File I/O - Writing Files - Lecture Notes

(05/25/2018)

Overview of Session

Continuing on yesterday's discussion, when we work with files there are three basic operations:

Session Objectives:

- Students should be able to investigate File and Directory metadata using the File class
- Students should be able to write text data to a file
- Students should be able to explain the concept of buffering in File I/O
- Students should understand the importance of releasing external resources

Class Preparation

- None

Customary schedule for session:

- File Object (0:30)
- Orders of Magnitude (0:15)
- Break (0:10)
- Buffering (0:20)
- Writing (0:20)

Topic List w/Notes
(and links to e-book section when available)

This is a short day. It builds on top of the day prior when working with Reading File I/O.

There is an opportunity to get students to better understand the idea of an underlying stream, but for most they'll pick up the concept of writing out to a stream pretty easily.

Use the extra time for review by writing a little program in class.

Writing File I/O

- The most common class used to write content out to a file is the **Console** (.NET) and the **PrintWriter** (Java).
- Other types of writers also exist: binary data, network data via memory stream

Demonstrate opening a file for writing.

Orders of Magnitude Performance

- In computing there is a significant difference (not always noticeable) between varying operations

- Computers are Fast (First four sections)
    - Demo: Computers are much faster with CPU cache than RAM, which is faster than file and network

File Buffering

- Because of performance differences, we want to write our content from memory to disk as little as possible
- To do this, we use **buffers**

A buffer is like a bucket you catch water in. Once the bucket is full, you can dump it out, ready to be filled again.

- Content is buffered and not saved (or cannot be seen) until the stream is *flushed* or *disposed*.

Flushing a buffer means transmitting any accumulated character output to a file.

- Forgetting to close the stream leaves it open in write mode.
    - By using try-with-resources block, the file closes automatically

Have you ever tried deleting a file only to be told that *this file is opened in another program*?

Writing

- You can write to a file using:
    - (**Java**) `print` methods like `println`, `printf`, etc.
    - (**.NET**) `Write`, `WriteLine`

Real World Usage Scenarios

- Logging
- Cleaning and sanitizing data inputs
- Transmitting data to external systems