Lex Input:

```
package jSHLang;

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

class MainClass {
    public static void main(String[] args) {
        FileReader fr = null;
        String input = ".\\files\\Code.shl";
        try {
            fr = new FileReader(input);
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
        System.out.println("Lexeme\tToken\tAttribute");
        Yylex yylex = new Yylex(fr);
        try {
            yylex.yylex();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

%%

%byaccj

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

PROGRAM_KW = (program)
MAIN_KW = (main)

PROCEDURE_KW = (procedure)

EMPTY_KW = (empty)

INTEGER_KW = (int)
REAL_KW = (real)
CHAR_KW = (char)

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

DO_KW = (do)
WHILE_KW = (while)
```

```
FOR_KW = (for)
IN_KW = (in)
REPEAT_KW = (repeat)

CASE_KW = (case)
DEFAULT_KW = (default)

RETURN_KW = (return)
EXIT_KW = (exit)
WHEN_KW = (when)

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 = [}]

TWO_DOTS_KW = "\.\."
DOT_KW = "\."

EQ_KW = [=]
NE_KW = (<>)
LE_KW = (<=)
LT_KW = [<]
GE_KW = (>=)
GT_KW = [>]

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

CHAR_CONSTANT = {SINGLE_QUOTE_KW} ({LETTER} | {DIGIT}) {SINGLE_QUOTE_KW}
REAL_CONSTANT =
(({DIGIT})|({NONZERO_DIGIT}({DIGIT})*))({DOT_KW})({DIGIT})*{NONZERO_DIGIT}
INTEGER_CONSTANT = {DIGIT}|{NONZERO_DIGIT}{DIGIT}*
IDENTIFIER ={LETTER}({LETTER}|{DIGIT})*

%%

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

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

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

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

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

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

	{FOR_KW} {
		System.out.println(yytext() + "\t" + "FOR_KW\t" + '-');
	}
	{IN_KW} {
		System.out.println(yytext() + "\t" + "IN_KW\t" + '-');
	}
	{REPEAT_KW} {
		System.out.println(yytext() + "\t" + "REPEAT_KW\t" + '-');
	}

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

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

{RETURN_KW} {
	System.out.println(yytext() + "\t" + "RETURN_KW\t" + '-');
}
{EXIT_KW} {
	System.out.println(yytext() + "\t" + "EXIT_KW\t" + '-');
}
{WHEN_KW} {
	System.out.println(yytext() + "\t" + "WHEN_KW\t" + '-');
}

{AND_KW} {
	System.out.println(yytext() + "\t" + "AND_KW\t" + '-');
}
{OR_KW} {
	System.out.println(yytext() + "\t" + "OR_KW\t" + '-');
}
{NOT_KW} {
	System.out.println(yytext() + "\t" + "NOT_KW\t" + '-');
}

{SEMICOLON_KW} {
	System.out.println(yytext() + "\t" + "SEMICOLON_KW\t" + '-');
}
{COLON_KW} {
	System.out.println(yytext() + "\t" + "COLON_KW\t" + '-');
}
{COMMA_KW} {
	System.out.println(yytext() + "\t" + "COMMA_KW\t" + '-');
}
{SINGLE_QUOTE_KW} {
	System.out.println(yytext() + "\t" + "SINGLE_QUOTE_KW\t" + '-');
}
{ASS_KW} {
	System.out.println(yytext() + "\t" + "ASS_KW\t" + '-');
}

{LP_KW} {
	System.out.println(yytext() + "\t" + "LP_KW\t" + '-');
}
{RP_KW} {
	System.out.println(yytext() + "\t" + "RP_KW\t" + '-');
}
{LB_KW} {
	System.out.println(yytext() + "\t" + "LB_KW\t" + '-');
}
{RB_KW} {
	System.out.println(yytext() + "\t" + "RB_KW\t" + '-');
}
{LCB_KW} {
	System.out.println(yytext() + "\t" + "LCB_KW\t" + '-');
```

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

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

{EQ_KW} {
      System.out.println(yytext() + "\t" + "EQ_KW\t" + '-');
}
{NE_KW} {
      System.out.println(yytext() + "\t" + "NE_KW\t" + '-');
}
{LE_KW} {
      System.out.println(yytext() + "\t" + "LE_KW\t" + '-');
}
{LT_KW} {
      System.out.println(yytext() + "\t" + "LT_KW\t" + '-');
}
{GE_KW} {
      System.out.println(yytext() + "\t" + "GE_KW\t" + '-');
}
{GT_KW} {
      System.out.println(yytext() + "\t" + "GT_KW\t" + '-');
}

