**Programming Assignment #1 - 2017311178 박준영**

1. 과제 수행 환경

* Linux Ubuntu 20.04

1. Grammar Setting (추가 된 요소)

* Lexer : 4개의 Math 명령어를 표현하기 위해 정의
* MIN: 'min'
* MAX: 'max'
* POW: 'pow'
* SQRT: 'sqrt'
* Parser : Assign과 그 외의 4개를 표현. 나중에 확장성을 위해 함수의 매개변수는 num으로 제한하지 않고 expr으로 선정.
* ID '=' expr #assignExpr
* MIN '(' expr ',' expr ')' #minExpr
* MAX '(' expr ',' expr ')' #maxExpr
* POW '(' expr ',' expr ')' #powExpr
* SQRT '(' expr ')' #sqrtExpr

1. Code Review

* AstNodes.java : 간단한 Binary Tree의 모습으로 구현
* AstNodes left; // 왼쪽 child Node
* AstNodes right; // 오른쪽 child Node
* String op; // operation or number
* BuildAstVisitor.java : parser tree에 따라 해당하는 AstNodes로 저장해 Tree를 구현한다.
* Method 소개
* void CheckRoot(AstNodes node) : root node가 없을 시 해당 node를 root로 설정한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

* AstNodes visitParensExpr(ExprParser.ParensExprContext ctx) : ctx.children = [‘(‘, expr, ‘)’] 임을 이용해 index 1의 expr에 visit한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

* AstNodes visitInfixExpr(ExprParser.InfixExprContext ctx) : 사칙연산을 처리하기 위한 method로 ctx.children = [expr, operation, expr] 임을 이용해 index 1의 operation을 switch 문을 이용해 ADD, SUB, MUL, DIV로 구분하여 Node에 저장한다. 이때 operation의 왼쪽과 오른쪽 expr 각각 visit하여 Node의 left, right에 저장한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

* AstNodes visitMinExpr(ExprParser.MinExprContext ctx) : min(2,3)와 같은 식을 처리하기 위함으로 ctx.children = [MIN, ‘(‘ , expr, ‘,’ , expr, ‘)’ ] 으로 이루어져 있으므로 Node의 opeartion은 “min”으로 저장, left, right는 각각 index 2, 4를 visit한다. MaxExpr, PowExpr도 마찬가지로 구현한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

* AstNodes visitSqrtExpr(ExprParser.SqrtExprContext ctx) : sqrt(4)처럼 앞에서 min과는 달리 매개변수가 하나이므로 index 2만 left에 visit하고 right는 비운다.

텍스트이(가) 표시된 사진

자동 생성된 설명

* AstNodes visitNumberExpr(ExprParser.NumberExprContext ctx) : leaf node의 경우 해당하는 숫자를 op에 저장해준다. 이때, 경로는 ctx.chidren.get(0).getChild(0)을 통해 접근 가능하다.

텍스트이(가) 표시된 사진

자동 생성된 설명

* AstNodes visitAssignExpr(ExprParser.AssignExprContext ctx) : ctx.children = [ ID, ‘=’, expr]으로 op는 Assign으로 저장, left는 ID의 text를 op에 저장한 새로운 node로, right는 index 2의 expr을 visit하여 저장한다.

텍스트이(가) 표시된 사진

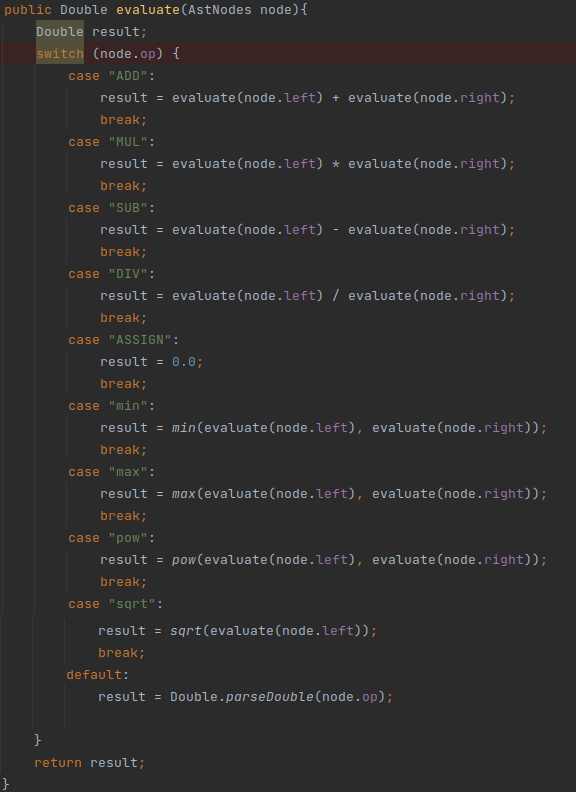
자동 생성된 설명

* AstCall.java : Build 한 AST를 이용해 tree의 내용을 출력한다.
* call method : DFS기반 알고리즘을 이용해 node를 출력한다. 이때 depth에 따른 차이를 두기 위해 매개변수 depth를 추가하였다.

텍스트이(가) 표시된 사진

자동 생성된 설명

* Evaulate.java : 마찬가지로 앞에서 build한 AST를 통해 입력 받은 문자열을 처리한다.
* Root부터 node를 탐색하는데 각 node의 op에 따라 알맞은 연산을 해주어 return한다. 만약 해당하는 op가 없을 경우 leaf node, 즉 숫자 Double로 형변환해주어 return한다.



* program.java : main 함수가 있는 실행파일
* BuildAstVisitor, AstCall, Evaluate을 호출하여 원하는 결과를 출력한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

1. Result

텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명