Kevin Nguyen, Jacqueline Wong - Team 4

CS 6301 - Software Comprehension and Analysis

**Common Coupling using Class-Level and Method-Level Granularity**

**Intermediary Project Update**

**Project Objective (Reiteration from Project Proposal)**: To create a common coupling detection tool using Eclipse ASTVisitor for Java open source code bases.

**Current Progress**

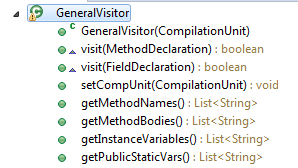
Currently, we have written a functioning program which detects common coupling. More details regarding the program and what exactly was coded are in the following section. We ran this program against Freemind and CMS Java as our preliminary results.

**Program Details**

We coded our program in Java and utilized Eclipse ASTVisitor in the program. The program currently consists of two files: GeneralVisitor.java and Main.java

GeneralVisitor.java is where we extend ASTVisitor to retrieve the code pieces of interest. For file-level granularity, we retrieved public static instance variables and method bodies. Public static instance variables will be accessible to external classes besides the one the instance variable was instantiated in. Method bodies will be used to identify where the public static variables are being used. For method-level granularity, we will need all instance variables (Not just public static ones) and the method names and bodies those instance variables were used in.

Based on this information, here are the methods we created for GeneralVisitor.java:



We overrided the ASTVisitor methods which visit MethodDeclaration and FieldDeclaration, and on each visit, we retrieve the information required from the node and store the information in List<> data structures. We have also created getter methods to get these data structures in Main.java.

Main.java is where the logic of the common coupling detection tool is. For our preliminary program, we have implemented base versions of both file-level and method-level granularity.

For file-level granularity, we looped through the list of all the public static variables identified in GeneralVisitor.java. We then also loop through all the method bodies and check if there are any occurrences of the public static variable in the method body. If there is an occurrence, we add the name of this file to a List<>. After all this, we print a list of all global variables which had 2+ external files using the global variable.

For method-level granularity, we followed the same format, except instead of public static variables, we used all instance variables to compare against the method bodies. If a method body had an occurrence of an instance variable, we added the method name to the List<> to be printed after.

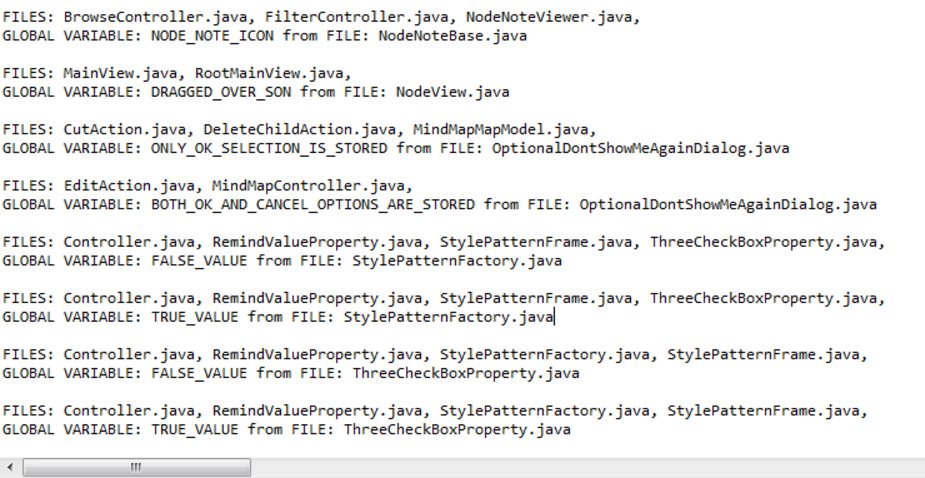
The full code of this version of the program described in the report can be found in the Github repository (<https://github.com/kntnguyen94/cs6301-sw-analysis/tree/master/CommonCoupling_CS6301_Team4/src>). We ran this program against Freemind and CMS Java to produce preliminary results. Sample of these preliminary results can be found in the next section.

