**CECS 524 ASSIGNMENT 8-2**

Name:Spuritha Mudireddy

Student ID: 030743269

**Code:**

import java.util.\*;

public class Expression

{

public static class Line\_Memory{

int lineNumber;

String line;

}

private static ArrayList<Line\_Memory> program=new ArrayList<Line\_Memory>() ; //the entire SIL program is in this array

private static int curr\_line; //the current line that is executing

private static boolean mEOL, mEOF;

public static HashMap<String,Integer> memory=new HashMap<String,Integer>();

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

String in;

int k=0;

while(sc.hasNextLine())

{

in=sc.nextLine().toUpperCase();

Expression.Line\_Memory lm=new Expression.Line\_Memory();

lm.lineNumber=Integer.parseInt(in.substring(0, in.indexOf(' ')));

lm.line=in.substring(in.indexOf(' ')+1);

program.add(lm);

if(lm.line.equals("END"))

{

parseProgram();

}

}

}

public static void parseProgram()

{

int end\_line=program.size();

int i=0,low,high;

while(i!=end\_line)

{

curr\_line=program.get(i).lineNumber;

int x=parse(program.get(i).line,curr\_line);

if(x==curr\_line)

{

i++;

}

else if(x<curr\_line)

{

low=0;

high=i;

i=search(x, low,high);

curr\_line=program.get(i).lineNumber;

}

else

{

low=i;

high=program.size()-1;

i=search(x, low,high);

curr\_line=program.get(i).lineNumber;

}

}

}

public static void takeInput(String [] vars)

{

Scanner sc = new Scanner(System.in);

String p=sc.nextLine();

String [] nums=p.split(" ");

int [] values=new int[nums.length];

//for(int i=0;i<nums.length;i++)

for(int i=0;i<nums.length;i++)

values[i]=Integer.parseInt(nums[i]);

if(vars.length!=values.length||values.length==0)

System.err.println("Line n missing input value");

else {

for (int i = 0; i < vars.length; i++) {

memory.put(vars[i], values[i]);

}

}

}

public static int parse(String in,int ln)

{

in.trim();

if(in.equals("END"))

{

System.exit(0);

}

String ins = in.substring(0, in.indexOf(' '));

String str = in.substring(in.indexOf(' ') + 1);

if(ins.equals("LET"))

{

memory.put(toString(str.charAt(0)),expr(str.substring(2)));

}

if(ins.equals("INPUT"))

{

takeInput(str.split(","));

}

if(ins.equals("GOTO"))

{

return Integer.parseInt(str);

}

if(ins.equals("IF"))

{

String condition = str.split("THEN")[0];

String action= in.split("THEN")[1];

String op="";

if(condition.contains("="))

op="=";

else if(condition.contains("<"))

op="<";

else if(condition.contains(">"))

op=">";

else if(condition.contains("!"))

op="!";

int x= calculate(expr(condition.split(op)[0]),expr(condition.split(op)[1]),op);

if(x==1)

{

int p= parse(action.trim(),curr\_line);

return p;

}

return curr\_line;

}

if(ins.equals("INTEGER"))

{

String [] variables=str.split(",");

for(int i=0;i<variables.length;i++)

{

memory.put(variables[i],0);

}

}

if(ins.equals("PRINTLN"))

{

if(containsOperand(str)||memory.containsKey(str))

System.out.println(atom(str));

else

System.out.println(str.substring(1,str.length()-1));

}

if(ins.equals("PRINT"))

{

if(containsOperand(str)||memory.containsKey((str)))

System.out.print(atom(str));

else

System.out.print(str.substring(1,str.length()-1));

}

return curr\_line;

}

public static int search(int target,int low,int high)

{

int mid=low + ((high - low) / 2);

while(low<=high)

{

mid=low + ((high - low) / 2);

if(program.get(mid).lineNumber==target)

return mid;

else if(program.get(mid).lineNumber<target)

low=mid+1;

else

high=mid-1;

}

return -1;

}

public static int expr(String s)

{

Stack<Integer> v=new Stack<Integer>();

Stack<String> op=new Stack<String>();

int i=0;

while(i<s.length())

{

String p=toString(s.charAt(i));

if(isNumeric(p))

{

String num="";

int in=i;

while(in<s.length()&&isNumeric(s.substring(in,in+1)))

{

num+=s.charAt(in);

in++;

}

i=in-1;

v.push(Integer.parseInt(num));

}

else if(memory.containsKey(p))

{

v.push(memory.get(p));

}

else if(s.charAt(i)=='(')

{

String brac="";

int x=i+1;

while(x<s.length()&&s.charAt(x)!=')')

{

brac+=s.charAt(x);

x++;

}

i=x;

if(isNumeric(brac))

v.push(Integer.parseInt(brac));

else

v.push(expr(brac));

}

else if(isOperand(p))

{

while(op.size()>0&&precedence(op.peek(),p))

{

String c=op.pop();

int op1=v.pop();

int op2=v.pop();

v.push(calculate(op1,op2,c));

}

op.push(p);

}

i++;

}

while(op.size()>0)

{

String c=op.pop();

int op1=v.pop();

int op2=v.pop();

v.push(calculate(op1,op2,c));

}

return v.pop();

}

public static int atom(String s)

{

if(isNumeric(s))

return Integer.parseInt(s);

else if(memory.containsKey(s))

return memory.get(s);

else

return expr(s);

}

public static String toString(char ch)

{

return Character.toString(ch);

}

public static boolean isNumeric(String s)

{

try

{

Integer.parseInt(s);

return true;

}

catch( Exception e )

{

return false;

}

}

public static boolean isOperand(String s)

{

if(s.contains("+")||s.contains("-")||s.contains("\*")||s.contains("/"))

return true;

return false;

}

public static boolean containsOperand(String s)

{

if(s.contains("+")||s.contains("-")||s.contains("\*")||s.contains("/"))

return true;

if(s.contains("THEN"))

if(s.contains("<")||s.contains(">")||s.contains("=")||s.contains("!"))

return true;

return false;

}

public static int calculate(int op2,int op1,String op)

{

if(op.equals("+"))

return op1+op2;

else if(op.equals("-"))

return op1-op2;

else if(op.equals("\*"))

return op1\*op2;

else if(op.equals("/"))

return op1/op2;

else if(op.equals("<"))

return (op2<op1)?1:0;

else if(op.equals(">"))

return (op2>op1)?1:0;

else if(op.equals("="))

return (op1==op2)?1:0;

else

return (op1!=op2)?1:0;

}

public static boolean precedence(String op1,String op2)

{

HashMap<String,Integer> hm=new HashMap<String,Integer>();

hm.put("+",1);

hm.put("-",1);

hm.put("\*",2);

hm.put("/",2);

if(hm.get(op1)-hm.get(op2)>0)

return true;

else

return false;

}

}

**Output:**

