# Institute of Technology, Carlow

## Year 2 Programming

## Java Calculator

## GUI

## (CA3)

**Name:** James Hall c00007006

**Date:** 27/01/2019

**Tutor:** Jason Barron

# Contents

# Introduction

The aim of this report is to explain the development of a calculator application written in Java using a Graphical User Interface (GUI). This report will contain:

•a thorough description of the problem,

•an image of the GUI.

•a copy of the Java code & executable jar file,

•a description of all the functions/routines which have been used, and

•Test data used and sample execution screen shots of outputs produced.

# Requirements

To receive a 40% pass mark, you must develop an application that

can carry out at least the following functions:

•Addition

•Subtraction

•Multiplication

•Division

To receive a 60%+ marks, you must develop an application that

can:

•Satisfy the 40% pass mark requirements

•Provide a way to clear the screen of numbers.

•Provide memory functions to remember numbers.

To receive a 70%+ mark, you must develop an Application that

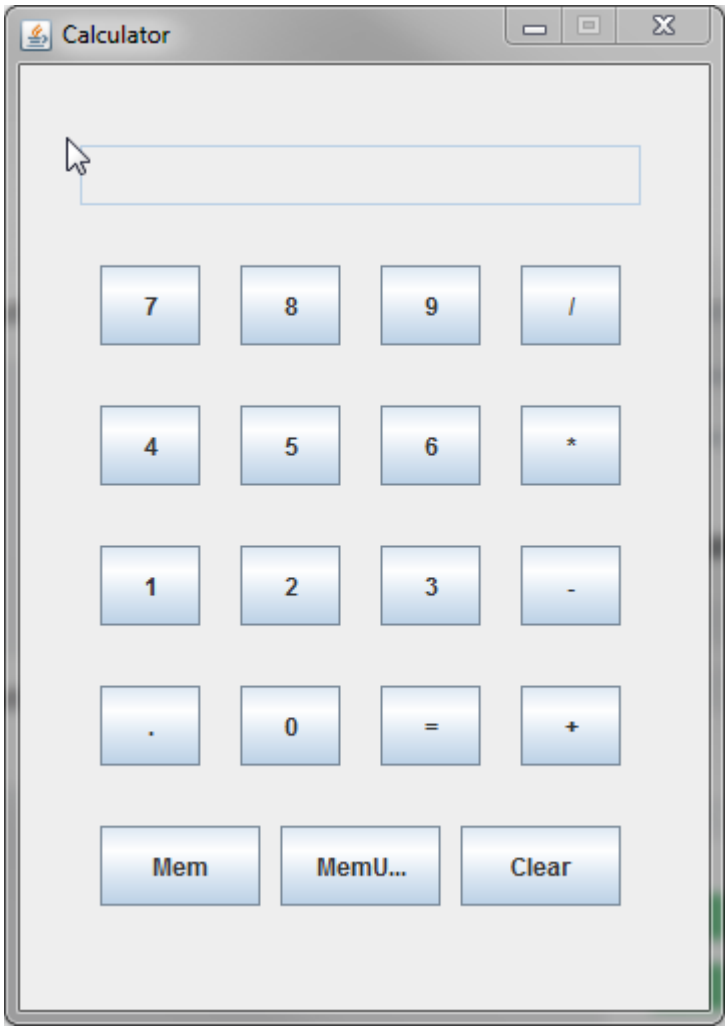
can:

•Satisfy the 60% pass mark requirements

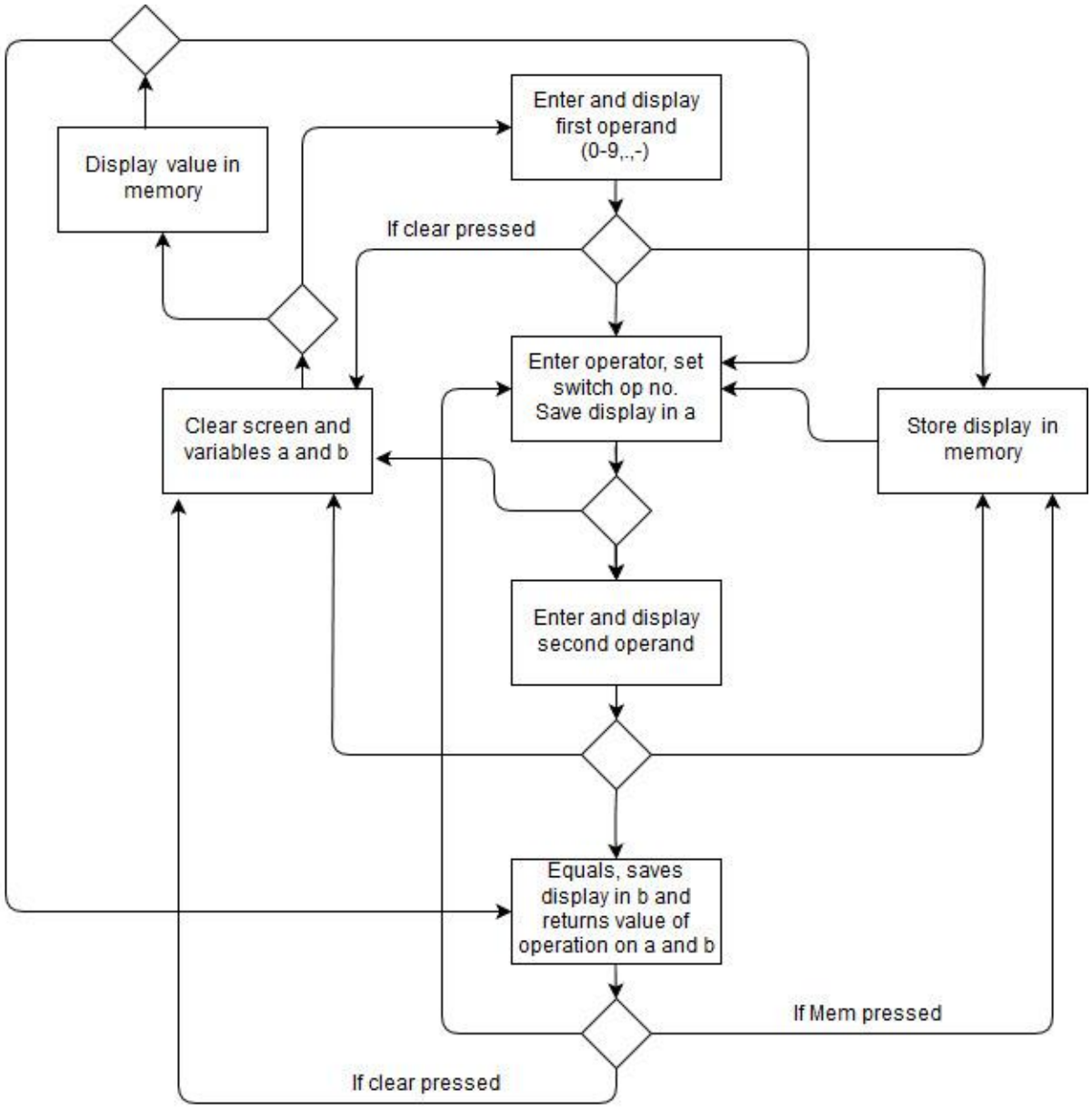
•Use programmer defined Exception classes for Error Handling (i.e. use

my own Exception Handling Classes)

## Graphical User Interface



## Calculation Logic



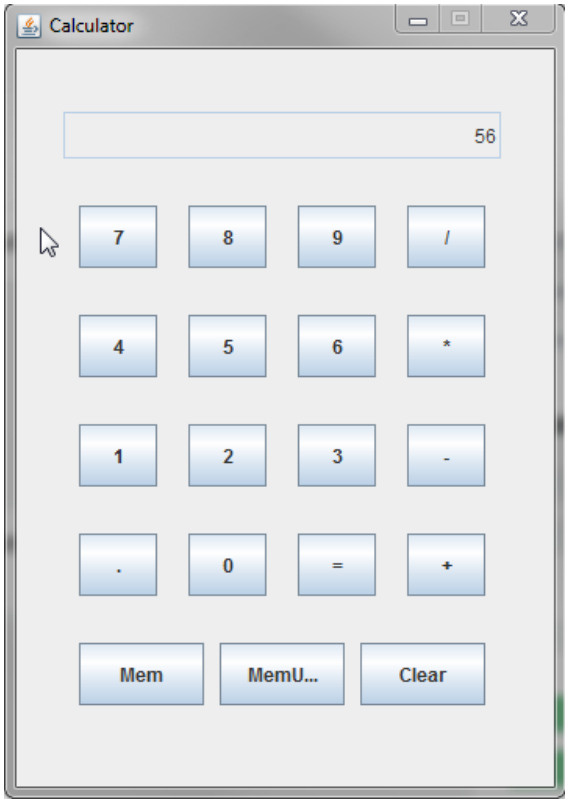
# Testing

Testing was carried out on a Windows 7 system.

