**CSCI 235 Spring 2020**

**Project #7 (Due: 5 pm on May 2)**

**Part B**

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

**DO NOT edit ExprNode.java, Painter.java or PaintPanel.java.**

**Preparing for action listener**

Convert the main method of GraphCalc to a constructor so that you can pass an object of this class to other classes. Its main method then will simply create a window by calling the constructor as follows:

public static void main(String[] args) {

GraphCalc theWindow = new GraphCalc();

}

**Task #1: Write a class that implements ActionListener for Go button.**

The class will get inputs from the JTextFields in the window and verify them. For right now, you can simply print values from the text fields and the function’s value at the min *x* or max *x*. Add code for handling exceptions as well.

**Task #2: Add code for drawing to the ActionListener class.**

We learned that a Java class could implement multiple interfaces. Thus, your ActionListener class can implement Painter interface as well. The heading of the class will be as follows:

public class GoListener implements ActionListener, Painter { …

The above class then should implement actionPerformed and paint methods. If your class has an instance (object) of PaintPanel from GraphCalc, you may need to use the following statement for setting Painter:

*graphPanel*.setPainter(this);

where *graphPanel* is the instance of PaintPanel, and this indicates the current instance of GoListener.

* GoListener implements Painter as well, and thus, this works for the Painter object too.

The paint method should:

1. Figure out the scale of graph so that you can associate pixels on the panel with points in the plane. For example, if the *x* range were -10 to 10, and the width of the panel were 200, the *x* value of -9 corresponds to the coordinate 10 on the vertical axis of the panel. With the same *y* range and the height of 200, the *y* value of -9 corresponds to the coordinate 190.
2. If 0 is within the *x* range, draw the *y*-axis; if 0 is within the *y* range, draw the *x*-axis.
3. Draw the curve. For each horizontal pixel coordinate,
   1. Find the corresponding *x* value.
   2. Find the value of the function at that *x* value: *f*(*x*). Use the Interpreter class.
   3. If the output value from the function is within the *y* range, fill in the pixel that corresponds to (*x*, *f*(*x*)) by drawing a filled rectangle of area 1 pixel.

Call repaint() on your PaintPanel (i.e., *graphPanel*.repaint()).

**What to submit:**

The following six files to Schoology:

* The three Java files for Number, Variable, and Operation
* Interpreter.java
* GraphCalc.java
* The class that implements the button’s ActionListener and Painter