# SEA Programming Language

SER 502 – Emerging Languages and Programming Paradigms

https://github.com/amitmaharana/SER502-Spring2020-Team1

# What is SEA?

Simple Everyday Arithmetic Programming Language

# Why use SEA?

- Easy to use
- Fast to implement

# Features Supported

- Datatypes: Integer, Boolean, String
- Data structure: Arrays (zero-indexed)
- Operators: Arithmetic, Relational, Logical, Assignment
- Conditional Constructs: if-else and ternary operator
- Looping Structures: for, while, range
- String Operations: length, concat, equals, split, substring
- Array Operations: length, accessing array elements by index

# Contents

# Grammar

```
grammar SEALang;

/** Starting of our program.*/
program : block;

/** List of either declaration or commands.*/
block : (declaration | command)+ ;

/** declaration: User can declare Int, String*/
declaration : TYPE VAR SEMICOLON ;

/** 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)+;

/** expression: This will perform airthmatic operations on numbers or variables.
*This will also evaluate ternary_block, and nested expressions.
*/
expression: OPB expression CPB #parExpression
               | left = expression op = MULTIPLY right = expression #multiplyExpression
               | left = expression op = DIVIDE right = expression #divideExpression
               | left = expression op = PLUS right = expression #plusExpression
               | left = expression op = MINUS right = expression #minusExpression
               | INT #intExpression
               | VAR #variableExpression
               | VAR OSB (INT | VAR) CSB #intArrayExpression;

/*condition: User can use NOT, nested conditions, comparators, and chaining of multiple conditions*/
condition: OPB condition CPB#parCondition
               | NOT condition #notCondition
               | left = expression op = comparator right = expression #comparatorCondition
               | left = condition op = multi_condition right = condition #multiConditionCondition
               | left = string_expression op = EQUALS right = string_expression #equalsStringCondition
               | BOOLEAN #boolCondition
               | VAR #variableCondition
               | VAR OSB (INT | VAR) CSB #boolArrayCondition;

comparator : EQUAL | NOT_EQUAL | LESSER_THAN | GREATER_THAN | LESSER_THAN_EQUAL | GREATER_THAN_EQUAL ;
multi_condition : AND | OR;
```

# Grammar

```
/*
condition_block is for ifelse, looping, ternary statements.
*/
condition_block : OPB condition CPB | condition;


/** if_block: User can use either only if, if-else, if-elseif-else, or nested if-else*/
if_block :
    IF OPB condition_block CPB
            OCB
                block
            CCB
    (else_statement)? ;
else_statement: ELSE
            OCB
                block
            CCB;

/** while_block: User can use nested while loops with conditions and execute a block.*/
while_block :
    WHILE condition_block
    OCB
        block
    CCB ;

/** for_block: User can use nested for loops and execute a block.*/
for_block :
    FOR OPB for_assign SEMICOLON condition_block SEMICOLON for_updation CPB
    OCB
        block
    CCB ;
for_assign : ((TYPE VAR ASSIGN expression)| (VAR ASSIGN expression) | );
for_updation : ((VAR ASSIGN expression) | (VAR INC) | (VAR DEC) |);

/** range_block: User can use nested for range loops and execute a block.*/
range_block: range_dec_block | range_inc_block;
range_inc_block :
    FOR VAR INC IN RANGE OPB range_from COMMA range_inc_to CPB
    OCB
        block
    CCB  ;
```

