

# Tecniche di Progettazione: Design Patterns

GoF: Interpreter

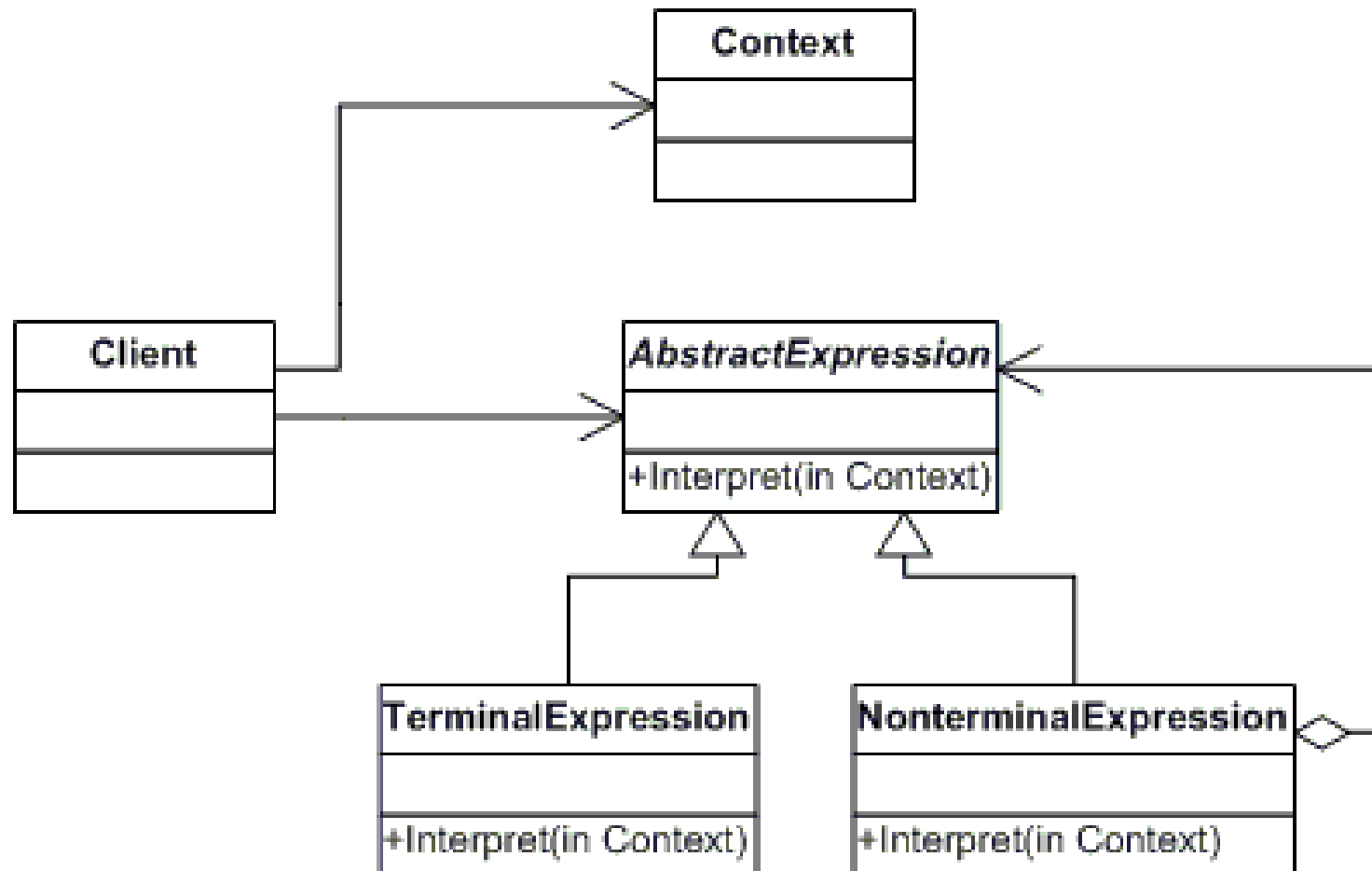
# Intent

---

Given a language, define a representation for its grammar along with an interpreter that uses the representation to interpret sentences in the language.

# Structure of Interpreter

---



# Flow of Interpreter

---

- ▶ Take the **Abstract Syntax Tree**
- ▶ Interpret the tree in the given **Context**
  - ▶ Each non-terminal node will interpret its children and return a result from that.
  - ▶ Terminal nodes return actual values for non-terminal nodes to use.
  - ▶ The context serves as a global data for interpreting the entire tree

# Consequences

---

## Good

- ▶ It's easy to change and extend the grammar
- ▶ Implementing the grammar is easy
- ▶ Easy to evaluate an expression in a new way