{ADD_KW} {
      System.out.println(yytext() + "\t" + "ADD_KW\t" + '-');
}
{SUB_KW} {
      System.out.println(yytext() + "\t" + "SUB_KW\t" + '-');
}
{MUL_KW} {
      System.out.println(yytext() + "\t" + "MUL_KW\t" + '-');
}
{DIV_KW} {
      System.out.println(yytext() + "\t" + "DIV_KW\t" + '-');
}
{MOD_KW} {
      System.out.println(yytext() + "\t" + "MOD_KW\t\t" + '-');
}

{CHAR_CONSTANT} {
      System.out.println(yytext() + "\t" + "CHAR_CONSTANT\t" + "-");
}
{REAL_CONSTANT} {
      System.out.println(yytext() + "\t" + "REAL_CONSTANT\t" + "-");
}
{INTEGER_CONSTANT} {
      System.out.println(yytext() + "\t" + "INTEGER_CONSTANT\t" + "-");
}
{IDENTIFIER} {
      System.out.println(yytext() + "\t" + "IDENTIFIER\t" + "Symbol Table
Entry");
}

"\s"|"\n"|"\r"|"\t" {
}

. {
}
```

Code:

```
program folan
      real r123qwe = 3.5;
      int aaa;
      int b[2..81];
      char ch='A';
      procedure p(int c, int d[2..8]){
            real r = 0.6;
            {
                  r := r - c;
                  case 2: d[5] := 4;
                  case 3: d[5] := 6;
                  case 4: d[5] := 7;
                  default: d[5] := 2;
            }
      }
      main {
            a:=3*2-4/2;
            if a<=6 and 2<=a then
                  a := 5;
            else
                  a:=10;
            for a in 2..10 repeat
                  a:=a+1;
            p(10, b);

            aaa := 2;
            do
                  b[aaa] := aaa;
                  aaa := aaa + 1;
            while 2<aaa and aaa<=81;

