X414.20 Fundamentals of Software Development

Coding Lab 7 – Intro to Visual Basic

Please complete these lab steps only while in the Zoom meeting.

It is easiest to print these instructions so that you can refer to them during the lab. Your screen will be busy enough with both Zoom window and your Amazon Workspace window.

This lab will be different from the previous ones in that you will watch me perform each task and then you will perform it on your AWS virtual computer. We will not split into Breakout Rooms this week. In general, you should follow along with me and not move ahead on your own. I want to move through this as a class so that I can illustrate things that may go wrong and options you might take.

Task 1 – Create a brand new application called WelcomeUCLA.

1. Open Visual Studio 2017 (purple icon on task bar or in Start Menu).
2. If you are prompted to enter a Microsoft account ID and password, enter what you already have for Windows Live, Microsoft 365 / Office 365, Hotmail, or Microsoft Games. If you need to set up a new one, go ahead and quickly do this now.
3. On the Get Started page, click **Create new project**.
4. On the left, make sure that **Visual Basic** / **Windows Desktop** is chosen, and then on the right, select **Window Forms App**.
5. Next to Name, type **WelcomeUCLA**. Keep the Framework as it is.
6. Click **OK**.

Task 2 – Explorer the VB interface

1. Await the Designer screen, showing the form and other tools.
2. Note the **Solutions Explorer** and the **Properties** windows (also called "panes"). If not there, make these visible from the **View** menu.
3. Follow Keith to relocate various tools to different locations on the screen.
4. Make sure that the **Solutions Explorer** is at the top right and **Properties** pane is underneath it. This is the best way to view the screen for this course.
5. Save everything as is by choosing **File / Save All**. Take the defaults. Note where the files will be stored.

Task 3 – Rename the form both internally and externally

1. Click on the form in the center. This brings the form's properties into the **Properties** pane.
2. The "internal" name of the form is what is reflected in the Properties pane under **(Name)** at the top of the pane. It is currently **Form1**.
3. Change the internal name to **frmWelcome** in the **(Name)** property. Doing a lower-case abbreviation of the type of control, followed by a capitalized descriptive name, is called the "Canonical Naming System", and has been used a lot by VB programs in the past decades. It is not mandatory, but helpful if you are learning VB. If you are prompted to change anything else, choose No.

4. The “external” name of the form is its actual file name in Windows. It is currently **Form1.vb**.
5. **Each of you will use a slightly different external name that contains your initials.** Change the external name to **frmWelcomeUCLA*f*l.vb,** where *f* is your first name and *l* is your last initial. (So if your name is Bob King, it would be **frmWelcomeUCLAbk.vb**.) You will do this by going to the **Solution Explorer**, and using the same method you’d use to change a filename in the Windows or Mac OS operating system, change **Form1.vb** to the external name I’ve specified for you.
6. Note that the form title bar has now changed to reflect the new filename, and if you click on the form itself the **Properties** pane shows the name of the form as **frmWelcome**.
7. The final step is to ensure that the Project reflects that **frmWelcome** is the **Startup Form** (since a project could have more than one form and **Form1** no longer exists). From the top menu, choose **Project** then **WelcomeUCLA Properties…** (toward the bottom of the menu). Change the **Start Form** (mid-screen) from **Form1** to **frmWelcomeUCLA**.
8. Finally, do a **File / Save All** to secure everything.
9. You may execute this project by pressing the **F5** function key (or by clicking **Start** with the green wedge). Do so now. Inspect the running program. It has Minimize and Maximize buttons, a control menu (top left), a title bar and borders that are currently sizable. You can move it around screen by clicking and dragging the title bar. The program reflects the current styling of Windows forms as specified by the version of Windows and the .net Framework.
10. Click the **X** in the upper right corner to quit the program.

Task 4 – Size your form and set the border style and title bar

1. Click the **Design** tab to view the form. You may also choose **View / Designer** from the top menu.
2. Give the form a title: **WelcomeUCLA App**. You do this by clicking the form, then visiting the **Text** property in the Properties pane. After typing it and pressing ENTER, you should see this reflected in the form’s title bar.
3. Let’s size the form to 800 x 600 pixels. You may do this by clicking in the **Size** property and typing **800, 600**. You can also click the plus sign next to **Size** to break the property into its **Width** and **Height** sub-properties.
4. After seeing the form size change, execute the program using the F5 key. You should see the running form with the new size.
5. While running, note that you can move the mouse near the borders and click and drag the borders larger or smaller. We want to restrict the user from doing this.
6. Exit the program.
7. In the **Properties** pane, find **FormBorderStyle**. It is currently **Sizable**. Change it to **FixedSingle**. (For this course, you will never use any of the other styles).
8. Now execute it again using F5. Note that you can no longer (as a user) change the borders.
9. Close the running program.

