

CE/CZ2002 Object-Oriented Design & Programming Assignment

APPENDIX B:






Attached a scanned copy with the report with the filled details and signatures.

Declaration of Original Work for CE/CZ2002 Assignment

We hereby declare that the attached group assignment has been researched, undertaken, completed and submitted as a collective effort by the group members listed below.

We have honored the principles of academic integrity and have upheld the Student Code of Academic Conduct in the completion of this work.

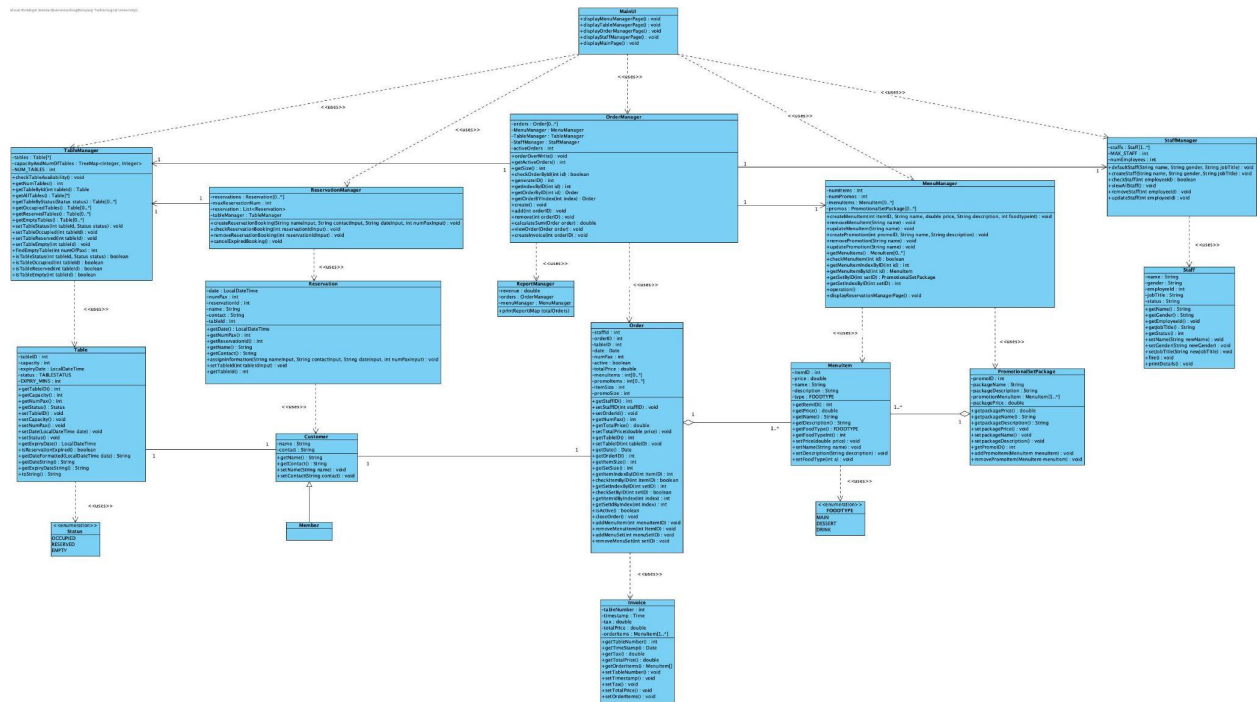
We understand that if plagiarism is found in the assignment, then lower marks or no marks will be awarded for the assessed work. In addition, disciplinary actions may be taken.

Name	Course (CE2002 or CZ2002)	Lab Group	Signature /Date
Lim En Ning	CZ2002	MACS	 14/11/2021
Lim Jin Hng, Timothy	CZ2002	MACS	 14/11/2021
Tan Jia Min	CZ2002	MACS	 14/11/2021
Ching Jia Yin, Vivienne	CZ2002	MACS	 14/11/2021
Choo Ming-Hui, Megan	CZ2002	MACS	 14/11/2021

Important notes:

1. Name must **EXACTLY MATCH** the one printed on your Matriculation Card.

1 UML Class Diagram (clearer image attached in zip file)



2 Design Considerations

Our design of a console-based Restaurant Reservation and Point of Sale System (RRPSS) application adopts an Object-Oriented Approach, which organises the application by using classes and objects. Our application adopts a total of 18 classes, and is modelled after a sushi restaurant.

2.1 OO Concepts

Object-Oriented Programming is centered around the four following principles: encapsulation, abstraction, inheritance and polymorphism. Our RRPSS project focuses mainly on encapsulation and inheritance principles.

