CS 340 Fall 2021
Programming Quiz 2
Due 11:59 PM Monday September 27
• Implement the private methods evaluate and toInfix. The implementation of the private methods must be recursive.

• evaluate returns the value of the expression tree. The operators in the tree include ^ (exponentiation), *, /, %, +, -, ! (unary minus)

• toInfix returns the fully parenthesized infix format of the expression. In the infix expression use - for the unary minus
Programming Quiz 2
Sample input and output

• Sample input: 10 2 + 4 ! 2 - *

• Each token in the input is separated by white space

• In the commend line argument put double quotes around the expression

• Sample output: (( 10 + 2 ) * ( ( - 4 ) - 2 )) = -72
import java.io.*;
import java.util.*;

public class ExpressionTree {

    private class Node {
        private Node left;
        private String data;
        private Node right;

        private Node(Node l, String d, Node r) {
            left = l;
            data = d;
            right = r;
        }
    }

    private Node root;
}
public ExpressionTree(String exp) {
    Stack<Node> operands = new Stack<>();
    Scanner s = new Scanner(exp);
    Node x;
    Node y;
    while (s.hasNext()) {
        String token = s.next();
        if (Character.isDigit(token.charAt(0)))
            operands.push(new Node(null, token, null));
        else {
            if (token.charAt(0) != '!')
                x = operands.pop();
            else
                x = null;
            y = operands.pop();
            operands.push(new Node(y, token, x));
        }
    }
    root = operands.pop();
}
public int evaluate() {
    //return the value expression tree
    //you can assume the tree is not empty
    return evaluate(root);
}

private int evaluate(Node r) {
    //PRE: r != null
    //return the value of the expression tree rooted at r

}
public String toInfix() {
    //return a fully parenthesized infix representation of the expression tree
    //you can assume the tree is not empty
    return toInfix(root);
}

private String toInfix(Node r) {
    //return a fully parenthesized infix representation of the
    //of the expression tree rooted at r
}

public static void main(String args[]) {
    //Simple test driver. I might use a different test driver
    ExpressionTree exp = new ExpressionTree(args[0]);
    System.out.println(exp.toInfix());
    System.out.println(exp.evaluate());
}
• Upload one zip file called pq2.zip to Canvas. The zip file must contain only one file called ExpressionTree.java. Do not upload your whole Eclipse project!