Boris Wrabel
2206447w

Assessed Exercise 3
Status Report

Summary

My program has been designed in accordance with the MVC architecture. At the top of the main window there are two buttons, open and quit, and a text field for displaying the name of an input file. The open button triggers a JFileChooser allowing user to select a CSV file. Once a file is loaded in, its name appears in the text field. The program terminates when user clicks in the quit button.

Using data from the input file the program draws a scatterplot in the centre of the main window. Each dot represents bond trade. The axes have tick marks and labels.

At the bottom of the main window there is a pair of combo boxes. They are empty by default. Next to them there is a text field for showing details of selected bond trade. When user loads file in, the headers of the input file are read into the combo boxes. When user clicks on a dot, its details, i.e the three column values, are shown in the text field. The content of the text field is updated each time user clicks on a dot.

Assumptions

To avoid situations where points with negative values would be plotted beyond scatterplot, the origin of scatterplot is determined by minimum values of selected columns. Maximum values of selected columns determine the maximum values of x and y labels.

Deficiencies

The main window of the program is not resizable. When user selects a file that is not in the CSV format there's no error message informing them of the incorrect behaviour.

Testing

In the early stages of the development to test whether the array list of bond trade objects was populated with objects, the System.out.println command was used. To test whether methods to compute minimum and maximum values in the array list worked the System.out.println command was used.

When user runs the program the main user interface window appears. As planned, the combo boxes located at the bottom of the window are empty. The text field which is placed next to them shows a sample text ‘Click a point for more info’. The text field at the top of the window shows a sample text ‘<Name of file>’. See screen dump 1. The program terminates when the quit button is clicked in.

As programmed, when user clicks in the open button the JFileChooser appears inside the main window. Screen dump 2 illustrates the behaviour. It also shows that the correct file format filter is in place.

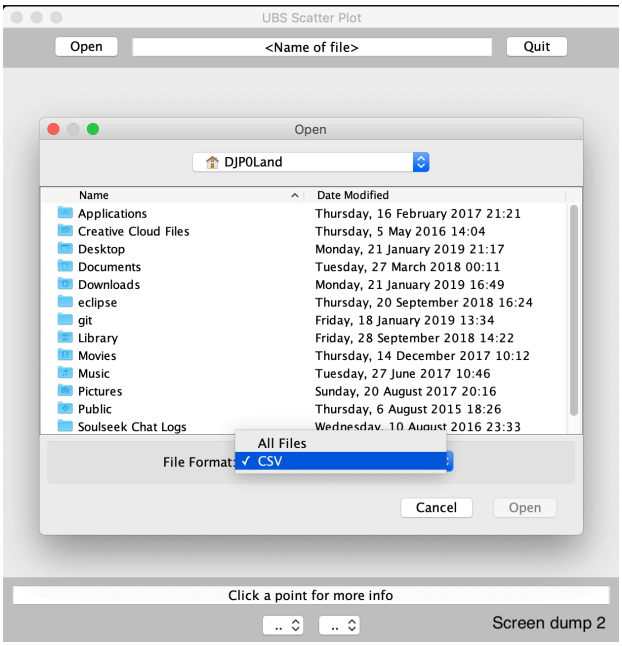
Screen dumps 3 and 4 demonstrate that the program updates the scatterplot according to the chosen file.

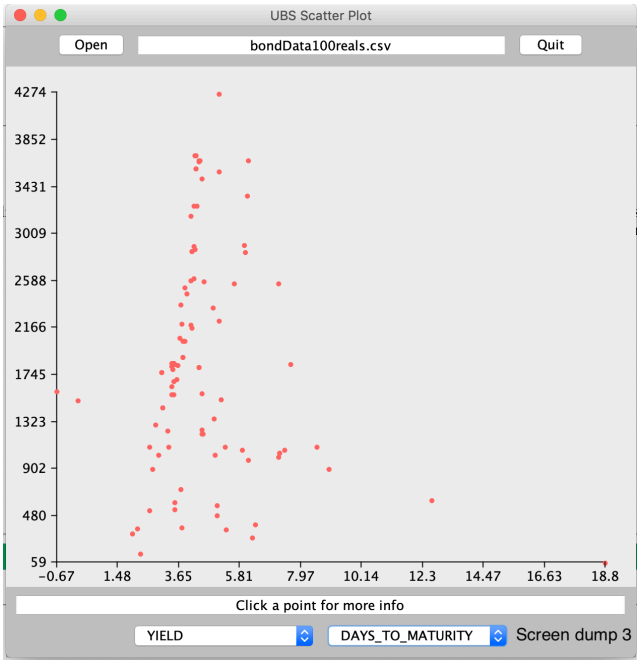
Screen dump 3 and 6 to 11 demonstrate user interactions with combo boxes: the dots representing bond trades are plotted properly according to the selected option.

