4COSC010C.3 Software Development II
[REF/DEF] Coursework 22/23

Module Code / Title:	4COSC010C.3 / Software Development II
Assessment Component:	Coursework [RFF\DEF]
Weighting:	50%
Qualifying mark:	30%
Academic Year:	2022 - 2023
Semester:	3 (Trimester)
Module Leader:	Deshan Sumanathilaka
Handed out:	
Due date:	**28th August 2023, 1pm**
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 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. For more detail you must inform the Student Centre 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. 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.

Snack King Queue Management System.

Task 1. **Arrays version.**

Design a program for a **Snack King Food center** which has 3 cashiers where each queue can have a 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 Shop will have exactly 100 pizzas in stock. For each customer added to the queue (Name should be tracked.), stock should be updated. (Assume each customer is served 10 pizza), and a warning message should be displayed when the stock reaches a value of 20 pizza. Operators should be able to perform the following tasks by selecting from a console menu.
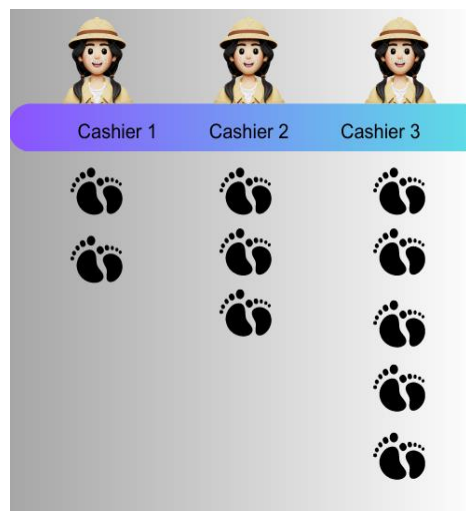

Figure 1.1 Snack King Queue Plan

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.

View all the queues which have at least one empty slot. Indicate the empty slot while printing the Empty Queues.

E.g.:   Queue 2: 2,3
Queue 3: 2,3,4,5

102 or ACQ: Add customer to a Queue.

103 or RCQ: Remove a customer from a Queue. (From a specific location)

Note: Pizza count will not be reduced.

104 or PCQ: Remove a served customer.

Note: Pizza count needs to be reduced.

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.

Note: Data needs to be read from the text file and store it back to the arrays.

108 or STK: View Remaining pizza Stock.

109 or AFS: Add pizza 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. Validation needs to be performed according.

Task 2. **Classes version.**

Create a second version of the **Snack King 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. (String)
     ii.   Second Name. (String)
     iii.   ID (int)
     iv.   No. of pizza required. (int)

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 pizza as

LKR 1350). Otherwise, the program should function as in Task 1.

Task 3. Add to a waiting Queue. **(Waiting Queue Size: 5)**
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 Listqueue should be automatically placed in the food queue.

Task 4. **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.(Turnitin)

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.

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

Criteria	Max for Subcomponent	Max Subtotal
Task 1 3.5 marks for each option (10 options)	35	**(40)**
Menu works correctly	5	
Task 2 FoodQueue class correctly implemented.	10	**(25)**
Customer class correctly implemented.	8	
Income of each Food Queue correctly implemented.	7	
Task 3 Waiting list queue implementation.	5	**(20)**
"102 or ACQ': Add works correctly.	5	
"104 or PCQ': Delete works correctly.	5	
Circular queue implementation	5	

Task 4 Test case coverage and reasons	10	**(10)**
Coding Style (Comments, indentation, style)	5	**(5)**
Total		**(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 viva, you will be graded with 0.

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 > REF/DEF. **Attach below files as separate files in the same submission.**

  - Self-assessment form (pdf)
  - Code.zip file with both Array version and the class version.
- Submit your work.

END