2.1.1 Encapsulation

Encapsulation conceals an object's private data from other classes. Class attributes are defined as private, these pieces of information are hidden and hence directly inaccessible from outside classes. Important attributes will have to be accessed using their respective `get()` methods.

The following is an example of how our application applies the encapsulation principle. Reserving a table requires input of the customer's name and contact number. In order to protect the data and identity of the customer from other classes, his/her name and contact number should be declared as private attributes, which can only be accessed using the public `getName()` and `getContact()` methods respectively.

```
public String getName() {  
    return name;  
}  
  
public String getContact() {  
    return contact;  
}
```

2.1.2 Inheritance

Inheritance refers when a new class, known as the subclass, inherits the properties and methods of its parent class, known as the superclass. Inheritance allows us to improve the efficiency and readability of our code as it eliminates the need for duplicated code.

For instance, we have defined our Member class as a subclass of Customer, as they share the same attributes name and contact, as well as their getters and setter methods. The only difference is that a Member is a specific type of Customer that is entitled to discounts. By using inheritance, we do not have to unnecessarily repeat the code in Customer for Member. The inheritance is done by using the “extends” keyword in Java.

2.1.3 Abstraction

Abstraction in java is done using either abstract classes or interfaces. Abstract classes have one or more methods that are declared without any implementation, known as abstract methods, along with other concrete methods if any. On the other hand, interfaces only consist of abstract methods, with no concrete classes. A single class can implement multiple interfaces. For our RRPSS application, we have done without the use of abstraction as we did not deem it necessary for any of the relationships between our classes.

2.2 SOLID Design Principles

SOLID stands for the following key design principles: Single Responsibility Principle, Open-Closed Principle, Liskov Substitution Principle, Interface Segregation Principle, and Dependency Inversion Principle. The SOLID principles reduce dependencies in the code, to facilitate engineers in making changes to one area of software without impacting others. The principles make designs more understandable and also easier to maintain and extend. These lead to adaptive and agile software.

Most prominently, our application applies the Single Responsibility Principle (SRP) which suggests that each class should only take on a single responsibility. This is because otherwise, dependency in code between classes may arise, and making changes to the code will become a tedious process.

Our application applies SRP as each class is in charge of a single job. For example, MenuManager controls all the logic involving the menu such as creating/removing items from the menu, while ReservationManager is responsible for handling all the reservations. Since the

responsibilities of each class is well delineated, modifications in code should only be necessary in the involved class.

2.3 Assumptions Made

Assumptions on Tables

Firstly, we assume that the number of tables, as well as the capacity of these tables in the restaurant are fixed, and cannot be changed with usage of our application.

Assumptions on Staff

There are three predetermined types of staff in our application: Chef, Manager and Waiter. It is assumed that all three staff types have access to the application and are able to create orders. The maximum number of staff our restaurant can hire is 100.

Assumptions on Menu

Items on the menu have to be uniquely named, but different items can share the same price and description. The same applies for Promotional Set Packages.

Assumptions on Reservation

Reservations may be made for any given time, as long as the date and time is in the future. Each reservation will expire 30 minutes after its reservation timing.

