CHAPTER 8: DATA FILES

Save data to sequential text files

Read data from the files back into the application

FILE HANDLING

- To process data more efficiently, many developers use text files to store and access information to use within an application
- The simplest type of data file is called a **sequential-access file**
- A sequential-access file is like a stream of data that must be read from beginning to end
- Sometimes referred to as a text file
- Text files have a **.txt** extension
- Can easily be created and modified using a text editor
  - Windows Notepad, for example

THE PROCESS OF USING A FILE

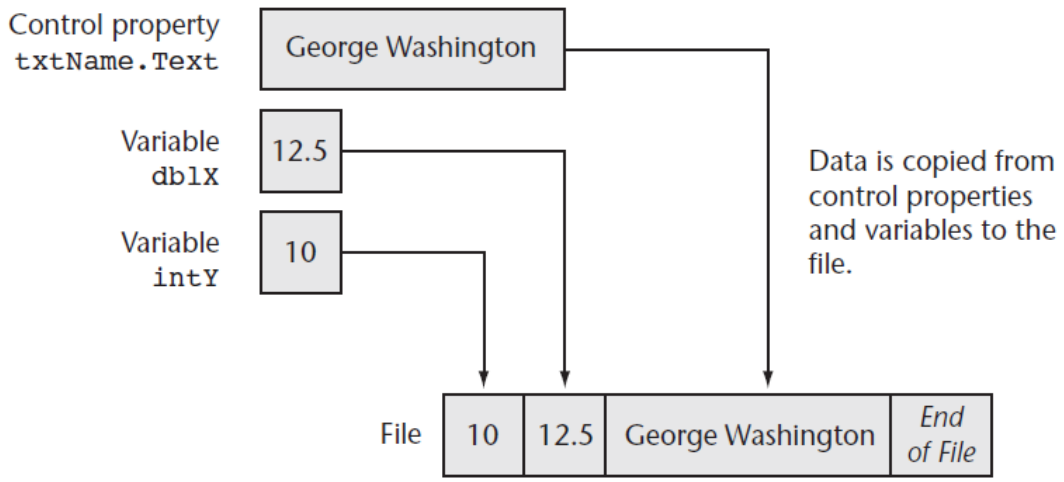
The file must be opened; If it does not yet exist, it must be created

Data is written to the file or read from the file

When the application is finished using the file, the file is closed

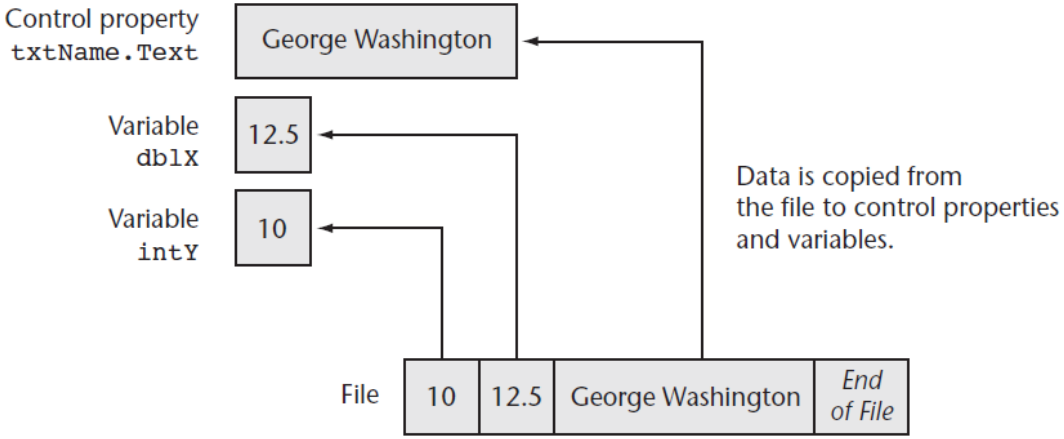
OUTPUT FILE

- An *output file* is a file into which a program writes data



INPUT FILE

- An *input file* is a file from which a program reads data



WRITING TO FILES WITH `STREAMWRITER` OBJECTS

- Two basic ways to open a file for writing
  - Create a new file
  - Open an existing file and append data to it
- A `StreamWriter` object performs the actual writing to the file
- Two required steps:
  1. Declare a `StreamWriter` variable
  2. Call either `File.CreateText` or `File.AppendText` and assign its return value to the `StreamWriter` variable

USING THE `IMPORTS` STATEMENT FOR THE `STREAMWRITER` OBJECTS

- To make the `StreamWriter` objects available to your program
  - Insert the following `Imports` statement at the top of your form's code file:

```
Imports System.IO
```

NOTE: It is possible to omit the `Imports System.IO` statement, but then every reference to the `StreamWriter` class must use its fully qualified name, which is `System.IO.StreamWriter`.

