Polite Smalltalk

polite programming

Author: Thomas Steinmann

Tutor: Jan Kurš

Why Polite Smalltalk

- Study by Jan K. and Mircea L.
- “Evaluation of the Impact of Identifier Names on the Readability and Maintainability of Programs”

=> A special programming language was needed

What was Polite?

- Sentence Identifiers (multiple words per identifier)
- Supported simple statements & Smalltalk class reference
- Was translated into a single Smalltalk method
- Came with a simple playground
    - No Class Browser

- No Class definition!

The Vision of Polite

```
Character, subclass: Polite Hero
    | health, strength |
    drive off: (an enemy) and save: (a lady)
        if: (self, wins against: an enemy)
        then: (the lady, is freed)

    level up
        strength := strength + 1
my hero, drive off: (the bandits) and save: (the lovely lady)
```

Class Definitions

Method Definitions

Global Message

(Compiler Structure)

Polite Syntax

- All names can be sentence identifiers
- Method calls separated with commas:
  my hero, new ≈ myHero new
- Indentation sensitive
- All smalltalk classes and objects are available:
  Ordered collection, new ≈ OrderedCollection new

Polite Classes & Methods

```
Character, subclass: Polite Hero
    | health, strength |
    drive off: (an enemy) and save: (a lady)
        if: (self, wins against: an enemy)
        then: (the lady, is freed)

    level up
        strength := strength + 1
```

Global Messages & Methods

A message without an explicit receiver:

```
PSGlobal, if: (my hero, wins against: the bandits)
    then: (my hero, saves: the lovely lady)
```

Global method definition

```
if: condition then: (block true)
    condition, if true: [block true, value]
```

Global Messages & Methods

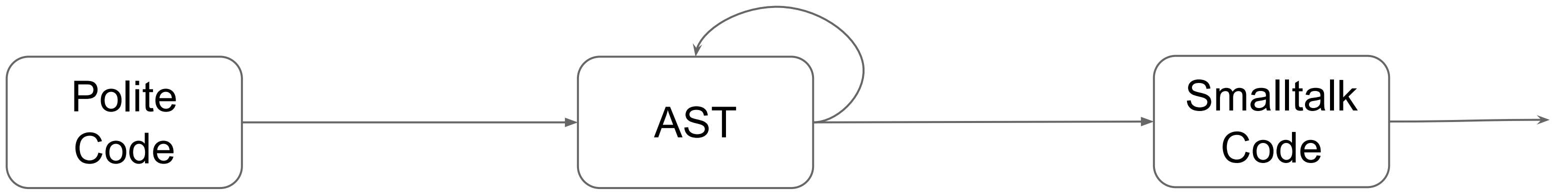
Expression using a global message

```
if: (my hero, wins against: the bandits)
    then: (my hero, saves: the lovely lady)
```

Expression in regular Smalltalk

```
(myHero winsAgainst: theBandits)
    ifTrue: [myHero saves: theLovelyLady]
```

