Mid-Module Assignment System Design

Selecting Candidate Classes (List of Classes)

I started this part by looking at the scenario and by going through the nouns to see if they fall into groups (Liang, 2003) and then highlighting these.

When an **item** is scanned, it is added to the **customers'** purchases in a **virtual shopping basket**.

Items can either be scanned with the **barcode reader** or they can be **weighed** using the **integrated scales**.

When the customer has finished scanning their items, they will be given the option to scan their **loyalty card**, if they have one.

The customer then purchases their items using several different **payment methods** (e.g., cash, card, ApplePay, AndroidPay, vouchers, loyalty points).

Supermarket staff will have the ability to **override transactions** where required, for example if a **price** has been scanned incorrectly, they will also be responsible for checking the **age** of a customer for restricted items.

A **completed transaction** will update the supermarkets **stock control systems** and generate an **alert** for **warehouse staff** if either the **shelves should be replenished** or if **stock needs to be reordered**.

I then separated out the candidate classes and decided to shorten some names as this would make the coding element potentially easier.

Candidate Classes 1	Candidate Classes 2
Customer	Customer
• Items	• Items
 Virtual Shopping Basket 	Virtual Shopping Basket - shortened
 Barcode Reader 	to Basket
• Price	Barcode Reader
• Age	 Price – An instance of items
Weight	 Age - An instance of items
 Integrated Scales 	 Weight - An instance of Integrated
Loyalty Card	Scales
 Loyalty Points 	Integrated Scales
 Payment Methods 	 Loyalty Card - An instance of Payment
 Payment 	 Loyalty Points - An instance of Loyalty
Cashcard	Card
 Applepay 	 Payment Methods - An instance of
Androidpay	Payment
Vouchers	Payment
Staff	Cash
Supermarket Staff	Cashcard
- Caponnamor Gran	Applepay

- Completed Transaction
- Supermarket
- Override Transactions
- Stock Control System
- Alert
- Warehouse Staff
- Shelves
- Stock

- Androidpay
- Vouchers
- Staff
- Supermarket Staff An instance of Staff
- Completed Transaction shortened to Transaction
- Supermarket Outside the scope of the system
- Override Transactions An instance of Transactions
- Stock Control System An instance of Staff
- Alert An instance of Stock Control System
- Warehouse Staff An instance of Staff
- Refill Shelves An instance of Stock Control System
- Re-order Stock An instance of Stock Control System

Classes

- Customer
- Items
- Basket
- Barcode Reader
- Integrated Scales
- Payment
- Cash
- Cashcard
- Applepay
- Androidpay
- Vouchers
- Transaction
- Staff
- Stock

List of the Relationships between the Classes

Customer

- A customer can scan many items using the Barcode Reader
- A customer can weigh many items item using the Integrated Scales
- A customer can choose to scan and use their Loyalty Card
- A customer can choose to pay using many different Payment Methods

Staff

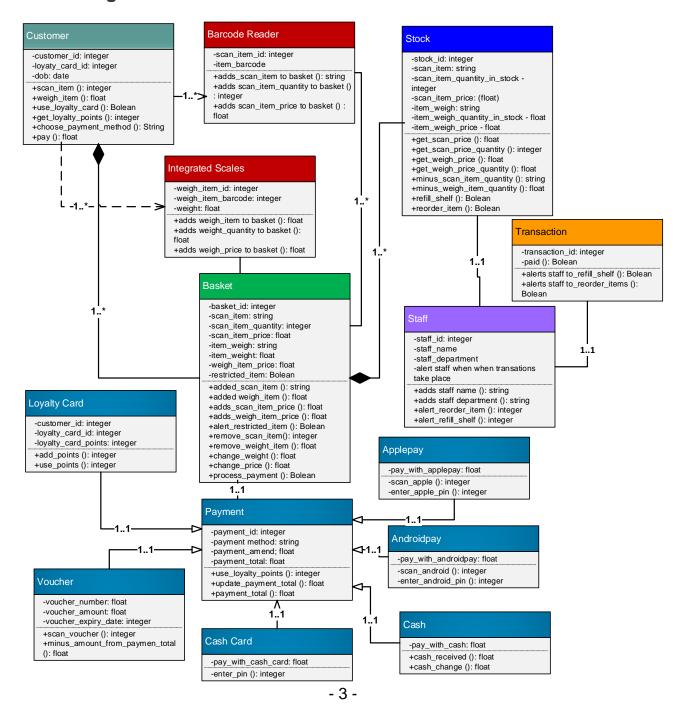
- Supermarket Staff can Override Transactions
 - Supermarket Staff will check the customer's age if the item is restricted then okay to add item or remove item.

- Supermarket Staff will check a price is correct and override it to the correct price if required.
- Supermarket Staff will check an item has been weighed correctly and override it to the correct weight if required.

Warehouse Staff

- Warehouse Staff are alerted after a Completed Transaction when stock needs to be re-ordered. There needs to be a minimum stock level to trigger an alert to re-order stock
- Warehouse Staff are alerted after a Completed Transaction when stock needs to refilled on the shelves. There needs to be a minimum stock level to trigger an alert to refill the shelves.