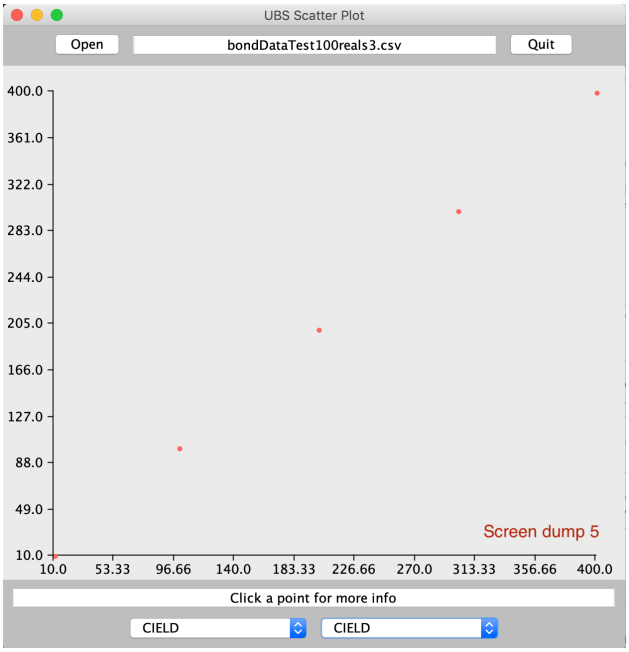
Screen dumps 12 and 13 show that when a CSV file is loaded in, the content of the text fields aforementioned changes accordingly.

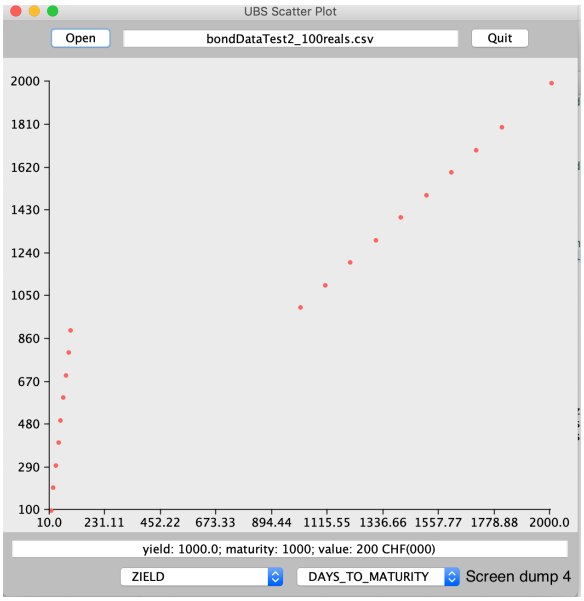
Test files with different headings were selected to show whether the content of the combo boxes changed when a new file was loaded.

