Vanna Moore

Program 2

**Main**

1 /\*For this program, I created classes for Stack, Queue, IsEquationValid, Translate, and Evaluate.  
 2 I used methods from those classes to validate and translate the equations. I couldn't get the evaluate to work.\*/  
 3   
 4 import java.io.File;  
 5 import java.io.FileNotFoundException;  
 6 import java.util.Scanner;  
 7   
 8 public class StackQueue{  
 9 public static void main(String[] args) throws FileNotFoundException{  
10 String A;  
11 File file = new File("C:\\Users\\vanna\\OneDrive\\Desktop\\CMPS 390\\Program2StacksQueues\\mathFile.txt");  
12 Scanner scan = new Scanner(file);  
13 isEquationValid eqStack = new isEquationValid();  
14 Translate infix = new Translate();  
15 // Evaluate ans = new Evaluate();  
16   
17 String eqA = scan.nextLine();  
18 System.out.print(eqA + ": ");  
19 eqStack.isValid(eqA);  
20 infix.translate(eqA);  
21 //ans.eval(eqA);   
22   
23 String eqB = scan.nextLine();  
24 System.out.print("\n" + eqB + ": ");  
25 eqStack.isValid(eqB);  
26 infix.translate(eqB);  
27 //ans.eval(eqB);   
28   
29 String eqC = scan.nextLine();  
30 System.out.print("\n" + eqC + ": ");  
31 eqStack.isValid(eqC);  
32 infix.translate(eqC);  
33   
34 String eqD = scan.nextLine();  
35 System.out.print("\n" + eqD + ": ");  
36 eqStack.isValid(eqD);  
37 infix.translate(eqD);  
38 // ans.eval(eqD);   
39   
40 String eqE = scan.nextLine();  
41 System.out.print("\n" + eqE + ": ");   
42 eqStack.isValid(eqE);  
43 infix.translate(eqE);  
44 //ans.eval(eqE);   
45   
46 String eqF = scan.nextLine();  
47 System.out.print("\n" + eqF + ": ");  
48 eqStack.isValid(eqF);  
49 infix.translate(eqF);  
50   
51 String eqG = scan.nextLine();  
52 System.out.print("\n" + eqG + ": ");  
53 eqStack.isValid(eqG);  
54 infix.translate(eqG);  
55   
56 String eqH = scan.nextLine();  
57 System.out.print("\n" + eqH + ": ");  
58 eqStack.isValid(eqH);  
59 infix.translate(eqH);  
60 //ans.eval(eqH);   
61 }   
62 }

**Stack Class**

1 public class Stack{  
 2 char[] stack = new char[20];  
 3 int top;  
 4   
 5 void init(){  
 6 int top = -1;   
 7 }  
 8   
 9 public void push (char c){  
10 top = top + 1;  
11 stack[top] = c;  
12 }  
13   
14 public char pop(){  
15 char c;  
16 c = stack[top];  
17 top = top-1;  
18 return c;  
19 }  
20   
21 boolean isStackEmpty(){   
22 boolean empty = false;  
23 if (top == -1){  
24 empty = true;  
25 }  
26 return empty;   
27 }  
28   
29 void showStack(){  
30 int j;  
31 for(j = 0; j <= top; j++){  
32 System.out.print(stack[j]);  
33 }  
34 }  
35 }

**Queue Class**

1 public class Queue{  
 2 char[] queue = new char[128];  
 3 int front, rear;  
 4   
 5 void init(){  
 6 front = 0;  
 7 rear = -1;  
 8 }  
 9   
10 void push(char c){  
11 rear = rear + 1;   
12 queue[rear] = c;   
13 }  
14   
15 char pop(){  
16 char x;  
17 x = queue[front];  
18 front = front + 1;  
19 return x;  
20 }  
21   
22 boolean isQueueEmpty(){  
23 boolean empty;  
24 empty = false;  
25 if (rear <= front){  
26 empty = true;  
27 }  
28   
29 return empty;  
30 }  
31   
32 void showQueue(){  
33 int j;  
34 if (front >= rear)  
35 return;  
36 else  
37 {  
38 for(j = 0; j <= rear; j++){  
39 System.out.print(queue[j]);  
40 }  
41 }  
42 }  
43 }

