7. Stacks and Queues :: Example Application :: Evaluating Arithmetic Expressions

How do we handle parentheses, e.g., how would we evaluate (1 + ((2 + 3) * (4 - 5)))?

Left parentheses are handled as a special type of operator with precedence level 0. They are always pushed onto the stack. When a right parenthesis is encountered we have to repeatedly evaluate the top until the matching left parenthesis is on top of the operator stack. We then pop the (and continue parsing.

Here is the final algorithm which handles parentheses:

```
Method evaluate(In: String pExpr) Returns Number
  Create operatorStack -- Stores Operators
  Create operandStack  -- Stores Operands
  While end of pExpr has not been reached Do
    Scan next token in pExpr
    If token is a Number Then
      Convert token to Number object named number
      operandStack.push(number)
    ElseIf token is "(" Then
      Convert token to Operator object named op
      operatorStack.push(op)
```

7. Stacks and Queues :: Example Application :: Evaluating Arithmetic Expressions

```
    ElseIf token is "+", "-", "*", or "/" Then
      Convert token to Operator object named op
      While precedence(operatorStack.peek()) > precedence(op) Do
        topEval()
      End While
       operatorStack.push(op)
    Else -- token is ")"
      While not operatorStack.peek() = "(" Do
        topEval()
      End While
      operatorStack.pop() -- Pops the "("
    End If
  End While
  While not operatorStack.isEmpty() Do
    topEval()
  End While
  Return operandStack.pop()
End Method evaluate
```

7. Stacks and Queues :: Example Application :: Evaluating Arithmetic Expressions

Evaluate (1 + ((2 + 3) * (4 - 5)))