Lab 1 Due next lab period – Week of April 18-22

Purpose:

1. Create a simple Windows GUI application using Visual Studio and VB.NET
2. Learn UI control event handling
3. Practise VB.NET programming

What to submit:

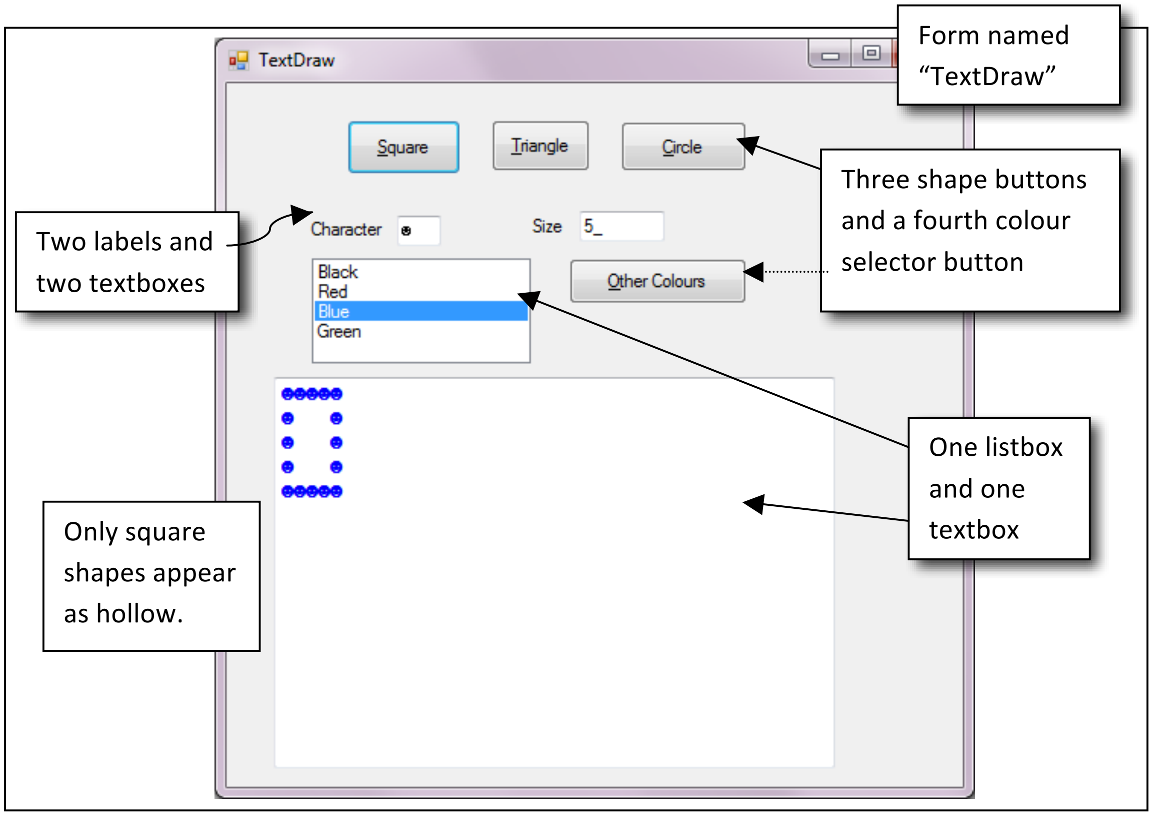
1. At the start of your next lab period, be prepared to demo your lab, and answer 1 or 2 questions
2. Submit your zipped project to D2L before your lab period.

Evaluation (18 marks):

1. Be prepared to demonstrate your solution at the start of next lab. If you are not ready at the start of lab or are not present, a mark of 0 will be given.
2. Your solution will be assessed based on 3 marks each:
   - Square, Triangle, and Circle buttons work correctly
   - Component Alignment
   - Naming of Components
   - Code quality and Comments
   - Modifying Output
   - Demo Questions
3. **All work must be individual.** Any plagiarism will result in a mark of 0 and possibly additional penalties.

Specifications:

Using the Microsoft Visual Basic IDE you will build a simple Windows GUI application, as shown below.


Form named "TextDraw"
Three shape buttons and a fourth colour selector button
Two labels and two textboxes
One listbox and one textbox
Only square shapes appear as hollow.
TextDraw
Square
Triangle
Circle
Character
Size
5_
Black
Red
Blue
Green
Other Colours

The three buttons labelled Square, Triangle and Circle have the following effect:

- Clicking on any of those three buttons draws that shape within the lower textbox using the selected character, colour and the size indicated from the other controls.
- The buttons have keyboard shortcuts corresponding to the first letter of the button text so for example, pressing *alt-S* is the same as clicking on the Square button.
- Default colour: black.
- Default character: asterisk (*).
- Default size: 4.
- Only the first character entered in the character textbox is used to draw the shapes.
- The size control should accept numbers 0 to 99 only.

Steps

1. Preliminary:

a. Read the instructions carefully and review the executable to understand what functionality is required in this lab

b. Create a new folder comp157 on your student network drive location to hold your Comp 157 labs.

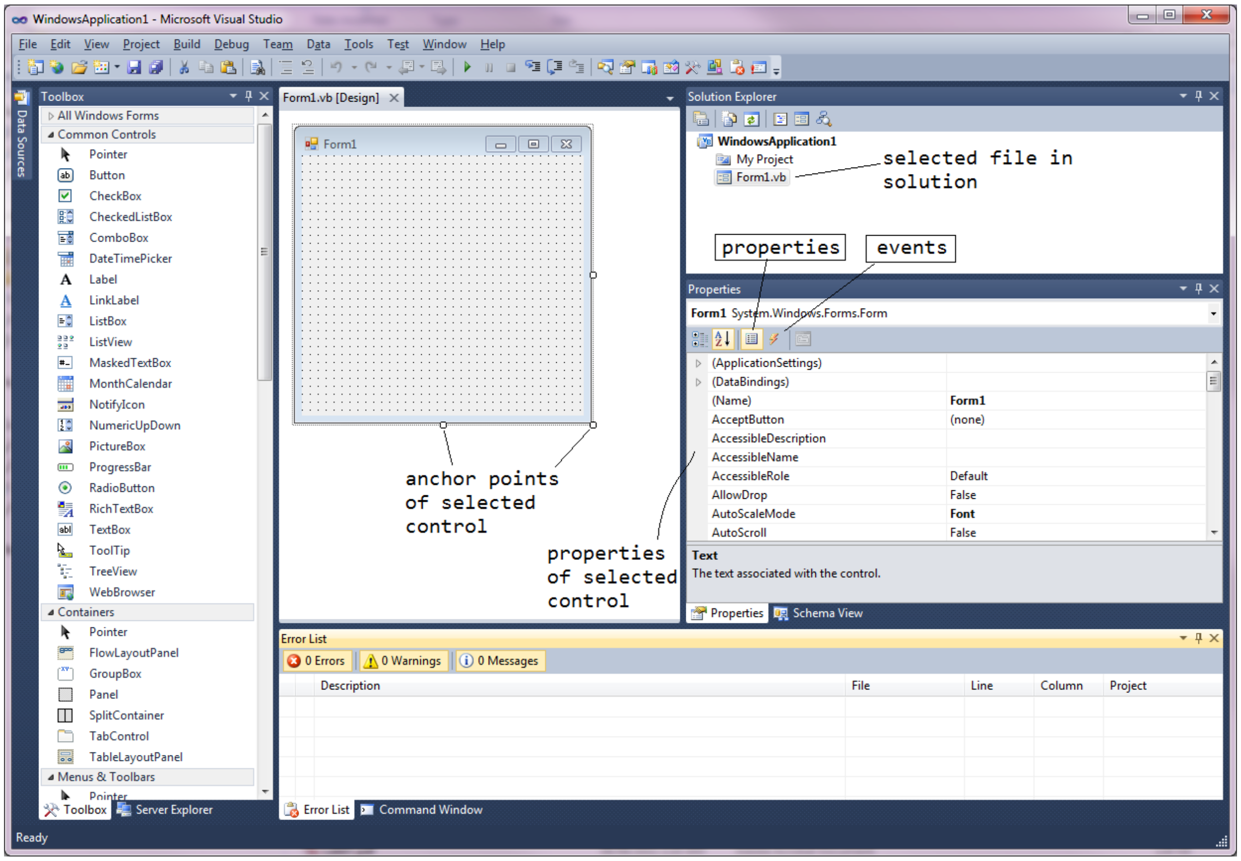
c. Start Visual Studio. If the main panel prompts you for a default project type, select General.