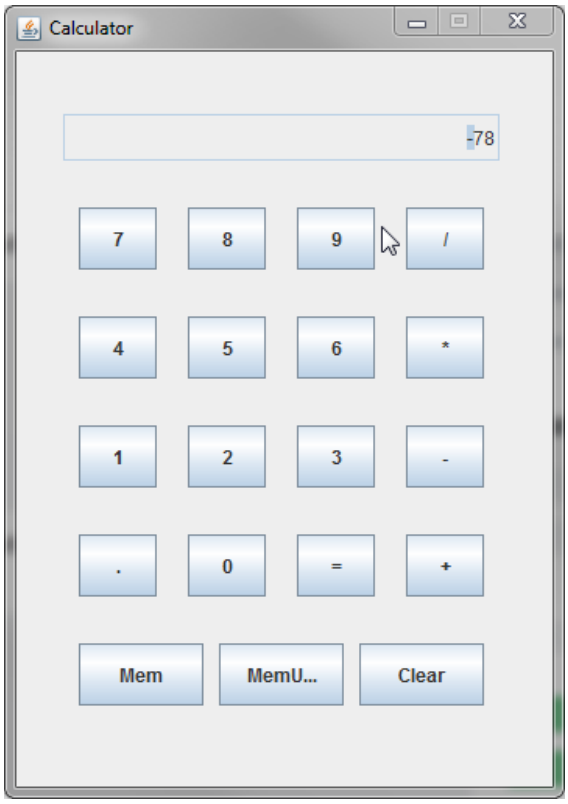
| Test No. | Test objective/scenario | Input | Expected Result | Actual result | Issue |
|---|---|---|---|---|---|
| **T1** | User enters a positive integer | Number(5), Number(6) | Number displayed | Number displayed | None |
| **T2** | User enters a negative integer | '-' operator, Number(7), Number(8) | Number displayed | Number displayed | None |
| **T3** | User enters a negative float | '-' operator, '.', Number(8) | Number displayed | Number displayed | None |
| **T4** | User presses Red X button | None | Program exits | Program exits | None |
| **T5** | User clicks on Display | None | None | None | None |
| **T6** | User enters an operator first. | Operator(+) | Enter Number message displayed | Enter Number message displayed | None |
| **T7** | User tries to divide by zero | '/', Number(0) | Error message to console | Error message to console | None |
| **T8.1*** | User tries to add two numbers. Presses first button. | Number(6) | Number displayed | Number displayed | None |
| **T8.2*** | User presses Relevant operator | '/,*,+,-' operators | Displayed Number stored in 'a' Screen is cleared | Screen is cleared | None |
| **T8.3*** | User presses Number | Number(23) | Number displayed | Number displayed | None |
| **T8.4[a]** | User presses Equals | '=' | Answer displayed | Answer displayed | None |
| **T9** | User enters Equals before number or second number | '=' | Nothing happens | Nothing happens | None |

