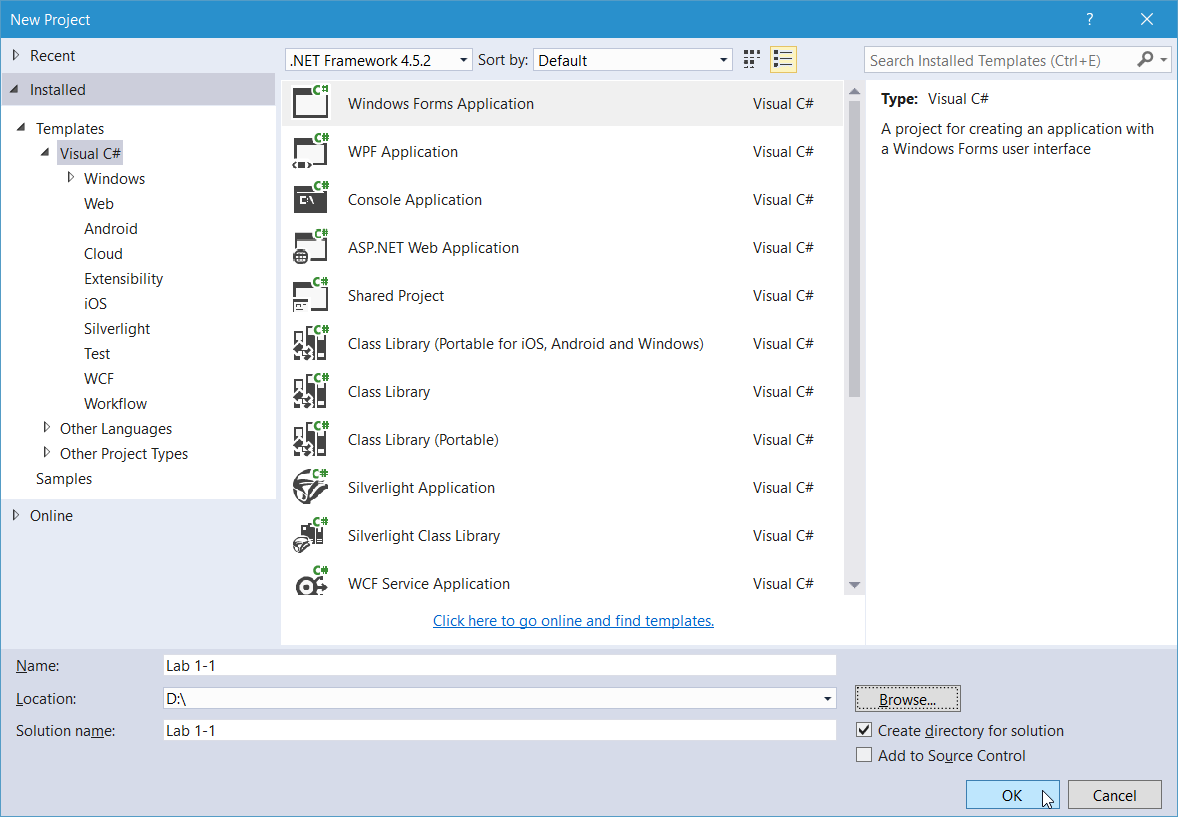
**Visual C# – Lab I**

The principal purposes of this lab exercise are to provide some introductory experience in:

