Lab 8 – GUI, Data Manager and Data Elements

Concepts tested by this program:
- Inheritance
- Interfaces
- Polymorphism
- Abstract classes
- Overriding methods
- Graphing with JavaFX

NOTE: This lab requires attention to detail. Follow the following instructions carefully.

When computing limits, it is sometimes not obvious whether a limit exists simply by studying the function. Graphing the function and viewing it may sometimes help. In this lab, you will use and extend an application that graphs several functions. Refer to Task #0 through Task #3 below for specific instructions.

Operation

- When the user selects the "Graph a Function" button, the application will display a list of functions and ask the user to select one of them to plot.
- The application will then ask the user for the left and right extents (i.e., between what values of x will the function be plotted), from which it will calculate the maximum and minimum values of the function that will be displayed.
- Then the application will transform the function to the coordinate system of the display and plot it.
- The user may then re-graph the same or a different function, or exit.

Specifications

Data Element – the data elements are each value of x, from left extent to right extent, by increments of one pixel, and the corresponding values of f(x). The x and f(x) transformed into the display coordinate system are an additional data element.

Data Structure – The abstract class Function holds an abstract method fnValue that is specific to each sub-class function, and an implemented function fnValueToPlot that uses fnValue to transform the values to the display coordinate system.

The following three functions have been implemented, each in a class that extends Function:

1. $f_1(x) = \frac{1}{x}$
2. $f_2(x) = \sin x$
3. $f_3(x) = \frac{8x-\sqrt{x}}{x^3-7x^2+15x-9}$

Each function has a toString() method that returns a description of the function, preceded by its index.

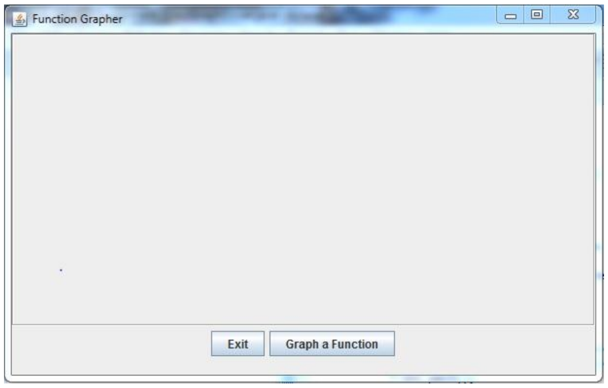
Data Manager – the GraphManager class implements the methods of GraphManagerInterface. The GraphManager does not use any GUI methods.

The GraphManager selects the correct instance of the function based on the integer chosen. It implements its toString method by calling the function toString methods. It receives the user-selected extents of x (left and right) and the grid width in pixels and computes the extents of f(x) (minimum and maximum). To do this it computes values of f(x) over the extents of x from the left to the right extent in increments of one pixel (i goes from 1 to gridWidth) in order to determine the maximum and minimum extent of f(x)

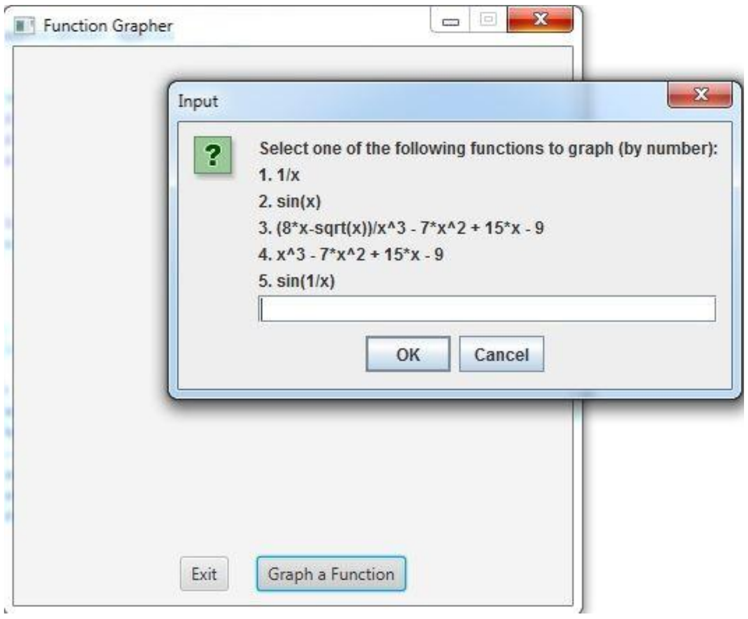
Then the GraphManager uses the selected function instance and calls the function's method fnValueToPlot inherited from the Function class to compute the value of the function within the coordinate system of the panel. This value is returned to the GUI to be plotted.

GUI Driver

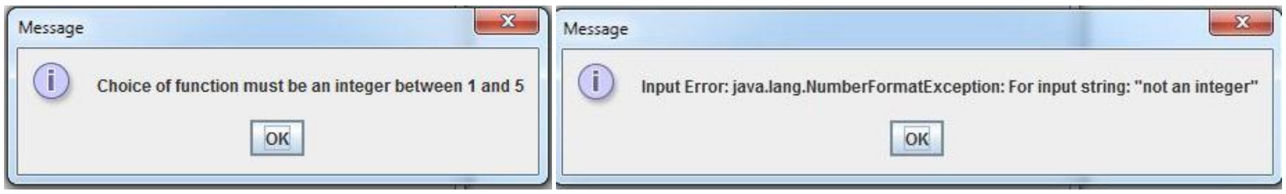
- At start-up, a blank panel displays with an Exit Button and a Graph Function button.



