System design report

The system design proposed to manage a self-service checkouts of an high street supermarket chain has 15 **classes**.

The superclass **Staff** has the attributes necessary for identifying staff members. These are inherited by **Supermarket Staff**, who check the customer age when a restricted product is scanned and override wrong transactions, and by **Warehouse Staff**, who replenish shelves or order new stock.

The superclass **Payment Methods** passes the attribute *amount to pay* to the six subclasses representing different payment methods: **Cash, Card, ApplePay, AndroidPay, Voucher, Loyalty Card**. Each subclass has its own attributes and operations necessary to perform payments.

Self-service Checkout has a barcode reader and scale. It reads product and loyalty card barcodes, weigh products and adds them to a virtual basket.

Virtual Basket has a list of scanned products and calculates the total price, takes payments, updates the stock control system and sends alerts when a restricted product is scanned.

Stock Control System has a list of products and the minimum and maximum amount of products on shelves and in the warehouse. It alerts warehouse staff when the minimum amounts are reached.

Product has the attributes for identifying the products in the designed system.

Customer attributes include *age* (necessary to purchase restricted products) and a shopping basket with the selected products. Moreover, the customer might own a loyalty card. The class allows the customer to scan products and the loyalty card, and

make purchases.

The class diagram presents the following relationships between classes:

- **Inheritance**: represented by an unfilled arrowhead leading from the subclass to the superclass (Anon, 2022).

  It is applied between the superclass Staff and its subclasses Supermarket Staff and Warehouse Staff which inherit the superclass attributes. Inheritance is also applied between the superclass Payment Method and its six subclasses Cash, Card, ApplePay, AndroidPay, Voucher, Loyalty Card which inherit *amount to pay*.
- **Composition**: represented by a filled diamond head leading from the child to the parent class. Destroying the parent class destroys the child class (Anon, 2022).

  Composition is applied between Self-service Checkout and Virtual Basket (checkouts always have virtual baskets) and Virtual Basket and Payment Methods (virtual baskets always require a payment method).
- **Aggregation**: represented by an unfilled diamond head leading from the child to the parent class. The child class is not strongly dependent on the parent class so if a parent class is destroyed, the child class continues to exist (Anon, 2022).