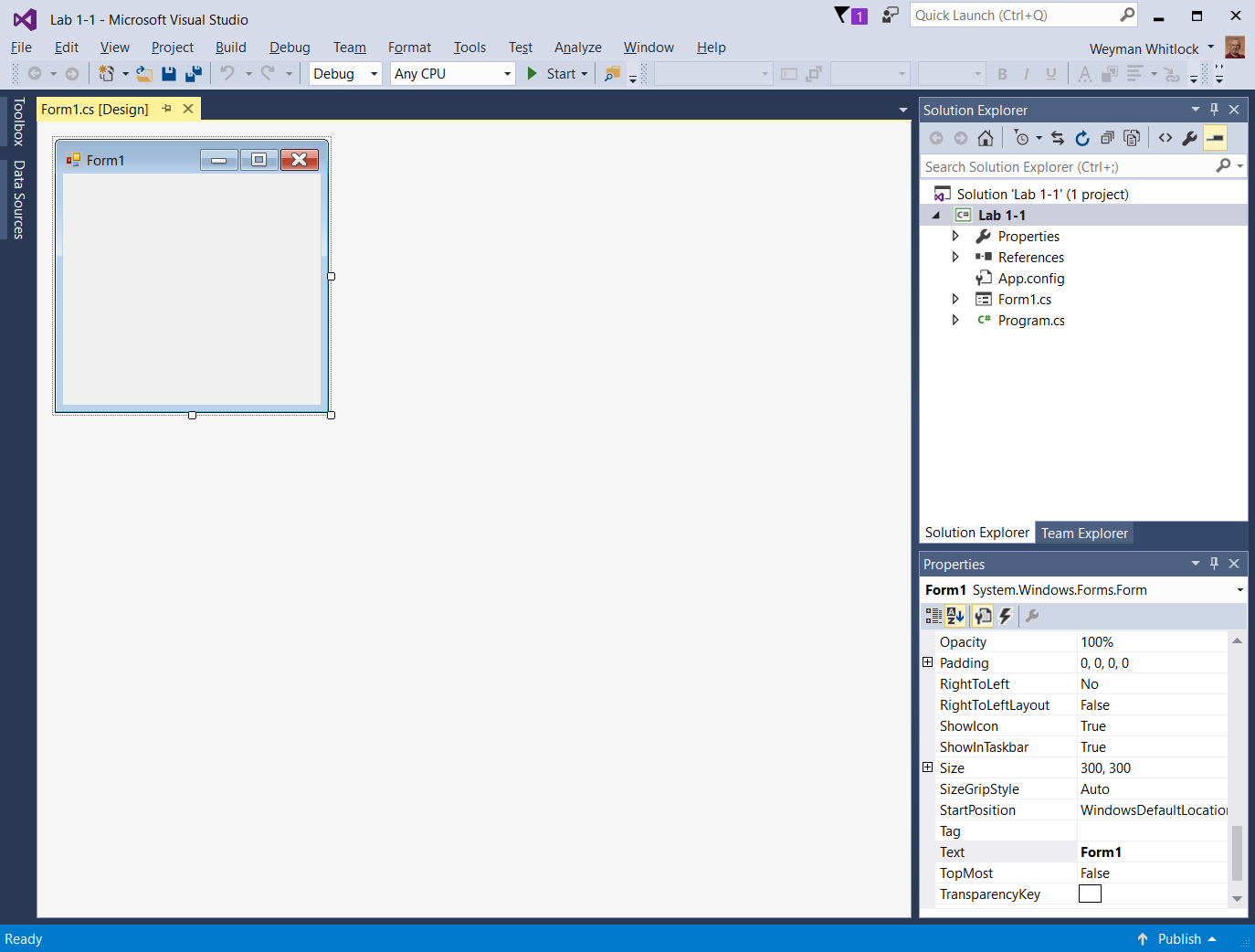
* using the Visual Studio Integrated Development Environment (IDE)
* creating new Visual C# projects
* using several Visual C# controls and their associated properties

Follow the instructions below:

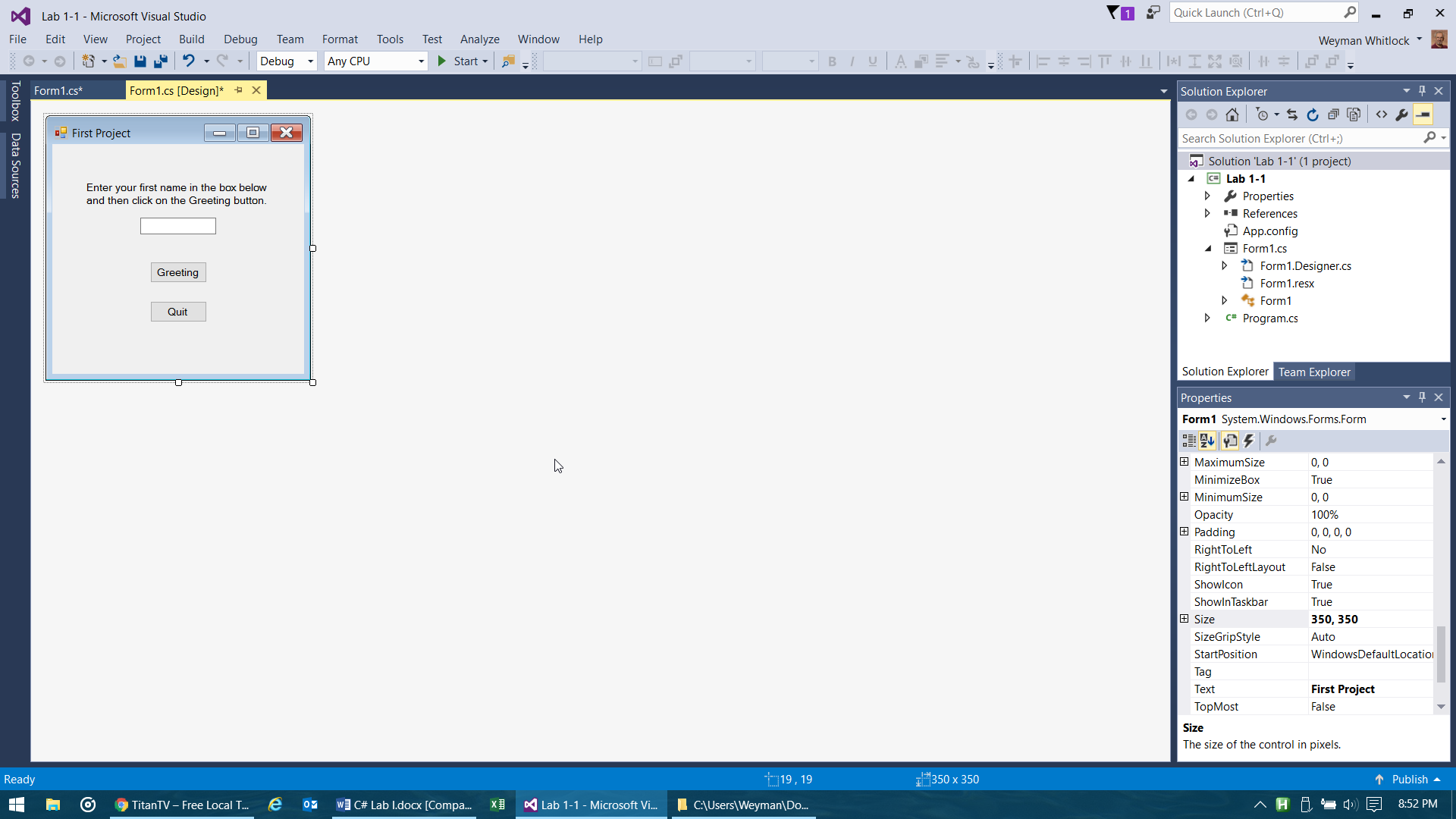
* Run Microsoft Visual Studio from the Windows Start menu.
* Depending on the configuration currently applied to the copy of Visual Studio 2015 that you are currently using, you should now be viewing a screen that looks similar to Figure 1-29 on p.28 of your textbook.
* To begin a new program, select the **File** command from the menu bar, then select the **New** command from the drop-down menu, and then the **Project** command from the submenu. (You can, instead, simply click the New Project button on the tool bar; this will be one of the first buttons on the left end of the tool bar.)
* A dialog box titled “New Project” will appear. The appearance of this dialog box should be similar to that shown in Figure 1-33 on p.31 of your textbook. Make sure that the project template type on the left side of the dialog box is set to **Visual C#** and that **Windows Forms Application** is selected in the center portion of the dialog box. Near the bottom of the dialog box, the new project has a generic name shown in the Name box (e.g., WindowsFormsApplication1). Change this Name to: **Lab 1**-**1** (which specifies C# Lab I, Project 1). The Location for the project should be set to wherever you want to save your work. (If you are working on a lab PC, save your work to a USB drive.) The screen shot below shows the settings for this dialog box. When you have your settings match­ing these (except possibly for Location, which may show a different drive letter associated with your USB drive), click the **OK** button at the bottom of the dialog box.



