Unit 5

Understanding UML (Practical)

Practical Activity - Creating a class diagram

Summary

I enjoyed this unit (once again) because it challenged me about existing UML modelling skills; caused me to think *why* (or not) to use this or that relationship. For this activity, I focused on the use of **Association Classes** and how best they can be leveraged as part of a class diagram. Of tremendous benefit was the reading material by Martin Fowler in his book titled UML Distilled. While I had a basic understanding of UML and various relationships, collaborating with fellow students and reading from the UML distilled has improved my confidence level in modelling correct UML object and class diagrams.

Another book referenced was The Elements of UML 2.0 Style by Scott Ambler, which I found pretty informative. However, I did not go through all the book's content because I already have been exposed to these tremendously valid and beneficial considerations when developing readable, clean, and consistent UML models.

Overall, the *practical* aspect of this unit I found greatly beneficial. Beneficial because I feel I have improved my understanding of UML and learned to give more profound attention to building models that matter.

Diagram Description

The UML diagram presented is my attempt at representing a fictitious supermarket using class diagrams. I based this diagram on Unit 3, wherein I built a conceptual UML diagram and proved it with a few object diagrams. To arrive at this class diagram iteration, I walked through each scenario from Unit 3 and applied the design as though I were walking into a proper supermarket. I tested the assertions for why Association is preferred over Composition or

Dependency—Aggregation was discarded because of the logical argument made by Martin Fowler in his book, UML Distilled.

