Development Plan

Andrew Johnstone: H00194083 Shayne Shaw: H00214626 Jack Rome: H00219766

Alex Birks: H00214362

Calum Thompson: H00187803

Chosen Collections

- HashSet storing the menu when read in from a txt file.
- TreeSet storing the collection of customers/orders.
- LinkedList storing the items within an order.

Data Structure Justifications

The coffee shop's menu will be created from the text file read in at the beginning of the program. A *Menu* object will then be instantiated and its corresponding hashSet will be filled with the reported items. The *MenuItem* objects will be stored as the 'value' with the 'key' being the item's ID. In doing so, this will allow for easy referencing and searching of the available menu items. Another property of the hashSet implementation of hashTables, that will be beneficial to our application, is that they ensure that only unique entities are added. This helps prevent multiple entries of the same item being added to the menu if there is an error in the text file.

All *Order* objects that have been created during the running of the application will be stored in a TreeSet data structure within the *OrderList* object which. This is created when the text file is read at the start of the application. Each *Order* object will be stored as the 'value' and their corresponding UniqueID will be used as the 'key'. By using the tree implementation, a Set of *Order* objects, sorted using their UniqueIDs, can be returned. Allowing easy access to the *Order* object, either in order of appearance or by searching by a specific ID.

Each *Order* object created will have to store the items that a customer has ordered. A LinkedList will be used to do this. By storing the customer's order as they are taken, a LinkedList will allow for easier and more natural storing and recital. LinkedLists also allow for efficient insertion and deletion. Therefore, if a customer wants to remove an item previously added to their order, this can be done with minimal disruption. Another property of LinkedLists, that makes them suitable for this application, is that the nodes can be dynamically allocated. This means that a limit on the orders doesn't have to be set. They also allow for duplicate items to be added.

Sequence Diagram

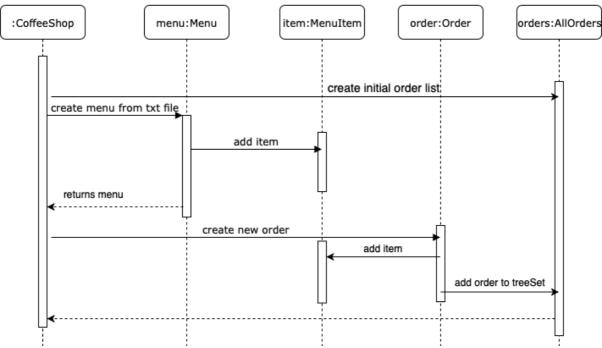


Figure 1: Sequence Diagram operation for importing menu and order files

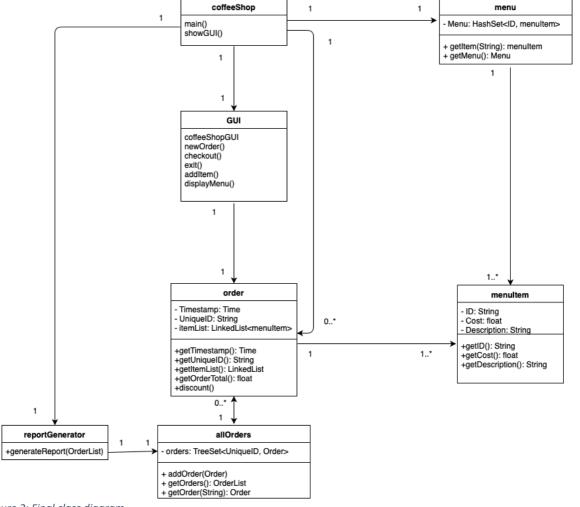


Figure 2: Final class diagram

Gantt Chart

GANTT				January 2019	Fe	February 2019		
Name	Begin date	End date	Assignee	Week 4 21/01/19	Week 5 28/01/19	Week 6 04/02/19	Week 7 11/02/19	Week 8 18/02/19
Initial Planning	22/01/19	31/01/19	ALL	-	[10 Day(s)]			
Initial Project ideas	22/01/19	22/01/19	ALL	[1 Day(s)]				
 Introduction of team members 	26/01/19	26/01/19	ALL	[10	Day(s)]			
Draft CRC & UML	26/01/19	26/01/19	Alex	[10	Day(s)]			
Create initial Gantt Chart	26/01/19	28/01/19	Shayne		[3 Day(s)]			
CRC Diagrams	28/01/19	29/01/19	Calum		[2 Day(s)]			
 Finalize Gantt Chart for report 	29/01/19	31/01/19	Jack		[3 Day(s)]			
Report 1 Meeting	29/01/19	29/01/19	ALL		[1 Day(s)]			
 Finalize UML Diagram 	29/01/19	29/01/19	AJ, Calum		[1 Day(s)]			
Finalize Report	29/01/19	31/01/19			[3 Day(s)]			
° Iteration 1 (Basic implementation - Just drinks)	01/02/19	05/02/19	ALL		,	[5 Day(s)]		
CoffeeShop Class	01/02/19	05/02/19	ALL			[5 Day(s)]		
menultem Class	01/02/19	05/02/19	AJ			[5 Day(s)]		
OrderList Class	01/02/19	05/02/19	Alex			[5 Day(s)]		
Menu Class	01/02/19	05/02/19	Calum			[5 Day(s)]		
Report Class	01/02/19	05/02/19	Calum			[5 Day(s)]		
Order Class	01/02/19	05/02/19	Jack			[5 Day(s)]		
GUI Class	01/02/19	05/02/19	Shayne			[5 Day(s)]		
 Progress Meeting 	04/02/19	04/02/19	ALL			[1 Day(s)]		
° Iteration 2 - Add in food items & Test Iteration 1	05/02/19	09/02/19	ALL			[50	ay(s)]	
° Iteration 3 - Add in "other" items and discounts & Test 2	09/02/19	12/02/19	ALL				[4 Day(s)]	
° Iteration 4 - Add in "extras", i.e. Bacon, extra shot, etc. & Test 3	12/02/19	14/02/19	ALL				[3 Day(s)]	
Final Report, Adjustments & Testing	14/02/19	21/02/19	ALL					