

Effective autocomplete

Patrik Ackland

The problem

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

Examples

Return type

```
int a = t.
```

- `getInt() : int - TestClass`
- `hashCode() : int - Object`
- `getChar() : char - TestClass`
- `equals(Object obj) : boolean - Object`
- `getClass() : Class<?> - Object`
- `getString() : String - TestClass`
- `toString() : String - Object`

Name

```
int a = t.n
```

- `notify() : void - Object`
- `notifyAll() : void - Object`
- 📖 `new` - create new object
- 📖 `nls` - non-externalized string marker

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

Select pdspec type: Demonstrate type

Demonstrate a type for JMenuItem. We will find expressions return that type when evaluated.

JMenuItem

Give a skeleton describing the form of the desired expression, using ??s for unknown expressions and names and **s for an unknown number of arguments.

??

Search top-level constructors Search operators

Call non-standard native methods (fast but dangerous) Log and undo side effects (sound but slow)

Continue search

Expression	Result
<input type="checkbox"/> new JMenuItem(menuName)	javax.swing.JMenuItem[,0,0,0x0,invalid,alignmentX=0.0,a
<input type="checkbox"/> new JMenuItem((java.lang.String)null)	javax.swing.JMenuItem[,0,0,0x0,invalid,alignmentX=0.0,a
<input type="checkbox"/> new JMenuItem()	javax.swing.JMenuItem[,0,0,0x0,invalid,alignmentX=0.0,a
<input type="checkbox"/> new JMenuItem((javax.swing.Action)null)	javax.swing.JMenuItem[,0,0,0x0,invalid,alignmentX=0.0,a
<input type="checkbox"/> new JMenu(menuName)	javax.swing.JMenu[,0,0,0x0,invalid,alignmentX=0.0,alignr
<input type="checkbox"/> new JMenu((java.lang.String)null)	javax.swing.JMenu[,0,0,0x0,invalid,alignmentX=0.0,alignr
<input type="checkbox"/> new javax.swing.JCheckBoxMenuItem(menuName)	javax.swing.JCheckBoxMenuItem[,0,0,0x0,invalid,alignme
<input type="checkbox"/> new javax.swing.JCheckBoxMenuItem((java.lang.String)null)	javax.swing.JCheckBoxMenuItem[,0,0,0x0,invalid,alignme
<input type="checkbox"/> new javax.swing.JRadioButtonMenuItem((javax.swing.Action)null)	javax.swing.JRadioButtonMenuItem[,0,0,0x0,invalid,alignr
<input type="checkbox"/> new javax.swing.JCheckBoxMenuItem((javax.swing.Action)null)	javax.swing.JCheckBoxMenuItem[,0,0,0x0,invalid,alignme
<input type="checkbox"/> new JMenu((javax.swing.Action)null)	javax.swing.JMenu[,0,0,0x0,invalid,alignmentX=0.0,alignr
<input type="checkbox"/> new JMenu()	javax.swing.JMenu[,0,0,0x0,invalid,alignmentX=0.0,alignr

Check all Uncheck all Check selected Uncheck selected

Filter expressions, results, and javadoc by words: Filter Clear

? OK Cancel

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