Text

Description automatically generated

**T.Y.B.Tech (CSE)**

[System Software and Compilers(SSC)](https://mitwpu.instructure.com/courses/3210)

**Lab Assignment No – 5**

**Name: Aniruddha Shende**

**Roll number: PD-05**

**Batch: D1**

**Panel: D**

Text, letter

Description automatically generated

A piece of paper with writing

Description automatically generated with low confidence

Text, letter

Description automatically generated

Text, letter

Description automatically generated

**Scanner for Java Code:**

%{

#include<string.h>

int i,m;

struct symtab

{

char name[200];

char type[200];

}sym[20];

FILE\* yyin;

int checksymtab(char \*temp);

%}

ws [\t ]

digit [0-9]

alpha [a-zA-Z]

id {alpha}({alpha}|{digit})\*

aspec "public"|"private"|"protected"

stat "static"

sc ";"

%%

import{ws}{alpha}+(.{alpha}+)\*(.\*?){sc} {printf("\n%s : Preprocessor directive to include header files",yytext);}

if |

case |

else |

while |

do |

switch{ws}\*\(.\*\) |

for |

static |

class {printf("\n %s : Keyword",yytext);}

{aspec} {printf("\n %s : Access Specifier", yytext);}

System\.out\.println{ws}\*\(\".\*\"\){sc} |

System\.out\.print{ws}\*\(\".\*\"\){sc} {printf("\n %s : Library Function",yytext);}

\/\/{ws}\*{alpha}\*({ws}\*{alpha}\*)\* {printf("\n %s : This is single line comment", yytext);}

public{ws}+static{ws}+void{ws}+main\(String{ws}+{id}"[]"\) {printf("\n%s : Defination of JAVA main function",yytext);}

{aspec}{ws}+{stat}\*{ws}+void{ws}+{id}{ws}\*\(({ws}\*{id}{ws}\*,)\*({ws}\*{id}{ws}\*)?\) {

printf("\n%s : Definition of function with return type void" ,yytext);

if(checksymtab(yytext) == 0) {

strcpy(sym[m].name,yytext);

strcpy(sym[m].type,"Function");

m++;

}

}

{aspec}{ws}+{stat}\*{ws}+int{ws}+{id}{ws}\*\(({ws}\*{id}{ws}\*,)\*({ws}\*{id}{ws}\*)?\) {printf("\n%s : Definition of function with return type int" ,yytext);}

{aspec}{ws}+{stat}\*{ws}+float{ws}+{id}{ws}\*\(({ws}\*{id}{ws}\*,)\*({ws}\*{id}{ws}\*)?\) {printf("\n%s : Definition of function with return type char" ,yytext);}

{aspec}{ws}+{stat}\*{ws}+char{ws}+{id}{ws}\*\(({ws}\*{id}{ws}\*,)\*({ws}\*{id}{ws}\*)?\) {printf("\n%s : Definition of function with return type float" ,yytext);}

{aspec}{ws}+{stat}\*{ws}+double{ws}+{id}{ws}\*\(({ws}\*{id}{ws}\*,)\*({ws}\*{id}{ws}\*)?\) {printf("\n%s : Definition of function with return type double" ,yytext);}

{id}{ws}\*\(({ws}\*{id}{ws}\*,)\*({ws}\*{id}{ws}\*)?\) {printf("\n%s : Function call" ,yytext);}

int{ws}+ {printf("\n%s : Declaration of integer variables",yytext);}

char{ws}+ {printf("\n%s : Declaration of character variables",yytext);}

float{ws}+ {printf("\n%s : Declaration of float variables",yytext);}

double{ws}+ {printf("\n%s : Declaration of double variables",yytext);}

boolean{ws}+ {printf("\n%s : Declaration of boolean variables",yytext);}

String{ws}+ {printf("\n%s : Declaration of string variables",yytext);}

byte{ws}+ {printf("\n%s : Declaration of byte variables",yytext);}

short{ws}+ {printf("\n%s : Declaration of short variables",yytext);}

[;,] {}

[{(] {printf("\n %s : Opening brace",yytext);}

[})] {printf("\n %s : Closing brace",yytext);}

[+\-\\*\\%=] {printf("\n %s : Arithmetic operator",yytext);}

[<>]=? {printf("\n %s : Relational operator",yytext);}

{id} {

printf("\n %s : Identifier",yytext);

if(checksymtab(yytext) == 0) {

strcpy(sym[m].name,yytext);

strcpy(sym[m].type,"Identifier");

m++;

}

}

{digit}\*(\.{digit}\*)? {

printf("\n %s : Arithmetic number",yytext);

if(checksymtab(yytext) == 0) {

strcpy(sym[m].name,yytext);

strcpy(sym[m].type,"Number");

m++;

}

}

{id}"["{digit}\*"]" {

printf("\n %s : Array",yytext);

if(checksymtab(yytext) == 0) {

strcpy(sym[m].name,yytext);

strcpy(sym[m].type,"Array");

m++;

}

}

%%

int checksymtab(char \*temp) {

for(int i = 0; i < 20; i++)

if(strcmp(sym[i].name, temp) == 0) return 1;

return 0;

}

int main(int argc,char\* argv[])

{

/\* printf("Enter the String: ");

yylex();\*/

yyin=fopen(argv[1],"r");

yylex();

fclose(yyin);

printf("\n\n\n\t--------------------\n");

printf("\tIndex\tSymbol name\t\t\tSymbol Type");

printf("\t\t\n--------------------------\n");

for(i=0;i<m;i++)

{

printf("\n\t%d",i+1);

printf("\t\t%s",sym[i].name);

printf("\t\t\t%s\n",sym[i].type);

}

yywrap();

}

extern int yywrap()

{

return 1;

}

**Output of the program:**

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence