Scenario:

- You have been hired as an entry-level JAVA developer for a small technical university.
    - “Middle Earth Technical University” (METU)
- You are working for “Advising Services” supporting academic advisors and their functions.
- Dean Gamgee has decided that each student needs to be individually assigned and academic advisor.
    - Currently advisors are seen on a drop by basis only and students get whoever is available.
    - Assigned advisors should improve student ←→ advisor communications and aid in improving student retention
- You have been paired with a JAVA developer that is also serving as your mentor to teach you the techniques used by METU and guide you through the specifications for each application.
- Your first task is to build a JAVA application that will help in the advisor assignment process.
- The process, at a high level, is:
    - A school/department staff member submits a comma delimited (.csv) file to the system
        - Comma delimited is used because software is already in use that can easily create the .csv file.
    - The system (using the app you are developing) loads the .csv file into a String array
        - It is loaded into the array so it may be used by other aps simultaneously in the future
    - The String array is then processed by your ap to 1) check for valid data 2) write invalid data to an error file 3) reformat the valid data and 4) write the reformatted valid data into a valid data file so it may be picked up and processed by other aps
    - The input file may be picked up by the system from the user’s workstation, but the two output files are always placed in the same folder on the same drive (server)

Input File Format:

- Comma delimited
- Text
- Five fields
    - Field 1: Advisor First Name
    - Field 2: Advisor Last Name
    - Field 3: Advisor ID
        - 10 digits in length, but can start with leading zeros that must be retained
    - Field 4: Major advised for code
        - Six non-numeric characters maximum
            - At least 3 characters must be present
        - Letters of the alphabet
        - All uppercase and must remain uppercase
    - Field 5: Maximum advisees that may be assigned
        - Integer values only
        - Do not worry about the “range” now

Small "valid" sample file follows:

```
Betty,Rubble,0007651235,CITBS,230
Vlad,Alucard,2008761232,BUSAD,100
Sam,Spade,0004562143,INFOBS,85
Fred,Flintstone,265098612,ENGR,175
Bram,Stoker,0000987345,BIOL,50
Frank,Enstein,0005629841,BUSCOM,75
Pearl,Pureheart,2983124567,ENGR,75
Wilma,Flintstone,0000761930,MATH,65
Douglas,McArthur,0003419120,CITBS,200
Yogi,Berra,0000451983,CJUS,125
Wilma,Rudolph,2012128643,POLS,225
```

- Your job is to read the file from wherever it is placed (yes you have access), data validate it for correctness, manipulate it into a slightly different format, and write a new version of all VALID rows to a new file placed in a standard location.
    - The standard location is already coded in the executable class for you…let's look at it.
- You must also create an error/suspense file that contains only the rows that contain errors and place it into a standard location.
    - Again…the standard location has been coded for you…look at it.
- Your teammate/mentor and you have decided to code the application using the main method to only call other methods to do the work. Your teammate has coded the "shell" you agreed on and it is your job to place code in the individual methods that do the actual work.
    - Let's look at the shell: `AcademicAdvisors.java` that is supplied for you to use
- Specifications:
    - Look inside the method declarations for specifications for each method you must code.
    - After you have finished the project, you must remove all commented instructions from every method. Minus 1 point per comment left in place per method.
    - You must use the definitions created for you at the top of the class.
        - Points are lost if you do not.
        - This helps to determine that you can read code
        - Note: You have seen examples of everything needed (as best I can tell) so before you ask questions, look at your demos from class.
    - All data written to any file must be written in a <u>**single**</u> write that writes an entire row at one time
    - Notice that you are reading a String with **5 fields**, and writing a String with **6 fields**
    - Notice the input and **bad** data file are comma delimited, but the **valid** data file you are writing is pipe delimited (|)

- All unchecked exceptions THAT ARE INDICATED BY THE COMPILER must be handled using try/catch.
    - Do **NOT** handle any other checked exception other than those indicated by the compiler! Yes, I know that in the "real world" you might do so!
    - These are basically your I/O exceptions that are thrown when opening or closing a file and when reading or writing to that file.
    - The required "handling" routine for I/O exceptions is:
        - Print an **appropriate/informative** message to the standard .err device stating the problem AND that the program is terminating.
        - Print the "stack trace"
        - Exit the program with a code of -1 if an error occurs.
- The single numeric data validation (number of advisees) must be handled using try/catch with the catch block handling the exception if it is thrown.
- For this program, **do not handle any other exceptions**…let's keep it simple! Violation will result in a point loss.
- Proper data types must be used.
- All current shop rules must be followed.
- All JAVA naming conventions must be followed.
- You may not change or delete any code that is supplied.
    - All supplied code must be used as-is.
    - You only add the code within the methods.
    - Changing existing code will result in the project not being accepted.
- Please re-read Project/Lab requirements in the syllabus.
- Review the IUPUI Code of Conduct
- Must be coded in a single class…the class you have been given.

- Project Value: 75 points (this is a fairly small one)
- Due Date: Monday October 14 no later than 11:59 PM.
    - Submit your source code only!
    - One file…no more!
- Grading will take approximately 7 – 10 days.
- Let's run through a file with some bad data…just for practice. Take good notes!

```
Betty,Rubble,0007651235,CITBS,230
Vlad,Alucard,2008761290,100
Sam,Spade,0004562143,INFOBS,85
Fred,Flintstone,265098612,ENGR,175.0
Bram,Stoker,0000987345,BIOL,50
Frank,0005629841,BUSCOM,75
Pearl,Pureheart,2983124567,ENGR,75
Wilma,Flintstone,0000761930,MATH,2o
Douglas,McArthur,0003419120,CITBS,200
Yogi,Berra,0000451983,CJUS,125
Wilma,Rudolph,2012128643,POLS,225
```