CSC317 Visual Programming: Day 03

- Attendance/download Day03 from D2L
- PowerPoint with Illustrations:
  - New object
    - ListBox
  - REPETITION CONTROL STRUCTURE
    - FOR Loop
    - Nested FOR loops
- Demo Problem: Squares and Multiply Tables
- Practice Problem: Prime Numbers

CSC317 Visual Programming: Day 03
Demo Project: Squares,Multiply Tables

- **Timers accomplish REPETITION by means of their Tick method – which repeatedly executes code at pre-programmed intervals**

- **FOR loops accomplish REPETITION nearly instantaneously by putting code to be repeated in the so-called BODY of the loop**

- **Results of the FOR loop need to be displayed in a lasting location such as a ListBox**

CSC317 Visual Programming: Day 03
Demo Project: Squares,Multiply Tables

The format for a FOR loop is as follows:

```
For variable_name = start_value To end_value [Step value]
        'BODY of loop goes here
Next
```

The words For, To, Step, and Next are keywords and display in blue in Visual Basic. Every statement in the body of the loop is executed for each value of the variable from the start_value to the end_value, spaced by the Step value (which is optional and is by default 1 if omitted).

CSC317 Visual Programming: Day 03
Demo Project: Squares,Multiply Tables

Here are some examples of FOR loops

```
For number = 1 To 100
        'BODY of loop executes 100 times
Next

For x = 2 To 100 Step 2
        'BODY of loop: x is all even numbers 2,4,6,…,100
Next

For y = 50 To 1 Step -1
        'BODY of loop: y goes "backwards": 50,49,48,…,1
Next
```

CSC317 Visual Programming: Day 03
Demo Project: Squares,Multiply Tables

A ListBox is an object shaped like a box. If invisible, it can be used to store lists of items, frequently created by loops. If visible, it can be used to display items. Here are its most common methods:

- ListBox_name.Click selects an item and location
  - ListBox_name.SelectedItem
  - ListBox_name.SelectedIndex (index starts at 0)
- ListBox_name.Items.Add(item) – one row at a time
- ListBox_name.Items.Clear() – clears all items

- If its Sorted property is true, it maintains a sorted list.

CSC317 Visual Programming: Day 03
Demo Project: Squares,Multiply Tables

Let us look at the demo project, which contains two ListBoxes. The BackColor of the form is Wheat, and there are labels above each ListBox. On the left, we display the squares of the first 25 positive integers. On the right, we display the products of all integers up to and including 12.

If you click on a row in either table, it displays a magnified version of it between the ListBoxes. This disappears when the mouse leaves the ListBox. A label is used in the middle and its visibility is turned on and off to accomplish this.

From now on, we will not list the specific properties of the objects in the design window.

CSC317 Visual Programming: Day 03
Demo Project: Squares,Multiply Tables
CODE

```
Private Sub Form1_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
        Dim number As Integer
        'Fill the table
        For number = 1 To 25 'loop: execute the next line for each of these values of number
            lstSquares.Items.Add(number & " X " & number & " = " & number * number)
        Next

        'NOTE: loop is nearly instantaneous, so results need to be displayed in a lasting location
        'such as a ListBox.  Contrast this with the Tick method for a timer, where there is a
        'perceptible delay, set by the programmer

        Dim i, j As Integer 'use for multiplying i times j
        'use nested loop - one inside another
        'Fill the table
        For i = 1 To 12
            For j = 1 To 12 'for each i, let j vary through its values
                lstMultiply.Items.Add(i & " X " & j & " = " & i * j)
            Next
        Next
        'NOTE: the Add procedure is executed 12 x 12 = 144 times, giving 144 rows in the ListBox.
        'A ScrollBar is automatically created.

    End Sub
```

CSC317 Visual Programming: Day 03
Demo Project: Squares,Multiply Tables
CODE

What about the special effect of magnifying the row that is clicked and displaying in the middle, as well as having this disappear when the mouse leaves the ListBox? Here is how it is done:

```
    Private Sub lstSquares_Click(ByVal sender As Object, ByVal e
As System.EventArgs) Handles lstSquares.Click, lstMultiply.Click
        lblDisplay.Visible = True
        lblDisplay.Text = sender.SelectedItem
    End Sub

    Private Sub lstSquares_MouseLeave(ByVal sender As Object,
ByVal e As System.EventArgs) Handles lstSquares.MouseLeave,
lstMultiply.MouseLeave
        lblDisplay.Visible = False

    End Sub
```

CSC317 Visual Programming: Day 03
Demo Project: Squares,Multiply Tables
CODE

In summary, when you pull down an object and a method in the code window, the header line ends in

```
Handles Object_name.Method_name
```

If you want the same code to apply to one or more methods of other objects, you append the header line with commas and the other objects and methods, connected by dots.

You should go back one slide to review this. Also note that there are two parameters in the header line: sender and e. Sender is the object actually invoking the method and e is an event (not used here). So

```
sender.SelectedItem
```

means either lstSquares.SelectedItem or lstMultiply.SelectedItem, depending on which object is being handled.

CSC317 Visual Programming: Day 03
Practice Project: Prime Numbers

- **You will name this project PrimeNumbers and save it in your Day03 folder.**

- **It should run like the demo project PrimeNumbers.exe already in your Day03 folder.**

A prime number is an integer > 1 that is divisible only by itself and 1. Use a labeled ListBox called lstPrime, which will contain, upon loading the form, all prime numbers between 2 and 2500. On the next screens are some explanations about the code, followed by the code itself, which you should enter and test.

CSC317 Visual Programming: Day 03
Practice Project: Prime Numbers

Your load method should have NESTED loops. That is, you need an "outer" loop with a number that goes from 1 to 2500. Within that, you only need to look for divisors up to the square root of the number (if there are two factors, one is always less than this and one is always greater than this). So you need an inner loop that goes from 2 up to

System.Math.Sqrt(number)

where the square root function Sqrt is kept in the System.Math class for your use.

CSC317 Visual Programming: Day 03
Practice Project: Prime Numbers

A candidate for a divisor of a number will actually be a divisor if

number MOD divisor= 0

because the MOD operator does an integer division of the first operand by the second one and gives the integer remainder.

It is useful to declare a Boolean variable

Dim **prime** as Boolean

which can have two values: True and False.

CSC317 Visual Programming: Day 03
Practice Project: Prime Numbers

One final comment: within your "inner" loop, if you find a divisor it makes no sense to look for more divisors, and you can exit a FOR loop "early" by means of the

Exit For

statement. Now enter and test the code on the following slide!

CSC317 Visual Programming: Day 03
Practice Project: Prime Numbers

```
    Private Sub Form1_Load(ByVal sender As Object, ByVal e As System.EventArgs)
Handles Me.Load
        Dim number As Integer 'it might be prime
        Dim divisor As Integer 'it might be a factor of number - use MOD to find
out
        Dim prime As Boolean 'start as True but make False if there are divisors

        'NOTE: only need test divisors up to the square root of the number

        For number = 2 To 2500 'check all numbers up to 1000
            prime = True 'hope that number is prime
            For divisor = 2 To System.Math.Sqrt(number) 'square root function is
in System.Math class
                If number Mod divisor = 0 Then
                    prime = False
                    Exit For
                End If
            Next
            If prime Then
                'no divisors found - prime is still True
                lstPrime.Items.Add(number)
            End If
        Next
    End Sub
```