The APLA Language

09.04.2017

Anuj Thula, Param Metha, Luis Quintanilla, Sai Aditya Bodavala
Team 31,
SER 502,
Arizona State University

Name:

The name of our language is APLA this is the Initials of all our names in a random order. Also, in *Marathi* language it means *"Ours".*

Features

- **Datatypes:** Integer, Boolean, String
- **Decisions:** If- then - else.
- **Looping:** While
- **Operators:**
    - **Assignment:** =
    - **Arithmetic:** +, -, *, /
    - **Logical:** and, or
- **Console write:** “println”**.**

Interpreter Used:

We have used the “antlr” interpreter in our language.

In computer-based language recognition, **ANTLR** (pronounced *Antler*), or **Another Tool For Language Recognition**, is a parser generator that uses LL(*) for parsing. ANTLR is a robust framework that has been developed for over 25 years and is used in various proven technologies such as Hadoop, Hive etc.

ANTLR is the successor to the **Purdue Compiler Construction Tool Set** (**PCCTS**), first developed in 1989, and is under active development. Its maintainer is Professor Terence Parr of the University of San Francisco.

Design:

Datatypes

- Support for integers, Boolean and String
- Keywords support for int, string, If - then - else, println, while.

Decision

- If - then - else support.
- Also support for if.

Loop

- Logical looping using While.

Exception Handling

- Integer exception handling.
- Variable, String exception handling.

Method

- User can also use methods in the program.

File Input

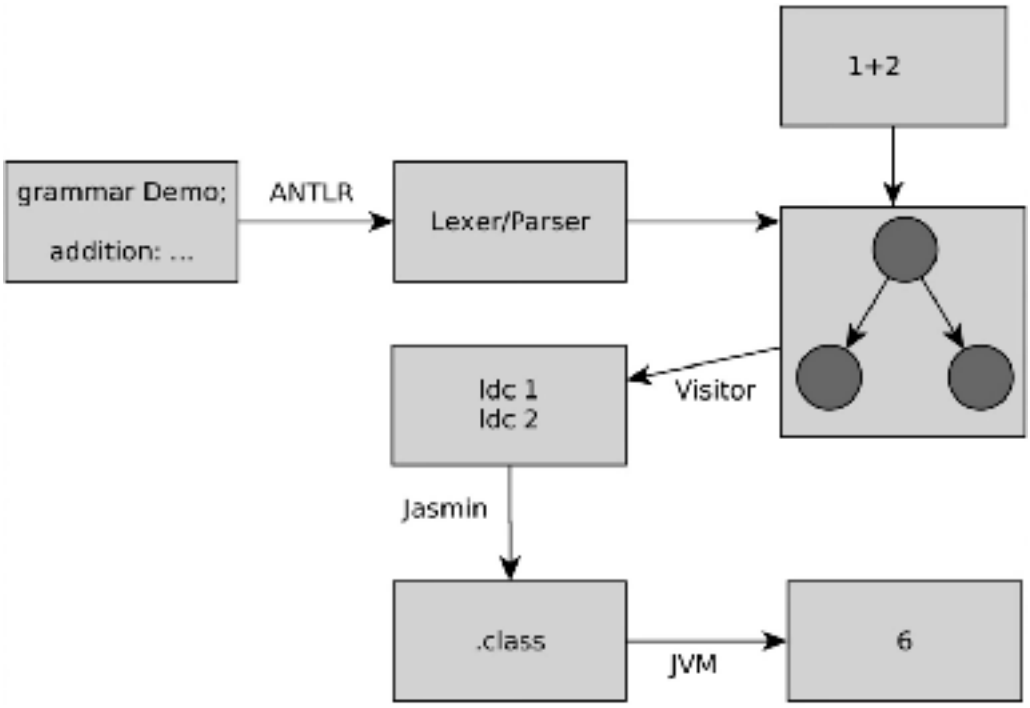
- The user input can also be accepted using a .txt file.

Assignment

- The user input can also be accepted using a .txt file.

RunTime:

- So for our language we have implemented a bottom-up parsing technique design which is shown by the diagram below:
- Bottom-up parsing starts from the leaf nodes of a tree and works in upward direction till it reaches the root node. Here, we start from a sentence and then apply production rules in reverse manner in order to reach the start symbol. The image given below depicts the bottom-up parsers available.
- The input would be a parse tree.



Grammar:

The following is the grammar we have used in our language:

```
grammar APLA;

program
    : programPiece+ ;

programPiece
    : statement #StatementPiece
    | method #MethodPiece
    ;

statement
    : println ';'
    | varAssignment ';'
    | assignment ';'
    | branch
    ;

branch
    : 'if' '(' condition=expression ')' True=section 'else'
False=section ;

section
    : '{' statement* '}' ;

expression
    : left=expression operator=('*' | '/') right=expression #MULTDIV
    | left=expression operator=('+' | '-') right=expression #PLUSMINUS
    | num=NUM #Number
    | varName=NAME #Variable
```

```
    | methodCall #MethodExp
    ;

assignment: varName=NAME '=' expr=expression ;

varAssignment
    : 'int' varName=NAME;

println
    : 'println(' argument=expression ')' ;

while_statement
: 'WHILE' expression_condition '{' block '}'

method
    : 'int' methName=NAME '(' ')' '{' statements=statementList
'return' returnVal=expression ';' '}' ;

statementList: statement* ;

methodCall
    : methName=NAME '(' ')' ;

NAME
    : [a-zA-Z][a-zA-Z0-9]*;

NUM
    : [0-9]+;

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