Class Diagram



Attributes and Operations for Classes and Rationale for the System Design

Customers can choose to pay via the self-service terminal. There is an ID for the customer and loyalty card and their date of birth is required to purchases restricted items.

The customer can scan barcode items and weigh and select an item. They can choose to use their loyalty card by selecting yes or no and pay using different methods.

-customer_id: integer
-loyaty_card_id: integer
-dob: date
+scan_item (): integer
+weigh_item (): float
+use_loyalty_card (): Bolean
+get_loyalty_points (): integer
+choose_payment_method (): String
+pay (): float

The Barcode Reader will scan any item that the customer puts through it.

When scanned the item is identified as well as the quantity and the price, these details are added to the Basket.

Barcode Reader

-scan_item_id: integer
-item_barcode

+adds_scan_item to basket (): string
+adds scan_item_quantity to basket ()
-: integer
+adds scan_item_price to basket (): float

The Integrated Scales weigh relevant items that the customer puts on it.

When weighed, the item is identified and the quantity and the price are added to the Basket.

-weigh_item_id: integer -weigh_item_barcode: integer -weight float +adds weigh_item to basket (): float +adds weight_quantity to basket (): float +adds weigh_orice to basket (): float

The Basket contains items the Customer scans or weighs as well as the quantities and prices.

Supermarket Staff, can access the basket to add, remove items, amend prices and quantities.

When Supermarket staff are alerted of a restricted item they verify the Customers' age and click yes or no if they are 18 or older, if not, the item can be removed.

The Customer can click to process the payment for the items in the Basket, using yes or no.

Basket -basket id: integer -scan_item: string -scan_item_quantity: integer -scan_item_price: float -item_weigh: string -item_weight: float -weigh_item_price: float -restricted_item: Bolean +added_scan_item():string +added weigh_item (): float +adds_scan_item_price (): float +adds_weigh_item_price (): float +alert_restricted_item (): Bolean +remove_scan_item(): integer +remove_weight_item (): float +change weight (): float +change_price (): float +process_payment (): Bolean

When a payment is processed the Customer is given a list payment options. There will be an option to use the Loyalty Card points, this updates the total to be paid.

Payment -payment_id: integer -payment method: string -payment_amend; float -payment_total: float +use_loyalty_points (): integer +update_payment_total (): float +payment_total (): float

A Customer can choose whether to use their Loyalty Card, they can also decide whether to use existing points or had points added.

-cus tomer_id: integer -loy alty_card_id: integer +add_points (): integer +use_points (): integer

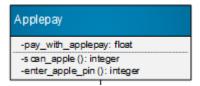
Vouchers can be selected as a Payment Method, this will check the amount and expiry date as well as using the voucher amount to pay for/towards the goods in the Basket.

-voucher_number: float -voucher_amount float -voucher_expiry_date: integer +scan_voucher (): integer +minus_amount_from_paymen_total (): float

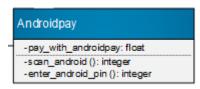
A Cash Card can pay for goods in the Basket, the Customer will be required to enter a pin.

Cash Card -pay_with_cash_card: float -enter_pin (): integer

Applepay can pay for goods in the Basket, the Customer will scan their device and enter a pin.



Androidpay can pay for goods in the Basket, the Customer will scan their device and enter a pin.



Cash can pay for goods in the Basket, the Customer will feed this into the checkout and be given the correct change.

Cash -pay_with_cash: float +cash_received (): float +cash_change (): float

Stock are items scanned or weighed. The system will have the quantities in stock and deduct stock when a transaction is completed. When the quantity gets below a certain level it triggers an alert to the Warehouse Staff to place an order or refill shelves based on the level of stock a shelf holds.

Stock -stock_id: integer -scan_item: string -scan_item_quantity_in_stock integer -scan_item_price: (float) -item_weigh: string -item_weigh_quantity_in_stock - float -item_weigh_price - float +get_scan_price (): float +get_scan_price_quantity():integer +get_weigh_price (): float +get_weigh_price_quantity(): float +minus_scan_item_quantity():string +minus_weigh_item_quantity (): float +refil_shelf (): Bolean +reorder_item (): Bolean

There are two different staff the system impacts; Supermarket and Warehouse Staff, they are identified by Department and access to what they can do in the system is governed by this. Supermarket Staff can override items, quantities, prices and alert them to check restricted items. Warehouse Staff reorder items when prompted and refill shelves.

```
-staff_id: integer
-staff_name
-staff_department
-alert staff when when transations
take place
+adds staff name (): string
+adds staff department (): string
+alert_reorder_item (): integer
+alert_refil_shelf (): integer
```

The completed Transaction triggers an alert to fill shelves or reorder items.

-transaction -transaction_id: integer -paid (): Bolean +alerts staff to_refil_shelf (): Bolean +alerts staff to_reorder_items (): Bolean

References

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

Liang, Y., 2003. From use cases to classes: a way of building object model with UML. Information and Software technology, 45(2), pp.83-93.

Seidle M, Scholz M. Huemer C. and Kappel G. (2014) UML @ Classroom: An Introduction to Object-Oriented Modeling, Springer Nature

Stevens P. (2000) Using UML: Software engineering with objects and components. Harlow, Essex: Addison Wesley