**isEquationValid Class**

1 public class isEquationValid{  
 2 int j;  
 3 char m;  
 4 boolean isGood;  
 5   
 6   
 7 boolean isValid(String eq){  
 8 Stack e = new Stack();  
 9   
10 e.init();  
11 for(j = 0; j < eq.length(); j++){  
12 m = eq.charAt(j);  
13 if (m == 40){  
14 e.push(m);  
15 }  
16 if (m == 41){  
17 e.pop();  
18 }   
19 }  
20   
21 if (e.isStackEmpty() == false) {  
22 isGood = false;  
23 System.out.println("Valid equation.");  
24 }   
25 else if(e.isStackEmpty() == true)  
26 {  
27 isGood = true;  
28 System.out.println("Invalid equation.");  
29 }  
30 return isGood;   
31 }  
32   
33 }

**Translate**

1 public class Translate{  
 2 char stack[] = new char[128];  
 3 int top;  
 4 int num, x, y, z;  
 5 String s;  
 6 char c;   
 7 char m;  
 8 char myOp;  
 9   
10 void translate(String eq){  
11 Stack post = new Stack();  
12 Stack op = new Stack();  
13 Stack postfix = new Stack();  
14 int j;  
15 post.init();  
16 op.init();  
17   
18 for(j = 0; j < eq.length(); j++){  
19 m = eq.charAt(j);  
20 if(m > '0' && m < '9'){  
21 postfix.push(m);   
22 }  
23 else if(m == '+' || m == '/' || m == '-' || m == '\*'){  
24 op.push(m);  
25 }  
26 else if (m == '('){  
27 }  
28 else if(m == ')'){  
29 while(op.isStackEmpty() == false){  
30 myOp = op.pop();  
31 postfix.push(myOp);   
32 }   
33 }   
34 }  
35 postfix.showStack();   
36 }  
37 }

**numStack Class**

1 public class numStack{  
 2 int[] stack = new int[20];  
 3 int top;  
 4   
 5 void init(){  
 6 int top = -1;   
 7   
 8 }  
 9   
10 public void push (int c){  
11 top = top + 1;  
12 stack[top] = c;  
13   
14 }  
15   
16 public int pop(){  
17 int c;  
18 c = stack[top];  
19 top = top-1;  
20   
21 return c;  
22 }  
23   
24 boolean isStackEmpty(){   
25 boolean empty = false;  
26 if (top == -1){  
27 empty = true;  
28 }  
29 return empty;   
30 }  
31   
32 void showStack(){  
33 int j;  
34 for(j = 0; j <= top; j++){  
35 System.out.print(stack[j]);  
36 }  
37 }  
38   
39 }

**Evaluate Class**

1 public class Evaluate{  
 2 Stack post = new Stack();  
 3 Stack op = new Stack();  
 4 int[] numStack = new int[20];  
 5 numStack num = new numStack();  
 6   
 7 char operator;  
 8 char c;  
 9 int j, k, x, y, z;  
 10 int answer;  
 11   
 12 void eval(String eq){  
 13   
 14 op.init();  
 15 for(j = 0; j < eq.length(); j++){  
 16 c = eq.charAt(j);  
 17   
 18 while(op.isStackEmpty() == false){  
 19 if(c > '0' && c < '9'){  
 20 post.push(c);  
 21 }  
 22 else if(c == '+' || c == '-' || c == '\*' || c == '/'){  
 23 operator = op.push(c);  
 24 }  
 25   
 26 while(op.isStackEmpty == false){  
 27 y = (int)post.pop();  
 28 x = (int)post.pop();  
 29 c = op.pop();  
 30   
 31 if (c == '+'){  
 32 z = x + y;  
 33 }  
 34 else if (c == '-'){  
 35 z = x - y;  
 36 }  
 37 else if (c == '\*'){  
 38 z = x \* y;  
 39 }   
 40 else if (c == '/'){  
 41 z = x / y;  
 42 }   
 43 num.push(z);   
 44   
 45 }  
 46   
 47 if(num.isStackEmpty() == false){  
 48 operator = op.pop();  
 49 y = num.pop();  
 50 x = num.pop();  
 51 if(operator == '+' || operator == '-' || operator == '\*' || operator == '/'){  
 52 if (c == '+'){  
 53 z = x + y;  
 54 }  
 55 else if (c == '-'){  
 56 z = x - y;  
 57 }  
 58 else if (c == '\*'){  
 59 z = x \* y;  
 60 }   
 61 else if (c == '/'){  
 62 z = x / y;  
 63 }   
 64   
 65 answer = z;   
 66 }  
 67 }  
 68 }  
 69 }   
 70 }  
 71 }

A screenshot of a computer

Description automatically generated