CS409: Software Architecture and Design

Detecting Code Smells

Assignment 1

Damyan Kalev
10-22-2018

Chosen problems

Long Parameter List

A method with more than five parameters.

Long Method

A method containing more than 10 statements.

Long Class

A class that contains more than 100 statements.

Data Class

A data class is a class that contains only fields and getters/setters. Might contain toString() as well. It does not contain any additional functionality and cannot independently operate on the data it holds.

Message Chains

The message chain happens when one class is highly coupled to other classes in chain-like delegations. For example, lets say we have a class A, that needs to retrieve data from class E. If A needs to retrieve first other objects, in this case B, C and D, there is a message chain.

a.getB().getC().getD().getE().getTheData();

The problem is that A is unnecessary coupled to all the intermediate classes.

Man in the Middle

A class that delegates most of its work to other classes. It does not really contribute and does simple delegation instead.

Solution

Long Parameter List

Simply get the parameters for each MethodDeclaration and check if more than five.

Long Method

This one, proved to be quite tricky as counting all the statements in a method might not give reasonable results. For example, for loop “for (int i = 0, i < n, i++) is counted as 3 separate statements, or if else branches are represented as children of the previous if node in the AST. Therefore, for the solution needed to define different visitors for most of the different statements: IfStmt, ForStmt, WhileStmt etc.

Long Class

The implementation is quite straightforward, as it uses the logic of LongMethod to get the length of the different methods and simply adds to that the constructor length and the number of fields.

Data Class

Most data classes have fields to store the data, getters, setters and they might include utility method such as toString(). I first check the number of fields (should have at least one). Then check if all methods are either getters, setters or toString. I have defined getters as public, not void and starting with “get” or “is” in their name. The last condition relies on good practice and can be improved by resolving the

return type. Setters are defined similarly as public, void type, with no more than 1 parameter and starting with “set” in their name.

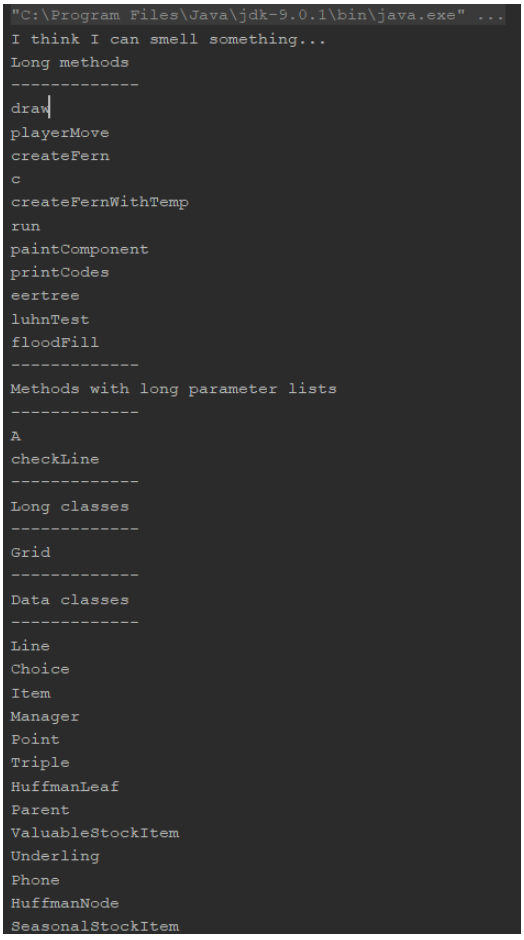
Message Chains

To detect a message chain (the solution is looking for chains with size larger than 3), I visit MethodCallExpr and check if both the parent and one of its children is MethodCallExpr as well. Then I check for both parent and child if their type is NOT void and they do not have the same name. I use the SymbolResolver to find the types of the nodes.

Man in the Middle

I first extract all variables from fields that are not primitives (ClassOrInterfaceType). Then find the total number of method calls and the number of method calls in return statements that contain one of the variables. The idea behind that is that most methods that delegate follow a similar pattern – “return variable.method()”. Then I check if the ratio of number of method calls following the pattern over number of total method calls is above certain threshold (currently 0.8) and if the number of variables is 1.

Results and Evaluation



Long Parameter List

Detects both A in ManOrBoy and checkLine in Grid. No false positives.

Long Method

Detects all the mentioned methods in the instructions (createFern, floodFil, and the methods in Grid). Have not found any false positives.

Long Class

The only long class it finds is Grid. No false positives and have not found any other long classes manually.

Data Class

Detects all the specified data classes and some more. Again, have not found any false positives.

Message Chains

Detects all mentioned in instructions (CipollasAlgorithm, Munchausen, NBodySim). Detects also Test, which might be a false positive.

Man in the Middle

Detects the example – AccountManager. Detects one more class – Phone. I am not sure if this is a false positive, as Phone does delegate all its work to java.lang.String.

```
Customer
Node
-------------
Message chains
-------------
Munchausen
CipollasAlgorithm
NBodySim
Test
-------------
Man in the middle
-------------
AccountManager
Phone

Process finished with exit code 0
```

Statement of Score 20/20

Most of the code smells are quite tricky to detect, especially with conditional logic. However, I believe I gained a good understanding of the core principles of the framework and some of my implementations use more advanced techniques, such as SymbolSolver. The code smell detectors perform well on the test set.