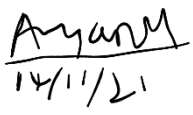
Declaration of Original Work for CE/CZ2002 Assignment

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
Pathak Siddhant	CZ2002	SSP6	
Bhatia Nipun	CZ2002	SSP6	
Arora Kanupriya	CZ2002	SSP6	
Madan Aryan	CZ2002	SSP6	

S.N.A.K. - Restaurant Reservation and Point of Sale System Application

AY 2021/22
CZ2002

SIDDHANT PATHAK, KANUPRIYA ARORA, NIPUN BHATIA, ARYAN MADAN

INTRODUCTION

Restaurant Reservation and Point of Sale System (RRPSS) is a console-based application designed and developed for restaurant staff to manage restaurant reservations, orders, and invoice generation. The application covers key features such as updating the menu, staff records, orders, reservations, etc.

This report covers the object-oriented programming (OOP) concepts and key design considerations used to implement the application. The design will also be represented in a UML Class Diagram and UML Sequence Diagram for one of the features, showing the interaction and relationship between the objects. Moreover, several test cases are included as well to ensure that the application meets the requirements stated beforehand.

ASSUMPTIONS

- Reservation can only be made in advance. Reservation will be automatically removed 5 minutes after the actual booking time.
- The currency will be in Singapore Dollar (SGD) and Good and Services Tax (GST) and service charges must be included in the order invoice.
- Once an order invoice is printed, it is assumed that payment has been made and the table is vacated.
- Customers with membership cards will be entitled to a discount.
- There is no requirement for access control and there is no need for authentication (login/logout) to use the application.

DESIGN CONSIDERATIONS

OOP concepts are applied comprehensively in this project, in both the design and the implementation of the RRPSS application. As mentioned by Kernighan and Plauger, our team aims to make our application "easy to maintain and modify". The architectural style that we have taken is the n-tier architectural style where higher layers make use of services provided by lower layers but lower layers are independent of higher layers. This further reinforces the separation of concerns and makes sure that the design is easy to maintain and modify.

Single Responsibility Principle (SRP)

The principle states that *there should never be more than one reason for a class to change*. To achieve this, our group makes sure that each class never assumes more than one responsibility since each responsibility is an axis of change, thereby improving cohesion between the classes. There are many examples of this principle being applied in source code. One would be the `menuItem` class, which only manages the attributes of the items in the menu- name, description, and type of item. Similarly, the table class handles the attributes of a single table. This principle can also be seen throughout the UI and manager classes as well, thereby reducing the functional overlaps. This phenomenon also helps us limit the ripple effect when and if we introduce some change into a specific part of the system.

Open-Closed Principle (OCP)

In our design, we have made sure that some of our modules are implemented such that they are open for extension but closed for modification. This allows us to modify and change the functionalities of the modules without changing the source code. It has been said that *"Abstraction is the key to OCP"*, and you can see that we have applied it by private classes and hiding any unnecessary information and details to realize OCP to its full extent. In other words, we want to be able to change what the modules do, without changing the source code of the modules. For example, the `Person` class is a class that extends to the `Customer` and `Staff` class which can be seen in Figure 3 in Appendix. This enables us to extend to more types of people in the application. For example, if we decide to introduce a chef class, to specify different chefs according to the cuisine they specify, we can do so without changing the source code of any of the classes that currently inherit it, i.e., `Customer` and `Staff`.

Liskov Substitution Principle (LSP)

It states that, "*a user of a base class should continue to function properly if a derivative of the base class is passed to it*". This means that even if arguments were to pass to the subclass instead of the base class, the pre-

conditions and the post-conditions of the subclass are met. Essentially, it means that the derived class is substitutable for its base class.

Interface Segregation Principle (ISP)

This principle champions the idea that using several client-specific interfaces is better than using a general all-purpose interface. We did not use any custom interface in any of the code, rather we segregated the responsibilities and established classes for each of them with role-specific duties.

Don't Repeat Yourself (DRY)

The Don't Repeat Yourself (DRY) principle states that duplication in logic and code should be eliminated to improve functionality and efficiency. For example, we have used inheritance with the parent class person so that the methods applicable can be accessed by both the children: `staff` and `customer`. This removes the need for duplication of methods involving age and gender, which are used by both classes.

OBJECT-ORIENTED CONCEPTS

Association

Association is implemented in several classes in the design. An association is a "has-a" relationship between two or more objects in which the objects have their lifetime and there is no owner. There is a one association relationship between two classes: `menuMgr` and `menuItem`. It is a unidirectional relationship as the `menuMgr` class contains an instance of the `menuItem` class. An association relationship is used here because the attributes and methods in the `menuItem` class are used by the `menuMgr`.

