First Phase

Negar Neda — 9331050 / Sare Soltani nejad — 933103

LEX INPUT:

```
package phase;

import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.lang;*.
import java.util.ArrayList;

class Phase {
            ArrayList simbol_table = new ArrayList();
    public static void main(String[] args) {
        FileReader filereader = null;
        String input = "testcase.txt;"
        try {
            filereader = new FileReader(input);
      }  catch (FileNotFoundException e) {
            e.printStackTrace();
        }
        System.out.println("Lexeme\t\t\t Token\t\t\t simbol_table");
        Yylex yylex = new Yylex(filereader);
        try {
            yylex.yylex();
       } catch (IOException e) {
            e.printStackTrace();
      }}}

%%
%byaccj
LETTER = [a-zA-Z]
NONZERO_DIGIT =  [1-9]
DIGIT ="0"| {NONZERO_DIGIT}
```

PROGRAM_KW = (program)
EMPTY_KW = (empty)

VAR_KW = (var)
INTEGER_KW = (integer)
REAL_KW = (real)
BOOLEAN_KW = (bool)

PROCEDURE_KW = (procedure)

IF_KW = (if)
THEN_KW = (then)
ELSE_KW= (else)

REPEAT_KW = (repeat)
EXIT_KW = (exit)

FOR_KW = (for)
TO_KW = (to)
DO_KW = (do)
BEGIN_KW = (begin)
END_KW = (end)
DOWNTO_KW = (downto)

DIV_KW = (div)
MOD_KW = (mod)
AND_KW = (and)
OR_KW = (or)
NOT_KW = (not)