d. When the main Visual Studio panel appears, select from the menu Tools | Options. Select Projects and Solutions | General, modify the Projects Location entry so that it represents your student Comp 157 folder as defined in step 2.

e. Select Text Editor | All Languages and check the Line Numbers checkbox so that line number will appear in your code window. Click OK on the Options panel to close it.

2. Create a new project.

a. From the list of installed templates select project type Visual Basic Windows Forms Application with the .NET Framework and name it Lab1.



Form1 System.Windows.Forms.Form

(ApplicationSettings)	
(DataBindings)	
(Name)	Form1
AcceptButton	(none)
AccessibleDescription	
AccessibleName	
AccessibleRole	Default
AllowDrop	False
AutoScaleMode	Font
AutoScroll	False

Text
The text associated with the control.

b. Resize the form's default dimensions larger to 500 by 500 pixels. Check the form's `Size` property.

c. Rename the form's `Text` property to `TextDraw`.

d. Select the form and open the control toolbox panel (if it is not already opened).

3. Add three Shapes buttons:

a. Drag three `button` controls to the form and change the names of the controls from Button1, Button2, Button3 to `btnSquare`, `btnTriangle`, and `btnCircle`. This does not change the text on the button.

b. Set the Text property of the buttons to Square, Triangle, and Circle.
   i. The `Text` property of the button control is the text appearing on the button.

c. Place an ampersand character "&" as a prefix to the buttons' `Text` properties to make the first letter the alt-keyboard shortcut.

4. Add a TextBox Control:

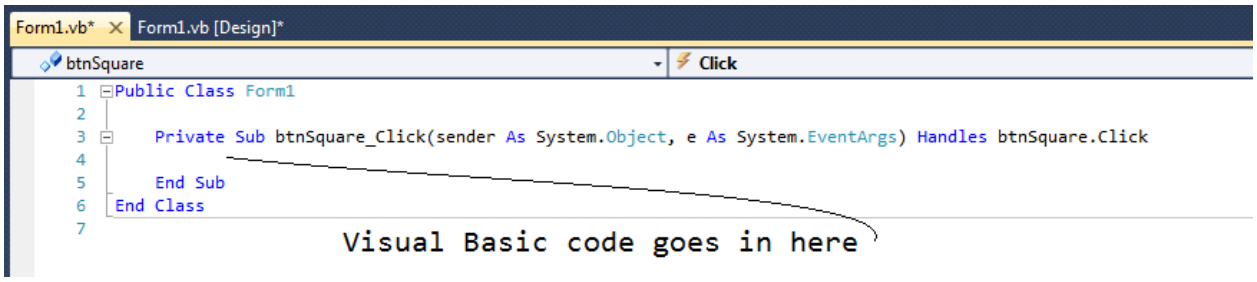
a. Drag a `TextBox` control to the form below the buttons and change the name of the control from `TextBox1` to `txtDisplay`.

b. Set the `Multiline` property to `True`.
   i. Once the TextBox control's `Multiline` property is set to `True,` you can resize its height to display several lines of text. Set the `Size` property to 280, 240.

c. Set the `Font` property to `Courier` New, size 12 point.

