**Name: Jason Truong**

For each of the below questions, write a short sentence or two to express (in your own words) your answer. Keep the answers short, but use complete, correct, English sentences.

If it helps to clarify the questions, feel free to mentally prefix all the questions with the phrase "According to the video…"

* After you’ve watched all the videos, please answer this question:  
  Of all the videos that you watched, if you could pick one video to be re-recorded by the instructor outside of class which would you choose? Why?  
  (Keep in mind the recording outside of class will omit any pauses from the instructor answering student questions, have less hemming and hawing, etc, and generally be more concise)

|  |
| --- |
| < Write your answer here > |

**VIDEO: File I/O (Input) In C#**

File I/O In C#

* What does “I/O” mean?

|  |
| --- |
| Reading or writing into a file |

* What is **input**? Specifically, where is the data starting out, and where does the data move to?

|  |
| --- |
| Input tries to get outside data into a file |

* What is **output**? Specifically, where is the data starting out, and where does the data move to?

|  |
| --- |
| Output tries to send data out of the program and stores it into the file |

* What are the three steps that you need to do in order to interact with the data contained within a file?

|  |
| --- |
| Open the file, asking the OS to open the file for us  Read the data from your file or Write data to your file  Close the file, by telling the OS to close it cause we're done |

* If we want to move, or delete, or rename (etc.) an entire file, do we need to do the above three steps?

|  |
| --- |
| No, they don't manipulate the file with those steps above. |

* Give an example of how the operating system can optimize I/O when **reading data out of a file**.

|  |
| --- |
| When reading data out of a file, it would be chunking information out of the file sequentially, and the OS would optimize its reading from the file.  And then remember this for future-reads. |

* Give an example of how the operating system can optimize I/O when **writing data to a file**.

|  |
| --- |
| When it writes data to a file, it will know if you keep using the data on the file continuously for the data to the hard drive later on. |

* What does the word ‘directory’ mean?

|  |
| --- |
| Its those folders you see on your PC. |

* What is a path?

|  |
| --- |
| These are the strings of directories or folders that are told as a path. |

* How do absolute paths differ from relative paths?

|  |
| --- |
| The absolute path starts at the top-most directory/folder and lists all the folders that you need to find the file.  The relative path starts in the current directory and describes the folders you need to open from there.  When not speaking in-verbatim, one's relative to the folder directional speaking from a specific point of the path, where absolute takes you all the way from the drive letter. |

* What is the program’s “current working directory”?

|  |
| --- |
| The current working directory is where relative paths would start from, it would just assume the current directory that is ... being worked in. |

* When writing a relative path, what is different about the beginning of the path?

|  |
| --- |
| The Drive Letter, like E:\. |

* Why is it important to tell Visual Studio what the current working directory is?

|  |
| --- |
| You'll tell Visual studios where to run. If not setup, it'll crash. |

* When you run the program is run (WITHOUT setting the current working directory) what happens?  
  What does this tell you?

|  |
| --- |
| You get a bug and the code crashes.  "Directory Not Found". |

* How do you open up the data file in Visual Studio?

|  |
| --- |
| Click on properties, click on debug, and in 'working directory' work from there and select the applicable folder. |

Manipulating Files

* For all things that we’re going to talk about, what do you need to put at the top of your C# source code file?  
  **NOTE: This is also needed for the code that reads information out of the file (not just for deleting/renaming/etc)**

|  |
| --- |
| using System.IO; |

* What does the backslash character ( **\** )do, in general?  
  What does **\\** mean?

|  |
| --- |
| A \-backslash is an escape character from the c-driven languages' string.  \\ means a single backslash-\ |

* What is a static method?

|  |
| --- |
| A static method can call it on the class directly, you don't need to make an object for it. |

* Why is it useful for File.Exists to be a static method?

|  |
| --- |
| You don't need to create another object when you're just looking for a File that exists within the Path  File.Exists(path) |

Reading Data From A File

* Give a brief, intuitive description of what are binary files?

|  |
| --- |
| An intuitive yet arbitrary lines of data.  .EXE, .Videos, .Image Files |

* When you open up a text file in Visual Studio (or Notepad, etc), what do you see?

|  |
| --- |
| Readable text comprehensable by humans. |

* Give some examples of text files

|  |
| --- |
| Notepad, source code files, HTML web pages |

* How can you check what data is in a text file?

|  |
| --- |
| Open the file. |

* Both File.OpenText and new StreamReader accept a parameter. What is that parameter?

|  |
| --- |
| A file path. |

* Give an example of what might happen if your program don’t close a file that it has opened.

|  |
| --- |
| The operating system would tell you to stop any manipulation of the file, because it forgets to close it. |

* What does the **using(...) { ... }** construct do for you?

|  |
| --- |
| The portions in the middle would create an object that would require you to have a 'disposable' file you can work with. |

* What is a ‘verbatim’ string? How do these differ from normal C# strings?

|  |
| --- |
| The @ sign is a 'verbatim string' rule, so there are no escape characters. Meaning it disables any escapes, so \ just means \.  You can still put .txt however, and various others alike. |

* What happens when your program tries to open a file that doesn’t exist?

|  |
| --- |
| It gives a "Could not find file" error for you. |

* What is an exception?

|  |
| --- |
| An object representing a runtime error.  Non-existant files, dividng by 0, calling characters @ on a string that has passed too large an index, being out of bounds. |

* Describe a typical way for a program to react to an exception being thrown

|  |
| --- |
| When an exception is thrown, there is a possible for a 'catch' to control the crash and handles or fixes it entirely. |

Parsing Data In A File

* What does it mean to ‘parse’ a file?

