NATIONAL UNIVERSITY OF SINGAPORE

CS2113/T – SOFTWARE ENGINEERING AND OBJECT ORIENTED PROGRAMMING

(Semester 1: AY2018/19)

Part 2

Time Allowed: 1 Hour 30 min

INSTRUCTIONS TO STUDENTS

1. Please write your Student Number only. Do not write your name.
2. This assessment paper contains **FIVE** questions and comprises **EIGHT** printed pages.
3. You are required to answer **ALL** questions.
4. Write your answers in the space provided.
5. This is an **OPEN BOOK** assessment.
6. You may **use pencils** to write answers; but write legibly.

STUDENT NO:									

This portion is for examiner's use only

Question	Marks	Remarks
Q1	/6	
Q2	/6	
Q3	/6	
Q4	/6	
Q5	/6	
Total	/30	

~ Blank ~

System description relevant to all questions in the assessment.

You are building software for managing restaurants that serve food for drive-through customers only (think of a drive-through McDonalds for example).

The customers place their order at the entry counter using an automated touch interface. After the customer completes the payment, the order is added to the order queue. The manager can monitor the status of the orders. Each cooking table in the kitchen caters to one type of food (appetizer, main course, dessert, and beverage). Orders in the order queue, filtered by the food type, are also displayed on the table-specific displays.

The order being prepared is marked as 'in-progress'. Once the order is ready, the cook marks it as 'complete'. The order is then moved to the 'pick-up' queue. The employee in the pick-up unit marks the order as 'delivered' once the food is packed and given to the customer.

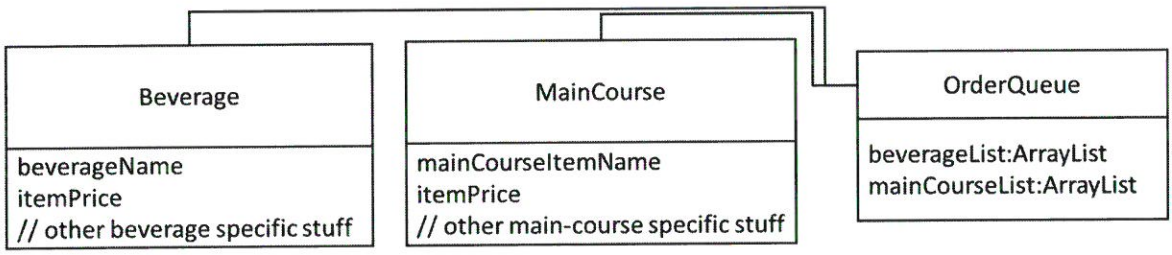
All orders are recorded in the database for bookkeeping and auditing.

Q1 [6 marks]

(a) [4 marks] Identify the different actors who use the system and write one must-have user story for each of the actors identified.

(b) [2 marks] Give one non-functional requirement for the application. (Choose a requirement that is specific to this application rather than one that is applicable to most other software)

Q2 [6 marks] You hired a contract employee to implement a minimal viable product (MVP) of your system. You specified that the MVP should support the main-course items and beverages. The contractor delivered the system according to the (partial) design below. Now, you want to add more types of food (i.e., dessert and appetizer). Explain the problem to a new contract employee and propose a new design to implement by drawing an appropriate UML diagram. Include all missing details. [Note: A minimal viable product has just the sufficient features to enable early adoption of the product. More features are added after receiving feedback from the customer.]



(a) Current problem (Which design principle is violated?)

(b) New design

Q3 [6 marks] Consider the partial design in **Q2**. The code for adding an order to the queue is as follows.

```
// . . . other code
if (type.equals("Beverage")) {
    beverageList.add(item);
} else {
    mainCourseList.add(item);
}
// . . . more code
```

Show the interaction between the objects in the above code fragment using an appropriate UML diagram.

Q4 [6 marks] Identify at least 4 coding standard violations and 2 two other suggestions to improve the code below as shown in the example.

Incorrect format for the header comment; should be "Adds"

```
/**
* Add an order to the order queue
* @param item should be non-null.
* @return
*/
Bill add_order_to_orderQueue (Dessert item) throws BulkOrderException {

    if (item.quantity > 10)
        throw new BulkOrderException("Please contact staff for bulk orders");

    bill = new Bill(item.quantity, item.price,  CommonConstants.GST);
    orderQueue.addItem(item);

    int orderNumber = orderQueue.getOrderInfo(item).getOrderNumber();
     bill.addOrderNumber(orderNumber); // add order number information to the bill

    return bill;
}
```

Q5 [6 marks] Consider the constructor below for the `Bill` class:

```
/**
 * Constructs the bill object with the final price calculated as
 * the product of the quantity and the price.
 * To associate the bill to an order, the order number has to be
 * set separately
 * @param quantity the number of items purchased; is +ve, non-zero
 * @param price the price of each item; is +ve, non-zero
 * @returns the Bill object with the final price calculated
 */
Bill(int quantity, int price);
```

(a) Give one positive and one negative test case for the constructor based on the description above

Positive
Negative

(b) Propose an effective and efficient set of test cases to test the constructor based on the description above. Give at least 4 test cases.

Quantity	Price

~~ End of assessment (Part 2) ~~

~ Blank ~