Computer Science AP
Capstone Project
Submission Date: June 8
Demo Days: June 9, 10   (to be confirmed closer to June)

Impress your classmates, future employers, or college and university officials. Create an original software solution to an interesting problem. Build software that's worthy of showing off and placing into your career portfolio. Those are some of the purposes of your Capstone Project.

You must create your software program using the Java programming language. Your program must be sophisticated, useful, and solves a problem of some kind. You must demonstrate an understanding of Object Oriented Programming techniques, advanced data structures (beyond arrays), and Object Oriented Analysis and Design principles – in short, show off all your computing science and software engineering knowledge.

The kind of software you create must make you stretch your learning, to help you grow as a student in computing science and software engineering.

There are ***at least*** five major steps to completing this project. These are the broad strokes of the project, but some small additional reporting requirements may be added as the need arises, and you will be informed when these additions are made.

Step 1 of 5: the Pitch
[*Recommended: 90 minutes, done at home*]

You must have an idea of what software to create, and gain approval, or at minimum verbal pre-approval, from your teacher to produce that piece of software as your Capstone Project. To seek approval, you will be creating a 2-minute pitch.

The 2-minute pitch must answer, essentially, three questions about your software:

1. What is it like?
    - Give an analogy for what your software will be like.

      e.g. "It's the X software for the Y problem type."
      e.g. "It's the Twitter for democracy activists needing to communicate without interference from governments."

2. What problem does it solve?
    - Succinctly define what problem your software will solve.
    - The problem may be of a scientific, engineering, business, etc., style of problem that, when solved, can continue to lead into other more interesting problems in an open-ended manner.

3. How does it solve the problem?
    - Answer how specifically your software will solve the problem.

      i.e. what approach are you taking to solving the problem, and how is it different than other possible approaches?

Deliverable:

1. A 2-minute pitch, written in correct sentences and paragraphs (as a MS Word or LibreOffice document), and submitted to D2L.

(Continued on next page!)

Step 2 of 5: the Design
[*Recommended: 90 minutes, plus any necessary time at home*]

Think carefully how your software needs to be built. What things and what kinds of things does your software need to model or represent? What types of actions can those things do? This step helps you design your software and requires two major components.

1. Write a short description using English sentences to describe the things your software will model, and what those things can do to themselves. You may write in either paragraph form or point form. (Computer Science AP students must instead write up SRP Analysis cards as per our Software Engineering and Architecture unit.)

   The purpose of doing this is to look for nouns and verbs in your description. The nouns (i.e. the things) are probably objects that needs to be built by writing Java classes. The verbs (i.e. the actions) associated with each noun or thing are probably instance methods in that Java class.

   Your description is complete when it describes all the things (nouns) and actions (verbs) that your software needs to fulfill its intended purpose.

2. When the description is complete, write up stub classes in Java using Netbeans for those nouns identified in the English description. In each of those classes, write up stub Java methods for each verb identified in the English description as being associated with that class.

Deliverables:
1. The English description (or SRP Analysis cards) of what things and actions needs to be modeled by your software to fulfill its intended usage and purpose. Write it in a MS Word or LibreOffice document, and submit to D2L.

2. The Java stub classes with stub methods corresponding to the nouns and verbs identified in the English description. Zip up all classes (e.g. the whole project folder) and submit to D2L.

Step 3 of 5: Implement prototype
[*Recommended: 3 classes*]

Write your software based on the stub classes previously created as part of your design.

If your software requires a standard Java GUI, you will need to write it correctly using proper MVC architecture as you were taught to in the Guided Projects. If your GUI has some special custom requirements, you will be required to explore how to implement those special requirements yourself.

The goal of creating your prototype is to create working software as quickly and correctly as possible (even if it doesn't work 100% correctly, it should work about 25% correctly). This prototype should quickly illustrate the potential of what the final software will be able to do, and be able to generate interest and excitement from onlookers.

Deliverables:
1. Prototype software. Project folder containing working source code must be zipped and submitted to D2L. You must also give a live demonstration of the prototype to your teacher.

(Continued on next page!)

Step 4 of 5: Iteratively and incrementally improve your prototype into the final product
[*Recommended: as much time in class and at home as necessary to complete your product before Demo Days*]

Once your prototype is complete, you will develop it into the final product iteratively and incrementally.

This means cycling through the following three steps:

A. Identify a small functionality or feature to implement into your software. GOTO step B.

B. Improve the software by writing, modifying, or fixing code to implement that particular small functionality or feature. GOTO step C.

C. Test that particular small functionality or feature in your software to ensure it works as intended.
   1. If a problem, bug, or error has been identified, then GOTO step B to fix it.
   2. Otherwise, GOTO step A.

Deliverables:

1. Final production software for release. Netbeans project folder containing working source code must be zipped and submitted to D2L.
2. You must also give a live demonstration of the final production software to other students during the class's Demo Days, and also to your teacher.

Step 5 of 5: Poster report
[*Recommended: as much time in class and at home as necessary to complete near the end of your software product development and before Demo Days*]

Once your final product is complete or nearly complete, you will document your solution to the original problem you had proposed to solve by creating a poster report. This means creating a digital poster (one or two letter or legal sized pages) with screenshots of your working software, as well as detailed textual descriptions, which documents:

A. The problem you are solving (as proposed and approved during step 1 of this project)

B. The design of your software or experiments to solve the stated problem, as well as rationale for why this design was chosen in comparison to other possible designs. The rationale may compare and contrast the advantageous and tradeoffs made in your design as opposed to other possible designs that were not chosen.

C. The software or experiment created from your design, including how it works, what it does, its features, etc.

D. Conclusions that can be made based on your software or experiment in its ability to solve the problem stated.

E. Finally, propose a few possible future works that can be made based on your software or experimental results here to e.g. learn more about the specific problem area, add essential functionalities.

Deliverables:

1. Completed poster report.
2. You must also give a live presentation of your poster report during the class's Demo Days as part of the step 4 process above.

Assessment:
In addition to the specifications and requirements above, the Pitch and Design will be assessed according to a Software Design Rubric; the Prototype will be assessed to a Java Technologies Assignment Rubric; the final software project will be assessed according to a Java Technologies Major Project Rubric; and the poster report will be assessed according to its quality of documentation according to a Java Technologies Major Project Rubric.