Assignment 3

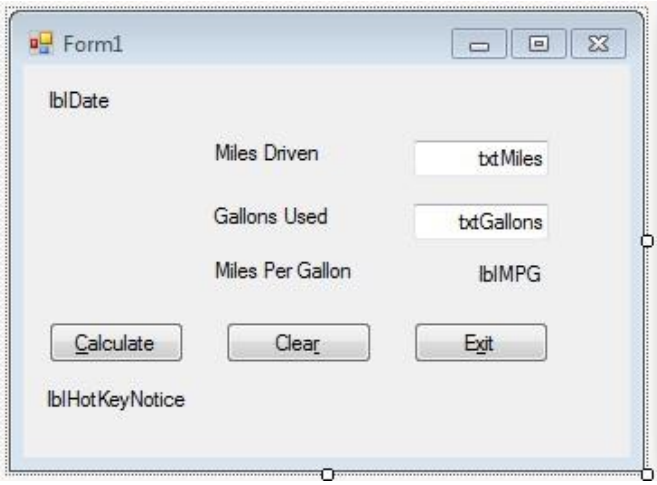
Chapter 14, Points: 100 (Independent assignment, not in the text, but Chapter topic related.)

Set Up

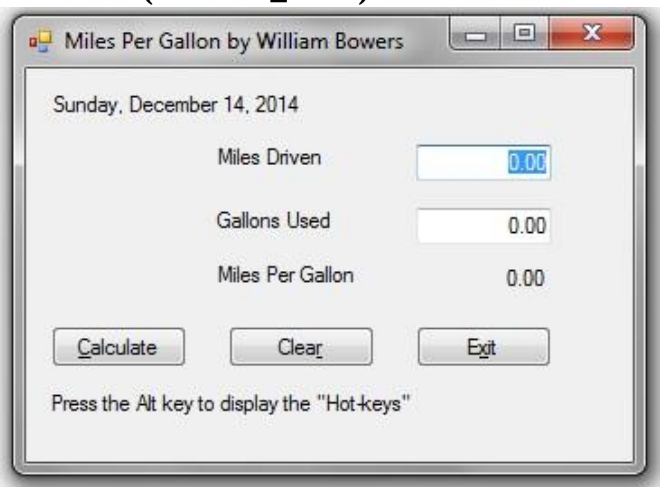
1. We will use our last Assignment 2 as a guide for this assignment, reconfigured as a Windows application.

Exhibit A. Design View and Runtime View. When in doubt as to an instruction, code to this Exhibit in these situations. *An Exhibit is the equivalent of written instruction(s) and, therefore, gradable in every aspect.*

Design



Runtime (on Form_Load)



Suggested Practices

The design view is my practice of managing a graphics program. You are free to use these techniques or remain as unorganized as you wish. Your grade will not depend on perfect order. True Rapid GUI Development depends on consistent planning and his does matter to an employer.

1. Labels
   a. Those used for just labeling, being descriptive of another object, are left as-is. For example, Label1 remains named Label1 with a Text value of "Miles Driven" because it is just text to describe the purpose of the txtMiles TextBox. Label1 is not used to calculate or display anything other than the "Miles Driven" verbiage.
   b. Those labels that perform a specific task, such as shown to display a computer-clock driven date, or display a notification are given a variable name for easy identification of the object and its revised variable name at-a-glance. Their changes typically occur in Form_Load. Labels, such as lblMPG, display calculations, and their source is typically from a method, such as Main(…) or another appropriate source.
2. TextBoxes

These accept input, as you know, but to avoid coding confusion, they are provided descriptive variable names for easier viewing during coding and runtime testing. Changes to these, whether to provide a default value or to display as empty and ready for the user typically occur on Form_Load, as well.

3. Command Buttons
    a. Command buttons are provided with unique, description variable names, such as btnCalculate, btnClear and btnExit. When many buttons are used, just leaving them as button1, Button2, etc., can be very confusing when coding, rearranging, displaying their Text property.
    b. Remain, whenever possible, within the Windows programming tradition of using Hot Keys; that is, underlining the keystroke when used in combination with the Alt-key to generate the code associated with the Button. The purpose of the lblHotKeyNotice is to let the user know they are available.

    Back about version Visual Studio 2005, Microsoft's Windows Desktop screen stopped displaying the hot key underlining as a default. During the remaining version releases, 2008, 2010 and 2012, the display had to be manually configured by the user, and that took some explanation an average user need not have been left with. Poor grammar, but the point is, never leave programming details and features to the end user. With Visual Studio 2012, the procedure to activate hot keys that do not show by default anymore is to press the Alt-key while in runtime. Once activated, the Windows Desktop is also activated, and the notification is redundant. But without it, experienced end users will complain as to the initially-apparent lack of shortcut keys.

