Programming Logic - Beginning

152-101

Unit 4 - Usability Lab

5 points

Name _____

Score ____ / 5

Update Value _____

Make all corrections and resubmit to earn update points

Overview

There are a number of things you can do to make your form more *user-friendly*—to increase its *usability*.

1. OneDrive or my website, download a copy of the Unit 4 Bowling starter program.
2. Open the Bowling solution, open the form in Design View and then open another tab in Code View.

Many users are quite adept at navigating forms using the keyboard. Your application should support those skills as much as possible.

Setting Tab Order

Users expect to *visit* the form fields from left to right, top to bottom. The order the fields are visited is called the *tab order*. The tab order is set with the TabIndex property on each control. The default value for TabIndex is 2147483647 to ensure that the default value does not overwrite any tab value you are setting.

3. In Design View, click the control you wish to change, then change the TabIndex property value to be the order you wish. The first item will start with index of 1, and subsequent items will be numbered in the order you wish for them to be visited by a user who is tabbing.

   Don't worry about the tab index of the labels on the window (the user can't visit them anyway), but if you want to include them, it doesn't hurt.

4. Click the TextBox for the bowler's name. Change that index to 0. Now click each input in turn moving down the form and change the value of this tab. You can either change it in the properties, or you can type TabIndex – "3" in the xaml where the "3" is whatever number this particular control is in the tab cycle. When you get to the buttons, click them left to right. When I do this, btnExit is given tab index 7 (I don't include the labels in the tab index).

5. Start the program. Because the bowler's name now has tab index 0, it should already have *focus*. Focus is an object-oriented term for the control that is currently accepting input. TextBoxes designate focus by the flashing cursor. Other objects designate focus in different ways.

   Test the form's tab order by pressing the Tab key on the keyboard repeatedly. The fields should be visited in the proper (left to right, top to bottom) order. Note how the DatePicker and buttons designate they have focus.

   If the tab order is incorrect, go back to step 4 and try again.

Access (Shortcut Keys)

Sometimes, users want to jump around on the form—not following the tab order. You can simplify this process as well by defining *access keys*. Access keys are keys that when pressed with the **Alt** key set focus to a control. If the control is a button, the button is also clicked.

To define access keys, you use the control's Text property. Select the letter in the control's text that you'd like to use as the access key. Typically, this is the first letter of the text but **access keys must be unique for every control on a form**. To designate a letter as the access key, simply type a _ before the letter in the Content property.

6. Select btnCalculate. In its Text property, add a _ before the beginning "C". Press Enter to accept the Text property changes. Note btnCalculate on the form now has an underline on the access key letter. This is how Visual Studio apps designate access keys to **users**.
7. Select btnClear. Because we already used "C" as the access key for btnCalculate, we'll have to pick a different letter for this button's access key. Let's use "L". Add the **_ before** the (lowercase) "L" in Clear.
8. Select btnExit. You might be tempted to use "E" as the exit button access key but most Windows programs use "X" and users will expect this of your program. The primary reason "E" is not used is most programs include an Edit menu. Alt-E typically accesses this menu. Make "X" the access key for the Exit button. The Content property will be E_xit for this one.

9. Run the program. If your version of the program does not include some test bowling scores, enter three bowling scores. Click in the bowler name TextBox to give it focus.
10. Wait a minute—where are the underlines on the button access keysBy default, most Windows versions hide the access keys till the user presses the Alt key (even briefly). This is a Windows *feature*-it's supposed to work that way. No worries—the access keys work even if the underlines are not visible.
    Tip: There is a Windows setting that can be changed to always display access keys.

    Tip: On the Mac, press Command-Option instead of Alt.
11. Press Alt-C. The Calculate button should have been clicked and the results displayed.
12. Press Alt-L. The Clear button should have been clicked and the form cleared.
13. Press Alt-X. The form should close.

Defining the Default and Cancel Buttons

To provide even more user-friendliness, you can define a form's *Default button* and its *Cancel button*. The Default button is automatically *clicked* whenever the user presses the Enter key. The Cancel button is automatically *clicked* when the user presses Esc.

