## Criterion C: Development

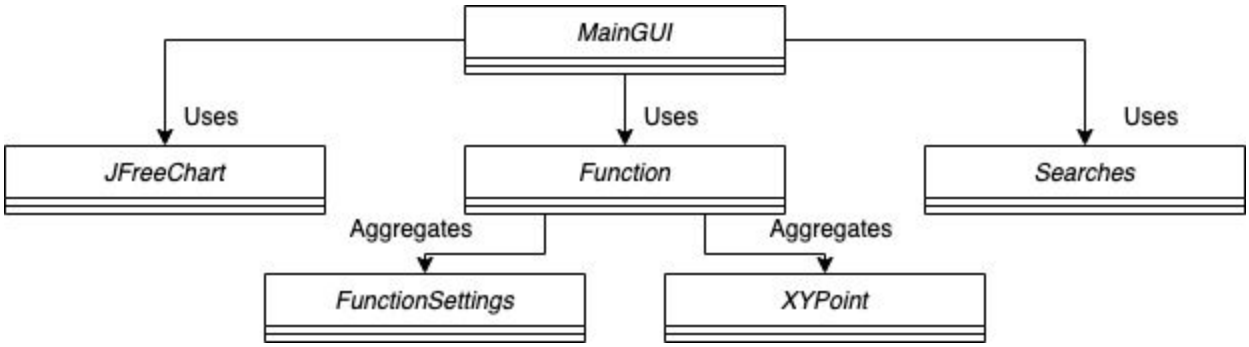
### Introduction

I have used the java development environment Netbeans to develop my program and have used Java's Swing tools along with JFreeChart, a Java chart library, to make my Graphical User Interface. My program customizes and displays several types of mathematical functions and their transformations through a table and a graph. The main unique feature requested by the client is a slide bar that adjusts the transformations on a continuously refreshing graph to make it appear animated.

### Summary List of All Techniques

- parameter passing
- user defined objects made from an OOP “template” class
- encapsulation of private methods that work on public attribute of a "template" class
- making an array of objects
- simple and compound selection (if/else)
- linear searching
- error handling (for example catching a divide by 0 error, or a null pointer while using an array of object)
- GUI tabs
- overloaded constructors, which work differently depending on the parameters sent
- try and catch
- use of Jfreechart specialized chart library for xyLineGraph
- arrayLists
- key pressed events
- enter key detection

### Structure of the Program



What?

My program uses OOP to create, edit, and display an array of functions. The main class *uses* all the java swing and jFreeChart elements on the GUI. The function class is an object composed of several data types and *aggregates* two other objects. It *aggregates* the functionSettings class, which stores settings from the Table and Settings tab, and the points class, which stores values for custom point entry. The main also *uses* the searching class for searching through objects for certain data.

Why?

My program displays a large amount of data points that are often changing continuously so it made sense for me to not waste space storing all the points. Most of my function types are mathematical calculations except for the specifically requested custom point entry which I use an arrayList database for. For the rest of the program I needed multiple instances of the same object so I created several template classes allowing me to utilize the OOP features of *abstraction*, *encapsulation*, and *polymorphism*.

- Abstraction: Functions are very complex and abstraction allows me to focus on only a few aspects.
- Encapsulation: It is when the attributes of a class are made private and can only be accessed through public get and set methods of that class.
- Polymorphism: It is when one can perform a single action in different ways with method overloading and overriding.

All of these OOP features facilitate easier debugging, quick maintenance, modularity, code reusability, and extensibility.

**Data Structures Used**

Array of Objects: I was working with multiple instances of the function object so I put them into an array for direct access to them and efficient searching. The static data type was appropriate because the user would not need to have that many functions simultaneously.

ArrayLists: One of the elements of the function object is an arrayList of points which stores all of the points that the user can enter. A dynamic data type is best because the user could want any number of data points. It does not need to be searched or sorted so an arrayList is ideal.