5. Add Event Handlers to Square Button:

a. Select the Square button on the form and then click on the "`Event`" (lightning bolt icon) button in the Properties panel. Find the `Click` event in the list of events, then double click on it.



```
Public Class Form1

    Private Sub btnSquare_Click(sender As System.Object, e As System.EventArgs) Handles btnSquare.Click

    End Sub
End Class
```

b. The Form1.vb window is shown to you with the stub for the button's click event. The procedure should be called `btnSquare_Click`.

   i. The VB code you write inside this `Click` event will create a string that generates a square pattern. To start though, the square shape will be filled in. Later you may edit the VB code within the inner nested loop so that the square shape appears as hollow
   ii. Declare a string variable `strOutput` and initialize it to empty string (""). Use the Visual Basic keyword `Dim` to declare variables inside VB procedures.
   iii. Declare an integer variable `intSize` and initialize it to 10.
   iv. Declare two integers `i` and `j`, which you will use in a nested loop below. (Note that the panel shows a thin yellow vertical bar just to the right of the line numbers. The yellow indicates unsaved Visual Basic code statements. Save the file and the bar colour switches to green for those new lines of Visual Basic program code now saved.)
   v. We want to clear out any shape already shown in the `txtDisplay` textbox. The VB statement is `txtDisplay.Clear()`
   vi. Define the nested loops using i and j which will add the asterisk character string to the end of the `strOutput` string, e.g. `strOutput = strOutput + "*"`

```
For i = 1 to 10    ' outer loop for each row
    For j = 1 to 10    ' inner loop for each column
```

```
            ... you add in the VB code...
        Next j
    Next i
```

   vii. A new line is output at the end of the inner loop `strOutput = strOutput + vbCrLf`
   viii. When the nested loop process is complete, assign the completed string to the `txtDisplay` control. `txtDisplay.Text = strOutput`

  c. Save your work.

6. Test your work:

  a. Press the green arrow pointing right in the toolbar, or selecting Debug | Start Debugging from the menu, or pressing the F5 key. You did not provide an Exit button yet, so you close the application window by clicking the X button in the top right.

7. Add Triangle Button:

  a. Repeat the coding process for the `Triangle` button. The `Triangle` should be displayed as a right angled triangle. The VB code for the `Triangle click` event is similar in structure to the `Square click` event with a difference: the triangle is shown filled in rather than hollowed out like the square. Hint: the number of asterisks shown on a row is equal to the row number.

8. Add the Circle Button:

  a. Repeat the process for the `Circle` button.
  b. The nested loop code for the circle is a bit more complicated.
   i. You will need to declare two additional variables `x` and `y` as `Single` in this Sub. The `x` and `y` act as co-ordinate values in an X-Y grid.
   ii. Declare `intRadius` as an `Integer` equal to 5.
    1. The outer loop makes `i` start at zero to `intRadius` times 2.
    2. Inside the outer loop assign `y` to `intRadius` minus `i`.
    3. The inner loop makes `j` start at zero to `intRadius` times 2.
    4. Inside the inner loop assign `x` to `j` minus `intRadius`.
   iii. Check if `x` squared plus `y` squared is less than or equal to `intRadius` squared. If it is, then append the asterisk character to the `strOutput` string; if not, then append a space character. The caret (^) symbol is used in VB.NET for exponentiation.

9. Test your Code:

  a. Confirm the `Square` button draws a square of size ten in the lower textbox.
  b. Confirm the `Triangle` button draws the triangle of height ten.
  c. Confirm the `Circle` button draws a (very rough) circle shape of radius 5. The VB code can be slightly amended to compensate for the difference between the height and width of the character. By adding an extra trailing space within the strings the circle can appear a bit more circular.

11.Add Label and Textbox:

a. Edit the form design (select the Form1.vb [Design] tab) drag a label and textbox to the form just below the `Square` button. You may have to adjust the `txtDisplay` textbox lower on the form to make space.

b. Change the `label` text to read `Character`.

c. Change the textbox name to `txtInputChar`.

d. Change the `Text` of the textbox property to "*".

e. For this control's `TextChanged` Event you will write some VB.NET code to check if this control's text length is greater than 1. If it is, then set the text to the first character. To refer to the form's textbox in VB.NET , you write

```
If txtInputChar.Text.Length > 1 Then
```

Use the VB substring method to extract the first character and set it to the `Text` property.

```
txtInputChar.Text = txtInputChar.Text.Substring(0, 1)
```

What this does is not allow more than one character to be entered into this textbox. This is actually the long way of doing it – there is an easier way you will see in the next control.

