



COMPSCI 280 S1 2011 Enterprise Software Development

Programming Fundamentals
File and Stream IO



Agenda & Reading

- ▶ **Agenda:**
 - ▶ File and stream IO
 - ▶ FileInfo & DirectoryInfo
 - ▶ Writing/Reading Text
 - ▶ Binary Streams(Reading/Writing Primitive Types)
- ▶ **Recommended Reading**
 - ▶ Basic File I/O
 - ▶ <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconbasicfileio.asp>
 - ▶ System IO namespace
 - ▶ <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpref/html/frlrfssystemio.asp>
 - ▶ C# for Java Programmers
 - ▶ Chapter 11
- ▶ **Hands-On Lab:**
 - ▶ Lecture12Lab



File and Stream IO

Add an using directive:
`using System.IO;`

- ▶ The System.IO namespace contains types that allow reading and writing on data streams and files.
 - ▶ Classes Used for File I/O
 - ▶ DirectoryInfo
 - Provides instance methods for creating, moving, and enumerating through directories and subdirectories
 - ▶ FileInfo
 - Provides instance methods for the creation, copying, deletion, moving, and opening of files
 - ▶ FileStream
 - Supports random access to files. FileStream opens files synchronously by default.
 - ▶ Streams provide a way to write and read bytes to and from a backing storage.
 - ▶ Classes used for reading from and writing to Text Streams
 - StreamReader & StreamWriter
 - ▶ Classes used for reading from and writing to Binary Streams
 - BinaryReader & BinaryWriter

Opens sequential file



FileInfo

```
FileInfo fi = new FileInfo("ABC.txt");
if (fi.Exists)
    Console.WriteLine(fi.FullName);
```

Example:FileIO

- ▶ Provides instance methods for the creation, copying, deletion, moving, and opening of files, and aids in the creation of a FileStream.
- ▶ Properties of the FileInfo object
 - ▶ Length: Gets the size of the current file.
 - ▶ Extension: Gets the string representing the extension part of the file.
 - ▶ FullName: Gets the full path of the directory or file
 - ▶ Exists: Gets a value indicating whether a file exists.
- ▶ Methods of the FileInfo object
 - ▶ Delete()
 - ▶ Permanently deletes a file.
 - ▶ MoveTo(String)
 - ▶ Moves a specified file to a new location, providing the option to specify a new file name.
 - ▶ CopyTo(String)
 - ▶ Copies an existing file to a new file, disallowing the overwriting of an existing file. Throw IOException if the destination file already exists.

```
f2 = fi.CopyTo(destPath + "ABC.txt");
f2.Delete();
```



DirectoryInfo

- ▶ Provides instance methods for creating, moving, and enumerating through directories and subdirectories.
 - ▶ Note: Use "." to denote the current directory; use ".." to denote the parent directory
- ▶ Properties of the DirectoryInfo object
 - ▶ Exists, FullName ...
- ▶ Methods of the DirectoryInfo object:
 - ▶ Create
 - ▶ Creates a directory.
 - ▶ GetFiles(String)
 - ▶ Returns a file list from the current directory matching the given searchPattern. Example: "*txt", "s*.txt"
 - ▶ GetDirectories()
 - ▶ Returns the subdirectories of the current directory.

```
DirectoryInfo di = new DirectoryInfo(".");
FileInfo[] fiArray = di.GetFiles();
foreach (FileInfo file in fiArray) {
    Console.WriteLine(file.Name);
}
```

5

COMPSCI280

Handout12



Writing To Files

- ▶ Declare a new StreamWriter object
 - ▶ If the file does not exist, a new one is created
 - ▶ The default location for file is the bin\Debug directory
- ▶ Use StreamWriter's WriteLine/Write methods
 - ▶ Copies the data to a buffer in memory
 - ▶ Write, WriteLine Method
- ▶ Call StreamWriter's Close method
 - ▶ Transfers the data from buffer to the file and release system resource used by the stream (or use an Using statement)
- ▶ Use try and catch for exception handling
 - ▶ To make your application more robust

True = Append data to this existing file

```
StreamWriter sw = new StreamWriter(filename, true);
```

```
sw.WriteLine("Well done.");
```

```
sw.Close();
```

Always close the file

```
StreamWriter sw = null;
try {
    ...
} catch (Exception ex) {
    ...
} finally {
    if (sw != null)
        sw.Close();
}
```

6

COMPSCI280

Handout12



Reading From Files

- ▶ Declare a new StreamReader object
 - ▶ Note: The file must already exist in the specified location or an error is generated
- ▶ Use StreamReader's ReadLine/ReadToEnd methods
 - ▶ ReadLine(), or
 - ▶ Reads the next line of data
 - ▶ Use a loop to read through
 - ▶ ReadToEnd()
 - ▶ Reads the stream from the current position to the end of the stream.
- ▶ Call StreamReader's Close method to releases system resources
- ▶ Again, use try and catch for exception handling

```
StreamReader sr = new StreamReader(filename);
```

```
while ((line = sr.ReadLine()) != null){
    Console.WriteLine(line);
}
```

```
Console.WriteLine(sr.ReadToEnd());
```

```
StreamReader sr = null;
try { ... } catch (Exception ex) { ...
} finally {
    if (sr != null) sr.Close();
}
```

7

COMPSCI280

Handout12



Binary Streams

- ▶ The FileStream Class is used for reading from and writing to files.
 - ▶ FileStream constructor parameter
 - ▶ FileMode – Open, Append, Create
 - ▶ FileAccess – Read, ReadWrite, Write
 - Read, write, or read/write access to a file
 - ▶ FileShare – None, Read, ReadWrite, Write
 - Define whether two process can simultaneously read from the same file
- ▶ BinaryWriter
 - ▶ Writes primitive types in binary to a stream
 - ▶ Constructor: BinaryWriter(Stream)
 - ▶ Methods: Write(int32), Write(String) ...
- ▶ BinaryReader
 - ▶ Reads primitive data types as binary values
 - ▶ Constructor: BinaryReader(Stream)
 - ▶ Methods: ReadInt32, ReadChar ...

8

COMPSCI280

Handout12



Writing primitive types to Files

- ▶ Declare a new `FileStream` object
 - ▶ Parameter: filename, `FileMode = create`
- ▶ Declare a new `BinaryWriter` object
 - ▶ Parameter: `FileStream` created above
- ▶ Use the `Write` method
- ▶ Close the `BinaryWriter`
- ▶ Close the `FileStream`

```
try {  
    fs = new FileStream(filename, FileMode.Create);  
    w = new BinaryWriter(fs);  
    for (int i = 0; i < 11; i++)  
        w.Write(i);  
} catch (Exception ex) {  
    ...  
} finally {  
    if ( w != null) {  
        w.Close();  
        fs.Close();  
    }  
}
```



Reading Primitive Types

- ▶ Declare a new `FileStream` object
 - ▶ Parameter: filename, `FileMode = Open`
- ▶ Declare a new `BinaryReader` object
 - ▶ Parameter: `FileStream` created above
- ▶ Use the `ReadInt32` method
- ▶ Close the `BinaryReader`
- ▶ Close the `FileStream`

```
try {  
    fs = new FileStream(filename, FileMode.Open, FileAccess.Read);  
    r = new BinaryReader(fs);  
    for (int i = 0; i < 11; i++)  
        Console.WriteLine(r.ReadInt32());  
} catch (Exception ex) { ...  
} finally {  
    if ( r != null) {  
        r.Close();  
        fs.Close();  
    }  
}
```