University of Westminster

School of Electronics and Computer Science

4COSC010C.3 Software Development 2 – Coursework	
Module leader	
Weighting:	50% of the module
Qualifying mark	30%
Description	Coursework
Learning Outcomes Covered in this Assignment:	LO1, LO3, LO4, LO5.
Handed Out:	
Due Date	Code due on Blackboard coursework upload
Expected deliverables	a) Zip the project directory of each implementation and upload as w1234557_arrays_only.zip, and w1234567_classes.zip. b) Submit test results (pdf) along with the code. c) Online demo
Method of Submission:	Blackboard
Type of Feedback and Due Date:	Written feedback and marks 15 working days (3 weeks) after the submission deadline. All marks will remain provisional until formally agreed by an Assessment Board.

Assessment regulations

Refer to section 4 of the "How you study" guide for undergraduate students for a clarification of how you are assessed, penalties and late submissions, what constitutes plagiarism etc.

Penalty for Late Submission

If you submit your coursework late but within 24 hours or one working day of the specified deadline, 10 marks will be deducted from the final mark, as a penalty for late submission, except for work which obtains a mark in the range 40 – 49%, in which case the mark will be capped at the pass mark (40%). If you submit your coursework more than 24 hours or more than one working day after the specified deadline you will be given a mark of zero for the work in question unless a claim of Mitigating Circumstances has been submitted and accepted as valid.

It is recognized that on occasion, illness or a personal crisis can mean that you fail to submit a piece of work on time. In such cases you must inform the Campus Office in writing on a mitigating circumstances form, giving the reason for your late or non-submission. You must provide relevant documentary evidence with the form. This information will be reported to the relevant Assessment Board that will decide whether the mark of zero shall stand.

Coursework Description

Fuel Queue Management System.

Task 1. Arrays version.

Design a program for a **Fuel center** which have 3 pumps where maximum 6 customers can be hold simultaneously in a queue. Implement the following functionalities as separate procedures. You can build up your test cases as you develop your program (see testing and Report section below). Fuel Center will have exactly 6600 liters in their stock. For each customer added to the queue, stock should be updated. (Assume each customer is served with 10 liters), and a warning message should be displayed when the stock reaches a value of 500 liters. Operator should be able to perform the following tasks by selecting from a console menu.

100 or VFQ:	View all Fuel Queues.
101 or VEQ:	View all Empty Queues.
102 or ACQ:	Add customer to a Queue.
103 or RCQ:	Remove a customer from a Queue. (From a specific location)
104 or PCQ:	Remove a served customer.
105 or VCS:	View Customers Sorted in alphabetical order (Do not use library sort routine)
106 or SPD:	Store Program Data into file.
107 or LPD:	Load Program Data from file.
108 or STK:	View Remaining Fuel Stock.
109 or AFS:	Add Fuel Stock.
999 or EXT:	Exit the Program.

Display all the menu options to the operator, When the operator types 102 or ACQ, it should do the add method. When the operator types 103 or RCQ, it should do the remove method.

Task 2. Classes version.

Create a second version of the **fuel queue management system** using an array of **FuelQueue** Objects. The Class version should be able to manage 5(Five) pumps parallelly. Create a class called **FuelQueue** and another class called **passenger**. The program should function as in Task 1. Each queue can hold up to 6 passengers with the following additional information.

i. First Name.
ii. Second Name.
iii. Vehicle No
iv. No. of liters required.

Note: In class version, **add customer to the Fuel queue (102 or ACQ)** option must select the queue with the minimum length.

Add an additional option to the menu. '110 or IFQ' that will give the user the option to print the income of each Fuel queue. (You can take price of a fuel liter as 430). Otherwise, the program should function as in Task 1.

Task 3. Add to a waiting Queue.
Add a waiting list to your Fuel Queue class version.

Modify your **'102 or ACQ Add customer to a Queue' and '104 or PCQ: Remove a served customer'** as follows:

- When you press '102 or ACQ' to add a new customer, the customer should be added to the Waiting List queue if the Fuel queues are full.
- When you press '104 or PCQ' to remove a served customer, the next customer in the Waiting List queue should be automatically placed in the fuel queue. Extra marks will be awarded if you implement the waiting list queue as a circular queue.

Task 4. JavaFX. Create a GUI for the operator to **view** the status of the queue.

- The operator should be able to see the passengers' details who are waiting in the fuel queue along with the passengers in the waiting queue.
- The operator should be able to **Search** the details of a passenger.

Task 5. **Testing & Report**. Create a table of test cases showing how you tested your program (see below for example). Write a brief (no more than one page) discussion of how you chose your test cases to ensure that your tests cover all aspects of your program. Copy your entire code along with the test cases to the given coursework report template and upload as a pdf file to the separate link provided in the black board.

Test Case	Expected Result	Actual Result	Pass/Fail
Fuel Queue Initialized Correctly After program starts, 100 or VFQ	Displays 'empty' for all queues.	Displays 'empty' for all Queues.	Pass
Add passenger "Jane" to Queue 2 102 or ACQ Enter Queue: 2 Enter Name: Jane	Display 'Jane added to the queue 2 successfully"	Display "Queue is full"	Fail

Note: Solutions should be java console applications up to task 3, javaFX application for task 4

Marking scheme

The coursework will be marked based on the following marking criteria:

Criteria	Max for Subcomponent	Max Subtot
Task 1 2.5 marks for each option (10 options) Menu works correctly	25 5	**(30)**
Task 2 Fuel Queue class correctly implemented. Passenger class correctly implemented. Income of each Fuel Queue correctly implemented.	10 8 7	**(25)**
Task 3 Waiting list queue implementation "102 or ACQ': Add" works correctly "104 or PCQ': Delete" works correctly Circular queue implementation	5 5 5 5	**(20)**
Task 4 JavaFX GUI for Viewing the Fuel Queue GUI for Search the passenger	 5 5	**(10)**
Task 6 Test case coverage and reasons	10	**(10)**
Coding Style (Comments, indentation, style) **Total**	5	**(5)** **(100)**
Demo: At the discretion of your tutor, you may be called on to give a demo of your work to demonstrate understanding of your solutions. If you cannot explain your code and are unable to point to a reference within your code of where this code was found (i.e., in a textbook or on the internet) then significant marks will be lost for that marking component.		
NOTE: If you do not attend your online demo only task 1 will be marked.		