Code selected

For our code review we chose to examine the Interpreter class, we did this because it is among the most complicated components of the project. Ryan Park did almost all the implementation because the rest of the group was unsure of how to implement the class and had agreed to implement other classes. This makes it a prime candidate for our code review because since the class has been implemented our group can go through the code and voice any questions we have regarding the implementation. If the code has been properly formatted and uses proper conventions we should have little to no questions about the implementation.

Good points

The class was written very well in many regards, the most distinguishing feature was the proper formatting of the code and how it followed the designated coding convention. Another key distinction was that all important fields and methods had a accompanying JavaDoc that explained the field or method concisely within one sentence. By doing so the code is very easy to understand as there is no needless filler that convolutes and snowballs out of control in the class. Finally all the variables and functions were given short names that perfectly described what their purpose was and what it did. With both the proper naming and JavaDocs the group had very few questions even though the actual class was quite large and foreign to most of the group. Overall the Interpreter class was implemented very well with proper conventions followed at all times, which resulted in a easy to read and understand implementation for one of the complex components of the project.

Bad points

There were only 2 aspects of the class we felt that could be considered undesirable in the implementation. The first being that there was a massive switch case that was responsible for the forth interpreter. This has roughly 40 cases and is slightly unwieldy to read but nothing can be done as it is not a formatting error rather the only means to implement it. The second negative aspect was the class could be optimized in a few key points such as it performs 2 passes which could be reworked into a more efficient 1 pass implementation. Some of the data structures could also be reworked such as using a regular map instead of a bidirectional map. Overall there was very little negative points because the few things that were there did not effect the project’s runtime adversely.

Overall group opinion/discussion

The group was extremely pleased with how well the class was implemented and how it was broken down into such small well documented sub problems that the interpreter which was once a mysterious and unknown part of the project could almost be read as a book with how smoothly everything was explained.

What we would change.

As stated in the previous section the group found very little wrong with how the class had been implemented and one of the two primary things (the forth interpreter) could not really be changed as it relies on checking which of the 40 possible actions it should take. The class could be optimized slightly so that it uses a onepass system and changing some of the data structures to less resource demanding variants. Ultimately both of these could be altered but this would be done if we have enough time to do so as it would be a considerable undertaking with very little to gain from.