Polite Runtime



Syntax Analysis

create AST

Compilation

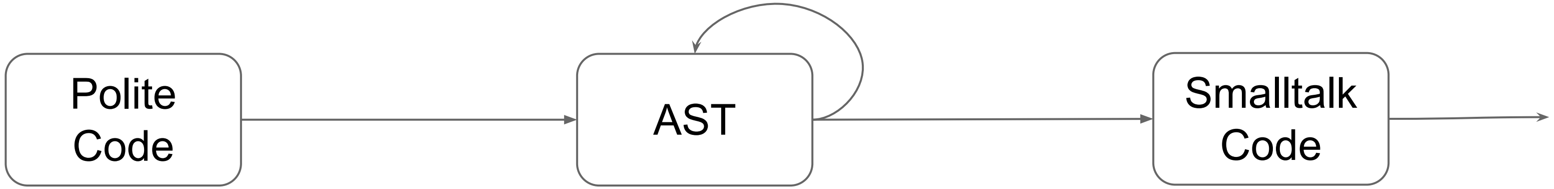
modify AST

compile to Smalltalk

Execution

evaluate & execute

Polite Runtime



Syntax Analysis

create AST

Compilation

modify AST

compile to Smalltalk

Execution

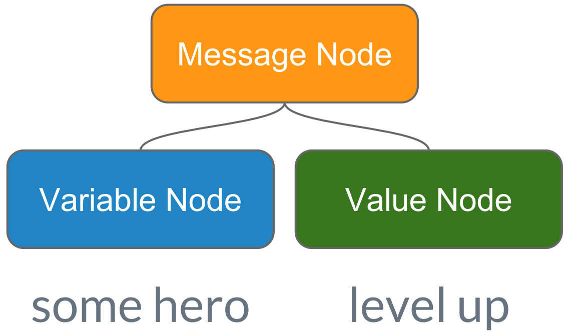
evaluate & execute

Syntax Analysis

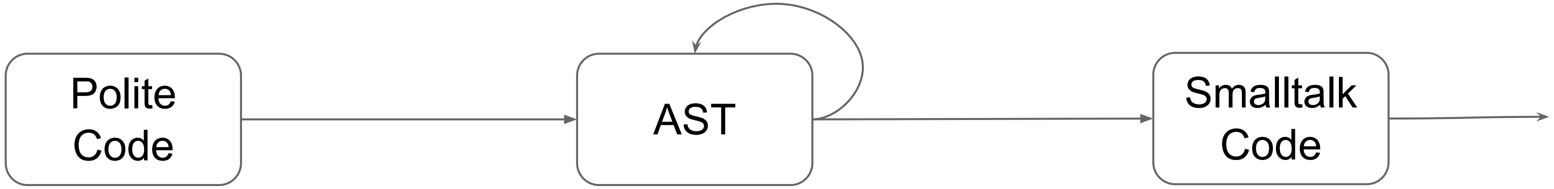
Example: some hero, level up

PSGrammar => [[] [[[[some hero] [[[level up] [ ]]]] []] nil] []]

