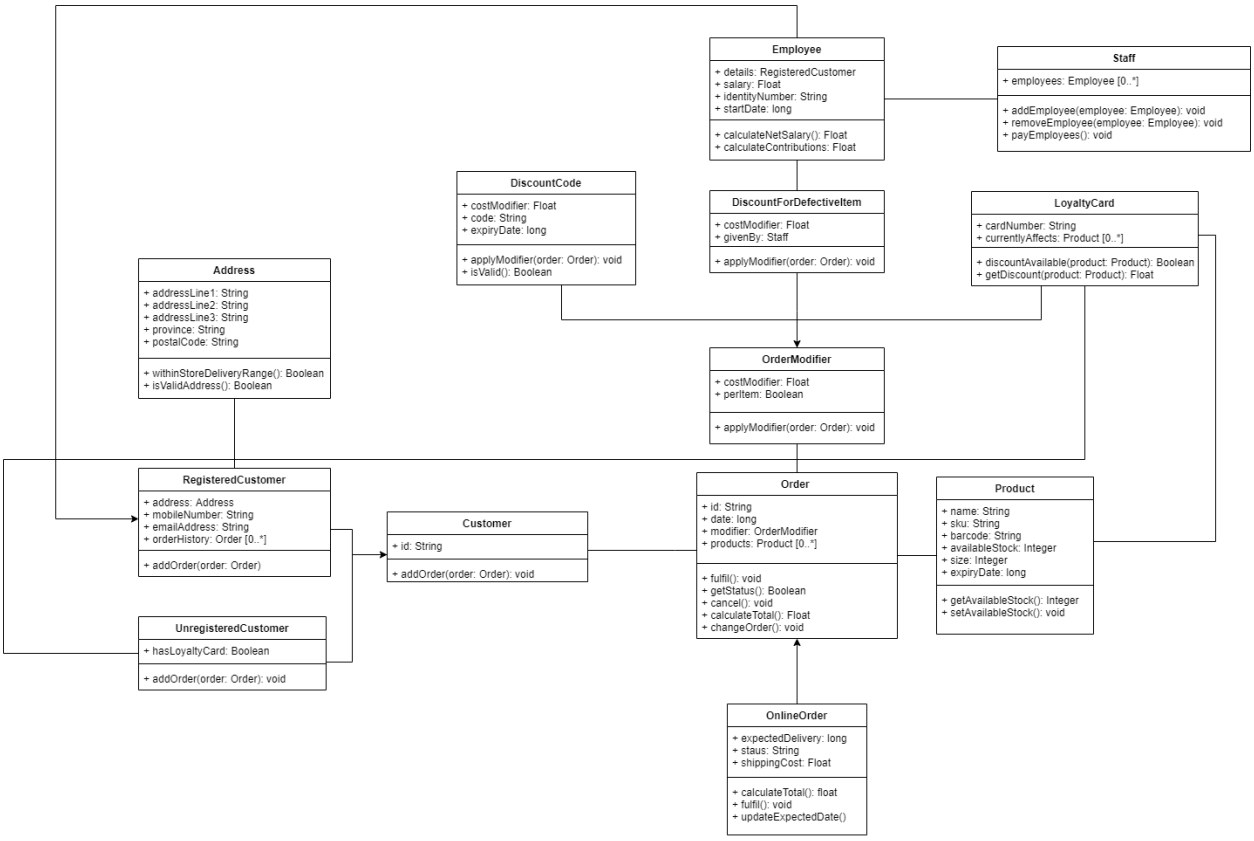
Post: Shan Swanlow

Here is my attempt at a class diagram:



For me, the tricky part of this exercise is that most modern supermarkets support highly flexible workflows and I wanted to try making the classes in the same way. For example, a supermarket in my area allows you to use a loyalty card without signing up, but if you choose, you can sign up with a mobile number, name, and link it to your card. For dates, I've left them as long because from experience, it's much easier to store dates in Unix format and process them in that way.

Another tricky part to write the classes in such a way that the Liskov Substitution Principle (LSP) holds. Often encouraged as a good practice, the concept of the LSP is that when you create objects that inherit from a parent class, you should be able to swap those subclasses with the parent class, without causing the application to behave unexpectedly (Reflectoring, N.D.).

References:

Reflectoring. (N.D.) The Liskov Substitution Principle Explained. Available from: https://reflectoring.io/lsp-explained/ [Accessed 24 May 2021].

Tutor Feedback: Beran Necat

Hi Shan,

Good attempt. The next step is to revisit the diagram and identifying the key relationships between classes. Please consider ASSOCIATION which tells us classes refer to each other for example order and payment. Also consider GENERALIZATION to express that one model element is based on another model element.

Regards, Beran