IMPROVING CODE COMPLETION

work by
Myroslava Romaniuk and Marcus Denker
What is Code Completion?

visitMethodNode: aRBMethodNode

\( ^\text{(self select: self meth)} \)

CompletionProducer

methodNames
Problem

- on a Parser level
- on a Model level
- on a Sorter level
Problem: parser

Old completion used a dedicated parser (Shout) that was originally shared with syntax highlighter.
Solution: use RB Parser

RBParser is used for syntax highlighting
We parse at every keystroke!
We can parse with Syntax Errors
Solution: parseSource

```plaintext
ast := theClass
   ifNil: [ ( RBParsen parseFaultyExpression: source )
         doSemanticAnalysis ]
   ifNotNil: [ ( RBParsen parseFaultyMethod: source )
         doSemanticAnalysisIn: theClass ].
TypingVisitor new visitNode: ast
```
Solution: TypingVisitor

AST has not enough information
self, super, class of literals, Globals and
direct assignments to temps.
Solution: type check

receiverClass

node isMessage ifFalse: [ ^nil ].

^node receiver propertyAt: #type ifAbsent: [ nil ].
Problem: model

Code is very hard to understand and change. And the implementation behind the model itself is unnecessarily complicated.
Solution: model

- Type annotated AST
- CompletionProducer for suggesting completion options based on node type
Solution: finding nodes

```
nodeForOffset: anInteger
| children |
"choosing the best node on the specific offset"
"when we are on a leaf, we take the leaf node"
(children isEmpty) ifTrue: [
  (self sourceInterval includes: anInteger)
  ifTrue: [\^self]].
"if the node has children then we check the children"
children do: [:each |
  (each sourceInterval includes: anInteger)
  ifTrue: [\^each nodeForOffset: anInteger]].
```
Solution: model results

- Using the AST simplifies the code a lot
- It is faster (no Benchmarks yet)
Problem: sorter

It was very difficult to implement a sorting strategy as there was no separate implementation of sorting.
Solution: sorter

- you can choose the sorting strategy you want in the settings (alphabetic by default)
- sorting strategies based on n-gram and OCompletion will be added later
Refactoring results

# of classes 43 vs 22
# of methods 485 vs 243
# lines of code 3369 vs 1383
More improvements

- added completion for symbols
- fixed AST implementation
Fixing AST bugs

- Incorrect stop in RBSequenceNode
- Incorrect start in ParseErrorNode
- Not recognising missing closing ‘|’ in temp declaration as incorrect syntax
Future work

- ML based sorting strategy
- completing with syntax errors
- going beyond selector completion
Thanks!

@myroslavarm  romaniuk@ucu.edu.ua

@marcusdenker  marcus.denker@inria.fr