Design Patterns

Interpreter Pattern

not-matthias

Contents

1 Description

One class for each symbol:

- Terminal
- Nonterminal

1.1 Terminal Symbols

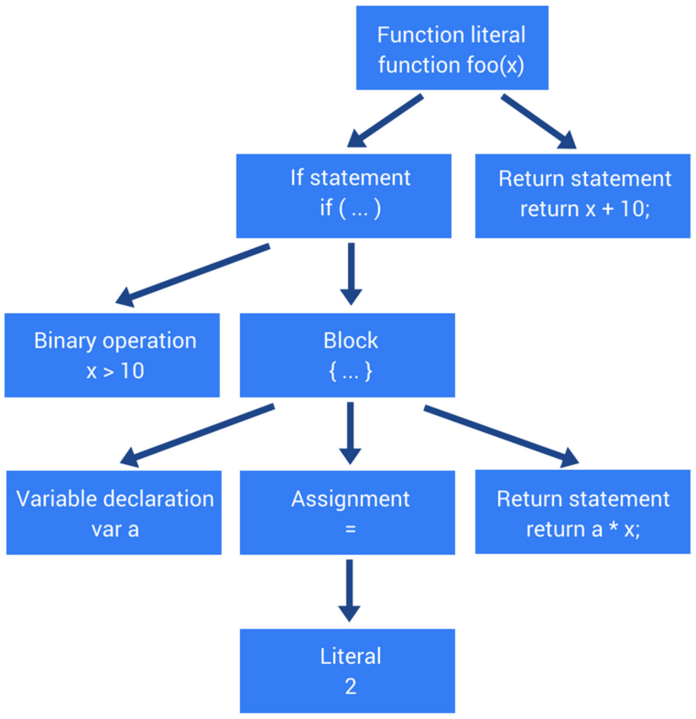
1.2 Nonterminal Symbols

1.3 Abstract Syntax Tree

]subsubsection
For example you have a simple JavaScript function:

```
function foo(x) {
if (x > 10) {
    var a = 2;
    return a * x;
}

return x + 10;
}
```



This AST has been simplified for visualization purposes. The actual AST would be much more complex and contain more data. There's a cool project, where you can show the actual AST of a JavaScript program: https://astexplorer.net/

1.3.1 Usages

- Code Formatters
- Extensions for IDEs

2 Purpose

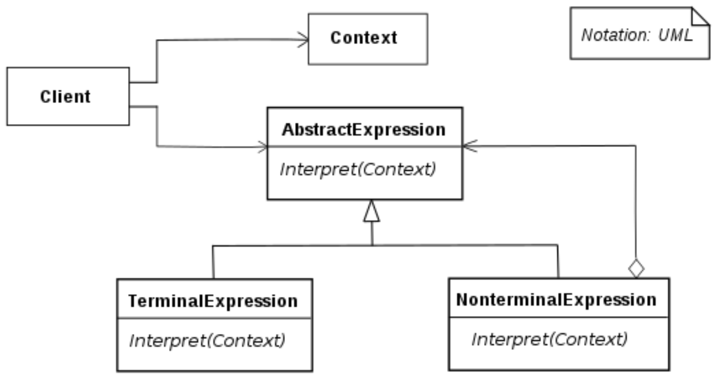
2.1 When should it be used?

Should be used when:

- there's a language to interpret.
  - Represent statements as AST
- the grammar is simple.
  - Use parsers for a large class hierarchy.

  - Doesn't use an AST. Saves space and time.
- efficiency is not a critical concern.
  - More efficient when translating the parse tree to another form.

3 UML



4 Example

5 Usages

5.1 Java

- java.util.Pattern
- java.text.Normalizer
- javax.el.ELResolver
- All subclasses of java.text.Format