():Node
+output()



Issues

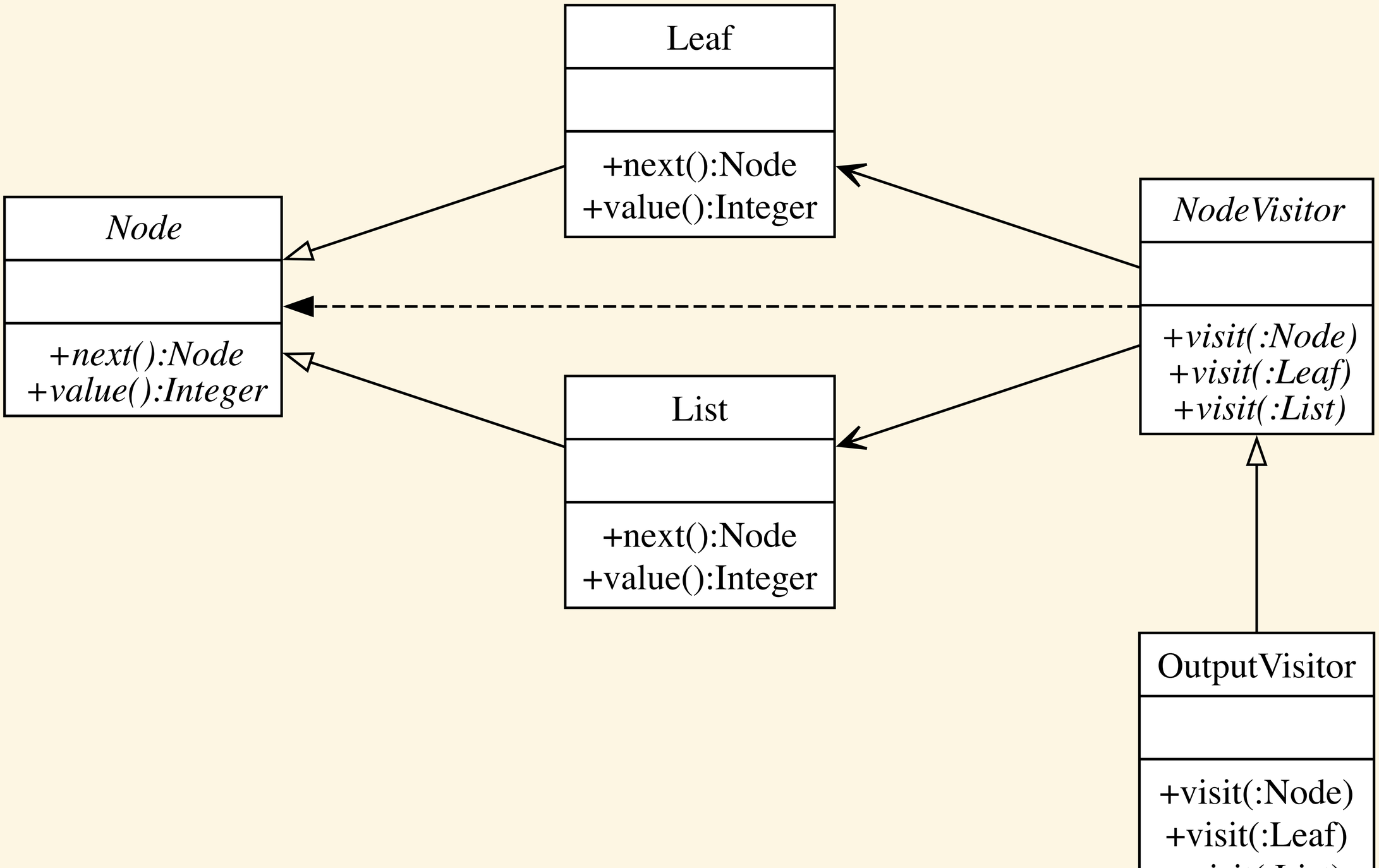
- Works well for a single operation, e.g., `output()`
- Tightly couples `output()` to list structure and handling
- Adding additional operations requires changing `Node`, `Leaf`, and `List`, and any of their derived classes
- Supporting divergent applications means many operations are not related to the rest of the operations
- Want the ability to add operations without changing the current list class

