CISC / CMPE 327 - Fall 2019

Course Project Assignment #4 - Back Office Rapid Prototype

In this assignment, you will design and rapidly program the first version of your team's Back Office. Since the next assignment will involve testing it, do not do full testing of your Back Office at this time (no marks are for correctness yet!)

As much as possible, try to keep in mind the principles of Incremental Development (IDP) and eXtreme Programming (XP) while creating your solution. Using Incremental Development, structure your programming as a sequence of subsets, where at each stage you implement only the next most essential feature, creating a running prototype before moving on to adding the next feature. Using XP, work together on your program using Pair Programming, where one of you advises on the higher-level design issues while the other does the actual coding.

What to Hand In

In this assignment, you will hand in:

1. A **design document (PDF file)** for your Back Office, giving the overall structure of your solution, showing the classes and methods as a diagram or table, with a brief (one sentence) description of the intention of each one.
2. The first version of the source code to implement your design (through GitHub). This version should run on at least some inputs but should not yet be completely tested (since that will be the next assignment, and it's better to leave some failures until then).

Your solution to this assignment will be judged on the clarity and readability of the design and the code, so work at using your best programming practices, including naming variables, classes, and methods meaningfully, commenting liberally to make it easy for another programmer to understand, and avoiding "code smells".

Your solution to this assignment will **not** be judged on its correctness, rather on the quality of its design and coding. This is to be a rapid first version, not a final product. Design and program it to work correctly, but don't worry about getting it fully debugged or tested yet, since that's what we'll do next in Assignment #5.

Marking Criteria

Assignment 4 will be marked according to the following criteria:

Design (Primarily in the design document) 4 marks

Architecture
- clearly documents structure of solution
- explicitly describes intention of each class and method
- accurately reflects solution structure
- clearly shows where inputs and outputs fit in

Completeness
- evidently addresses all required functionality
  (solution has parts to address all required operations and results)
- has specified inputs, outputs, and files (only!)
  (produces a log on standard output, has **two** input files -
  Old Master Accounts File and Merged Transaction Summary File,
  two output files - New Master Accounts File and New Valid Accounts File,
  and does not assume any other inputs or outputs)

<u>Source Code</u> (Details of the source code itself) 4 marks

Structure and format
- code is structured and formatted such that architecture is clearly visible in code
- naming of classes and methods clearly reflect their role in the solution
- as little cloning or redundancy as possible

Maintainability
- avoidance of coding tricks and hacks
- simplest solution possible, no frills and extras
- clearest solution possible, no gratuitous optimizations

Internal Documentation
- clear naming of all variables and constants to reflect their role in the solution
- comments at beginning of every class and method clearly documenting their interface and intention
- comment at beginning of main program documenting overall program intention, input and output files, and how the program is intended to be run

<u>Overall Presentation & Quality</u> 2 marks

=======
<u>Total</u> 10 marks