* You should now be looking at a screen that is *similar* to that shown at the top of the next page. Note that the particular computer you are using may not have the exact same selection of screen windows open, or they may not be in exactly the same locations as shown on the screen shot on the next page. You may wish to experiment with the Visual Studio IDE at this point to see if you can adjust your screen’s appearance to match that shown below (although that is not required to complete this lab exercise).



* The screen shot above shows the Toolbox window (minimized to a tab on the left), the Solution Explorer window on the top right, the Properties window on the bottom right, and a blank form titled Form1 in the middle. (See also Figure 1-35 on p.32 of your textbook.) If any of the windows shown above are not displaying on your screen, you can use the View menu in the menu bar to select the type of window (e.g., Properties Window) that you wish to add to the screen display. (If you have difficulty getting this screen appearance, ask your instructor for assistance.)
* The screen shot below shows the end goal of this project. The form appearance will vary with the version of Windows being used and the computer’s screen resolution. Shown is the design view of the form on a laptop computer running Windows 10. Follow the instructions that begin below to create this application.



1-1. Note that the title bar of your blank form currently reads Form1. This is the default title bar text assigned to a new form by Visual C#. Let’s change this. First, click once anywhere on the form to select it. Now, look at the Properties window (in the lower right-hand area of your screen). Approximately 50 properties are displayed that describe the appearance and behavior of the form. (If you do not see this lengthy list of properties dis­played, you have not yet clicked on the form as instructed above.) Scroll down through the list of properties until you find the **Text** property. The value for this property is currently Form1. Click in the area where this value is displayed and edit it (using Backspace and/or Delete keys as necessary) to a new value of **First Project**. Click outside of this text area (e.g., click on the form, for example) and you will see that the title bar of your form has changed to display the new text.

1-2. Now you will add some *controls* to your form. A control is the name used for the various objects that can be added to a form to provide program functionality. As shown in the screen shot on the previous page, your form needs to have four controls added to it: 1) a *label* to display the instructions, 2) a *text box* to hold the name that the user will type, 3) a *button* to allow the user to display a greeting from the program, and 4) another *button* to allow the user to end the program. Each of these control types is listed in the *Toolbox* located along the left side of the screen. (The Toolbox is currently minimized to a tab; place your mouse pointer on the tab to open the Toolbox). The steps to add controls to your form are next.

1-3. Find the Label control type in the Toolbox. Place your mouse pointer over this control type and press, and keep held down, your left mouse button. Drag your mouse pointer somewhere over your program form and then release the mouse button. An instance of a label control should be created on your form displaying the text **label1**. (Note: An alternative way to create a control on the form is to double-click the control in the Toolbox.) The Properties window on your screen should now be displaying the proper­ties for this Label control. (If it is not, click once on label1 to select it.) Now, change the **(Name)** property (near the top of the properties list) of this control to be **instructionsLabel.** (Note the specific case of the letters used in this name.) Next, change the value of the **Text** property of the label. The current value of this property is label1. Change it to be the text you see displayed in the screen shot on the previous page (i.e., **Enter your first name in …**). The label will extend off the form. So, change the value for the label’s AutoSize property to be **False**. Then, drag the resizing handles around the perimeter of the label control to make the label appear similar in size to that shown on the screen shot on the previous page.

1-4. Follow steps similar to those just described to add a TextBox control to your form. Set the **Name** property of this control to **firstNameTextBox**. The **Text** property for this control should be left blank.

1-5. Add the two Button controls to your form. **Name** these two buttons **greetingButton** and **quitButton**. Adjust their **Text** properties to match what displays on the buttons in the screen shot above.

1-6. The appearance of your form should now be similar to that shown on the screen shot on the previous page of this handout. Do not be concerned at this point if there are very minor differences in appearance. You will get more practice in fine tuning the appearance of your forms later. However, if there appears to be any major differences in your form from that shown in the screen shot, ask your instructor for assistance.  
  
The next set of steps will enable the actual functionality of the program. Our goal is to provide the Visual C# code necessary to enable the program to handle the following two user actions:

\* Clicking the Greeting button should display a personalized greeting on the screen, utilizing the name that the user has typed into the text box.

\* Clicking the Quit button at any time should cause the program to terminate.