Task 5 – Add your first label to the form (the header--or think of it as a title)

1. Let’s add our first control to the form (by the way, remember that the form itself is a control). We wish to add a **Label**. Labels display information, either static text or something that can change at run-time. We wish to add a big header saying **Welcome to UCLA** toward the top center of the form.
2. Click the **Toolbox** over to the left edge of the screen.

3. Most controls we will work with are in the **Common Controls** category (a couple later will be under **Containers** or **Menus & Toolbars**).
4. Find **Label** and double-click it. This is the easiest way to add a control to the form. Keith will show you an alternative.
5. Click on an empty part of the form to see it. You should see **Label1** up in the corner. We are going to move it closer to its ultimate location and change both its name and what it displays.
6. Click and drag **Label1** over to the high center of the screen, as Keith is showing you. Don't be concerned with its precise location right now.
7. ALWAYS change the name of the control to something other than the default. Let's call it **lblTitle**, using our canonical naming system. Make sure it is highlighted (you'll see a little box around it with a little square in the upper left corner). Then visit the **(Name)** property in the **Properties** pane. Change the name to **lblTitle** there.
8. The last step changes the control's name but not what it displays. That's always in the **Text** property. Change the **Text** property to **Welcome to UCLA**
9. Let's make this bigger and bold. Make sure the label is still selected, and in the **Properties** pane, go to **Font** and click in the value area and then click the **…** to the right (that is called an *ellipsis*).
10. A dialog box appears. Change the **Size** to **24** and the **Font style** to **Bold**. Then click OK. (By the way, in this course, ONLY use the Microsoft Sans Serif font.)
11. Now run the program to verify the changes.

Task 6 – Center your label horizontally in the form

1. Visual Studio has great tools to help you line up things without having to "eyeball" it. We would like to center the **lblTitle** label horizontally on the form.
2. Make sure **lblTitle** is selected.
3. From the **Format** menu item in the top menu, choose **Center in Form**, then **Horizontally**. The label will now be centered properly. You will need to do this any time you change the location or the contents of this label.
4. Now run the program to verify the changes.
5. Do a **File / Save All**.

Task 7 – Add a label and textbox to hold a Name

1. We'd like to add the ability for the user to type a name on the form.
2. Using the **Toolbox**, create a **Label**, call it **lblName** and have it display the word **Name**. Move this over to the left where Keith shows you.
3. Next to this, using the Toolbox, create a TextBox. Textboxes are used for input as well as changeable output. Call it txtName and don't have it display anything (keep the Text property empty). Move it next to the **lblName** label.
4. Align the two by ensuring you see the magenta horizontal line. This aligns the "baselines" of the two controls.
5. Stretch out the name on the right side to accommodate a typical full name, by clicking and dragging the right-most little square over to the right.
6. Keith will show you some things about a "fuller" version of the textbox but just watch, don't do.
7. Now, run the program. You will see that you can enter text into the textbox, as well as to edit it. It will scroll if your text is longer than the box.

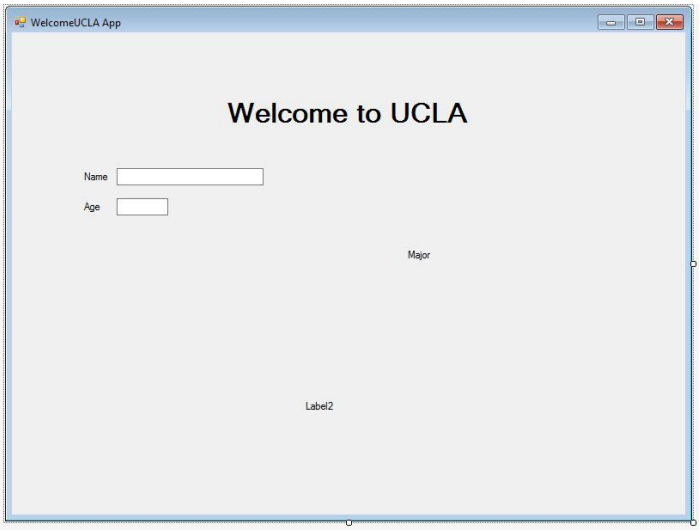
8. Exit the program.
9. Do a **File / Save All**.

