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

This leads us to a preliminary algorithm:

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

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

```
Method topEval() Returns Nothing
  right ← operandStack.pop()
  left ← operandStack.pop()
  op ← operatorStack.pop()
  If op is + Then
    operandStack.push(left + right)
  ElseIf op is - Then
    operandStack.push(left - right)
  ElseIf op is * Then
    operandStack.push(left * right)
  Else
    operandStack.push(left / right)
  End If
End Method topEval
```

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

```
Method precedence(In: Operator pOperator) Returns Int
  If pOperator is * or / Then
    Return 2
  Else
    Return 1
  End If
End Method precedence
```