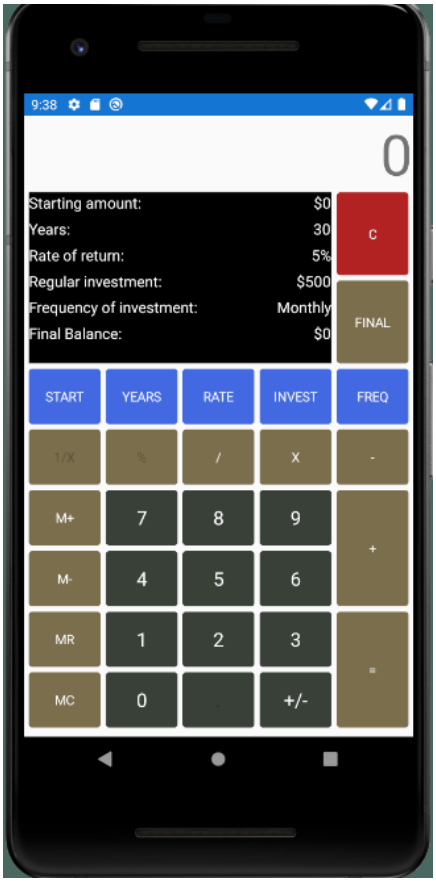
CSE 382
Project #3 - Investment Calculator
Fall 2022



You will create a calculator app, named `Project3Calc` (use this name for your Git repo), that is loosely based on a calculator that I have used. The figure shows this investment calculator in portrait mode on an Android Pixel 2 emulator. Aside from performing basic integer arithmetic, the calculator will comput how much money an investment account will have after a number of years, given information about starting balance, rate of return, years maintained, etc.

You are to implement a calculator that looks like the one shown but provides only the functionality described below. The following features that are to be provided by your implementation:

- The GUI layout and styles must be implemented using exclusively XAML.
    - You must set up your XAML so that it is easy to perform stylistic changes.
    - The layout, colors, and buttons must appear very close to the appearance shown above: sizing, alignment, layout, spacing, etc should be nearly identical to that shown in the figure. Aspects that need to be accounted for:
        - Text: color, font size
        - Buttons: size, corner radius, enabling/disabling, separation between buttons.
    - Notice, the button-sets have different colors: functions, digits, operations, and investment options. You are free to alter the colors but there must be groups of buttons that share a color.
        - There are 4 different groups of buttons - Investment (blue), numbers (brownish), etc.
- Your calculator will be functionally less extensive than a commercially available calculator.
    - Only integer operations will be supported. Division will be integer division.
    - The following buttons should have their usual meaning:
        - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
        - +, -, *, /, = (operations are integer arithmetic)
        - C (clear)
    - The buttons 1/x, %, and . (i.e., dot) should be disabled.
    - The blue buttons control the investment parameters.
- Your calculator should be run in portrait mode using two similar, but different sized devices. Your layout should look reasonable for both devices. To do this, do use HeightRequest, StartAndFill, etc. Create your submission video using one of the following:
    - iOS iPhone 13 and iPhone 13 max (for example)

    - Android Pixel 5 and Pixel 4XL (for example)
    - UWP with the window sized to be approximately that of a phone held in portrait mode. The second one should be about 10% larger than the first.
- You are not required to handle the situation where the phone is held in landscape mode.
- As a simplification, you do not have to support multiple operators strung together (e.g., 1+ 2 * 3"). Instead, you can assume the operations are a single operation such as (12 * 9 =).

Notes.

- A video has been provided to illustrate the operation of your calculator.
- You are to complete this assignment without using MVVM (which we have not yet discussed in this course) architecture (e.g., using Command). Instead, you are to implement this using only XAML and basic event handlers that you define.
- Your program should not crash under any scenario.
- Code for computing the final balance of an investment is given at the end of this document.
- Submit via Git using the project name `Project3Calc`.
- Like the previous project, you must submit a video demonstrating your project. In addition to announcing yourself and displaying your code, you must perform the following at a rate of about 2 button clicks per second. Here is the script you should follow:
    - Intro (maximum of 60 seconds)
        - Announce yourself
        - State what works and what does not
        - Show your XAML. Highlight the work that you did to convince the knowledgeable viewer that it is easy to change colors, sizes, etc.
    - Basic operations (maximum of 60 seconds)
        - C 1 + 2 =
        - C 123 + 987 =
        - C 8 * 3 =
        - C 8 / 3 =
        - C 8 - 3 =
        - C 3 - 8 =
        - C 35 +/- + 9 = // Show that 1/x, %, and . are disabled.
    - Memory (maximum of 30 seconds)
        - C MR
        - 123 M+ C MR
        - C 100 M+ MR
        - C MC 99 M- C MR
    - Investment (less than 30 seconds)
        - C 100 START 1 YEARS 5 RATE 0 INVEST FREQ ANNUAL FINAL
        - 1 INVEST FINAL
        - FREQ QUARTERLY FINAL
        - C 0 START 35 YEARS 6 RATE 200 INVEST FREQ MONTHLY FINAL
        - C 34 +/- // show that the investment buttons become disabled.
        - +/- // show that the investment buttons become enabled again.
        - C 5 - 10 = // show that the investment buttons become disabled.

        - C // show that the investment buttons become enabled again.
    - Second device (less than 30 seconds)
        - Run the code with a 2nd device that has a slightly different screen size than the first. Comment on how your XAML adapts to the new size appropriately. You do not have to redo the script; simply show the controls' appearance.

Scoring.

- **(70) Operation**. The application must operate correctly.
    - Digits and basic arithmetic operations
    - Memory
    - Investment
- **(30) XAML description of GUI**.
    - The GUI's layout and style must effectively, and exclusively, be done using XAML.
    - Event handlers will reside in the C# code behind.
    - Layout reasonably adapts to two similar, but different, sized devices.

Computing Final Balance. You can use the following code to compute the final balance.

```
/*
* start is the starting balance
 * years is the number of years that the investment will encompass
 * perc is the annual rate of return (e.g., 6% should sent in as 0.06)
 * investment is the amount of money added to the account on a regular basis
 * depositsPerYear is the number times per year (12, 4, or 1) that a deposit is made.
 */
private int Compute(int start, int years, double perc, int investment, int depositsPerYear) {
        double bal = start;
        double monthlyRate = perc / 12.0;
        int monthsToDeposit = 12 / depositsPerYear;
        for (int y = 0; y < years; y++) {
                for (int m = 1; m <= 12; m++) {
                        bal += bal * monthlyRate;
                        if (m % monthsToDeposit == 0) {
                                bal += investment;                  // make deposits at the end of the month
                        }
                }
        }
        return (int)Math.Round(bal);
}
```