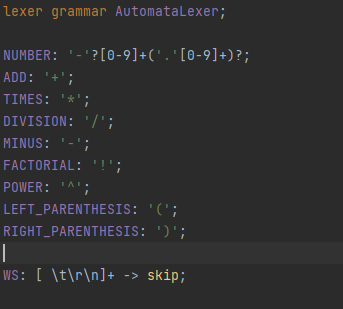
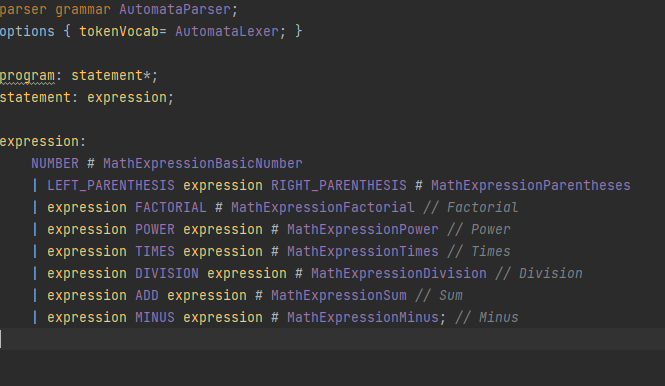
# Grammar

## Lexicon

We first set up our lexicon so that we can create our parser rules. We wanted to make sure that we could support decimal numbers and negative numbers. 

We also added the required math symbols to create equations.

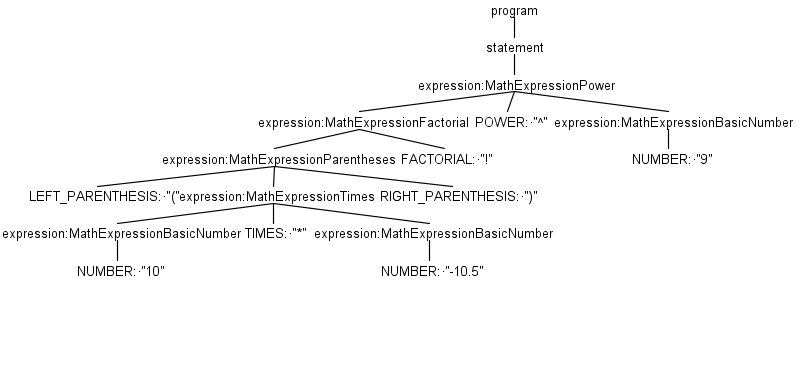
## Parser rules

After creating the lexicon we started working on the parser rules. We decided to first make a generic program rule. We did this so that we could easily add more types of statements later. 

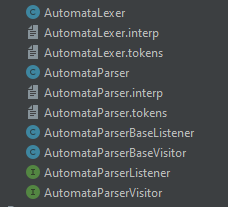
For the assignment of week 1, we only created one type of statement the expression. Here we added all the required mathematical operations. We also decided to add the parenthesis. So we can write slightly more complex mathematical equations.

The parse tree that you would get for an expression is this:

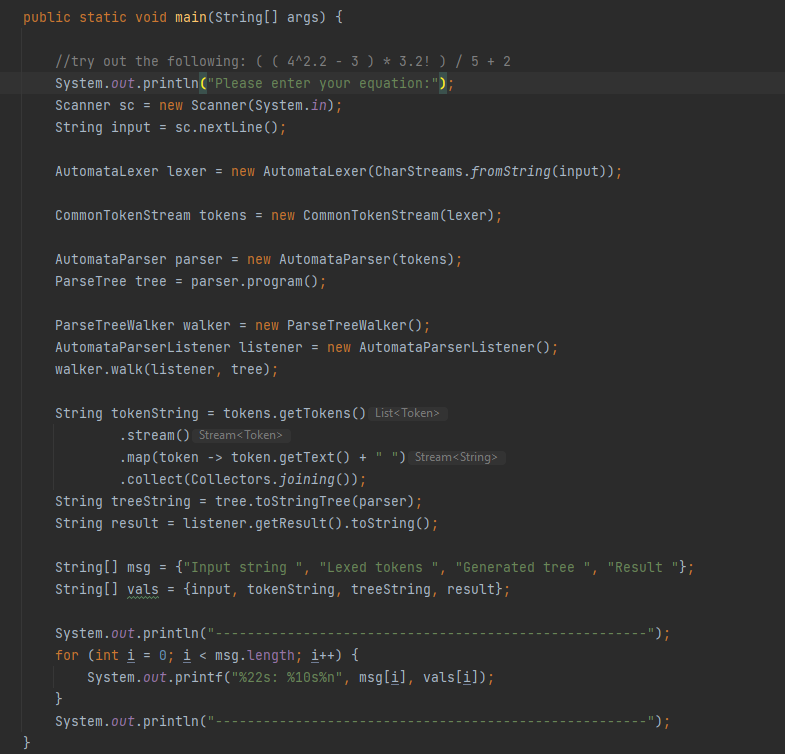
(10 \* -10.5)! ^ 9 + 1 / 8 – 2

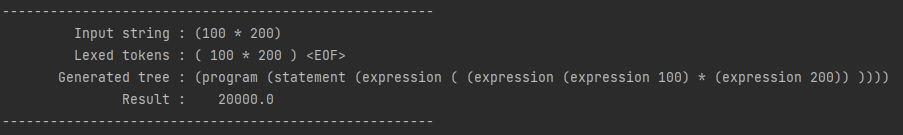
Result in this parse tree:

The java files that we generated where:



# Program

We created a program where you can fill in your equation using the console. 

The output is the answer to your equation. The program also specifies the input tokens and the values in the parse tree.

# Listener

To compute the formula we used a stack. The listener traverses the tree in postfix order. With each step, we add the result into our stack. Eventually after processing each step. You get the computed result inside of the stack. The code that we wrote looks like this::

