# Final Project - Famous Test Takers

Lots of famous mathematicians, philosophers, scientists and logicians have come and gone, and many are still here. They are hailed as our brightest minds. But, did they always get good grades? I don’t know. Probably. But, we can pretend a little!

For this project, you and your team are going to create an application that addresses something near and dear to your hearts – grades. The attached CSV file contains the names of such aforementioned thinkers and their completely fictitious grades in Calculus I. The grades for the class consist of six exams. These are famous thinkers with high IQs. They can take it. Well, the living ones can.

Please note the structure of the file:

1. Each line of the file is a comma-separated list.
2. The last name is listed separately from the first name.
3. Each line contains eight entries, two of which are devoted to the name.
4. Six of the entries are for test scores.

Requirements for Project completion:

1. The required calculations and procedures for this project must use functions/methods as necessary
2. You must use object orientation; that is, you must create classes in addition to the ‘main’ class. **Groups that do not use separate classes will receive an *automatic zero* for the submission.**
3. You must use at least one array.
4. You must employ exception handling (where needed and reasonable).
5. You must open the given CSV file and process it as required.
6. You must create a GUI for this application that displays required information.
   1. You are allowed to hard-code the CSV file name into the application, but you will get 3% extra if you can figure out how to allow the user to navigate to the file using the GUI.
   2. The GUI must employ such controls as labels, buttons, and text boxes as necessary.
   3. ***I recommend Swing***. It is more stable and less prone to introducing errors.
7. The full set of functional requirements are as follows:
   1. Allow the user to navigate through the list of people, displaying them one at a time.
   2. For each person, display their lowest score, highest score, and average score.
   3. Allow the user to request the average for each individual exam; call them Exam I – Exam VI.
   4. Allow the user to enter a new famous thinker and their test scores. The new person’s information should be written to the file when the application terminates.
   5. Allow the user to search for the name of a test taker and display their information.

Hints:

1. You may need as many as three different GUI ‘regions’ to satisfy UI expectations.
2. You could choose to use 8 different parallel arrays to carry out the required processing and display. It would get the job done, but I don’t recommend it. Hard to keep track of everything.
3. Another way to structure the processing, and therefore the code, would be to make each person as object. But where do you store them???