CREATING A TEXT FILE (1 OF 2)

- Declare a `StreamWriter` variable using the following general format:

```
Dim streamWriterVariable As IO.StreamWriter
```

  - *`streamWriterVariable`* is the name of the object variable
  - You may use `Private` or `Public` in place of `Dim` at the class-level or module-level
  - Here's an example:

```
Dim outFile As IO.StreamWriter
```

CREATING A TEXT FILE (2 OF 2)

- Use `CreateText` method to open the file for output
- Call the `IO.File.CreateText` method, passing the name of a file. For example:

```
outFile = IO.File.CreateText("employee.txt")
```

  - The computer will search for the `employee.txt` file in the current project's `\bin\Debug` folder
  - The return value from `IO.File.CreateText` is assigned to the `StreamWriter` variable named `outFile`

OPENING AN EXISTING FILE AND APPENDING DATA TO IT (1 OF 2)

- Use the `AppendText` method to open the existing text file for append
- First, declare a `StreamWriter` variable
- Call the `IO.File.AppendText` method, passing the name of an existing file. For example:

```
outFile = IO.File.AppendText("employee.txt")
```

  - If the file does not exit it will be created

OPENING AN EXISTING FILE AND APPENDING DATA TO IT (2 OF 2)

- The following example:

  Opens a file in append mode and writes additional data to the file

Before

Jim Weaver
555-1212
Mary Duncan
555-2323
Karen Warren
555-3434

```
' Declare an object variable
Dim friendFile As StreamWriter

' Open the file.
friendFile =
IO.File.AppendText("MyFriends.txt")

' Write the data.
friendFile.WriteLine("Bill Johnson")
friendFile.WriteLine("555-4545")

' Close the file.
friendFile.Close()
```

After

Jim Weaver
555-1212
Mary Duncan
555-2323
Karen Warren
555-3434
Bill Johnson
555-4545

WRITING DATA TO A FILE (1 OF 3)

- Use either the `Write` method or `WriteLine` method to write data to the file
  - `Write` method general format:

```
streamWriterVariable.Write(Data)
```

    - Writes an item of data without writing a newline character. For example:

```
outFile.Write("Hello")
```

Result

Hello|

the next character will be written immediately after the o

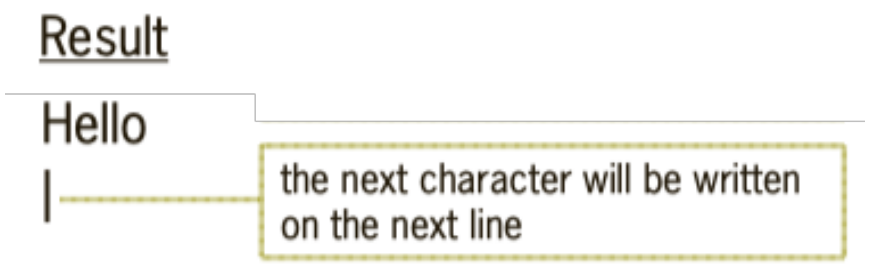
WRITING DATA TO A FILE (2 OF 3)

- `WriteLine` method general format:

```
streamWriterVariable.WriteLine(Data)
```

  - Writes a newline character after the data. For example:

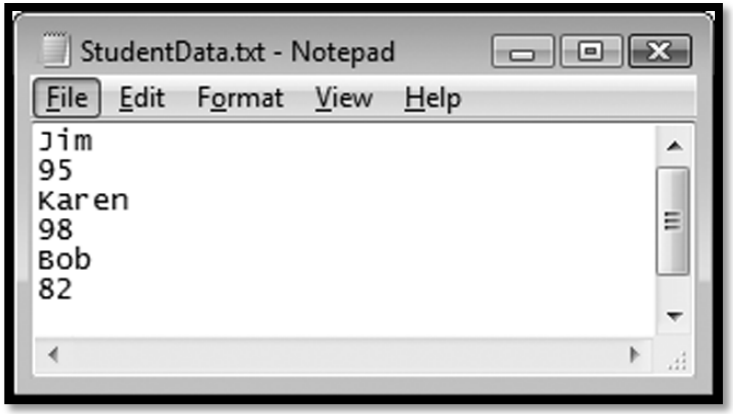
```
outFile.WriteLine("Hello")
```



WRITING DATA TO A FILE (3 OF 3)

