Milestone 1

Team 1

**Github Link:** [**https://github.com/amitmaharana/SER502-Spring2020-Team1**](https://github.com/amitmaharana/SER502-Spring2020-Team1)

**SEA Programming Language**

**ANTLR4**

We have used Java with ANTLR4 API to develop our language. ANTLR (Another Tool for Language Recognition) uses grammar to generate lexer and parser which can build and traverse parse trees. ANTLR is a powerful parser generator for reading, processing, executing, or translating structured text or binary files. It's widely used to build languages, tools, and frameworks. It is useful any time you need to evaluate expressions unknown at compile-time [1].

ANTLR is code generator. It takes so called grammar file as input and generates two classes: lexer and parser. Lexer runs first and splits input into pieces called tokens. Each token represents meaningful piece of input. The stream of tokens is passed to parser which does all necessary work. It is the parser that builds abstract syntax tree, interprets the code or translate it into some other form [2].

**Design**

A program in SEA Language consists of a block, a block may have multiple declarations or commands. Our language supports the following primitive data types - boolean, integer, string and supports various of operations on them.

A declaration is basically an assignment of a value or expression to a variable. The command which are supported by our language includes conditional structures like, if then else and ternary conditions and loops like while, for and for range loop. Commands on also includes expressions and statements.

Statements is a one-line instruction to display a variable. conditional blocks contain a condition terminal, which basically gives true or false value doing various operations. Expressions contain mathematical operations.

**Grammar**

grammar SEALang;

/\*\* Starting of our program.\*/

program : block;

/\*\* List of either declaration or commands.\*/

block : (declaration | command)+ ;

/\*\* declaration: User can declare Int, Boolean, String, and expressions.\*/

declaration : TYPE VAR (ASSIGN INT |

ASSIGN BOOLEAN |

ASSIGN STRING |

ASSIGN expression)? SEMICOLON ;

/\*\* condition: User can use NOT, nested conditions, comparators, and chaining of multiple conditions.\*/

condition : (NOT)? ((OPB condition CPB | BOOLEAN | expression comparator expression | VAR) condition\_chain);

condition\_chain: multi\_condition condition condition\_chain | ;

comparator : EQUAL | NOT\_EQUAL | LESSER\_THAN | GREATER\_THAN | LESSER\_THAN\_EQUAL | GREATER\_THAN\_EQUAL ;

multi\_condition : AND | OR ;

/\*\* command: User can use multiple and nested If-else, loops, assignment operator, and display data types\*/

command : (if\_block |

while\_block |

for\_block |

range\_block |

assign\_block |

show)+;

/\*\* if\_block: User can use either only if, if-else, if-elseif-else, or nested if-else\*/

if\_block :

IF OPB condition CPB

OCB

block

CCB

(ELSE IF OPB condition CPB

OCB

block

CCB)\*

(ELSE

OCB

block

CCB)? ;

/\*\* while\_block: User can use nested while loops with conditions and execute a block.\*/

while\_block :

WHILE OPB condition CPB

OCB

block

CCB ;

/\*\* for\_block: User can use nested for loops and execute a block.\*/

for\_block :

FOR OPB ((TYPE VAR ASSIGN INT)| (VAR ASSIGN INT) | ) SEMICOLON condition SEMICOLON ((VAR (INC | DEC | ASSIGN expression)) | ) CPB

OCB

block

CCB ;

/\*\* range\_block: User can use nested for range loops and execute a block.\*/

range\_block :

FOR VAR IN RANGE OPB (INT | VAR | expression) COMMA (INT | VAR | expression) CPB

OCB

block

CCB ;

/\*\* assign\_block: User can use this to assign expressions or boolean or strings to a variable.\*/

assign\_block : VAR ASSIGN (expression | BOOLEAN | STRING) SEMICOLON ;

/\*\* show: User can use this to display a variable.\*/

show : 'show' (VAR | INT | BOOLEAN | STRING) SEMICOLON;

/\*\* ternary\_block: User can use ternary operator and evaluate expressions.\*/

ternary\_block : condition QUESTION expression COLON expression ;

/\*\* expression: This will perform airthmatic operations on numbers or variables.

\*This will also evaluate ternary\_block, and nested expressions.

\*/

expression : term expression\_com;

expression\_com : (MINUS | PLUS) term expression\_com | ;

term : util term\_com ;

term\_com : (MULTIPLY | DIVIDE) util term\_com | ;

util : (VAR | INT | OPB ternary\_block CPB | OPB expression CPB) ;

TYPE : 'Int' | 'Boolean' | 'String' ;

PLUS : '+' ;

MINUS : '-' ;

MULTIPLY : '\*' ;

DIVIDE : '/' ;

ASSIGN : '=' ;

EQUAL : '==' ;

NOT : '!' ;

NOT\_EQUAL : '!=' ;

LESSER\_THAN : '<' ;

GREATER\_THAN : '>' ;

LESSER\_THAN\_EQUAL : '<=' ;

GREATER\_THAN\_EQUAL : '>=' ;

INC : '++' ;

DEC : '--' ;

AND : '&&' ;

OR : '||' ;

OPB : '(' ;

CPB : ')' ;

OCB : '{' ;

CCB : '}' ;

SEMICOLON : ';' ;

COLON : ':' ;

COMMA : ',' ;

QUESTION : '?' ;

IF : 'if' ;

ELSE : 'else' ;

WHILE : 'while' ;

FOR : 'for' ;

RANGE: 'range' ;

IN : 'in' ;

VAR : [a-z]+ ;

INT : [0-9]+ ;

STRING : '"' (~["\r\n] | '""')\* '"' ;

BOOLEAN : TRUE | FALSE ;

TRUE : 'True' ;

FALSE : 'False' ;

COMMENT : DOUBLE\_SLASH ~[\r\n]\* -> skip ;

DOUBLE\_SLASH : '//' ;

WS : [ \n\t\r]+ -> skip ;

**Examples**

**Example 1**

Int a = 3;

Int b = 5;

Boolean bo = False;

Int c;

c = ( (!(a > b) && (a == 5) || bo) ? a+3 : b+5) \* ((a+b)/(b-a));

show a;

A screenshot of a cell phone

Description automatically generated

**Example 2**

Int a = 3;

Int b = 5;

//for loop

for (Int i = 0; i < b; i++){

a = a + b;

}

show a;

A picture containing black

Description automatically generated