ASU W. P. CAREY SCHOOL of BUSINESS
ARIZONA STATE UNIVERSITY

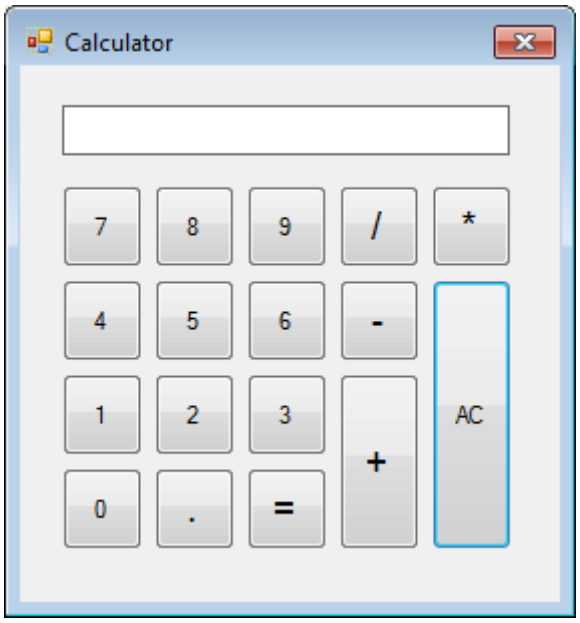
CIS 345 – Business Information Systems Development II – Fall 2014

Assignment 8

Due on Blackboard: November 14 at 10 PM

Create a basic four-function Calculator using Windows Form Applications within Visual Studio. The application should function like common calculators, which allow addition, subtraction, multiplication, and division of two numbers. It should have AC button, which clears all the elements of the calculator as well as an "equals" and a decimal button.

Develop the calculator using either a basic approach creates the controls at design-time using the *Designer* OR for an advanced challenge, create all controls at runtime using code. (See *Advanced Challenge*)



Skills developed: Utilizing Visual Studio Form applications, creating and using event handlers, implementing logic.

Project Details and Requirements

A sample executable file is uploaded and available on Blackboard for you to test program behavior.

Project Name: Calculator
Form Name: Calculator

Your Calculator should:

1) Add, Subtract, Multiply, and Divide two numbers
    - Be able to use the text being displayed as number 1. The text entered by the user after the math operator should be used as number 2.
2) Have a Clear button
3) Offer the user the ability to use decimals
4) Be able to show decimals in results
5) Show accurate results for all four math functions.

*You do not *need* to handle exceptions, but this would be a good opportunity to handle the DivideByZeroException and display "Error" when someone does attempt to divide zero by zero.*

Test your program and match its behavior with these test cases:

User input	Result
2 5 + 4.78 =	29.78
1 + 2 = * 2 =	6
6 / 5 = + 1.75 =	2.95
2 + AC 3 + 2.5 =	5.5

Use the following as hints in implementing your own logic for the Calculator:

1) Use a Label rather than a TextBox to show your Result. Use a white background and set the Height and Width within the Minimum Size property using the Designer.
    a. Change Font and Font Size as needed using the Designer.
2) Store the two numbers as strings. Convert the numbers to doubles only when the user wants a calculation performed (presses 'equals').
3) Use string concatenation (joining) for constructing the string to be displayed in the Result box.
4) Store what Math operation the user wants to conduct as a string.
5) Use a switch to perform your calculation according to value of the Math operation string.

Optional Enhancement

Add logic so that the user does not need to press the "*equals"* button but rather could continue the calculation by pressing any operator button. With most calculators, pressing an operator button such as "+" or "-" results in a calculation, display of the result, and also prepares the calculator to do another calculation using the selected operator, e.g. "2 + 5 * 3 =" results in 21. In this case, the plus and the multiplication operators are reacting differently – the plus operator button does not perform a calculation but the multiplication operator button does perform a calculation (2 + 5).

You will need to work out logic for all possible combinations of numbers and operators being entered such that the same event handler reacts differently based on whether one or both of the *operands* i.e. numbers have been entered already or not.

Advanced Challenge

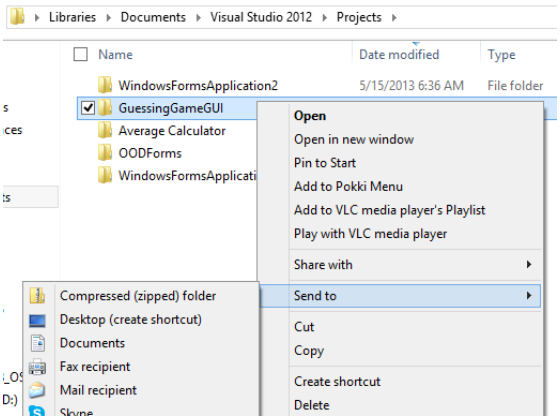
In order to get practice in creating controls dynamically at runtime, you may implement the Calculator using controls that are created at runtime. For this, you will probably want at least one array of Buttons for the numbers 0-9. You may optionally have one more array of Buttons for the mathematical operators. You may want to utilize the "Tag" property of controls to store data. The Tag property can be used as a variable to store some data about the control. For example, numberButtons[0] might have a Tag property of 0 to indicate that this button represents the number zero. Doing so will ensure that you are able to find out information about a button without relying on the name of the event handler.

Each of the buttons will need event handlers. Since you will not have buttons at design-time, you will need to ensure two things: 1) that you create a generic event handler in code, and 2) that you register that event handler at runtime after you instantiate your buttons. If you end up using the Tag property, you might be able to manage all the number buttons with just one event handler since you will be able to know which button the user clicked on by accessing the Tag property. This would be a far more efficient and sophisticated way of managing event handlers vs. creating 10 different event handlers. Each event handler does the same thing, just for a different value – so it is preferable to create one method that uses a different value to account for each button.

Submission Instructions

Submission should be made using a zip file that contains all of the Visual Studio C# project files. You will need to ***zip the entire project folder*** along with the .sln and .suo files. The folder will automatically contain the class source files as well as the executable file that is generated in \ProjectName\bin\Debug folder. Upload file to the Blackboard assignment drop box.

Zip the entire top-level folder by right-clicking the folder and selecting Send to | Compressed (zipped) folder.



Using built-in windows zip tools: http://windows.microsoft.com/en-us/windows/compress-uncompress-files-zip-files

Make sure you check the following. Your grade is dependent on all these criteria being met.

- You have included your name as a comment within your class.
  - e.g. " // Assignment 8, Jane C. Smith, CIS 345, Tuesday 9:00 AM"
- Zip filename is: FirstNameLastName_Assignment8.zip
- Your code is commented and you are using all prescribed programming conventions.
- Your code utilizes PascalCase and camelCase as appropriate.

Verify your zip file before you submit

- Check for actual class files being present in the folder before you zip it.
- Check your zip file size after zipping – if it is 1K, it likely contains only a shortcut.
- Uncompress your zip file before submitting and verify that files are present.
- Download your zip file after submitting, uncompress, and again verify that your files are present. Test your files in Visual Studio after uncompressing.
- Make sure you have submitted your file and not just saved a draft on Blackboard. A blue clock indicates a submission in progress, i.e. a draft, not a submission.

This takes an extra couple of minutes. Please do it if your grade is important to you. If you do this, you will not end up submitting a bad file. If you submit an empty file, or one containing only a shortcut, or a bad zip file, or a bad project file, you will receive a score of zero and your only recourse will be to do the makeup assignment at the end of the semester.