Lesson 6: Error Handling

Try-catch-Finally

Provides a way to handle some or all possible errors that may occur in a given block of code, while still running code.

```
Try
    [ tryStatements ]
    [ Exit Try ]
[ Catch [ exception [ As type ] ] [ When expression ]
    [ catchStatements ]
    [ Exit Try ] ]
[ Catch ... ]
[ Finally
    [ finallyStatements ] ]
End Try
```

More than one **catch** clause can be included. A try block must include at least one catch clause. Notice that a square bracket follows the keyword **catch** to indicate that the parentheses and the argument list are optional. Omitting the argument list makes the catch generic-meaning any exception that is thrown is handled by executing the code within that catch block. The finally clause is also optional. It can be included if there is a segment of code that needs to be executed, no matter what. When code is included in all three blocks for the try…catch..finally constructs, the statements inside the try block are attempted first.

```
public class EHClass
{
void ReadFile(int index)
{
// To run this code, substitute a valid path from your local machine
string path = @"c:\users\public\test.txt";
System.IO.StreamReader file = new System.IO.StreamReader(path);
char[] buffer = new char[10];
try
{
file.ReadBlock(buffer, index, buffer.Length);
}
catch (System.IO.IOException e)
{
Console.WriteLine("Error reading from {0}. Message = {1}", path, e.Message);
}
finally
{
if (file != null)
{
file.Close();
}
}
// Do something with buffer...
}
}
```