Task 8 – Add a label and textbox to hold an Age

1. Add a label and textbox to hold an age. Label should be called **lblAge** with the text **Age**. TextBox should be called **txtAge** with empty text. These should be placed below the Name grouping. The labels should left-align and the textboxes should also left-align. Baselines should be aligned on the new label and textbox. The **txtAge** textbox should be shorter than the **txtName** textbox.
2. Run the program. Notice that you can now enter data into the age textbox. Note also that it doesn't restrict you from entering actual strings. We'll fix that next week.
3. Exit the program.
4. Do a **File / Save All**.

Task 9 – Add labels for upcoming controls: Major and Enrollment Room message

1. You will add two more labels to the form, as follows (See the sample below for placement)
    a. One mid-form to the right showing **Major**, with the name **lblMajor**.
    b. One centered toward the bottom with the name **lblEnrollRoom**. Keep the default text property for now.

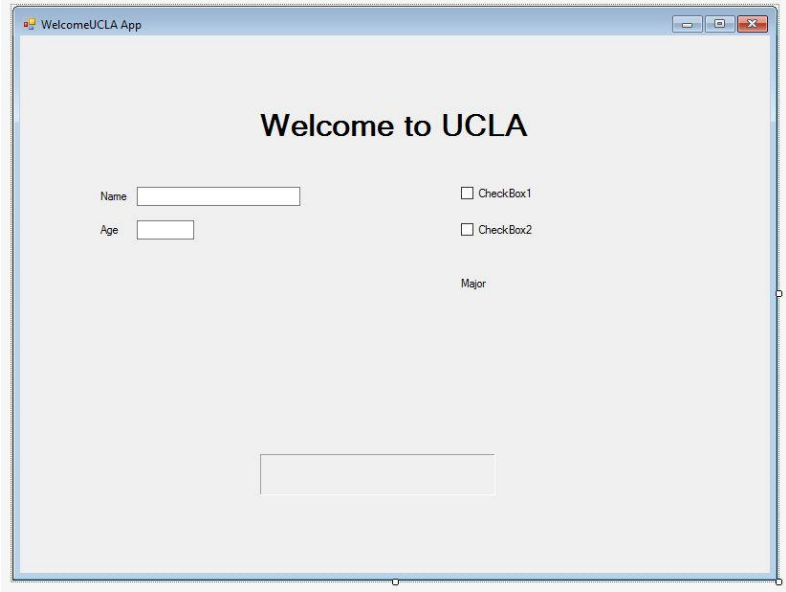


2. The **lblEnrollRoom** is going to display a special message later on that shows which enrollment room is appropriate for the selected major. We want this to have a special 3D border style and to expand to a specified size no matter what text is inside of it. Click on this label to select it.
3. In the **Properties** pane, choose **BorderStyle** and change it to **Fixed3D**.
4. Run the program so that you can see what this looks like. The text is recessed inside of a 3D border.
5. Exit the program.
6. Let's size the label. Normally, labels are auto-sized, based on the text displayed. We'd like a bigger size though. In **Properties**, choose **AutoSize** and change this to **False**. The label now has sizing boxes all around it.

7. Click and drag the lower right corner box to make it taller and wider, as shown by Keith. When done, you will see **Label1** in the upper left corner. Don't worry; we'll fix that in a moment.
8. Now we'd like the text inside to be displayed as a bit larger and bold, and in the center of the box horizontally, as well as vertically. Horizontal positions are Left-Center-Right and vertical positions are Top-Middle-Bottom. Let's change the position first.
9. Go to the **TextAlign** property for this label. Out of the 9 graphic boxes displayed, choose the one in the center both horizontally and vertically. After selecting it, you'll see the word **MiddleCenter** as the property's value. And the label's text is now floating in the middle and center of the borders.
10. Now change the font of what is displayed in this label to 16 point bold. Remember how?
11. Now center this label horizontally in the form. Remember how?
12. Erase the text that is inside the label. Remember how?
13. Run the program to view it. Notice the empty label? Later we will write code to have a string value display in this label.
14. Let change the color of the background in this label only. There are a zillion custom colors available, but in this course we will use one of the 244 "web" colors available. Keith will discuss the Windows color scheme colors, but you will not be using these. We want to alter the **BackColor** property and make it the color called **SeaShell**. Can you figure out how to do this?
15. Run the program to view it. (subsequent pictures of the form below won't show this color)
16. Choose **File / Save All**.