SEMICOLON_KW = [;]
COLON_KW= [:]
COMMA_KW= [,]
SINGLE_QUOTE_KW = "\u0027"
ASS_KW = (=:)

```
LP_KW = [)]
RP_KW = [(]
LB_KW = "]"
RB_KW  =  "["
LCB_KW=  [}]
RCB_KW=   [{]

QUESTION_KW =   [?]
EQUALS_KW=   [=]
DOT_KW =    ".\"
LE_KW = {DOT_KW}(le)
LT_KW = {DOT_KW}(lt)
GT_KW = {DOT_KW}(gt)
GE_KW = {DOT_KW}(ge)
EQ_KW = {DOT_KW}(eq)
NE_KW = {DOT_KW}(ne)

ADD_KW=   [+]
SUB_KW  =  [-]
MUL_KW =   [*]
DIV_KW =   [/]
MOD_KW=  [%]


WHITE_SPACE = [\n\r\t]
NEWLINE =[\n]
LineTerminator = \r|\n|\r\n
InputCharacter = [^\r\n]
COMMENTS="//"{InputCharacter}*{LineTerminator}?



BOOLEAN_CONSTANT = (true) | (false)
IDENTIFIER ={LETTER}({LETTER}|{DIGIT})*
INTEGER_CONSTANT = [#]{DIGIT}|[#]{NONZERO_DIGIT}{DIGIT}*
REAL_CONSTANT =
[#](({DIGIT})|({NONZERO_DIGIT}({DIGIT})*))({DOT_KW})(({DIGIT})*{NONZERO_DIGIT} | 0)
```

```
ERROR_NO_SHARP= {DIGIT}|{NONZERO_DIGIT}{DIGIT}* |
(({DIGIT})|({NONZERO_DIGIT}({DIGIT})*))({DOT_KW})({DIGIT})*{NONZERO_DIGIT}

ERROR_ZERO = [#]{DIGIT}*{DOT_KW}{DIGIT}* | {DIGIT}*{DOT_KW}{DIGIT}* |
[#]0*{NONZERO_DIGIT}{DIGIT}* | 0*{NONZERO_DIGIT}{DIGIT}*


%%
}PROGRAM_KW} {
        System.out.println(yytext() + "\t\t\t" + "PROGRAM_KW\t\t\t" + '-');
{

}EMPTY_KW} {
        System.out.println(yytext() + "\t\t\t" + "EMPTY_KW\t\t\t" + '-');
{
}VAR_KW} {
        System.out.println(yytext() + "\t\t\t" + "VAR_KW\t\t\t" + '-');
{
}INTEGER_KW} {
        System.out.println(yytext() + "\t\t\t" + "INTEGER_KW\t\t\t" + '-');
{
}REAL_KW} {
        System.out.println(yytext() + "\t\t\t" + "REAL_KW\t\t\t" + '-');
{

}PROCEDURE_KW} {
        System.out.println(yytext() + "\t\t\t" + "PROCEDURE_KW\t\t\t" + '-');
{
}IF_KW} {
        System.out.println(yytext() + "\t\t\t" + "IF_KW\t\t\t" + '-');
{

}THEN_KW} {
        System.out.println(yytext() + "\t\t\t" + "THEN_KW\t\t\t" + '-');
{
```

```
}ELSE_kw} {
        System.out.println(yytext() + "\t\t\t" + "ELSE_kw\t\t\t" + '-');
{
}REPEAT_KW} {
        System.out.println(yytext() + "\t\t\t" + "REPEAT_KW\t\t\t" + '-');
{
}EXIT_KW} {
        System.out.println(yytext() + "\t\t\t" + "EXIT_KW\t\t\t" + '-');
{
}END_KW} {
        System.out.println(yytext() + "\t\t\t" + "END_KW\t\t\t" + '-');
{
}FOR_KW} {
        System.out.println(yytext() + "\t\t\t" + "FOR_KW\t\t\t" + '-');
{
}TO_KW} {
        System.out.println(yytext() + "\t\t\t" + "TO_KW\t\t\t" + '-');
{
}DOT_KW} {
        System.out.println(yytext() + "\t\t\t" + "DOT_KW\t\t\t" + '-');
{
}BEGIN_KW} {
        System.out.println(yytext() + "\t\t\t" + "BEGIN_KW\t\t\t" + '-');
{
}DOWNTO_KW} {
        System.out.println(yytext() + "\t\t\t" + "DOWNTO_KW\t\t\t" + '-');
{
}DIV_KW} {
        System.out.println(yytext() + "\t\t\t" + "DIV_KW\t\t\t" + '-');
{
}MOD_KW} {
        System.out.println(yytext() + "\t\t\t" + "MOD_KW\t\t\t" + '-');
{
}AND_KW} {
        System.out.println(yytext() + "\t\t\t" + "AND_KW\t\t\t" + '-');
{
}OR_KW} {
```

```
            System.out.println(yytext() + "\t\t\t" + "OR_KW}\t\t\t;('-' + "
{
}NOT_KW} {
            System.out.println(yytext() + "\t\t\t" + "NOT_KW\t\t\t" + '-');
{
}SEMICOLON_KW} {
            System.out.println(yytext() + "\t\t\t" + "SEMICOLON_KW\t\t\t" + '-');
{
}COLON_KW} {
            System.out.println(yytext() + "\t\t\t" + "COLON_KW\t\t\t" + '-');
{
}COMMA_KW} {
            System.out.println(yytext() + "\t\t\t" + "COMMA_KW\t\t\t" + '-');
{
}ASS_KW} {
            System.out.println(yytext() + "\t\t\t" + "ASS_KW\t\t\t" + '-');
{
}LP_KW} {
            System.out.println(yytext() + "\t\t\t" + "LP_KW\t\t\t" + '-');
{
}RP_KW} {
            System.out.println(yytext() + "\t\t\t" + "RP_KW\t\t\t" + '-');
{
}LB_KW} {
            System.out.println(yytext() + "\t\t\t" + "LB_KW}\t\t\t;('-' + "
{
}RB_KW} {
            System.out.println(yytext() + "\t\t\t" + "RB_KW\t\t\t" + '-');
{
}LCB_KW} {
            System.out.println(yytext() + "\t\t\t" + "LCB_KW\t\t\t" + '-');
{
}RCB_KW} {
            System.out.println(yytext() + "\t\t\t" + "RCB_KW\t\t\t" + '-');
{
}QUESTION_KW} {
            System.out.println(yytext() + "\t\t\t" + "QUESTION_KW\t\t\t" + '-');
```

```
{
}EQUALS_KW} {
        System.out.println(yytext() + "\t\t\t" + "EQUALS_KW\t\t\t" + '-');
{
}LE_KW} {
        System.out.println(yytext() + "\t\t\t" + "LE_KW\t\t\t" + '-');
{
}LT_KW} {
        System.out.println(yytext() + "\t\t\t" + "LT_KW\t\t\t" + '-');
{
}GT_KW} {
        System.out.println(yytext() + "\t\t\t" + "GT_KW\t\t\t" + '-');
{
}GE_KW} {
        System.out.println(yytext() + "\t\t\t" + "GE_KW\t\t\t" + '-');
{
}EQ_KW} {
        System.out.println(yytext() + "\t\t\t" + "EQ_KW\t\t\t" + '-');
{
}NE_KW} {
        System.out.println(yytext() + "\t\t\t" + "NE_KW\t\t\t" + '-');
{
}ADD_KW} {
        System.out.println(yytext() + "\t\t\t" + "ADD_KW\t\t\t" + '-');
{
}SUB_KW} {
        System.out.println(yytext() + "\t\t\t" + "SUB_KW\t\t\t" + '-');
{
}MUL_KW} {
        System.out.println(yytext() + "\t\t\t" + "MUL_KW\t\t\t" + '-');
{
}DIV_KW} {
        System.out.println(yytext() + "\t\t\t" + "DIV_KW\t\t\t" + '-');
{
}MOD_KW} {
        System.out.println(yytext() + "\t\t\t" + "MOD_KW\t\t\t" + '-');
{
```

```
}BOOLEAN_CONSTANT} {
        System.out.println(yytext() + "\t\t\t" + "BOOLEAN_CONSTANT\t\t\t" + '-');
{
}BOOLEAN_KW} {
        System.out.println(yytext() + "\t\t\t" + "BOOLEAN_KW\t\t\t" + '-');
{
}IDENTIFIER} {
        System.out.println(yytext() + "\t\t\t" + "IDENTIFIER\t\t\t" + '-');
{
}INTEGER_CONSTANT} {
        System.out.println(yytext() + "\t\t\t" + "INTEGER_CONSTANT\t\t\t" + '-');
{
}REAL_CONSTANT} {
        System.out.println(yytext() + "\t\t\t" + "REAL_CONSTANT\t\t\t" + '-');
{
}ERROR_NO_SHARP} {
        System.out.println(yytext() + "\t\t\t" + "ERROR_NO_SHARP\t\t\t" + '-');
{
}ERROR_ZERO} {
        System.out.println(yytext() + "\t\t\t" + "ERROR_ZERO\t\t\t" + '-');
{
\"s"|"\n"|"\r"|"\t} "
{
} .
{
```