12.Modify the three shape buttons' `Click` event code

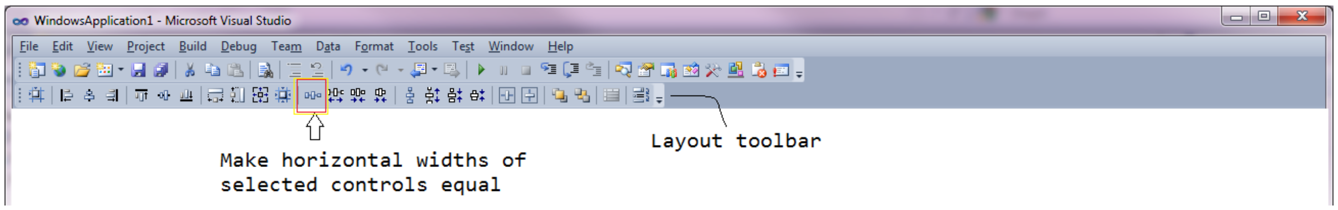
a. so that instead of the asterisk, the `txtInputChar.Text` is used.

13.Test your code

a. Verify that the buttons draw the shapes using the character you specify in the textbox.

14.Format your buttons:

a. Drag out a rectangular area on the design form that encloses just the three shape buttons. Use the menu's Format | Horizontal Spacing to ensure even spacing among the three shape buttons and the shape buttons are aligned at the bottom edges.


WindowsApplication1 - Microsoft Visual Studio
File Edit View Project Build Debug Team Data Format Tools Test Window Help
Make horizontal widths of selected controls equal
Layout toolbar

15.Add a Label Control:

a. Edit the form design again and drag over a label control and a `MaskedTextBox` control to the right of the character controls.

b. Change the label to read `Size`.

c. Select the `MaskedTextBox` control and change the `Text` property to 10.

d. Change the `Mask` property to 00 so that only number 0 to 99 will be accepted in this control.

16.Modify the `Square, Circle` and `Triangle` button's `Click` event code

a. to set `intSize` to `MaskedTextBox1.Text.` For `Circle`, set `IntRadius` to `intSize` divided by two.

17.Test your code:

a. Check that the buttons draw the shapes using the size you enter. Save your work.

18.Add a ListBox Control:

a. Resize the form length dimension to at least 625 pixels.
b. Drag a `ListBox` control to the form.
c. Change the name property to `lstColour`.
d. Resize the `ListBox` so that four rows of text lines can be displayed.
e. We want to initialize colour names to this list box as the application starts so we need to define some code to this form's `Load` event.
    i. Select the form and then the form's `Load` event. In the form event code for Load, you will need to add in the colours to the list box.
    `lstColour.Items.Add("Black")`
    `lstColour.Items.Add("Red")` , etc for Blue and Green
    `lstColour.SelectedIndex = 0` ' makes the first element (Black) in the list the default

f. The Event `SelectedIndexChanged` is what the list box responds to when you make a change to the list box selection.
    i. Create the event handler and add VB.NET code that assigns `lstColour.SelectedIndex to intColour`, then uses a case statement on `intColour` to set `txtDisplay.ForeColor = Color.`NN where NN is the colour name (e.g. Black, Red, etc). For example, Case 0 would set `txtDisplay.ForeColour = Color.Black`

19.Add Another Button:

a. Drag over a fourth button control to the right of the form's `ListBox` control.
b. Label this button '`Other Colours`' with the first O as the keyboard shortcut (prefix the O with an ampersand character &).
c. The `Click` event for this button will bring up the display of the Windows built-in colour dialog and allow the user to select any of the dialog's colours for the shape. To do this declare `dlgColour` as a new `ColorDialog` object in the click event for this button:

```
    Dim dlgColour As New ColorDialog()
```