- The following writes three students' first names and scores to a file:

```
' Write data to the file.
studentFile.WriteLine("Jim")
studentFile.WriteLine(95)
studentFile.WriteLine("Karen")
studentFile.WriteLine(98)
studentFile.WriteLine("Bob")
studentFile.WriteLine(82)
```



Jim<*newline*>**95**<*newline*>**Karen**<*newline*>**98**<*newline*>**Bob**<*newline*>**82**<*newline*>

- In addition to separating the contents of a file into lines, the newline character also serves as a delimiter
  - A *delimiter* is an item that separates other items
  - Data must be separated in order for it to be read from a file

CLOSING A FILE

- Use the `close` method to close an output file as soon as you finished using it.
- General format:

```
streamWriterVariable.Close()
```

  - For example:

```
outFile.Close()
```

- This ensures that the data is saved, and it makes the file available for use elsewhere in the application

READING FILES WITH `STREAMREADER` OBJECTS (1 OF 2)

- A `StreamReader` object reads data from a sequential text file
- Create a `StreamReader` object variable using the following general format:

```
Dim streamReaderVariable As IO.StreamReader
```

  - For example:

```
Dim inFile As IO.StreamReader
```

READING FILES WITH `STREAMREADER` OBJECTS (2 OF 2)

- The `File.OpenText` method opens a file and stores the address of the `StreamReader` object variable using the following general format:

```
IO.File.OpenText(Filename)
```

  - For example:

```
inFile = IO.File.OpenText("employee.txt")
```

READING DATA FROM A FILE (1 OF 2)

- The `ReadLine` method in the `StreamReader` class reads a line of data from a file using the following general format:

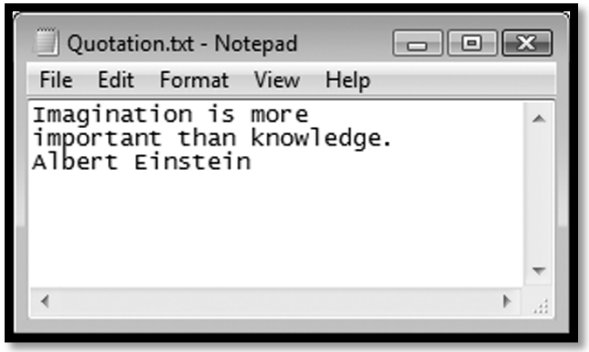
```
streamReaderVariable.ReadLine()
```

- The method reads a line from the file associated with *`streamReaderVariable`* and returns the data as a string
- For example, the following statement reads a line from the file and stores it in the variable:

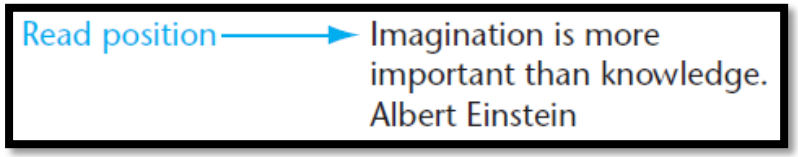
```
strEmployeeName = inFile.ReadLine()
```

READING DATA FROM A FILE (2 OF 2)

```
Dim textFile As StreamReader
textFile =
IO.File.OpenText("Quotation.txt")
```



- Data is read from a file in a forward-only direction
- When the file is opened:
  - Its *read position* is set to the first item in the file
- As data is read:
  - The read position advances through the file




```
strInput = textFile.ReadLine()
```

DETERMINING WHETHER A FILE EXISTS

- To determine if a file exists before opening it, you can call the `File.Exists` method using the following general format:

```
IO.File.Exists(Filename)
```

- The method returns *True* if the files exists or *False* if the file does not exist

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

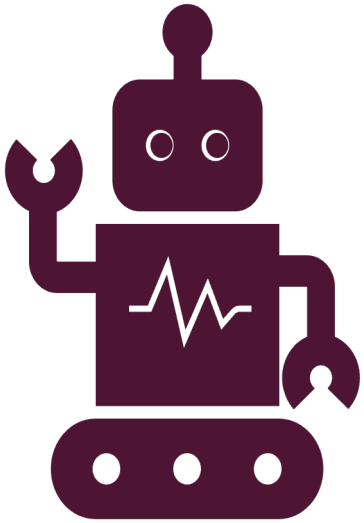
CLOSING THE FILE

- The `StreamReader` class has a method named `Close` that closes an open `StreamReader` object using the following general format:

```
streamReaderVariable.Close()
```

- The following statement closes a `StreamReader` object variable named `readFile`:

```
inFile.Close()
```



END OF TOPIC … DATA FILES