Task 10 – Add checkboxes for Student and Resident

1. Checkboxes allow a value to be on/off, yes/no, true/false, and are an efficient way to record this information. We want one textbox to reflect if the user is a student (or not) and another to reflect if they are a resident (or not).
2. Let's visit the **Toolbox** to grab a couple of **CheckBox** controls. Double-click twice on **CheckBox**.
3. Once we click on the form, we'll see two new controls that are checkboxes. Note that each has its own built-in label, so we do not need to separately create label controls for these. Move these into place as shown by the graphic. They should be left-aligned.



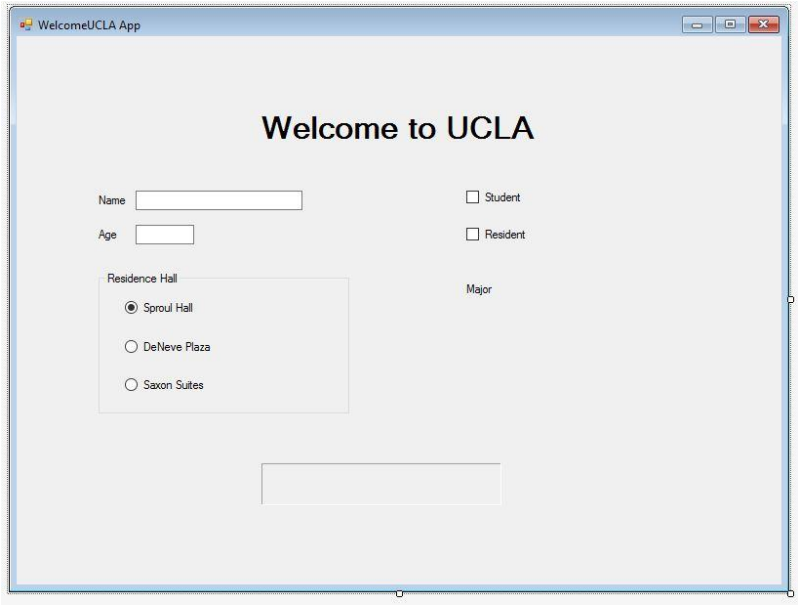
4. The top one should be named **chkStudent** and display the word **Student**. The bottom one should be named **chkResident** and display the word **Resident**. Do this now.
5. Now run the program and see how you can check and uncheck each of these. Please note that checkboxes don't have anything to do with each other. They are completely independent. They are NOT alternatives to each other (one of the 2, one of the 3, etc.).
6. Exit the program.
7. I like setting default values for checkboxes using code but let me show you how you can also do it directly. Let's assume that the user is a student and pre-check that before run-time. Click on the student checkbox and then visit the **Checked** property. **False** means unchecked and **True** means checked. Change the property to **True**. You will see the box checked in the Designer.
8. Run the program, and you will also see the box checked initially at run-time. It can still be changed by the user.
9. Exit the program.
10. There's a newer property (hehe, since 2003) that you can alternatively use to change the checkbox status. It is called **CheckState**. Its three properties are **Unchecked**, **Checked** and **Indeterminate**. The first two map directly to the **Checked** property's **False** and **True,** and if you change one property the other will change automatically. **Indeterminate** is a setting some Windows program use to indicate that the value is irrelevant at that time. We will not be using this in this course, so you may ignore it. However, go ahead and select **Indeterminate** to see what it looks like. And then change it back to **Unchecked**.
11. Choose **File / Save All**.

Task 11 – Add a groupbox and radio buttons for the residence halls

1. Radio buttons are alternatives to each other. Only one radio button in any group may be selected. (Ask your mature instructor about why these are called radio buttons….) You may have any number of them in a group. If they are directly on the form, all of them are automatically considered in the same group. However, it is much better to group them in a

groupbox. This allows you to have several sets of them. Plus it gives a much better design for the user. Let's set up a groupbox and three radio buttons representing residence halls.