Required Modifications

4. Form_Load Requirements
    a. The runtime form must be centered on the screen at runtime. While our text is not rich in information on how to do this, the internet is; enjoy the research. We will expect this with probably all future runtime screen placement.
    b. Code the Form's Titlebar to display the string "Miles Per Gallon by ", concatenated with your full name.
    c. Display the lblHotKeyNotice string, "Press the Alt key to display the \"Hot-keys\"".
    d. Set the default display of txtMiles, txtGallon and lblMPG to 0.00 and set the focus on txtMiles. Within the txtMile and txtGallons runtime displays, use the SelectAll( ) methods to highlight all characters so the end user doesn't have to manually highlight the characters in order to modify them.
    e. Populate lblDate's Text with the assignment of DateTime.Now.ToLongDateString();

5. Command Button Requirements

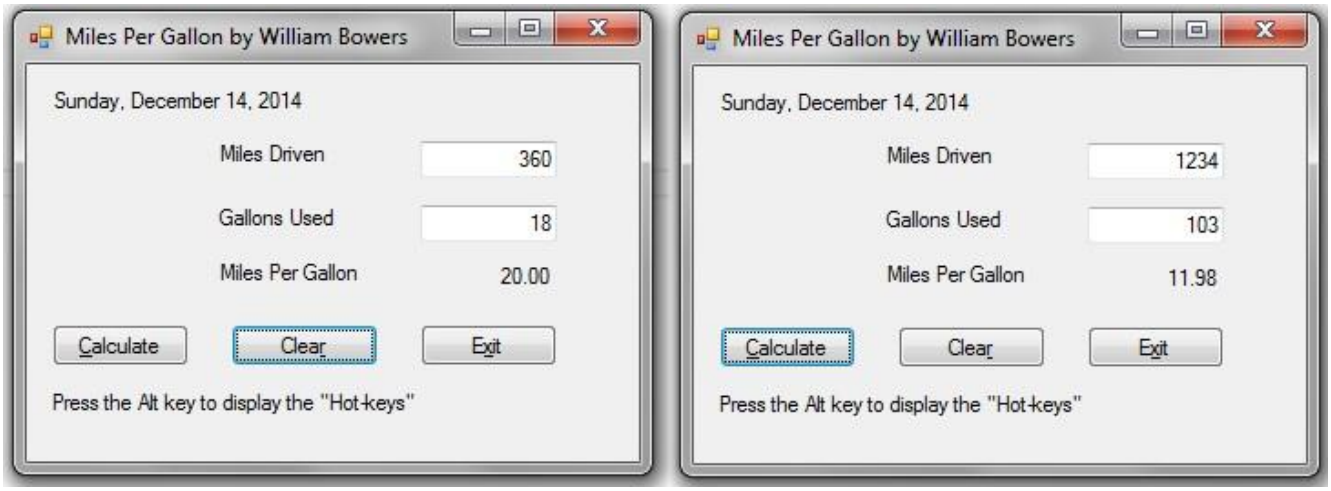
   a. Calculate Button
      1) The Text property is "Calculate, with the first C as the Hot Key.
      2) Use a double data type with a variable name of your choosing to accept the user's input from txtMiles. Remember, it's a string, so you have to convert it to a double in order to perform math with it. Also, while the text shows using decimal data types, there are two reasons to use a double: first, don't forget a double exists; second the decimal is a Microsoft C# only data type, and it won't convert to a double if you convert this program to another programming language.
      3) Do the same as the above step to get, convert and use the input from txtGallons.
      4) Perform the calculations of dividing the miles by the gallons to derive miles the per gallon solution.
      5) Format the miles per gallon solution to a string and have it display in lblMPG wih a minimum and maximum of two (2) decimal places.

   b. Clear Button
      1) The Text property is "Clear, with the last letter "r" as the Hot Key.
      2) Re-initialize both TextBoxes and the Label lblMPG to "0.0".
      3) Set the Focus back to txtMiles.
      4) SelectAll( ) all available characters.

   c. Exit Button Requirements

      The Text property is "Exit, with the letter "x" as the Hot Key. This is also a Microsoft and Windows standard.

Exhibit B. A couple of runtime check number examples. When in doubt as to an instruction, code to this Exhibit in these situations. *An Exhibit is the equivalent of written instruction(s) and, therefore, gradable in every aspect.*



Submission Requirements

Zip up the visual studio project folder containing all the files of your project and submit it through the drop box on blackboard.