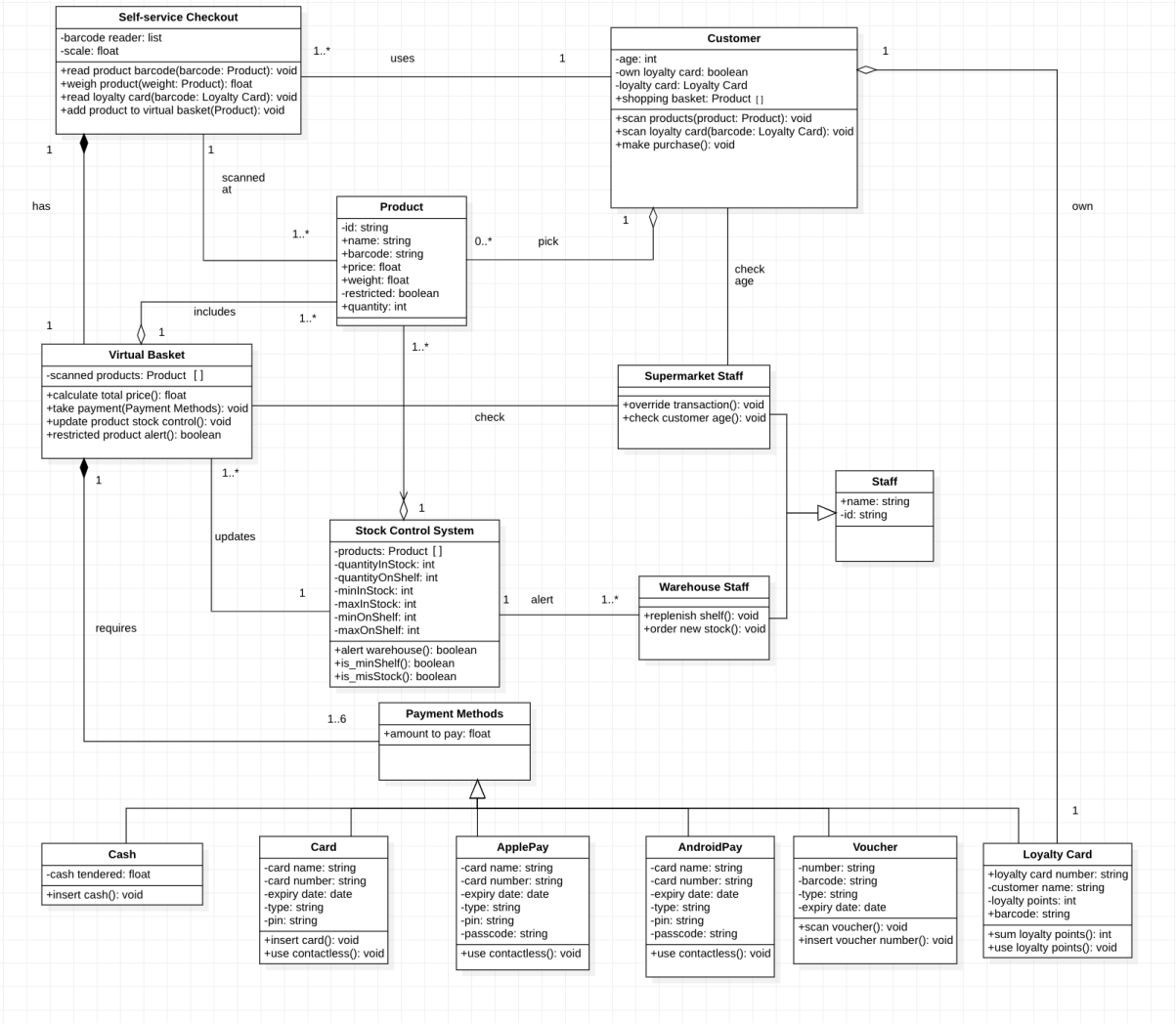
  Aggregation is applied between Customer and Product (customers can pick products or not), Customer and Loyalty Card (customers may or may not own a loyalty card), Virtual Basket and Product (products are included in the virtual basket if scanned), Stock Control System and Product (the stock control

system should have the product details).

- **Association**: represented by a line connecting two classes.

  It is applied between Customer and Self-service Checkout (customers use the self-service checkouts), Supermarket Staff and Virtual Basket (staff check the basket to override wrong transactions), Supermarket Staff and Customer (staff check the customer age if a restricted product is scanned), Virtual Basket and Stock Control System (virtual baskets share the list of purchased products to update the stock control), Stock Control System and Warehouse Staff (the stock control system sends alerts to staff requiring to replenish shelves or order new stock).

Supermarket self-service checkout class diagram



REFERENCES

Anon. (2022) Understanding UML. Available from: https://www.my-course.co.uk/Computing/Computer%20Science/OOIS/OOIS%20Lecturecast%203/content/index.html#/lessons/S8SISrcH5u1sjQDjxAyjA9O7t6SOCmKH [Accessed 23 April 2022].

Anon. (2022) Understanding UML. Available from: https://www.my-course.co.uk/Computing/Computer%20Science/OOIS/OOIS%20Lecturecast%203/content/index.html#/lessons/_sgpGK_0ngvG5sl2F10nHVTpNn25WmGt [Accessed 23 April 2022].

Anon. (2022) Understanding UML. Available from: https://www.my-course.co.uk/Computing/Computer%20Science/OOIS/OOIS%20Lecturecast%203/content/index.html#/lessons/S8SISrcH5u1sjQDjxAyjA9O7t6SOCmKH [Accessed 24 April 2022].