Inheritance

It is a mechanism where you can derive a class from another class for a hierarchy of classes that share a set of attributes and methods. The subclasses can easily extend from base classes, affording great reusability and increasing efficiency in the code. For example, the `person` class is the parent class of the `Staff` and `Customer` class. Both the `Staff` and `Customer` classes inherit attributes such as `name, age`, and `gender` from the person class. In this way, we were able to avoid code repetition (refer to figure 3 in Appendix).

Nested Classifier

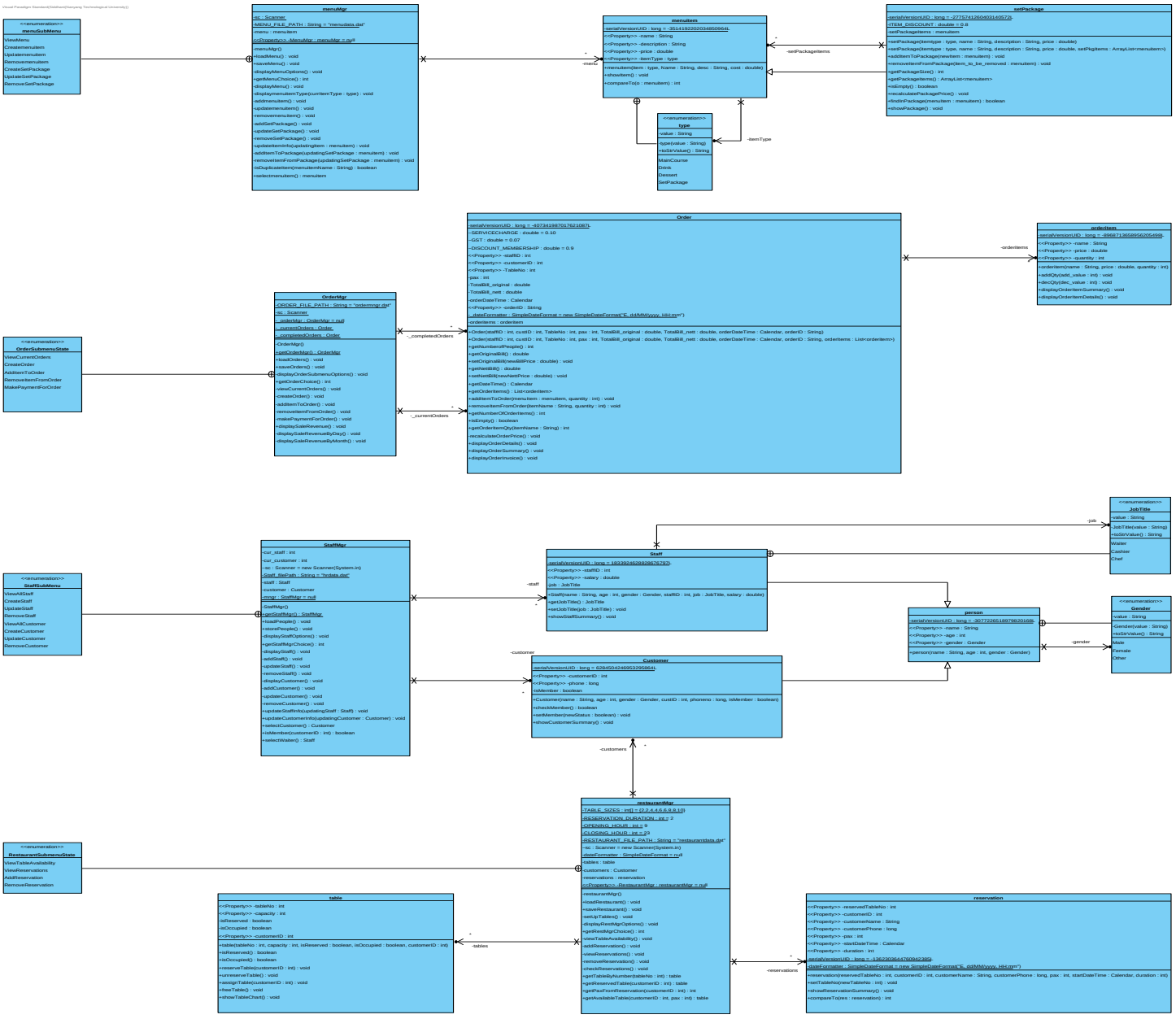
A class or interface could be used as a namespace for various classifiers including other classes, interfaces, use cases, etc. This nesting of classifiers limits the visibility of the classifier defined in the class to the scope of the

namespace of the containing class or interface. As shown in figure 4 in the Appendix, the `enum` class is nested inside the `menuItem` class. Nested enums are implicitly static.