*Steps same bar operator [a]Screenshots of four results provided below

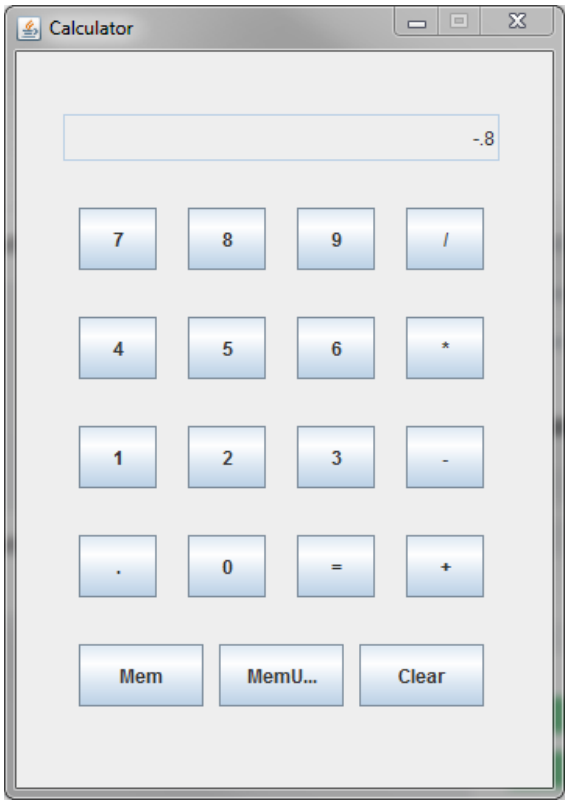
## T1



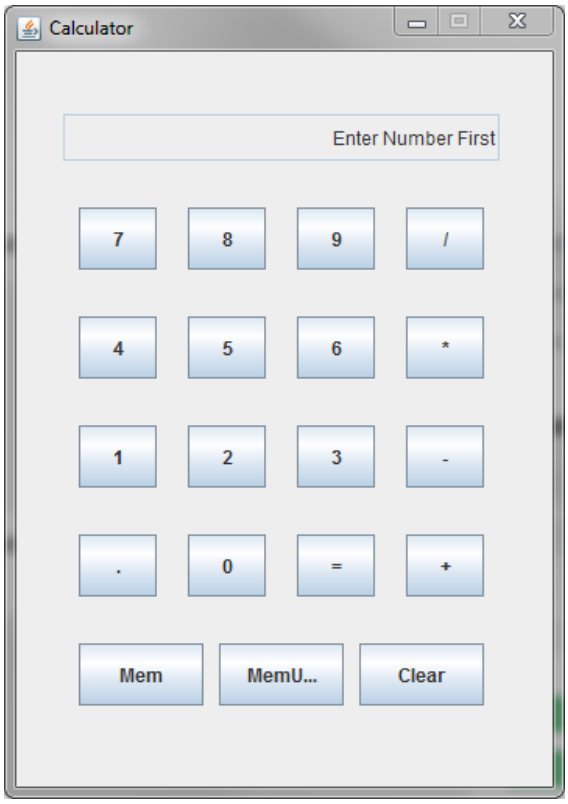
T2


Calculator
-78
7
8
9
/
4
5
6
*
1
2
3
-
.
0
=
+
Mem
MemU...
Clear

T3


Calculator
-.8
7
8
9
/
4
5
6
*
1
2
3
-
.
0
=
+
Mem
MemU...
Clear

T6

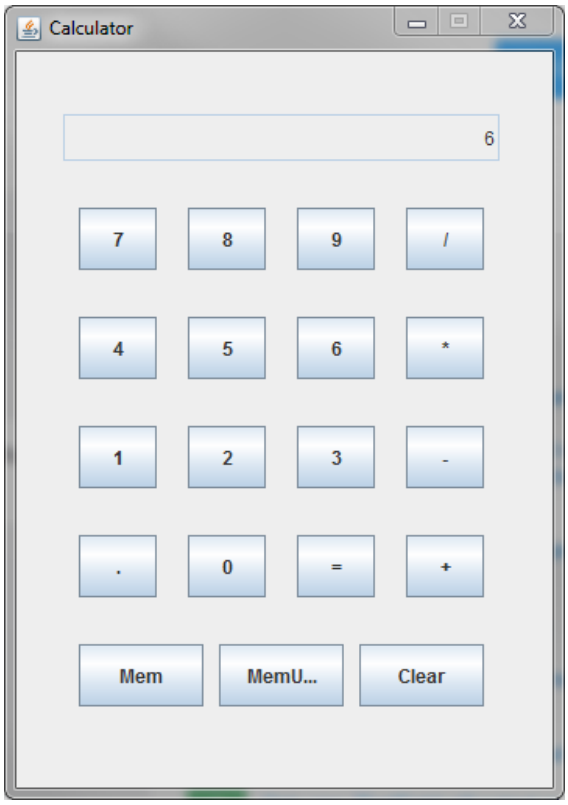



T7

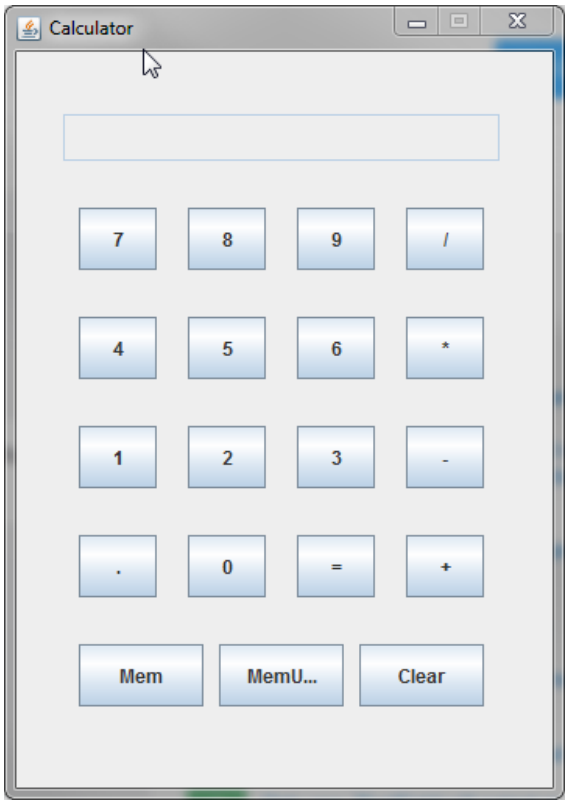
```
For input string: "-"
Exception in thread "AWT-EventQueue-0" myCalculator.MyDivideByZeroException: Tried to divide by Zero
        at myCalculator.Calculator.actionPerformed(Calculator.java:252)
        at javax.swing.AbstractButton.fireActionPerformed(Unknown Source)
        at javax.swing.AbstractButton$Handler.actionPerformed(Unknown Source)
        at javax.swing.DefaultButtonModel.fireActionPerformed(Unknown Source)
        at javax.swing.DefaultButtonModel.setPressed(Unknown Source)
        at javax.swing.plaf.basic.BasicButtonListener.mouseReleased(Unknown Source)
        at java.awt.Component.processMouseEvent(Unknown Source)
        at javax.swing.JComponent.processMouseEvent(Unknown Source)
        at java.awt.Component.processEvent(Unknown Source)
```