# Grammar

```
range_dec_block :
      FOR VAR DEC IN RANGE OPB range_from COMMA range_dec_to CPB
      OCB
          block
      CCB  ;
range_from : (INT | VAR | expression);
range_inc_to : (INT | VAR | expression);
range_dec_to : (INT | VAR | expression);

/* String operations */
string_operations:  left = string_expression DOT CONCAT OPB right = string_expression CPB #concatOperation
    | string_expression DOT LENGTH OPB CPB #lengthOperation
    | string_expression DOT SPLIT OPB STRING CPB #splitOperation
    | string_expression DOT SUBSTRING OPB expression CPB #substringOperation
    | string_expression DOT SUBSTRING OPB expression COMMA expression CPB #substringDoubleOperation
    | INTEGER DOT TOSTRING OPB expression CPB  #integerToStringOperation
    | BOOL DOT TOSTRING OPB condition CPB #booleanToStringOperation
    | STRING #stringOperation
    | VAR OSB (INT | VAR) CSB #stringArrayOperation;

/* Arrays */
array : int_array | bool_array | string_array;
int_array : OSB (INT (COMMA INT)* |) CSB;
bool_array : OSB (BOOLEAN (COMMA BOOLEAN)* |) CSB;
string_array : OSB (STRING (COMMA STRING)* |) CSB;

/* Arrays Properties */
array_properties: VAR DOT LENGTH #arrayLengthProperty;

/** ternary_block: User can use ternary operator and evaluate expressions.*/
ternary_block : condition_block QUESTION ternary_true_block COLON ternary_false_block ;
ternary_true_block : (expression | condition);
ternary_false_block : (expression | condition);

/** assign_block: User can use this to assign expressions or strings to a variable.*/
assign_block : VAR ASSIGN (condition | expression | ternary_block | string_operations | array | array_properties) SEMICOLON ;
string_expression: (VAR | STRING);

/** show: User can use this to display a variable.*/
show : 'show' (VAR | INT | BOOLEAN | STRING) SEMICOLON;

INTEGER : 'Integer';
```

# Grammar

```
BOOL : 'Boolean';
TYPE : 'Int' | 'Bool' | 'String' | 'Int[]' | 'Bool[]' | '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 : '}' ;
OSB : '[' ;
CSB : ']' ;
SEMICOLON : ';' ;
COLON : ':' ;
COMMA : ',' ;
QUESTION : '?' ;
DOT : '.';
IF : 'if' ;
ELSE : 'else' ;
WHILE : 'while' ;
FOR : 'for' ;
RANGE: 'range' ;
IN : 'in' ;
LENGTH : 'length';
CONCAT : 'concat';
EQUALS : 'equals';
TOSTRING : 'toString';
SPLIT : 'split';
SUBSTRING : 'substring';
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 ;
```

# Lexical Analyzer

Lexing or tokenization is provided by ANTLR, which takes the .sea file as input and generates tokens which is sent as input to the parser

<u>Sample input</u>

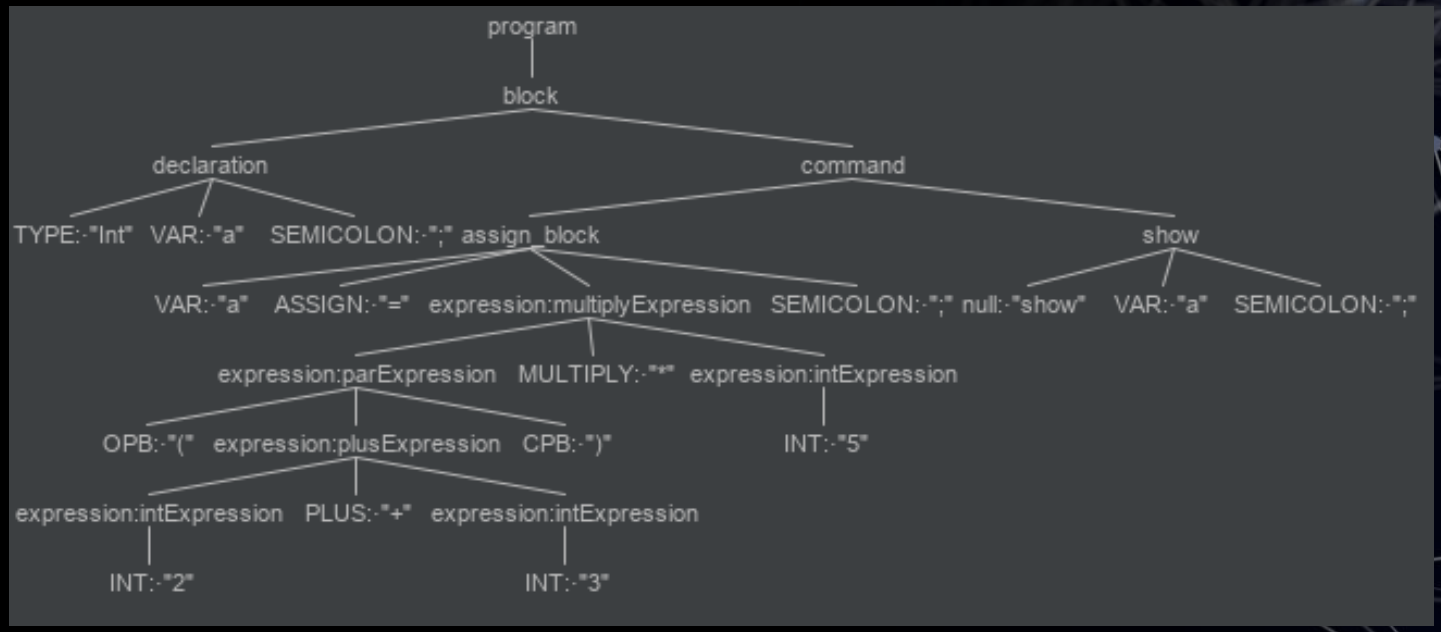
Int a;

