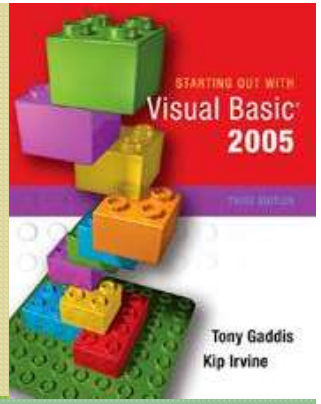
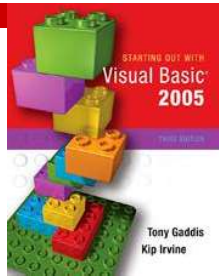


9.1 Using Files

A File Is a Collection of Data Stored on a Computer Disk

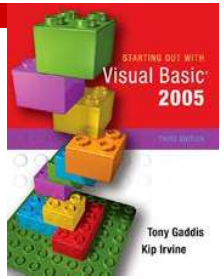
Information Can Be Saved to Files and Later Reused

The Life Span of Data



- Thus far, all of our data has been stored in controls and variables existing in RAM
- This data disappears once the program stops running
- If data is stored in a file on a computer disk, it can be retrieved and used at a later time

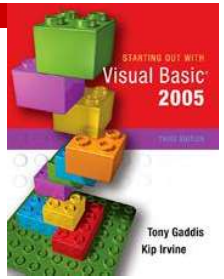
Three Steps in Using a File



1. The file must be opened

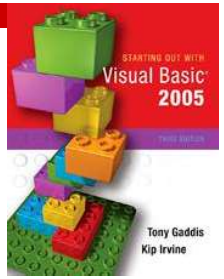
   If it does not yet exist, it will be created
2. Data is read from or written to the file
3. The program closes the file

Reading and Writing to a File



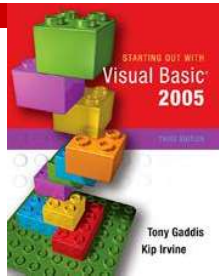
- Data must be in retrieved from disk and put in memory for an application to work with it
- Data is transferred from disk to memory by:
  - *Reading* it from an *input file*
  - Placing it in variables or control properties
- Data is transferred from memory to disk by:
  - *Writing* it to an *output file*
  - Getting it from variables or control properties
- Data is frequently placed in the text property of a control

File Types/Access Methods



- *Text file* type
  - Character based text
  - Contents can be viewed by Notepad
- *Binary file* type
  - Pure binary form
  - Contents cannot be viewed with a text editor
- Access Methods
  - *Sequential access* – a continuous stream of data written and read as a whole from beginning to end
  - *Random access* – access in any order with data written to or read from specific places in the file
  - Like the difference between a casette tape and a CD

Establishing StreamWriter Objects



- A *StreamWriter* object is used to write to a sequential text file in the following way:
  - Declare a variable of type StreamWriter

```
Dim phoneFile As System.IO.StreamWriter
```

  - Create a StreamWriter object and assign it to the StreamWriter variable using either the
  - CreateText method for new files

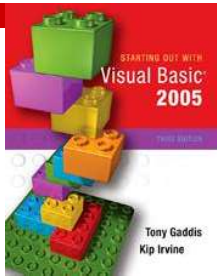
```
phoneFile = System.IO.File.CreateText("phonelist.txt")
```

  - AppendText method for existing files

```
phoneFile = System.IO.File.AppendText("phonelist.txt")
```

- Variable **`phoneFile`** now defines a stream of data that can be written to **`phonelist.txt`**

Using a String Variable as File Name

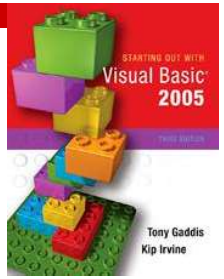


- Filename can be a string literal as already shown but a string variable is more flexible
  - User can select the file they wish to edit
  - What if Notepad could only edit “textfile.txt”?
- Example with string variable as filename

```
Dim custFile As System.IO.StreamWriter
Dim fileName as String
fileName = "customer.txt"
custFile = System.IO.file.AppendText(fileName)
```

- Can allow the user to enter the filename
  - Substitute `txtFile.text` for `"customer.txt"`
  - User can then enter filename in a text box

File Paths



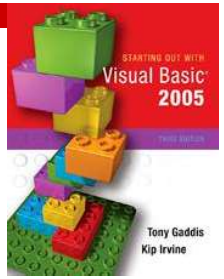
- Filename can include the file path
  - Can be a complete file path with drive letter

    `"C:\WordProc\memo.txt"`
  - Refer to a file in the default drive root directory

    `"\pricelist.txt"`
  - Or include no path information at all

    `"mytext.txt"`
- If no path information specified, the *bin* folder of the current project is used

