Documentation

Rhyder Quinlan

Student Number
C00223030

Module
Object Oriented Software Development

Lecturer
Jason Barron

Table of Contents

Introduction

The following documentation will cover the fundamental information on an assignment given by Jason Barron. The aim of the assignment was to build a functioning calculator application written in Java. Some key points to note are:

- The assignment descriptor required a Graphical User Interface (GUI), with this in mind I chose to use the Package java.swing.
- The IDE I chose was Eclipse, mainly due to it's very effective design interface using WindowBuilder. Screenshots of this interface can be found further in the document.

The assignment questionnaire laid out multiple requirements needed in the calculator, showing grading in a level type format. Please see the requirements section of this documentation.

Requirements

The most basic requirements laid out by the assignment questionnaire were simply full functionality of:

- Addition
- Subtraction
- Division
- Multiplication

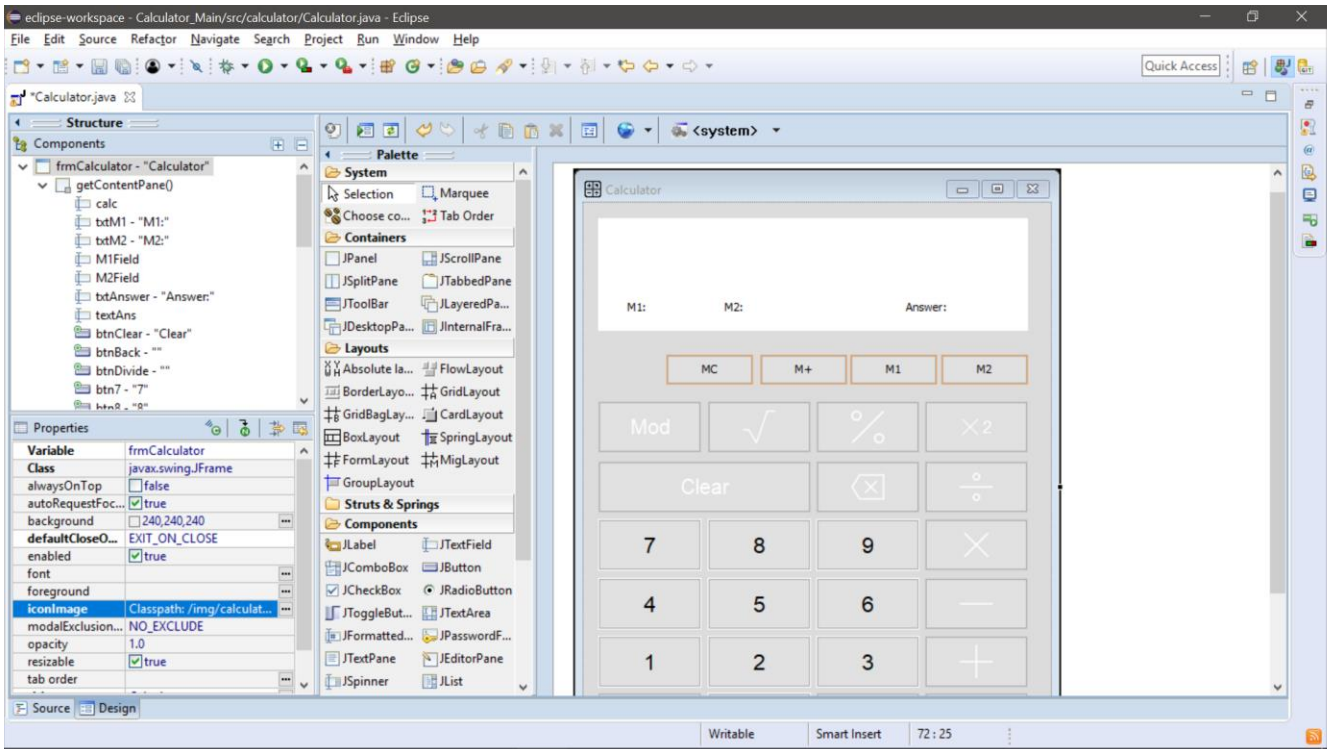
The requirement criteria for a higher mark were:

- Ability to clear the screen
- Ability to save numbers to memory
- Error handling

Other requirements vital to the project were a well designed user interface, correct use of java convention and written-style as well as commenting for the benefit of future users.

