SSN COLLEGE OF ENGINEERING, KALAVAKKAM

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

UCS1602 - Compiler Design

Programming Assignment 3

Implementation of Syntax checker using Lex and Yacc Tools

Name: Jayannthan P T *Dept: CSE 'A'* *Roll No.: 205001049*

Programming Assignment-2 - Implementation of Syntax checker using Lex and Yacc Tools (Java Programming Language)

Source code:

syntax.l

```
%{
%{
    #include<stdio.h>
    #include<stdlib.h>
    #include<string.h>
    #include "y.tab.h"
    int yylex(void);
    int yyerror(char* s);
    extern int yylval;
    int debug=0;
    int line=0;
%}

%x COMMENT

%%
"/*"            { BEGIN(COMMENT); }
<COMMENT>"*/"   { BEGIN(INITIAL); }

[ \t]+ ;
\n {line++;}
[0-9]+ { printf("%s is an integer\n", yytext);return NUM;}
("int"|"float"|"double"|"long"|"short"|"byte"|"char"|"String"|"Boolean"|"void")
{printf("%s is a data type\n", yytext);return DTYPE; }
^import.* { printf("%s is an PPD\n", yytext);return PPD;}
"if" {printf("IF PART\n"); return IF;}
"while" {printf("WHILE LOOP\n"); return WHILE;}
"for" {printf("FOR LOOP\n"); return FOR; }
"else" {printf("ELSE PART\n"); return ELSE;}
```

```
"this" {printf("%s is a keyword\n", yytext);return THIS;}
"class" {printf("%s is a keyword\n", yytext);return CLASS;}
"public" {printf("%s is a keyword\n", yytext);return PUBLIC;}
"private" {printf("%s is a keyword\n", yytext);return PRIVATE;}
"static" {printf("%s is a keyword\n", yytext);return STATIC;}

[_a-zA-Z][a-zA-Z0-9_]* { printf("%s is an identifier\n", yytext);return ID; }
";" {printf("End of statement\n\n");return EOS;}
(">"|"<"|"<="|">="|"!="|"==") { printf("%s is comparison operator\n",yytext); return
COMPARISON_OP; }
("+="|"-="|"*="|"/="|"%="|"=") { printf("%s is an assign op\n", yytext);return ASSIGN_OP;
}
("++"|"--") { printf("%s is an incr-decr op\n", yytext);return INDE_OP; }
">>" { printf("rs operator\n"); return RSHIFT; }
"<<" {printf("ls operator\n"); return LSHIFT; }
"!" { printf("NOW operator\n"); return NOT; }
"||" {printf("OR operator\n"); return OR; }
"&&" {printf("AND operator\n"); return AND; }
"{" { printf("\n-----START_BLOCK-----\n"); return *yytext; }
"}" { printf("-----END_BLOCK-----\n\n"); return *yytext; }
("+"|"-"|"*"|"/"|"%") { printf("%s is an arith op\n", yytext);return ARITH_OP; }


"\\" {return *yytext;}/*spl chars*/
"." {printf("%s\n",yytext); return *yytext;}
"," {printf("%s\n",yytext); return *yytext;}
"(" {printf("%s\n",yytext); return *yytext;}
")" {printf("%s\n",yytext); return *yytext;}

. {
  char msg[25];
  sprintf(msg,"Unknown token found: <%s>\n", yytext);
  yyerror(msg);
}
%%
```

syntax.y

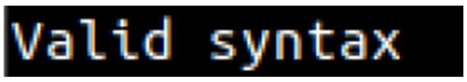
```
%{
    #include <stdlib.h>
    #include <stdio.h>
    int yylex(void);
    extern FILE* yyin;
    #include "y.tab.h"
    int error = 0;
    /*extern int debug;*/
    extern int line;
%}

%token NUM DTYPE EOS PPD IF WHILE FOR ELSE ID COMPARISON_OP ASSIGN_OP INDE_OP RSHIFT
LSHIFT NOT OR AND ARITH_OP NEW THIS CLASS PUBLIC PRIVATE STATIC
```

```
%%
program:
    statement_list
    ;

statement_list:
    statement
    | statement_list statement
    ;

method_declaration:
    PUBLIC STATIC DTYPE ID '(' DTYPE ID ',' DTYPE ID ')' '{' statement_list '}'
    | PUBLIC STATIC DTYPE ID '(' DTYPE ID ')' '{' statement_list '}'
    | PUBLIC STATIC DTYPE ID '(' DTYPE '[' ']' ID ')' '{' statement_list '}'
    | PUBLIC DTYPE ID '(' DTYPE ID ')' '{' statement_list '}'
    | PUBLIC DTYPE ID '(' DTYPE ID ',' DTYPE ID ')' '{' statement_list '}'
    | PUBLIC DTYPE ID '(' DTYPE ID ',' DTYPE ID ',' DTYPE ID ')' '{' statement_list '}'
    ;

class_declaration:
    PUBLIC CLASS ID '{' statement_list '}'
    | CLASS ID '{' statement_list '}'
    | CLASS DTYPE ID '{' method_declaration '}' EOS
    ;

statement:
    declaration_statement
    | assignment_statement
    | comparison_statement
    | logical_statement
    | increment_decrement_statement
    | block_statement
    | selection_statement
    | iteration_statement
    | method_declaration
    | class_declaration
    ;

declaration_statement:
    DTYPE ID EOS
    |
    ;

assignment_statement:
    ID ASSIGN_OP expression EOS
    ;

comparison_statement:
    expression COMPARISON_OP expression EOS
    ;

logical_statement:
    expression OR expression EOS
    | expression AND expression EOS
    | NOT expression EOS
    ;

increment_decrement_statement:
    ID INDE_OP EOS
    ;

block_statement:
    '{' statement_list '}'
    ;

selection_statement:
    IF '(' expression ')' statement
    | IF '(' expression ')' statement ELSE statement
    ;

iteration_statement:
    WHILE '(' expression ')' statement
    | FOR '(' assignment_statement comparison_statement increment_decrement_statement ')'
statement
    ;

expression:
    ID
    | NUM
    | NEW ID
    | THIS
    | expression ARITH_OP expression
    | '(' expression ')'
    ;

%%


int yyerror(){
    fprintf(stderr, "\n\t %s Error at line %d\n\n",stderr, line);
    error = 1;
    return 0;
}

int yywrap(){
    return 1;
}

int main(int argc, char **argv){
    /*yydebug = 1;*/
    if(argc != 2){
        fprintf(stderr, "Enter file name as argument!\n");
        return 1;
    }
    yyin = fopen(argv[1], "rt");
    if (!yyin){
        fprintf(stderr, "%s File not found!\n",stderr);
        return 2;
    }
    yyparse();
    if(!error){
        printf("Valid syntax!\n");
    }
    return 0;
}
```

Input Code:

```
public class Main {
    public static void main(String[] args) {
        int i = 0;
        int a = 5;
        while (i < 10) {
            if (i < a) {
                i = i + 1;
            } else {
                i = i - 1;
            }
        }
        System.out.println("The final value of i is " + i);
    }
}
```

Output:



```
Valid syntax
```

Learning Outcome:

- Understood the working of yacc for debugging of programs.
- Understood the role of yacc in running a program
- Understood how to write yacc programs