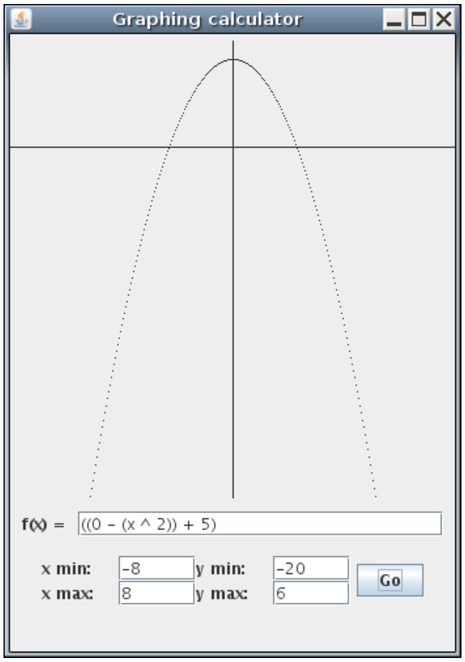
CSCI 235 Spring 2020

Project #7 (Due: 5 pm on May 2) **Apr. 20**
Part A

Documentation and style will be assessed as part of your grade. **Follow the full code guidelines.**

This project is to write a program that operates as a graphing calculator. It will have a window that allows a user to enter in a function and set min and max values for *x* and *y* (to specify the viewing range); it will draw a graph of the function within the specified range. The original program that was written by Dr. VanDrunen looks like this:



f(x) = ((0 – (x ^ 2)) + 5)

x min: -8 y min: -20
x max: 8 y max: 6

You are not required to mimic the GUI exactly, but your program should have a window with the basic features/elements that are found in the example window.

The program should have three parts:

1. Parsing and interpreting the function
2. Making the window
3. Drawing the actual graph

Interpreting the function
Download project files from Schoology.

The first task is similar to the Parse Tree program that we did in class and lab.

The calculator will display the graph of a function whose body is defined by the following expression:

expression → *numeral* | *x* | (*expression op expression*)
op → + | - | / | * | ^

Expressions now use floating point instead of integer. Thus, the values will be `double` rather than `int`.
Expressions can have a variable *x*.
The operator ^ (exponentiation) has been added.

Recall that the expression must be fully parenthesized. In addition, we do not allow a unary negation, and thus -3x^2 + 5 must be provided in the form of ((0 – (3 * (x ^ 2))) + 5) or (((0 – 3) * (x ^ 2)) + 5).

As we did for Parse Trees, the calculator program must interpret an expression in String by building appropriate trees.

Task #1: The Tree classes

The `ExprNode` interface has the `evaluate` method, `double evaluate(double x);`
The method now returns a double and has a double, formal parameter. This parameter <u>indicates the value of the variable *x*</u>.

Your task is to write classes `Number`, `Variable`, and `Operation` which will implement `ExprNode`. You will need to figure out how each `evaluate` method will differ. (Hint: What will those classes do with the formal parameter *x*? The `Variable` class is **the only one that will use it**, the `Number` class will ignore it, and the `Operation` class will pass it along in the recursive calls.)

Task #2: Building the trees
The class `ExprStringSlicer` does the same work, and the class `Interpreter` has a static method `parse()`. Your task is to write the recursive `parse` method. Its base cases will handle `Number` and `Variable` as the variable *x* has been introduced.

You can test your work using the main method of the class, giving an expression (in quotes) and an *x*-value on the command line, i.e., `$java Interpreter "((0 – (x ^ 2)) + 5)" 3.5`

Don't worry yet about handling erroneous input.

Task #3: Displaying the window
The `PaintPanel` and `Painter` are provided. Your task is to complete the main method of `GraphCalc`. The class already has some code for GUI components as seen below. You can rearrange those components and add them to your window, panels, or layout managers.

```
public static void main(String[] args) {

       JFrame window = new JFrame("Graphing calculator");
       window.setLayout(new FlowLayout());
       window.setSize(350, 600);

       PaintPanel graphPanel = new PaintPanel(350, 350);

        JTextField funcField = new JTextField(25);
        JTextField xminField = new JTextField(5);
        JTextField yminField = new JTextField(5);
        JTextField xmaxField = new JTextField(5);
        JTextField ymaxField = new JTextField(5);

        xminField.setText("-10");
        xminField.setText("-10");
        xmaxField.setText("10");
        ymaxField.setText("10");

        JButton go = new JButton("Go");

       window.add(graphPanel);
       JPanel panel2 = new JPanel();
       panel2.setLayout(new FlowLayout());
       panel2.add(go);
       window.add(panel2);
       window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
       window.setVisible(true);
    }
```

The last task of this project will be drawing the graph on the `PaintPanel` whenever the Go button is pressed.