TODO:

* Scanner (DONE ? )
* AST (DONE?)
* Parser (DONE?)
* TypeChecker
* Interpreter

**program -> classdefs declarations fundefs instructions**

declarations -> declarations declaration

declarations -> <empty>

declaration -> TYPE inits ;

**declaration -> ID classinits ;**

declaration -> error ;

inits -> inits , init

inits -> init

init -> ID = expression

**classinits -> classinits , classinit**

**classinits -> classinit**

**classinit -> ID**

instructions -> instructions instruction

instructions -> instruction

instruction -> print\_instr

instruction -> labeled\_instr

instruction -> assignment

instruction -> choice\_instr

instruction -> while\_instr

instruction -> repeat\_instr

instruction -> return\_instr

instruction -> break\_instr

instruction -> continue\_instr

instruction -> compound\_instr

print\_instr -> PRINT expression ;

print\_instr -> PRINT error ;

labeled\_instr -> ID : instruction

**assignment -> access = expression ;**

**access -> ID**

**access -> ID.ID**

choice\_instr -> IF ( condition ) instruction

choice\_instr -> IF ( condition ) instruction ELSE instruction

choice\_instr -> IF ( error ) instruction

choice\_instr -> IF ( error ) instruction ELSE instruction

while\_instr -> WHILE ( condition ) instruction

while\_instr -> WHILE ( error ) instruction

repeat\_instr -> REPEAT instructions UNTIL condition ;

return\_instr -> RETURN expression ;

continue\_instr -> CONTINUE ;

break\_instr -> BREAK ;

compound\_instr -> { declarations instructions }

condition -> expression

const -> INTEGER

const -> FLOAT

const -> STRING

expression -> const

**expression -> access**

expression -> expression + expression

expression -> expression - expression

expression -> expression \* expression

expression -> expression / expression

expression -> expression % expression

expression -> expression | expression

expression -> expression & expression

expression -> expression ^ expression

expression -> expression AND expression

expression -> expression OR expression

expression -> expression SHL expression

expression -> expression SHR expression

expression -> expression EQ expression

expression -> expression NEQ expression

expression -> expression > expression

expression -> expression < expression

expression -> expression LE expression

expression -> expression GE expression

expression -> ( expression )

expression -> ( error )

**expression -> access ( expr\_list\_or\_empty )**

**expression -> access ( error )**

expr\_list\_or\_empty -> expr\_list

expr\_list\_or\_empty -> <empty>

expr\_list -> expr\_list , expression

expr\_list -> expression

fundefs -> fundef fundefs

fundefs -> <empty>

fundef -> TYPE ID ( args\_list\_or\_empty ) compound\_instr

**fundef -> ID ID ( args\_list\_or\_empty ) compound\_instr**

args\_list\_or\_empty -> args\_list

args\_list\_or\_empty -> <empty>

args\_list -> args\_list , arg

args\_list -> arg

arg -> TYPE ID

**arg -> ID ID**

**classdefs -> classdef classdefs**

**classdefs -> <empty>**

**classdef -> accessmodificator CLASS ID classcontent**

**classdef -> accessmodificator CLASS ID EXTENDS ID classcontent**

**classcontent -> { fieldefs methoddefs }**

**fielddefs -> fielddefs fielddef**

**fielddefs -> <empty>**

**fielddef -> accessmodificator declaration**

**methoddefs -> methoddefs methoddef**

**methoddefs -> <empty>**

**methoddef -> accessmodificator fundef**

**accessmodificator -> PRIVATE**

**accessmodificator -> PROTECTED**

**accessmodificator -> PUBLIC**