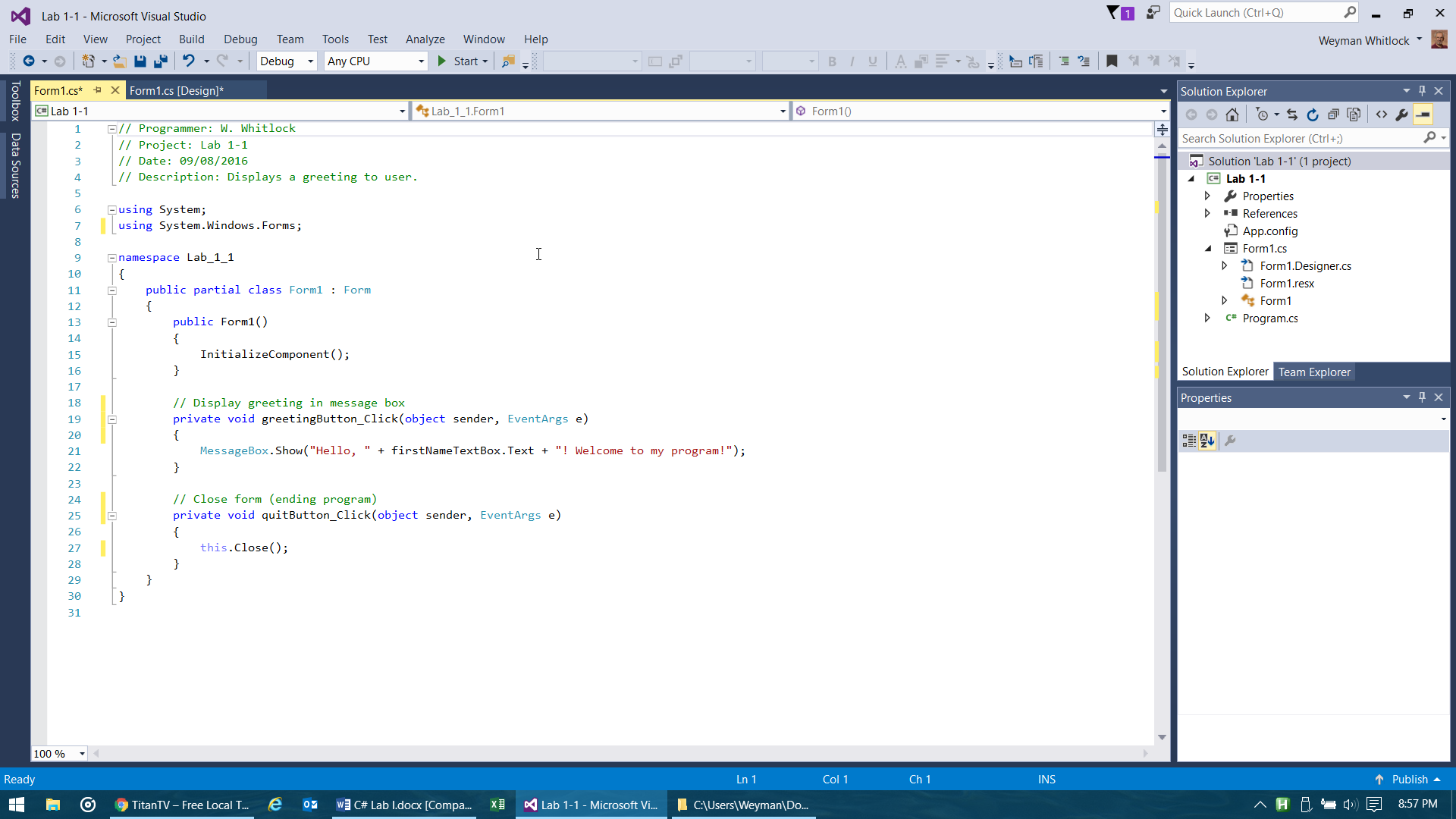
1-7. To add code to the program you have begun, it is necessary to change your view of the program from *design view* (which is what you have been working in) to *code view*. To do this, click on the **View** menu at the top of your screen (the third menu from the left in the menu bar). From the drop-down menu that appears, click the **Code** menu item (the first one in the list). Your screen will change to show the Code window associated with your program. Click to place your mouse pointer at the very top of this window, before all of the automatically generated code. Press the Enter key on your keyboard to insert a blank line above all other text, then move your cursor back to the top line and type in the following four *comment* lines to provide some initial documentation for your program. Each line must begin with two slash characters so that Visual C# will recognize the text as a comment rather than C# code. When you are done, the first four lines in your code window will look similar to this:  
  