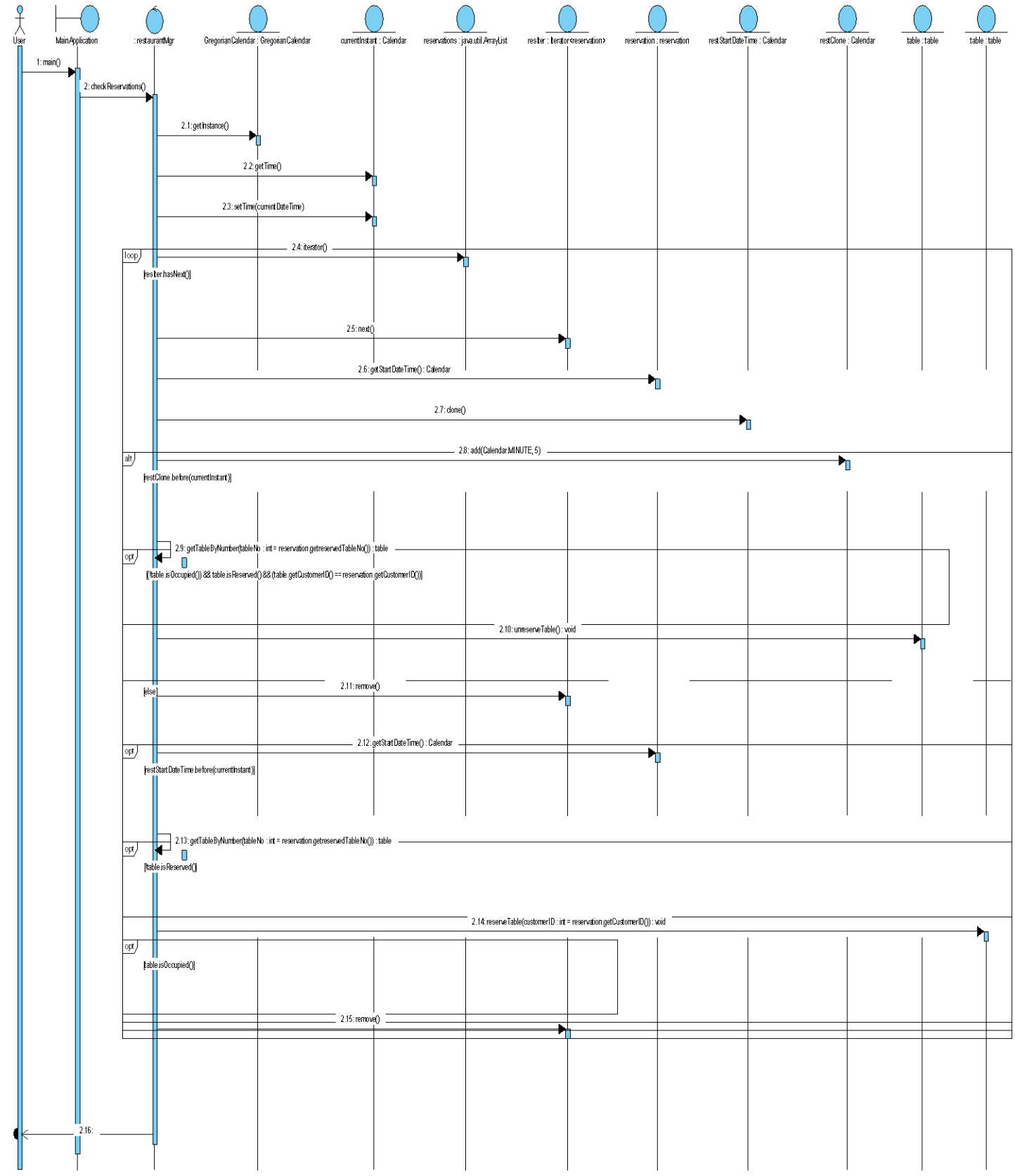
Data Structures

Firstly, for file IO, we serialize and deserialize objects to file (.dat). This is preferred over the text file as the code implemented is cleaner and only one .dat file is needed for the entire system. For the tables, reservations, and customers, arrays are implemented to maintain records for availabilities and assign customers to vacant tables.