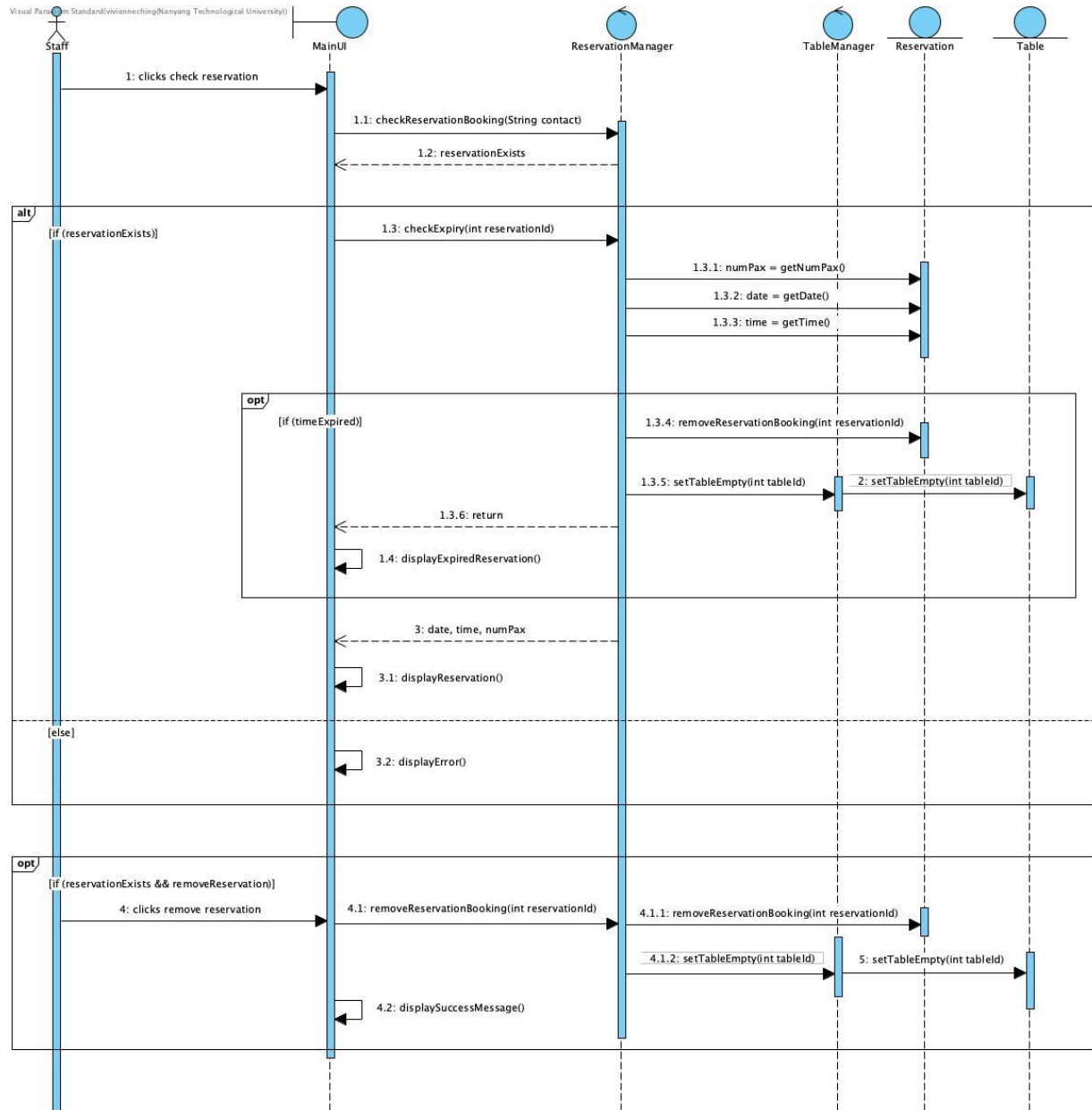
Assumptions on Order

There is no capacity for the number of items that can be added to the order and we assume an unlimited supply of every menu item.

Assumptions on Sales Report

We use the date the order was created to determine whether it should be in the report, rather than the invoice date. This is to account for all orders during the period stated.

3 Sequence Diagram (Clearer image attached in zip file)



4 Screen Captures of Test Cases (Not Included in Video)

4.1 Order Manager

4.1.1 to 4.1.4

```
-----
Manage Orders
-----
Select task:
(1) Create Order
(2) Add Item to Order
(3) Remove Item from Order
(4) Print Order Invoice
(5) Print Sale Revenue Report
(6) Back to Main

Enter choice (1-6): 1
Input your staffID : 20
Error : Input valid staffID
Input reservation ID (enter -1 if there is no reservation):1
Table -1 and NumPax: -1 assigned from reservation
No such reservation!
Input reservation ID (enter -1 if there is no reservation):-1
Input no of people : 25
Maximum 10 per table only
Input no of people : 10
Table 27 assigned
1. Add Menu Item
2. Add Promo Item
3. Finish
3
Error : Order is empty. Please add items before finishing

1. Add Menu Item
2. Add Promo Item
3. Finish
1
Input Menu Item ID to add : 5
Cake added

1. Add Menu Item
2. Add Promo Item
3. Finish
3
Order creation successful
Sun Nov 14 11:23:59 SGT 2021
Order ID : 1
Table ID : 27
Staff ID : 20
```

Test Case Input	Expected Output	Actual Output
4.1.1 Finishing order without adding any menu items/sets	“Error: Order is empty. Please add items before finishing.”	“Error: Order is empty. Please add items before finishing.”
4.1.2 Input nonexistent staff ID Eg 20	“Error: Input valid staffID”	“Error: Input valid staffID”
4.1.3 Input number of people as more than 10 Eg 25	“Maximum 10 per table only”	“Maximum 10 per table only”

4.1.4 Input reservation ID that does not exist Eg 1	“Table -1 and NumPax: -1 assigned from reservation No such reservation!”	“Table -1 and NumPax: -1 assigned from reservation No such reservation!”
--	---	---

4.2 Table Manager

4.2.1

```

-----
                        Manage Tables
-----
Select task:
(1) Check Table Availability
(2) List Occupied Tables
(3) List Reserved Tables
(4) List Available Tables
(5) List All Tables
(6) Back to Main

Enter choice (1-6): 1
Number of People: 100
No tables available.

