**Object description**

The set-out purpose is to produce a class and object model for a self-scanning checkout system for a supermarket.

**Encapsulation and abstraction**

Classes are encapsulated, as in data and functions are within one unit that is a class. Items can be added via barcode or weight and can be easiest implemented by implementing two methods in the cart table. Initially, each item was represented by inheritance, however, the easy way is to implement an additional method in the cart table for each input function.

**Inheritance and polymorphism  
Payment method**

For payment methods, inheritance is the suitable relationship as each payment type is a specialisation of the abstract class, ‘Payment’. If other relationships are implemented, it means that data in ‘Payment’ would be replicated in each payment method table. Additionally, it saves workload if the supermarket wants to implement or remove types of payment; for example, make cryptocurrency an available payment type.

**Composition  
Payment and Order**

After payment is completed, the order is validated, and a notification is sent to the inventory department to inform of items that need restocking.

**Cart and Order**

Similarly, a cart can exist without an order, but an order is dependent on a cart existing specifying the content of an order. Instead of creating a single order table, the cart table is necessary because it can easily be modified and allows the customer to change the cart before purchase.

**Loyalty Payment and Member**

A loyalty card is owned by a member. A member is not called a customer because one can shop at the supermarket without having a loyalty card, but one must be a member to hold a loyalty card. A loyalty card is dependent on a member owning it. There is not a customer table as there is no requirement that a person logs in or verifies their identity before starting the scan. Had the supermarket however chosen to implement mobile self-scanners, the ‘members’ table would replace the customer stick man as verification would be necessary to start mobile scanning items to collect points, access discounts, and verify users.

**Aggregation and Association**

Aggregation has not been used in the class diagram; because aggregation and composition are kinds of plain association, efforts have been made to simplify and reduce the use to what is strictly necessary for the informal understanding of the class diagram.  When an aggregation or composition is correct, so is plain association. (Fowler, 2003 75). The admin table has associations with the ‘Cart’ and ‘Order’ table as a staff member may need access to verify the age of the consumer or edit/update an order.

The ‘Scanee’ table refers to the check-out machine and provides a time limit of acceptable use. For example, if the scannee is inactive for a period of five minutes, it resets.

**Table of Authorities**

 Fowler M. (2003) *UML Distilled: A Brief Guide to the Standard Object Modeling Language.* 3rd ed. Boston, MA: Addison Wesley