VIRTUAL EVALUATOR:

INFIX –POSTFIX EVALUATOR:

ADIL ABUWANI

Date: 12/13/2016

MAC 286 PROJECT

https://github.com/adilabuwani/PrefixPostFixApp.git

OBJECTIVE:

- This virtual machine will allow the user to enter an expressions

such as: A+B*C/D-3+F

and will evaluate this expression to postfix.

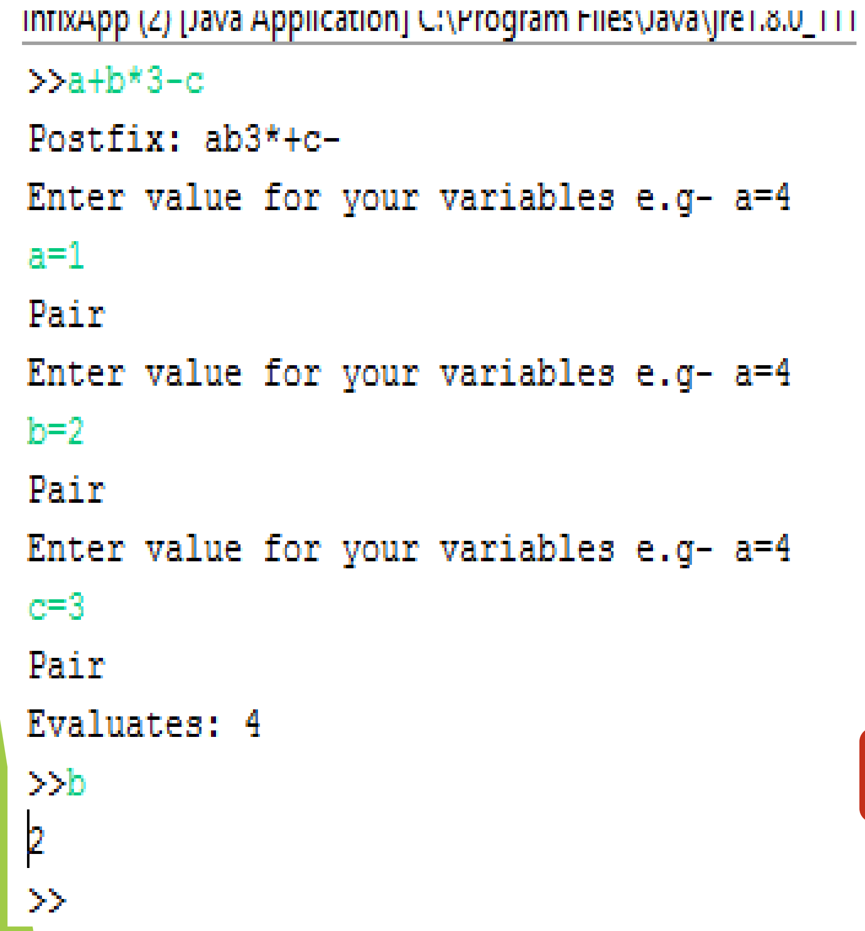
In addition, this evaluator will also allow the user to assign values to variables

Such as: A=1; OR B=3; or C=5

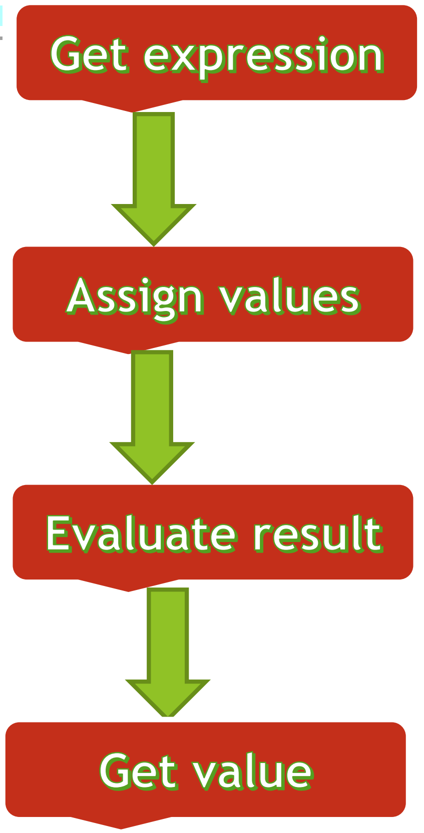
And will evaluate the result

The user will also be able to do stuff such as: A; OR B; and the user will know the value assigned to the variable

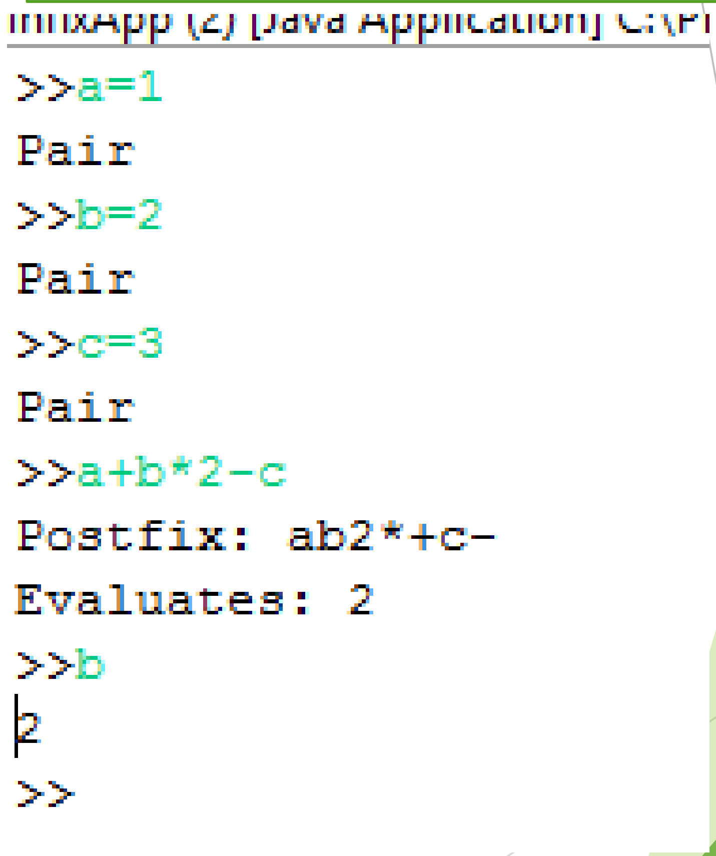
BIG PICTURE: A VIEW OF THE PROGRAM:



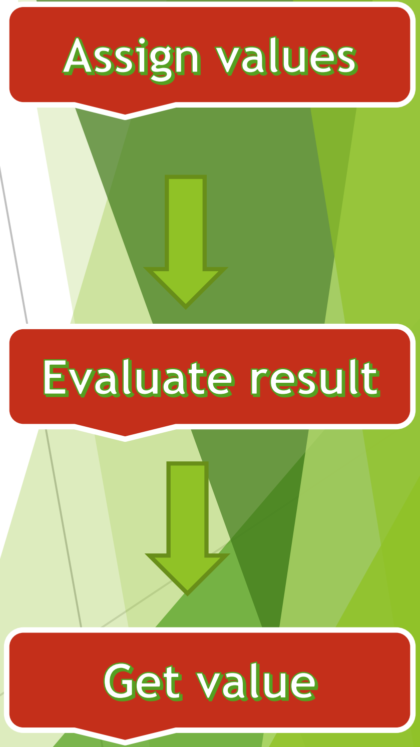
```
InfixApp (2) [Java Application] C:\Program Files\Java\jre1.8.0_111
>>a+b*3-c
Postfix: ab3*+c-
Enter value for your variables e.g- a=4
a=1
Pair
Enter value for your variables e.g- a=4
b=2
Pair
Enter value for your variables e.g- a=4
c=3
Pair
Evaluates: 4
>>b
2
>>
```



OR start with assigning values:

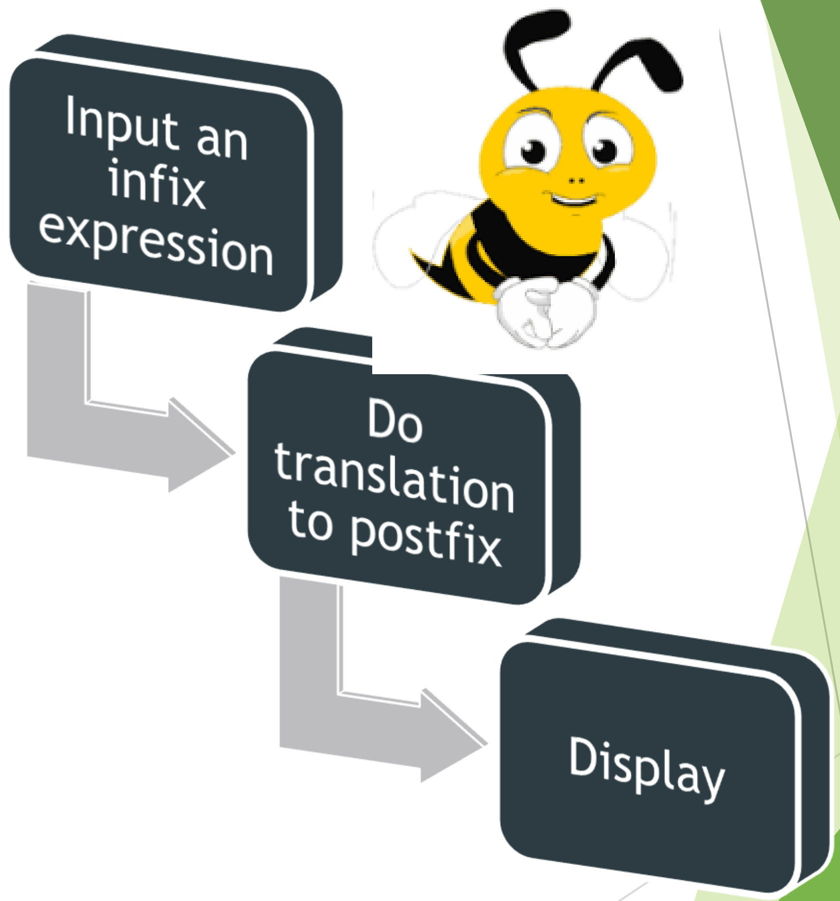


```
InfixApp (2) [Java Application] C:\P
>>a=1
Pair
>>b=2
Pair
>>c=3
Pair
>>a+b*2-c
Postfix: ab2*+c-
Evaluates: 2
>>b
2
>>
```