## Bad

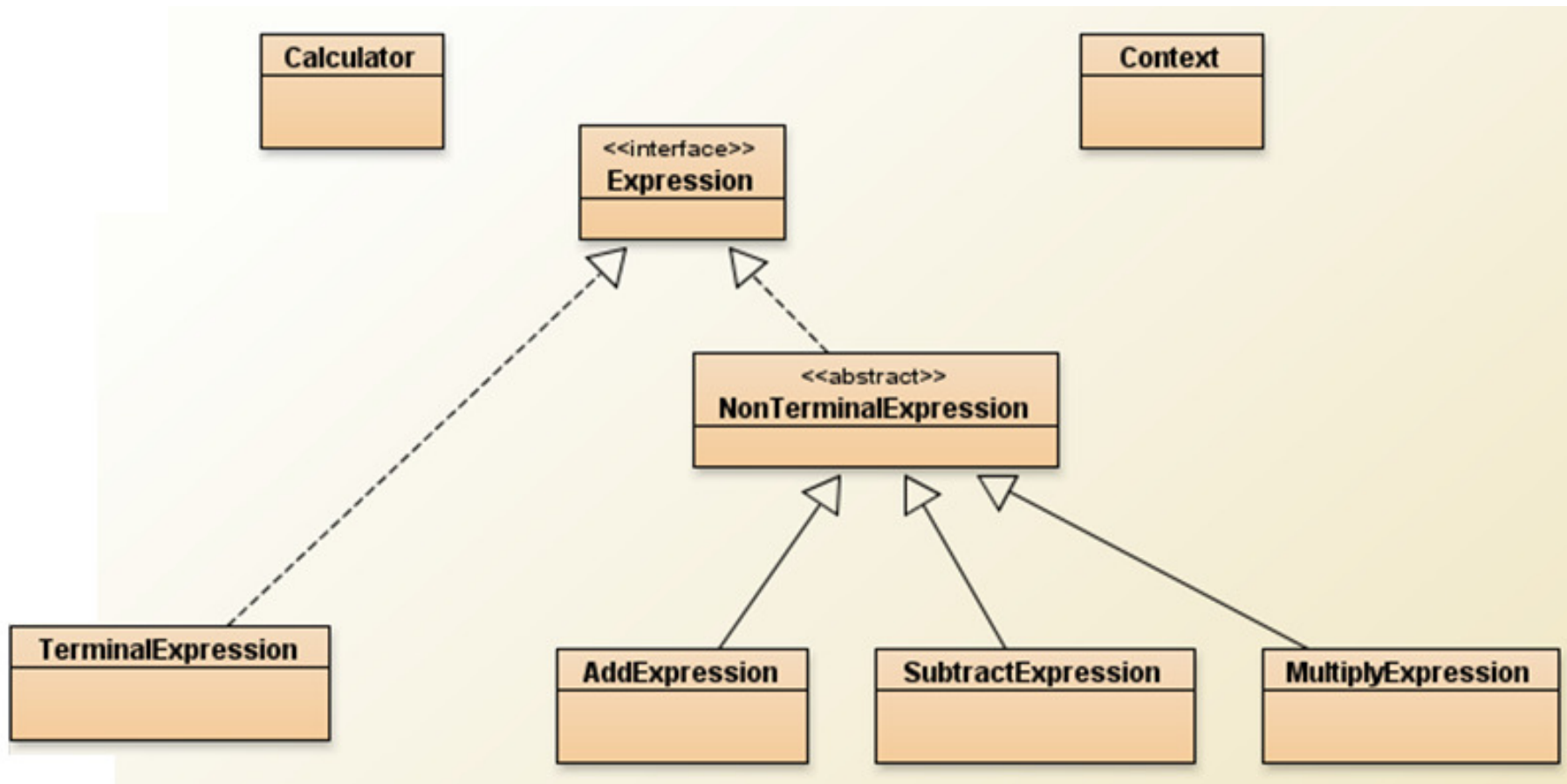
- ▶ Complex grammars are hard to maintain
  - ▶ One class for each rule

# Example: Calculator

---

- ▶ <http://www.java2s.com/Code/Java/Design-Pattern/InterpreterPatternCalculator.htm>
  - ▶ ill-defined because it also performs parsing and syntactic tree construction.
  - ▶ The interpreter starts from the tree.
  - ▶ Redefined in NewCalculator
  
- ▶ Another example in:  
[http://www.dofactory.com/Patterns/PatternInterpreter.aspx#\\_self2](http://www.dofactory.com/Patterns/PatternInterpreter.aspx#_self2)

# Interpreter: Calculator



# Related Patterns

---

- ▶ **Composite**

- ▶ Abstract syntax tree is an instance of Composite Pattern

- ▶ **Flyweight**

- ▶ Shows how to share terminal symbols within the abstract syntax tree

- ▶ **Iterator**

- ▶ The interpreter can use an iterator to traverse the structure

- ▶ **Visitor**

- ▶ Can be used to implement the method interpret: the interpreter is in a separate Visitor object.



# Homework: Tiny

---

- ▶ Complete the exercise, in order to:
  1. Define an Increment and a Decrement commands. Test for by making a sum.