## T8

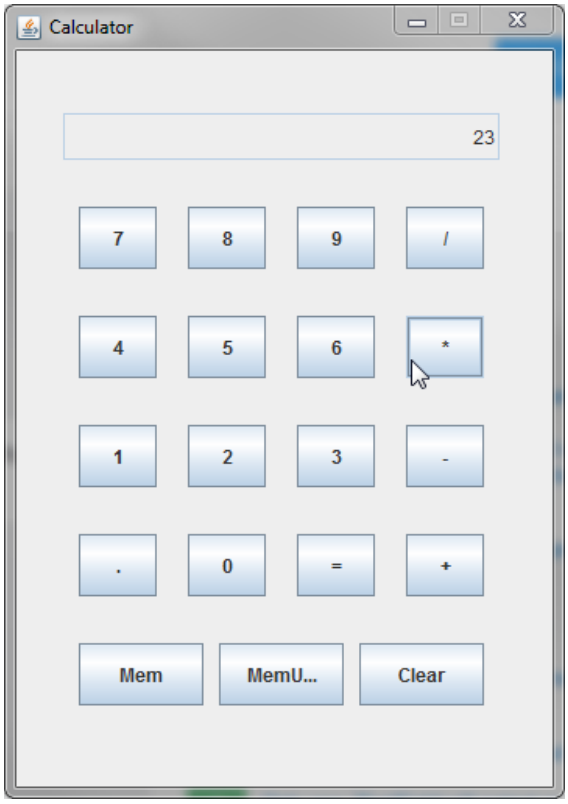
T8.1


Calculator
6
7
8
9
/
4
5
6
*
1
2
3
-
.
0
=
+
Mem
MemU...
Clear

T8.2

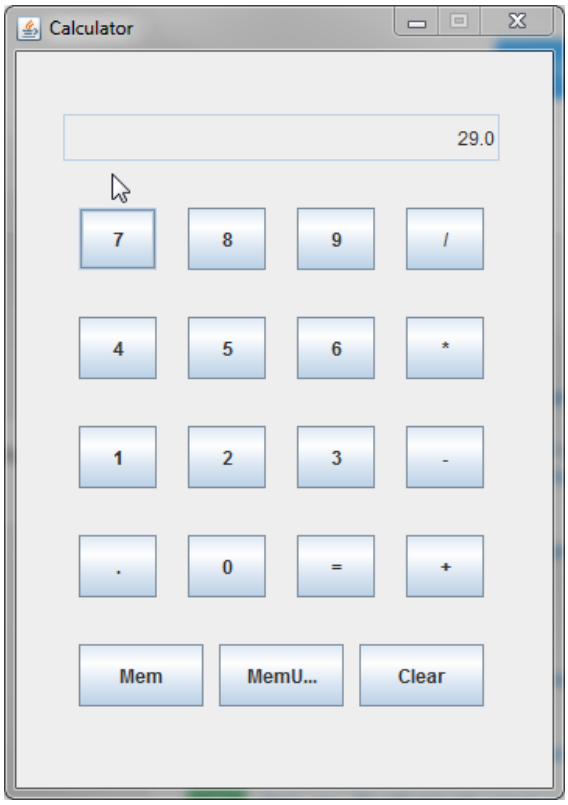

Calculator
7
8
9
/
4
5
6
*
1
2
3
-
.
0
=
+
Mem
MemU...
Clear

T8.3

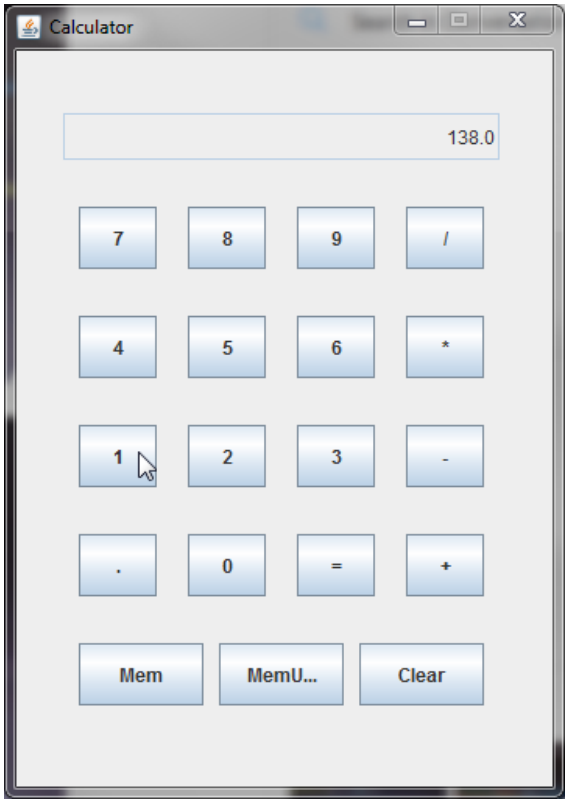



T8.4

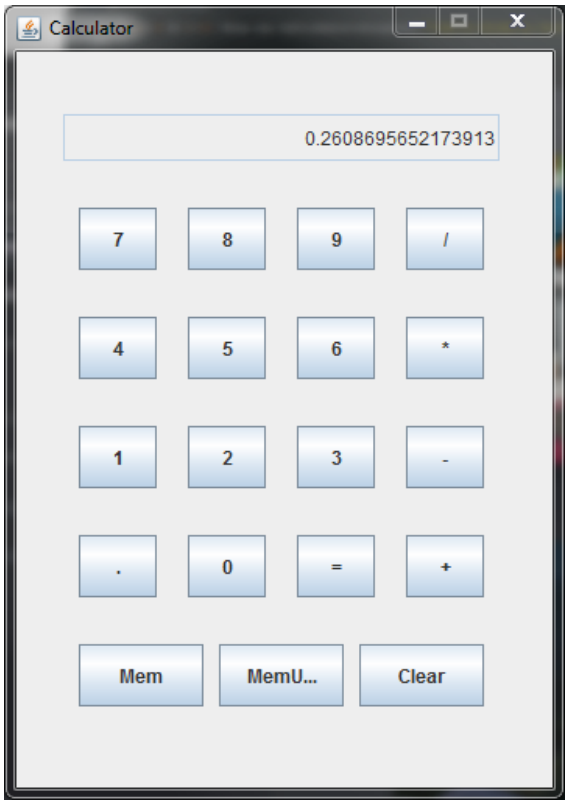
Addition



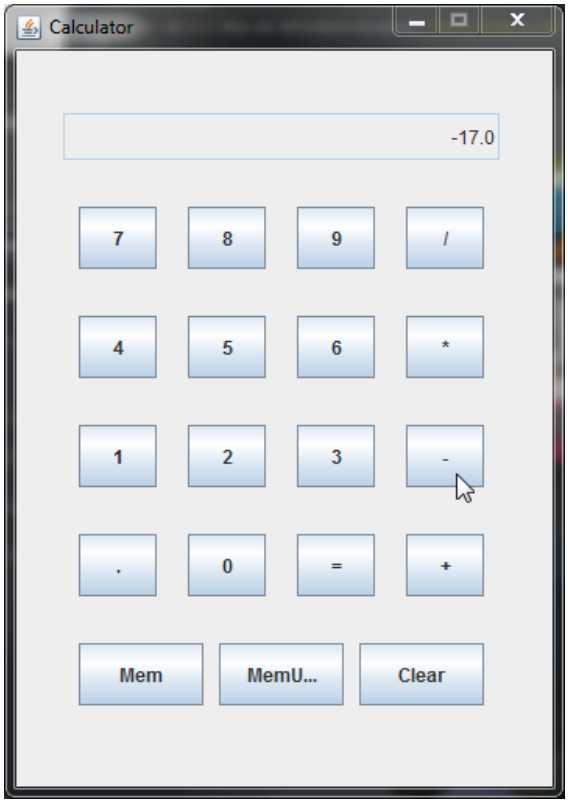
Multiplication




Division



Subtraction




## Error Log and Available Improvements

The Calculator was returning an Exception when '-' was pressed twice in a row. This has been caught in a try catch block, as have NumberFormatException's caused by pressing the '=' button.

I would have liked to include more mathematical functions in the program and if I was to improve the Program I would add the ability to accept and correctly compute problems containing brackets using PEMDAS order of operations.

## Code

### CheckButton Class

```
package myCalculator;

//Checks if Number or Decimal can be used (Prevents concatenation to "Enter Number first" message)

public class CheckButton{
```

```
	public static boolean checkNum(String s){

		return(!s.equals("") && (!Character.isDigit(s.charAt(0)) &&
		s.charAt(0)!='.' && s.charAt(0)!='-'));

	}

	//Checks if Operator can be used

	public static boolean checkOper(String s){

		return(!s.equals("") && (Character.isDigit(s.charAt(0)) ||
		(((s.charAt(0)=='.'||s.charAt(0)=='-') && s.length()>=2) ||
		(s.length()>2&&s.charAt(1)=='.'||s.charAt(0)=='-'))));

	}
}
```

## MyDivideByZeroException

```
	package myCalculator;

	public class MyDivideByZeroException extends IllegalArgumentException {

			public MyDivideByZeroException(String msg)

		{

		super(msg);

	}
}
```

## Calculator Class

This if statement checks if there is already a decimal point in the String.

```
		if(evt.getSource()==bDec){

			if(!display.getText().contains(".")){

			//code

			}

		}
```

The Subtract/minus button contains checks to see if it used for a negative number or an operator. A try catch block is used for attempting a minus

```
		if(evt.getSource()==bSub){