**// Programmer: Your Name  
// Project: Lab 1-1  
// Date: 02/18/2017  
// Description: Displays a greeting to user.**

1-8. Now you are ready to create the C# code that will cause your program to react correctly when the user clicks on the buttons on your form. To accomplish this, you must create two *event handlers*. These are code blocks that *handle* the events that can occur during the running of your program. A user clicking a button is such an event. So, you will write an event handler for the Greeting button that displays a greet­ing message when the user clicks that button, and you will write an event handler for the Quit button that causes your program to end when the user clicks that button.

1-9. Return to design view by clicking the tab labeled **Form1.cs [Design] \*** that is above the code view window. Then, double-click on the **Greeting** Button. Visual Studio now returns you to code view where you can see that new code has been added to the bottom of your program code. The purpose of this code block is to handle the Click event of the Greeting button. The flashing insertion point is positioned within this new code block. You must now enter the code that you wish to execute when this event occurs.

1-10. Make sure that your insertion point cursor is on the blank line between the beginning and ending lines of the event handler code that was created above. Then, type the following line of code, exactly as shown:   
  
**MessageBox.Show("Hello, " + firstNameTextBox.Text + "! Welcome to my program!");**  
It is not important at the moment that you completely understand this line of code. You will receive more instruction about the use of message boxes later. For now, just understand that the purpose of this code is to read the user’s name from the text box on the form and to use it in a greet­ing message that will be displayed in a dialog box. (Remember, also, that it is a very good idea to thor­oughly comment your code. **See the screen shot below** for examples of using comments in your code.)

1-11. Now, follow a similar approach to what you just did in the previous two steps to create another event handler, this time for the **Quit** button. Thus, double-click on the **Quit** button in design view. Then, in the event handler code that appears, insert the following line of code between the beginning and ending lines of code for this event handler:   
  
**this.Close();**  
This simple line of code will cause your program to end when the Click event of this button occurs.

1-12. Your code should now look similar to this (scroll your screen if necessary to see all of the code):

