**California State University, Sacramento**
**College of Engineering and Computer Science**

# CSc 20: Programming Concepts and Methodology II

**Lab 05**

## Objective:

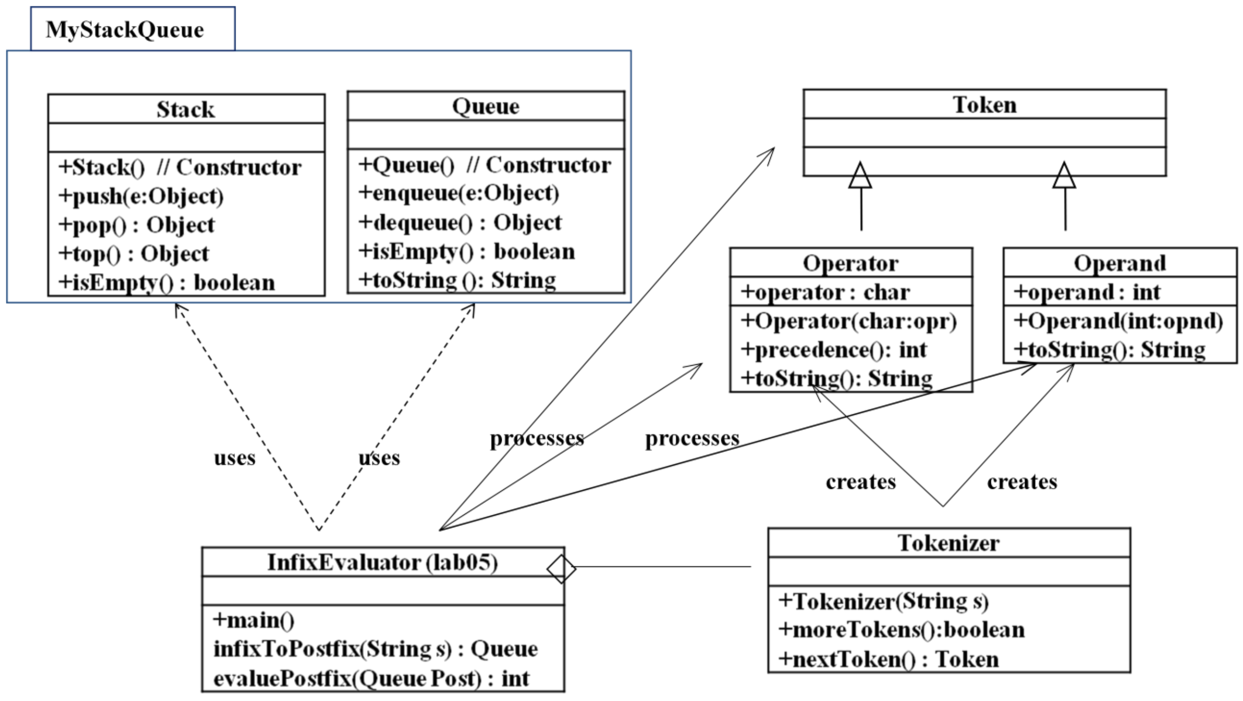
The objective of this lab is to get you some experience in using stacks, queues, class packages and UML.

The programming assignment:

In this lab, you are to write an infix evaluator. Given an input string of characters like "12+34*(56-7)-18/9", first print the postfix equivalent "12 34 56 7 - * + 18 9 / -" and then print the value of the expression.

Notes: In this lab,

1. You must use the algorithms given in class to convert infix expressions to postfix expressions and to evaluate postfix expressions.
2. You must use the instructor's stack and queue package.
3. You must use the instructor's frame work depicted by the following class diagrams.
4. Copy directory lab05 from ~wang/sample20.
5. Complete methods infixToPostfix and evaluatePostfix.



Some programming Hints:

1. To create a stack:

   ```
   Stack theStack = new Stack();
   ```

2. To push ‘#’ into the stack:

   ```
   theStack.push(new Operator(‘#’));
   ```

3. To create a Tokenizer from a string s:

   ```
   Tokenizer T = new Tokenizer(s);
   ```

4. Repeat until no more tokens:

   ```
   While (T.moreTokens()) { …..}
   ```

5. To get the next token:

   ```
   Token Tkn = T.nextToken();
   ```

6. To pop a token into a variable of the type Token:

   ```
   Token Current = (Token) theStack.pop();
   ```

7. To cast a Token into an Operator:

   ```
   Opr = (Operator)Tkn;
   ```

8. To check if a Token variable Tkn contains an Operand:

   ```
   if (Tkn instanceof Operand)…
   ```

9. To check if an operator is a ‘(‘:

   ```
   if (Opr.operator==’(‘)…
   ```

10. To perform an operation:

```
switch(Opr.operator) {
        case '+': result = opnd1 + opnd2; break;
…
}
```

11. To check the operator on the top of the stack:

```
((Operator)theStack.top()).operator
```