|  |
| --- |
| Take data in a format and distinguish them from each other. |

* What is a token?  
  What are they separated by? (Give specific examples of whitespace)

|  |
| --- |
| A descreet unit of user input separated by whitespace. |

* List all the tokens on the following line:  
  **1 “John Smith”**

|  |
| --- |
| 1 int,double,String  "John String  Smith" String |

* Why are fixed-width / monospaced fonts useful when looking at text data files (and programming language files)?

|  |
| --- |
| The text is super organized, almost to the point of a grid-like collumn to row lettering. |

* What does the ReadLine() command do?

|  |
| --- |
| It will read the lines up until the newline character, throwing away the \n(ewline) character.  Doing it again will call the next line and rid the \n character. |

**It looks like I lost the drawings from slide 23 (“File Input Answer”) – sorry about that!**(I think that the program is simple enough that you should be able to follow along anyways)

* What is “leading whitespace”?  
  What is “trailing whitespace”?

|  |
| --- |
| The leading whitespace are the blank spaces/lines in the front  The trailing whitespace are just the leading whitespace, but instead in the back.  These include \t |

* What TWO THINGS happen when you give Int32.TryParse a string that contains a valid integer?

|  |
| --- |
| It gives a valid string an Int32 and parses it into an Integer and returns a 'True' bool.  If false, it will parse it directly as 0 and then return a 'False' bool. |

* How do Int32.TryParse, Double.TryParse, and Int16.TryParse differ?

|  |
| --- |
| Int32 will parse whole numbers  Double will try to parse any real number  And short/Int16, will parse whole numbers up to 6500 |

**Note that even though there aren’t a lot of questions about the example programs you should study these programs closely. They’re valuable because they can both explain how to use these concepts (and code) to solve problems and they provide useful models to follow when creating your own programs.**

* If we want our program to read all the lines in the file (regardless of how many lines there are), how do we set up the program?  
  (You can answer this by briefly describing the differences between this program and the prior program)

|  |
| --- |
| The specific portion of the code that changes the program from reading a particular point into a 'read all lines', is the  while (sLine != null)  where sLine is the file.ReadLine(); |

* What does ReadLine() return when there’s nothing else in the file?

|  |
| --- |
| Null. |

* If we want our program to ignore non-numeric tokens, how do we set up our program?  
  (You can answer this by briefly describing the differences between this program and the prior program)

|  |
| --- |
| The array of the individual characters, the string[] of tokensFromLine, and another loop foreach. |

* What is a ‘delimiter’ character?

|  |
| --- |
| The token's/characters unprovoked or delimited by the specified characters, which in this case are blank-spaces and tabs. |

* What does the .Split() method do?

|  |
| --- |
| It would split the string its referencing to with the specified delimiters. It can remove the split entries that are empty. |

* What does the ReadToEnd() method do?

|  |
| --- |
| Gets all the lines. It can be .Split to break the string and parse them into individual tokens where splittable into an Array. |

* What is a downside to reading the entire file into memory all at once?

|  |
| --- |
| If the file is a huge file, like 10gigs, it will eat that much of the memory; or more. Which is pretty tremendous. |

VIDEO: File I/O (Output) In C#

* When talking about file output, where is the data going to?

|  |
| --- |
| From the program to outside the program into an external file. |

* We’re going to be using StreamWriter objects to write to files, using several methods.  
  Which methods are these, and where have you seen (and used) these methods before?

|  |
| --- |
| Console.Write and Conosle.WriteLine |

* When printing to a file, will the output immediately be written to the file on the disk?  
  If not, where is that information being stored?

|  |
| --- |
| No, it's cached in memory first to avoid slowing down the drive. |

* When will the information be written to the disk?

|  |
| --- |
| When you call Dispose(). |

* C# provides a way to ensure that the Dispose() method is called. Copy the example code from the slides and put in a brief comment explaining what each line does.

|  |
| --- |
| StreamWriter output = new StreamWriter("out.txt");  output.WriteLine("Hello, file!");  output.WriteLine("This is a second line of output.");  output.Dispose();  The first line represents a StreamWriter object with the path leading to the 'out.txt' file.  The second and third line write a message on the file, each writing their own message, then creating a newline on the file; by the end of it, we're on the 3rd line of the .txt file.  .Dispose() will flush out the information from the memory and write it into the disk. |

* Why will we NOT be using the C# using construct in these slides?

|  |
| --- |
| We're using the 'dispose' pattern instead of the 'using' pattern for getting more used to the universal construct of the other C-fundamental languages; Java, C... etc. |

* If you create a new StreamWriter object (in order to write data to a file) and the file does NOT exist, what happens?

|  |
| --- |
| If the file does not exist, it will be created. |

* If you create a new StreamWriter object (in order to write data to a file) and the file DOES exist, what happens?

|  |
| --- |
| If the file exists, it will be overwritten. |

* How can you check that your program has successfully written the data to the file?

|  |
| --- |
| You'd have to open the file yourself. |

* For the ‘Hours’ question, put in a quick explanation of what each part means:  
  *123 Kim 12.5 8.1 7.6 3.2*  
  E.g., explain what the ‘123’ means, what the ‘Kim’ means, etc.

|  |
| --- |
| 123 would represent their ID number  Kim is their name  The 12.5, 8.1, 7.6, and 3.2 represent the hours worked on those unspecified days. |

* In the answer to the Hours problem, explain what this line does, and how it works:  
  *while( (sLine = input.ReadLine() ) != null )*

|  |
| --- |
| While the current line being read isn't an empty line or out of bounds, it will continue til'. |

* Why is it necessary to check if count is zero?

|  |
| --- |
| To check if the Line that is being checked has that Employee's ID information. |