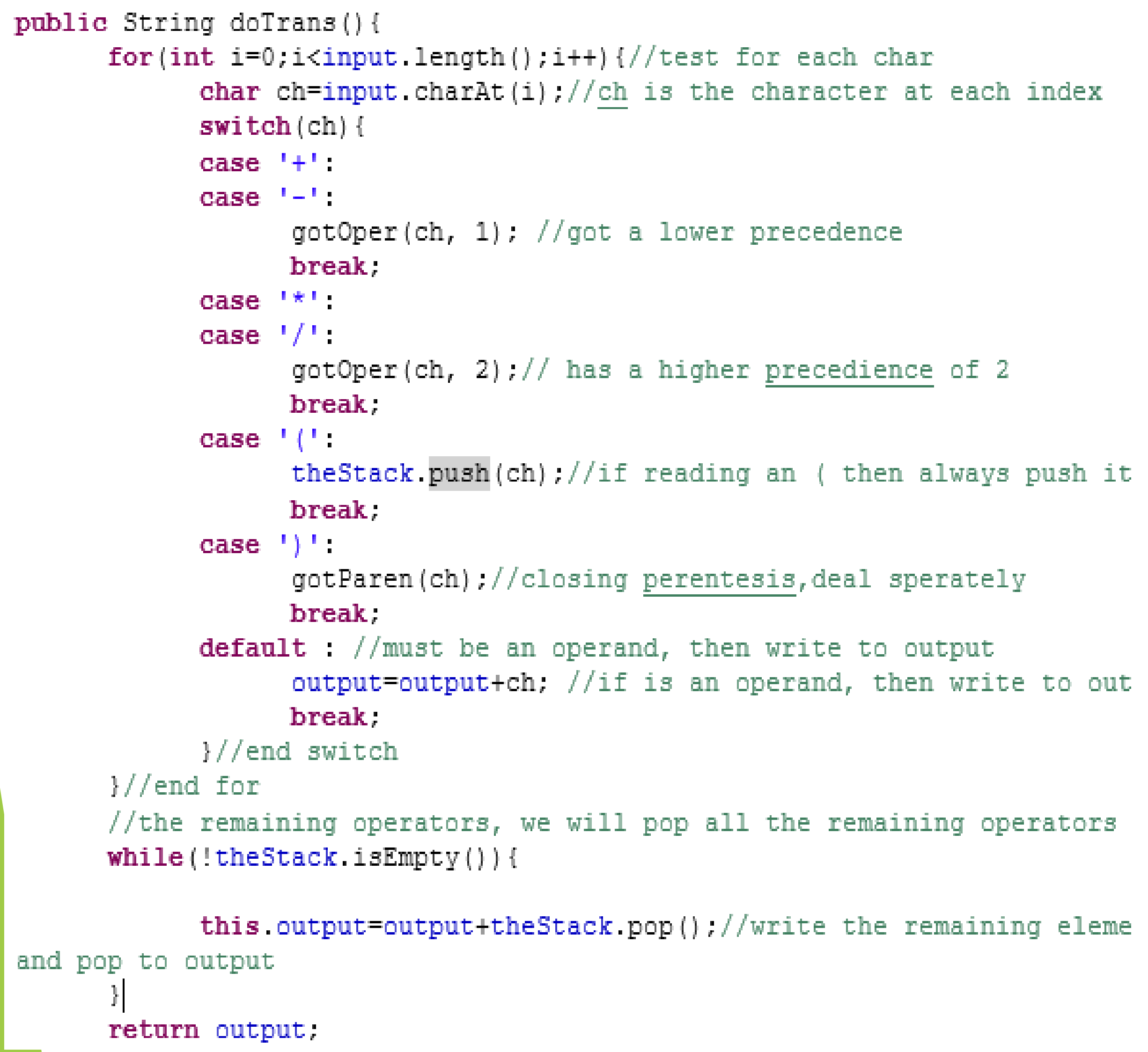


PROGRAM MECHANISM:

InToPost Class



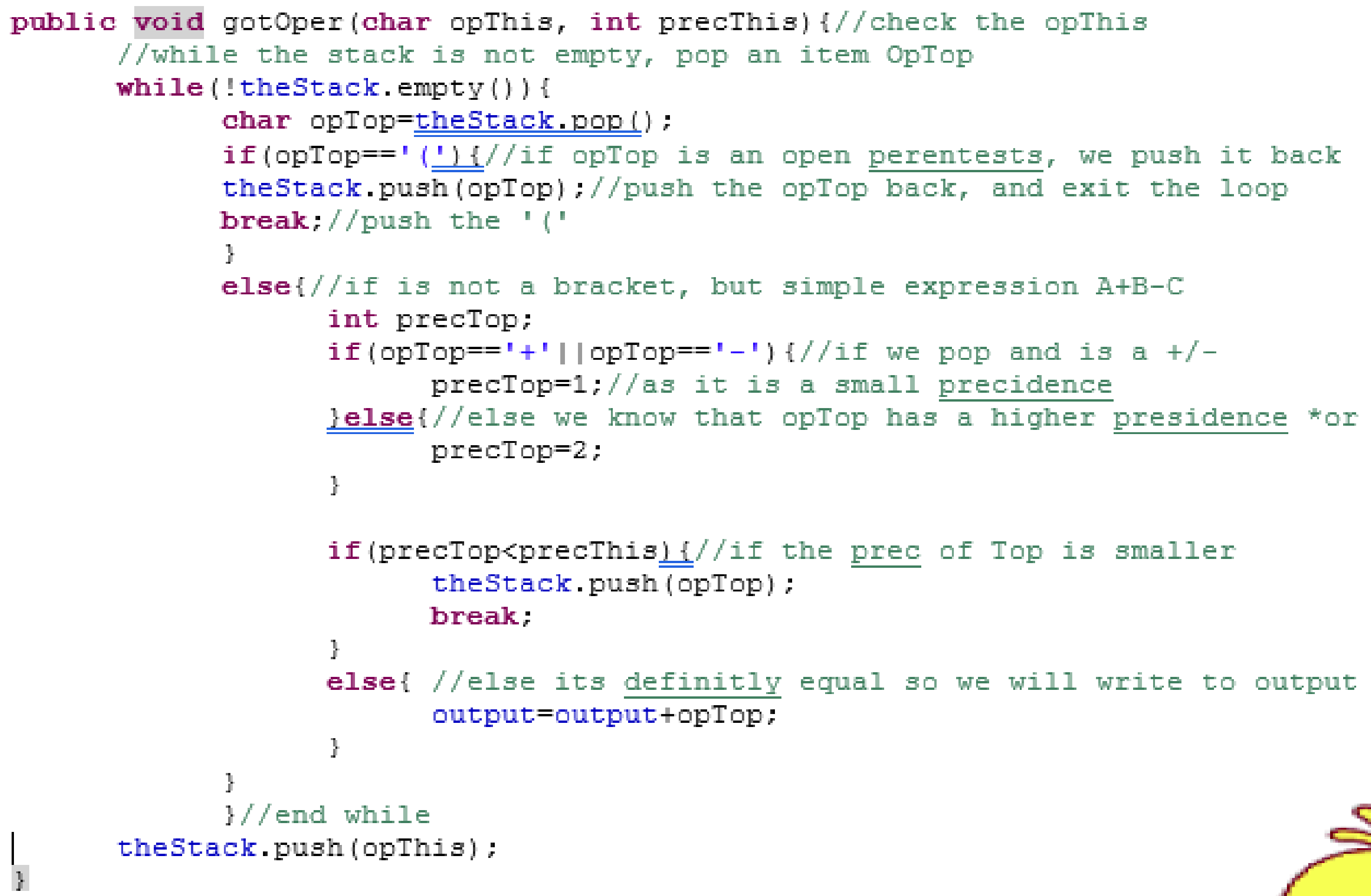
InToPost Class



```
public String doTrans(){
        for(int i=0;i<input.length();i++){//test for each char
                char ch=input.charAt(i);//ch is the character at each index
                switch(ch){
                case '+':
                case '-':
                        gotOper(ch, 1); //got a lower precedence
                        break;
                case '*':
                case '/':
                        gotOper(ch, 2);// has a higher precedience of 2
                        break;
                case '(':
                        theStack.push(ch);//if reading an ( then always push it
                        break;
                case ')':
                        gotParen(ch);//closing perentesis,deal sperately
                        break;
                default : //must be an operand, then write to output
                        output=output+ch; //if is an operand, then write to out
                        break;
                }//end switch
        }//end for
        //the remaining operators, we will pop all the remaining operators
        while(!theStack.isEmpty()){

                this.output=output+theStack.pop();//write the remaining eleme
and pop to output
        }
        return output;
```

