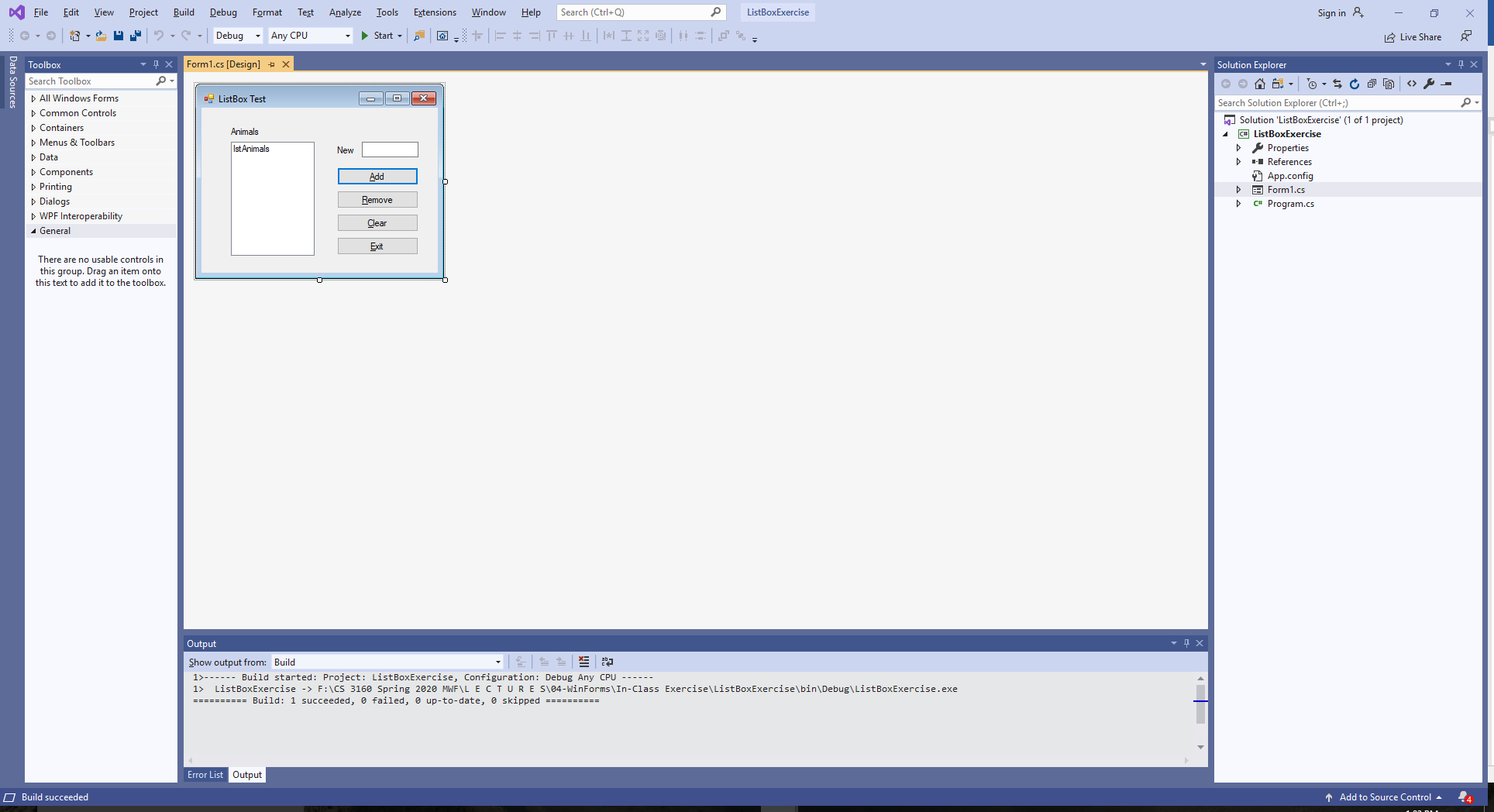
# Purpose

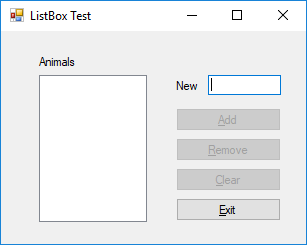
Practice creating a Windows Forms Application using Visual Studio 2019, setting properties, events, handling on click event, code-behind, and MessageBox.

# Instructions

1. Launch Visual Studio 2019, create a new C# Windows Forms Application (WinForm).
2. Remember to pin Toolbox and select Common Controls.
3. Create a form as pictured below.



1. When the form is shown to the user, it should initially look like this, i.e., some controls are disabled, the focus should be in the New textbox, and the only two buttons enabled should be the Add and the Exit. Utilize the Form Load event to enable/disable controls or set controls' Enabled property appropriately during design time.



