CS 350 Task 3: Project Proof of Concept Realization

Description

At this point in the project, we are transitioning to the design level. The background knowledge from Task 1 contributed to establishing notional requirements and specifications in Task 2. The premise of Task 3 is to convert those decisions into a semi-detailed representation of what needs to be done in the implementation stage, which is the primary role of the team activities coming.

This task ostensibly addresses the core elements of the overall project deemed to be most critical and in need of investigation before we commit to lesser elements of the design. To facilitate consistency because everyone had different solutions to Tasks 1 and 2, the specifications are provided.

The goal of this task is threefold. For the pre-task, you must critically consider the stages of "read → understand → plan" to ensure that you truly know what to do and how for the "execute → verify" stages, which is the actual task. The post-task is the "reflect" stage. This task is an exercise in both thinking and doing. Past experience consistently demonstrates that most students cannot write a basic program for a semi-real problem that compiles, runs, and gets the correct answer. Do not overestimate your abilities. Seriously apply what we have been doing so far throughout this quarter. Nothing in the solution requires programming skills beyond CS 210, 211, and 300, as well as basic algebra and trigonometry. Therefore, there is no reason not to be able to succeed except obstacles of your own doing. Do your best to recognize and avoid them.

Real-world documentation is rarely this complete, correct, or friendly. It is always your job to find and resolve the gaps, overlaps, inconsistencies, contradictions, and so forth.

Pre-Task

Read the Javadoc specifications and submit a plain text file with any questions or clarifications you need addressed. I will answer everyone's questions individually by email. Do as complete and correct a job as possible with this stage because your implementation decisions will depend on your understanding of the problem and the required form of the solution. The problem space is covered in Lecture 19.

It is acceptable to pose subsequent questions by email, but try to resolve the obvious ones now. We will not have a general Q&A session at the start of each lecture on content, only on procedure. You are responsible for yourself to get clarification from me as needed. Do not rely on others to do the thinking for you.

In the pre-task submission, rank the classes and their methods from most difficult to least, one per line, with a blank line between classes. Follow the specifications exactly.

Actual Task

Implement the specifications. Depending on the results from the pre-task, we may do only a subset. Do not jump to coding until the design is clear.

Post-Task

The details of the post-task will be determined based on how the actual task goes. The basis is to reflect on what went right and especially what went wrong, as well as why.

Deliverables

Submit the pre-task as a single file through the Pre Due link.

Submit the actual task through the Actual Due link. You may submit the files individually or zipped. Solutions that do not compile receive no credit.

Submit the post-task as a single file through the Post Due link.