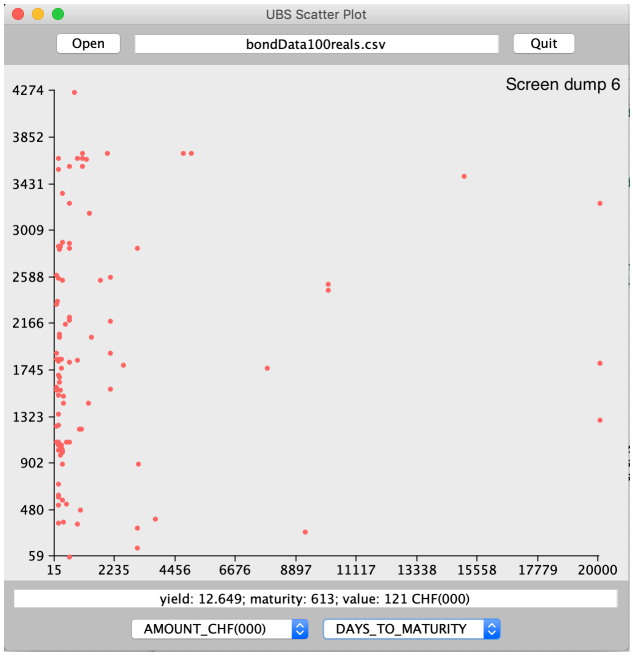

UBS Scatter Plot
Open
<Name of file>
Quit
Click a point for more info
Screen dump 1

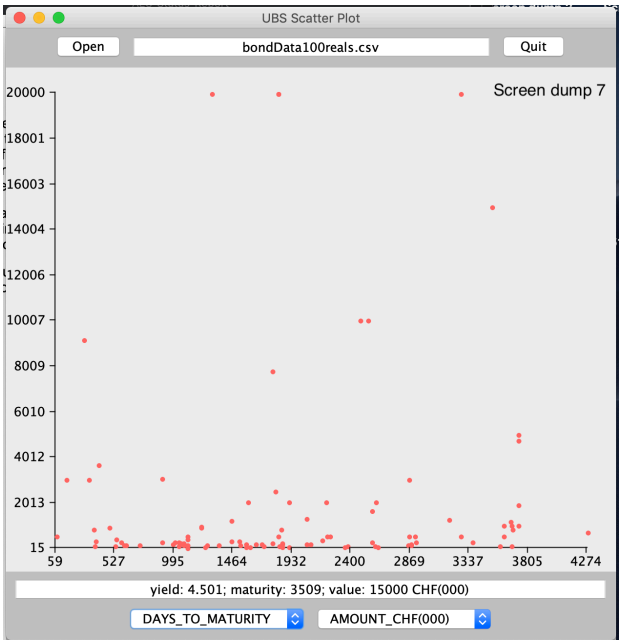

UBS Scatter Plot
Open
<Name of file>
Quit
Open
DJP0Land
Name
Date Modified
Applications
Thursday, 16 February 2017 21:21
Creative Cloud Files
Thursday, 5 May 2016 14:04
Desktop
Monday, 21 January 2019 21:17
Documents
Tuesday, 27 March 2018 00:11
Downloads
Monday, 21 January 2019 16:49
eclipse
Thursday, 20 September 2018 16:24
git
Friday, 18 January 2019 13:34
Library
Friday, 28 September 2018 14:22
Movies
Thursday, 14 December 2017 10:12
Music
Tuesday, 27 June 2017 10:46
Pictures
Sunday, 20 August 2017 20:16
Public
Thursday, 6 August 2015 18:26
Soulseek Chat Logs
All Files
CSV
Cancel
Open
Click a point for more info
Screen dump 2

