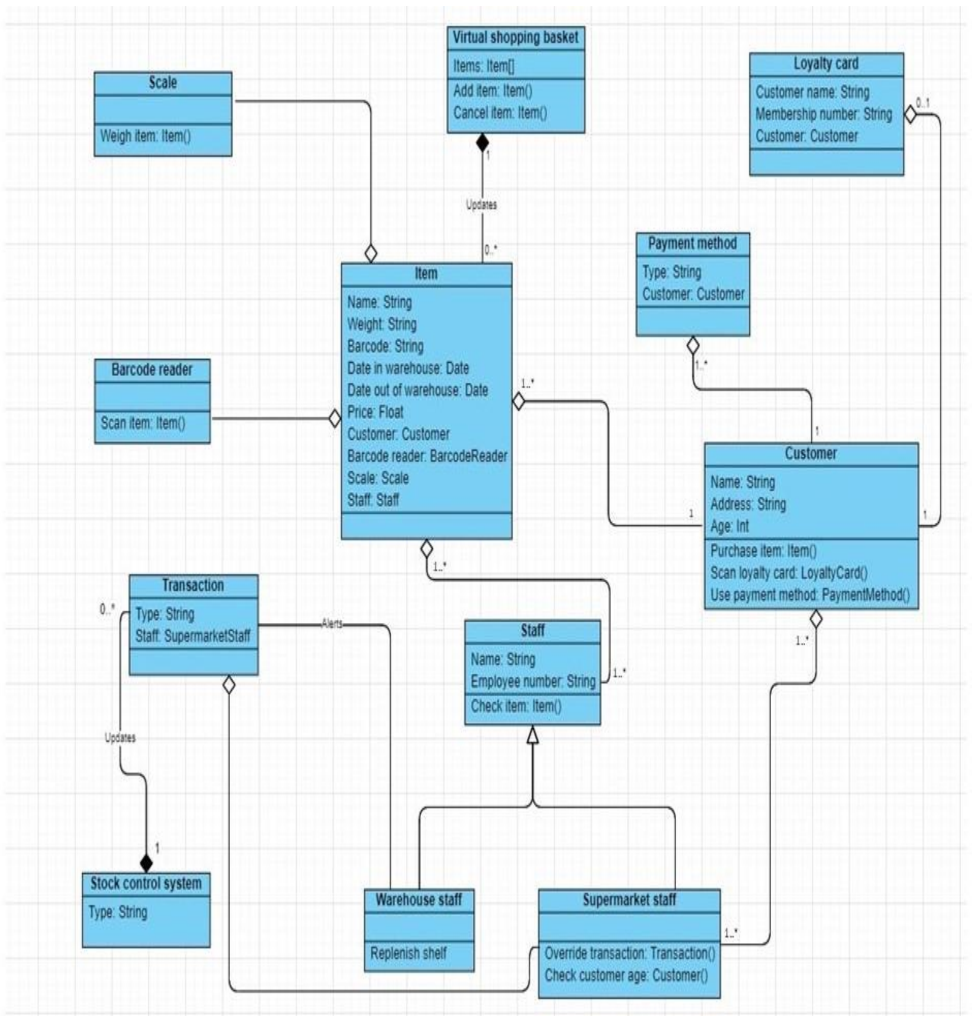
Mid - Module Assignment: System Design

In system design, the focus is first on choosing classes which are the building blocks in a system and then on creating relationships between classes.



UML Class Diagram of a self-service checkout

Here, based on a provided self-service checkout scenario, a software system has been designed. Going through the scenario and highlighting the nouns and noun phrases is a method that can be used for finding classes. Considering this method, **candidate classes** are Item, Virtual shopping basket, Barcode reader, Scale, Customer, Loyalty card, Payment method, Staff (Supermarket & Warehouse), Transaction, Stock control system.

Item is the key component which other classes are related to directly or indirectly. Some classes including *Customer*, *Scale*, *Barcode reader* and *Staff* are directly related to Item; *Scale*s weigh the *item*s, *Barcode reader*s scan the *item*s, *Customers* purchase the *item*s, *Staff* check the *item*s. Adding or removing the items updates another class called *Virtual shopping basket*.

Customer is a class which is related to two other classes - *Loyalty card* and *Payment method*: *Customer*s scan *Loyalty card*s and use different *Payment method*s.

Staff is another class which is the parent of two classes - *Warehouse staff* and *Supermarket staff*. These children are related to *Transaction* class; *Supermarket staff* override the *transaction*s which update *Stock control system* class.

Considering candidate classes, their **relationships** are as follows:

A customer purchases item.

A staff checks item.

A scale weighs item.

A barcode reader scans items.

A customer can scan loyalty cards.

A customer uses payment methods.

An item is added to a virtual shopping basket.

An item removed from a virtual shopping basket.

An item updates a virtual shopping basket.

A supermarket staff can override transactions.

A supermarket staff can check customers.

A transaction alerts warehouse staff.

A transaction updates stock control system.

Considering above descriptions, different relationships can be created between candidate classes:

- Inheritance relationships:
  “It is the most famous, well-known, and over-used relationship in object-oriented programming” (Philips, 2018). In this relationship, children classes inherit attributes and methods from their parent class. Here, *Supermarket staff* and *Warehouse staff* inherit some attributes and methods from *Staff*. They also can have their own attributes and methods; for example, *Supermarket staff* have two methods which only belongs to themselves. “Inheritance is symbolized with a straight connected line with a closed arrowhead pointing towards the parent class” (Lucidchart, 2022).
- Association relationships:
  It is a relationship that shows two classes are connected in any way. As an example, *Transaction* and *Warehouse staff* classes are related together and this relationship is shown by a solid line.
- Aggregation relationships:
  Aggregation is a special type of association between classes where one class (the whole) is composed of other classes (the parts) (Sommerville, 2016). The

existence of the parts is not dependent on the existence of the whole. To show aggregation, a diamond shape is added to the class which acts the whole. As an example, *Item* class is composed of *Customer*, *Staff*, *Barcode reader* and *Scale*.

- Composition relationships:

  Composition is another type of association between classes where the existence of the parts is dependent on the existence of the whole. A black-filled diamond shape is used to represent composition relationship. As an example, the relationship between *Transaction* and *Stock control system* classes is a typical composition.

Finally, to complete the relationships between classes, multiplicity indicators are used (Ambler, 2003). As an example, 1..* (one or many) objects of *Item* class update 1 object of *Virtual shopping basket*.

References

- N.D. (2019). UML Class Diagram Tutorial. Available from: https://www.lucidchart.com/pages/uml-class-diagram/#section_0 [Accessed 17 April 2022]
- Philips, D. (2018). Python 3 Object-Oriented Programming. 3rd ed. Packt Publishing.
- Sommerville, I. (2016). Software Engineering. 10th ed. Essex: Pearson Education.
- Ambler, I. (2003). The Elements of UML Style. 1st ed. Cambridge University Press.