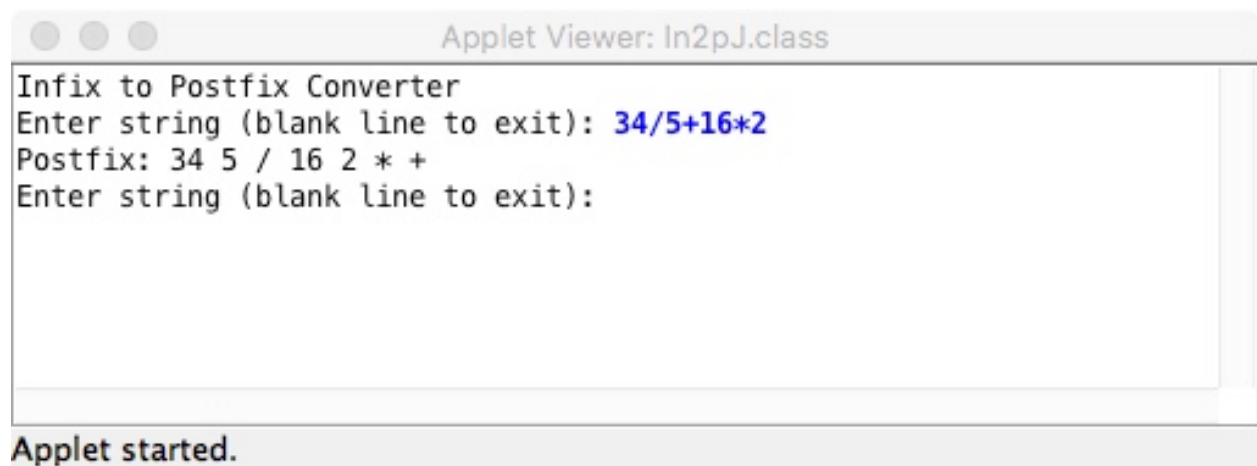


Department of Electrical and Computer Engineering
ECSE 202 – Introduction to Software Development
Assignment 3
Application of Stacks and Queues: Expression Parsing

Problem Description

Write a program, In2pJ, which reads in an infix expression from the console and outputs a postfix expression that can subsequently be processed by a simple interpreter. This output will be used in the next assignment to create a simple 4-function calculator, as outlined in the attached file. You are not being asked to do a full implementation of the calculator, only the first stage which converts an algebraic expression of the form $34 / 5 + 16 * 2$ (infix) to postfix notation, which for this example is $34\ 5\ /\ 16\ 2\ *\ +$.

Your program should operate as follows:



```
Applet Viewer: In2pJ.class
Infix to Postfix Converter
Enter string (blank line to exit): 34/5+16*2
Postfix: 34 5 / 16 2 * +
Enter string (blank line to exit):
Applet started.
```

Note that in the above example, the program is run in an infinite loop – you do not need to do this for the assignment. Expressions will be limited to operands, binary operators, and parentheses only.

Method

You are to use the Shunting Yard algorithm, the simple version of which is described in the attached file, and uses 3 data structures: 2 queues for holding input and output and a stack for holding operators. You have already seen the Stack class in Assignment 2. One of the difficulties in writing this program is in parsing the input string and separating it into operators and operands. The example below shows how the StringTokenizer class for this purpose.

```

import java.util.StringTokenizer;
import acm.program.ConsoleProgram;

public class TestParse extends ConsoleProgram {
    public void run() {
        String str = readLine("Enter string: ");
        StringTokenizer st = new StringTokenizer(str,"+-*/",true);
        while (st.hasMoreTokens()) {
            println("-->" + st.nextToken());
        }
    }
}

```

Here's what happens when you run the program:

```

Enter string: 34/5+16*2
-->34
-->/
-->5
-->+
-->16
-->*
-->2

```

You need to devise an appropriate loop (hint, look at the while loop in the above example) that enqueues this data in the input queue. From here, implementation of the Shunting Yard program follows the recipe outlined in the attached description. Once your program is working, run through each of the following test cases, saving the results to a file.

Test Cases:

Enter string (blank line to exit): 34/5+16*2

Postfix: 34 5 / 16 2 * +

Enter string (blank line to exit): 5+9.27/1.4*3 + 2/3

Postfix: 5 9.27 1.4 / 3 * + 2 3 / +

Enter string (blank line to exit): 1.1-2.2*3.4/5.6

Postfix: 1.1 2.2 3.4 * 5.6 / -

Enter string (blank line to exit): (3+4*5)*(6*(9-5))

Postfix: 3 4 5 * + 6 9 5 - * *

Enter string (blank line to exit): 1.4*(2.3-5.6)/(6.2-4.1)

Postfix: 1.4 2.3 5.6 - * 6.2 4.1 - /

Instructions

Write the In2pJ program as described in the preceding sections. It should operate interactively and be able to replicate the output from the test cases shown above. To obtain full marks your code must be fully documented and correctly replicate the test cases.

Your submission should consist of 5 files:

1. Stack.java - stack class
2. Queue.java - queue class
3. listNode.java - node object
4. In2pJ.java - program
5. In2pJ.txt - output

Upload your files to myCourses as indicated.

fpf/February 16, 2020