Final Class Diagram

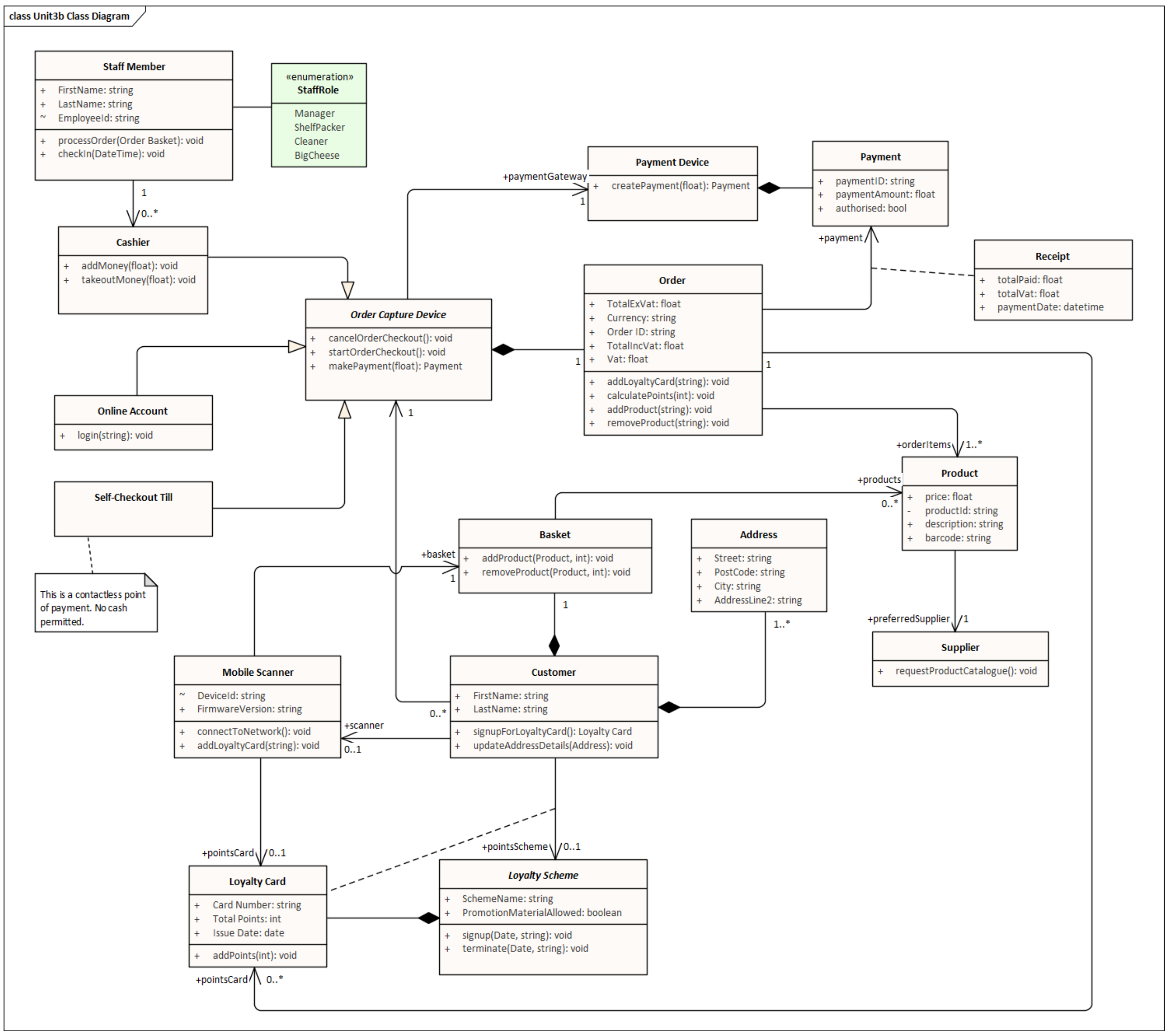
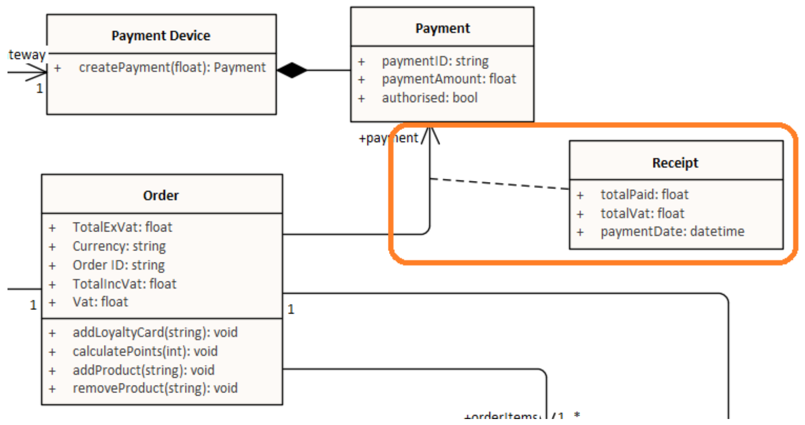
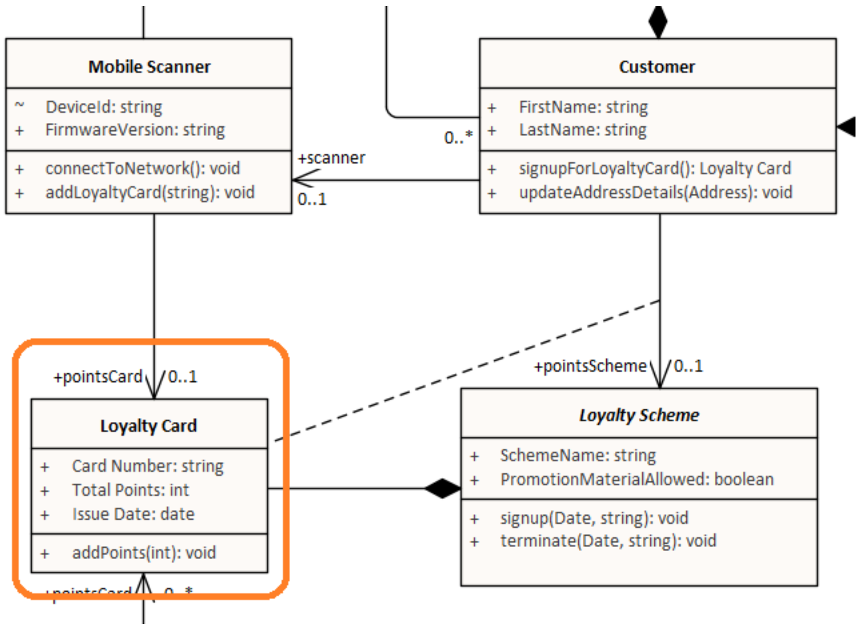


Figure 1 Class diagram extended from Unit 3 (created by the author)

In this unit, I utilised association classes to represent data that is required when one class has a reference to another class, as shown in the following snippets:



Here the thinking is that a Receipt is the *output* of a Payment and is associated with a specific Order instance. Thus, it follows what I consider to be a real-world example is that a Receipt cannot exist *unless* an Order has been Paid (Payment).



In this scenario of the class diagram, I modelled that a Loyalty Card means nothing outside the context of a Loyalty Scheme. I found this scenario slightly challenging because of the Composition relationship to a Loyalty Card. What is meant to be modelled is that a Loyalty Card can only exist when a customer becomes a scheme member. The Customer does not cause the Loyalty Card to exist, but rather, their *Signup* action triggers the Loyalty Card to be printed and sent to the Customer. In this sense, the Loyalty Scheme owns the Loyalty Card (Composition), but the output from *Signup* is to associate it with a Customer.