File Stream

Using File for Data Storage

- When a program needs to save data for later use, it writes the data in a file
- There are always three steps:
  - Open the file: create a connection between the file and the program
  - Process the file: either write to or read from the file
  - Close the file: disconnect the file and the program
- In general, there are two types of files:
  - Text file: contains data that has been encoded as test using scheme such as Unicode
  - Binary file: contains data that has not been converted to text. You cannot view the contents of binary files with a text editor.
- We work with text files for now

File Accessing

- A file object is an object that is associated with a specific file and provides a way for the program to work with that file

- The .NET Framework provide two classes to create file objects through the **System.IO** namespace
  - **StreamWriter**: for writing data to a text file
  - **StreamReader**: for reading data from a text file

- You need to write the following directives at the top of your program

```
Using System.IO;
```

Writing Data to a File

- Start with creating a StreamWriter object

```
StreamWriter outputFile;
```

- Use one of the File methods to open the file to which you will be writing data. Sample File methods are:
  - File.CreateText
  - File.AppendText

- Use the **Write** or **WriteLine** method to write items of data to the file
- Close the connection.

Sample Code

```
StreamWriter outputFile;
outputFile = File.CreateText(“courses.txt”);
outputFile.WriteLine(“Introduction to Computer Science”);
outputFile.WriteLine(“English Composition”);
outputFile.Write(“Calculus I”);
outputFile.Close();
```

- The **WriteLine** method writes an item of data to a file and then writes a newline characters which specifies the end of a line
- The **Write** method writes an item to a file without a newline character

CreateText vs. AppendText

- The previous code uses the File.CreateText method for the following reasons:
  - It creates a text file with the name specified by the argument. If the file already exists, its contents are erased
  - It creates a StreamWriter object in memory, associated with the file
  - It returns a reference to the StreamWriter object

- When there is a need not to erase the contents of an existing file, use the AppendText method

```
StreamWriter outputFile;
outputFile = File.AppendText(“Names.txt”);
outputFile.WriteLine(“Lynn”);
outputFile.WriteLine(“Steve”);
outputFile.Close();
```

Specifying the Location of an Output File

- If you want to open a file in a different location, you can specify a path as well as filename in the argument

- Be sure to prefix the string with the @ character

```
    StreamWriter outputFile;
outputFile = File.CreateText(@"C:\Users\chris\Documents\Names.txt");
```

Reading Data from a File

- Start with creating a StreamReader object

```
StreamReader inputFile;
```

- Use the **File.OpenText** method to open the file to which you will be writing data

```
inputFile = File.OpenText("students.txt");
```

- Use the **Read** or **ReadLine** method to write items of data to the file
  - StreamReader.ReadLine: Reads a line of characters from the current stream and returns the data as a string.
  - StreamReader.Read: Reads the next character or next set of characters from the input stream.

- Close the connection

Reading a File with a Loop

- StreamReader objects have a Boolean property named EndOfStream that signals whether or not the end of file has been reached
- You can write a loop to detect the end of the file.

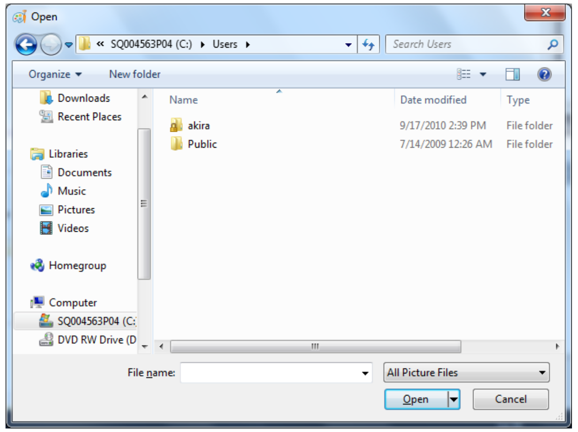
```
while (inputFile.EndOfStream == false) { }
```

- Or

```
while (!inputFile.EndOfStream) { }
```

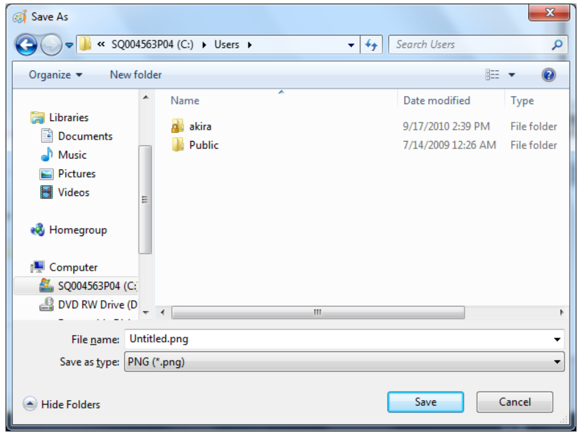
The OpenFileDialog and SaveFileDialog Controls

- using System.Windows.Forms;
- The **OpenFileDialog** and **SaveDialog** controls allow your application to display standard Windows dialog boxes for opening and saving files
- Unlike Label, Button, and TextBox, they are invisible controls
- The OpenFileDialog control displays a standard Windows *Open* dialog box.
- The SaveDialog control displays a standard Windows *Save As* dialog box



Name	Date modified	Type
akira	9/17/2010 2:39 PM	File folder
Public	7/14/2009 12:26 AM	File folder

File name:
All Picture Files



Name	Date modified	Type
akira	9/17/2010 2:39 PM	File folder
Public	7/14/2009 12:26 AM	File folder

File name: Untitled.png
Save as type: PNG (*.png)

Detecting the User’s Selection

- The **showDialog** method returns a value that indicates which button the user clicks to dismiss the dialog box
  - If the user clicked the Open button, the value DialogResult.OK is returned
  - If the user clicked the Cancel button, the value DialogResult.Cancel is returned
  - The following is an example that calls the ShowDialog method to determine the user’s choice:

```
if (openFile.ShowDialog() == DialogResult.OK) { }
else if (openFile.ShowDialog() == DialogResult.Cancel) { }
else { }
```

The Filename and Initial Directory Property

- When the user selects a file with the Open dialog box, the file's path and filename are stored in the control's **Filename** property
- The following is an example of how to open the selected file:

```
if (openFile.ShowDialog() == DialogResult.OK)
{
  inputFile = File.OpenText(openFile.Filename);
}
else { }
```

- You can specify a directory to be initially displayed with the InitialDirectory property. For example,

```
openFile.InitialDirectory = "C:\Data";
```

Displaying a Save As Dialog Box

- Use the following to call the SaveFileDialog control's ShowDialog method

```
saveFile.ShowDialog();
```

- Use the following to detect the user's choice

```
if (saveFile.ShowDialog() == DialogResult.OK) { }
```

- Use the following to open the selected file

```
if (saveFile.ShowDialog() == DialogResult.OK)
{
  outputFile = File.CreateText(openFile.Filename);
}
```