Overview (1):
- Before We Begin
  - Some administrative details
  - Some questions to consider
- Counted Loops
  - Introduction
  - Experimenting with counted loops
- ListBox Control
  - Introduction

Administrative Details (1):
- Lab Exercises
  - You should be working on Ex 5-3 this week
    - Due February 28
  - Still have a few exercises and tests that were previously distributed but have not been picked up yet
    - If you have not picked up any exercise or test yet, you can after the lecture

Before We Begin

Some Questions to Consider (1):
- What is a loop?
- Why are loops important in any programming language?
- What is a conditional loop?
- How many forms of conditional loops are available?
- Describe each form of the conditional loop?
- What must we, as programmers ensure for every loop?

Counted Loops
Introduction to Computer Use II

Introduction (1):
- Recall Conditional Loops
  - Basically, we iterate the loop statements as long as the loop condition holds
  - Useful when we do not know how many times the loop will execute
  - May execute any number of times before condition is not met
  - Many times we do know exactly how many times the loop should execute.
  - In such a situation, we can use a counted loop instead.

Introduction (2):
- What Exactly is a Counted Loop?
  - A loop that is executed a specific number of times
  - We of course need to know how many times the loop will iterate before we start the loop
  - Central to the counted loop is the counter variable known as the loop index that keeps track of how many times the loop has iterated
  - Value of the loop index is tested after each iteration to determine whether or not to exit the loop if less than the number of total loop iterations then continue with next iteration.

Counted Loops (1):
- What Exactly is a Counted Loop? (cont.):
  - Three components
    1. Initialize the counter
    2. Increment the counter
    3. Test the counter to determine when it is time to terminate the loop

Counted Loops (2):
- Counted Loops and Visual Basic
  - General form (syntax) of the counted loop
    ```
    For loopIndex = initialValue To endValue [Step Increment]
    statements of loop body
    Next
    ```
  - loopIndex
    - Loop counter must be a numeric value
  - initialValue
    - Initial value of loop index may be a constant, variable, numeric value or numeric expression

Counted Loops (3):
- Counted Loops and Visual Basic (cont.)
  - General form (syntax) of the counted loop
    ```
    For loopIndex = initialValue To endValue [Step Increment]
    statements of loop body
    Next
    ```
  - endValue
    - Loop terminates when index = endValue may be a constant, variable, numeric value or numeric expression

Counted Loops (4):
- Counted Loops and Visual Basic (cont.)
  - General form (syntax) of the counted loop
    ```
    For loopIndex = initialValue To endValue [Step Increment]
    statements of loop body
    Next
    ```
  - Step Increment
    - Step is a keyword and increment is amount to increase index after each iteration optional and if not present default = 1
  - Next is a keyword
**Counted Loops (5):**
- Counted Loops and Visual Basic (cont.)
  - Outline of the counted loop operation
    - Prior to starting loop, index set to "initialValue"
    - Final value for the loop index is set to the value of endValue
    - After index is initialized, it is tested to see if it is greater than endValue → if not, loop statements executed, otherwise loop terminates
    - Next statement causes index to be incremented by "Increment" or 1 if no increment is specified
    - Value of index is then compared again to endValue

**Counted Loop Examples (1):**
- Some Examples
  - Let's examine some different “For” statements
  - Make sure you understand each of the following
    - For index = 2 To 100 Step 2
    - For countValue = startValue To EndValue Step IncrementValue
    - For countValue = 0 To coefficientType.ListCount - 1
    - For index = (someValue - 5) To totalPossible
    - For curRate = 0.5 To 0.25 Step 0.05
    - For negativeCounter = 10 To Step -1

**Counted Loop Examples (2):**
- Further Examples
  - Some complete counted loop examples

**Counted Loop Specifics (1):**
- Negative Increment (Counting Backwards)
  - As shown in the previous examples, we can count backwards with a counted loop
    - Use a negative number for the increment and explicitly specify it with the “Step”
    - When the Step is negative, VB tests for less than as opposed to greater than
    - For index = 10 To 0 Step -1

**Counted Loop Specifics (2):**
- Conditions Satisfied Before Loop Entry
  - At times, final value will be reached before entry into the loop
    - Statements in the loop body will not be executed at all in such a case

**Counted Loop Specifics (3):**
- Altering the Value of Loop Control Variables
  - Once we enter the body of the loop, initialValue, endValue and increment have already been set
    - But we can alter these values within the loop body → this will have no effect on the loop (the number of times the loop iterates will not change!)
Counted Loop Specifics (4):
- Endless Loops
  - Although changing the initial, end and increment values doesn't affect the loop, changing the loop index can have an affect on the loop
  - Can have a loop that never ends!

Counted Loop Specifics (5):
- Exiting For / Next Loops
  - Usually, a "For" loop should execute until it completes (e.g., until index reaches the final value)
  - There may be times however where we want to exit before the index reaches the final value
  - Visual Basic provides the "Exit For" statement to exit a "For" loop early
  - Typically, the End For will be part of an If statement → will allow us to exit the loop given a particular condition

Counted Loop Specifics (6):
- Exiting For / Next Loops (cont.)
  - Example
    - Program that continually (in a loop) takes in user input and performs some operation on it and if input is the string "Exit" then exit program

LibBox Control

Introduction (1):
- As an Aside
  - Recall that an object contains properties that can be accessed, modified etc.
  - An object can also have methods associated with it
    - A method is a sub-program (think of the event handlers we know) that can take zero or more arguments and returns one value
    - Since a method is associated with (belongs to) an object, it is accessed in the same manner as an object's properties → using the "dot" notation `objectName.methodName`

Introduction (2):
- What is a ListBox Control?
Introduction to Computer Use II

**Introduction (3):**
- **What is a ListBox Control? (cont.)**
  - An object containing a list of output
  - If the data displayed in the ListBox exceeds its height, a scroll bar appears
  - Displays on each row a string value, generically called an item
  - The item must be displayed on the ListBox using the `AddItem` method of the ListBox

```
listBoxName.AddItem(stringExpression)
```

**Introduction (4):**
- **What is a ListBox Control? (cont.)**
  - Example → displaying a row in a listBox called List1

```
Private Sub Form_Load()
    Dim testString As String
    testString = "This is a test of the ListBox control"
    List1.AddItem (testString)
End Sub
```

After executing the above code segment, the following is observed in the ListBox control placed on the form.

**Introduction (5):**
- **What is a ListBox Control? (cont.)**
  - When we add information to the ListBox (via the "addItem" method), the new information is appended to the next line
  - But what if we don’t want to append and wish to start “clean” → there is a method to clear the ListBox of any information it may currently hold thus allowing you to “start fresh”
  - The method to clear the ListBox is "Clear()" and takes no arguments → ListBox.Clear()

**Live Demos (1):**
- "Live" Examples of Counted Loops and ListBoxes
  - Lets look at some simple examples of working with counted loops and ListBox controls in Visual Basic