Brian Forsberg

Java Candidate Experience Report

The most difficult part of this project was coming up with a way to recursively parse the functions and function arguments. After several hours of brainstorming and research, I decided I had two options: come up with my own recursive descent parser, or use a framework such as ANTLR to do it for me. Either way, the process would have to look something like this:

ANTLR would be able to create a parse tree from the input, and from there I could go through the parse tree and create nodes for the AST. The problem with ANTLR is that a decent amount of time is required to learn how to use it. Creating my own recursive descent parser would also require a decent amount of time simply because it is difficult.

Once the AST is created, evaluating the result would be relatively simple. Each type of expression would have its own evaluate method, and that method would be called on each of the expression's arguments. This would continue until the evaluator reaches a constant value, at which point the results would bubble up.

The let expression is a little trickier. To implement it, I think I would have to pass each LetExp body an environment containing variable information. Nested let expressions would require nested environments.

Overall, I spent a small portion of my time writing code. Most of my time was spent mapping out a solution on paper. I figured that it would have been useless to try to implement a solution that I didn't understand. The small part of code that I did complete was the tokenizer. Had I been given more time, I would have liked to continue with ANTLR, incorporate a logging layer, and set up a maven build.