2. We'll do the groupbox first. In the **Toolbox**, under **Containers**, double-click on **GroupBox**.
3. Click the form and see the new groupbox. Click and drag it (using the double-headed arrow) to a position under the name and age, left-aligned with the name and age labels. Make it a little bigger so the right edge is under the "o" in Welcome.
4. The control should be named **grpResidence**. And the text should read **Residence Hall**. We make this singular to indicate that only a single one will be selected.
5. Now let's add three radio buttons: **rdoSproul** (displaying **Sproul Hall**), **rdoDeNeve** (displaying **DeNeve Plaza**) and **rdoSaxon** (displaying **Saxon Suites**).
6. Each of these radio buttons is treated somewhat like a checkbox in terms of value. Each has a Checked property, and if true, the radio button will appear selected. And if one is True, the others are automatically set to false. Let's go ahead and make Sproul Hall the default selected hall. Think -- how do we do this?
7. Now run the program. Note how Sproul Hall is the default, but we can easily change to either of the other two during run-time. And notice how they automatically deselect when you choose a different one.
8. Your form should now look like this:



9. While a left-align for these three radio buttons is easy during placement, Keith will illustrate for you how the Format menu can help you align several things as well as to make the distance between them equal. You can play around with this during your personal time with the program.
10. Do a **File / Save All**.

Task 12 – Add a combo box for the Major

1. We plan to have five majors listed in what's called a **ComboBox** (in order): English, Cybernetics, Kinesiology, Mathematics, Music. A **ComboBox** is like a combination of a textbox with a bunch of radio buttons. It lets you choose a single value out of many alternatives, and it does this in a

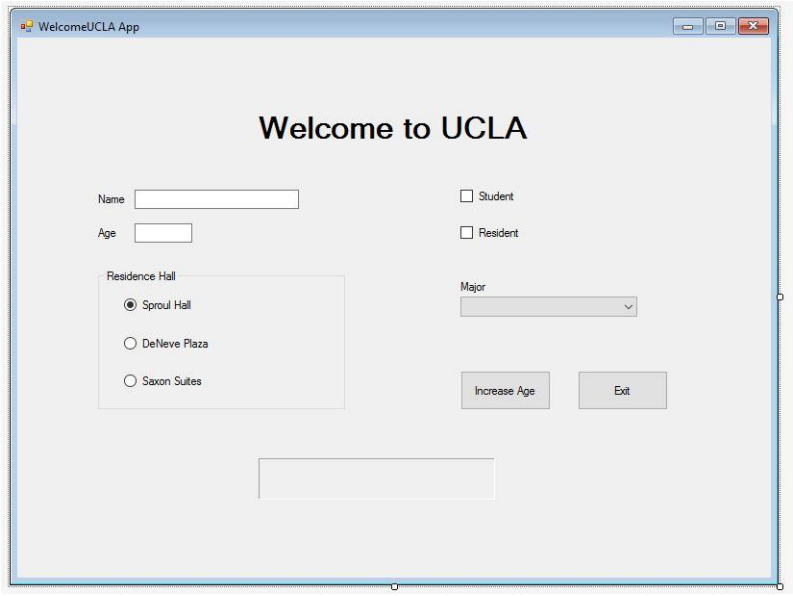
very space-efficient manner. These are three styles of **ComboBox**, and we'll be viewing two of them.

2. Create the **ComboBox** for major. It's in the **Common Controls** area of the **Toolbox**.
3. Drag the **ComboBox** directly under the **Major** label and make sure it is aligned.
4. Make it a bit wider.
5. Before we play with it, let's load it with its values. There are two ways to do this. One way is through the **Properties** pane. Choose the **Items** property and click on the ellipsis next to **(Collection)**. This brings up a **String Collection Editor** (fancy name for the place where you enter the values). The other way is to hit the little arrow on top of the **ComboBox**. This always gives you one of the most important properties to consider. If you go this route, choose **Unbound Mode / Edit Items…**. The **String Collection Editor** will appear.
6. Type the five majors into the editor, one per line, in the order I specified above. Click OK.
7. This is the default mode for ComboBox, and is actually NOT the common way you normally see comboboxes. Run the program to see how this mode works. Note that nothing appears in the combobox when we first run the program (we can fix this). And also, when clicking the little down arrow, you'll see the five majors. Select one, and you'll see it appear in the combobox as if it were typed in. The problem is….you can actually change its text or delete it and type whatever you want. Try this now.
8. Exit the program.
9. Comboboxes historically do NOT let you type whatever you want. They display the list the programmer specified, and you must choose one of those values. You can change your combobox to this behavior by changing its **DropDownStyle** property. When the value is **DropDown**, we get the more unusual behavior. Change this to **DropDownList**. Now you will see a different view of the combobox and the behavior will be more what you are used to. Run the program again after making this change. Now you can click anywhere on the box and no text insertion marker will appear. All you can do is to select what is in the box already. ***This should be your standard way of setting up any combo box.*** I'm sorry the default is the unusual way. Bad Microsoft!
10. Exit the program.
11. We'll worry about how to display a default major a bit later. Let's move on for now.
12. Choose **File / Save All**.