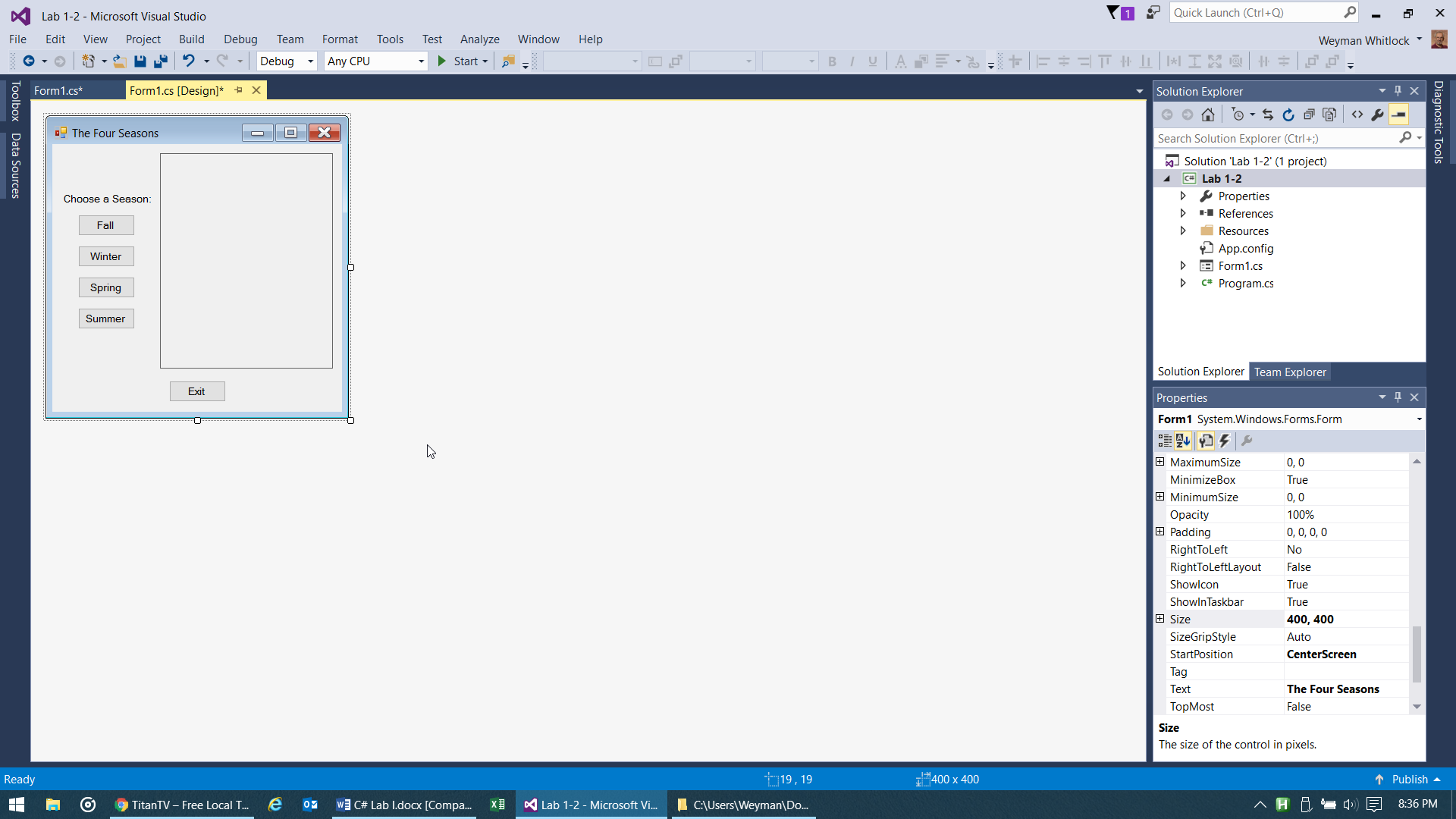
The initial comment lines and the code lines beginning with the word **using** are still in the program, just not shown here to save space.

1-13. You have completed the steps necessary for your program to correctly handle the user clicking on either of the buttons on your form. All that’s left is to test your program to see if it runs as expected. To do this, click on the **Debug** menu at the top of your screen in the menu bar. From the drop-down menu that appears, click the **Start Debugging** item. (You may prefer to simply press the **F5** key on your keyboard to run your programs, instead.) Your program should now start running and your form should be displayed on the screen. If you receive an error message instead, ask your instructor for assistance.

1-14. Test out your program by typing your first name into the text box and then clicking the **Greeting** button. Do you then see a dialog box appear on the screen with a greeting message for you? If not, see your instructor for assistance. Click the **OK** button in the dialog box to close it. If you like, you can try other names in the box to see the results. Now, see if your program ends as expected when you click on the **Quit** button. If all goes well, your form should close and you will be returned to the code screen where you were before running your program. If this does not happen, ask your instructor for assistance. (If you wish to further edit your form’s appearance, switch to design view and experiment as you please.)

1-15. When you have finished the steps listed above, click on the **File** menu at the top of your screen in the menu bar. From the drop-down menu that appears, click **Save All**. Visual Studio saves your program to the location you specified earlier (e.g., your USB drive).

1-16. After you have saved your work, you may close the project. Click on the **File** menu at the top of your screen. From the drop-down menu that appears, click the **Close Solution** item. Your project will close. (However, Visual Studio will not close.)