a = (2 + 3) * 5;

show a;

# Parser

Parsing is also provided by ANTLR, it performs syntactic analysis on the tokens generated by the lexer and generates a parse tree



# Runtime

Based on the parse tree, first the intermediate code stack is populated and then processed by the runtime

```
0 = "DECLARATION:Int:a"
1 = "SET_INT_VAL:2"
2 = "SET_INT_VAL:3"
3 = "PLUS"
4 = "SET_INT_VAL:5"
5 = "MULTIPLY"
6 = "ASSIGN:a"
```

```
public void execute() throws ArithmeticException, VariableNotDeclaredException, VariableAlreadyDefinedException,
        LogicalOperatorException, StringOperatorException, ArrayOperatorException {
    int size = mIntermediateCode.size();

    for (mIndex = 0; mIndex < size; mIndex++) {

        String value = mIntermediateCode.get(mIndex);
        String[] data = value.split(IntermediateConstants.SEPARATOR);
```

# Installation

## Step 1

```
91700@LAPTOP-0CAPUCTG MINGW64 /D
$ git clone git@github.com:amitmaharana/SER502-Spring2020-Team1.git
Cloning into 'SER502-Spring2020-Team1'...
Enter passphrase for key '/c/Users/91700/.ssh/id_rsa':
remote: Enumerating objects: 255, done.
remote: Counting objects: 100% (255/255), done.
remote: Compressing objects: 100% (150/150), done.
remote: Total 718 (delta 128), reused 197 (delta 80), pack-reused 463
Receiving objects: 100% (718/718), 321.22 KiB | 1.51 MiB/s, done.
Resolving deltas: 100% (335/335), done.

91700@LAPTOP-0CAPUCTG MINGW64 /D
$ cd SER502-Spring2020-Team1/
```

## Step 2

```
91700@LAPTOP-0CAPUCTG MINGW64 /D/SER502-Spring2020-Team1 (master)
$ mvn compile package
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------< SER502-Spring2020-Team1:SEALang >-------------------
[INFO] Building SEALang 1.0
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- antlr4-maven-plugin:4.8:antlr4 (antlr) @ SEALang ---
[INFO] ANTLR 4: Processing source directory D:\SER502-Spring2020-Team1\src\main\
antlr4
[INFO] Processing grammar: SEALang.g4
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ SEALang --
-
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory D:\SER502-Spring2020-Team1\src\main\r
esources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ SEALang ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 16 source files to D:\SER502-Spring2020-Team1\target\classes
```

## Step 3

```
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  5.786 s
[INFO] Finished at: 2020-04-26T17:45:45-07:00
[INFO] ------------------------------------------------------------------------

91700@LAPTOP-0CAPUCTG MINGW64 /D/SER502-Spring2020-Team1 (master)
$ java -jar target/SEALang.jar data/helloworld.sea
"Hello World"
```

# Sample Program

```
91700@LAPTOP-0CAPUCTG MINGW64 /D/study/asu/SER502-Spring2020-Team1 (development)
$ cat data/prime.sea
// Write a program to check if a number is prime or not

Int n;
Int i;
Int c;
Bool isprime;
String text;

n = 11;
i = 2;
text = "is a prime number";

while(i < n){
        if((n - (n / i) * i) == 0){
                c = c + 1;
        }
        i = i + 1;
}

isprime = (c == 0)? True : False;

show n;
show text;
show isprime;
91700@LAPTOP-0CAPUCTG MINGW64 /D/study/asu/SER502-Spring2020-Team1 (development)
$ java -jar target/SEALang.jar data/prime.sea
11
is a prime number
true
```

# Thank You

Amit | Eric | Shubham | Shwetank

Team 1