Practical no: 6

Problem Statement: Write a program using Lex specifications to implement lexical analysis Phase of compiler to count no. of words, lines and characters of given Input file.

Name: Aditi Dinesh Mulay

Class: T.E. Computer

Subject: SPOS

Div: A

Roll no: 02

PRN No. 71918146B

SPOS

Assignment B-3

Aditi Dinesh Mulay
T.E. Comp Div: A
Roll no: 02

Aim: Design lex program to count no. of words, lines, and characters of given input file.

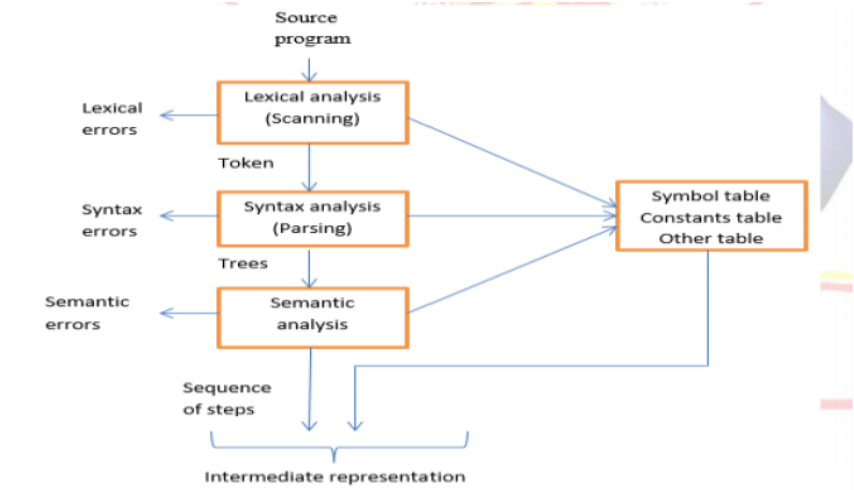
Problem statement:- Write a program using lex specifications to implement lexical analysis Phase of compiler to count no. of words, lines and characters of given Input file.

Software Requirements: o/s - Ubuntu Kylin
S/w name - LEX Tool (flex)

Objectives:
1) To understand LEX Concepts.
2) To implement LEX program for no's of count.
3) To study about Lex & Java.
4) To know important about Lexical Analyzer.

How the INPUT is MATCHED?

When the generated scanner is run, it analyzes its input looking for strings, which match any of its patterns. If it finds more than one match, it takes the one matching the most text. If it finds two or more matches of same length, the rule listed first in the flex input file is chosen. Once the matched is determ the next corresponding to match is made available in the global character pointer "yytext", and its length in the global integer "yyleng". The action corresponding to the matched pattern is then executed & then the remaining i/p is scanned for another match. If no match is

