6.178 Introduction to Software Engineering in Java

Lecture 7: Files & I/O

What is I/O?

- Stands for Input/Output
- Standard Input/Output
  - Provided by Operating System
  - System.out.println() is Standard Output
- Files
  - Read/Write
  - Files exist permanently

How Does It Work?

- Java reads Streams of Bytes
  - Byte Streams are raw binary data
  - Raw binary data is hard to work with
- Java provides abstractions to help
  - Character Streams convert binary data to the local character set (Usually UTF-8)
  - Slow because they read one character at a time

How Does It Work? (Con)

- Buffered Streams
  - Java keeps a buffer in its local memory
  - This buffer contains room for a lot of characters
  - When reading, a Buffered Stream reads a bunch of characters at once and stores them in Java memory
  - When writing, a Buffered Stream writes to a buffer until told to write to output

Standard Input

- Provided by Operating System
- Java has two ways of interacting with Stdin
  - Program arguments
    - Provided as an argument to main()
  - Input Streams (read from System.in)
    - Usually wrapped in Buffered Streams

Standard Output

- Provided by Operating System
- System.out
  - println(String string) prints with a newline at the end
  - print(String string) prints without a newline
- System.err
  - Used for error handling (not technically stdout)
  - Prints in red in Eclipse

Exceptions

- Streams expect to read until reaching a natural conclusion (pressing return key or reaching EOF) and write until closed.
- This doesn’t always happen
  - Operating Systems crash
- Java doesn’t like unexpected behavior
  - Statements reading I/O throw exceptions or use try/catch blocks (use the former)

Exceptions (con)

- What happens when a stream closes?
  - Exceptions terminate the program and print the error
  - Try/Catch blocks swallow the error and let the programmer deal with it (generally bad)
- Method signatures show exceptions
  - public static void main(String[] args) throws IOException { }

Exercise

- Adventure:
  - clone https://github.mit.edu/6178-iap16/<your_kerberos>_ lec7.git
  - Instructions are in the README.md
  - Complete Problem 1
  - TLDR: Create a Bridgekeeper class
    - Bridgekeeper takes an argument name (given in run configurations)
    - Bridgekeeper asks you for your name, your quest, and your favorite color
    - If you aren’t sure of your answers he won’t let you pass

Reading Files

- Files are read as a stream of bytes (chars and/or buffers if using Java abstractions)
- Located at a path
  - “myfile.txt” at your project level
  - “files/myfile.extension” if organizing
  - “/User/graeme/Documents/…..” if outside project
- Have an encoding
  - Usually UTF-8 (not always)

Reading Files (con)

- Files have extensions
  - Describe what kind of binary format they use
  - Plain Text: .txt, Comma Separated Values: .csv, Tab Separated Values: .tsv
  - Abstraction that helps developers recognize how data will be structured
  - Sometimes there will be libraries to wrap reading specific extensions

Reading Files (example)

- BufferedReader reader = new BufferedReader(new FileReader(filename));
  - File treated as a Buffered Stream
- While (true) {
  - String line = reader.readline();
  - if (line == null) break;
  - List<String> csvLine = Arrays.asList(str.split(“,”));
- }

Writing Files

- Java has several means of writing files
  - FileWriter
  - Files.write()
  - FileOutputStream
- Java writes binary data to files
  - Appending a file extension is meaningless
  - Make sure that the data you write is formatted properly by your program

Writing Files (example)

- List<String> thingsToWrite;
- FileWriter writer = new FileWriter(“outfile.txt”);
- for (String thingToWrite: thingsToWrite) {
  - writer.write(thingToWrite + “\n”);
- }
- writer.close()
- // Note that you must close the writer when you finish writing

Exercise

- Adventure:
  - clone https://github.mit.edu/6178-iap16/<your_kerberos>_ lec7.git
  - Instructions are in the README.md
  - Complete Problems 2, 3, and 4
  - TLDR:
    - Create an Oracle to read fortunes from fortunes.txt
    - Create a Scribe to write messages to messages.txt
    - Tell your oracle to read from your messages