**Computer Programming I**

**Essential Standard 6.00 Apply tools to obtain and validate user input.**

**Objective 6.01 Apply Procedures to Develop Menus, ListBox, and ComboBox objects (3%)**

**Why Add a Menu?**

* Adding a menu increases the interactivity of your application.
* The menu allows the user to interface with your application.
* What menus can be added to your application form?
* The MenuStrip Control
	+ Adds a menu bar to your form
* Use the prefix **mnu** when naming your menu
	+ Example: mnuMain
* Once you click the MenuStrip control, it doesn’t matter where you click on the form. It will automatically place the menu at the top.

**The MenuStrip Control**

* Note that when you add the MenuStrip, it puts an icon in the gray area at the bottom called the **component tray**.


* To add code to your menu item, go to your code view.
	+ Right Click 🡪 View Code
	+ View Menu 🡪 Code
	+ F7
	+ Double Click the Menu Item
* In the Class Name List, select your menu object.
* In the Method Name List, select the appropriate method.
* In this case, we are selecting the Click event
* Type your code inside the Click event.
* For example if you wanted the application to close when Program is selected, type the following:
 Application.Exit()
* Note that an Autolist is provided as you type.
* For more information
http://msdn.microsoft.com/en-us/library/ms171649.aspx

**The ListBox Control**

* List boxes and combo boxes increase the interactivity of your application giving choices to the user.
* They allow the user to interface with your application.
* In this presentation we will look at how to add these new control objects to your application.

* The ListBox Control
	+ The ListBox displays a list from which the user can select one or more items.
	+ If the list is longer than the list box, a scrollbar is automatically added.
* The ListBox Control Properties
	+ (Name)
		- The name of a ListBox should start with the prefix **lst**
		(This is the lowercase letter L, not a number 1.)
	+ Items
		- Click in this property where it displays (Collection). A new dialog box called the String Collection Editor will appear. Type your list box items here
		 
		- Be careful with your spelling.
		- Make sure you hit the Enter key after the last item.
	+ Text
		- The text property returns the actual string value of the item.
	+ Enabled
		- The enabled property is set to true by default. If this property is set to false, the list box is not available for the user.
	+ Sorted
		- The sorted property is set to false by default. If this property is set to true, your collection of items will be sorted ascending.
	+ Visible
		- The visible property is set to true by default. If this property is set to false, the list box is not visible to the user.
* Using The ListBox Control
	+ SelectedIndex Property
		- This property allows the programmer to compare what the user has selected in the list box with a value.
		- The first item in the list box is at index position 0. If the user selects the first item in the list box, the SelectedIndex value will be 0, if the second item is selected, the SelectedIndex value is 1.
		- If no item is selected, the SelectedIndex value is -1.

|  |  |
| --- | --- |
| Item | Selected Index Value |
| First | 0 |
| Second | 1 |
| Third | 2 |

* + - This property is only available at runtime.

		 
	+ SelectedItem Property
		- This property allows the programmer to compare what the user has selected in the list box with the item itself as a **string** value.
		- The comparison is exact, so your spelling, capitalization, and spacing matters.
		- This property is only available at runtime.

|  |  |
| --- | --- |
| Item | Selected Item Value |
| First | Chocolate |
| Second | Strawberry |
| Third | Vanilla |



* + ListBox.SelectedIndexChanged & SelectedIndex
		- You will use a SelectedIndexChanged event to compare either index values or the item to determine the list box item selected



* + ListBox.SelectedIndexChanged & SelectedItem
		- You will use a SelectedIndexChanged event to compare either index values or the item to determine the list box item selected.
		- Your string must match the item *exactly* what is typed in the Collection.



* + During RUNTIME only, you can also use add and delete items from a ListBox using Items.Add, Items.Remove and Items.Clear.

	Me.lstFlavors.Items.Add(“Cherry”)
	‘Adds Cherry to the lstFlavors collection

	Me.lstFlavors.Items.Remove(“Vanilla”)
	‘Removes Vanilla from the lstFlavors collection

	Me.lstFlavors.Items.Clear()
	‘Clears the entire list.

**The ComboBox Control**

* The ComboBox displays a list in a drop-down box.
* It is actually a combination of a textbox and a listbox.



**Textbox**

**Listbox**

* The top of the ComboBox allows the user to type input.
* The second part of the ComboBox displays a list of items from which the user can select.
* The ComboBox Control Properties
	+ (Name)
		- The name of a ComboBox should start with the prefix cbo
	+ Items
		- Click in this property where it displays (Collection). A new dialog box called the String Collection Editor will appear. Type your items here.
		- Be careful with your spelling.
		- Make sure you hit the Enter
		key after the last item.
	+ Text
		- The text property returns the actual string value of the item.
	+ Enabled
		- The enabled property is set to true by default. If this property is set to false, the list box is not available for the user.
	+ Sorted
		- The sorted property is set to false by default. If this property is set to true, your collection of items will be sorted ascending.
	+ Visible
		- The visible property is set to true by default. If this property is set to false, the list box is not visible to the user.
* Using The ComboBox Control – Visual Studios 2010
	+ Whether a value was keyed or an item selected, the **text** property will hold the value.
	+ Example:

 MessageBox.Show(Me.cboComboBox.Text)

 ’value typed or item selected is shown.

* Using The ComboBox Control – Older versions than 2010 (2007, 2005)
	+ Prior to Visual Studios 2010, an If…Then…Else statement was needed to determine if a value was entered or an item selected. 2010 users can use the text property as shown in the previous example.
	+ If needed, the following If…Then…Else statement can be used to determine whether a value was entered or an item selected.
	+ Example:

If Me.cboComboBox.SelectedIndex >= 0 Then

 MessageBox.Show(Me.cboComboBox.SelectedItem) ’item selected

Else

 MessageBox.Show(Me.cboComboBox.Text) ’value typed

End If

**ListBox & ComboBox SelectedIndexChanged**

* + Just like you create a TextChanged event for a TextBox, you can create a SelectedIndexChanged event for a ListBox or ComboBox.
	+ You can use this event to clear the answer from a label.

	Private Sub lstChoice\_SelectedIndexChanged(ByVal sender As Object, ByVal e As System.EventArgs) Handles lstChoice.SelectedIndexChanged

 Me.lblResult.Text = Nothing ’Clears Label

 End Sub

**When to use a ComboBox vs. a ListBox**

* Use a ComboBox when there is a list of *suggested* choices.
* A ListBox gives the user a list of choices whereas a ComboBox allows the user to enter input (the TextBox portion) in addition to providing options.

**Conclusion**

* + This PowerPoint provided an overview of adding a ListBox and a ComboBox in Visual Studio.
	+ For more information on this topic
		- http://msdn.microsoft.com/en-us/library/2chz8edb(v=VS.80).aspx