CSE 403, Project Phase 2b: SDS (50 points)
Software Design Specification (SDS)

The SDS milestone is a set of documents about your project's design. Your design should specify how to implement an object-oriented product to meet the requirements in your SRS. Among other things, your SDS should answer the questions: what are your classes, what are the responsibilities of each class, and how do the classes collaborate?

Your SDS must contain items from the following three (3) major categories:

1. UML class diagram

Submit a **UML class diagram** for your system in the format shown in Fowler, Chapter 3. Your diagram should display all major classes, attributes (fields), methods (do not list `get` / `set` / `is` methods), inheritance/interface relationships, and associational relationships (named and directed, with multiplicity adornments).

Your design will be evaluated on completeness as well as level of thought, attention to principles discussed in class, and proper UML syntax. Follow Riel's OO design heuristics, such as:

- use encapsulation (Heuristic 2.1)
- keep related data and behavior in the same place (Heuristic 2.9)
- minimize each class's public interface (Heuristic 2.3, 2.6)
- emphasize cohesion and limit coupling (Heuristic 2.7, 2.8)
- avoid "god classes" (Heuristic 3.2)
- avoid insignificant or irrelevant classes (Heuristic 2.11, 3.7, 3.8)
- model-view separation and model independence from view (Heuristic 3.5)
- avoid irrelevant "agent" or "controller" classes (Heuristic 3.10)

Distribute your project's functionality and allow for features to be developed in parallel as much as possible.

If your project is an Android mobile app, please include activities and views in your diagram, but you do not need to list their fields or methods. If your project is a web application built using a MVC framework such as Ruby on Rails, include model / domain classes and controllers in your class diagram, but you do not need to include view classes. You do not need to explicitly draw web pages themselves, but if pages are backed by various stateful classes, do include those.

2. UML sequence diagrams (with optional state diagram)

Submit two **(2) UML sequence diagrams** that depict your product executing two of its important use cases. These can be the same use cases you wrote about in your SRS. The sequence diagrams should follow the format of the examples from Fowler, Chapter 4. Your diagram should show all participants (objects) in the sequence, all important directed messages between them and their return values (if any), as well as interaction frames with proper operator adornments as appropriate. Use good design with decentralized control; no one class or object should do the bulk of the work.

If your app is a mobile app, show the entire path through from the user's initial action through the UI down to any lower-level code such as data structures and objects that model underlying system state and behavior. If your product is a web app, the sequence diagrams should show the "life" of a user's web request. Show the request's path through your UI, server, and/or data layers as it interacts with each to accomplish the task.

Accompanying at least one of the sequence diagrams should be a brief **pseudo-code description** of the same algorithm or process, similar to Fowler's Figure 4.4.

As an alternative, if you want to turn in **one (1) UML state diagram** in place of one of your sequence diagrams, this is okay. The state diagram should follow proper state diagram syntax as described in Fowler, Chapter 10. Your diagram should include: a starting pseudo-state; at least 3-4 other meaningful states, properly labeled; transitions between those states as appropriate, properly labeled; and a final state, properly labeled, as appropriate.

3. Coding style guidelines

Submit a **coding standards document** (this can be a page on your project's wiki, if you like) explaining what style conventions you plan to follow. A reference document or link with an example would be helpful. Also explain how you plan to enforce a consistent coding style between group members. Describe any tools you plan to use to enforce these conventions, and/or any methodologies your group members plan to use to enforce them.

Your group will be required to perform significant **code reviews**, as described in the Alpha spec. Plan to set aside time to perform these code reviews, where one developer's work will be looked over by one or more others. In your coding standards document, briefly describe how your code reviews will be done and how you will provide evidence of these reviews to the grader. Will you take review notes by hand? Will you use tools built into the version control system to annotate the reviewed code? Once the review is complete, what actions are taken to correct the issues and submit? Etc.

See the course web site's Links page for links to some useful tools for style checking that you may want to list here.

Submission and Grading:

Submit your SDS documents online by checking them into your version control system on **GitHub** by the due date. You may submit documents on paper in lecture as well if you like; this means you must have those documents finished by the start of class time on the next lecture on-or-after the due date in order to hand them in on time. If you submit the SDS by checking in your diagrams on GitHub, please check them in using a **file format** that can be easily opened by the customer TA without any special software. For example, if you create your diagrams using the Violet UML tool, it generates documents in the .violet file format; but also **export them to .PNG or .GIF images** or PDF and check in the exported version to your version control area as well, so the TA can open them.

Part of your grade will come from the plausibility, thoughtfulness, and level of detail of your work. For example, if you are listing classes in your class diagram, take care not to forget important classes that would reasonably needed to solve the task you are working on. Do not forget to include classes for all aspects of your system, such as user interface, data modeling, database interaction, any "helper" classes or utility code, etc. In the past, some groups have focused too much on user interface classes and have done a poor job capturing the details of the "model" of your system, the classes that actually represent the important data and behavior necessary for the task.

List the contents of each class in detail, including a comprehensive set of fields, methods, and constructors. (Do not list "getter" or "setter" methods that simply provide clients access to data of the object.) Part of your grade comes from following proper UML syntax, including syntax for classes, interfaces, attributes, methods, access modifiers, parameters, return values, and types. Also list any and all relationships that occur between classes. Your relationships should have proper syntax for all elements mentioned in the reading, such as direction, multiplicity, and a label explaining the relationship.

In UML sequence diagrams, we want to see that you have listed a reasonable set of steps and calls to achieve a concrete goal described in your use cases from your SRS. Take care not to omit important steps or details. Make sure that the state of the system at the end of the path through the diagram matches the expectations from the use case to which it relates. Follow proper sequence diagram syntax for objects, interaction frame bars, messages between objects, and parameters. Choose a flow of control that is reasonable for your type of project and the goal being achieved in the scenario. Ideally no one class or object should do too large a share of the work to complete the scenario.

A small part of your grade comes from the looks or aesthetics of your documents. They do not need to be beautiful or excessively formatted, but your customers need to be able to read them and extract information from them. This means they should be clearly written, with proper spelling and grammar, clear wording, and drawn and formatted with a enough organization to present your ideas clearly to the reader.

Remember that part of your grade comes from having a meaningful in-person **interaction with your customer** before the phase is due to show your progress, ask questions, get feedback, and generally make sure you are on the right track.