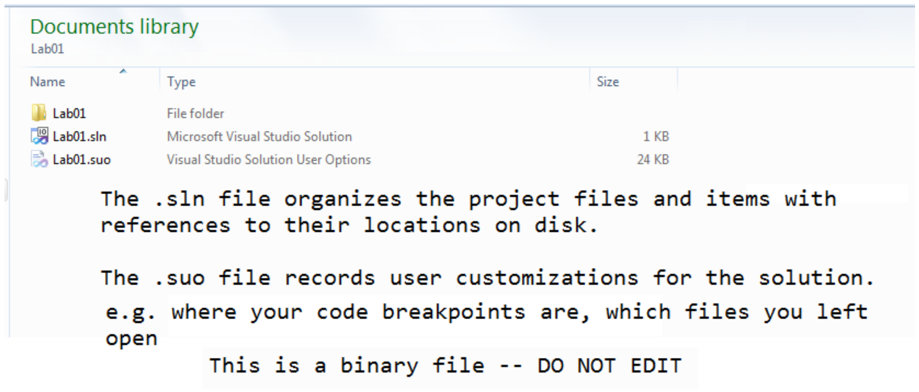
d. In the VB code set the following properties for `dlgColour`: `AllowFullOpen` to `False`; `ShowHelp` to `True`; and `Color` to `Color.Black`. Check if the `dlgColour's ShowDialog` method is equal to `DialogResult.OK` (indicating the user has made a colour selection from the dialog and has clicked OK to confirm), if so, then set the `txtDisplay ForeColor` property to `dlgColor's Color` property.

20.Add an Exit Button:

a. Add a new button at the bottom of the form. Label the button “Exit” and provide the VB statement `Application.Exit()` for its `Click` event so that the application will close when clicked.

21.Save your work:

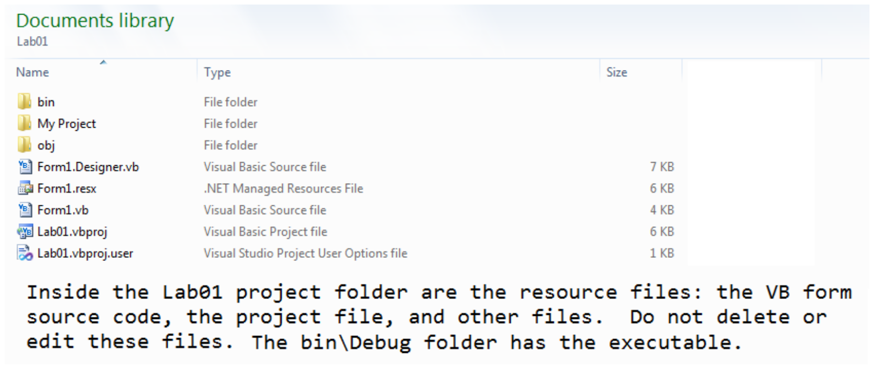
a. then open Windows Explorer to the folder location where your project is saved (see step 4). You should find the following in the folder:



Name	Type	Size
Lab01	File folder	
Lab01.sln	Microsoft Visual Studio Solution	1 KB
Lab01.suo	Visual Studio Solution User Options	24 KB

```
The .sln file organizes the project files and items with
references to their locations on disk.

The .suo file records user customizations for the solution.
e.g. where your code breakpoints are, which files you left
open
    This is a binary file -- DO NOT EDIT
```



Name	Type	Size
bin	File folder	
My Project	File folder	
obj	File folder	
Form1.Designer.vb	Visual Basic Source file	7 KB
Form1.resx	.NET Managed Resources File	6 KB
Form1.vb	Visual Basic Source file	4 KB
Lab01.vbproj	Visual Basic Project file	6 KB
Lab01.vbproj.user	Visual Studio Project User Options file	1 KB

```
Inside the Lab01 project folder are the resource files: the VB form
source code, the project file, and other files.  Do not delete or
edit these files.  The bin\Debug folder has the executable.
```