Character:	Action:
operand	Default: write to output
Open parenthesis (	Push to stack
Close parenthesis)	While stack not empty, Pop an item, If item is not (, write it to output Quit loop if item is (
Operator (opThis)	If stack empty Push opThis, **else** ***While stack not empty***, -Pop an item, -If item is (, push it, or -If item is an operator-opTop If opTop < opThis, push opTop If opTop = opThis, pop opTop Quit loop if opTop < opThis or item is (-Push opThis to stack
No more items	While stack not empty pop items to output

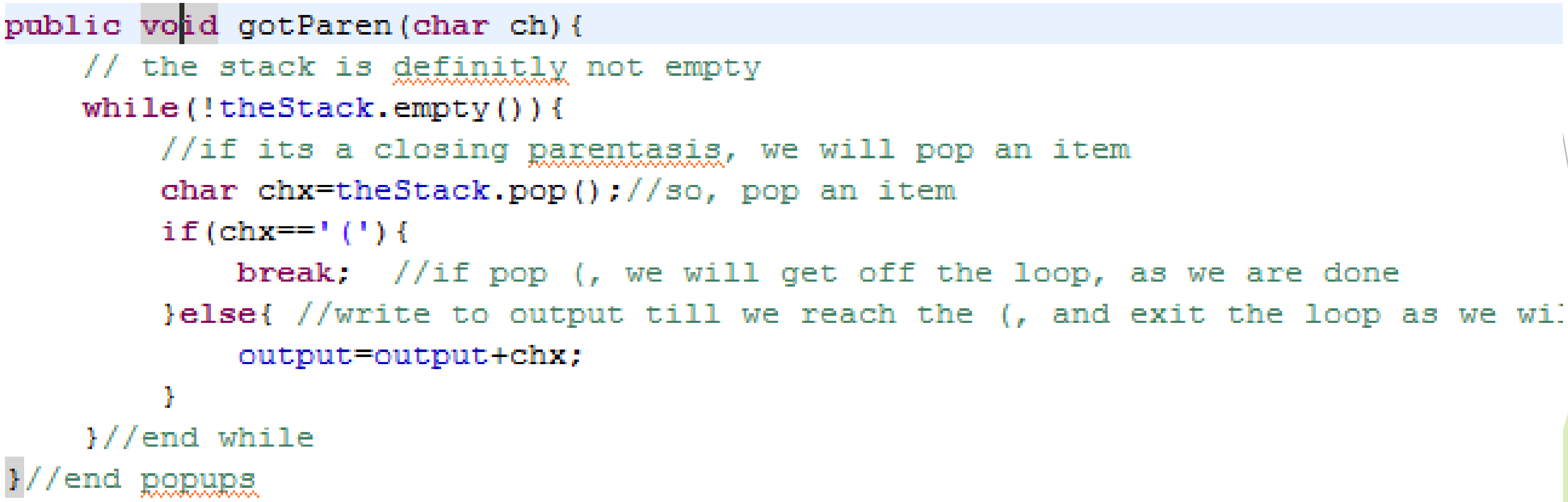
gotOper(opThis, precThis)



```
public void gotOper(char opThis, int precThis){//check the opThis
      //while the stack is not empty, pop an item OpTop
      while(!theStack.empty()){
            char opTop=theStack.pop();
            if(opTop=='('){//if opTop is an open perentests, we push it back
            theStack.push(opTop);//push the opTop back, and exit the loop
            break;//push the '('
            }
            else{//if is not a bracket, but simple expression A+B-C
                  int precTop;
                  if(opTop=='+'||opTop=='-'){//if we pop and is a +/-
                        precTop=1;//as it is a small precidence
                  }else{//else we know that opTop has a higher presidence *or
                        precTop=2;
                  }

                  if(precTop<precThis){//if the prec of Top is smaller
                        theStack.push(opTop);
                        break;
                  }
                  else{ //else its definitly equal so we will write to output
                        output=output+opTop;
                  }
            }
            }//end while
      theStack.push(opThis);
}
```

While (Stack !empty)
Optop:
If opTop==“(“ push to stack
Else if opTop==“+”or “-” (opTop is low precedence) then push opTop
Else write opTop to output (opTop is of equal precedence)
}
Push opThis to stack

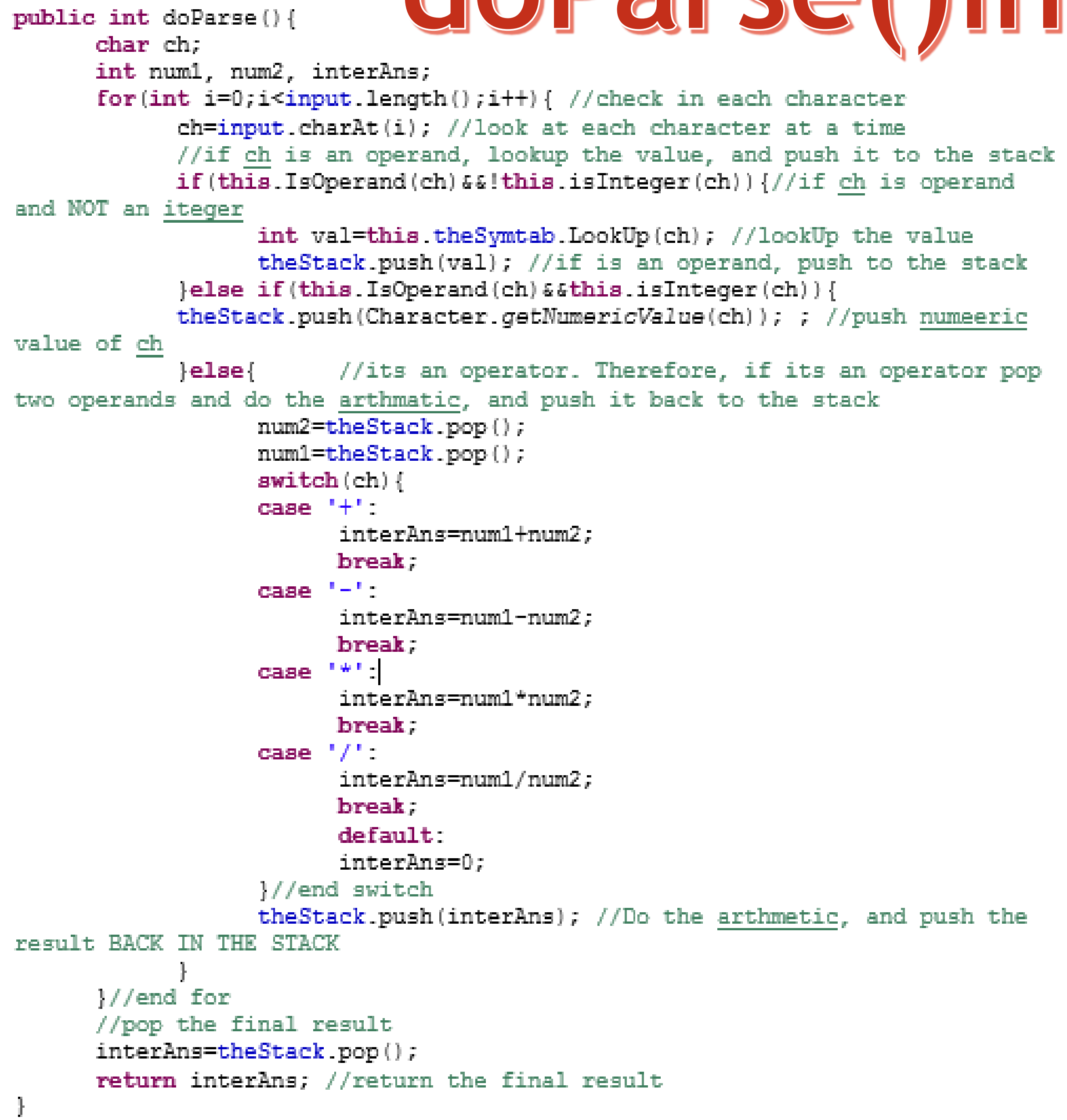
gotParen(ch)



```
public void gotParen(char ch){
    // the stack is definitly not empty
    while(!theStack.empty()){
        //if its a closing parentasis, we will pop an item
        char chx=theStack.pop();//so, pop an item
        if(chx=='('){
            break;  //if pop (, we will get off the loop, as we are done
        }else{ //write to output till we reach the (, and exit the loop as we wi
            output=output+chx;
        }
    }//end while
}//end popups
```

ParsePost Class

doParse()int



```
public int doParse(){
      char ch;
      int num1, num2, interAns;
      for(int i=0;i<input.length();i++){ //check in each character
            ch=input.charAt(i); //look at each character at a time
            //if ch is an operand, lookup the value, and push it to the stack
            if(this.IsOperand(ch)&&!this.isInteger(ch)){//if ch is operand
and NOT an iteger
                  int val=this.theSymtab.LookUp(ch); //lookUp the value
                  theStack.push(val); //if is an operand, push to the stack
            }else if(this.IsOperand(ch)&&this.isInteger(ch)){
            theStack.push(Character.getNumericValue(ch)); ; //push numeeric
value of ch
            }else{      //its an operator. Therefore, if its an operator pop
two operands and do the arthmatic, and push it back to the stack
                  num2=theStack.pop();
                  num1=theStack.pop();
                  switch(ch){
                  case '+':
                        interAns=num1+num2;
                        break;
                  case '-':
                        interAns=num1-num2;
                        break;
                  case '*':
                        interAns=num1*num2;
                        break;
                  case '/':
                        interAns=num1/num2;
                        break;
                        default:
                        interAns=0;
                  }//end switch
                  theStack.push(interAns); //Do the arthmetic, and push the
result BACK IN THE STACK
            }
      }//end for
      //pop the final result
      interAns=theStack.pop();
      return interAns; //return the final result
}
```

