Link to video: https://www.youtube.com/watch?v=cib7P7yLUC4

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 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
Amanda Ling Zhi Qi (U2022213G)	CZ 2002	SDDP 3	Amanda /13 Nov
Fabrianne Effendi (U2021488E)	CZ 2002	SDDP 3	Fabrianne /13 Nov
Paul Soloman Low Si En (U2022421F)	CZ 2002	SDDP 3	Paul /13 Nov
Tan Hong Fang Merzen (U2022182E)	CZ 2002	SDDP 3	Merzen /13 Nov
Yam Hui Jing (U2021656A)	CZ 2002	SDDP 3	Hui Jing /13 Nov

1. INTRODUCTION

Restaurant Reservation and Point of Sale System (RRPSS) is an application to computerise the processes of making reservations, recording of orders and displaying of sale records. It will be solely used by the restaurant staff.

1.1 Our Assumptions

In implementing the code, several assumptions are made, in addition to the assumptions required by the project:

- There are 5 tableIDs in total with different seating capacities (2, 4, 6, 8 and 10) and 5 different time slots.
- The 5 time slots for reservation are the same every day, meaning it is always from 10.00-12.00, 12.00-14.00, 14.00-16.00, 16.00-18.00 and 18.00-20.00.
- Discount for a member of Fusion Bistro will be 10%.
- Each customers is assigned a 2-hour time slot
- Each reservation will be auto-removed after that particular time slot the customer has booked has passed. (e.g. If it is 2.05pm, all tables in the time slot 10.00-12.00 and 12.00-14.00 will be removed.)

2. DESIGN PRINCIPLES

2.1 Single Responsibility Principle

We designed our classes with high cohesion and loose coupling. This means that each class only takes on **one responsibility** (there should never be more than one reason for a class to change)

High cohesion

In our application, all classes are designed to fulfill only a single purpose. For example, the menu and the menu items were done through 2 different classes. This allows the menu class to focus on interacting with the users by providing methods such as create, update, remove a display of the items. Hence, the only reason that changes would have to be made to the menu class is when there are new business requirements that affect how the menu is intended to interact with the user.

Loose coupling

In our design, modifying the menu item such as including a new attribute in the main course class (subclass of the menu item) to indicate whether it is vegetarian, would not affect the menu class. This is because we segregated the responsibility of the menu and menu item class. Hence, modification to a specific type of menu item (main course, drinks, etc) would be done on the menu item class instead. This is desirable because classes dependent on the menu class would not be affected by the change since the modifications were done on the menu item class instead.

Ultimately, with high cohesion and low coupling, we are able to reduce the number of classes being dependent on any single class. This implies that there are less possibilities for changes as each responsibility is an axis of change. Ultimately, our design ensures that changes to one class are less likely to affect another class.

2.2 Open-Closed Principle

In our design, the open-closed principle is mainly demonstrated through two interfaces, MenuItem and Menu.

Menu

Through abstraction, we generalized a menu to consist of 6 methods. Create, remove, update, display, getItem and write which are applicable to all types of menu. Subsequently, we used an abstract class to implement the interface and finally, the sub classes (Main Course menu, Drinks menu, etc) to extend from the abstract class.

a. *Closed for modification*

   Through the use of an interface as an abstraction layer, the menu class is closed for modification as it does not permit any type of implementation.

b. *Open for extension*

   Through the use of the menu as an abstraction layer, we allow the programmer to change what a menu does (by implementing these methods in the subclass) without having to change the source code of the menu interface. For example, the drink menu is able to display its item in a particular way (Showing whether a drink is hot or cold) without having to change the menu interface. Additionally, the abstract class which implements the menu is open for extension which allows other ways of displaying and creating the various menus.

Ultimately, this allows changes made to the subclass (Main course menu, Dessert menu,etc) without affecting how the main or other classes such as the order class uses it.This minimizes the amount of changes that has to be made to the overall source when a change/ new menu item is created.

2.3 Liskov Substitution Principle

The Liskov Substitution Principle states that a user of a base class should continue to function properly if a derivative of that base class is passed to it. The team ensured that an object of a superclass is replaceable by objects of its subclasses without changing the behaviour of the program. Notably, we ensured that 2 conditions are satisfied.