**Main Unique Algorithms**

The refreshGraph() method seen below is the method that interacts with jFreeChart library GUI element XYLineChart. It is called whenever the display tab is open and any element of the currently selected function is changed. This means that when any slide bar is being used, it allows for continuous data entry and continuous calling of this method resulting in the graph's apparent animation. The function creates an XYLineChart to certain specifications including the size, appearance, colors, and thickness of the chart, axes, and functions. It calls functionDataset() to get the dataset it displays.

```
281     public void refreshGraph(){
282         JFreeChart xylineChart = ChartFactory.createXYLineChart(
283                 "Graph",
284                 "X-Axis", "Y-Axis",
285                 functionDataset(),
286                 PlotOrientation.VERTICAL,
287                 true, true, false);
288
289         ChartPanel chartPanel = new ChartPanel( xylineChart );
290         final XYPlot plot = xylineChart.getXYPlot( );
291         graph.removeAll();    // clear panel before add new chart
292         graph.add(chartPanel, BorderLayout.CENTER);
293         graph.validate();       // refresh panel to display new chart
294         chartPanel.setPreferredSize( new java.awt.Dimension( 620 , 400 ) );
295
296         plot.getRenderer().setSeriesPaint(0, Color.GRAY); //Setting color for axes
297         plot.getRenderer().setSeriesPaint(1, Color.GRAY);
298
299         String color = functions[functionElementNumber()].getFunctionSettings().getColor();
300         if(!toggleTransformation.isSelected() && !toggleOriginal.isSelected()){
301             plot.getRenderer().setSeriesPaint(2, Color.BLACK);
302             plot.getRenderer().setSeriesStroke( 2, new BasicStroke( 1.0f ));
303         }else{
304             plot.getRenderer().setSeriesStroke( 2, new BasicStroke( 2.0f ));
305             colorFunction(2, plot, color);
306         }
307
308         plot.getRenderer().setSeriesStroke( 3, new BasicStroke( 2.0f ));
309         colorFunction(3, plot, color);
310
311         chartPanel.setDomainZoomable(true);
312         chartPanel.setRangeZoomable(true);
313     }
```

functionDataset() is a large class that has no parameter but returns the datasets that need to be displayed on the graph including the x-axis and y-axis.. Keep in mind that it only returns datasets with points completely within the x and y window bounds of the selected function.

Pseudocode:

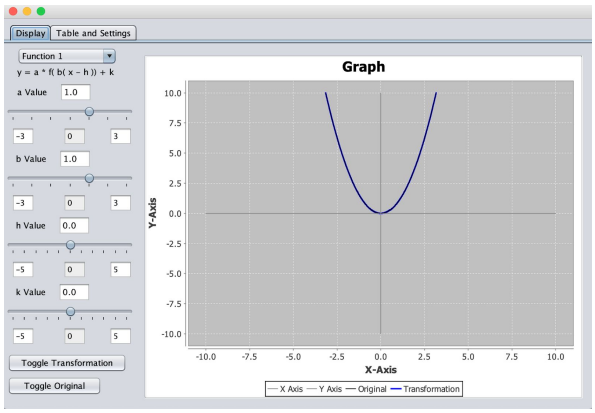
```
Private XYDataSet functionDataset(){
        Create xySeriesCollection

        Create the x and y axes lines according the x and y window minimums and maximums
        Add each axes dataset to the xySeriesCollection

        If the functionType is not custom point entry{
                If the original toggle is on{
                        Create the dataset of the function without transforming it (only using
                        calculation() method)
                        Add the original dataset to the xySeriesCollection
                }
                If the transformation toggle is on{
                        Create the dataset of the function with transforming it (using the
                        transformationCalculation() method)
                        Add the transformation dataset to the xySeriesCollection
                }
        }

        else (which means the functionType is custom point entry) {
                If the original toggle is on{
                        Creates and adds a dataset of the arrayList of points to the xySeriesCollection
                }
                If the transformation toggle is on{
                        Creates and adds a dataset of the transformations of each individual point of the
                        arrayList of points to the xySeriesCollection
                }

        Return xySeriesCollection
}
```

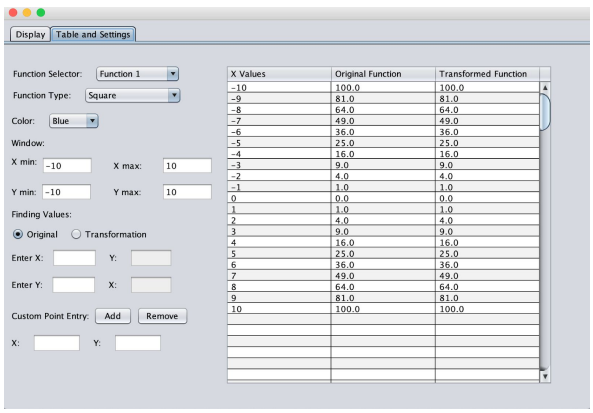
The last unique algorithm is translating the mathematical topic of transformations into code. It takes in an x value as the parameter and returns the corresponding y value for the specific functions transformation. It can be shown with $y = a * f(b(x - h)) + k$. It calls the method calculationNum() which gets the function's type such as square root and applies that base calculation f(x).

```
175     public double transformationCalculation(double num){
176         double a = functions[functionElementNumber()].getA();
177         double b = functions[functionElementNumber()].getB();
178         double h = functions[functionElementNumber()].getH();
179         double k = functions[functionElementNumber()].getK();
180         num -= h;
181         num *= b;
182         double answer = calculation(num);
183         answer *= a;
184         answer += k;
185         return answer;
186     }
```

