CECS 524

CSULB ID: 016131932

Assignment 2

SIL PROGRAM

ANTLR WORKS :

```
grammar simplegrammar;


@header {
import java.util.HashMap;
import java.util.ArrayList;
import java.util.Scanner; }



@members
{


HashMap memory = new HashMap();
private ArrayList<String> ExistingVarList = new ArrayList<String>();
private boolean variableDefined(String strVarName)
{
return ExistingVarList.contains(strVarName);
}
private void AddVariable(String strVarName)
{
ExistingVarList.add(strVarName);
}
Scanner sc = new Scanner(System.in);
}


prog:stat+  ;
```

```
stat:expr NEWLINE {System.out.println($expr.value);}
|  'LET' ID'='expr NEWLINE
{
if(memory.containsKey($ID.text))
{
try
{
memory.put($ID.text, $expr.value);
}
catch(Exception ex)
{
System.err.println("Invalid Input");
}
}
else
System.err.println("Undefined local variable "+$ID.text);


}
| 'PRINT' STRING NEWLINE{String literal =$STRING.text;
System.out.print(literal.substring(1,literal.length()-1));
}
| 'PRINT' expr { System.out.print($expr.value);}
| 'PRINTLN' STRING
NEWLINE {String literal =$STRING.text;
System.out.println(literal.substring(1,literal.length()-1));
}
| 'PRINTLN' expr { System.out.println($expr.value);}
| 'INTEGER' variable(','variable)* NEWLINE
|  'INPUT' iden(','iden)* NEWLINE
| 'END'
{
```

```
System.exit(0);

}


;

/*

PRINT     :    'PRINT''('

expr {System.out.print($e.value);}

|STRING

{

System.out.print($STRING.text);

}

')'

;

*/

expr returns [int value]

:  e=multExpr {$value = $e.value;}

(  '+' e=multExpr {$value += $e.value;  }

|  '-' e=multExpr {$value -= $e.value;}

)*

;




multExpr returns [int value]

:  e=atom {$value = $e.value;}

(  '*' e=atom {$value *= $e.value;  }

|  '/' e=atom {$value /= $e.value;}

)*

;


atom returns [int value]

:  INTEGER {$value = Integer.parseInt($INTEGER.text);}
```

```
|  ID

{

Integer v = (Integer)memory.get($ID.text);

if ( v!=null ) $value = v.intValue();

else System.err.println("undefined variable "+$ID.text);

}

|  '(' expr ')' {$value = $expr.value;}

;



iden : ID {

if(variableDefined($ID.text)){

System.out.println("Value "+$ID.text);

}else

{

Integer value=sc.nextInt();

memory.put($ID.text, new Integer(value));

}

};


variable

:ID

{if(memory.containsKey($ID.text)){


System.err.println("Duplicate Variable "+$ID.text);

}

else

{

memory.put($ID.text, new Integer(0));}

//AddVariable($ID.text);

};
```

```
ID  :('a'..'z'|'A'..'Z'|'_') ('a'..'z'|'A'..'Z'|'0'..'9'|'_')*

;

INTEGER :    '0'..'9'+

;

NEWLINE:'\r'? '\n' ;


WS  :  (' '

|'\t')

+

{

skip();

}

;


STRING

:  '"' ( ESC_SEQ | ~('\\'|'"') )* '"'

;


ESC_SEQ

:  '\\' ('b'|'t'|'n'|'f'|'r'|'\"'|'\''|'\\')

;



COMMENT

:  '//' ~('\n'|'\r')* '\r'? '\n'{$channel=HIDDEN;}

;


SIL.java :


import java.io.File;
import java.io.IOException;

import org.antlr.runtime.ANTLRFileStream;

import org.antlr.runtime.CommonTokenStream;

import org.antlr.runtime.RecognitionException;

public class SIL {

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

RecognitionException {

/*simplegrammarLexer lexer =

new simplegrammarLexer(new ANTLRFileStream(args[0]));

CommonTokenStream tokens = new CommonTokenStream(lexer);

simplegrammarParser parser = new simplegrammarParser(tokens);

parser.program();*/

String pathname="H:/input1.s";

File file1=new File(pathname);

String name_of_file=file1.getName();

String extension_of_file=name_of_file.substring(name_of_file.lastIndexOf('.')+1);

if(extension_of_file.toLowerCase().equalsIgnoreCase("s"))

{

simplegrammarLexer lexer =new simplegrammarLexer(new ANTLRFileStream(pathname));

CommonTokenStream tokens=new CommonTokenStream(lexer);

simplegrammarParser parser=new simplegrammarParser(tokens);

parser.prog();

}

else

{

System.out.println("\nFile does not end with extension '.s'\nInvalid Extension!");

}


}

}
```

OUTPUTS:

1.Input :

```
PRINTLN "Hello, world!"

END
```

Output:

```
Hello, world!
```

2.Input:

```
INTEGER A

PRINT "Enter A:"

INPUT A

PRINT "A="

PRINTLN A

END
```

Output:

```
Enter A:12
A=12
```

3.Input:

```
//This program calculates the area of rectangle
PRINTLN "Calculate the area of rectangle"
INTEGER L, W, AREA
PRINT "Enter length and width:"
INPUT L,W
PRINT "Area is "
LET AREA = L * W
PRINTLN AREA
END
```

Output:

```
Calculate the area of rectangle
Enter length and width:10
10
Area is 100
```

4.Input

```
PRINTLN "Calculate Payroll - Double Pay Overtime"
PRINT "Enter rate of pay:"
INTEGER rate, hours, overtime_hours, netpay
INPUT rate
PRINT "Enter hours up to 40:"
INPUT hours
PRINT "Enter overtime hours:"
INPUT overtime_hours
LET netpay = rate * hours + rate * overtime_hours * 2
PRINT "Your net pay = "
PRINTLN netpay
END
```

Output:

```
Calculate Payroll - Double Pay Overtime
Enter rate of pay:10
Enter hours up to 40:20
Enter overtime hours:10
```

```
Your net pay = 400
```

5.Input:

```
//Calculate the area of a triangle
PRINTLN "CALCULATE THE AREA OF A TRIANGLE"
PRINT "ENTER BASE:"
INTEGER BASE, HEIGHT, AREA
INPUT BASE
PRINT "ENTER HEIGHT:"
INPUT HEIGHT
LET AREA = (BASE * HEIGHT) / 2
PRINT "AREA = "
PRINTLN AREA
END
```

Output:

```
CALCULATE THE AREA OF A TRIANGLE
ENTER BASE:10
ENTER HEIGHT:10
AREA = 50
```