Effective autocomplete

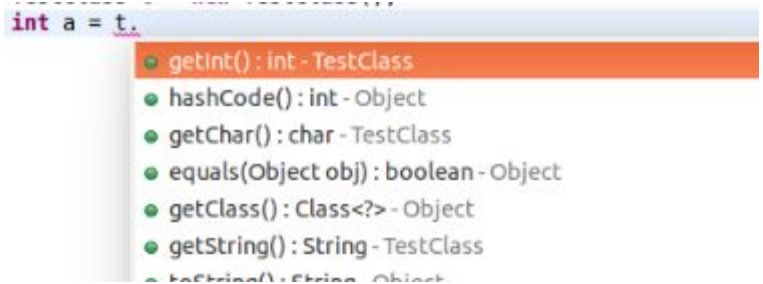
Patrik Ackland

The problem

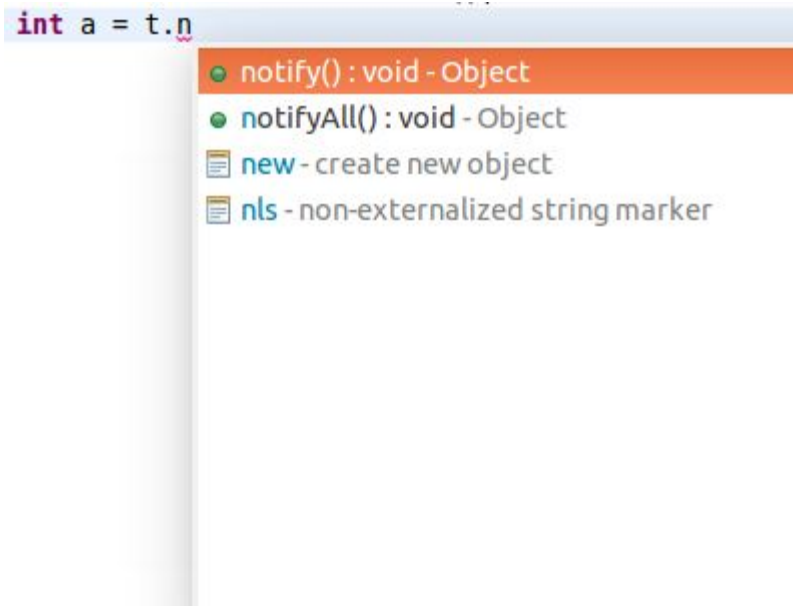
- Autocomplete today is very simple
- Gives method name based on return type/letters typed

Examples

Return type



Name



Problem

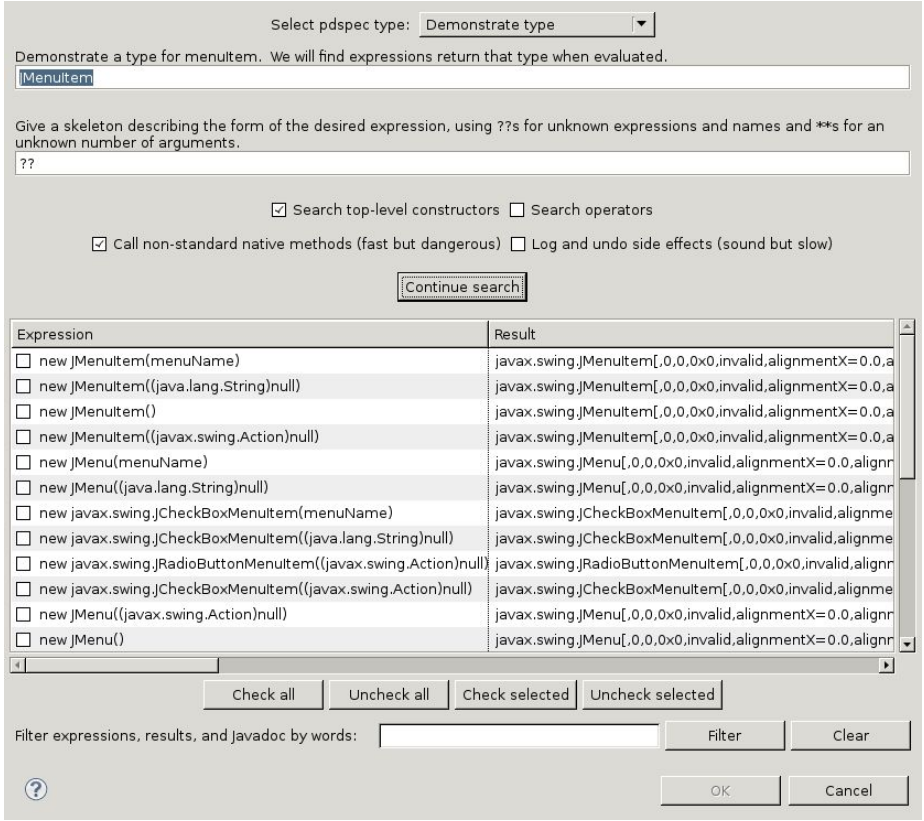
- Works for simple cases of depth 1 for right-hand side.
  - int a = t.getInt(); Didn’t save a lot of typing
- Not effective for more complex statements.
  - int a = object.getAnotherObject[0].getInt(); Would save more typing
- Not dynamic
- Does not use the environment or history.
  - How have statements like this looked in the past?
  - int age = numberOfShoes(); Correct but probably wrong.

Related work - CodeHint (2014)

- Dynamic analysis
- User asks for return type and describes expression
- Several iterations to find type. (99% of analyzed code has depth of at most 4)
- Suggestion based on analysis of 10m lines of code
- Probabilistic model
- Probability of accessing member m of type T

$$P(m|T)P(T) = \frac{\#\text{ accesses of } m \text{ on } T}{\#\text{ of accesses on } T} \times \frac{\#\text{ of accesses on } T}{\#\text{ of accesses}}.$$

Related work - CodeHint



Ideas for project

- Autocomplete more than just right-hand side of expression
  - Several lines
  - Function
- More accurate/faster suggestion with different model
  - machine learning
  - natural language processing
- Require less input from user
  - Currently: x’.toString().contains(“Eve”);

More autocomplete

Automatically generate for loop from this statement:

```
public static foo(int[] V) {
    int sumOfEvenIndices=0;
    for (
```

Or generate several lines based on user specification

New model

- Use NLP or ML to improve model.
- Use more than method and type (CodeHint).
- Can words that are used be of any help to improve suggestions?
- Probabilistic model based on analyzed code. What if we are writing new code?
- int age = person.getNumberOfShoes().

Require less from user

- User specifications for CodeHint
  - Improves accuracy
  - Requires more from user
  - Tricks like toString