Report

Name: Jin Zhenming (김진명)

Student ID: 2022315690

Java version: 22

This is the report for my program in PA1.

1. Expr.g4

I added "assignExpr" and "identifierExpr" to let the parser support the assignment expression and the identifiers. To make the process of the identifiers easier. I defined "id" which represents the identifier in the expression.

I added "[-]?" (1 or 0 '-') to make the INT rule in lexer rules able to handle negative numbers. Similarly, I used "[-+]?" to handle +/- signs in the real numbers.

2. Evaluate.java

A file that I use to calculate the result of each node in the AST recursively.

Each node will be calculated in different ways according to their type.

3. AstNodes.java

I defined AstNode as an abstract class, and I used polymorphism to express different nodes to build the AST, also doing the calculation of each node.

NormCalcNode stands for "Normal Calculation Node" which corresponds with the "infixExpr" defined in the grammar file (For instance, "Node + Node").

NumNode stands for "Number Node" which corresponds with the "numberExpr" defined in the grammar file (For instance, 114514, -12, +1919.81, and more).

AsgnNode stands for "Assign Node" which corresponds with the "assignExpr" defined in the grammar file (For instance, AcFun = 20070606).

IdtNode stands for "Identifier Node" which corresponds with the "identifierExpr" in the grammar file.

All these nodes have a function that will call the method in Evaluate.java to calculate their results.

4. BuildAstVisitor.java

This file overrides methods in ExprBaseVisitor.java. The methods in it return different types of AstNode based on the method overridden.

5. AstCall.java

The method called "Call" in the class AstCall will output the AST. The output process will do recursively. Each type of node has its output method.

NormCalcNode: Print the operator and recursively output two expressions on its left and right.

NumNode: Print the value stored in the NumNode object.

AsgnNode: Print the operator and output the identifier and its value recursively.

IdtNode: Print the name of the identifier stored in this object.

6. program.java

The main program of the calculator.

For loop is used to access the root node of each AST.

I use loops to deal with the input with multiple commands and it prints the AST of each command separately.

Similarly, I use a loop to calculate the result of each command separately. Since we need something to store and access all the identifiers and their value, therefore a hashmap is used as memory in the calculation method and the main program.