System Programming and Operating Systems Lab

ASSIGNMENT 8

Name: Shrirang Mhalgi
Roll No: 322008
Batch: B 1

1 Date of Completion:

01/03/2019

2 Aim:

To design suitable data structure and implement pass-1 of two pass macro-processor using OOP features in Java.

3 Objectives:

To design pass-1 of two pass macro-processor.

4 Theory:

A macro processor is a program that copies a stream of text from one place to another, making a systematic set of replacements as it does so. Macro processors are often embedded in other programs, such as assemblers and compilers. Sometimes they are standalone programs that can be used to process any kind of text.

A macro instruction (macro) is a notational convenience for the programmer. It allows the programmer to write shorthand version of a program (module programming). A Macro represents a commonly used group of statements in the source programming language.A macro instruction (macro) is a notational convenience for the programmer. It allows the programmer to write shorthand version of a program (module programming)

The macro processor replaces each macro instruction with the corresponding group of source language statements (expanding) Normally, it performs no analysis of the text it handles.It does not concern the meaning of the involved statements during macro expansion. The design of a macro processor generally is machine independent.

Two new assembler directives are used in macro definition

MACRO: identify the beginning of a macro definition
MEND: identify the end of a macro definition

Usually in the Pass 1 of the two pass Macro-Processor,Macros are defined and in the second pass of the macro-processor,macros are expanded.Data Structures required in the Pass 1 of the

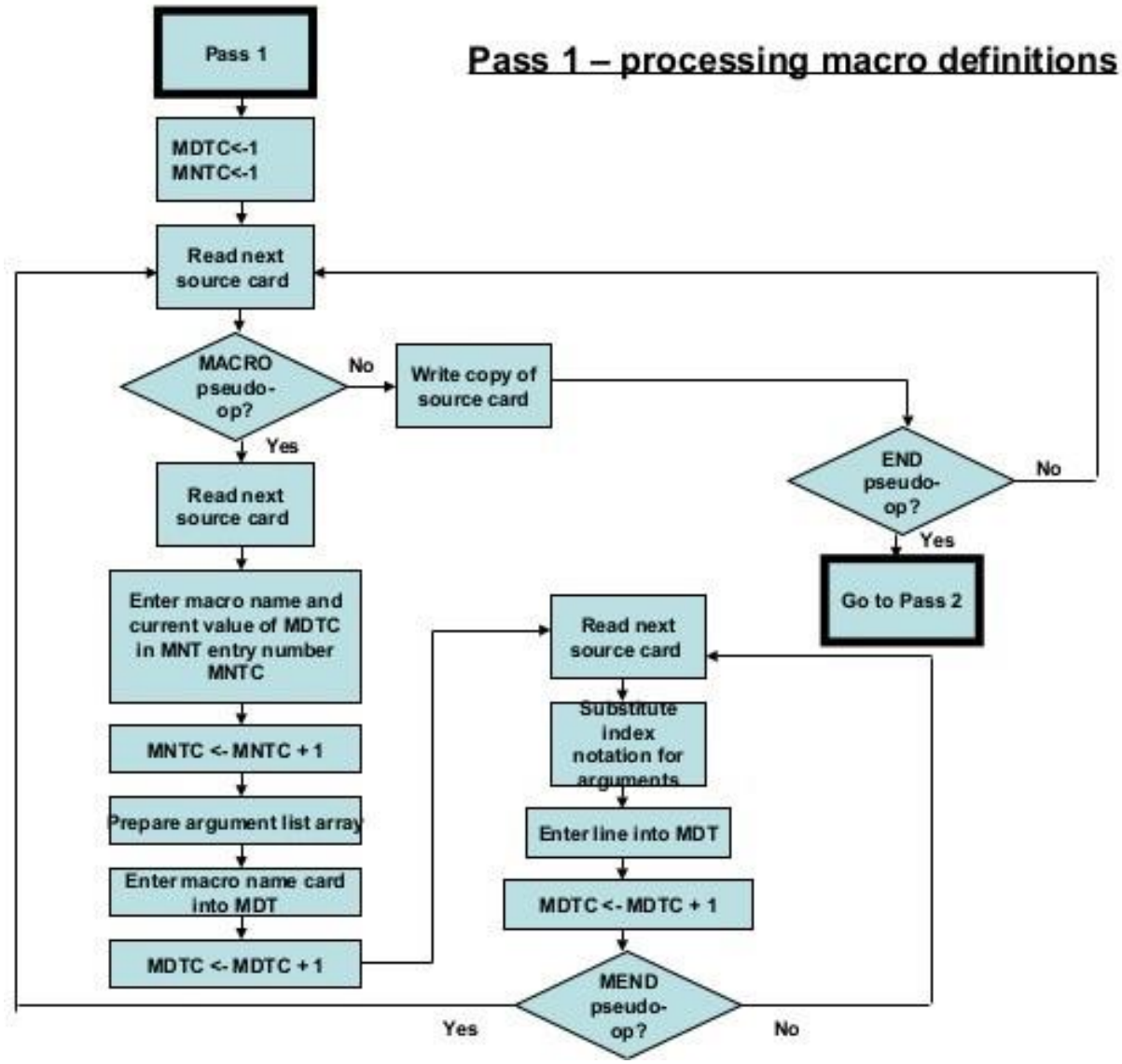
Macro-Processor are as follows:-

1. Input Source File
2. Intermediate File
3. MNT (Macro Name Table)
4. MDT (Macro Defination Table)
5. MNT Pointer
6. MDT Pointer