From postfix, Check on each character at a time

- Eg: ab+3-

If character is operand and **not a value**

- Look-Up the value from table and push to stack

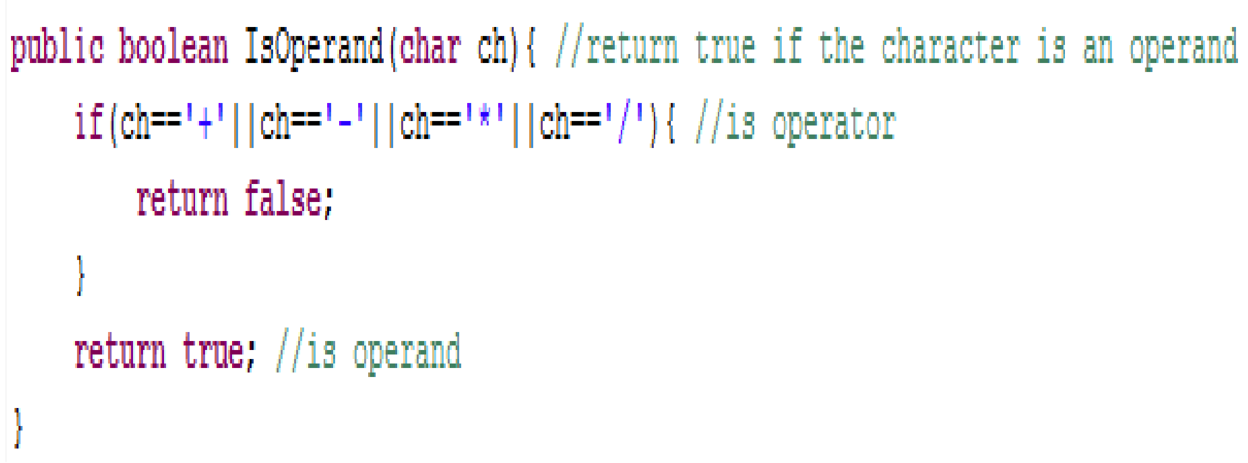
If character is an operand **and a value**

- Push directly to stack (No need to look up! Already a value)

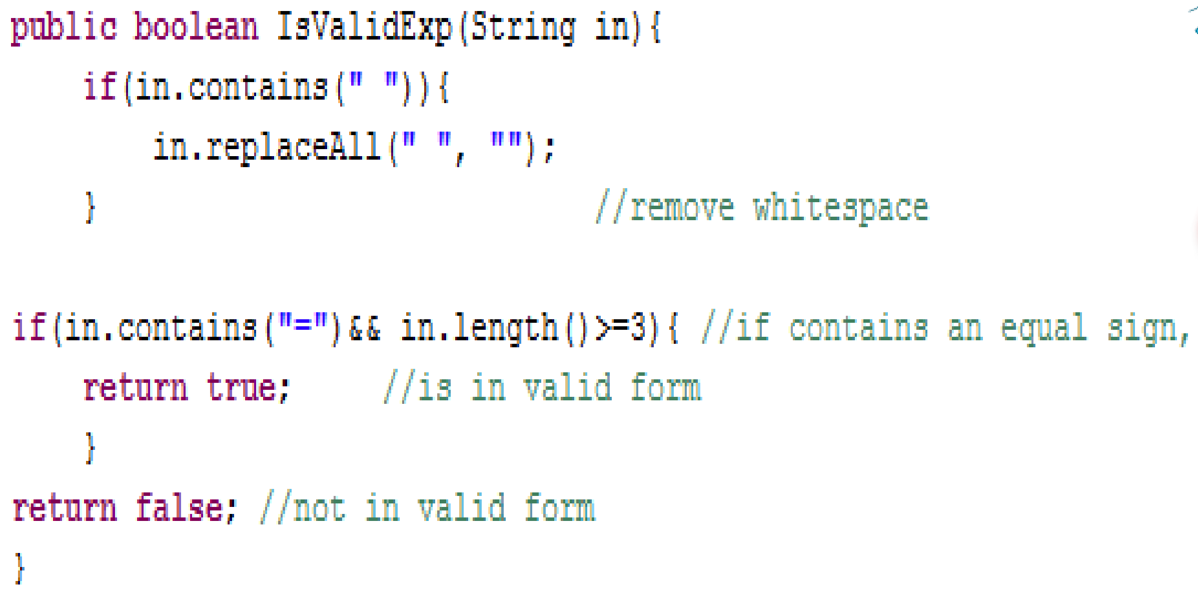
If character is an operator

- Pop TWO items from stack
- Evaluate with operator
- Push back to the stack

Final item in the stack is the answer evaluated



```
public boolean IsOperand(char ch){ //return true if the character is an operand
    if(ch=='+'||ch=='-'||ch=='*'||ch=='/'){ //is operator
        return false;
    }
    return true; //is operand
}
```



```
public boolean IsValidExp(String in){
    if(in.contains(" ")){
        in.replaceAll(" ", "");
    }                               //remove whitespace

if(in.contains("=")&& in.length()>=3){ //if contains an equal sign,
    return true;     //is in valid form
    }
return false; //not in valid form
}
```

Is it an Operator? operand? Or an integer?

Eg: A+b*3+5

A=operand

+=operator

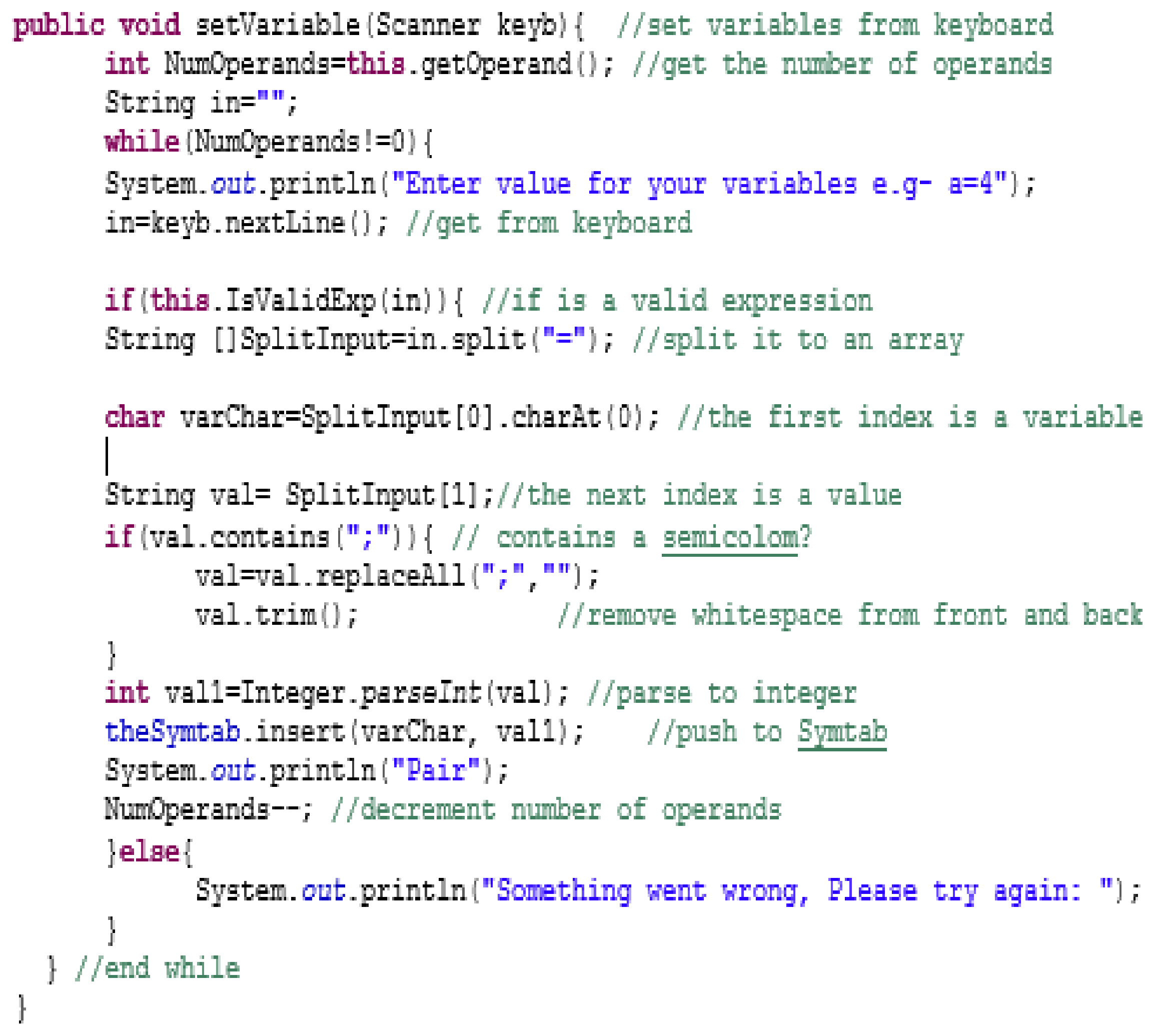
3=value

Valid expression?