```

4.2.2

```

-----
                        Manage Tables
-----
Select task:
(1) Check Table Availability
(2) List Occupied Tables
(3) List Reserved Tables
(4) List Available Tables
(5) List All Tables
(6) Back to Main

Enter choice (1-6): 2
0 occupied tables in total.

```

Test Case Input	Expected Output	Actual Output
4.2.1 Input number of people more than maximum table capacity Eg 100	“No tables available.”	“No tables available.”
4.2.2 Selecting option to list occupied tables when none are occupied	“0 occupied tables in total”	“0 occupied tables in total”

4.3 Reservation Manager (Demonstrated in video)

Test Case Input	Expected Output	Actual Output
4.3.1 Input date in the wrong format	“Invalid format for date! Date format should be as follows: yyyy-MM-ddTHH:mm:ss This reservation ID does not exist!”	“Invalid format for date! Date format should be as follows: yyyy-MM-ddTHH:mm:ss This reservation ID does not exist!”
4.3.2 Input a number with less than 8 digits for phone number	“This phone number is invalid!”	“This phone number is invalid!”
4.3.3 Input date and time in the past for reservation timing	“Reservation time must be in the future! This reservation ID does not exist!”	“Reservation time must be in the future! This reservation ID does not exist!”

4.4 Menu Manager

4.4.1

```
-----
                        Manage Menu
-----
Select task:
(1) Create Menu Item
(2) Remove Menu Item
(3) Update Menu Item
(4) Create Promotional Package
(5) Remove Promotional Package
(6) Update Promotional Package
(7) Print Menu
(8) Print Promotional Package
(9) Back to Main

Enter choice (1-9): 3
Name of menu item you want to update: Hot Green Tea
Which would you like to update? (1) Name, (2) Price, (3) Description, (4) Food Type
1
Enter new name: Water
Name cannot be updated to an existing name
```

Test Case Input	Expected Output	Actual Output
4.4.1 Update a name of menu item to one that already exists Eg 3, “Hot Green Tea”, 1, “Water”	“Name cannot be updated to an existing name”	“Name cannot be updated to an existing name”

4.5 Staff Manager

4.5.1

```
-----  
                        Manage Staff  
-----  
Select task:  
(1) Create Staff  
(2) View All Staff  
(3) Remove Staff  
(4) Update Staff Details  
(5) Back to Main  
  
Enter choice (1-5): 3  
Enter employee ID: 8  
Invalid ID. Unable to remove.
```

4.5.2

```
-----  
                        Manage Staff  
-----  
Select task:  
(1) Create Staff  
(2) View All Staff  
(3) Remove Staff  
(4) Update Staff Details  
(5) Back to Main  
  
Enter choice (1-5): 4  
Enter employee ID: 23  
Invalid ID. Unable to update.
```

4.5.3

```
-----  
                        Manage Staff  
-----  
Select task:  
(1) Create Staff  
(2) View All Staff  
(3) Remove Staff  
(4) Update Staff Details  
(5) Back to Main  
  
Enter choice (1-5): 1  
Enter name: Jane  
Enter gender(F/M): N  
Enter job title(Chef/Manager/Waiter): Chef  
|  
Unable to create new staff.  
Staff's gender entered is invalid.
```

4.5.4

```

-----
                        Manage Staff
-----
Select task:
(1) Create Staff
(2) View All Staff
(3) Remove Staff
(4) Update Staff Details
(5) Back to Main

Enter choice (1-5): 1
Enter name: Jane
Enter gender(F/M): F
Enter job title(Chef/Manager/Waiter): Cashier

Unable to create new staff.
Staff's job title entered is invalid.

```

Test Case Input	Expected Output	Actual Output
4.5.1 Remove staff with ID that does not exist Eg 3,8	"Invalid ID. Unable to remove"	"Invalid ID. Unable to remove"
4.5.2 Update Staff with ID that does not exist Eg 23	"Invalid ID. Unable to update"	"Invalid ID. Unable to update"
4.5.3 Input invalid gender Eg "N"	"Staff's gender entered is invalid"	"Staff's gender entered is invalid"
4.5.4 Input invalid job title Eg "Cashier"	"Staff's job title entered is invalid"	"Staff's job title entered is invalid"

5 Video Demonstration

https://youtu.be/SjYjgRi1k_0