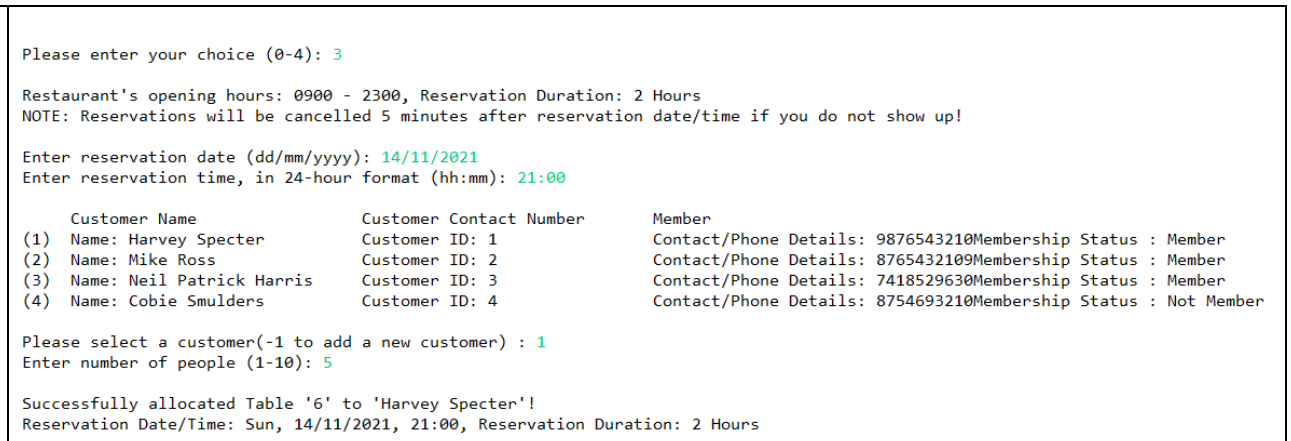
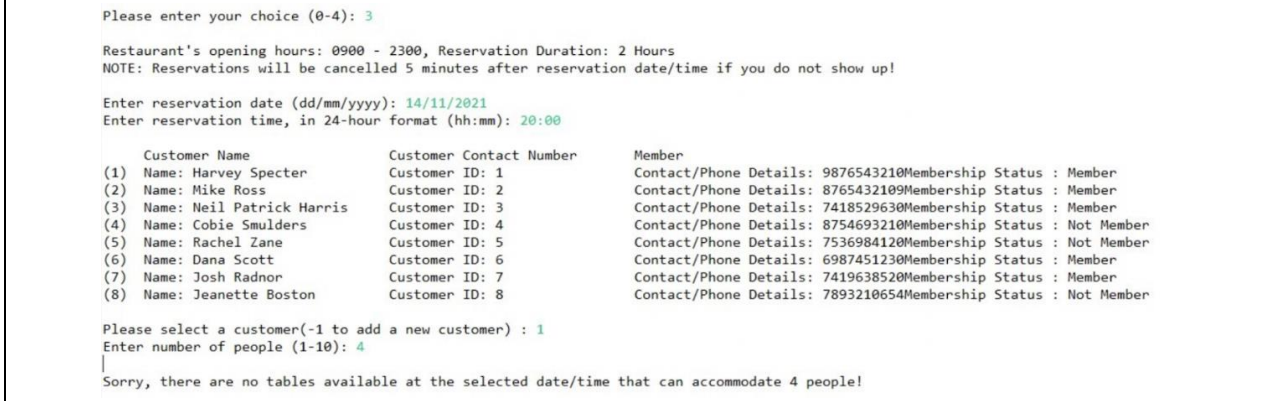
UML CLASS DIAGRAM



UML SEQUENCE DIAGRAM



TEST CASES FOR FUNCTIONAL REQUIREMENTS

Creating a reservation	
Exception handling while creating a reservation	
No vacancies available while making a reservation	

Reservation Expiry

```
****************************************** Reservations ******************************************

(1)  Table Number: 5
     Customer ID: 1          Name: Harvey Specter                          Contact: 9876543210
     Number of People: 5     Reservation Date/Time: Sun, 14/11/2021, 17:35  Duration: 2 Hours

(2)  Table Number: 5
     Customer ID: 2          Name: Mike Ross                               Contact: 8765432109
     Number of People: 5     Reservation Date/Time: Sun, 14/11/2021, 20:00  Duration: 2 Hours

(3)  Table Number: 4
     Customer ID: 4          Name: Cobie Smulders                          Contact: 8754693210
     Number of People: 4     Reservation Date/Time: Sun, 14/11/2021, 20:00  Duration: 2 Hours

****************************************** Reservations ******************************************
```

