SWEN30006 Software Modelling and Design
Workshop 3: Design Models and Coding – Partial Solution

School of Computing and Information Systems
University of Melbourne
Semester 2, 2020

Disclaimer: The solution provided in this document is one of many possible solutions. Your solution may also be correct or suitable even though it looks different from the solution in this document. This is because the solution can be developed based on different reasoning and design decisions. In addition, the solution in this document may be *incomplete* for the purpose of learning. We encourage you to discuss with the tutor during the workshop hours to check the correctness of your solution.

Part 1 From Domain to Design Class Diagrams

Your answer to this question will depend on the solution you came up with in the previous workshop. For reference, Figure 1 shows one possible solution, based on the domain model solution from last week's workshop. Some things that are important to check in your solution are:

- Use of visibility modifiers
- Use of return types
- UML syntax
- Consistency with your domain model
- Appropriate allocation of responsibilities

As with most modelling, there is room for variation in answers. So, the important thing is that you are able to justify and clearly communicate the choices you have made in your solution.

1. A low representational gap means that a developer's mental model of a domain more closely matches the software implementation. This increases the ability for the developer to understand, hence modify, the system. It also helps match requirements from the domain with their implementation.
2. Your answer to this question will depend on your specific design and domain models.

Part 2 Creating a Dynamic Domain Model

Again, there isn't a single correct answer for this question. As such, the important thing is to be able to explain and justify your choices. Your ability to do this should be shown through your answers to the two discussion questions. This is a key skill for the first project!

One thing you should make sure of is that your design sequence diagram is consistent with your design class diagram and system sequence diagrams. That is, methods should belong to the same classes, use the same parameters, and have the same result types in all of your diagrams.

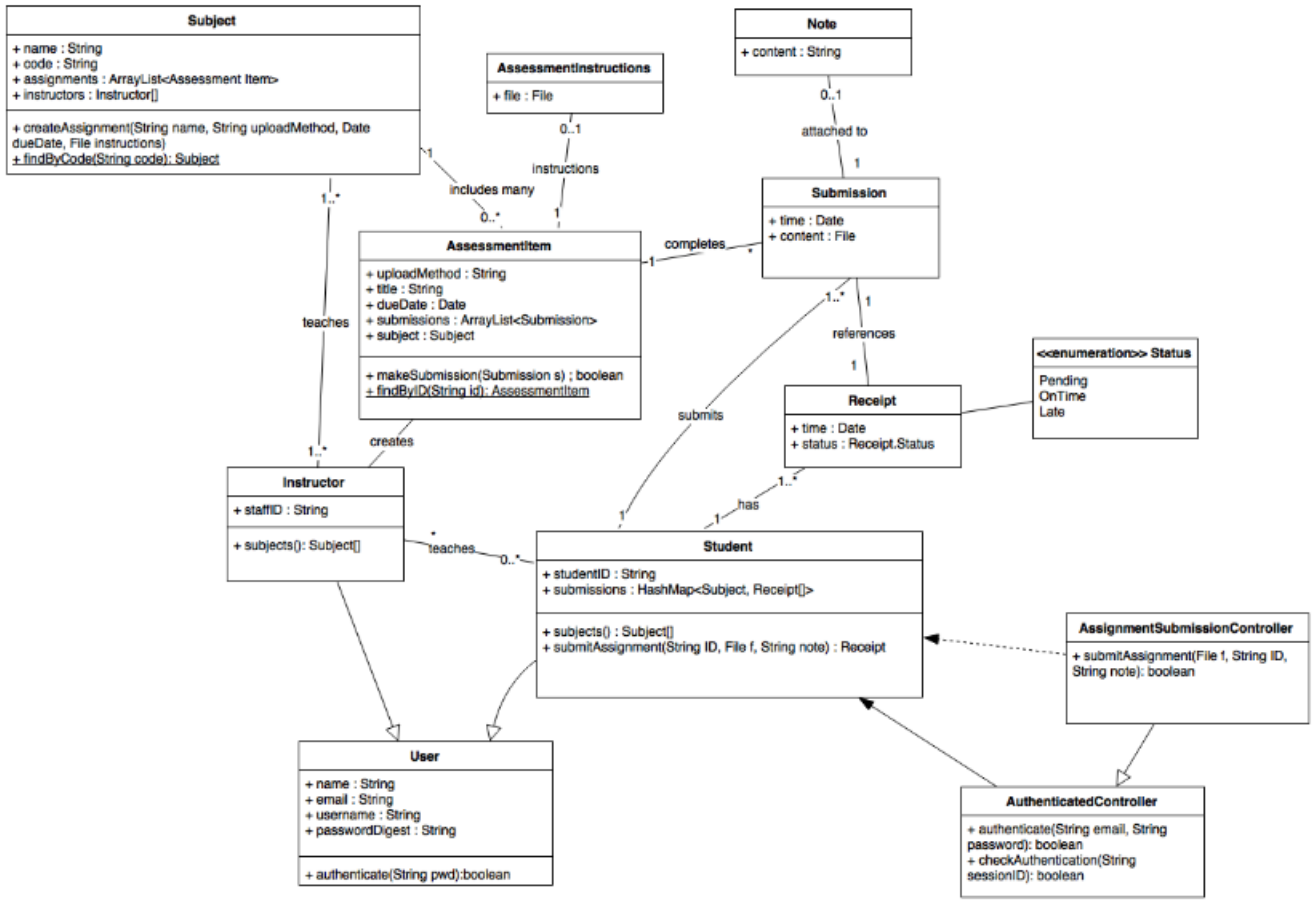

Figure 1: A potential design class diagram solution

Part 3 From Design to Implementation

The key thing you should have noticed when completing this task is that you had to make a lot of assumptions. This is because there is not enough information in the static domain and design models to really understand how the system should work. Dynamic models, which we intentionally did not provide, would have helped you overcome this difficulty. Because of this, your implementation may differ a lot from those completed by your peers.