Many windows have a Default button, but for various reasons you'll learn about later, I don't usually include a Cancel button. However, in the Unit 4 programs you are required to demonstrate the ability to define Cancel buttons.

Tip: Many web pages also have *Default* buttons. When you press Enter, these buttons are automatically clicked.

Like you will in the Unit 4 programs, let's set the Calculate button as the Default button and the Clear button as the Cancel button.

14. Select the **button.** The isDefault property and the isCancel property will be where you set the individual buttons with this property.
15. Scroll through the properties list and change the IsDefault property to btnCalculate and the IsCancel property to btnClear.

    Note the change to the form's appearance—the Calculate button is now highlighted. It almost seems to have focus. This is the Windows way of designating this as the Default button—the button that is clicked when the user presses Enter. There is no visual designation of the Cancel button.

16. Run the program. Add test data if necessary then click in the TextBox for the bowler's name. Note the Calculate button is still highlighted. Press Enter. The calculate button should be *clicked*.
    Press Esc. The form should clear.
17. Once again, give bowler name focus. Press the Tab key until the Calculate button has focus. In addition to the highlighting the Calculate button also receives a (faint) dotted border. Press the Tab key again. The Clear button receives focus but note the highlighting has been removed from button Calculate. When a button has focus, it automatically clicks when you press Enter. Since button Clear has focus, button Calculate shouldn't respond to Enter. Whenever another button has focus, the Default button feature is disabled. As soon as some other control gets focus (select bowler name again), button Calculate is highlighted again.

Setting Focus in Code

The final usability feature we'll implement is to automatically set focus when the user clicks a button. This can save the user a couple of steps by anticipating what the user will likely want to do next.

18. Run the program and click the Calculate button. Note where focus is. It remains on the clicked button. Click the Clear button—same thing happens.
19. What do you think the user is most likely to want to do after calculating? If there were a print button, giving it focus (so it's ready to click on Enter) might be a good idea. We don't have one, so after calculating let's anticipate the user will want to clear the form and start over.

20. Access the code. Scroll to the end of the btnCalculate_Click event. Add the following command:

```
btnClear.Focus();
```

    I like to add a blank line above this command but it's not required.
21. Run the program and click the Calculate button. As we expect, btnClear automatically receives focus, ready for the user to press Enter to clear the form.
22. Where should we set focus after clicking button clear? Some might suggest btnExit, but why would the user want to exit immediately after clearing the form? Instead, let's set focus to the first input (txtBowlerName in this case).
23. Access the code and scroll to the end of the btnClear_Click event. Add the following command:

```
txtBowlerName.Focus();
```

24. Test the program. Click Calculate. Press Enter. Focus should be in the bowler name TextBox.

    Note the user can apply these features in any combination.

    Enter to Calculate. Enter again to Clear.
    Alt-C to Calculate. Enter to Clear.
    etc.

25. We don't add a focus command to button Exit (the form's closing) but just about any other button should set focus appropriately after completing its processing.

    We could set focus in Form_Load, but normally the default behavior (automatically focus on item with first tab index) works just fine.

    Tip: Because these set focus commands are so important, they are typically included in the pseudocode and flowcharts by the program designer. Be sure to add them to your designs.

26. If you are in the lab with the instructor present, demonstrate the following (in order). If the instructor is not available, zip and email your copy of the Bowling application.

- ❒ Run the program. If necessary, enter test scores.
- ❒ Demonstrate the tab order stopping when focus returns to the bowler name
- ❒ Press Alt-C to calculate.
- ❒ Note where focus is.
- ❒ Press Alt-L to clear.
- ❒ Note where focus is.
- ❒ Press Alt-X to exit.
- ❒ Run the program again. If necessary, enter test scores.
- ❒ Press Enter to calculate (Accept button)
- ❒ Press Esc to Clear (Cancel button)
- ❒ Exit the program and claim your points.