Task 13 – Add buttons for increasing age and for exiting the program

1. Buttons are like commands we can give to the form. We'd like to add an **Exit** button, plus another way that when clicked increases the age value (in the textbox) by one. Buttons by themselves do nothing. We have to attach "event code" to the button to get it to perform any actions. Every control has a variety of events it can respond it. Coming up we will discuss the most common event for each control. For a button, it is the **Click** event. When the user clicks the **Exit** button, we'd have code that exists the program. For the **Increase Age** button, we'd have code that adds 1 to the value in the **age** textbox.
2. Let's add to Buttons from the Toolbox to the form. Place them below the major. The left one will be called btnIncreaseAge (displaying Increase Age). The right one will be called btnExit (displaying Exit).

3. Keith will show you some tricks on how to make these larger and the same size. After you do the same, your form should look like this:



4. Run the program. You will see that the buttons do nothing, but graphically you do see that it is "clicked". In a moment, we will write code that reacts to the click event.
5. Choose **File / Save All**.

Task 14 – Add event code for clicking the Exit button

1. Code is added to "the back" of the form (a good way to visualize it). We will eventually have form_load code (which is executed when the form is first loaded, sort of like main() in a C program), other event-driven code (executed when a particular event happens to a particular control or object), and other code that we can invoke when doing one of the first two. All of these are in the form of subroutines. A subroutine is like a void function in C. It is called, but in the case of the first two above, the program automatically calls it if the form is loaded or if an event occurs.
2. Let's examine the code window. Choose **View / Code** from the top menu. All you will see is a small piece of "skeleton" code, showing **Public Class** frmWelcome and **End Class**. All the code for the form goes in between these two lines. Never erase or change these lines!!! They mean everything in the world to your application!
3. Switch back to the Designer view by choosing **View / Designer** or by clicking the **Design** tab.
4. Creating one or more event subroutines is how we write code for the 2nd case mentioned above. Let's create an event subroutine for when the user clicks the Exit button.
5. Double-click the **Exit** button. The code window should appear with some new skeleton code added to what's already there. It is the first and last line for the event subroutine. It is automatically named **btnExit_Click()**. Your code will go in between these lines. Keith will describe this to you in more detail.
6. Visual Basic code is line-oriented, with each statement starting on a new line. As you type the line, you will see options pop-up for you to choose. And when you cursor off a line or press ENTER, its syntax is immediately checked and the line will then be spaced out nicely (or will appear with a squiggly line underneath, indicating a syntax error).

7. The statement to tell VB to close a form (last or only form closed in a project will close the entire application) is **Me.Close()** Type this now. When you press the . you will see options appear. Start typing **close** and you will see it hone in on the word. Once the word is selected in the list, press the **Tab** key to finish and then **cursor down**. It will automatically add parentheses. **Me.Close()** is an example of a *method call*. **Me** presents the current form, and **Close()** is the method. A method is an action that an object already knows how to do. This statement is basically saying "Hey form, close yourself!"
8. Run the program. You will find that clicking on the Exit button automatically invokes **btn_Click()** and exits the program.
9. Return to the **Designer** view.
10. Choose **File / Save All**.