5 Algorithm:

1. begin macro processor
2. EXPANDING : = FALSE
3. while OPCODE END do
4. begin
5. GETLINE
6. PROCESSLINE
7. end while
8. end macro processor
9. procedure PROCESSLINE
10. begin
11. search MNT for OPCODE
12. if found then
13. EXPAND
14. else if OPCODE = MACRO then
15. DEFINE
16. else write source line to expanded file
17. end PROCESSLINE

6 Flowchart:


Pass 1 – processing macro definitions
Pass 1
MDTC<-1
MNTC<-1
Read next source card
MACRO pseudo-op?
No
Write copy of source card
Yes
Read next source card
END pseudo-op?
No
Yes
Go to Pass 2
Enter macro name and current value of MDTC in MNT entry number MNTC
MNTC <- MNTC + 1
Prepare argument list array
Enter macro name card into MDT
MDTC <- MDTC + 1
Read next source card
Substitute index notation for arguments
Enter line into MDT
MDTC <- MDTC + 1
MEND pseudo-op?
Yes
No

7 Code:

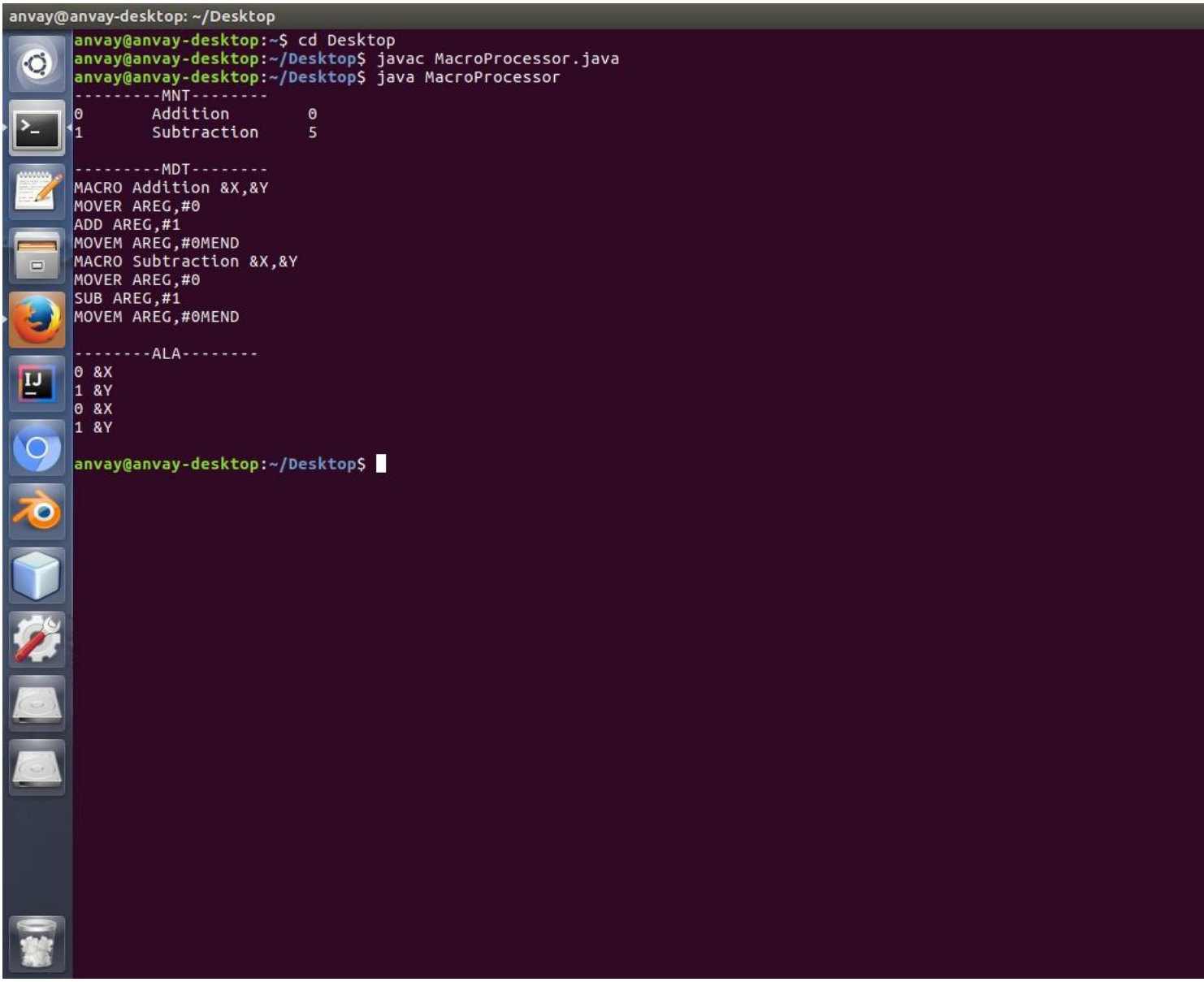
```
;-----------

import java.io.*;
import java.lang.String;
import java.util.Scanner;


public class MacroProcessor
{
static int mdtp=0,mntp=0;

public static void main(String []args)throws Exception
{
File f=new File("program.txt");
Scanner s=new Scanner(f);

String tokens[];                    // Array for storing tokens generated

String MDT_Def,macro_def_line;

MNT[] mnt_obj=new MNT[5];          // Array of objects for MNT
MDT[] mdt_obj=new MDT[5];                  // Array of objects for MDT

String[] ALA;                      // Data member of MNT Class

while(s.hasNextLine())
{
macro_def_line=s.nextLine();              //read line in macro_def_line variable
tokens=macro_def_line.split(" ");         //split the line on space

for(int i=0;i<tokens.length;i++)
{
if(tokens[i].equals("MACRO"))          //if macro
{
//-----MNT----
mnt_obj[mntp]=new MNT(mdtp,mntp);       //create MNT object
mnt_obj[mntp].name=tokens[i+1];            // name of macro

// mnt_obj[mntp].print();                  // print MNT Contents

//---------MDT-------------------------------
MDT_Def=macro_def_line;                 //store line in MDT Def

while(true)                              //read until MEND not found
```

```
{
String x=s.nextLine();                    //read macro defintion line byline

if(x.equals("MEND"))
{
MDT_Def=MDT_Def.concat(x);      //MDT also contains MEND statement
