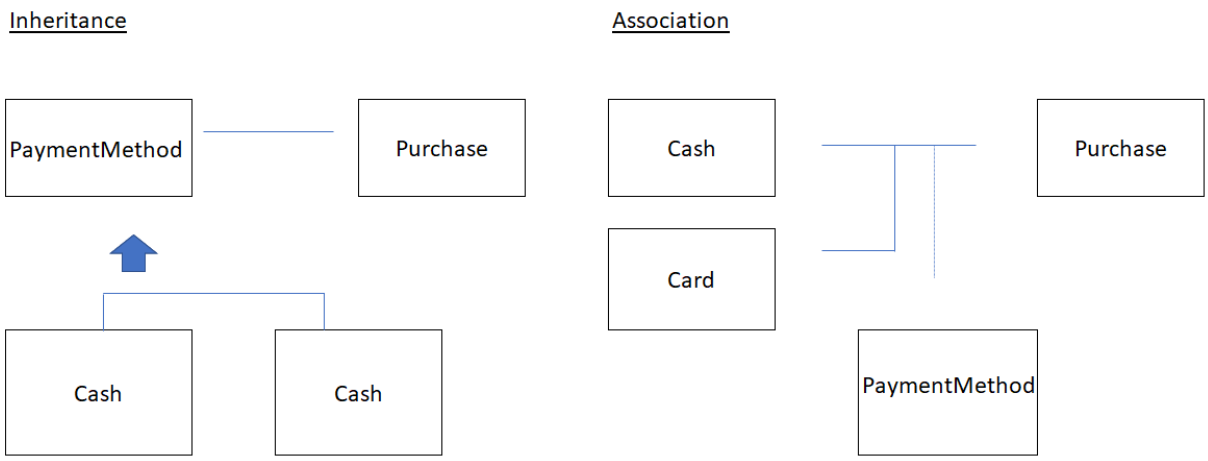
Inheritance vs Association

Regarding the 1. Assignment which asked for the implementation of a self-service checkout in a supermarket, I was thinking about the implementation of a payment method class which should essentially act as an intermediary between a purchase class and two classes, cash and card. Loosely speaking, every purchase in such a self -service system requires the choice of one or more payment methods where payment methods itself can be divided into cash and/or card payments. The latter incorporates aside from bank card also loyalty points, voucher codes, ApplePay and AndroidPay. In thinking about the relationships between these classes, I was specifically focused on the connection between purchase method and the two classes, cash and card. The first impression led me to the implementation of an inheritance relationship where the payment method class represents a mother function (or superclass as it is called in Computer Science). However, I then read about an association class which indeed can be seen as an intermediary in a three party relationship. An example here can be that students attend to courses but they first have to register for them (Registration class in an intermediary in this case). Hence, taking the case of the supermarket, it seems that an association class would not represent the relationships in the best way. As described in (IBM, 2021) an association class “further defines” the relationship between two other classes. In this sense, in the payment method example it indeed seems to be a bridging point between two classes. On the other side, inheritance does not really lay between two classes rather than to connect it to one superclass more dependently (the subclasses inherit all the attributes and all the methods from it). In other words, because of this dependency or “inheritance”, it is important that the PaymentMethod class doesn’t disappear without thinking carefully about the future structure of the system. And this is exactly what we want, if the PaymentMethod class disappears, it is important to

think in a new way about all the connections. Hence, inheritance ensures this in the way that your system simply collapses when a superclass of an inheritance relationship isn't working anymore. It let you rethink about the system design and where to use the right attributes and methods. In association (because of a less dependent relationship), the system doesn't collapse directly and you can still work with the existing one in simply connecting other classes with one another (for example students will be directly connected to courses without the existence of a course enrolment class). At the end it is also a matter of choice, what the designer wants. In the example of the purchase method class, the idea was to show stronger dependency in terms of an inheritance relationship.



Reference List:

IBM (2021) Association Classes. Available from:

https://www.ibm.com/docs/en/rsm/7.5.0?topic=diagrams-association-classes