Eg: A=3

Contains “=“ ?

Assigning Values:



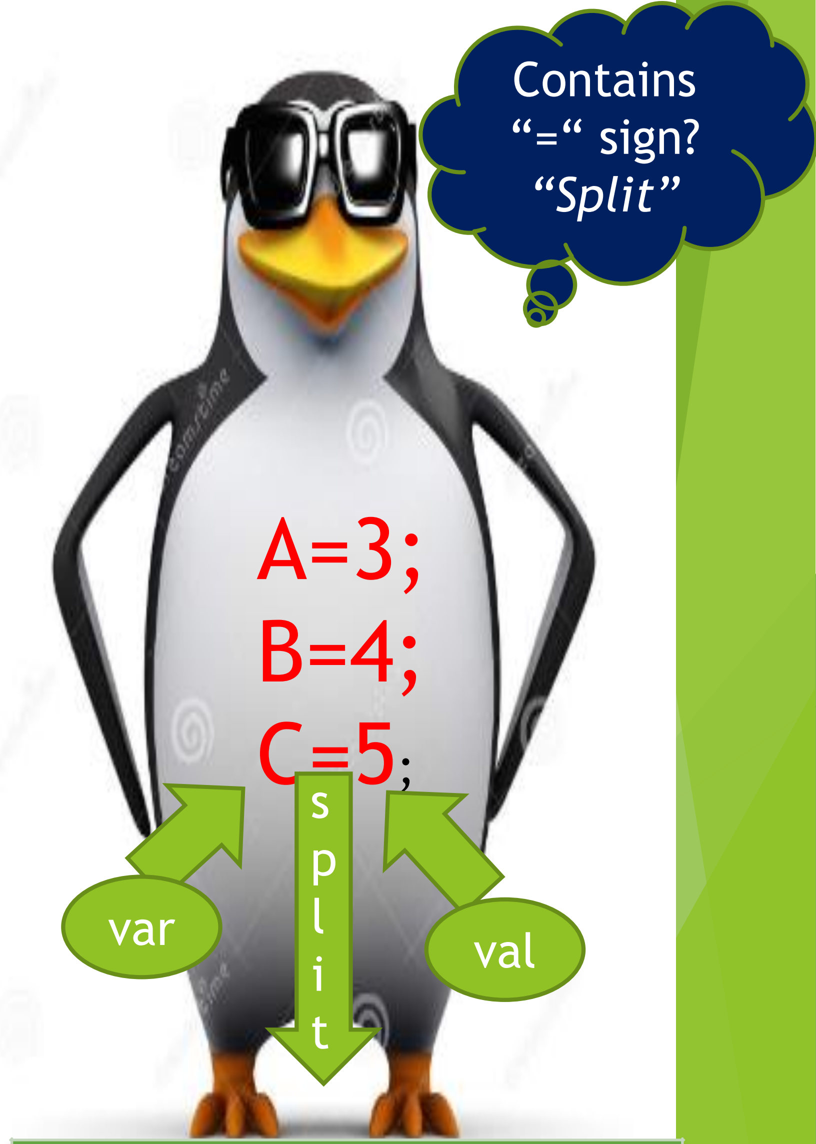
```
public void setVariable(Scanner keyb){  //set variables from keyboard
      int NumOperands=this.getOperand(); //get the number of operands
      String in="";
      while(NumOperands!=0){
      System.out.println("Enter value for your variables e.g- a=4");
      in=keyb.nextLine(); //get from keyboard

      if(this.IsValidExp(in)){ //if is a valid expression
      String []SplitInput=in.split("="); //split it to an array

      char varChar=SplitInput[0].charAt(0); //the first index is a variable

      String val= SplitInput[1];//the next index is a value
      if(val.contains(";")){ // contains a semicolom?
            val=val.replaceAll(";","");
            val.trim();               //remove whitespace from front and back
      }
      int val1=Integer.parseInt(val); //parse to integer
      theSymtab.insert(varChar, val1);    //push to Symtab
      System.out.println("Pair");
      NumOperands--; //decrement number of operands
      }else{
            System.out.println("Something went wrong, Please try again: ");
      }
   } //end while
}
```

Get input from user

Check if expression is in valid form

Split string containing variable on left and value on right

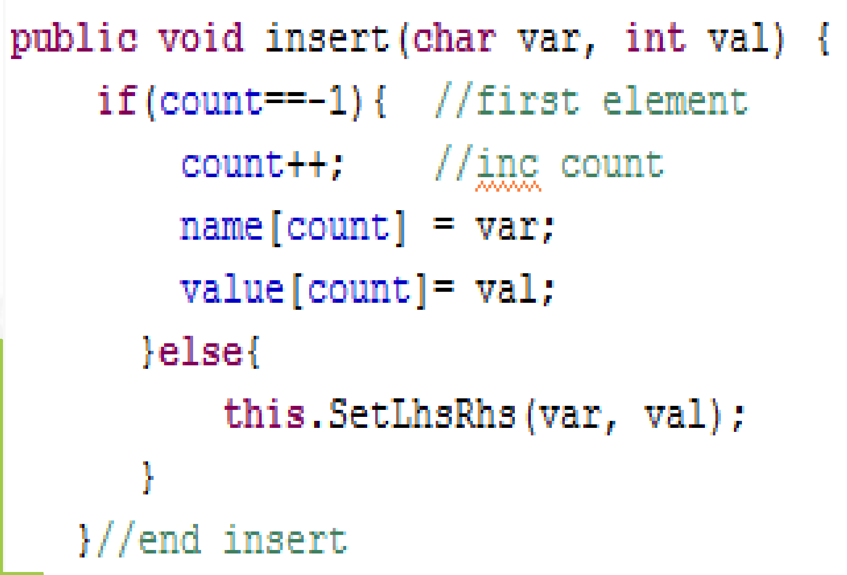
Push variable and value to SymTab



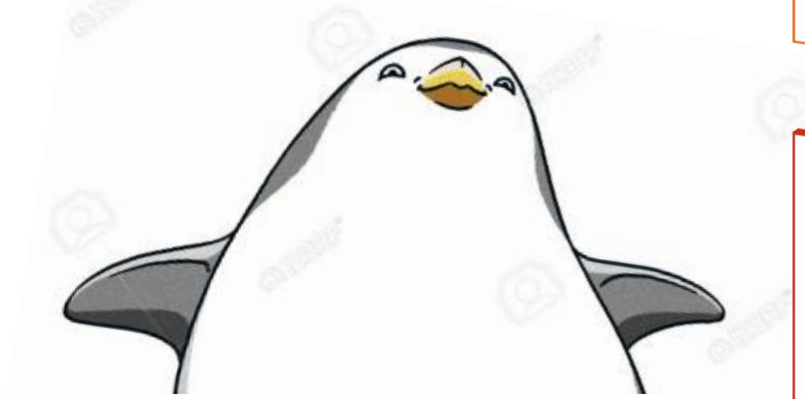
variable	value
var	val

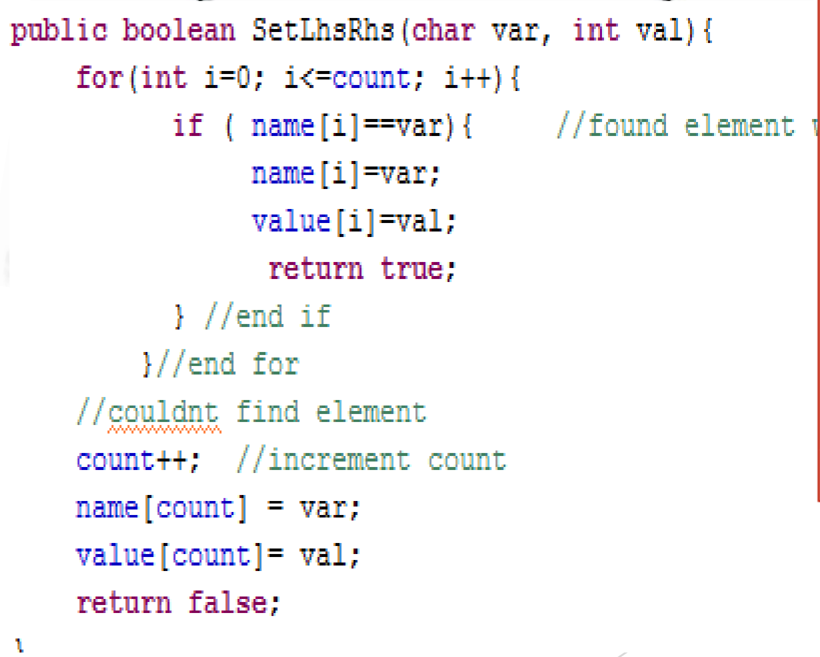
SymTab Table

Inserting an Item:



```
public void insert(char var, int val) {
    if(count==-1){  //first element
        count++;    //inc count
        name[count] = var;
        value[count]= val;
    }else{
        this.SetLhsRhs(var, val);
    }
 }//end insert
```