Project Development

As previously mentioned, the project development process was done in Eclipse IDE version Oxygen.3 Release (4.7.3). A tool I found most useful was WindowBuilder that offered me a drag-and-drop style interface for designing and implementing my GUI. The following screenshot is taken from this interface:



```
frmCalculator - "Calculator"
  getContentPane()
    calc
    txtM1 - "M1:"
    txtM2 - "M2:"
    M1Field
    M2Field
    txtAnswer - "Answer:"
    textAns
    btnClear - "Clear"
    btnBack - ""
    btnDivide - ""
    btn7 - "7"
```

Properties

Variable	frmCalculator
Class	javax.swing.JFrame
alwaysOnTop	false
autoRequestFoc...	✓ true
background	240,240,240
defaultCloseO...	EXIT_ON_CLOSE
enabled	✓ true
font	
foreground	
iconImage	Classpath: /img/calculat...
modalExclusion...	NO_EXCLUDE
opacity	1.0
resizable	✓ true
tab order	

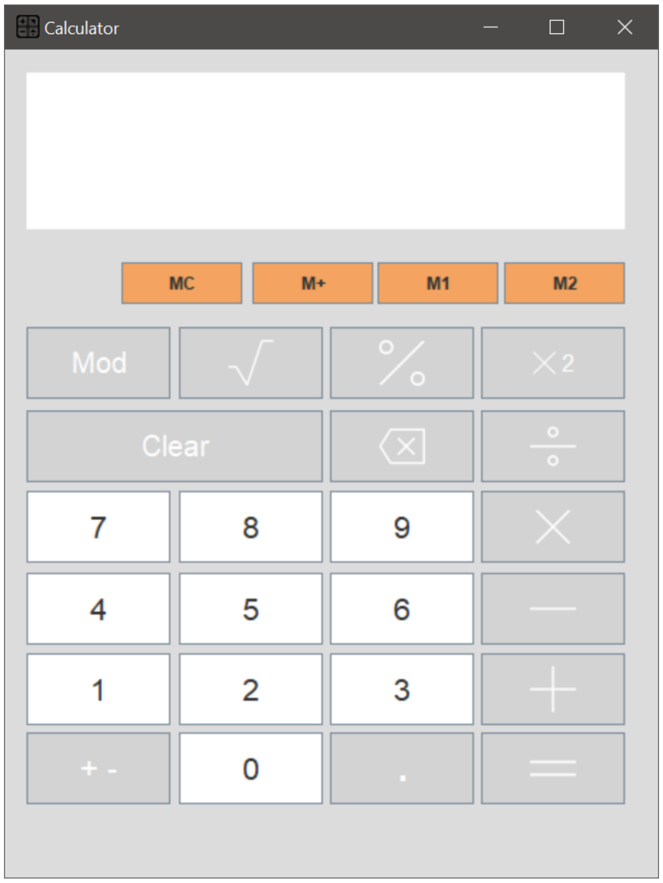
In terms of coding the functionality of the calculator I found Eclipse's vast assistive tools to be very helpful, these include syntax checking, assistive tips etc. This made code less error prone, as well as quicker to develop with Eclipse's extensive plug-in and shortcut capabilities.

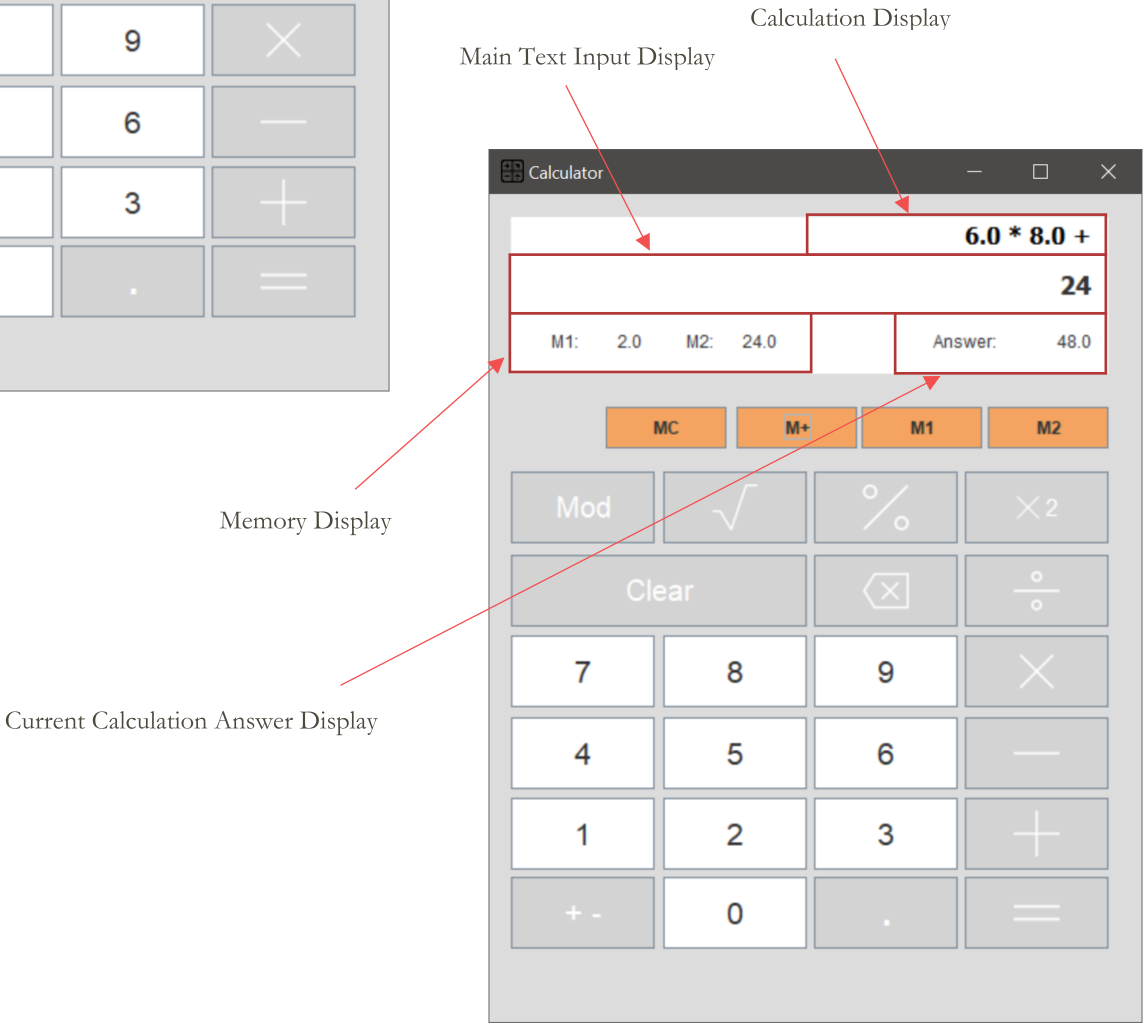
During development I relied strongly on the Git platform GitHub. This was due to working on the assignment both at Institute of Technology Carlow and at home. The Git repository can be found at:

https://github.com/rhyderQuinlan/Calculator_Main

Graphic User Interface

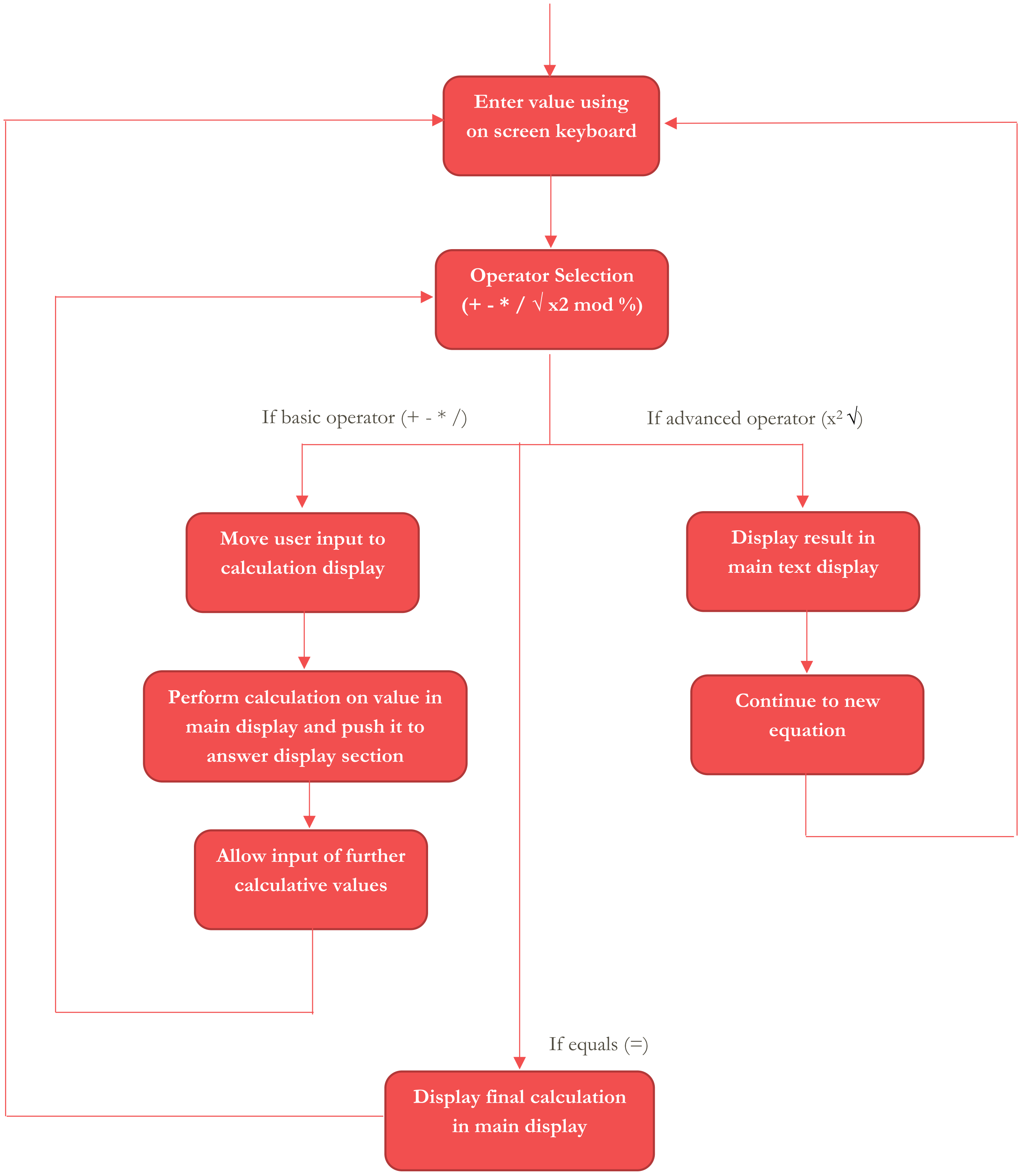
All development of the GUI was completed using WindowBuilder, which automatically generates relevant code such as variable identifier and initialization, event handling and code structuring. Here is the final design:





Calculation logic

In terms of the functionality of the calculator, the flow of how accurate arithmetic was calculated is as follows:


Enter value using on screen keyboard
Operator Selection (+ - * / √ x2 mod %)
If basic operator (+ - * /)
If advanced operator (x² √)
Move user input to calculation display
Perform calculation on value in main display and push it to answer display section
Allow input of further calculative values
Display result in main text display
Continue to new equation
If equals (=)
Display final calculation in main display

The following methods are for the basic functions of addition, subtraction, division, multiplication and modular. These functions use a variable operatorCount to keep track of the number of operations. Opposed to the square root, squared etc functions that do not use a counter.

In the following switch function, firstNumber represents the previously entered/calculated part of the equation and number represents the current number entered that is to be calculated by the existing equation.

```
switch (operatorStorage) {
	case "+":
		result = firstNumber + number;
		break;
	case "-":
		result = firstNumber - number;
		break;
	case "/":
		result = firstNumber / number;
		break;
	case "*":
		result = firstNumber * number;
		break;
	case "%":
		result = firstNumber % number;
		break;
}
```

The square root function is calculated using Java's Math.sqrt function. The x^2 function is calculated by multiply the value currently on display by itself.

Memory function

The memory function is a common feature on calculators, however often works differently on varying types. In this calculator the "M+" button will store whatever value is currently on display into the next chosen memory slot. The slots are chosen by alternating between M1 and M2. So if the next memory slot to be written into in M1, once M+ is clicked then the value will be pushed into M1, then M2 will become the next slot to be written to. "MC" stands for Memory Clear, and does exactly that. Emptying all memory slots.

Specifications & Functionality

The final .jar file has the following functionality:

Basic

- Addition
- Subtraction
- Multiplication
- Division
- Equals

Intermediate

- Memory Storage
- Screen Clear
- Error Handling

Extras

- Mod function
- Square Root Function
- Squared Function
- Backspace
- Inverse

One attribute of this calculator is if the user wants to enter in multiple values he/she does not need to keep pressing equals to see the current answer. The current answer of all calculations is displayed in the bottom right of the Display Panel.

Testing

I underwent some basic use cases to test that when a user tries certain operations the program can react correctly.

A side note is the system was tested on Microsoft Windows OS, this being the main target audience.

Test No.	Objective	Case (Input)	Expected Result	Actual Result	Date	Status
1	Test Addition	12+13.1+9.7	34.80	34.80	9/04/2018	Successful
2	Test Subtraction	50-46.8-0.4	2.80	2.80	9/04/2018	Successful
3	Test Multiplication	9.7*12*2	232.8	232.80	9/04/2018	Successful
4	Test Division	4/12/0.05	0.67	0.67	9/04/2018	Successful
5	Test Square Root	Root 144	12	12	9/04/2018	Successful
6	Test Squared Function	$(36)^2$	1296	1296	9/04/2018	Successful
7	Test Decimal Point can be entered only once	Double click '.'	*.***	*.***	9/04/2018	Successful
8	Complex Calculation	$(6)^2+4$	40	40	9/04/2018	Successful

Error Log

An error is an unlimited amount of zeroes can be added before the decimal point, this does not cause an arithmetic error, however is not a normal feature seen on calculators. I have found this error late and will not have time to fix.

Most errors I had to deal with evolved around Empty String errors, to the best of my knowledge all have been rectified.

Conclusion

The calculator functions well in many tests I underwent.

The final product from this development process is a .jar file, as this runs well when requiring cross-platform distribution.