目录

[Executive Summary 1](#_Toc117257803)

[Requirements 2](#_Toc117257804)

[Design 2](#_Toc117257805)

[View 2](#_Toc117257806)

[Model 2](#_Toc117257807)

[View Model 2](#_Toc117257808)

[Meeting Notes 3](#_Toc117257809)

[Sync-up meeting 10.19 3](#_Toc117257810)

[Takeaways: 3](#_Toc117257811)

[To-Learn lists: 3](#_Toc117257812)

[To-Do lists: 3](#_Toc117257813)

[Sync-up meeting 10.26 3](#_Toc117257814)

[Questions: 3](#_Toc117257815)

# Executive Summary

Goal: create a WPF desktop application that mimics a simple calculator with the following operations, +, -, \*, / and clear, and a standard keypad layout

Deliverables:

1. software requirements, break tasks into smaller tasks underneath each requirement if necessary
2. a git repository on GitHub
3. written in C# using Window Presentation Foundation (WPF) and Model-View-View Model (MVVM) design pattern
4. unit tests
5. a 25-min presentation on MVVM, using the example(s) from the code
6. a working application

Target demo date: 10/27

GitHub repository: [screw-44/CalculatorV3 (github.com)](https://github.com/screw-44/CalculatorV3)

# Requirements

The software application shall allow the user to input integer and decimal numbers.

The software application shall allow the user to perform basic calculations including addition, subtraction, multiplication, and division.

The software application shall have a layout like Microsoft Calculator. All the digit button cluster in the left bottom. Calculator sign buttons on the right in cyan. Clear, Delete, Equal Button in the top in gold, just below the Text Box.

The software application shall

* display the input digits when the user pushes the button from 0~9 and dot.
* Display calculator signs too, but just like the Microsoft calculator, when you type a calculator sign in the same place multiple times, it only shows the last typed calculator sign.
* Display the calculated result when pressing the = button, in the form “A+B=C”。
* After showing the calculated result, the next input should clean the screen.

The software application shall be able to detect valid input and inform the user.

Expand this.

Mark: The scope is the same as the requirements.

# Design

## View

The layout mimics the Microsoft calculator, With buttons: 0123456789, +,-,×, ÷, =, delete, and clear.

Display the num buttons in the left bottom and in order as the Microsoft calculator. Text Box in the very top, and the clear, delete, = button below it in gold. The calculator sign in the right and in cyan.

The Text in the text box should starts at right.

## Model

Since it only is a simple calculator, it should be sufficient to only have three objects.

{

Double num1;

Double num2;

Calculate function;

}

## View Model

Function1: Parse the object1 from string to double.

Function2: Change the calculate function based on the pressed calculator sign button.

Function3: Fire the calculate function and show the result in the text box.

Object 1: A string representing the num input from the textbox.

Object 2: A string binding to the textbox as input.

Binding Command:

1. Should have command parameters to tell which button is pressed.

Add object 1 and object 2 different characters based on which button is pressed.

1. Fire function1 when the ‘=’ button or calculator sign button is pressed.

If num1 is null, cast the input to num1, else cast it to num2.

1. Fire function2 when the calculator sign button is pressed.
2. Fire function3 when the ‘=’ button is pressed, or the calculator sign button has been pressed twice.

Logics stored in another calculator library.

# Meeting Notes

## Sync-up meeting 10.19

### Takeaways:

* Development loophole: Requirements => Scope => Design Implement Test => Deploy => Gather feedback => Requirements.
* Make more documentation and notes inside of the code.
* Word documentation for planning, Readme for code deployment; Notes inside of the code for maintenance.

### To-Learn lists:

* The package: CommunityToolkit.Mvvm
* Using Command Parameters to merge all the functions into one big function.

### To-Do lists:

* Omit the name and click mode in the button, the press and release modes both are ideal for the job.
* Using proper signs for multiply and divide.
* Characters inside the text box should start from the right.

## Sync-up meeting 10.26

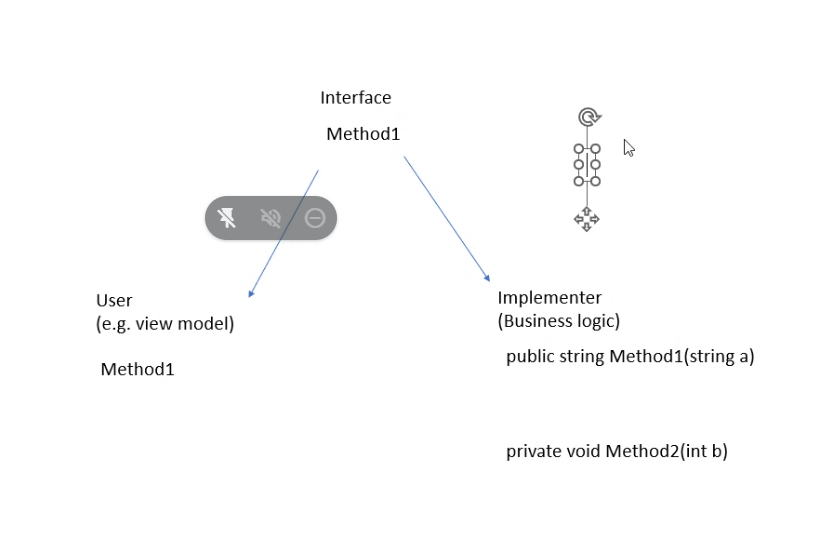
### Questions:

* The command parameter automatically cast char into a string(seems to), can we work around this problem?
* I try to unit test my ViewModel, but many of its logics are coupled with the View and involve some private methods and objects.
* If I put business logic in a separate module, how do I interact with Model? Is it more suitable to push the business logic in a function of the ViewModel instead of a class?

Expand input validation part.

Using diagrams more, like UML, sequence diagrams.

ITERFACE!



How does interface plays in the Dependency injection and unit testing.