## Graphical User Interface Work

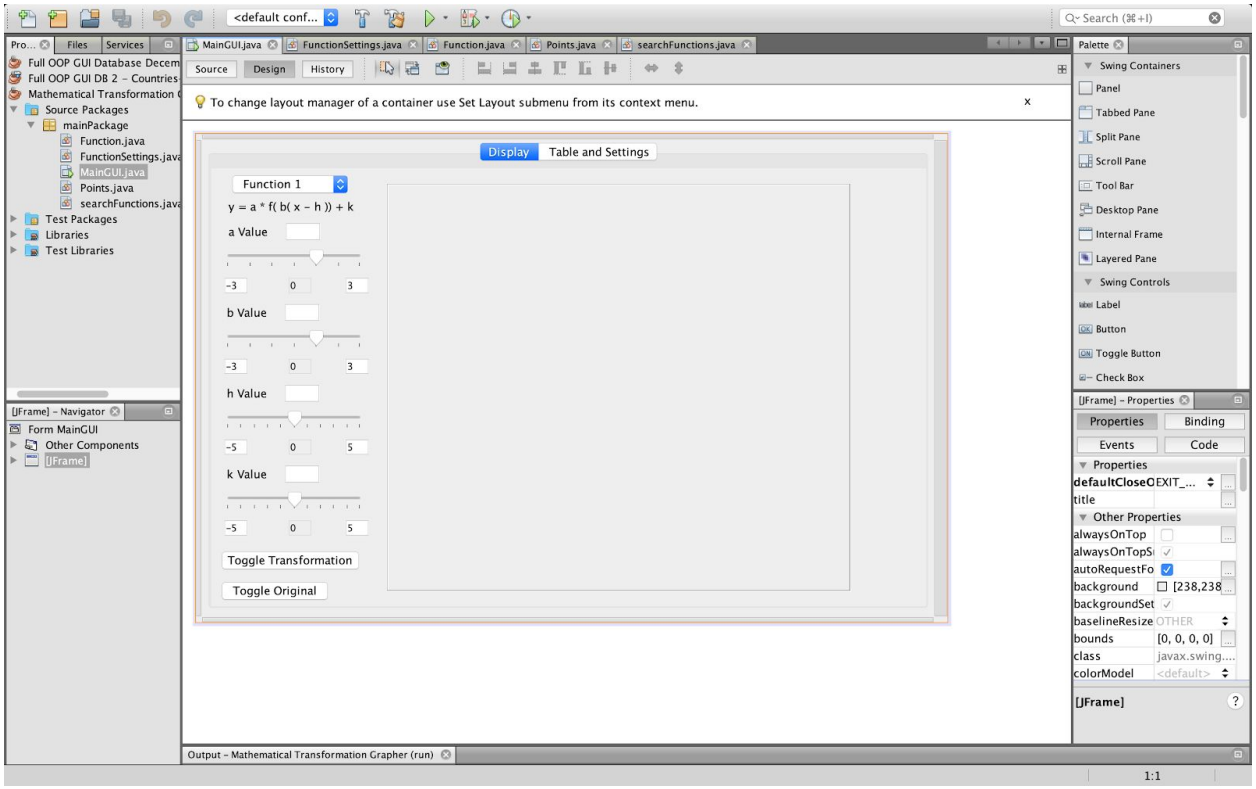
My GUI, seen below, provides a way for the program to take different inputs from the user and display the information the program was meant for. My Display tab opens a panel with several java swing controls and the XYLineGraph. My Table and Settings tab opens a panel with more java swing controls and a table displaying the specific points of the selected function. The function selector combo boxes at the top of each paneselect the function that is being used. Another important GUI element is the slider which was specifically requested by my client for its ability to input many numbers into the program quickly.



**Software Tools Used**

This program uses the NetBeans IDE which is ideal because it allows me to work using OOP and classes. It also has a Java library containing useful tools, such as java.util.ArrayList and lets me use many different GUI elements like slide bars. The design tab used to work on my mainGUI class provides me with a palette of many GUI elements and the means to edit them.




**Criterion C Word Count: 911**