4COSC010C.3 Software Development II
Coursework 22/23

Module Code / Title:	4COSC010C.3 / Software Development II
Assessment Component:	Coursework
Weighting:	50%
Qualifying mark:	30%
Academic Year:	2022 - 2023
Semester:	3 (Trimester)
Module Leader:	Deshan Sumanathilaka
Handed out:	Monday 29th May 2023 (week 3)
Due date:	**Monday 10th July 2023 at 1pm** (week 9) **Coursework demonstrations will be from 10th week onwards**
Learning outcomes:	LO1: Choose appropriate algorithms and data structures for problem solving and implement these using a programming language LO3: Develop solutions to common programming problems using classes to implement fundamental object orientated concepts LO4: Implement common data structures and data sorting algorithms LO5: Undertake basic requirements gathering and data modelling.
Expected deliverables:	You must submit the following files: - Self-evaluation form (word or pdf) - Your IDE Project file exported into a Code.zip file.
Method of submission:	Submission on Blackboard.
Marks	Marks will be given within 20 working days (4 weeks) after the submission deadline. All marks will remain provisional until formally agreed by an Assessment Board.
Feedback	Constructive feedback will be given 15 working days (3 weeks) after the submission deadline.

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 submissions

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 Student Centre in writing of a mitigating circumstances form, giving the reason for your late ornon-submission. You must provide relevant documentary evidence with the form. This information will be reported to the relevant Assessment Board that will d ecide whether the mark of zero shall stand. For more detail you must inform the Student Centre in writing on a mitigating circumstances form, giving the reason for your late ornon-submission. You must provide relevant documentary evidence with the form. This information will be reportedto the relevant Assessment Board that will decide whether the mark of zero shall stand. For more detailed information regarding University Assessment Regulations, please refer to the following website:http://www.westminster.ac.uk/study/current-students/resources/academic-regulations.

Coursework Description

General notes

- Use inbuilt methods when possible.
- Use descriptive names for your variables and user-defined methods.
- Add comments to explain your code and use a good style.
- Re-use the methods you implement in your coursework when possible.
- Use an IDE (IntelliJ IDEA)
- Reference within your code any code adapted from external or other sources, or any technology that you may have used to implement your solution.
- For each task, read first what is requested, think about the design (you can use pen +paper) and then implement it.

Foodies Fave Queue Management System.

Task 1. Arrays version.

Design a program for a **Foodies Fave Food center** which has 3 cashiers where each queue can have maximum of 2,3,5 customers respectively. 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). The Food Center will have exactly 50 burgers in stock. For each customer added to the queue (Name should be tracked.), stock should be updated. (Assume each customer is served 5 burgers), and a warning message should be displayed when the stock reaches a value of 10 burgers. Operators should be able to perform the following tasks by selecting from a console menu.

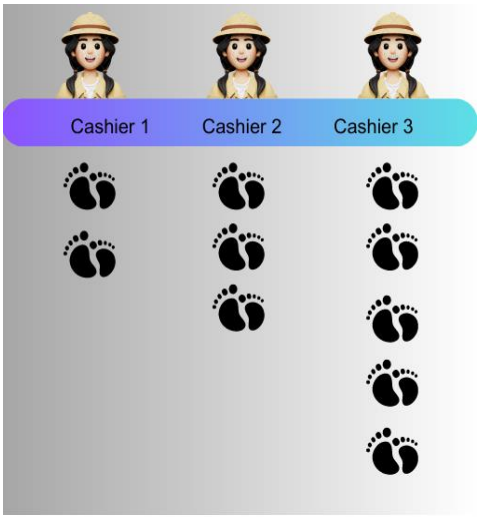

Figure 1.1 Foodie Fave Queue Plan

100 or VFQ: View all Queues.
The queues need to be printed in the below format.

```
*****************
*    Cashiers   *
*****************
  O    O    O
  O    X    X
       X    X
            X
            X
```

X – Not Occupied O – Occupied

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 burgers Stock.
109 or AFS: Add burgers to 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 **Foodie Fave queue management system** using an array of **FoodQueue** Objects. The Class version should be able to manage 3 Queues parallelly. Create a class called **FoodQueue** and another class called **Customer**. The program should function as in Task 1. Each queue can hold up to 2,3,5 customers respectively with the following additional information.

i. First Name.
ii. Second Name.
iii. No. of burgers required.

Note: In class version, **add customer to the 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 queue. (You can take the price of a burger as 650). Otherwise, the program should function as in Task 1.

Task 3. Add to a waiting Queue.
Add a waiting list to your 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 Food 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 food 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 queues.

- The operator should be able to see the customer's details who are waiting in the food queue along with the customers in the waiting queue.
- The operator should be able to **Search** the details of a customer.
- GUI should be called through Command Prompt. (112 or GUI)

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
Food Queue Initialized Correctly After program starts, 100 or VFQ	Displays 'empty' for all queues.	Displays 'empty' for all Queues.	Pass
Add customer "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 Subtotal
Task 1 2.5 marks for each option (10 options) Menu works correctly	25 5	**(30)**
Task 2 FoodQueue class correctly implemented. Customer class correctly implemented. Income of each Food 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		**(10)**

GUI for Viewing the Food Queue GUI for Search the Customer	5 5	
Task 5 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.		

Submitting your coursework

- Create a ZIP file of your project created based on the used IDE (i.e IntelliJ): Make sure the downloaded zip file is in working condition. You need to do the demonstration using downloaded zip file from Black Board during VIVA.
- Rename your code.zip file and the self-assessment form in the following format.
  - Code File: <IITNO>_<UoWNo>.zip
  - Self-Assessment form : <IITNO>_<UoWNo>.pdf
- Go to Blackboard's module page and select Assessment (coursework) > Coursework submission. **Attach below files as separate files in the same submission.**
  - Self-assessment form (word or pdf)
  - Code.zip file with both Array version and the class version.
- Submit your work.

END