Asgn 1 Demo Specs (& Related Notes)

Pseudocode for what TestDriver does

```
• Deletes Log.txt file AND Backup.txt file
• Run Setup – sending in "Sample" for the fileNameSuffix
      so Setup can pass fileNameSuffix along to RawData's constructor which opens the
           right file:  pathString + "A1RawData" + fileNameSuffix + ".txt"
• Run PrettyPrintUtility
• FOR LOOP with i going from 1 to 3
      Run UserApp sending in i (as a string?  or an int?  or. . .)
          so UserApp can pass i to UI's constructor which opens the right file:
          pathString + "A1TransData" + fileNameSuffix + ".txt"
• Run PrettyPrintUtility
• Run Setup – sending in "" (i.e., an empty string) for the fileNameSuffix . . .  so. . .
• FOR LOOP with i going from 4 to 4
      Run UserApp sending in i . . .
```

WHAT TO DO FOR THE DEMO

1. A1RawDataSample.csv, A1RawData.csv and the 4 A1TransData?.txt files must be in the correct folder in your project
2. Run the TestDriver program
3. Print Log.txt file in WordPad or…
    - *Use a FIXED-WIDTH FONT (like Courier New) so record fields line up nicel*
    - *Use a smaller font, if needed, to avoid wrap-around in Log file printout*
    - *NOTE: This in ONE LONG FILE which includes TestDriver's running Setup and UserApp and PrettyPrintUtility multiple times – all captured in a SINGLE Log file*
4. Print all of your program code files.

WHAT TO HAND IN (in the order specified below)

1. Cover sheet (fill in the top & sign it)
2. Printout of Log.txt file
3. YOUR program code: *(IN THIS ORDER) (There are at least 8 actual separate files]*
    - TestDriver program
    - Setup program
    - UserApp program
    - PrettyPrintUtility program
    - RawData class
    - UI class
    - DataTable class
    - NameIndex class
    - any other code files/classes you used in 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 in the **SPECS** (so "everyone's on the same page")
    - good **MODULARIZATION** using OOP (except the TestDriver and PrettyPrintUtility don't use OOP), short modules (no method > 1 page/screen-ish), sharing of DataTable and NameIndex classes and using the modularization described in the specs and in class (so "everyone's on the same page")
    - following the **REQUIREMENT SPECS** closely, so that your "boss's" specs act as a form of external documentation (which does NOT need repeating within your code).
- A **top-comment** on each physical file with: overall project/app name, the module name & code author's name
- A **comment-line-of-*'s** between chunks of code (e.g., methods, constructor, …)
- Comments on **tricky code** or unusual ways of doing things or things which don't follow the specs (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)
- Both Setup and UserApp use the input stream processing algorithm (on RawData and on TransData, respectively). So, looping through the 2 data files is done by Setup and UserApp controlling things and NOT inside RawData class and UI class (for TransData).