Language Name: Sparky

General System Requirements:

Processor: Intel and AMD processors (Processors with instruction set capable of imperative paradigm)

Operating System on which compiler and runtime are built: Windows OS.

Type of Language: Imperative

Tools Used: Git, Eclipse, Antlr (https://www.antlr.org/)

Steps to install and run Antlr on Windows:

- Download https://www.antlr.org/download/antlr-4.8-complete.jar.
- Add antlr4-complete.jar to CLASSPATH, either:
- Permanently: Using System Properties dialog > Environment variables > Create or append to CLASSPATH variable
- Temporarily, at command line:
- SET CLASSPATH=.;C:\Javalib\antlr4-complete.jar;%CLASSPATH%
- Create batch commands for ANTLR Tool, TestRig in dir in PATH
- antlr4.bat: java org.antlr.v4.Tool %*
- grun.bat: java org.antlr.v4.gui.TestRig %*

Alternative Steps using Dos Key commands to run Antlr:

- Doskey antlr4=java org.antlr.v4.Tool $*
- Doskey grun =java org.antlr.v4.gui.TestRig $*

Grammar Snippet

```
1.  grammar Sparky;
2.
3.
4.  //prog : LIVE declare* ball DIE;
5.  prog: LIVE declare* ball DIE;
6.  ball : expression* ;
7.
8.  declare:  datatype STUFF EQUALTO assignedstuff SEMICOLON
9.          |  datatype STUFF SEMICOLON;
10.
11.
12. /*
13. expression: loopum SEMICOLON| expression PLUS term SEMICOLON
14.                 | expression MINUS term SEMICOLON| term  SEMICOLON| assignment| yesnost
    atement;
15. */
16. // removing left recursion by alpha-beta rule.
17.
18. expression : assignment e1 | loopum SEMICOLON e1 | term  SEMICOLON e1;
19. e1: PLUS term SEMICOLON e1| MINUS term SEMICOLON e1 | yesnostatement e1 |;
20.
21.
22.
```

```
23.
24. loopum :IF term2 THEN  in_loop (ELSE  term2)? FI
25.   |  WHILE  term2 LOOP in_loop  POOL;
26.
27.
28.
29. /*term: term MUL term2
30.       |term DIV term2 | NUMBER | STUFF; */
31. // removing left recursion by alpha-beta rule.
32.
33. term : NUMBER term1 | STUFF term1;
34. term1: MUL term2 term1 | DIV term2 term1|;
35.
36.
37. term2: LSmoothBrace yesnostatement RSmoothBrace;
38. in_loop: LCurlyBrace expression SEMICOLON RCurlyBrace;
39.
40.
41. assignedstuff: NUMBER|BOOLEANVALUE;
42.
43. assignment: STUFF EQUALTO expression;
44.
45. //YESNOSTATEMENT
46. yesnostatement: BOOLEANVALUE| expression YESNOOPERATOR expression;
47.
48. //primitive types
49. datatype: YUPNUP | INTEGER | STRING | DOUBLE | DECIMAL | CHAR;
50. YUPNUP: 'boolean';
51. INTEGER: 'int';
52. STRING: 'string';
53. DOUBLE: 'double';
54. DECIMAL: 'float';
55. CHAR : 'char';
56.
57. //NUMBER
58. NUMBER: [0-9]+;
59.
60. //STUFF
61. STUFF: [a-zA-Z_] [a-zA-Z_0-9]*;
62.
63. //BOOLEANVALUE
64. BOOLEANVALUE: YUP|NOPE;
65. YUP: 'true';
66. NOPE: 'false';
67.
68.
69.
70. //IFTE
71. IF : 'if';
72. ELSE:'else';
73. WHILE: 'while';
74.
75. //Separators
76. LSmoothBrace : '(';
77. RSmoothBrace : ')';
78. LCurlyBrace  : '{';
79. RCurlyBrace  : '}';
80. LSquareBrace : '[';
81. RSquareBrace : ']';
82. SEMICOLON    : ';';
83. COMMA        :  ',';
```

```
84.
85. //Operators
86. YESNOOPERATOR: ASSEQ| LESS_THAN| MORE_THAN | LESS_THAN_EQ | MORE_THAN_EQ ;
87. EQUALTO : '=';
88. ASSEQ : '==';
89. PLUS: '+';
90. MINUS : '-';
91. MUL : '*';
92. DIV : '/';
93. LESS_THAN : '<';
94. MORE_THAN: '>';
95. LESS_THAN_EQ : '<=';
96. MORE_THAN_EQ : '>=';
97.
98. WS: [ \t\r\n\f]+ -> skip;
99.
100.
101.       LIVE: L I V E;
102.       DIE: D I E;
103.       FI: F I;
104.       THEN : T H E N;
105.       LOOP : L O O P;
106.       POOL : P O O L;
107.       fragment T: [tT];
108.       fragment H: [hH];
109.       fragment E: [eE];
110.       fragment N: [nN];
111.       fragment I: [iI];
112.       fragment L: [lL];
113.       fragment O: [oO];
114.       fragment P: [pP];
115.       fragment F: [fF];
116.       fragment V: [vV];
117.       fragment D: [dD];
```

Example of how final Code snippets will look:

```
1.  Example 1
2.  int a = 7;
3.  int b = 5;
4.  int c;
5.  c = a+b;
6.
7.
8.  ..................................
9.
10. Example 2
11. IF (YUP)
12. THEN {x=3;}
13. ELSE
14. {x=4;}
15. FI;
```