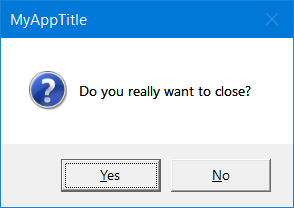
1. When the user types an animal name, enable the Add button.
2. When the user presses the Add button, add the animal to the list box (lstAnimals.Items.Add(txtAnimal.Text);
3. Enable the Remove buttons and the Clear button when there is at least one item in the Listbox. Set focus back to the animal text box but clear its contents (txtAnimal.Focus();)
4. When the Clear buttons is pressed, clear all contents of the list box control.
5. When use selects an animal in the list box, enable the Remove button.
6. When the user clicks on the Remove button, remove the animal from the list. If the list is empty, disable the Remove and Clear buttons. After removal, select top items in the list box, if any are left (lstAnimals.Items.RemoveAt(lstAnimals.SelectedIndex);
7. Set form AcceptButton to be the Add button and the CancelButton to be the Exit button.
8. Close the form when the Exit button is pressed (this.Close();)

# Additional Instructions ~ Bonus Opportunity [3 pts]

You are encouraged to work in pairs. Both students must be present in class during this lab activity. To earn BONUS points, both students should submit a compressed solution folder and complete a Word document via Canvas by midnight of the class activity day. Turn this sheet along with your compressed solution via Canvas.

Team member #1: \_\_\_Sidney Sanders\_\_\_\_\_\_ Team member #2: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. For this exercise, use the animal Listbox application we started last time in class. Make sure all of it is finished first before adding elements to this bonus opportunity.
2. Add a MessageBox() to ask the user if they want to exit when the Exit button or the "X" at the top of the form is pressed. Create an on FormClosing event handler for your code. Add message box shown below.



**DialogResult r;**

**r = MessageBox.Show("Do you really want to close?");**

|  |  |
| --- | --- |
| Question | Answer |
| Where did you put the code provided above? Record your answer to the right. | Double clicking on the exit button brings me to the form1.cs and I created a dialog box to confirm with the user of exiting the program |
| Test your app by clicking on the Exit button and on the "X" at the top of the form. Did the MessageBox come up in both cases? Record the findings to the right. | The message box did show up for both actions. One thing I cant understand is that the message box pops up twice, you have to hit yes twice fro it to work. After trouble shooting for a while I couldn’t figure out how to stop the second box from popping up |
| Modify the text box to have a more user-friendly title rather than the default. How did you do it? | After the dialog message, you add a comma and then you can add a custom title for the message box. |
| How did you get the “?” icon to display in the message box? | Adding another comma adding the MessageboxButtons YesNo hen I added another comma and added a MessageboxIcon.Question to have the icon on the message box along with the message. |

1. Add validation to the textbox on your form to ensure only letters are accepted. Implement validation via the on KeyPress event handler. Add code below to your event handler. Use the name provided below.

**private void FilterTextBox(object sender, System.Windows.Forms.KeyPressEventArgs e)**

**{**

**// Let the system handle the key for letters**

**if (Char.IsLetter(e.KeyChar))**

**e.Handled = false;**

**else**

**// Ignore: tell the system we handled the character**

**e.Handled = true;**

**}**

|  |  |
| --- | --- |
| Question | Answer |
| Test your app and try to provide numeric input. Record what happens in the area to the right. | When trying to type characters other than letters, nothing prints, which means you cant delete things either. |
| Copy code in the event handler to Copy/Paste butter. Now delete the event handler. Recompile. Record what happens in the area to the right. | there are no limitations to what is written in the text box |
| Paste it back in. Recompile. Record what happened in the area to the right. | Nothing is accepted other than alphabetical input |
| Now remove the event handler by selecting properties of the textbox control and select Clear. Recompile. Record what happened in the area to the right. | If all are cleared then numbers are aloud and the add button doesn’t get enabled because txtbox change is cleared out |
| What did you have to do to use the provided event handler name? | Add the event handler back into the properties window when selecting the txtNew text box |

1. Comment out code in on KeyPress handler. Add validation to the textbox via ErrorProvider to show a small error icon by the text box if numbers are entered by dragging the EventProvider control onto your form. Add event handlers Validated and Validating.

**// this code should be placed into the Validated event handler**

**TextBox tb = (TextBox)sender;**

**errorProvider1.SetError(tb, "");**

**// some of the code below will be useful in the Validating event handler.**

**TextBox tb = (TextBox)sender;**

**if (String.IsNullOrEmpty(tb.Text))**

**tb.Text = "0";**

**else if (!Regex.IsMatch(tb.Text, "^[a-zA-Z]+$")){**

**e.Cancel = true;**

**MyErrorProvider.SetError(tb, "Invalid animal name");**

**tb.SelectAll();**

**}**

|  |  |
| --- | --- |
| Question | Answer |
| After dragging ErrorProvider onto your form, change its name to MyErrorProvider. Describe how you accomplished that to the right. | By going into the property window in the form design and changing the (Name). |
| Change the blink rate of the error icon to 150. Describe how you did it in the area to the right. | Un the behavior category in the properties window you cansee the blinkrate is set to 250 so I changed it to 150. |
| What namespace did you have to include to use the Regex class? | using System.Text.RegularExpressions; |