Visitor Design Pattern

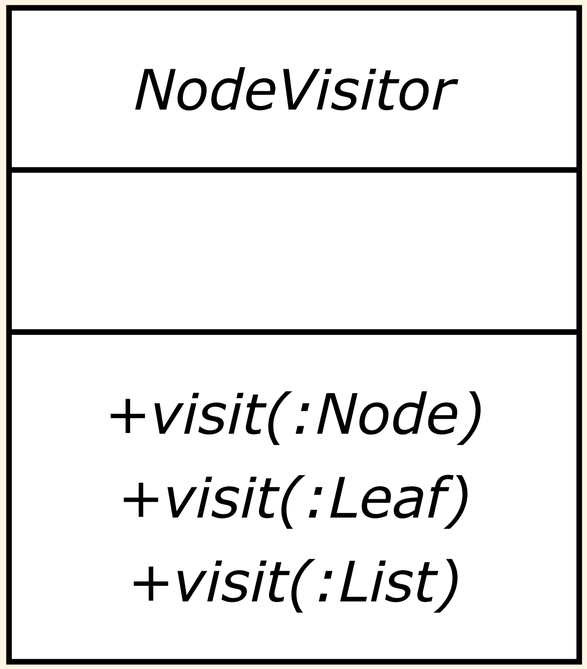
Add operations to classes without changing them

- Behavioral Pattern
- Emulates *multiple dispatch*

OutputList Problem



Problem



- When `visit()` is passed a `Node*`, only the `visit(Node*)` is called, even if the parameter is a `Leaf` or `List`

- **Polymorphism is not working!**

Dispatch

single dispatch	Polymorphism depends on the calling object only
multiple dispatch	Polymorphism depends on both the calling object and the argument objects

single dispatch

Polymorphism depends on the calling object only

E.g., in `object->operation(argument);`

Polymorphism works for `object`

Polymorphism **does not apply to `argument`**

C++ (and most equivalent languages) directly support only *single dispatch* using a *vtable*

multiple dispatch

Polymorphism depends on both the calling object and the argument objects

e.g., `object->operation(argument);`

Polymorphism works for `object`

Polymorphism works for `argument`

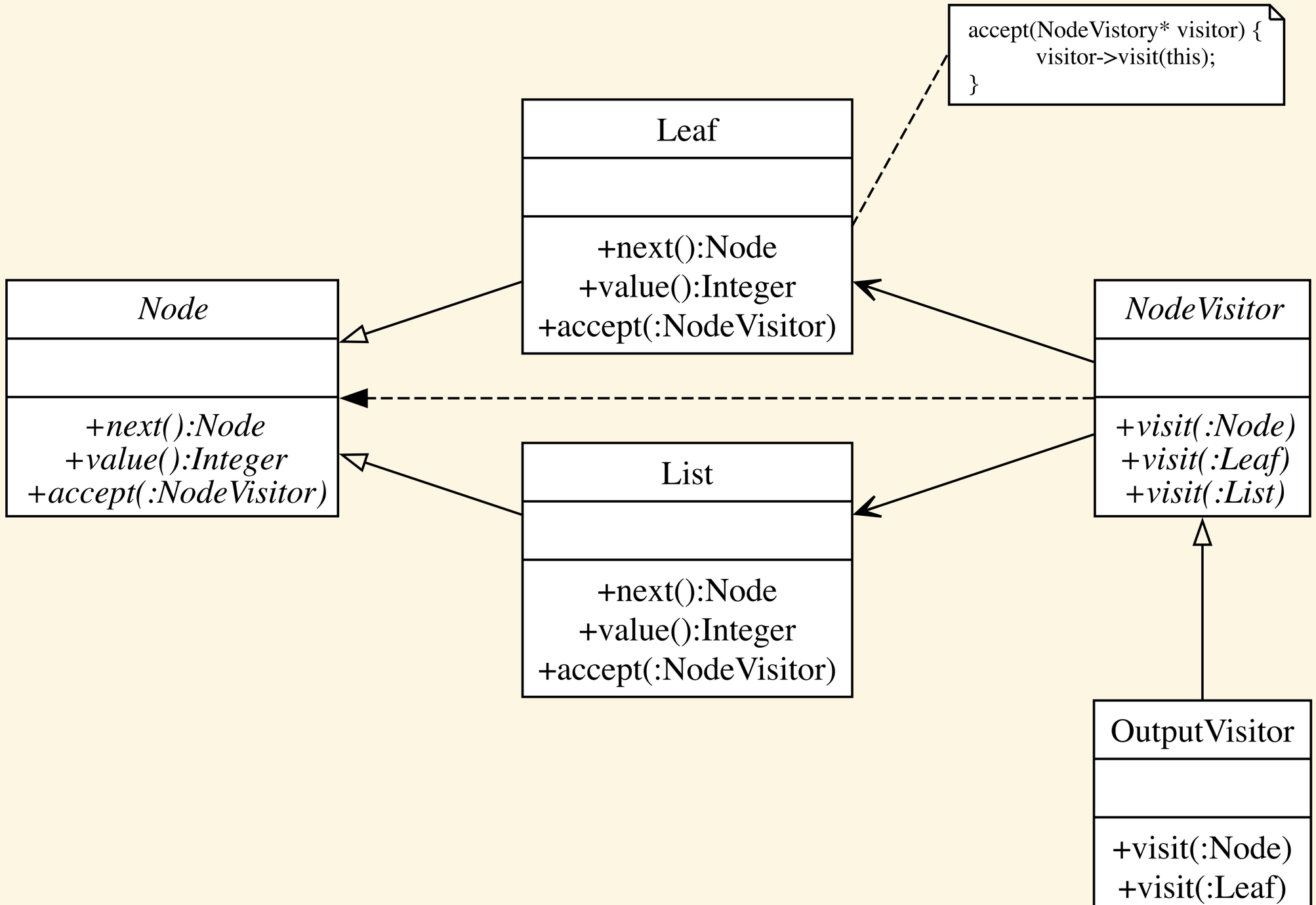
Polymorphism works for both `object` and `argument`