SAMPLE CODE :

```
program compilerPhaseOne
        var
                sare ,negar : real;
                num : integer;
        procedure calc(x,y,z,a,b,c : integer; k : real);
                var m : integer;
                begin
                        x:= #0.1 + #12.000;
                        y:= #0012 * #2;
                        z:= #51.0 % #39;
                        a:= #1;
                        b:= #1;
                        if(x .lt y) then
                                m := x;
                        else
                                m := y/2;
                        if(z .gt m) then
                                m := z;

                        repeat
                                m := m - #1;
                                exit if (m .le #1)
                                m := m + #1;
                        end

                        c := a and b;
                        if ( c .eq #0) then
                                c:= a or b;

                end
```

OUTPUT:

Lexeme	Token	simbol_table
program	PROGRAM_KW	-
compilerPhaseOne	IDENTIFIER	-
var	VAR_KW	-
sare	IDENTIFIER	-
,	COMMA_KW	-
negar	IDENTIFIER	-
:	COLON_KW	-
real	REAL_KW	-
;	SEMICOLON_KW	-
num	IDENTIFIER	-
:	COLON_KW	-
integer	INTEGER_KW	-
;	SEMICOLON_KW	-
procedure	PROCEDURE_KW	-
calc	IDENTIFIER	-
(	LP_KW	-
x	IDENTIFIER	-
,	COMMA_KW	-
y	IDENTIFIER	-
,	COMMA_KW	-
z	IDENTIFIER	-
,	COMMA_KW	-
a	IDENTIFIER	-