PSParser =>



Polite Runtime



Syntax Analysis

create AST

Compilation

modify AST

compile to Smalltalk

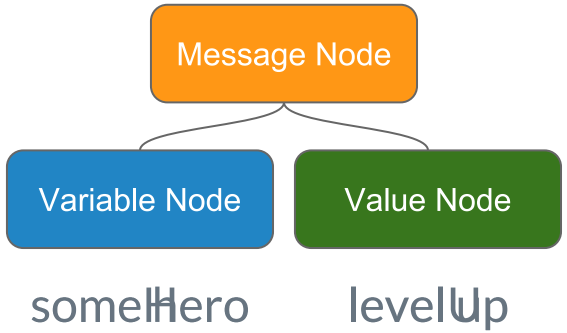
Execution

evaluate & execute

Compilation - PSImpolatizator

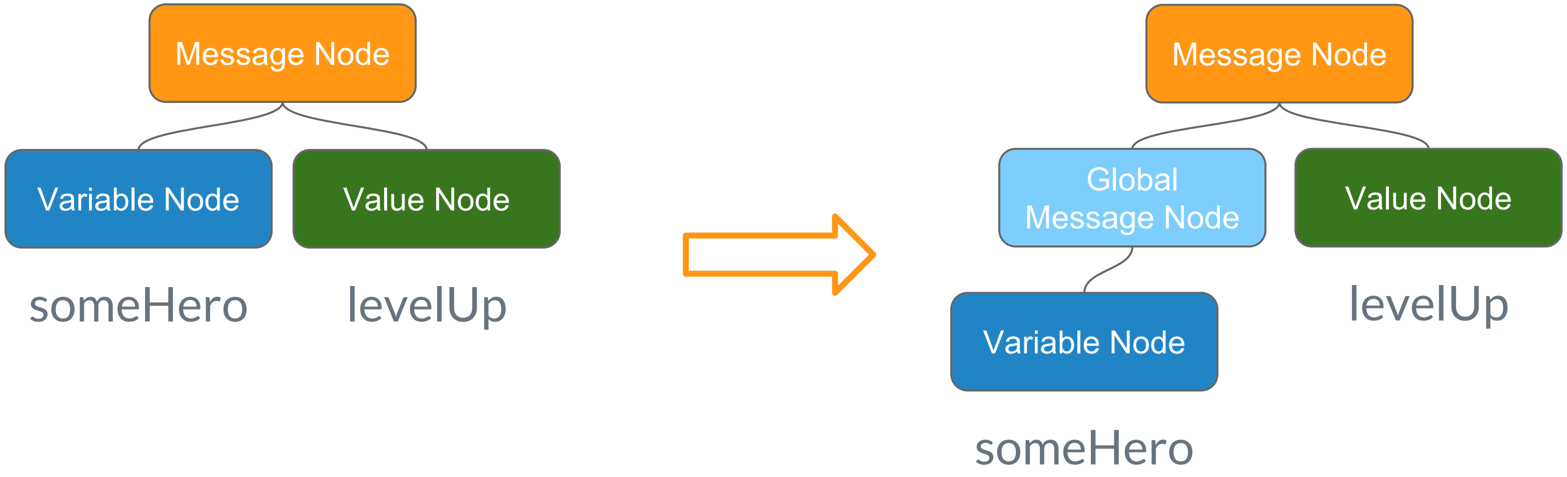
Example: some hero, level up

PSImpolatizator =>



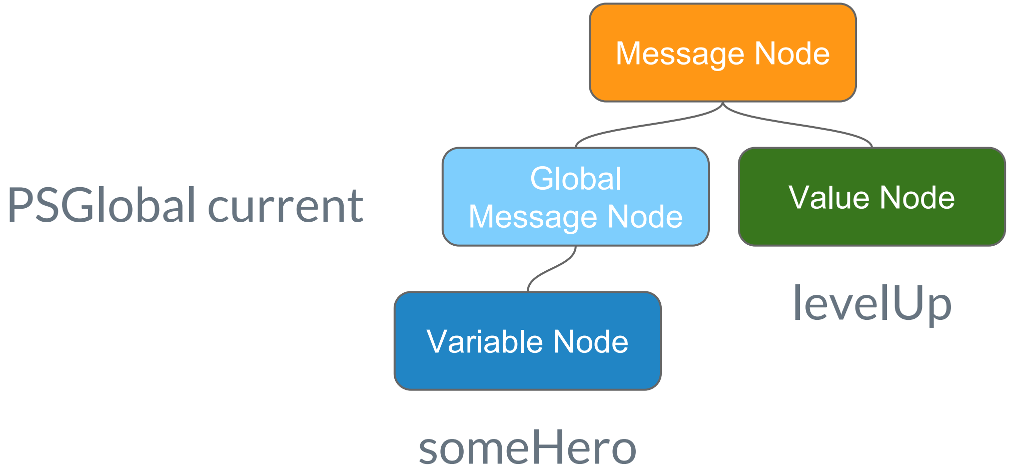
Compilation - PSGlobalMessageSearchVisitor

Example: some hero, level up



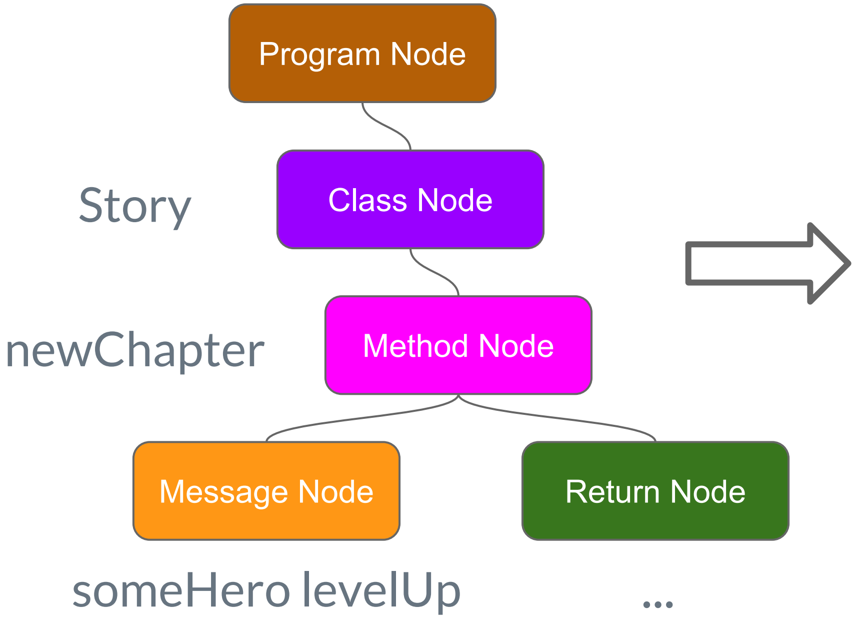
Compilation - PSCompiler

Example: some hero, level up



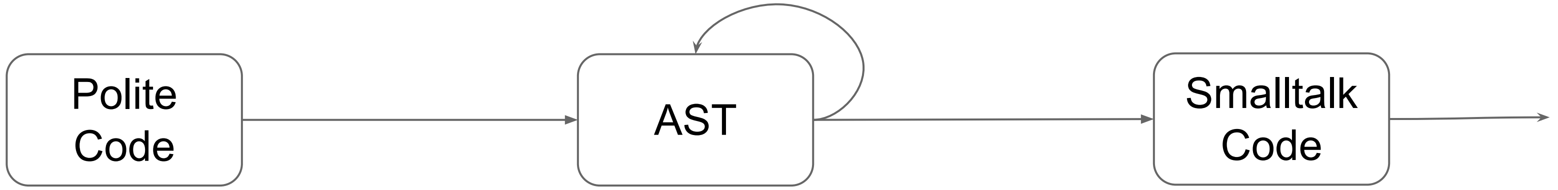
=> ‘PSGlobal current someHero levelUp’

