

19CSE401 - Compiler Design

JLex Programming

Lab sheet-5

Done By:

N Sai Pavan Krishna

AM.EN.U4CSE19347

- The image shows a screenshot of an IDE (likely IntelliJ IDEA) with a dark theme. The Explorer panel on the left shows a project structure with a folder named 'lab5' selected. Inside 'lab5', there are files: 'Main.class', 'q1.jlex', 'Token.class', 'Yylex.class', 'Yylex.java', and 'Yylex.java~'. The main editor displays the source code of 'q1.jlex', which is a Java file containing macros and a simple parser implementation. The code defines macros for string, digit, letter, and special characters, and implements a simple parser that prints the input string and its components. The Terminal panel at the bottom shows the execution output, including warnings about unused macros and the final output: 'Hello World', 'abc123', '123', '123', 'Integer: 123', '12.3', and 'Float: 12.3'.

```

16  }
17  }
18
19  %%
20  string = "\""\".\"\"\"
21  digit = [0-9]
22  letter = [a-zA-Z]
23  special = ![@#$%^&*()_+ ]
24  whitespace = [ \t\n]
25
26  %type Token
27  %eofval{
28      return new Token(null);
29  %eofval}
30
31  %%
32  {string} {System.out.println("String: "+yytext());}
33  {digit}+ {System.out.println("Integer: "+yytext());}
34  {digit}+".{(digit)+ {System.out.println("Float: "+yytext());}
35  {whitespace}+ {/*Skip white spaces*/}
36  . {}
37

```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** JUPYTER

```

(base) nspk@Crisp lab5 % jflex q1.jlex
Reading "q1.jlex"

Warning : Macro "special" has been declared but never used.

Warning : Macro "letter" has been declared but never used.
Constructing NFA : 30 states in NFA
Converting NFA to DFA :
.....
12 states before minimization, 9 states in minimized DFA
Old file "Yylex.java" saved as "Yylex.java~"
Writing code to "Yylex.java"
(base) nspk@Crisp lab5 % javac Yylex.java
(base) nspk@Crisp lab5 % java Main
"Hello World"
String: "Hello World"
"abc123"
String: "abc123"
"123"
String: "123"
123
Integer: 123
12.3
Float: 12.3
^C
(base) nspk@Crisp lab5 %

```

2. Consider the following token

Token	Lexemes		Token	Lexemes
MAIN	main		PRINTF	printf
LPAREN	{		SCANF	scanf
RPAREN	}		RETURN	return
LBRACE	(INT	int
RBRACE)		FLOAT	float
ID	For all identifiers		CHAR	char
NUM	Integer constants		/* - -- */	Multiline comment
STR	String Constant		//	Single line comment
REAL	Floating-point constants		SEMI	;
IF	If		COMMA	,
WHILE	while		ARITHMETIC	+, -, *, /, ++, --
SWITCH	switch		LOGIC	&&, , !
CASE	case		RELATIONAL	<, <=, >, >=, ==, !=
BREAK	break			

Write a Jlex program that generates the token of the form **<Token, lexeme>** except for keywords, for the given program. If none of the patterns matches for a lexeme, given an error statement specifying the line number in the program.

```

int main()
{
    int c, n, f = 1;

    printf("Enter a number to calculate its factorial\n");
    scanf("%d", &n);

    for (c = 1; c <= n; c++)
        f = f * c;

    printf("Factorial of %d = %d\n", n, f);

    return 0;
}

```