**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 two major categories:

# UML class diagram

Submit a **UML class diagram** for your system in the format shown in the sample SDS. 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 OO design heuristics, such as:

* keep related data and behavior in the same place
* minimize each class's public interface
* avoid "god classes"
* avoid insignificant or irrelevant classes
* model-view separation
* avoid irrelevant "controller" classes

If your project is a J2ME mobile app, please include views in your diagram, but you do not need to list their fields or methods. **If your project is a web application, 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 (for example a when a professor wants to select a course to teach this page will need to get the data of the courses being offered.)

# UML sequence diagrams

Submit two **UML sequence diagrams** that depict your product executing your use cases. The sequence diagrams should follow the format of the examples on your sample SDS. Your diagram should show all participants (objects) in the sequence, all important directed messages between them along with their arguments and return values (if any). 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.