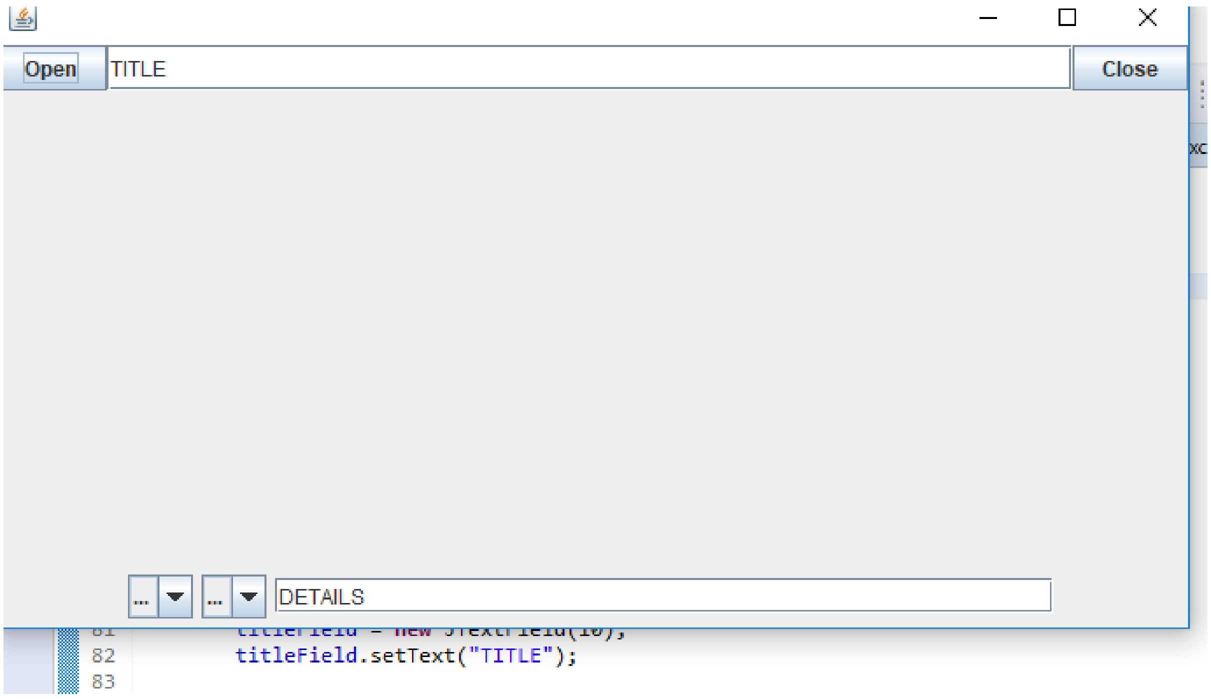
Programming Assessed Exercise 3

Stuart Rawlinson – 2376095

The program meets each of the requirements outlined in the brief.