Writing Data to a File



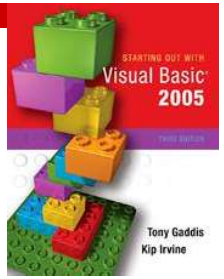
- The *WriteLine* method of a *StreamWriter* object actually writes data to the file

```
ObjectVar.WriteLine(Data)
```

  - Streamwriter object identified by `ObjectVar`
  - The method's `Data` argument consists of constants or variables with data to be written
- WriteLine appends an invisible *newline character* to the end of the data
- Omit argument to write a blank line to a file

```
ObjectVar.WriteLine()
```

Closing a StreamWriter Object

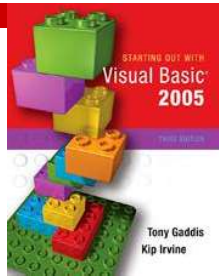


- Should close files when finished with them
  - Avoids losing data
  - Data is initially written to a buffer
  - Close writes unsaved data from the buffer to the file
- The *Close* method of a *StreamWriter* object clears the buffer and closes the file

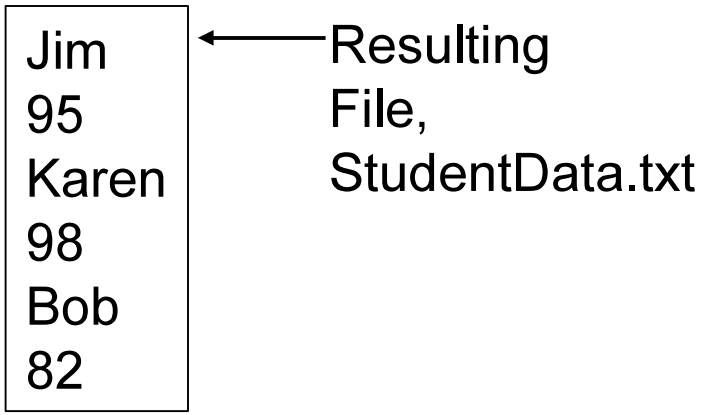
```
ObjectVar.Close()
```

  - Streamwriter object identified by `ObjectVar`

Writing Data to a File Example

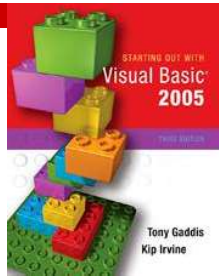



```
Dim studentFile As System.IO.StreamWriter
studentFile = System.IO.File.CreateText("StudentData.txt")
studentFile.WriteLine("Jim")
studentFile.WriteLine(95)
studentFile.WriteLine("Karen")
studentFile.WriteLine(98)
studentFile.WriteLine("Bob")
studentFile.WriteLine(82)
studentFile.Close()
```



- Tutorial 9-1 is an example of an application that writes data to a file

Importing a Namespace



- System.IO is referred to as a *namespace*
  - A group of logically related classes
  - *System.IO* contains StreamWriter and other file related classes
- Can shorten references to such classes by *importing* the namespace in your code

```
Imports System.IO
```

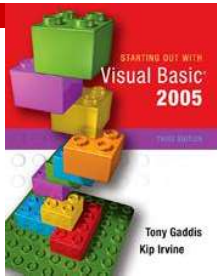
  - Allows us to use

```
Dim custFile As StreamWriter
```

  - Instead of

```
Dim custFile As System.IO.StreamWriter
```

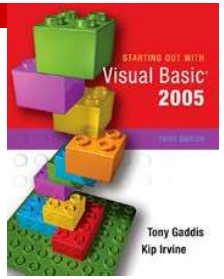
Appending to a File



- If opening an existing file with *CreateText*
  - Existing contents are removed
  - New text overwrites the old text
- If opening an existing file with *AppendText*
  - Existing contents are retained
  - New text adds on to the end of the old text
- If adding a new friend to *friendFile*, you'd use

```
friendFile = System.IO.File.AppendText("MyFriends.txt")
```

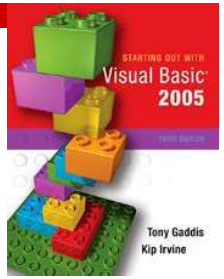
The StreamWriter Write Method




```
ObjectVar.Write(Data)
```

- The *Write* method does not place a newline character after each data item
- Usually need to provide some sort of delineation or *delimiter* between data items
  - A blank space could be used
  - Comma is a more common delimiter

Write Method Example



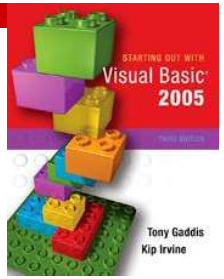

```
Dim name As String = "Jeffrey Smith"
Dim idNum As Integer = 47895
Dim phone As String = "555-7864"

outputFile.Write(name)
outputFile.Write(" ")
outputFile.Write(idNum)
outputFile.Write(" ")
outputFile.WriteLine(phone)
```

