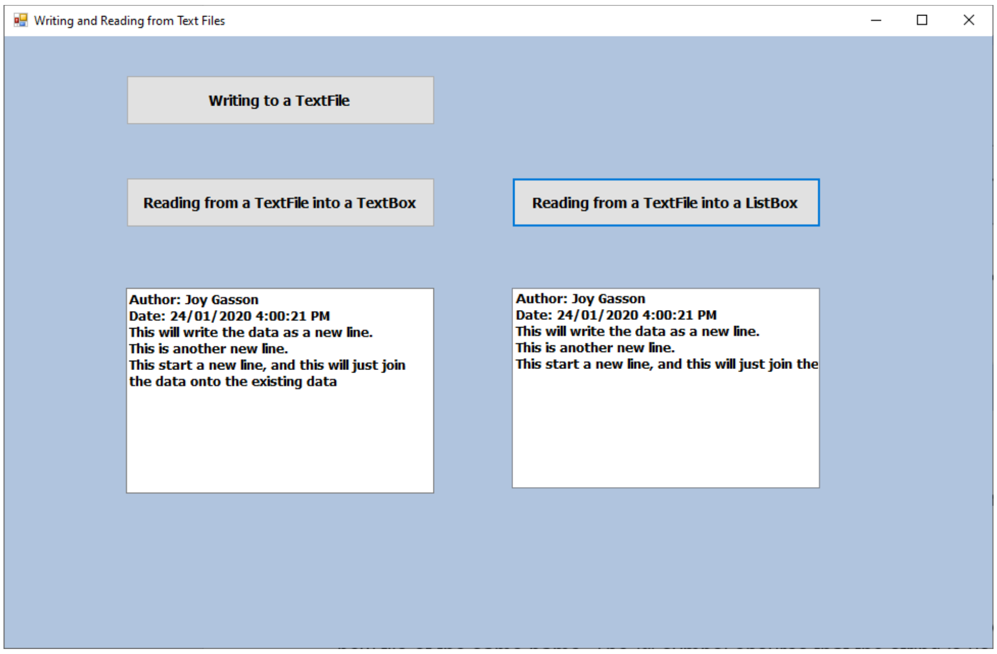
File Handling

1. Writing and Reading .txt Files

So far, our programs have been short lived; we run them then close them down. It is often useful to read and write data to a local text file. C# uses a Stream object as in intermediary (or conceptual buffer) between the file and the form. A stream is linear, it flows in one direction: the data is written from the form to a StreamWriter object, and then from the StreamWriter object to the file. Similarly, the data is written from the file to a StreamReader object, and then from the StreamReader object to the form.


Writing and Reading from Text Files
Writing to a TextFile
Reading from a TextFile into a TextBox
Reading from a TextFile into a ListBox

```
Author: Joy Gasson
Date: 24/01/2020 4:00:21 PM
This will write the data as a new line.
This is another new line.
This start a new line, and this will just join
the data onto the existing data
```

```
Author: Joy Gasson
Date: 24/01/2020 4:00:21 PM
This will write the data as a new line.
This is another new line.
This start a new line, and this will just join the
```

The commands to read and write are similar to those you used with the console.

Add `using System.IO;` to the list of namespaces at the top of your program, to provide file and directory support classes.

To Write to File:

Create a new file called text.txt in the root of the C: drive, or if it already exists, replace it with a new file of the same name. The @ symbol ensures that the string is used literally and that the \ is not interpreted as an escape character.

```
StreamWriter aFile = new StreamWriter(@"test.txt");
```

`Writeline` sends a single line of text to the file and automatically append a carriage return at the end of the line.

```
aFile.WriteLine("Author: Joy Gasson");
aFile.WriteLine("Date: " + DateTime.Now);
aFile.WriteLine("This will write the data as a new line.");
```

`Write` sends data to the file but does not automatically append a carriage return to create a new line.

```
aFile.Write("This start a new line, ");
aFile.Write("and this will just join the data onto the existing data");
```

Every file must be closed, or the file will be locked open and no other program will be able to use it.

```
aFile.Close();
```

To Read from File:

Reading from file is handled by the StreamReader class.

```
StreamReader bFile = new StreamReader(@"test.txt");
```

The StreamReader will throw an error message if the specified file cannot be found, so you need to account for this in your code:

```
textBox1.Text = bFile.ReadToEnd();
```

The `ReadTo End()` will reads the entire file and places the contents of the file into a variable.

Or, you could read one line at a time using the `ReadLine();` method, checking first that you haven't reached the end of file.

```
StreamReader bFile = new StreamReader(@"test.txt");
while (!bFile.EndOfStream)
{
   textBox1.Text = bFile.ReadLine();
}
```

Once again, it is important to close the file.

```
bFile.Close();
```