Figure 1: Reservation List at 5:35 pm

```
****************************************** Reservations ******************************************

(1)  Table Number: 5
     Customer ID: 2          Name: Mike Ross                               Contact: 8765432109
     Number of People: 5     Reservation Date/Time: Sun, 14/11/2021, 20:00  Duration: 2 Hours

(2)  Table Number: 4
     Customer ID: 4          Name: Cobie Smulders                          Contact: 8754693210
     Number of People: 4     Reservation Date/Time: Sun, 14/11/2021, 20:00  Duration: 2 Hours

(3)  Table Number: 6
     Customer ID: 1          Name: Harvey Specter                          Contact: 9876543210
     Number of People: 5     Reservation Date/Time: Sun, 14/11/2021, 21:00  Duration: 2 Hours

****************************************** Reservations ******************************************
```

Figure 2: Reservation List at 5:40 pm. Table 1 reservation was removed due to expiry

Printing Invoice

- Once an order invoice is printed, it is assumed that payment has been made and the table is vacated.
- GST included in the invoice

```
                                        S.N.A.K.

                                76 Nanyang Drive, Block N2.1, #01-08

=================================== Order Invoice ====================================

Order ID: 202111141612253023                    Staff ID: 2              Customer ID: 1
Order Date/Time: Sun, 14/11/2021, 16:12         Table No: 3                     Pax: 4

     (1)  : 1x   PANDAN SHAKE                                                  $27.80

--------------------------------------------------------------------------------------
                                                                    Subtotal: $27.80

                                                                  +10% Service Charge
                                                               +7% Goods & Service Tax
                                                               -10% membership discount

                                                            Total Payable(incl GST): $29.27
--------------------------------------------------------------------------------------

                                Thank you for dining with us!

=================================== Order Invoice ====================================

Table '3' is now available!
```

Exception handling while removing an item from an order

```
Please enter your choice (0-5): 4

     Order ID                Staff ID       Customer ID    table Number
(1)  202111141612253023      2              1              3

Please select the order to remove from (0 to cancel): 1

What would you like to remove from the order?

(1)  1x   SPAGHETTI TOMATO AND BASIL
(2)  1x   PANDAN SHAKE

Please select the order item to remove from (0 to cancel): 1
Enter the quantity to be removed: 5
Invalid input! Maximum quantity is 1..
Failed to remove item from order, please try again..
```

APPENDIX

Figure 3: Usage of Open-Closed Principle

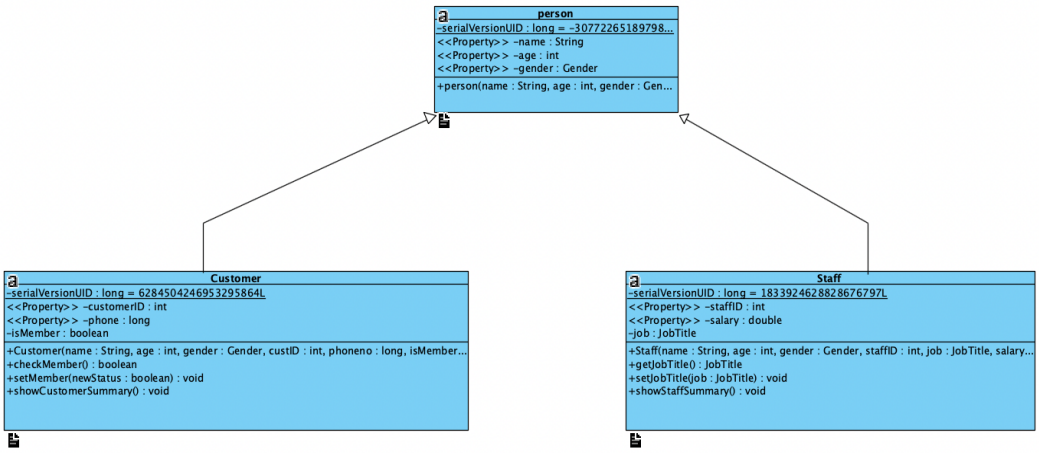
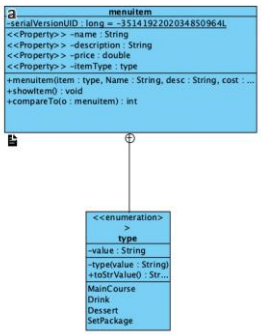


Figure 4: Usage of Nested Classifier



The YouTube link for the video is : https://youtu.be/I2sfudXOy3k

The link is set to public view for ease of access.