The Resulting File

Jeffrey Smith 47895 555-7864

StreamReader Objects



- Use *StreamReader* objects to read from a file
- Define and open similar to StreamWriter:

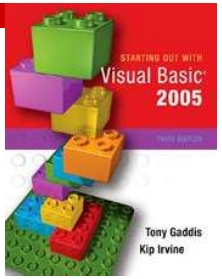
```
Dim ObjectVar As System.IO.StreamReader
ObjectVar = System.IO.File.OpenText(Filename)
```

- Sample code:

```
Dim phoneFile As System.IO.StreamReader
phoneFile = System.IO.File.OpenText("phonelist.txt")
```

- Variable **`phoneFile`** now defines a stream of data that can be read from **`phonelist.txt`**

Reading Data from a File



- The *ReadLine* method of a *StreamReader* object actually reads data from the file

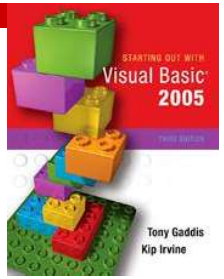
```
dataVar = ObjectVar.ReadLine()
```

  - Streamwriter object identified by `ObjectVar`
  - The result of the method, the data read from the file, is assigned to string variable `dataVar`
- Sample code:

```
Dim custFile As System.IO.StreamReader
custFile = System.IO.File.OpenText("customer.txt")
custName = custFile.ReadLine()
```

- custName holds the data read from the file
- StreamReader also has a *Close* method

Determining Whether a File Exists

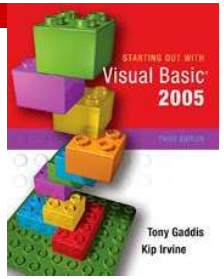


- The *File.OpenText* method issues a runtime error if the file does not exist
- Avoid this by using the *File.Exists* method
  - Format is **`File.Exists(filename)`**
  - Returns a boolean result that can be tested:

```
If System.IO.File.Exists(filename) Then
   ' Open the file.
   inputFile = System.IO.File.OpenText(filename)
Else
   MessageBox.Show(filename & " does not exist.")
End If
```

- Tutorial 9-2 shows how to read text file data

Detecting the End of a File

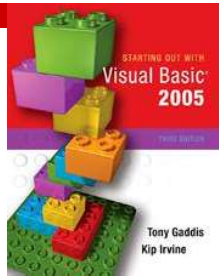


- The *Peek* method tests if you've reached end of file (no more characters to read)
  - Format is **`objectvar.Peek`**
  - If no more characters, the value -1 is returned

```
Imports System.IO
Dim scoresFile As StreamReader
Dim input As String
scoresFile = File.OpenText("Scores.txt")
Do Until scoresFile.Peek = -1
      input = scoresFile.ReadLine()
      lstResults.Items.Add(input)
Loop
scoresFile.Close()
```

- Tutorial 9-3 demonstrates the *Peek* method

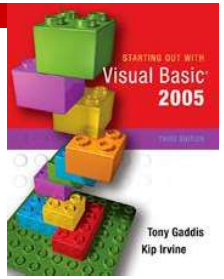
Read Method



- *Read* method returns the integer code of the next character in the file
  - *Chr* function converts integer code to character
- This loop appends one character at a time to input until no more characters are in the file

```
Imports System.IO
Dim textFile As StreamReader
Dim input As String
textFile = File.OpenText("names.txt")
Do While textFile.Peek <> -1
      input &= Chr(textFile.Read)
Loop
textFile.Close()
```

ReadToEnd Method



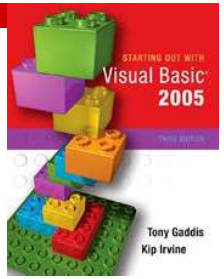
- *ReadToEnd* method returns the rest of the file from the current read position to end of file
- Functions differently from ReadLine method
  - ReadToEnd method ignores line delimiters
- The statement

  `input = textFile.ReadToEnd`

  reads the file contents and stores it in **`input`**

```
Imports System.IO
Dim textFile As StreamReader
Dim input As String
textFile = File.OpenText("names.txt")
input = textFile.ReadToEnd
textFile.Close()
```

Write Then Read an Entire Array




```
Imports System.IO
Dim intValues(9)
-------------------------------------------
Dim outputFile as StreamWriter
outputFile = File.CreateText("values.txt")
For count = 0 To (intValues.Length - 1)
      outputFile.WriteLine(intValues(count))
Next count
outputFile.Close()
-------------------------------------------
Dim inputFile as StreamReader
inputFile = File.OpenText("values.txt")
For count = 0 To (intValues.Length - 1)
      intValues(count) = Val(inputFile.ReadLine)
Next count
inputFile.Close()
```