		//Error check for attempting sub operation on display="-"
```

```
        try{
              //Check if used for negative operand
              if(display.getText().equals("")||Character.isLette
              r(display.getText().charAt(0))) {
                    clearDis();
                    display.setText(display.getText().concat("-
                    "));
              }
              else
              if(CheckButton.checkOper(display.getText()))//Chec
              k if used for subtract operation{
                    a=Double.parseDouble(display.getText());
                    opType=2;
                    clearDis();
              }
              else display.setText(ENTERNUM);
        }
        catch(NumberFormatException e){
              display.setText(ENTERNUM);
              System.out.println(e.getMessage());
        }
  }
```

The logical operations are carried out upon pressing the Equals button. The variable opType which is set when pressing an operator determines the operation and result

```
  if(evt.getSource()==bEq){
        if(opType==4 && b==0) throw new
        MyDivideByZeroException("Tried to divide by Zero");
        try{
              if(!display.getText().equals("")){
                    b=Double.parseDouble(display.getText());
                    switch(opType){
                          case 1: result=a+b;
                                break;
                          case 2: result=a-b;
                                break;
```

```
                        case 3: result=a*b;
                              break;
                        case 4: result=a/b;
                              break;
                        default: result=0;
                  }
                  display.setText(""+result);
            }
      }
      catch(NumberFormatException e){
            display.setText(ENTERNUM);
            System.out.println(e.getMessage());
      }
}
```

The clear button sets all values to 0.

```
if(evt.getSource()==bClr)
{
      clearDis();
      a=0;
      b=0;
      result=0;
}
```