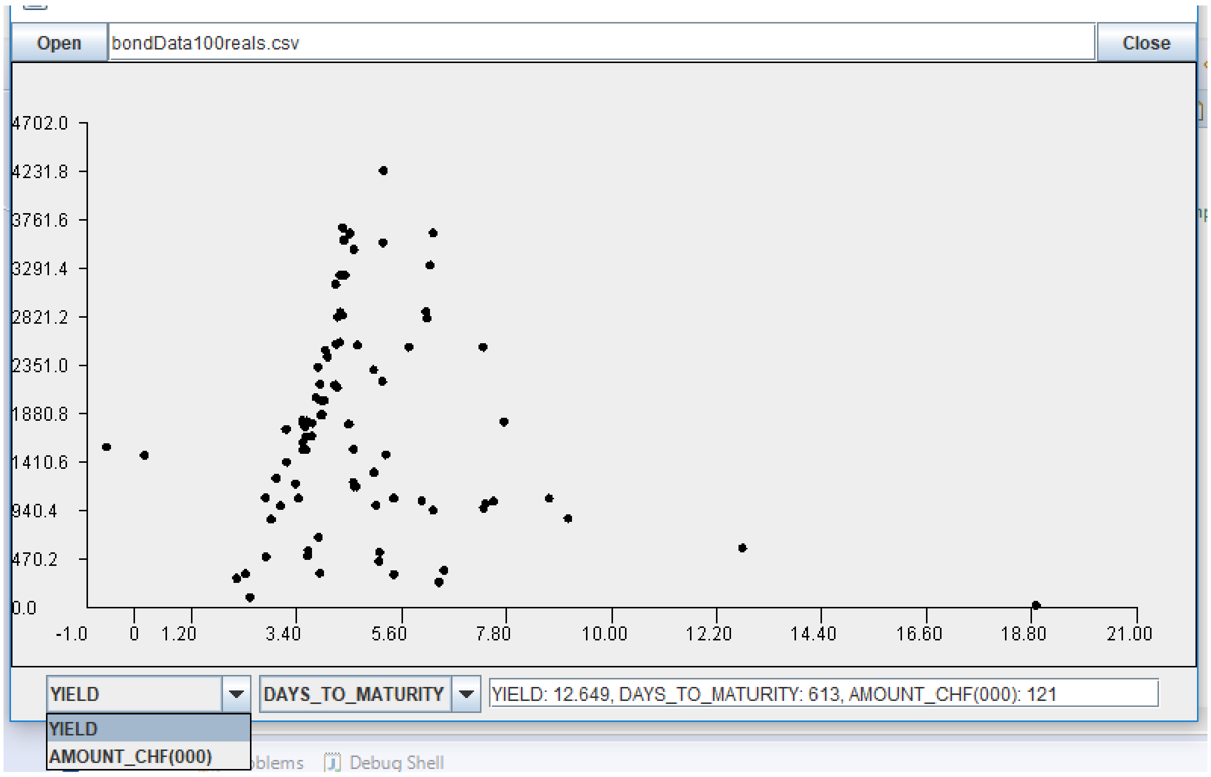


TITLE

DETAILS

```
        titleField = new JTextField(10);
        titleField.setText("TITLE");
```

Above is an example of the GUI when the user runs the program. Upon reading the correct file the Graph draws accurately and each combo box has the option to change to the third variable:

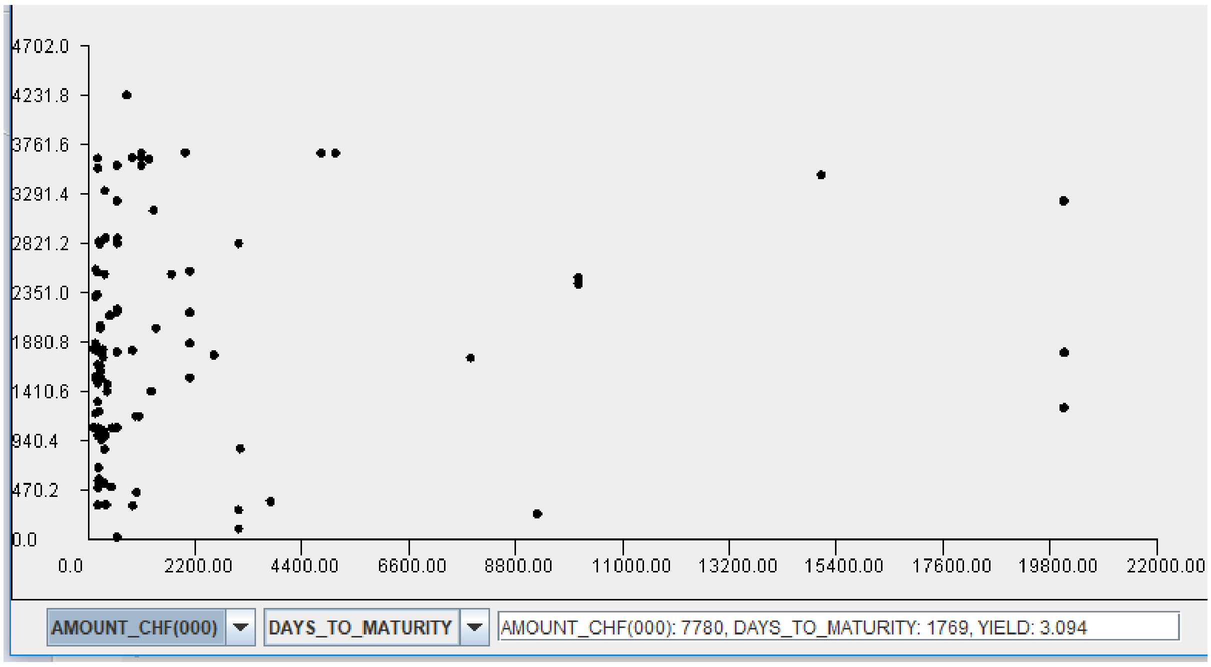


bondData100reals.csv

YIELD: 12.649, DAYS_TO_MATURITY: 613, AMOUNT_CHF(000): 121

As seen above, the program displays ten even increments on the scale between the minimum value of the scale and the maximum value of the scale – an additional 10% was added to the maximum and minimum (if below zero) values found in the data to make for a clearer graph.

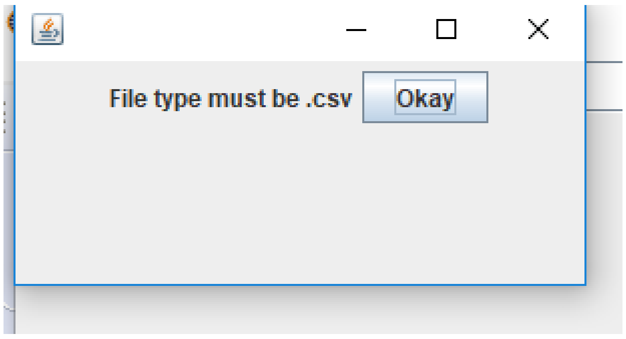
When the AMOUNT_CHF value is chosen for the x axis, the graph changes accordingly. At all times the user can click on a point in the graph to display the data associated with the selected bond trade.