1. *Pre-conditions of subclass are not stronger than the base class*

   Through the use of an interface, we ensure that the pre-conditions (inputs) are not stronger than the interface. For instance, by having our classes for specific menu items (main course, drinks, etc) implement the menu item interface, we are able to ensure that this criteria is satisfied because the subclass implementing the interface must conform to these exact number of input arguments as the interface.

2. *Post-conditions of subclass are no weaker than the base class method*

   Similarly, this criteria is ensured through the use of a menu item interface. This is because the return type of the subclasses (main course, drinks, promotional set package and dessert) will have to match the return type of the interface. Additionally, the implementation of the interface method is similar across all 3 methods. For example, regardless of the subclass being used, the display method would always print the details

pertaining to that subclass. Hence, even if the menu item interface is substituted for a subclass, the user can continue to function as usual because the behaviour of the method of the subclasses are similar to that of the interface.

2.4 Interface Segregation Principle

Interface Segregation Principle states that classes should not depend on interfaces that they do not use. The team ensured that in our design, the interfaces were segregated according to their specific needs.

For example, the MenuItem interface was implemented by the GenericMenuItem abstract class for the sole purposes of allowing for modifications to specific menu items. The abstract methods of the MenuItem interface was used by the GenericMenuItem abstract class and subsequently, the sub class (Main course, dessert, drinks) which inherits the abstract class. Ultimately, in our design, the interfaces are segregated specifically to the needs as **all abstract** methods are **used** by and hence **relevant** to **all** the classes.

2.5 Dependency Injection Principle

The Dependency Injection Principle (DIP) is a key design consideration as it allows the higher level modules to be independent of lower level modules such that any changes in the lower modules will not affect the higher module. The team satisfied the DIP through the use of 2 interfaces, **menu item** and **menu** which acts as an **abstraction layer** between the high level modules (the main function) and the low level modules such as main course menu and main course. This is done through 2 main steps

1. *High level modules should not depend upon low level modules, and that both should depend upon abstractions*.

   In our design, the main function (high level module) only knows what the low level module (main course menu, drinks menu, etc) wants to achieve through the menu interface (abstraction layer). This is because the menu interface only provides the methods without any implementation. The same principle applies to the menu item.

   Through the use of polymorphism subsequently, dynamic binding can be done which allows for the main function to invoke the correct subclass method without having to know the actual object type (main course menu, etc). For instance, by invoking the display method of the menu object, the correct display method is called based on the actual object created at runtime. This allows for the main to be independent of the low level modules.

2. *Abstractions should not depend upon details, and details should depend upon abstractions.*

   In our design, the higher level modules are only concerned with the usage of the abstraction rather than how it is implemented. Similarly, the lower level modules (main menu course, dessert, etc..) simply follow the abstraction and implement it correctly. For instance, the focus of the main course menu (lower level module) is simply to implement these various methods such as display, create, update.

Ultimately, by applying the Dependency Injection Principle, we are able to minimize potential changes as high level modules such as the main and order class depend only on the abstraction layer and not the low level modules. This means that changes made to the source code of the lower level modules will not affect the high level modules.

3. OBJECT-ORIENTED CONCEPTS

3.1 Abstraction

Abstraction reveals the essential characteristics of a class and hides unnecessary details and irrelevant data about an object or an instantiated class, reducing the complexity of the code. This allows other classes to use the abstracted code without much modification as they would share the same conceptual boundaries[1].

The creation of new classes in our system were all performed with abstraction in mind.

For example, the creation of the Staff class is a form of abstraction as it allows other classes to create or retrieve Staff information without knowing the details of this method.

Another example of abstraction would be the creation of a *GenericMenuItem* class as it allows for extensibility in the future, in which new menu types can be added without modifying the method that calls the classes, reinforcing the Open-Closed Principle.

3.2 Encapsulation/Information Hiding

Encapsulation restricts direct access to certain details and implementation of a class from users, protecting an object's private data. In short, the entities created have private attributes which are only accessible through the get and set methods, making them a read-only class. This prevents other classes from directly accessing the private data[2].