Task 15 – Add form_load() event code to initialize the Major combo box

1. While initializing a textbox or label to a string works using the **Text** property, it does NOT work using the Text property in a DropDownList-style combobox. We must initialize this instead by using code. The code should be run when the form first loads, so we use the form_load() event subroutine to do this.
2. To add code to the form_load() subroutine, double click anywhere on an empty area of the form. The code screen should automatically appear, with the cursor in-between two skeleton lines. The top of these lines shows the name of the form with **_Load** appended to it. We commonly call this the form_load() subroutine, even though it is named specifically.
3. Add the statement **cboMajor.Text = "Mathematics"** to make Mathematics the default major. Type slowly so that you see what happens as you type. You hone in quickly on the **Text** property in the list, and when typing the first **"** the second automatically appears. This is essentially how you assign a string literal to a property or to a variable in VB. We are actually treating the **Text** property of **cboMajor** in the same manner as a variable here, placing a string literal inside of it.
4. Run the program and you will see that the Major combobox now has Mathematics as its default value.

Task 16 – Add event code for clicking the Increase Age button

1. Let's add code for increasing by 1 whatever number is in the Age textbox. We need a simple assignment statement to do this, just like we would in C. Except we are not assigning something to a variable yet. We are doing it to the **Text** property of **txtAge**. Again, the **Text** property of thing can be thought of like a variable.
2. Where will this code go? It's based on the event—the action the user creates. Therefore, we want this code to belong to the click event of the Increase Age button.
3. Make sure you are in Designer view. Then double-click the Increase Age button. An event subroutine skeleton called **btnIncreaseAge_Click()** appears.
4. The only statement we'll have here will be the one that increases the value in txtAge.Text by 1. That can be either **txtAge.Text = txtAge.Text + 1** or **txtAge.Text += 1** (remember I told you that most language use this form of an increment now). Add this statement to the **btnIncreaseAge_Click()** subroutine.
5. We can text this easily now by running the program. Run the program and place a value in the Age textbox, such as 35.

6. Now click the **Increase Age** button. You will see the value in the Age textbox increase to 36. Doing this repeatedly keeps upping the value.
7. What happens if the value in the textbox is not a number? We haven't talked about data types yet, but realize that the Text property of a control is a completely flexible variable-like storage space that doesn't care about data type…..until you try to do math with it. If the math doesn't work (like adding a numeric value to a string), your program will crash. Go ahead and add the letters **xyz** to the end of the value in the textbox.
8. Now click **Increase Age**. The program will crash, giving you an error message. Keith will describe the details of the message, but it basically says you can't add a number to a string.
9. The program is in a state of suspended animation. To continue working with the program, you must **Stop** it. Click the little red VCR/DVD style box in the toolbar above to stop the program. You may then either correct the code or run the program again. You cannot do anything meaningful until you stop the program after an error.
10. Run the program again. This time do NOT enter a value into the Age textbox.
11. Click **Increase Age**. The program will again crash. Even though you never typed anything into the textbox, there was a value in it—the *null* string. That's a string with no characters in it. And we can't add a number to ANY type of string.
12. You will be adding *validation code* to the program next week so that this error won't happen again. For now, let's move on.
13. Return to the **Designer** view.
14. Choose **File / Save All**.

Task 17 – Add event code for initializing the Student checkbox to checked

1. Let's add a statement so that the Student checkbox is automatically checked upon entry to the form. Think about it—where does this statement go?
2. Changing a property is very much like an assignment statement. We want this checkbox to be checked, so that means we'd like the **Checked** property to be **True**. Add the statement **chkStudent.Checked = True** to the place where you think this should go.
3. Let's add a comment to this statement. After it, place a single apostrophe and then type a comment like *' Assume user is a student upon form entry*
4. Test this out by running the program.
5. Choose **File / Save All**.

Task 18 – Add event code for changing the status of the Student checkbox

1. We'd like to enable and disable some controls based on the status of the Student checkbox. Every control has an **Enabled** property that is either **True** or **False**. By default, they are all **True**. There's also a **Visible** property that makes things appear and disappear. It is also based on **True** and **False**. To selectively change some of these based on the Student checkbox, we need an **If** statement. You can use an **If** statement in a variety of ways, similar to what we did in C. There's a plain **If-Then** statement, an **If-Then-Else** statement, and a linear nested **If** statement. Right now, we will do an **If-Then-Else** statement. An **If-Then** statement is similar, except it doesn't have an **Else** part. If no need for an "else" part, leave off the **Else**.
2. Because we'd like to do the enabling or disabling when the Student checkbox is clicked (and its status changes), the event code needs to be attached to the Student checkbox. The event we

are responding to is the CheckedChanged event. It is not the Click event. The VB designers made it the CheckedChanged event so that the event happens even if it the code (rather than the user) that makes the change in status.