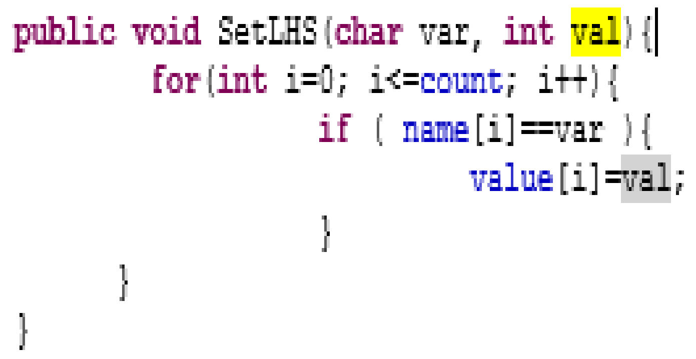
```
public boolean SetLhsRhs(char var, int val){
    for(int i=0; i<=count; i++){
        if ( name[i]==var){     //found element with same var, replace
            name[i]=var;
            value[i]=val;
             return true;
        } //end if
      }//end for
    //couldnt find element
    count++;  //increment count
    name[count] = var;
    value[count]= val;
    return false;
}
```

If Symtab is empty:

- Push element to the top of SymTab Table

If Symtab is not empty:

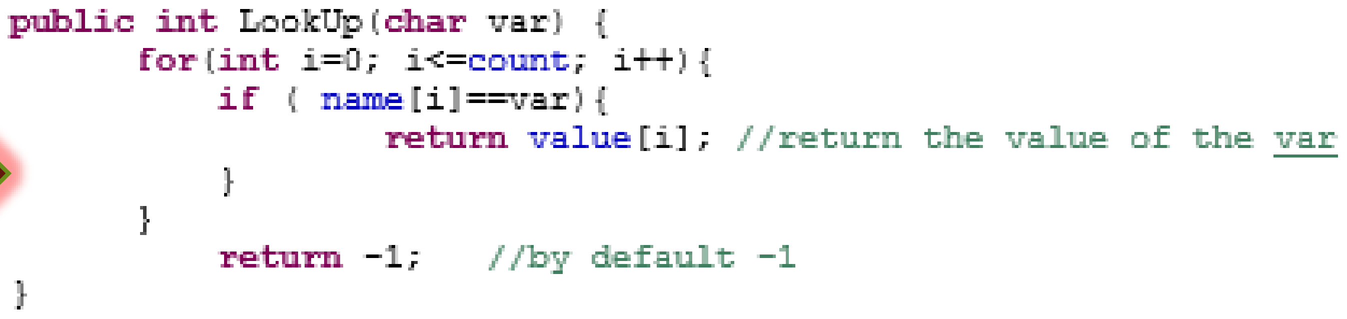
- First LookUp for var(LHS) in Symtab to check if LHS exist.
- If variable is found, replace with new Value(RHS)
- If variable not found, push new variable and its value to the top of SymTab



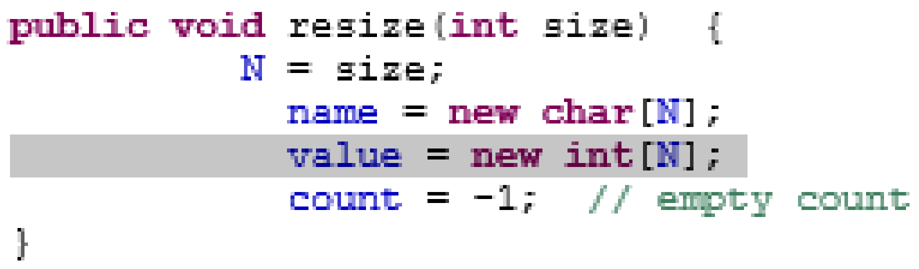
```
public void SetLHS(char var, int val){
        for(int i=0; i<=count; i++){
                if ( name[i]==var ){
                        value[i]=val;
                }
        }
}
```

SetLHS(var, val)
-Find LHS in SymTab, and replace with new RHS

LookUp(var)
-LookUp LHS in SymTab, and return Its Value
-By default -1 if LHS cannot be found in SYMTAB



```
public int LookUp(char var) {
        for(int i=0; i<=count; i++){
            if ( name[i]==var){
                    return value[i]; //return the value of the var
            }
        }
            return -1;    //by default -1
}
```

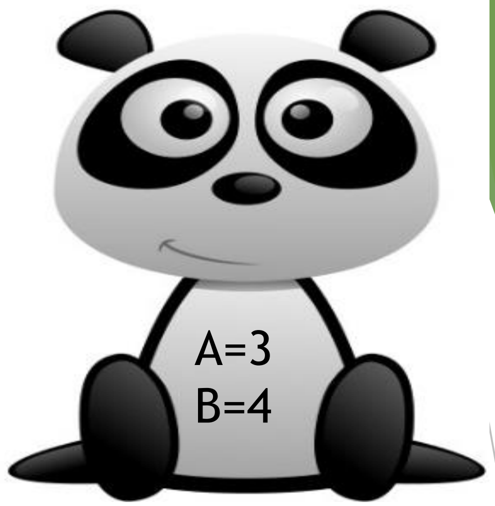


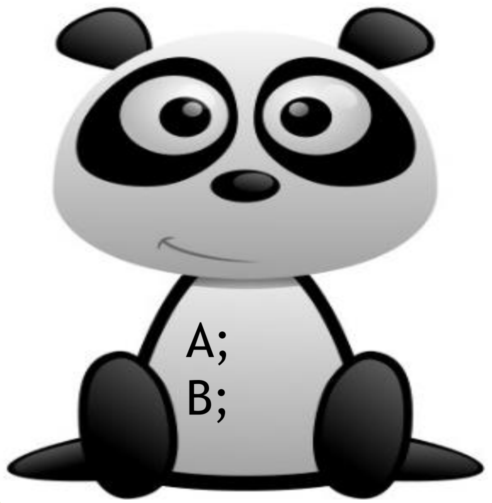
```
public void resize(int size)  {
            N = size;
              name = new char[N];
              value = new int[N];
              count = -1;  // empty count
}
```

MAIN APP: ACTION!!!

Input from user

If input contains "=" :
-assign variables
-push to SymTab

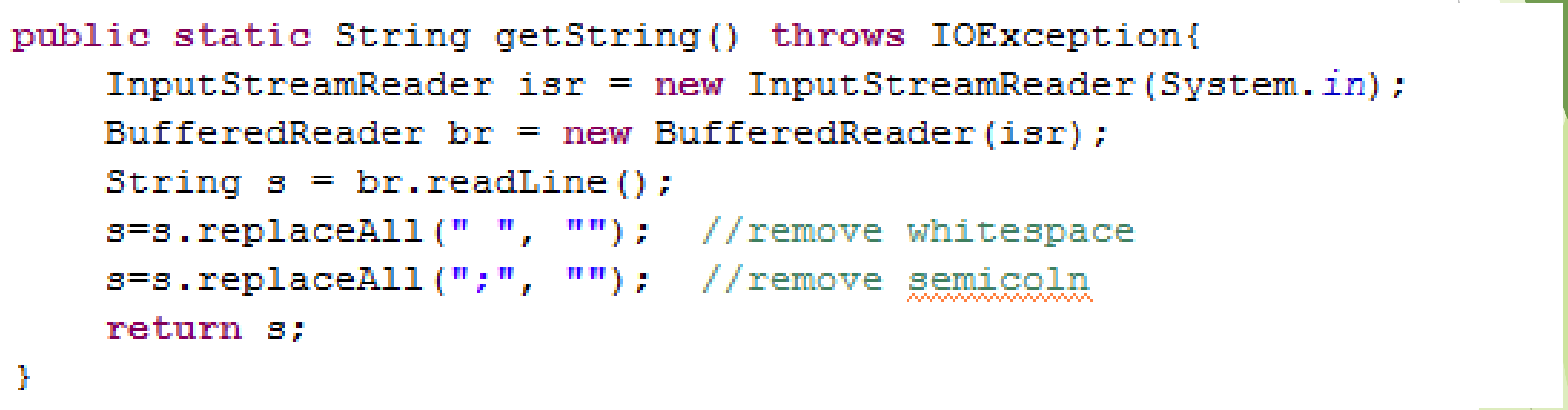




If input is an operand:
-lookup for value in SymTab
-return the value assigned.

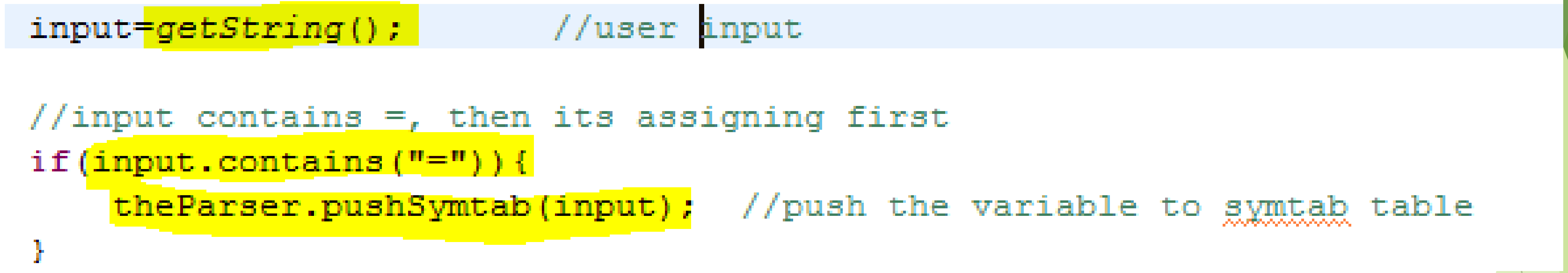
If Input is an expression:
-evaluate to postfix
-set variables from user
-push to SymTab

Get Input from user:



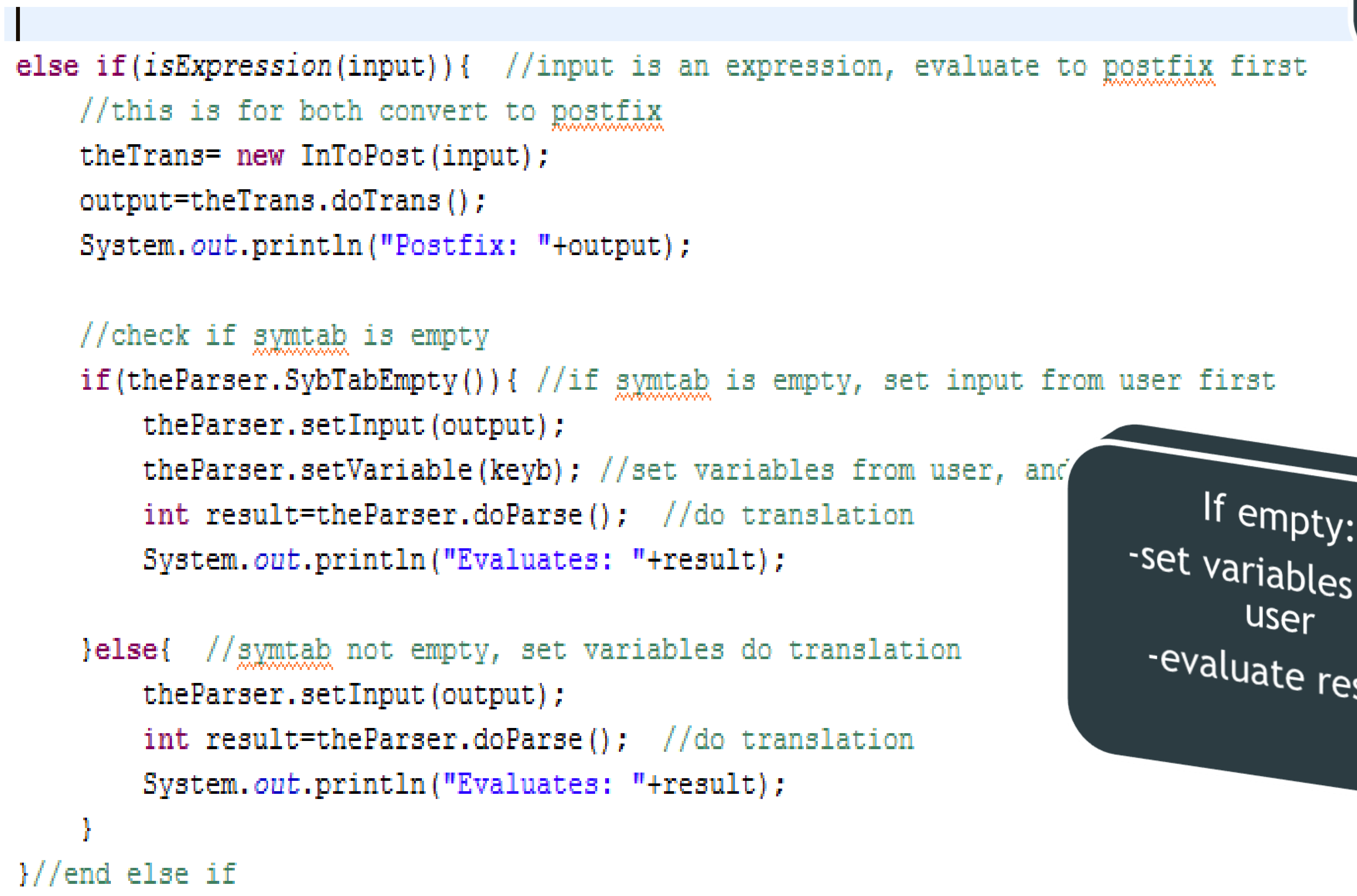
```
public static String getString() throws IOException{
    InputStreamReader isr = new InputStreamReader(System.in);
    BufferedReader br = new BufferedReader(isr);
    String s = br.readLine();
    s=s.replaceAll(" ", "");  //remove whitespace
    s=s.replaceAll(";", "");  //remove semicoln
    return s;
}
```

Input contains “=“:

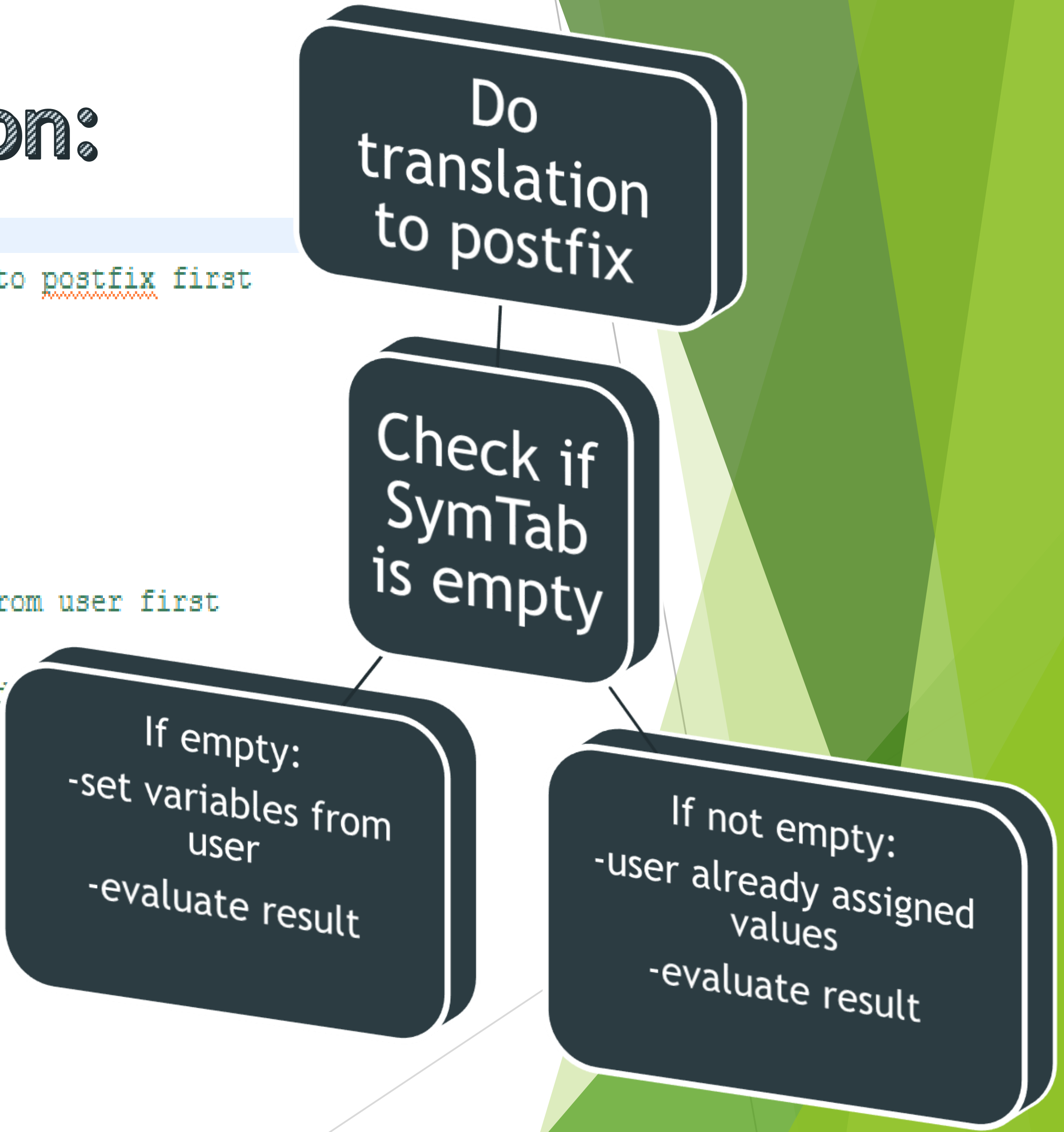


```
input=getString();          //user input

//input contains =, then its assigning first
if(input.contains("=")){
    theParser.pushSymtab(input);  //push the variable to symtab table
}
```

Input is an expression:



```
else if(isExpression(input)){  //input is an expression, evaluate to postfix first
    //this is for both convert to postfix
    theTrans= new InToPost(input);
    output=theTrans.doTrans();
    System.out.println("Postfix: "+output);

    //check if symtab is empty
    if(theParser.SybTabEmpty()){ //if symtab is empty, set input from user first
        theParser.setInput(output);
        theParser.setVariable(keyb); //set variables from user, and
        int result=theParser.doParse();  //do translation
        System.out.println("Evaluates: "+result);

    }else{  //symtab not empty, set variables do translation
        theParser.setInput(output);
        int result=theParser.doParse();  //do translation
        System.out.println("Evaluates: "+result);
    }
}//end else if
```