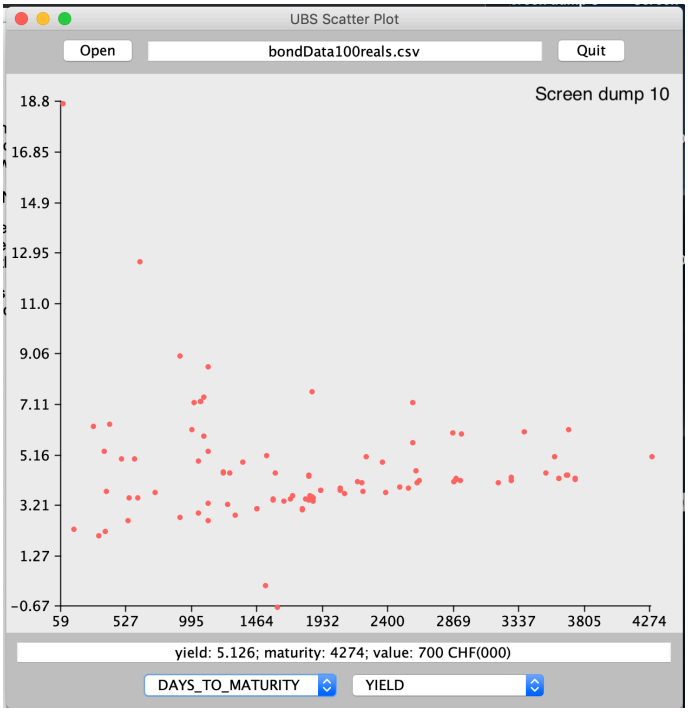

UBS Scatter Plot
Open
bondData100reals.csv
Quit
4274
3852
3431
3009
2588
2166
1745
1323
902
480
59
-0.67
1.48
3.65
5.81
7.97
10.14
12.3
14.47
16.63
18.8
Click a point for more info
YIELD
DAYS_TO_MATURITY
Screen dump 3

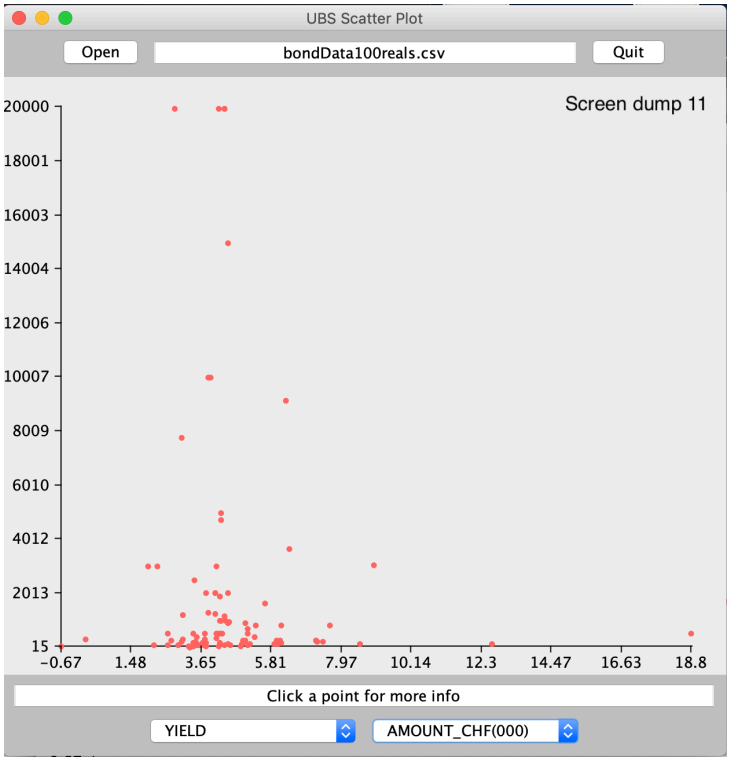

UBS Scatter Plot
Open
bondDataTest100reals3.csv
Quit
400.0
361.0
322.0
283.0
244.0
205.0
166.0
127.0
88.0
49.0
10.0
10.0
53.33
96.66
140.0
183.33
226.66
270.0
313.33
356.66
400.0
Screen dump 5
Click a point for more info
CIELD
CIELD

