Interpreter Pattern

Armon Saied
MPCS 51050
1 May 2018

Intent:

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

Rather than building ad hoc algorithms, we use an interpreter to act on frequent patterns. E.G. Regular expressions

An Example: Simple Arithmetic Interpreter

In Java we know these are valid “arithmetic” expressions:

- 1000
- 11 / 90
- a = 1
- (a + b) * (3 - x)

And these are not:

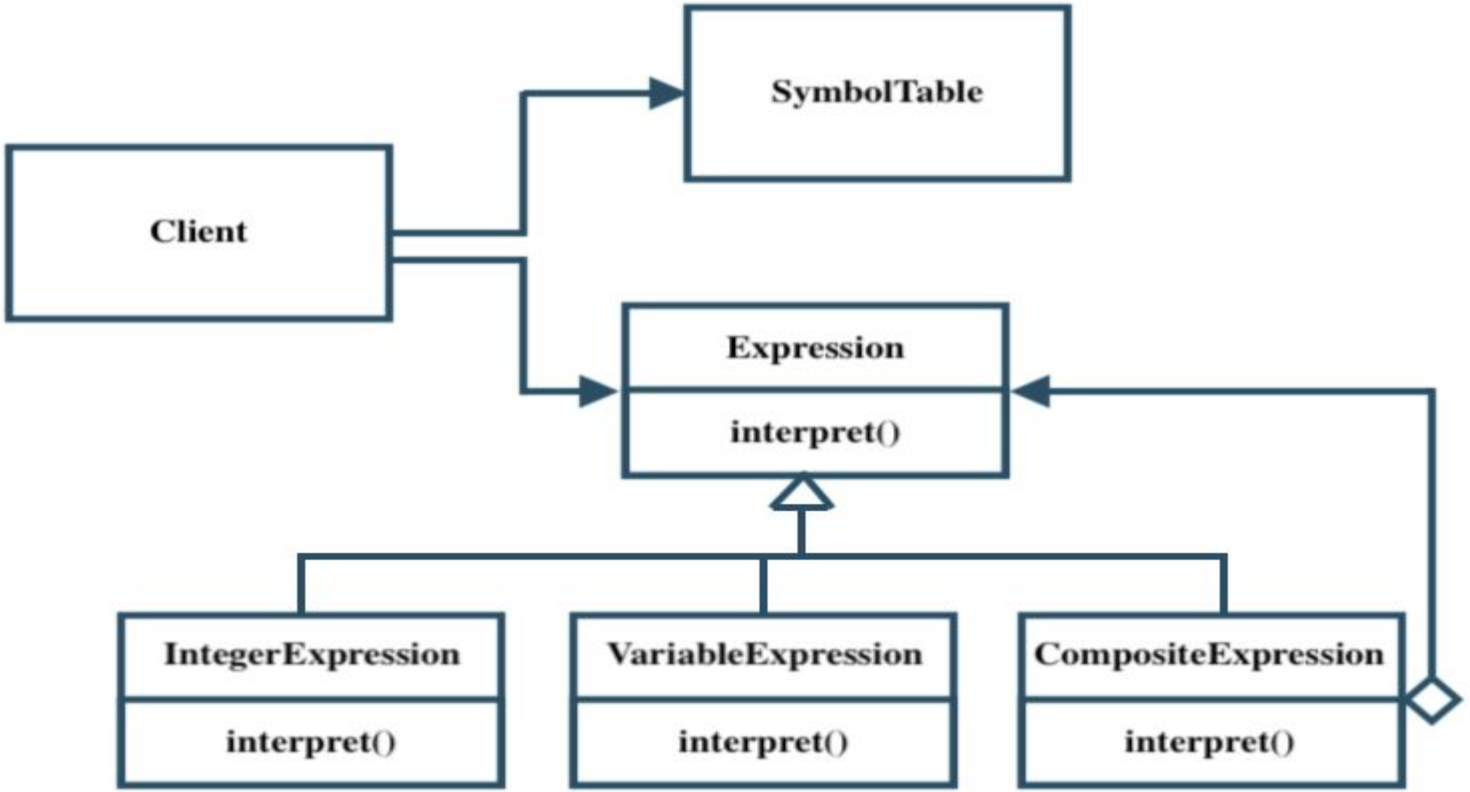
- 10 /
- 19 x

Arithmetic Example Continued...

- Let’s define a rule set...
- Valid expression if it’s:
  - An integer
  - A variable
  - Two expressions separated by an operator
  - An expression in parentheses

- This recursive definition forms basis of the abstract syntax tree (AST) of a language. NOTE: The AST is an *instance* of the composite pattern.

Arithmetic Example - UML



Code

Closing thoughts

- It’s recommended to use on simple grammars only, otherwise class hierarchy too complex.

- Basic idea: “An operation distributed over a class hierarchy based on the Composite pattern” (Gamma) where the class hierarchy specifies a language.

- **When to use:** “Use the Interpreter pattern when there is a language to interpret, and you can represent statements in the language as abstract syntax trees.”

Questions?