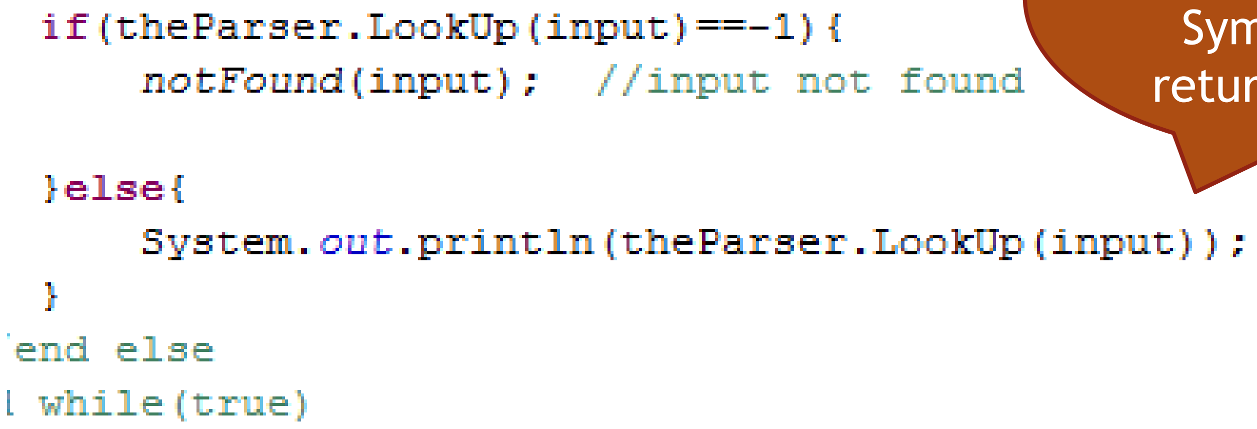


Do translation to postfix
Check if SymTab is empty
If empty: -set variables from user -evaluate result
If not empty: -user already assigned values -evaluate result

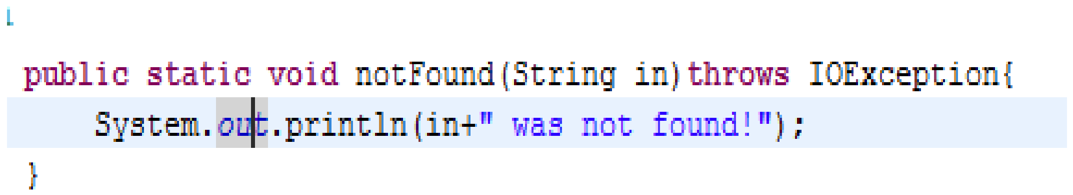
Input is an operand:

Not Found? Throw extension

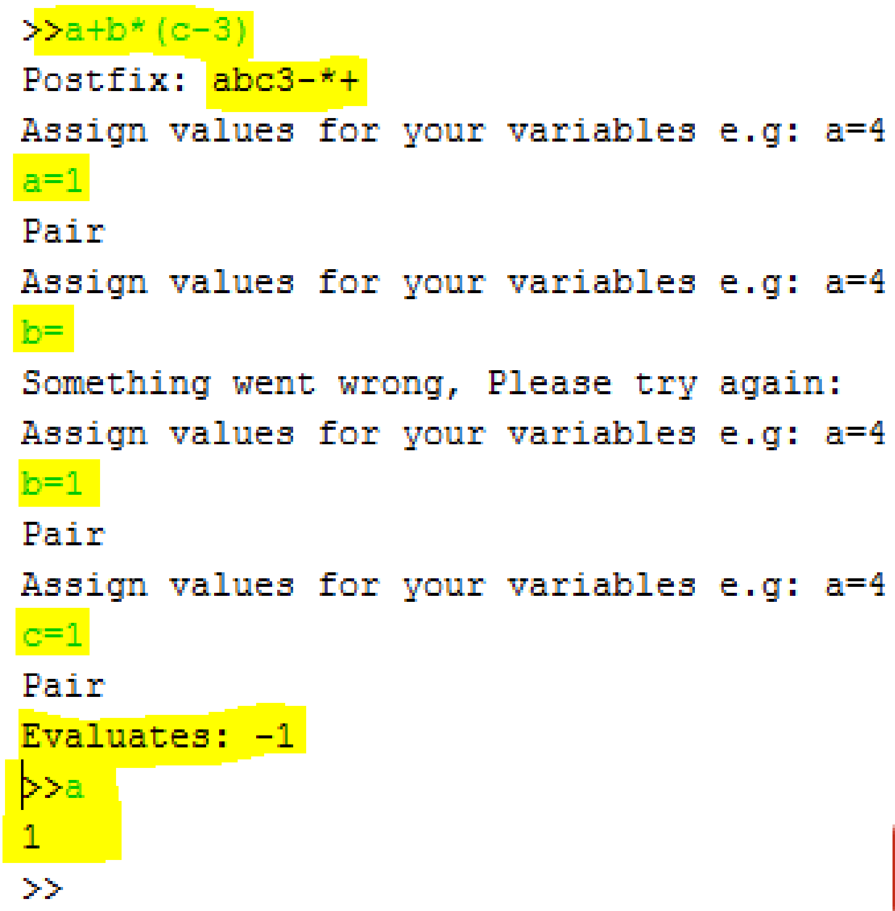
LookUp the value assigned from SymTab, and return its value



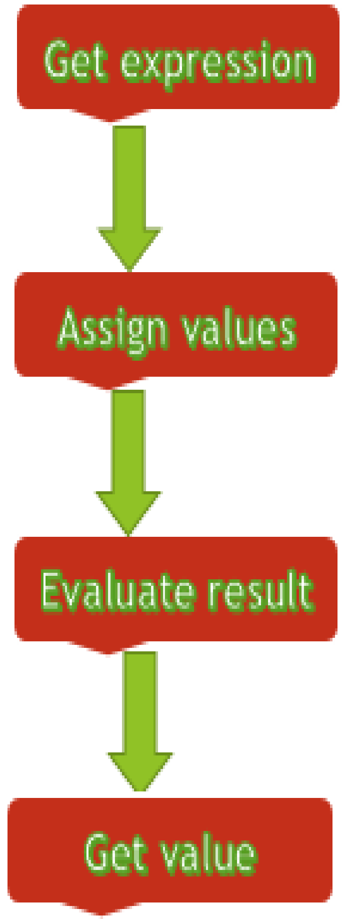
```
 if(theParser.LookUp(input)==-1){
     notFound(input);   //input not found

 }else{
     System.out.println(theParser.LookUp(input));
 }
'end else
l while(true)
```



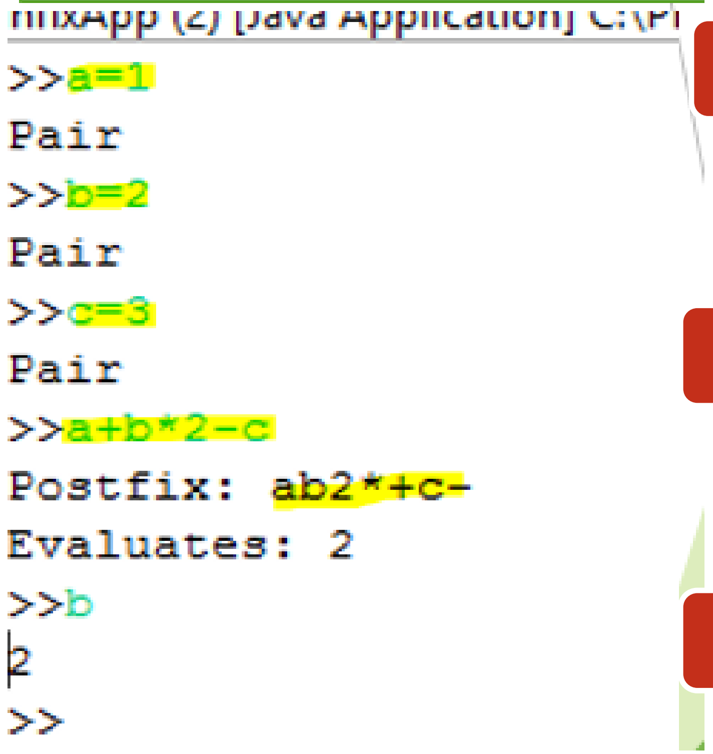
```
public static void notFound(String in)throws IOException{
    System.out.println(in+" was not found!");
}
```



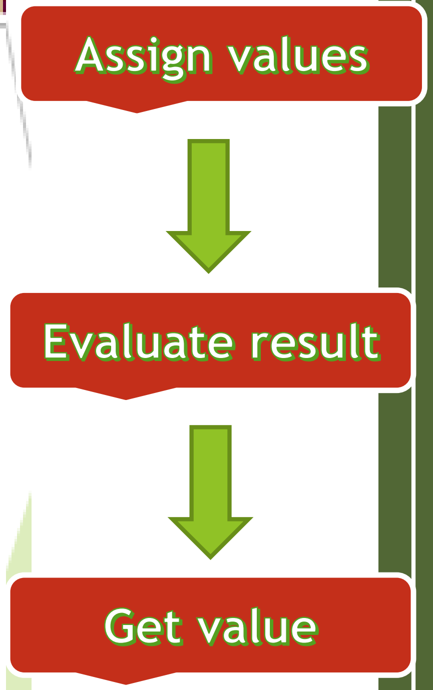
```
>>a+b*(c-3)
Postfix: abc3-*+
Assign values for your variables e.g: a=4
a=1
Pair
Assign values for your variables e.g: a=4
b=
Something went wrong, Please try again:
Assign values for your variables e.g: a=4
b=1
Pair
Assign values for your variables e.g: a=4
c=1
Pair
Evaluates: -1
>>a
1
>>
```



OR start with assigning values:



```
nfixApp (2) [Java Application] C:\P
>>a=1
Pair
>>b=2
Pair
>>c=3
Pair
>>a+b*2-c
Postfix: ab2*+c-
Evaluates: 2
>>b
2
>>
```



THANL YOU !

Clone, and Contribute to my App at:

GitHub

https://github.com/adilabuwani/PrefixPostFixApp.git