,	COMMA_KW	-	
b	IDENTIFIER	-	
,	COMMA_KW	-	
c	IDENTIFIER	-	
:	COLON_KW	-	
integer	INTEGER_KW	-	
;	SEMICOLON_KW		-
k	IDENTIFIER	-	
:	COLON_KW	-	
real	REAL_KW	-	
)	RP_KW	-	
;	SEMICOLON_KW		-
var	VAR_KW	-	
m	IDENTIFIER	-	
:	COLON_KW	-	
integer	INTEGER_KW	-	
;	SEMICOLON_KW		-
begin	BEGIN_KW	-	
x	IDENTIFIER	-	
:=	ASS_KW	-	
#0.1	REAL_CONSTANT		-
+	ADD_KW	-	
#12.000	ERROR_ZERO		-
;	SEMICOLON_KW		-
y	IDENTIFIER	-	
:=	ASS_KW	-	

#0012	ERROR_ZERO	-	
*	MUL_KW	-	
#2	INTEGER_CONSTANT		-
;	SEMICOLON_KW		-
z	IDENTIFIER	-	
:=	ASS_KW	-	
#51.0	REAL_CONSTANT		-
%	MOD_KW	-	
#39	INTEGER_CONSTANT		-
;	SEMICOLON_KW		-
a	IDENTIFIER	-	
:=	ASS_KW	-	
#1	INTEGER_CONSTANT		-
;	SEMICOLON_KW		-
b	IDENTIFIER	-	
:=	ASS_KW	-	
#1	INTEGER_CONSTANT		-
;	SEMICOLON_KW		-
if	IF_KW	-	
(	LP_KW	-	
x	IDENTIFIER	-	
.lt	LT_KW	-	
y	IDENTIFIER	-	
)	RP_KW	-	
then	THEN_KW	-	
m	IDENTIFIER	-	

:= ASS_KW -

x IDENTIFIER -

; SEMICOLON_KW -

else ELSE_kw -

m IDENTIFIER -

:= ASS_KW -

y IDENTIFIER -

/ DIV_KW -

2 ERROR_NO_SHARP -

; SEMICOLON_KW -

if IF_KW -

(LP_KW -

z IDENTIFIER -

.gt GT_KW -

m IDENTIFIER -

) RP_KW -

then THEN_KW -

m IDENTIFIER -

:= ASS_KW -

z IDENTIFIER -

; SEMICOLON_KW -

repeat REPEAT_KW -

m IDENTIFIER -

:= ASS_KW -

m IDENTIFIER -

- SUB_KW -

#1	INTEGER_CONSTANT		-
;	SEMICOLON_KW		-
exit	EXIT_KW	-	
if	IF_KW	-	
(	LP_KW	-	
m	IDENTIFIER	-	
.le	LE_KW	-	
#1	INTEGER_CONSTANT		-
)	RP_KW	-	
m	IDENTIFIER	-	
:=	ASS_KW	-	
m	IDENTIFIER	-	
+	ADD_KW	-	
#1	INTEGER_CONSTANT		-
;	SEMICOLON_KW		-
end	END_KW	-	
c	IDENTIFIER	-	
:=	ASS_KW	-	
a	IDENTIFIER	-	
and	AND_KW	-	
b	IDENTIFIER	-	
;	SEMICOLON_KW		-
if	IF_KW	-	
(	LP_KW	-	
c	IDENTIFIER	-	
.eq	EQ_KW	-	

#0 INTEGER_CONSTANT -

) RP_KW -

then THEN_KW -

c IDENTIFIER -

:= ASS_KW -

a IDENTIFIER -

or OR_KW} -

b IDENTIFIER -

; SEMICOLON_KW -

end END_KW -

BUILD SUCCESSFUL (total time: 0 seconds)