---

[1] *What is Abstraction in OOPs? Java Abstract Class & Method*. (n.d.). Retrieved 12 November 2021, from https://www.guru99.com/java-data-abstraction.html

[2] *OOP Concept for Beginners: What is Encapsulation – Stackify*. (n.d.). Retrieved 12 November 2021, from https://stackify.com/oop-concept-for-beginners-what-is-encapsulation/

For example, the Order class has private attributes such as orderID, tableID, customer, etc. As they are private, they are only accessible through their respective getter methods. As such, Order class has full control of what is stored in its private fields and how to react to messages received by other classes.

3.3 Inheritance

Inheritance is the derivation of a class from another class for a hierarchy of classes that share a set of attributes and methods. This allows the derived child class to reuse the parent's code as the child has a "is-a" relationship with the parent[3].

This is implemented in several classes in the design. For example, the *MainCourseMenu*, *DrinkMenu* and *DessertMenu* inherit attributes and methods from their parent class, the *GenericMenu* class. For instance, the child classes can implement common methods such as the *getItem()* method which is inherited from the parent class, encouraging code reuse.

The same is applied to GenericMenuItem as the parent class of *Drink*, *Dessert* and *MainCourse*. They are able to use methods which are inherited from the parents and override any method as needed to suit their more specific needs.

3.4 Polymorphism

Polymorphism is the ability of an object reference being referred to different types. That is, a method through a superclass can result in different actions in the subclasses, depending on the object using the method. The team used polymorphism to shape the design of our restaurant system and fulfil design principles such as the Open-Closed Responsibility Principle and Liskov Substitution Principle.

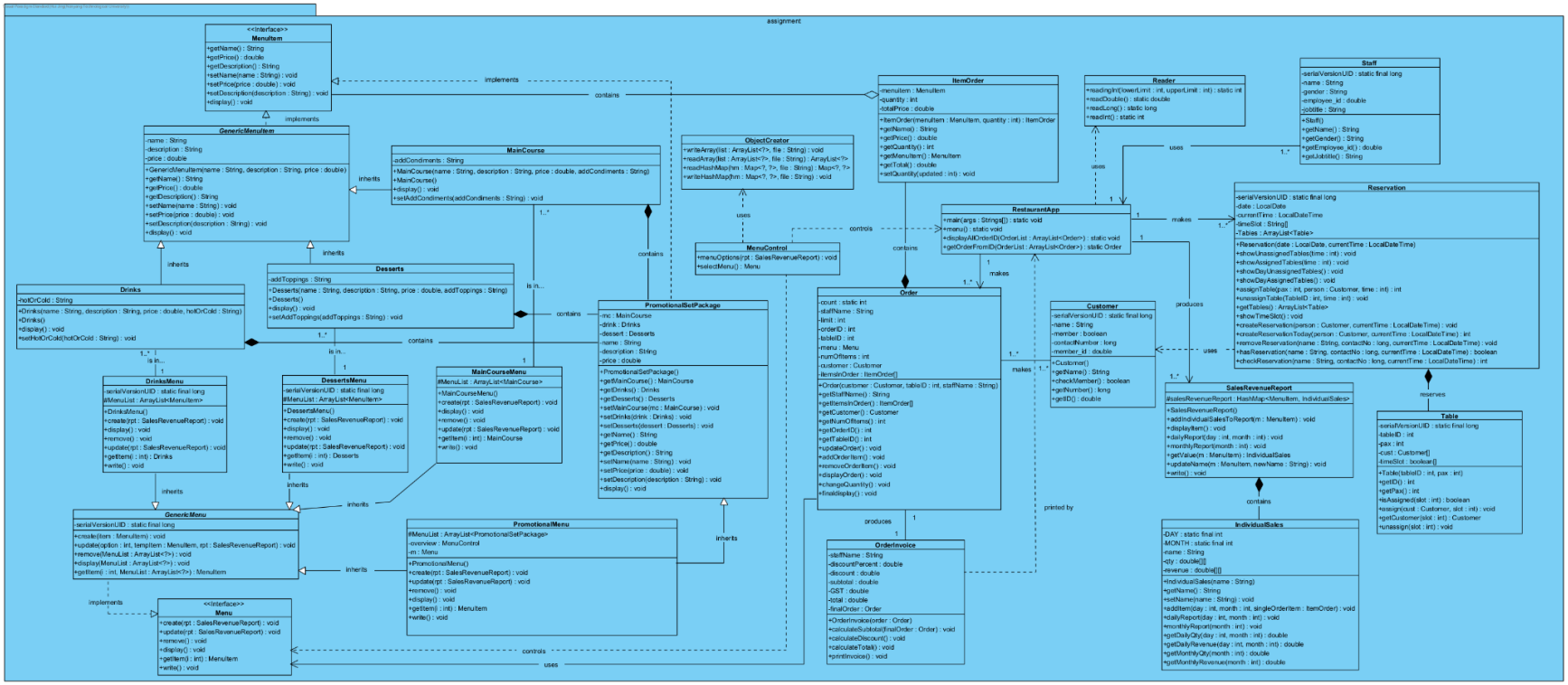
For example, subclasses of the GenericMenuItem class, such as the MainCourse, Desserts and Drinks, can override methods from its parent class such as getName(), getPrice() and display(), which are called by its child classes. This is done by defining a method with exactly the same signature and return type. For example, the display method of the GenericMenuItem class can be overridden by its subclasses to display a more catered view of the sub-menus e.g. having an additional addToppings column in the display for Desserts. Hence, the same method name and signature can cause different actions to occur, depending on the type of object on which the method is invoked. This facilitates the adding of new classes to a system with minimal modifications to the system's code, allowing for future extensions to be done.