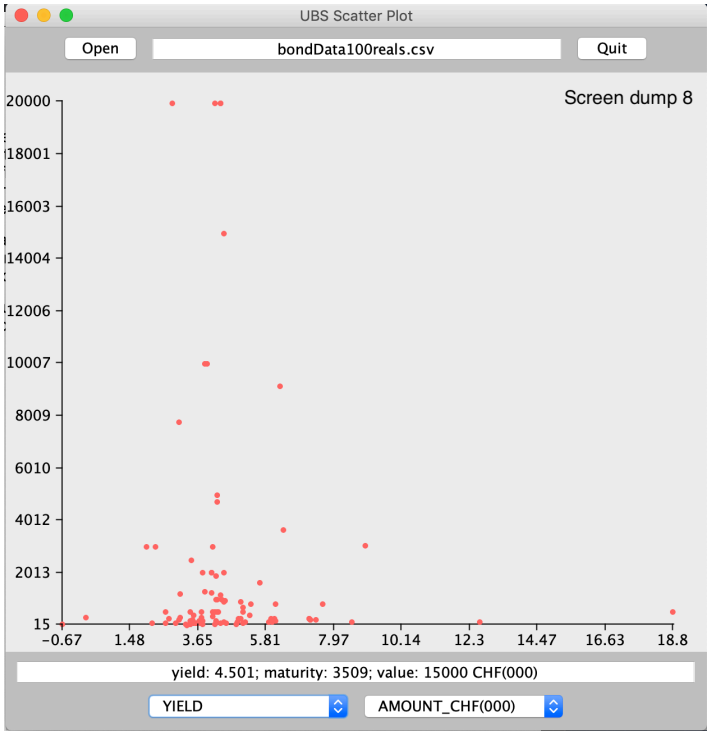

UBS Scatter Plot
Open
bondDataTest2_100reals.csv
Quit
2000
1810
1620
1430
1240
1050
860
670
480
290
100
10.0
231.11
452.22
673.33
894.44
1115.55
1336.66
1557.77
1778.88
2000.0
yield: 1000.0; maturity: 1000; value: 200 CHF(000)
ZIELD
DAYS_TO_MATURITY
Screen dump 4

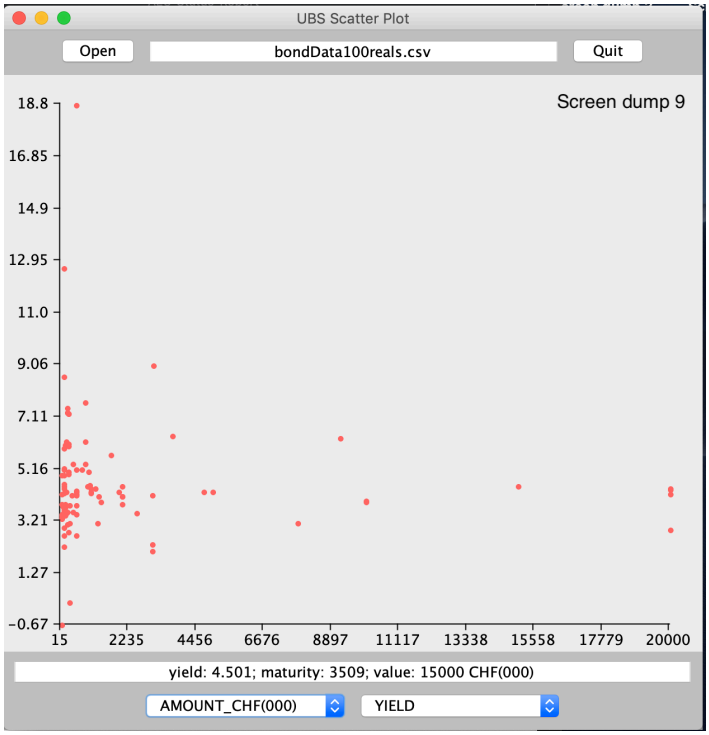

UBS Scatter Plot
Open
bondData100reals.csv
Quit
Screen dump 6
4274
3852
3431
3009
2588
2166
1745
1323
902
480
59
15
2235
4456
6676
8897
11117
13338
15558
17779
20000
yield: 12.649; maturity: 613; value: 121 CHF(000)
AMOUNT_CHF(000)
DAYS_TO_MATURITY

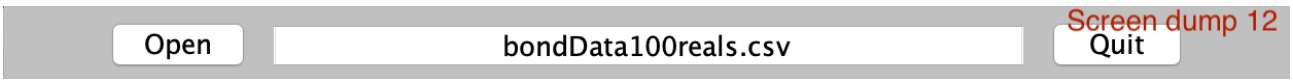

UBS Scatter Plot
Open
bondData100reals.csv
Quit
Screen dump 7
20000
18001
16003
14004
12006
10007
8009
6010
4012
2013
15
59
527
995
1464
1932
2400
2869
3337
3805
4274
yield: 4.501; maturity: 3509; value: 15000 CHF(000)
DAYS_TO_MATURITY
AMOUNT_CHF(000)