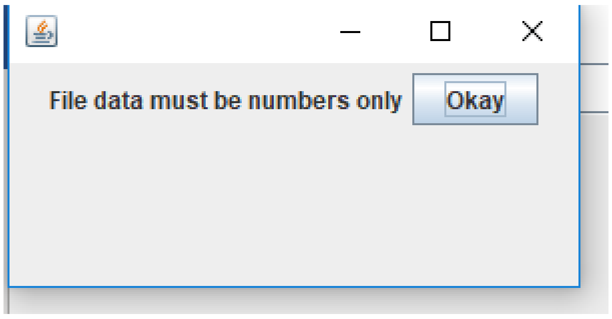


AMOUNT_CHF(000): 7780, DAYS_TO_MATURITY: 1769, YIELD: 3.094

The variables associated with each axis can be changed between the given inputs indefinitely. Similarly, clicking on a new point on the scatter plot graph updates the details given any number of times while the program is running.

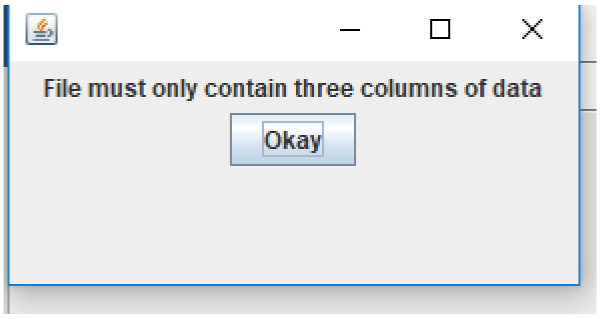
It was assumed that the program should exclusively handle csv files and only ones containing numerical data. Some exception classes were designed with this in mind. The program displays a pop-up notification to the user if they attempt to use the wrong type of input. Below are screenshots showing the pop-ups. When dismissed, the user can select a different file without affecting the running of the program.


File type must be .csv


File data must be numbers only

The files used to test these input exceptions of the program are included in my submission.

The program was designed with flexibility in mind, with the names of each data type deliberately kept mutable and changing with the combo box selections, rather than elements that became fixed when the file was read. This was to accommodate the original plan of using array-lists to potentially hold more than the three variable types in the sample file, turning the system into a graph-making program which can handle any number of variable types. It became apparent that this was both beyond the scope of the assignment and would require too much time to complete to my satisfaction. Instead, I implemented another exception class to ensure that the csv file used for input only held three columns of data. This ensured there were no input problems stemming the amount of data being read from the input file. This exception was handled using the same pop-up GUI class as the others:

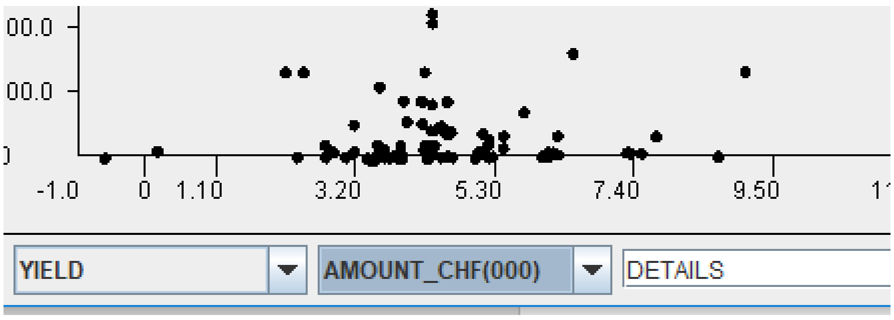

File must only contain three columns of data

The final result is that the program is flexible in that any file with three sets of numerical data can be used. The graph will generate and the user will be able to switch between the selected variables. A sample sheet of three columns of randomised numbers is included in my submission to test this.

Finally, the program also successfully exits when the ‘Close’ button is pressed.

Some deficiencies include the presence of ‘empty’ methods associated with the Mouse Listener in the Graph class. Using the Mouse Listener interface required methods such as ‘mouseExited’ etc, but no use for them could be found in the scope of this assignment.

The ellipse for each Bond object is plotted to the graph using the coordinates of the upper left corner, as in other shapes drawn through swing. The size of the ellipse means that it often has the appearance of being below the actual coordinate – particularly when the value is near or at the minimum value for the x axis, as shown below:



Accounting for this by shifting the points lead to further inaccuracies and making each ellipse smaller made it extremely difficult to click on the desired point, so I decided to leave the program as it was.