3. In the **Designer** view, double-click on the Student checkbox. Code view should appear with skeleton subroutine appear: **chkStudent_CheckedChanged()**.
4. Unlike C, VB **if** statements contain a full line with the word **if**, the Boolean expression and the word **then**. Type the line: **if chkStudent.Checked = True** and then press ENTER. Automatically a **Then** and an **End If** appear. So VB frames the entire selection statement with **if** at the start and **End If** at the finish. This way nothing like curly braces are needed (like in C).
5. Where the cursor is placed is automatically the "then" part of the statement. All statements are automatically indented here. If the Student checkbox is checked, we would like to enable the Major label and combobox, and make the Enrollment Room message appear. If it is unchecked, then we'd like to disable the Major label and combobox and make the Enrollment Room message disappear.
6. Where the cursor is presently, type the three statements that will enable the Major label and combobox, and make the Enrollment Room message appear (based on the properties I mentioned above).
7. Now that you've finished the "then" part, we need to do the "else" part. Type the word Else on a line below the three you just typed, and press ENTER. **Else** is automatically placed in alignment with **If** and **End If**.
8. You may now place the "else" part statements below the word **Else**. They will be automatically indented. Type the three statements that will disable the Major label and combobox and make the Enrollment Room message disappear (based on the properties I mentioned above).
9. Run the program to test it.
10. Choose **File / Save All**.

Task 19 – Add event code for changing the status of the Resident checkbox

1. Let's do something similar for the Resident checkbox. When it is checked, the residence halls should be enabled and when unchecked, they should be disabled.
2. How do we create the event subroutine for this?
3. How do we write the If statement for this?
4. Before you start typing a bunch of statement to enable or disable the residence hall radio buttons, realize that enabling or disabling the group box will automatically enable or disable everything inside of it. So write the one appropriate statement in the "then" part and another in the "else" part to accomplish this.
5. Run the program to test it.
6. You probably found something wrong at the outset. The Resident checkbox was unchecked, yet all the residence hall radio buttons were enabled. What simple statement do we need to add to the program to ensure that when the form is loaded, the radio buttons are all disabled? Where do we add this?
7. Run the program to test it.
8. Choose File / Save All.

Task 20 – Submit your files in Canvas

1. Save your project.
2. Re-open your Google Chrome. Be sure to choose Chrome from inside your AWS, not the one on your physical computer.
3. You should be in Canvas. If not, browse there and login.
4. Go to **Modules**. Then **Module 7**. If it doesn't display automatically, click the wedge next to it to open things up. Click on **Coding Lab – M7** – **Intro to Visual Basic Controls and Coding**.
5. Click the big **Submit Assignment** button.
6. Under **File Upload**, click **Choose File**.
7. In the File box, navigate to your profile on **D**, then to **source**, then to **repos**, then to your solution folder and finally to your project folder.
8. You will submit two .vb files. One contains your designer spec for the form, and the other is the code in the form. For each of these .vb files you generated tonight:
    a. highlight the file
    b. click it and then click **Open** (or you could simply double-click the file).
    c. The file should appear next to the **Choose File** button.
9. Remember, you will need to choose **+Add Another File** for the second file. **There should be a total of 2 files.** If your initials were BK, they would be ***frmWelcomeUCLAbk.vb*** and ***frmWelcomeUCLAbk.Designer.vb***. Please do NOT include any other files.
10. You may add a comment to the **Comments…** area if you wish.
11. Then click **Submit Assignment**. The button will change to **Submitting** and then you will be returned to the Coding Lab screen. Assignment should be marked as Submitted! In the upper right corner.
12. Later, to check your score, click on **Grades** in the second-level menu on the left.
13. If I haven't graded your program yet, you will see an icon. Other, you will see a score. You can click on the name of the assignment to see my comments and to issue further comments or respond to mine.
14. Coding Labs will typically receive a score of 5 if you completed the lab and your submission was substantially correct. You may receive a score of 4 if there are some problems, and 3 if it is incomplete. You will receive a 0 if you did not participate in the lab. Ignore any "Total" scores that Canvas may show. They are meaningless.

Congratulations on completing your sixth Coding Lab!!!