**Milestones/ Preliminary Results**

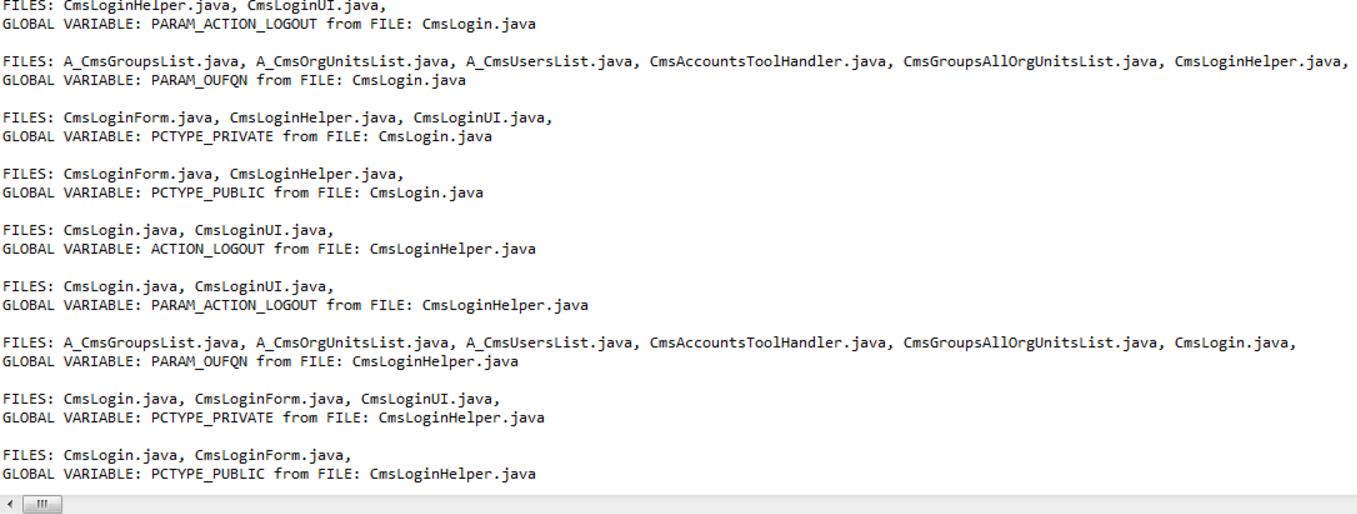
These results are only snippets of the full preliminary results. Full preliminary results can be found at the following links:

* Freemind (File-Method Granularity) - <http://pastebin.com/Mtec5PnQ>
* CMS (File-Method Granularity) -
* Freemind (Method-Level Granularity) - Results are cut off due to console output max length <https://drive.google.com/open?id=0B0HfBS_aeJTbV1M0T3M5OV9fTk0>
* CMS (Method-Level Granularity) - Results are cut off due to console output max length <https://drive.google.com/open?id=0B0HfBS_aeJTbZklJZ21tUlhOQ0k>

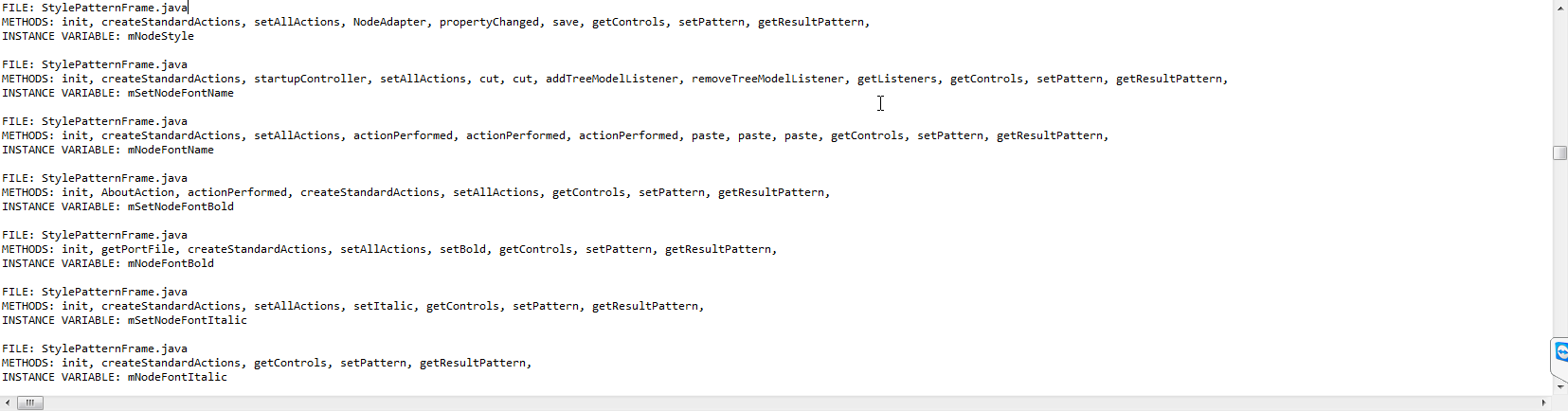
FreeMind Output (File-Level Granularity)