---

[3] *OOP Concept for Beginners: What is Inheritance? – Stackify*. (n.d.). Retrieved 12 November 2021, from https://stackify.com/oop-concept-inheritance/

4. CLASS DIAGRAM AND SEQUENCE DIAGRAM

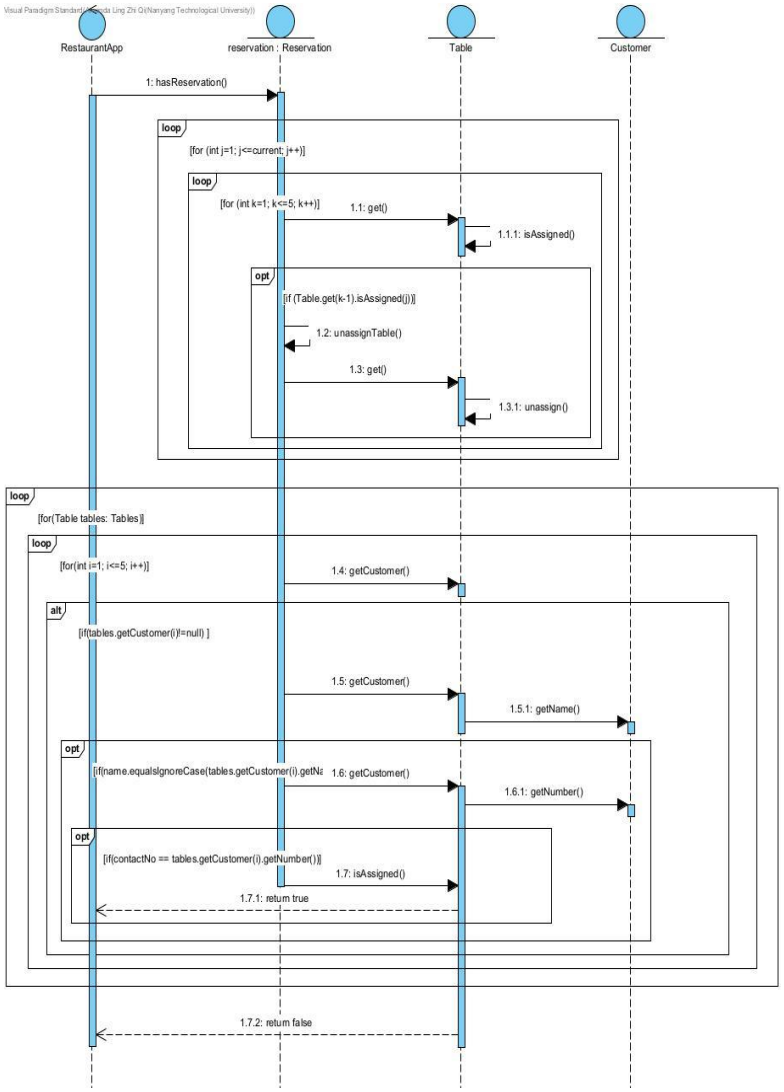
