Interpreter Pattern

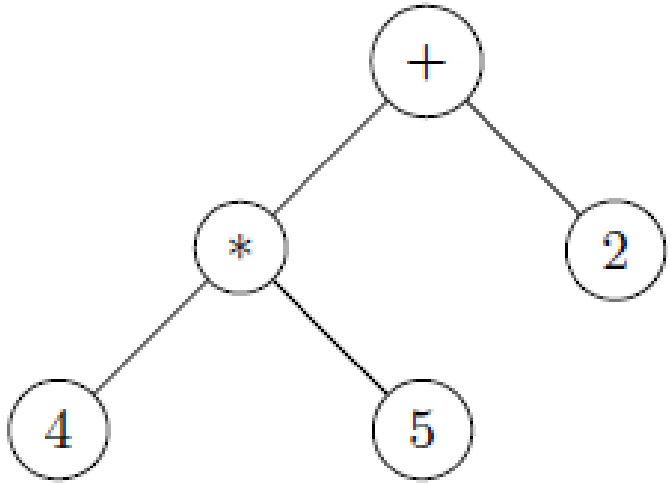
By: Morten, Sune & Yevhen

What is it

- A way to interpret languages
- Languages can be many things
  - Progamming language
  - Human language
  - Unit conversion
  - Mathmatical language
- Well defined input with well defined behaviour
- A way to define the components of a language
- A way to define the rules of a language

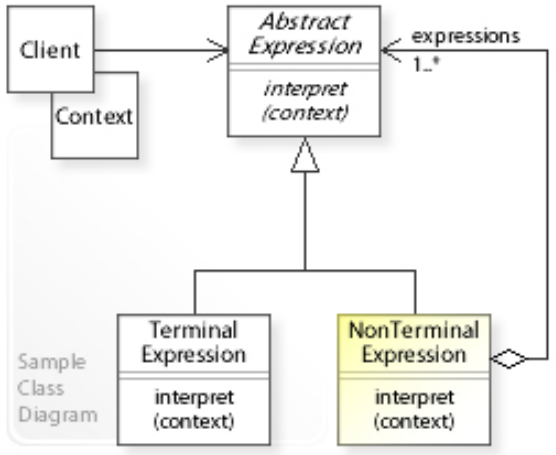
What is the pattern

- Recursive tree structure
- Tells you how to expand an expression into sub components
  - How to evaluate each component
  - How to combine the components



How is it implemented

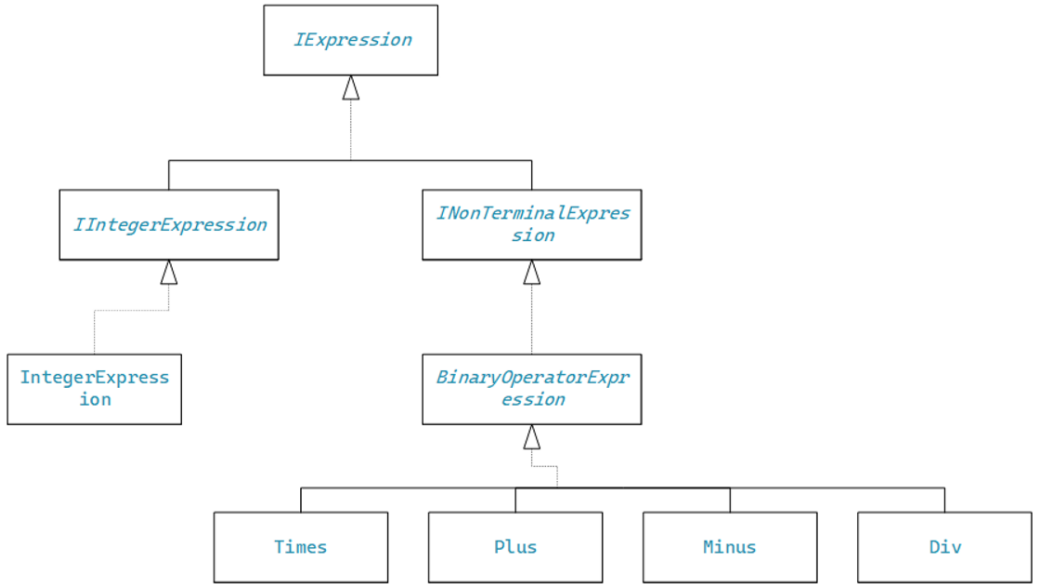
- Terminal and non terminal expressions
- Non terminal expressions consists of terminal and/or non terminal expressions
- Terminal expressions evaluate to one thing



Source: http://w3sdesign.com/?gr=b03&ugr=struct

How did we implement it

- Expressions implemented as a C# interface
- Terminal expressions evaluate to themselves
- Non terminal expressions are implemented as binary operator expressions
- Our implementation is:
  - Reverse Lukasiewicz notation



What is it like

- Composition pattern
  - Same class and tree structure
  - Different purpose
  - Different implementation