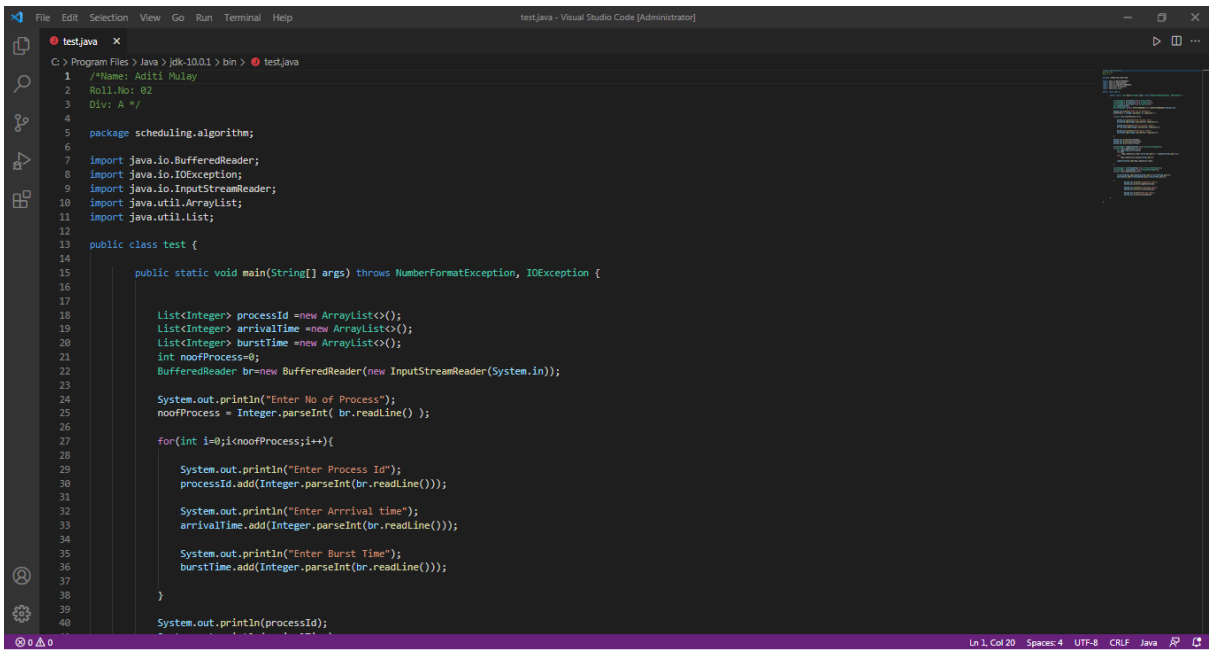


Front end of toy compiler

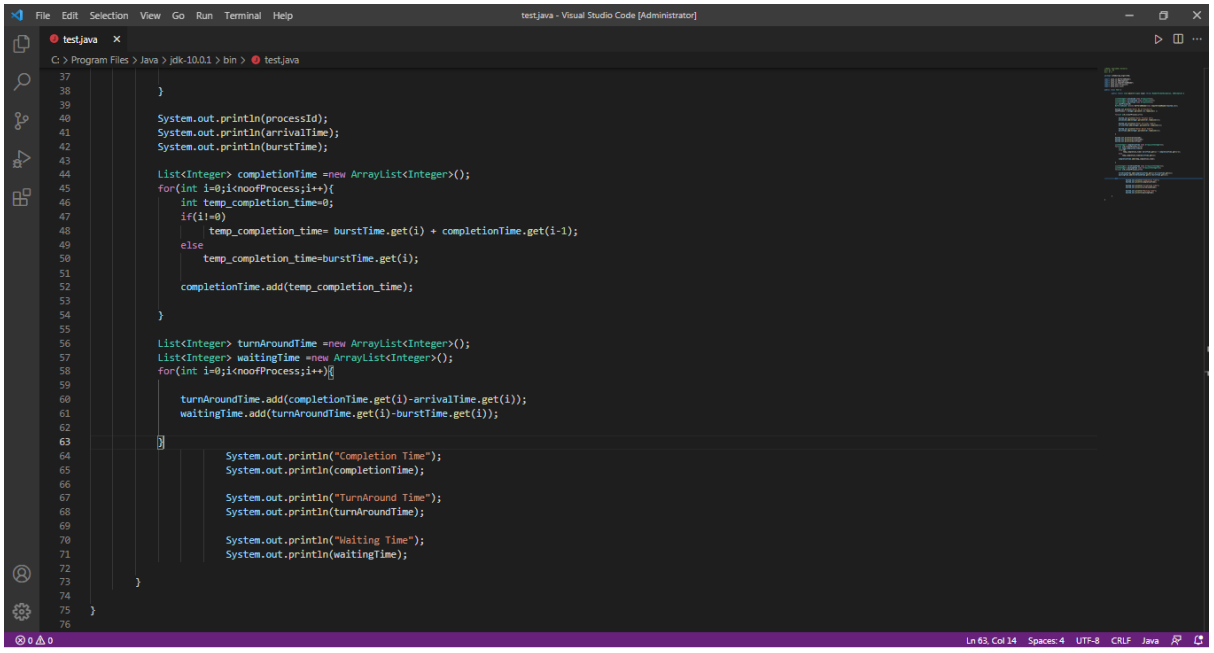
found, then the default rule is executed: the next character in the i/p is associated matched & copied to the standard output.

Conclusion:

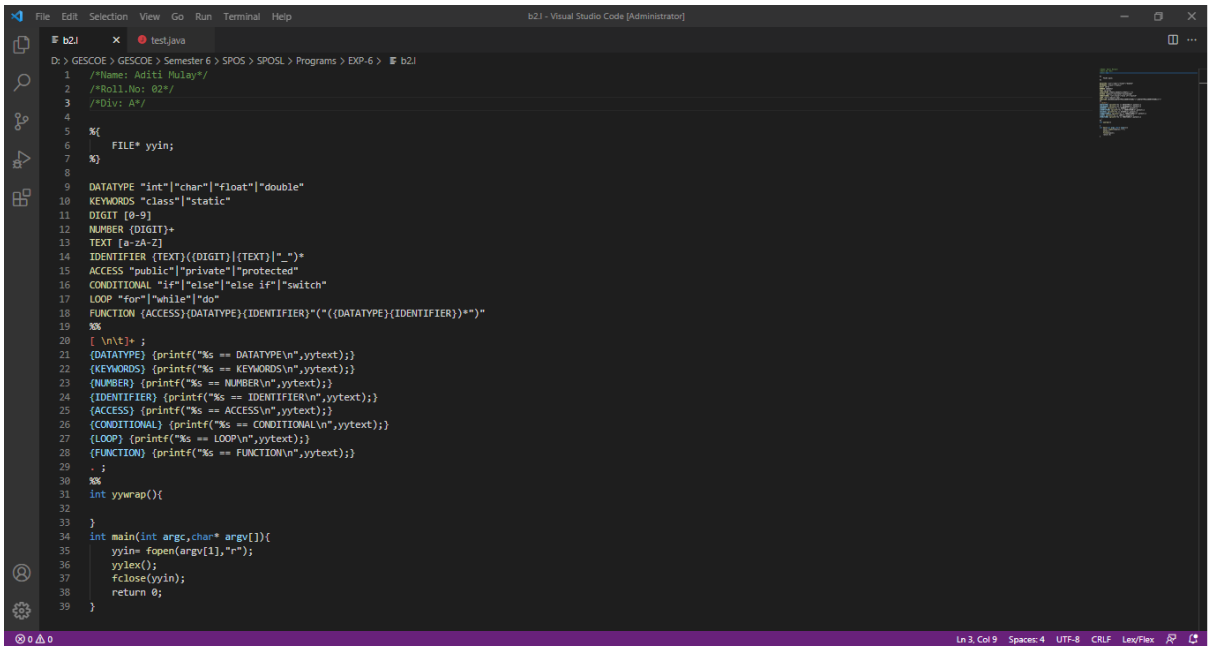
Thus, we have studied lexical analyzer & implemented an application for lexical analyzer to count total number of words, chards & line etc.



```java
/*Name: Aditi Mulay
Roll.No: 02
Div: A */

package scheduling.algorithm;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.List;

public class test {

	public static void main(String[] args) throws NumberFormatException, IOException {


		List<Integer> processId =new ArrayList<>();
		List<Integer> arrivalTime =new ArrayList<>();
		List<Integer> burstTime =new ArrayList<>();
		int noofProcess=0;
		BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

		System.out.println("Enter No of Process");
		noofProcess = Integer.parseInt( br.readLine() );

		for(int i=0;i<noofProcess;i++){

			System.out.println("Enter Process Id");
			processId.add(Integer.parseInt(br.readLine()));

			System.out.println("Enter Arrrival time");
			arrivalTime.add(Integer.parseInt(br.readLine()));

			System.out.println("Enter Burst Time");
			burstTime.add(Integer.parseInt(br.readLine()));

		}

		System.out.println(processId);
```



```java
		}

		System.out.println(processId);
		System.out.println(arrivalTime);
		System.out.println(burstTime);

		List<Integer> completionTime =new ArrayList<Integer>();
		for(int i=0;i<noofProcess;i++){
			int temp_completion_time=0;
			if(i!=0)
				temp_completion_time= burstTime.get(i) + completionTime.get(i-1);
			else
				temp_completion_time=burstTime.get(i);

			completionTime.add(temp_completion_time);

		}

		List<Integer> turnAroundTime =new ArrayList<Integer>();
		List<Integer> waitingTime =new ArrayList<Integer>();
		for(int i=0;i<noofProcess;i++){

			turnAroundTime.add(completionTime.get(i)-arrivalTime.get(i));
			waitingTime.add(turnAroundTime.get(i)-burstTime.get(i));

		}
				System.out.println("Completion Time");
				System.out.println(completionTime);

				System.out.println("TurnAround Time");
				System.out.println(turnAroundTime);

				System.out.println("Waiting Time");
				System.out.println(waitingTime);

	}
}
```


File Edit Selection View Go Run Terminal Help
b2.l - Visual Studio Code [Administrator]
b2.l
test.java
D: > GESCOE > GESCOE > Semester 6 > SPOS > SPOSL > Programs > EXP-6 > b2.l
/*Name: Aditi Mulay*/
/*Roll.No: 02*/
/*Div: A*/
%{
FILE* yyin;
%}
DATATYPE "int"|"char"|"float"|"double"
KEYWORDS "class"|"static"
DIGIT [0-9]
NUMBER {DIGIT}+
TEXT [a-zA-Z]
IDENTIFIER {TEXT}({DIGIT}|{TEXT}|"_")*
ACCESS "public"|"private"|"protected"
CONDITIONAL "if"|"else"|"else if"|"switch"
LOOP "for"|"while"|"do"
FUNCTION {ACCESS}{DATATYPE}{IDENTIFIER}"("({DATATYPE}{IDENTIFIER})*")"
%%
[\n\t]+ ;
{DATATYPE} {printf("%s == DATATYPE\n",yytext);}
{KEYWORDS} {printf("%s == KEYWORDS\n",yytext);}
{NUMBER} {printf("%s == NUMBER\n",yytext);}
{IDENTIFIER} {printf("%s == IDENTIFIER\n",yytext);}
{ACCESS} {printf("%s == ACCESS\n",yytext);}
{CONDITIONAL} {printf("%s == CONDITIONAL\n",yytext);}
{LOOP} {printf("%s == LOOP\n",yytext);}
{FUNCTION} {printf("%s == FUNCTION\n",yytext);}
. ;
%%
int yywrap(){
}
int main(int argc,char* argv[]){
yyin= fopen(argv[1],"r");
yylex();
fclose(yyin);
return 0;
}
Ln 3, Col 9 Spaces: 4 UTF-8 CRLF Lex/Flex