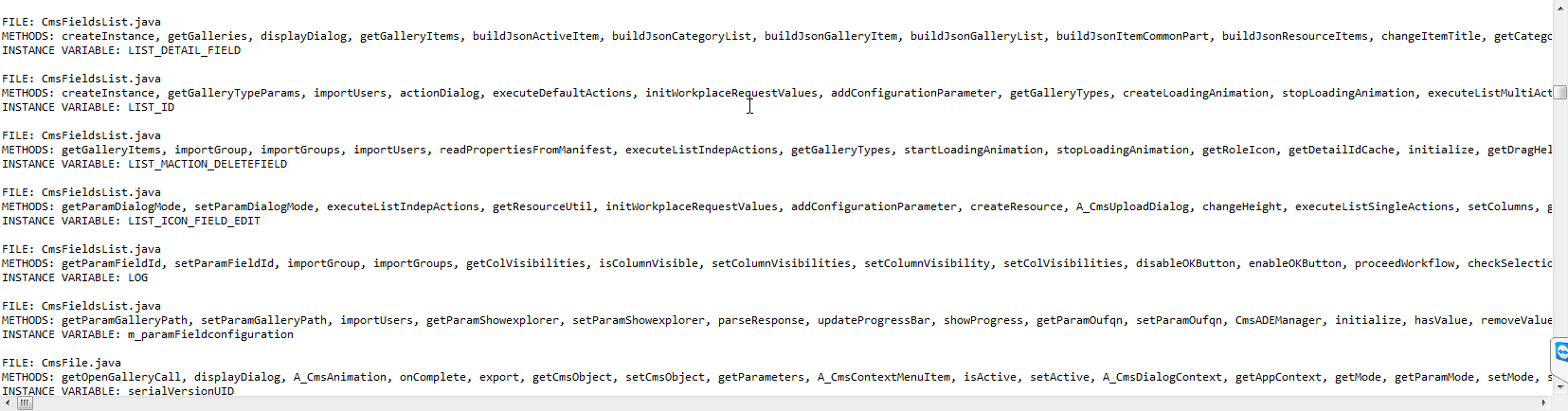
CMS Output (File-Level Granularity)



Freemind Output (Method-Level Granularity)



CMS Output (Method-Level Granularity)



**To-Do List**

The current implementation of the common coupling tool is not the final iteration of the tool. There are still other factors we need to consider to identify all common coupling cases. For example, protected static variables can still create instances of common coupling, yet we have not addressed this in our current iteration of the tool. Furthermore, we only look at whether the variables are used in method bodies, but not in instance variable declarations.

For the final project, we will take a look at other cases where common coupling might occur and add these cases into the final iteration of the code. We will also refactor and reorganize the code as well to make it more polished.

We will also be choosing several other systems to test this tool on beyond just Freemind and CMS. For all these systems, a gold set wil be created to compare our outputted results to. This gold set will be created by manually going through code files and identifying global data and which modules use them.

**Remaining Schedule**

|  |  |
| --- | --- |
| **Date** | **Deliverable List** |
| 4/14 | Intermediary Project Report Due. ~~Preliminary results for 1-3 systems.~~ |
| 4/21 | Complete final code revisions and finalize presentation-ready results. |
| 4/26-4/28 | Final Project Presentation Date. |
| 5/3 | Final Project Report Due. |