SWEN 383 - Software Design Principles and Practices

HealthNCare App - Version 1.0 - Deliverables

Overview

All of the deliverable items shown below are due on the date and time shown in your section's course calendar.

NOTE: The ***essence*** of the project is design, though operational Java code must be produced in conformance to the design to help illustrate and validate the design's quality. The design activities, in total, are worth ***twice*** as much as the code itself. Keep in mind that, as this is a course on design, a functioning implementation of a bad design is worth ***much less*** than a "good" design with some missing parts in its implementation.

Design Sketch (10%)

The purpose of the ***Design Sketch*** is to get your team moving - it is all too easy to lay back, think deep thoughts, mull things over, and then realize you do not have enough time to do the job. So, fairly early in the project, your team is going to have to create a *reasonable* first try at a design as the basis for discussions with your instructor. This first try document will consist of the following:

1. A title page with the title "Design Sketch", the date, the team identification (e.g., "Team A"), and the names of all team members.

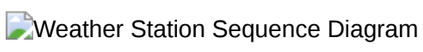
2. A rough, crude class diagram. For instance, below is one of the first whiteboard diagrams we created for the **WeatherStation** project:

   
   Weather Station Class Diagram

   Do not to spend time in **LucidChart** or other drawing programs putting lipstick on a pig - wait until we have an acceptable pig. There will be a need for a clean, fully specified diagram in the future, but not now.

   Right now, the goal is to arrive at a reasonable collection of classes, with inheritance and associations, that in total will provide the required functionality. Remember to incorporate the two required design patterns. To keep things simple, you can abstract all of the **UI** functionality into two *pseudo classes*: one for **Control** and the other for **View**. These will, of course, have to be refined into sets of real UI classes in the formal design document.

3. Rough, crude sequence diagrams; here is one of our earlier versions of the **WeatherStation** sampling loop.

   
   Weather Station Sequence Diagram

   In the case of this project, we are looking for sequence diagrams showing the object interactions for the following tasks:

   1. Read in a food database consisting of three basic foods, a recipe that contains two of the basic foods, and a recipe that contains the first recipe and the remaining food.
   2. Add two servings of a food to the log entry for the current date.
   3. Compute the total number of calories for the current date, assuming the log consists of a basic food and a recipe consisting of two basic foods.

   As in the case of the class diagram, the object is to ***think*** about how the design will accomplish the actions above at run time. Feel free to add a couple more sequence diagrams to the three required ones (we encourage you to).

4. A narrative in which the team provides:

   1. A brief description of each of the application classes other than those in the UI. The description must include each class's responsibilities (what the objects in the class ***know*** and what they can ***do***).
   2. A ***brief*** (2 paragraph max.) rationale for organizing the system in the way that you have. In particular, what are the advantages and disadvantages for both initial implementation and on-going maintenance / upgrades to a system that uses the design.
5. Feel free to test out your ideas by writing some simple skeleton classes (almost everyone feels better when there is working, if crippled, code to see). Just be ready to change it all based on discussions with and feedback from your instructor.

The sketch design ***must*** be present in the team's **doc** directory as a PDF file with the **exact** name **designsketch.pdf**. What is more, the version the team decides to submit ***must*** be committed to the team's repository and tagged **exactly** as **V1.0-DesignSketch** by the due date and time. Your instructor may modify these requirements in class, so be alert for any changes he or she specifies.

Preliminary Design (20%)

This is an intermediate internal milestone for your team. Your instructor may supply verbal or written feedback. The goal is to create a draft using the provided design document format the captures your initial design sketch. Use the provided example application (Thermometer) as a guide.

Tag in your repository as **V1.0-Initial**

Skeleton Java Implementation (10%)

This is an intermediate internal milestone for your team. Your instructor may supply verbal or written feedback. The goal is to get the construction (coding) process underway as a means of verifying your initial design. Minimally you should have Java skeletons for all classes and interfaces you have identified in your design. A suggestion for starting to implement your actual methods may be to select a specific scenario (for example: "user adds a new food") and follow that through to completion. This deliverable will also help the team to gauge the amount of effort required to complete the construction process for R1. Note that your initial design may change as a result of the implementation phase.

Tag in your repository as **V1.0-Skeleton**

Version 1.0 Team Presentation (10%)

Based on class time availability, all teams should be prepared to give a brief (5-10 minutes) "informal" yet professional presentation of their project on the date(s) specified in the schedule. All teammates must participate. The presentation should address the team's approach to the design process and rationale for the decisions reached. Use your diagrams to justify and support your claims (Make sure they are legible when projected!). You should also summarize the team's dynamic, challenges and successes. Teams should prepare to present a **short demo** on the features you completed. Be prepared to answer any questions from your peers and provide any disclaimers about the limitations of your functionality. Update the **README** file in your repo to reflect the execution instructions and any disclaimers/shortcomings/known bugs.

Please ensure your slides are legible on the projector and that all team members come prepared.

Version 1.0 Design (30%)

Complete design document. Follow the instructions embedded within the document and use the example application as a guide.

Version 1.0 Java Implementation (20%)

Although your team may be tempted to complete only a small portion of the overall known features, you should account for the risk of known upcoming requirement additions. Mitigate this risk by implementing higher value (more complex) functionality first to avoid back-loading your project. Your submission will be evaluated both on its quality as well as the compliance with the design.

Submit your Final Version 1.0 Design document, Presentation file (.pptx) and Java code in your repository as **V1.0-Final**. Also complete the corresponding submission in myCourses.