- The GUI creates and uses a GraphManager object.
- A JOptionPane is presented to prompt the user to choose one of the functions. The GraphManager's toString method is used to display the possible functions. If the user enters a non-integer, the exception will be caught and displayed, and will prompt for another entry. If the user enters an integer outside the range of 1 to 5, the user will be prompted for another entry until the user enters a valid integer. Once the user enters a valid integer, the GUI calls the GraphManager's setFunctionChoice method.



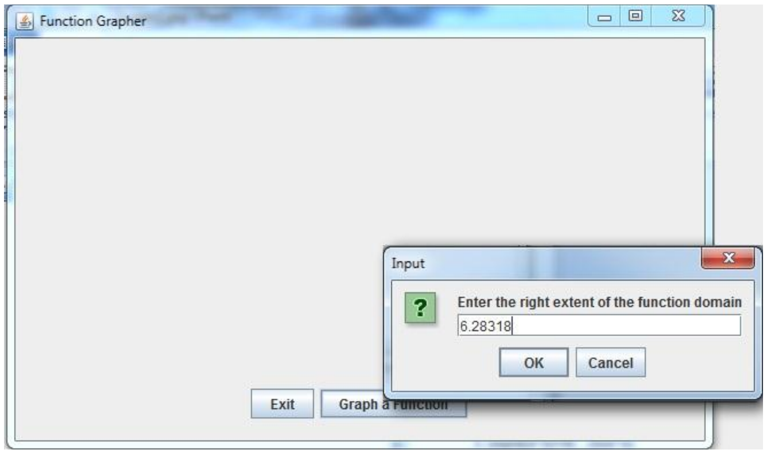
Select one of the following functions to graph (by number):
1. 1/x
2. sin(x)
3. (8*x-sqrt(x))/x^3 - 7*x^2 + 15*x - 9
4. x^3 - 7*x^2 + 15*x - 9
5. sin(1/x)



Choice of function must be an integer between 1 and 5

Input Error: java.lang.NumberFormatException: For input string: "not an integer"

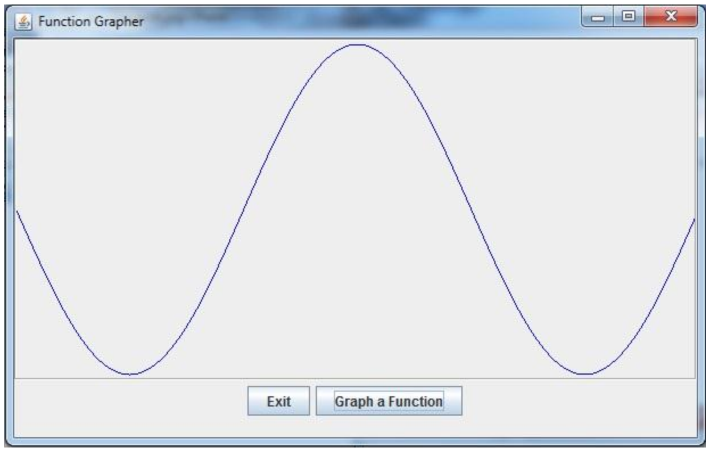
- A JOptionPane is used to select the left and right extents of x (i.e., the minimum and maximum values of x). Exceptions will be thrown and caught if the user enters a non-number, error messages will be printed if an exception is thrown, and the user will be prompted for another entry until a valid number is entered. Once the user enters a valid number, the GUI calls the GraphManager's setExtents method to store the left and right extents and computes the maximum and minimum extents of the chosen function.



- When the user selects the “Graph a Function” button and enters valid choices, the GUI calls the drawGraph() method. The Pane that holds the graphs is a Canvas class. It’s method getGraphicsContext2D() is used to get an object with which to draw on the Canvas. For a GraphicsContext instance called gc, use the following method to draw a line segment between the two points (x1,y1) and (x0,y0).

  gc.strokeLine(x0, y0, x1, y1);

  The draw method iterates through each pixel from i=0 to i=gridWidth, and using the function instance retrieved from the GraphManager, calls the method fnValueToPlot inherited from the Function class to display the function within the coordinate system of the panel. It uses the above draw method to plot a line segment from x0=i to x1= i+1.



- The user will be allowed to repeat the process for the same or another function.

TASK #0 – Study the UML Class Diagram and the code.

Familiarize yourself with the code. Follow it from the class GraphDriverFX and its method “start()”, to MainPaneFX, to GraphPanelFX and GraphManager. Run it and observe its behavior. It will be incomplete when you first run it.

TASK #1 – Implement the Line-drawing

When you first run the application, the graph will not be plotted, although the prompts will appear and be handled. In the method drawGraph(double, double, GraphicsContext), add a call to gc.strokeLine(x0Draw, y1Draw, x1Draw, y0Draw) to plot each line segment of the graph.

Experiment with all three functions provided, varying left and right extents. Notice the gray horizontal line that is plotted at y=0 for reference.

TASK #2 – Add another function

Note that function #3 is a complicated, fairly unpredictable function. Run it with several extents, and note the singularities, where the graph goes to “infinity”. Then notice that the denominator of function #3 is just a cubic polynomial.

Implement the denominator of function #3 as function #4. Run function #4 to estimate where the function goes to zero, which will of course make function #3 undefined. Include in your writeup an estimate of the point or points at which function #4 goes to zero.

TASK #3 – Add a fifth function

Very interesting behavior happens with the function f(x)=sin(1/x). Implement it, and run it with various extents. Include in your writeup what you think is happening with the function. At what value of x does the limit not exist?

Turn in the following:

The corrected java src directory in a compressed (zip or tar) file

A document with comments from Task #2 and #3