TCP1201 Objected-Oriented Programming and Data Structures

Project (40%)

Trimester 1, Session 2020/2021
Faculty of Computing and Informatics
Multimedia University

DUE DATE: 25 October 2020, 11:59pm

1. Foreword

- This is a group project with maximum of **TWO (2)** students per group.
- No late work will be accepted and no deadline will be extended.
- **Group leader** of each group is to upload the deliverables to **MMLS**.
- Strictly no copying from any other sources except codes given in this course. If detected all parties involved will get zero marks.

2. Task

This project is the improvement of the Assignment whereby you are now required to complete the Phase 2 Delivery feature of Foodeliver.

The four (4) main roles in the Phase 2 are listed below:

1) Admin – the administrator of Foodeliver who is responsible for adding riders in the system. The admin can also view the rider queue, all consolidated order histories, and order statistics in the system.
2) Restaurant – a restaurant can add/update/delete its menu in Foodeliver, view order history, and update order status (preparing, ready, delivering, collected, etc.)
3) Customer – a customer can purchase food from Foodeliver, view order status and history, and collect food at the restaurant if self-collect is the choice.
4) Rider – a rider can deliver an order, view order status and history, and check how many more riders before his turn. The system assigns orders to riders based on queue.

Each student shall handle minimum 2 roles.

Any update from one role shall be reflected in the record of the other roles. For example, if a customer places an order for delivery, the restaurant and a rider should receive the order, etc.

It is recommended to design each role as a program by itself, and share the data in files.

Design your classes, data fields, and methods wisely. Do not over design by adding classes, data fields, or methods that are not used or do not bring significant value to the system, e.g. I/C or passport number, date of birth, etc.

To make testing easier and save time during presentation, your program shall have the following features and data pre-loaded when the system starts executing:

i. Never clear screen.
ii. Pre-load with 3 restaurants, each with 3 dishes.
iii. Pre-load with 3 customers, each with 2 delivered orders.
iv. Pre-load with 2 riders, each with 3 delivered orders.

3. Items to be submitted:

i. Java source code. Make sure the code can be compiled and run.
ii. Java documentation based on Javadoc.
iii. A PDF report showing:
    a. Include the leader and member details and contact number like in the table below:

ID	**Name**	**Phone**	**Role (Admin, Customer, Restaurant, Rider, etc.) 2 roles per members**

    b. A complete UML Class Diagram that matches the Java source code.
    c. User manual on navigating the system with screenshots of the output

Zip the THREE (3) files/folders above and label it in the following format:

TC0X_TT0X_Student1ID_StudentName_Student2ID_Student2Name.zip

For example: Both Ali Yusof Bin Mohamed Ahmad (ID: 123456789) and Pravin Sriram A/L Subramanioum (ID: 234567890) are from lecture session TC01 and tutorial session TT02. Their zipped file will be:

TC01_TT02_123456789_Ali_Yusof_234567890_Pravin_Sriram.zip

Group leader takes the responsibility to upload the zipped file to MMLS under his/her name.
A presentation is required for this project.

Mark Sheet (40%)

Criteria	Item (Mark for an item is awarded if it works and student can explain)
1. OOP Design, UML Class Diagram, and Java Documentation (17 marks)	**1.1. UML Class Diagram (2m)** 2.0m – no error 1.5m – one type of error 1.0m – more than one types of errors 0.0m – no diagram
	1.2. Association, Aggregation, and/or Composition (3m) 3m – Excellent design (correct modeling without flaw) 2m – Fine design (mostly correct modeling with minor flaw) 1m – Poor design (incorrect modeling with major flaw) 0m – Do not use
	1.3. Inheritance, Polymorphism, and/or Abstract Class (3m) 3m – Excellent design (correct modeling without flaw) 2m – Fine design (mostly correct modeling with minor flaw) 1m – Poor design (incorrect modeling with major flaw) 0m – Do not use
	1.4. Implement Comparable and Comparator interfaces (2m) 1m – Comparable interface 1m – Comparator interface
	1.5. Implement a custom generic Queue data structure (2m) 2m – Correct implementation without aggregating or extending from Java library. Do not use Java Queue or LinkedList class in your code. 1m – Correct implementation by aggregating or extending from Java Queue or LinkedList class.
	1.6. Good identifier and small methods (2m) 2.0m – All identifiers are properly named and Do not have any long methods (more than 100 lines). 1.5m – Some identifiers are not properly named or has a long method 1.0m – Some identifiers are not properly named and has a long method 0.5m – Many identifiers are not properly named and has a long method
	1.7. Java Documentation for custom Queue class (3m) 1m – Correct class and methods descriptions 1m – Correct parameter description 1m – Correct return description
2. Program Execution (20 marks)	**2.1. User friendliness (2m)** 2.0m – Input and output are clear without ambiguity. 1.5m – One input or output are unclear with ambiguity. 1.0m – More than one input or output are unclear with ambiguity. 0.5m – The program is unusable.
	2.2. Correct program features and output (18m) The values of all useful relevant data must be shown when displaying info. i. Consist of 4 programs: Admin, Customer, Restaurant, Rider. (2m] ii. Use files for saving and sharing data. (1m) iii. Catch FileNotFoundException gracefully by providing an error message such as “File Restaurants.txt is not found”. (1m) iv. Admin can view and add riders in the system. (1m)

	v. Restaurants can view, add, update, and remove food in its menu. (2m) vi. Customers can view all food in system. (1m) vii. A customer can choose to self-collect or delivery when ordering foods. (1m) viii. Admin, customers, restaurants and riders can view order history and status. (2m) ix. Matching change of status for an order among admin, customer, restaurant and rider (2m) x. A rider can delivery food from different restaurants. (1m) xi. A rider can check how many more riders before his turn. (1m) xii. The rider queue assigns riders correctly. (1m) xiii. Admin can view order statistics based on two different sorting orders: by rider with highest delivery count, and by restaurant/customer with highest-order amount. (2m)
3. Presentation (3 marks)	Clear presentation (3m) 3m – Overall smooth and well prepared 2m – Overall fine but has some hiccups 1m – Unclear
5. Plagiarism, late submission, or no presentation	0 mark for the whole Project