* Now, you will create a second project with fewer step-by-step instructions provided to you. Make sure and ask your instructor for assistance if at any time you need it.
* Name this second project Lab 1-2 (which specifies C# Lab I, Project 2).
* The design view of your finished form for the second project (as seen in Windows 10) is shown below:

This form (Form1) includes:

A Label control

(**instructionLabel**)

Five Button controls

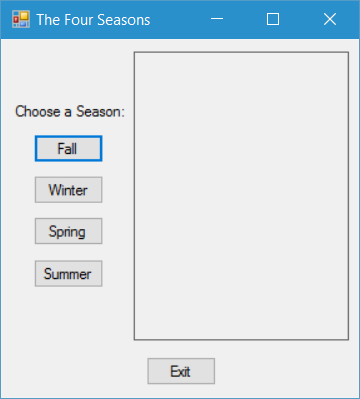
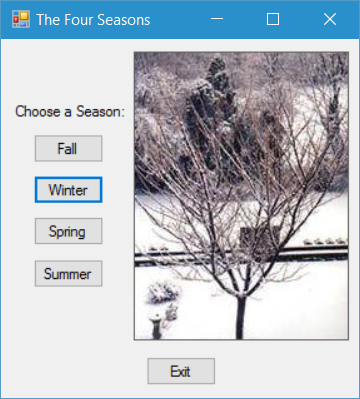
(**fallButton**, **winterButton**, **springButton**,

**summerButton**, **exitButton**)

A PictureBox control

(**seasonsPictureBox**)

(You can alter the value shown for your form’s **Size** property if necessary in order to match this appearance.)

* See pages 90-93 in the textbook for information on how to utilize PictureBoxes to display images.
* There are four graphic images that are available (in a .zip file) for download on the class Canvas site (in the *Modules* area). Download this file (C# Lab I Graphics.zip) to your computer. Make sure to note the loca­tion (folder) on your computer to which you download the file. (You may want to download to your USB drive.) After downloading this file, unzip it to obtain the four images.
* The graphic files that you downloaded are named Fall.jpg, Winter.jpg, Spring.jpg, and Summer.jpg. The program will allow the user to click the button for each season to see a corresponding graphic image display in the picture box. Follow the procedure described on p.91 of the textbook to *import* these four images as *resources* for your project. Repeat *Step 3* shown on p.91 for each of the four image files.
* After you have imported the four graphic files, and have exited the Select Resource dialog box (by clicking its OK button), the last graphic file that was imported should now be displaying in the PictureBox control. For this program, it is desired that the PictureBox control be empty when the program begins running. To accomplish this, the Image property of the PictureBox control must be edited. With the PictureBox control selected, locate the Image property in the Properties window and delete the existing property value entirely. The Image property value will then display as: **(none)**
* Now, there are two additional properties to set for the seasonsPictureBox control. Set the BorderStyle property to **FixedSingle**. And, set the SizeMode property to **CenterImage**. (See p.92 for setting details.)
* Next, it is necessary to have the clicking of each of the four season buttons cause the display of a different image in the picture box. To get you started on this, double-click on the first button (fallButton) to open up the Code view window with the Click event handler for the Fall button begun for you. In that event handler type the following line of code between the opening and closing curly braces:  
    
  **seasonsPictureBox.Image = Lab\_1\_2.Properties.Resources.Fall;**
* Create similar Click event handlers for the other three season buttons, associating each one of them with the appropriate graphic resource. (Change the word **Fall** above to the appropriate season for each button.) [Note: you must double-click each button to create its event handler, not just copy-and-paste existing code.]
* Finally, as you did for the first project in this lab exercise, create a Click event handler for the Exit button that will end the project when the Exit button is c licked by the program user. [Also, don’t forget to add comments to your code for each event handler, as well as an initial comment similar to that for Lab 1-1.]
* Save your project (click File in the menu bar, then select the Save All item) and then run it. The appearance of the form when the program runs should be as shown on the left below. Clicking the Winter button should change the form appearance to that shown on the right below. Test the other three season buttons to make sure that they display the correct images. Finally, test the Exit button to make sure that it ends the program.
* When you have completed this project, you may exit Visual Studio by selecting the File command from the menu bar, and then selecting the Exit command from the drop-down menu. (Don’t forget to take your USB drive with you, if you were using one on a lab computer.)
* Don’t forget to get additional practice on introductory user interface construction skills by attempting several of the Chapter 2 end-of-chapter Programming Problems on pages 112-115 of the textbook. (Recommended: try problems 4 and 5 on p.114; solutions to these two problems will be posted on Canvas.)  
  Such practice is crucial to mastering the skills presented in this chapter. Visit your instructor during his office hours to discuss questions or problems that you may have with these Programming Problems.