Compilation - PSCompiler



```
Object subclass: Story
    Story>>newChapter
        PSGlobal current someHero levelUp
        ^ ...
```

Polite Runtime



Syntax Analysis

create AST

Compilation

modify AST

compile to Smalltalk

Execution

evaluate & execute

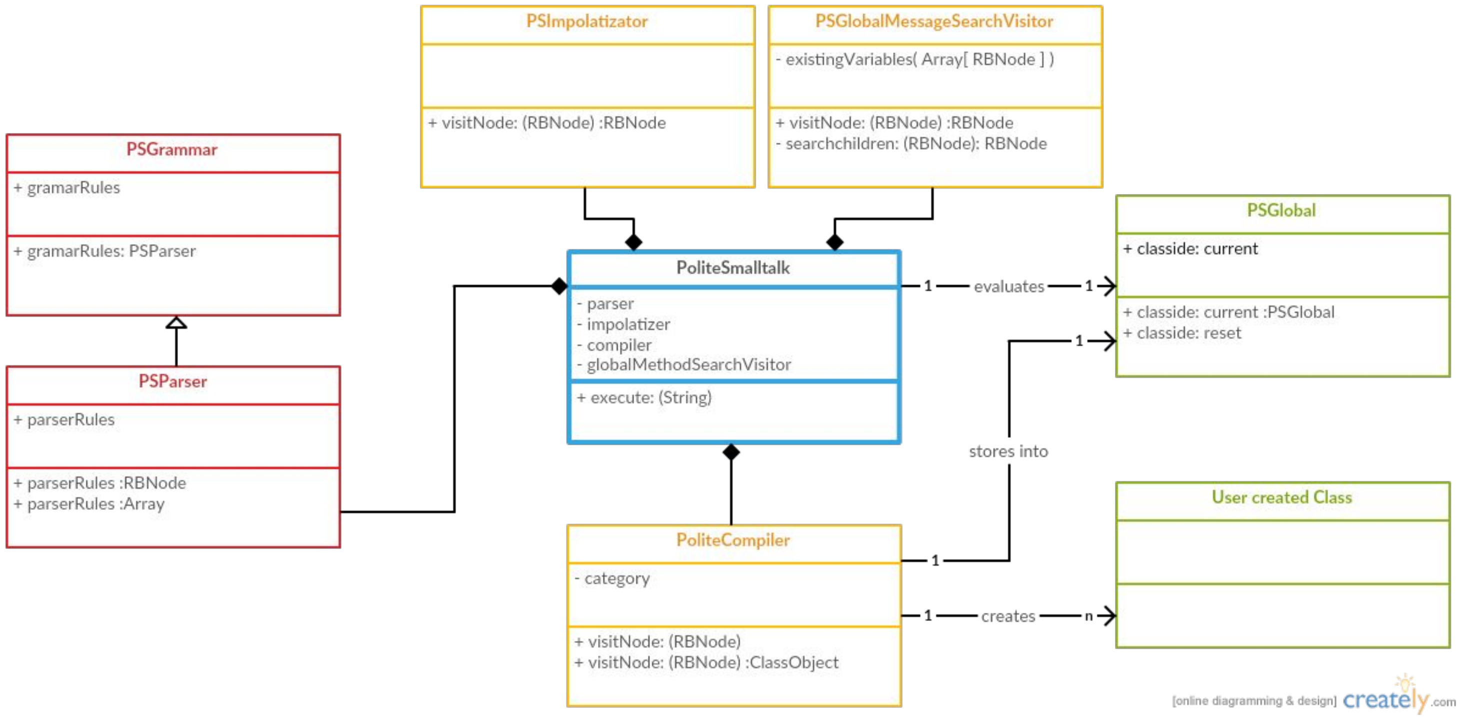
Execution

Example: some hero, level up

```
PSGlobal>>PSMainProgram
    PSGlobal current someHero levelUp
```

Polite Core as a UML

Syntax Analysis Compilation Execution



What is Polite Smalltalk?

- Sentence Identifiers
- Object Oriented Language
- Classes, Methods, globals
- Improved compiler architecture
- Improved Playground

- Parsed and compiled into valid Smalltalk

Some familiar Code

The LO Game from Pharo By Example

Conclusion

- ▷ Polite is an interesting tool to get into Smalltalk programming.
- ▷ Class and method definition are successfully implemented
- ▷ Global methods & messages help with writing even more polite code

- ▷ Learned to love Pharo Smalltalk
- ▷ Lots of insight into Visitors, ASTs, compiling, etc

Questions?

PPFailure: Answer expected at 1

References

http://scg.unibe.ch/archive/papers/Lung13a-Planning.pdf