            exit when b[56] <> 58;
      }
```

Lexeme Token Attribute:

Lexeme	Token	Attribute
program	PROGRAM_KW	-
folan	IDENTIFIER	Symbol Table Entry
real	REAL_KW	-
r123qwe	IDENTIFIER	Symbol Table Entry
=	EQ_KW	-
3.5	REAL_CONSTANT	-
;	SEMICOLON_KW	-
int	INTEGER_KW	-
aaa	IDENTIFIER	Symbol Table Entry
;	SEMICOLON_KW	-
int	INTEGER_KW	-
b	IDENTIFIER	Symbol Table Entry
[	LB_KW	-
2	INTEGER_CONSTANT	-
..	TWO_DOTS_KW	-
81	INTEGER_CONSTANT	-
]	RB_KW	-
;	SEMICOLON_KW	-
char	CHAR_KW	-
ch	IDENTIFIER	Symbol Table Entry
=	EQ_KW	-
'A'	CHAR_CONSTANT	-
;	SEMICOLON_KW	-
procedure	PROCEDURE_KW	-
p	IDENTIFIER	Symbol Table Entry
(	LP_KW	-
Int	INTEGER_KW	-
c	IDENTIFIER	Symbol Table Entry
,	COMMA_KW	-

int	INTEGER_KW	-
d	IDENTIFIER	Symbol Table Entry
[	LB_KW	-
2	INTEGER_CONSTANT	-
..	TWO_DOTS_KW	-
8	INTEGER_CONSTANT	-
]	RB_KW	-
)	RP_KW	-
{	LCB_KW	-
real	REAL_KW	-
r	IDENTIFIER	Symbol Table Entry
=	EQ_KW	-
0.6	REAL_CONSTANT	-
;	SEMICOLON_KW	-
{	LCB_KW	-
r	IDENTIFIER	Symbol Table Entry
:=	ASS_KW	-
r	IDENTIFIER	Symbol Table Entry
-	SUB_KW	-
c	IDENTIFIER	Symbol Table Entry
;	SEMICOLON_KW	-
case	CASE_KW	-
2	INTEGER_CONSTANT	-
:	COLON_KW	-
d	IDENTIFIER	Symbol Table Entry
[	LB_KW	-
5	INTEGER_CONSTANT	-
]	RB_KW	-
:=	ASS_KW	-
4	INTEGER_CONSTANT	-
;	SEMICOLON_KW	-

case	CASE_KW	-
3	INTEGER_CONSTANT	-
:	COLON_KW	-
d	IDENTIFIER	Symbol Table Entry
[	LB_KW	-
5	INTEGER_CONSTANT	-
]	RB_KW	-
:=	ASS_KW	-
6	INTEGER_CONSTANT	-
;	SEMICOLON_KW	-
case	CASE_KW	-
4	INTEGER_CONSTANT	-
:	COLON_KW	-
d	IDENTIFIER	Symbol Table Entry
[	LB_KW	-
5	INTEGER_CONSTANT	-
]	RB_KW	-
:=	ASS_KW	-
7	INTEGER_CONSTANT	-
;	SEMICOLON_KW	-
default	DEFAULT_KW	-
:	COLON_KW	-
d	IDENTIFIER	Symbol Table Entry
[	LB_KW	-
5	INTEGER_CONSTANT	-
]	RB_KW	-
:=	ASS_KW	-
2	INTEGER_CONSTANT	-
;	SEMICOLON_KW	-
}	RCB_KW	-
}	RCB_KW	-

main	MAIN_KW	-
{	LCB_KW	-
a	IDENTIFIER	Symbol Table Entry
:=	ASS_KW	-
3	INTEGER_CONSTANT	-
*	MUL_KW	-
2	INTEGER_CONSTANT	-
-	SUB_KW	-
4	INTEGER_CONSTANT	-
/	DIV_KW	-
2	INTEGER_CONSTANT	-
;	SEMICOLON_KW	-
if	IF_KW	-
a	IDENTIFIER	Symbol Table Entry
<=	LE_KW	-
6	INTEGER_CONSTANT	-
and	AND_KW	-
2	INTEGER_CONSTANT	-
<=	LE_KW	-
a	IDENTIFIER	Symbol Table Entry
then	THEN_KW	-
a	IDENTIFIER	Symbol Table Entry
:=	ASS_KW	-
5	INTEGER_CONSTANT	-
;	SEMICOLON_KW	-
else	ELSE_KW	-
a	IDENTIFIER	Symbol Table Entry
:=	ASS_KW	-
10	INTEGER_CONSTANT	-
;	SEMICOLON_KW	-
for	FOR_KW	-

a	IDENTIFIER	Symbol Table Entry
in	IN_KW	-
2	INTEGER_CONSTANT	-
..	TWO_DOTS_KW	-
10	INTEGER_CONSTANT	-
repeat	REPEAT_KW	-
a	IDENTIFIER	Symbol Table Entry
:=	ASS_KW	-
a	IDENTIFIER	Symbol Table Entry
+	ADD_KW	-
1	INTEGER_CONSTANT	-
;	SEMICOLON_KW	-
p	IDENTIFIER	Symbol Table Entry
(	LP_KW	-
10	INTEGER_CONSTANT	-
,	COMMA_KW	-
B	IDENTIFIER	Symbol Table Entry
)	RP_KW	-
;	SEMICOLON_KW	-
aaa	IDENTIFIER	Symbol Table Entry
:=	ASS_KW	-
2	INTEGER_CONSTANT	-
;	SEMICOLON_KW	-
do	DO_KW	-
b	IDENTIFIER	Symbol Table Entry
[	LB_KW	-
aaa	IDENTIFIER	Symbol Table Entry
]	RB_KW	-
:=	ASS_KW	-
Aaa	IDENTIFIER	Symbol Table Entry
;	SEMICOLON_KW	-

aaa	IDENTIFIER	Symbol Table Entry
:=	ASS_KW	-
aaa	IDENTIFIER	Symbol Table Entry
+	ADD_KW	-
1	INTEGER_CONSTANT	-
;	SEMICOLON_KW	-
while	WHILE_KW	-
2	INTEGER_CONSTANT	-
<	LT_KW	-
aaa	IDENTIFIER	Symbol Table Entry
and	AND_KW	-
aaa	IDENTIFIER	Symbol Table Entry
<=	LE_KW	-
81	INTEGER_CONSTANT	-
;	SEMICOLON_KW	-
exit	EXIT_KW	-
when	WHEN_KW	-
b	IDENTIFIER	Symbol Table Entry
[	LB_KW	-
56	INTEGER_CONSTANT	-
]	RB_KW	-
<>	NE_KW	-
58	INTEGER_CONSTANT	-
;	SEMICOLON_KW	-
}	RCB_KW	-