AKA *multimethods*

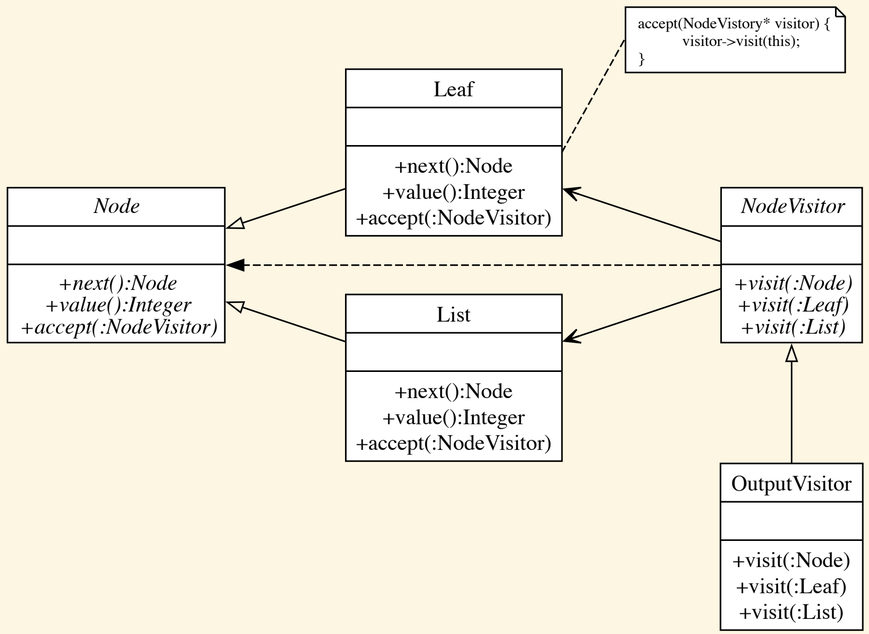
Few languages support *multimethods*, e.g., CLOS, Julia

For the rest of the languages, multiple dispatch is emulated

OutputList Solution



Benefits



- All operations are separated from the list
- Can add operations as needed of many different kinds
- Do not have to change the code or recompile the `Node`, `Leaf`, or `List`
- Can add more subclasses as needed
- May want to make the `visit(Node*)` an empty method that does nothing (a NOP) so a visitor can ignore a particular type of `Node`

Set of Visitors