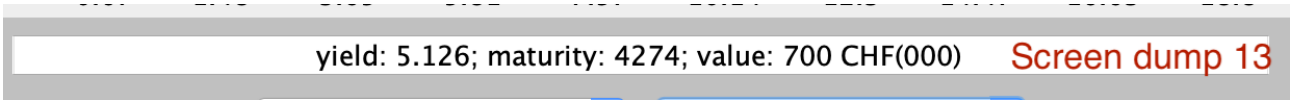

UBS Scatter Plot
Open
bondData100reals.csv
Quit
Screen dump 10
18.8
16.85
14.9
12.95
11.0
9.06
7.11
5.16
3.21
1.27
-0.67
59
527
995
1464
1932
2400
2869
3337
3805
4274
yield: 5.126; maturity: 4274; value: 700 CHF(000)
DAYS_TO_MATURITY
YIELD

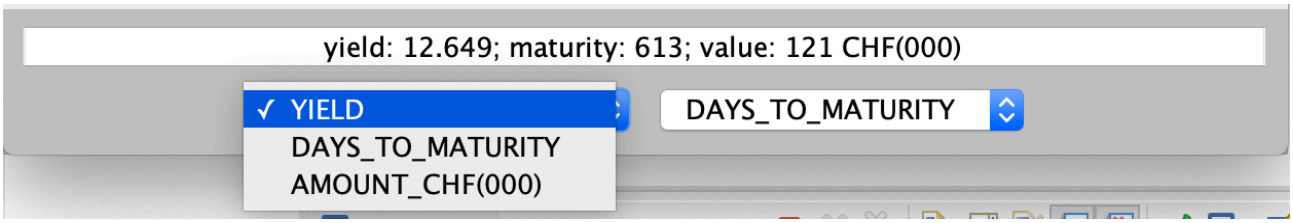

UBS Scatter Plot
Open
bondData100reals.csv
Quit
Screen dump 11
20000
18001
16003
14004
12006
10007
8009
6010
4012
2013
15
-0.67
1.48
3.65
5.81
7.97
10.14
12.3
14.47
16.63
18.8
Click a point for more info
YIELD
AMOUNT_CHF(000)

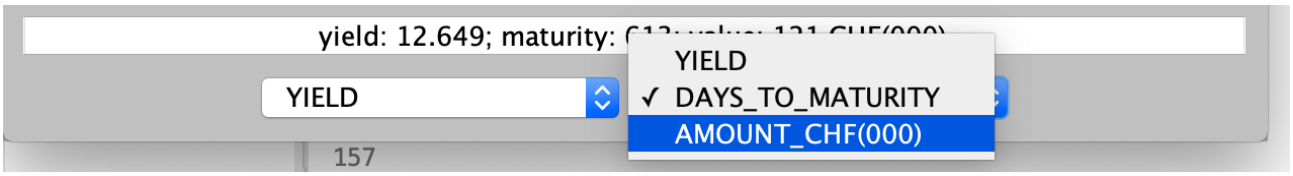

UBS Scatter Plot
Open
bondData100reals.csv
Quit
Screen dump 8
20000
18001
16003
14004
12006
10007
8009
6010
4012
2013
15
-0.67
1.48
3.65
5.81
7.97
10.14
12.3
14.47
16.63
18.8
yield: 4.501; maturity: 3509; value: 15000 CHF(000)
YIELD
AMOUNT_CHF(000)

