Session 6: Using UML in Object-Oriented Code Design

The following is a simplified summary of the process of going from requirements to object-oriented design, using UML. The following steps are not always performed in this order, and are often iterated many times as the design is refined.

1. Perform a use case analysis to determine product requirements

2. Identify the conceptual (domain) classes that participate in your use cases. These are real-world entities that would be involved in performing the tasks of your use case if it were performed manually, such as customer, database, cash register, statement, etc. Often the secondary actors give rise to classes that are needed to interface with the external entity.

3. Identify any software classes (not real-world) that would help organize the tasks to be performed. These computer software classes may have no real-world counterpart. For example, you might include a session manager class to supervise the user interaction, or you may need classes to perform data manipulations.

4. Assign responsibilities to classes. Think about how to assign responsibilities in such a way as to simplify your design. You will need to determine what classes have the responsibility of creating and destroying other classes. Make use of inheritance whenever appropriate.

5. Construct interaction diagrams (sequence diagrams or collaboration diagrams) for your use cases. You may find that it is useful to add additional classes to perform the use cases.

6. Construct a class diagram using the classes you have identified. Include all the associations that are needed for your interaction diagrams to work. Supply any operations and attributes needed to implement your interaction diagrams.

7. Construct state and/or activity diagrams as needed to work out the details of any complex behaviors.

8. Look at your design for possible improvements to the object-oriented design, for example making more use of inheritance, and changing class responsibilities to decrease
coupling or increase cohesion. Modify your UML diagrams whenever you add new capabilities that were not in the original diagrams.

9. Generate class definitions (i.e. header files) for the classes you have defined.

10. Implement the methods needed, again making changes to your UML diagrams as you identify new operations and interactions that were not previously in the diagrams.