mdtp++;                                   //increment mdtp for next macro definition
break;
}

MDT_Def=MDT_Def+"¥n";                    //add new line to MDT contents
MDT_Def=MDT_Def.concat(x);               //put next statement of macro definition

mdtp++;

}

if(MDT_Def.contains("X"))                        //Replace argument with #1 ,#2.....
MDT_Def=MDT_Def.replaceAll("X","#0");

if(MDT_Def.contains("Y"))
MDT_Def=MDT_Def.replaceAll("Y","#1");

MDT_Def=MDT_Def.replaceFirst("#0","X"); //Macro name line contains parameters a
is
MDT_Def=MDT_Def.replaceFirst("#1","Y");

mdt_obj[mntp]=new MDT(MDT_Def);                  //create mdt object and store contents
//mdt_obj[mntp].print();                         // print mdt contents

//———ALA———
ALA=tokens[i+2].split(",");                      // arguments at 3rd index of token arra
mnt_obj[mntp].setALA(ALA);                       // setALA for each macro

mntp++;
}

}

}
print(mnt_obj,mdt_obj);
s.close();
}

static void print(MNT[] obj,MDT[] obj1)
{
System.out.println("————MNT————");
```

```
for(int  i=0; i <mntp ; i++)
obj[i].print();

System.out.println();
System.out.println("----MDT----------");
for(int  i=0;i<mntp; i++)
objl[i].print();

System.out.println();
System.out.println("- ---- ALA----------");
for(int  i=0;i<mntp; i++)
obj[i].printALA();

System.out.println();


}


}

class MNT
{
int no , address ; S
tring name ;
String [] ALA;

MNT( int mdtp , int mntp)
{
no=mntp ;
if( mntp==0)
address=mdtp;              //First Macro will be at 0th position in MDT
else
address=mdtp+1;
name="";
}

public void print()
{
System.out.print(this.no + "¥t"+this.name + "¥t" + this.address);
System.out.println();
}

public void setALA(String[] obj)
{
this.ALA=obj;
}
```

```
public void printALA ()
{
System.out.println("0 " + this.ALA[0]);
System.out.println("1 " + this.ALA[1]);

}

}

class MDT
{
String def;

MDT(String defintion)
{
this.def=defintion;
}

public void print()
{
System.out.print(this.def);
System.out.println();
}
}
```

8 Output:



9 Conclusion:

Through this assignment we understood the working of Pass 1 of Macro-Processor in which the macro is defined in the form of 2 major data structures which are MNT(Macro Name Table) and MDT(Macro Defination Table).