Asgn 1 (CS3310 F15) Demo Specs (& Related Notes)

TO DOseudocode for Main

- DELETE 2 output files: Log.txt, Backup.txt
  (Yes, this should be done by the code, even if you manually deleted these OR you're opening them in truncate mode, when appropriate).
- Call setupMain – sending in "AZ" for fileNameSuffix
  (where that method will concatenate it with the pathString, "RawData" & ".csv" which are all hard-coded)
- Call prettyPrintMain – no parameters need to be supplied since that method automatically uses IndexBackup.txt (hard-coded into code)
- For loop with i going from 1 to 3
  - Call userAppMain sending in i for fileNameSuffix
    - (where that method will concatenate it with the pathString, "TransData" & ".txt" which are all hard-coded)
- Call prettyPrintMain – no parameters needed

WHAT TO DO FOR THE DEMO

1. Delete the 2 output files: Log.txt and Backup.csv (This is also handled by the programs because of how the files are opened by constructors in the appropriate classes).
2. Run Setup program
3. Run PrettyPrint program (either Jia's (JG) or Devlin's (DG) version)
4. Run UserApp program
5. Run PrettyPrint program (either…)
6. Print Log.txt file in WordPad or…
   - *Use MUST a **FIXED-WIDTH FONT** (like Courier New) so things line up nicely*
   - *Use a smaller font, if needed, to avoid any wrap-around in Log file printout*
   - *NOTE: The Log file is **ONE LONG FILE** which includes the output from running the programs in steps 2,3,4,5 above – all captured in a* **SINGLE Log file**
7. Print Backup.csv file in WordPad or…
   - (Yes, PrettyPrint already printed it to the Log file, with things nicely aligned).
   - (Yes, things won't line up, fields won't be justified/truncated like they are in PrettyPrint, numbers won't have embedded commas, there'll be commas separating the fields, etc.)
8. Print all of your program code files.
   - [I don't need PrettyPrint since everyone is using either the JG or DG version].

WHAT TO HAND IN (in the order specified below)

1. Cover sheet (fill in the top & sign it)
2. Printout of Log.txt data file
3. Printout of Backup.csv data file
4. Printout of YOUR code files: ***(IN THIS ORDER)*** *(There are at least 6 actual separate files]*
   - Setup PROGRAM
   - UserApp PROGRAM
   - RawData OOP CLASS
   - UIoutput OOP CLASS
   - DataStorage OOP CLASS
   - UIinput OOP CLASS
   - any other code files you wrote for your program

HOW MUCH COMMENTING IS NEEDED?

- **Self-documenting** code including:
  - descriptive **NAMING** of programs, methods, classes, objects, records, fields, namespaces/packages, variables, constants, etc. *[according to traditional C#/Java/C++ naming conventions]*
  - using the same naming as used in the **SPECS** (so everyone's on the same page, and so the specs serve as external documentation)
  - good **MODULARIZATION**, short modules (no method > 1 page/screen-ish), sharing of DataStorage & UIoutput class and using the modularization described in the specs and in class (so everyone's on the same page)
  - following the **PROJECT SPECS** closely, so that the project designer's specs serve as part of the external documentation (which therefore does NOT need repeating within your code).
- A **top-comment** on each physical file with: overall project/app name, the module name the code author's name & date code was last changed
- Internal comments on **tricky code** or unusual ways of doing things or things which don't quite follow the specs exactly (since a maintenance programmer would read the specs and ASSUME that the program would OF COURSE follow them)
- You do **NOT need line-by-line** commenting

NOTES:

- Re-read specs for A1 to make sure you're doing everything right (to maximize points)
- Setup and UserApp controller modules all use the input stream processing algorithm (on RawData and TransData files, respectively) – i.e., loop through the data til done {, doing 1) read in a single record/line then 2) completely deal with it}.