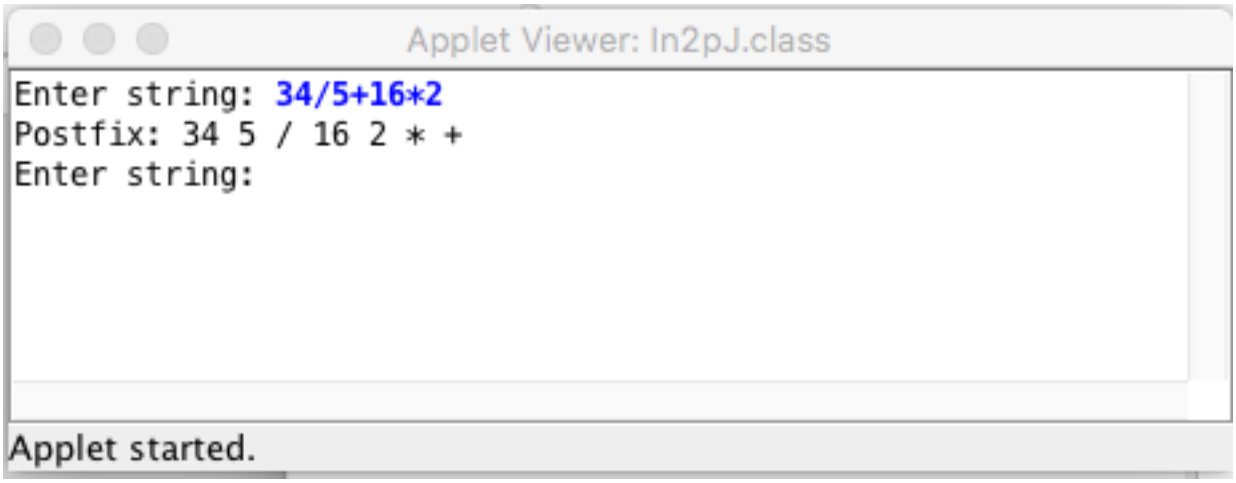
Department of Electrical and Computer Engineering
ECSE 202 – Introduction to Software Development
Assignment 4
Introduction to Java Programming

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 Assignment 5 to create a simple 4-function calculator, as outlined in the lecture slides for Lectures 14 and 15, Pages 31-36. 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 * +. (Note: even though this is part of the "C" lectures, this program is to be written in Java).

Your program should operate as follows:



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 and binary operators only (bonus points awarded for also handling unary operators and parentheses).

Method

You are to use the Shunting Yard algorithm, the simple version of which is described on Page 32 of the lecture slides, and uses 3 data structures: 2 queues for holding input and output and a stack for holding operators. Begin by writing 3 classes:

1. ListNode.java – that defines the data object used by the queue and stack routines
2. Stack.java – that provides the push() and pop() methods, and instance variable top
3. Queue.java – that provides the enqueue() and dequeue() methods, and instance variables front and rear.

It is best to write suitable test programs to verify that the Stack and Queue classes are implemented correctly. Once this is accomplished, you can proceed with your implementation of the switchyard algorithm.

One of the difficulties in writing this program is in parsing the input string and separating it into operators and operands. Fortunately Java provides the stringTokenizer class that makes this easy. You can look up how this works online. Here is a simple test program you might want to try.

```
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 notes (there also a lot of information available online). Once your program is running, run through each of the following test cases, saving the results to a file.

Test Cases:

Enter string: 34/5+16*2
Postfix: 34 5 / 16 2 * +
Enter string: 5+9.27/1.4*3 + 2/3
Postfix: 5 9.27 1.4 / 3 * + 2 3 / +
Enter string: 1.1-2.2*3.4/5.6
Postfix: 1.1 2.2 3.4 * 5.6 / -
Enter string: 6/7+3/4+1/2
Postfix: 6 7 / 3 4 / + 1 2 / +
Enter string: 9.8+3*6.7/2-4
Postfix: 9.8 3 6.7 * 2 / + 4 –

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/October 23, 2017