Project 11

February 13th, 2019
CS461 - Skrien
Jackie Hang and Kyle Slager

Overview:

The main focus of this project was fixing our parser from the last project so that we could have a working version going forward. As the overall goal of this class is to create a fully functioning compiler, it is very necessary that our parser operates perfectly so that this goal can be reached. Both of us were in the same group for the previous parser project, so we only had one codebase to draw from. Unfortunately, our project had more errors than seemed reasonable to fix in the given amount of time, so we requested a new codebase with only one broken method, the parsePrimary method. In addition to making changes to the parsePrimary method, we were also asked to create three subclasses of the Visitor class, each with a fairly different direction. It is not necessarily clear as of now what the purpose of creating these Visitor subclasses is beyond furthering our understanding of the visitor pattern.

Elegance:

When approaching fixing the parsePrimary method, since there was a Primary and a Suffix division, we decided to split the method into two for greater elegance. As for the Visitor subclasses, we tried to make them as efficient as possible. The first focus was on overriding as few methods as possible in each subclass. We chose to solely override the methods that we had to gain information from in each subclass. In the MainMainVisitor we had to check the class and the method names so we needed to override both. In the StringConstantsVisitor we only needed to check the ConstStringExprs which meant only overriding that visit method. In the NumLocalVarsVisitor we needed to get a class name, method name, and check all of the DeclStmts to determine how many there were in each method. This meant overriding all three of those visit methods. We chose to add the number of variables from each method and the method names to array lists and then in the class visit, we concatenate the class name onto the method names. In the getNumLocalVars method we add all of these things into the HashMap that we return.

Inelegance:

As far as inelegances go, the main one might be that we have no ToolBarController as of now. Our group removed it at one point last semester (we think project 9), and we felt that it might not be necessary to add it back in for this project, as the main focus was on the parsePrimary and Visitor subclasses. Additionally, there was some minor code duplication in the parsePrimary, but we thought that separating that small bit of code into a separate method might affect the readability of the code, so we refrained from making a change to that. The last inelegance was that in the NumLocalVarsVisitor's Class_ visit method, we used a substring tacked onto the front of the method names to determine whether or not the method already had a class name added to it. This string was "%%%%," which we determined would have no adverse effects on the program as a class or method could never include a series of percentage signs. We still felt that this may constitute inelegance as it may seem like a brute force solution to the problem.

Additionally, we have to give a special thanks to Zeb for helping us with some of our parsePrimary.

Division of Work:

Jackie focused mainly on working on the parsePrimary method, while Kyle primarily worked on the new Visitor subclasses. Jackie made more of the changes to update the IDE. Kyle did more of the writeup. The overall division of labor was probably pretty close to even.