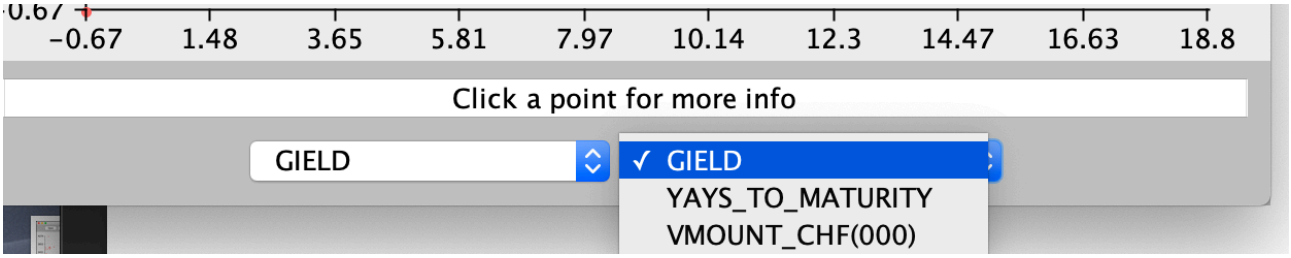

UBS Scatter Plot
Open
bondData100reals.csv
Quit
Screen dump 9
18.8
16.85
14.9
12.95
11.0
9.06
7.11
5.16
3.21
1.27
-0.67
15
2235
4456
6676
8897
11117
13338
15558
17779
20000
yield: 4.501; maturity: 3509; value: 15000 CHF(000)
AMOUNT_CHF(000)
YIELD


Open
bondData100reals.csv
Quit
Screen dump 12


yield: 5.126; maturity: 4274; value: 700 CHF(000)
Screen dump 13


yield: 12.649; maturity: 613; value: 121 CHF(000)
YIELD
DAYS_TO_MATURITY
AMOUNT_CHF(000)
DAYS_TO_MATURITY


YIELD
YIELD
DAYS_TO_MATURITY
AMOUNT_CHF(000)
157




```
// method for getting maximum value in the bondTradeList array list.
public double getMaxValue (int comboBoxSelection) {

    double maxValue = 0;
    BondTrade elementMaxYield, elementMaxMaturity, elementMaxAmount;

        if(comboBoxSelection == 0) {
            elementMaxYield = Collections.max(bondTradeList, Comparator.
            maxValue = elementMaxYield.getYield();
            System.out.println(maxValue);
        }
        else if(comboBoxSelection == 1) {
            elementMaxMaturity = Collections.max(bondTradeList, Comparat
            maxValue = elementMaxMaturity.getDaysToMaturity();
            System.out.println(maxValue);
        }
        else if (comboBoxSelection == 2) {
            elementMaxAmount = Collections.max(bondTradeList, Comparator
            maxValue = elementMaxAmount.getAmount();
            System.out.println(maxValue);
        }
    return maxValue;
}
```

```
// method for getting minimum value in the bondTradeList array list.
public double getMinValue (int comboBoxSelection) {

    double minValue = 0;
    BondTrade elementMinYield, elementMinMaturity, elementMinAmount;

        if(comboBoxSelection == 0) {
            elementMinYield = Collections.min(bondTradeList, Comparator.comparingDouble(BondTrade::getYield));
            minValue = elementMinYield.getYield();
            System.out.println(minValue);
        }
        else if(comboBoxSelection == 1) {
            elementMinMaturity = Collections.min(bondTradeList, Comparator.comparingInt(BondTrade::getDaysToMaturity))
            minValue = elementMinMaturity.getDaysToMaturity();
            System.out.println(minValue);
        }
        else if (comboBoxSelection == 2) {
            elementMinAmount = Collections.min(bondTradeList, Comparator.comparingInt(BondTrade::getAmount));
            minValue = elementMinAmount.getAmount();
            System.out.println(minValue);
        }
    return minValue;
}
```

```
// read from line 2 and populate the bondTradeList array list
while(scanningSelectedFile.hasNextLine()) {
    line = scanningSelectedFile.nextLine();
    tokens = line.split(",");

    // parsing string to double and it so that an instance of the bond trad
    double yield = Double.parseDouble(tokens[0]);
    int daysToMaturity = Integer.parseInt(tokens[1]);
    int amount = Integer.parseInt(tokens[2]);

    // create a bond trade object
    BondTrade bondTradeObject = new BondTrade(yield, daysToMaturity, amount
    // add bond trade objects into the bondTradeList arrayList
    model.getBondTradeList().add(bondTradeObject);
}
System.out.println(model.getBondTradeList().toString());
scanningSelectedFile.close(); // close the scanner
```