4.1 UML Class Diagram



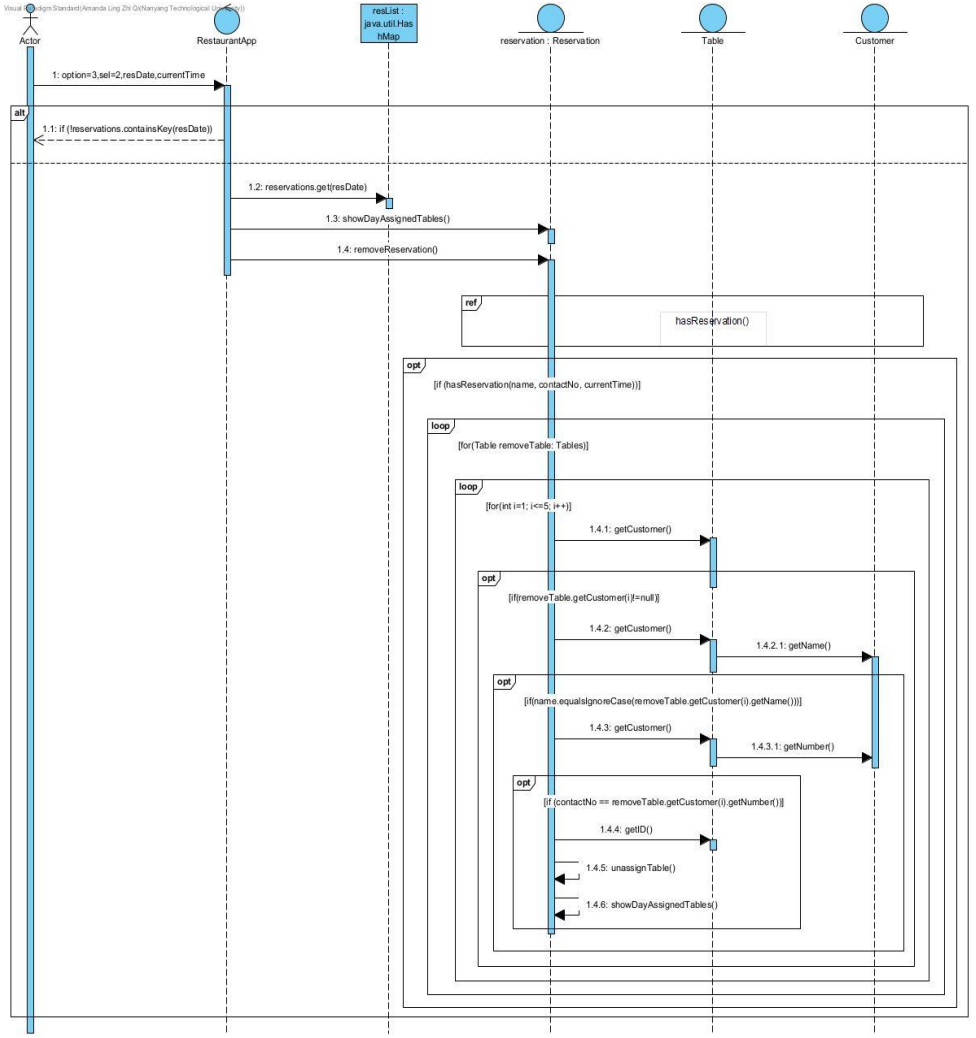
For a clearer image, please click on the link below: https://drive.google.com/file/d/1dnTZr0kAEmgIJXW-9oo78JTT02UeAmPN/view?usp=sharing

4.2 UML Sequence Diagram

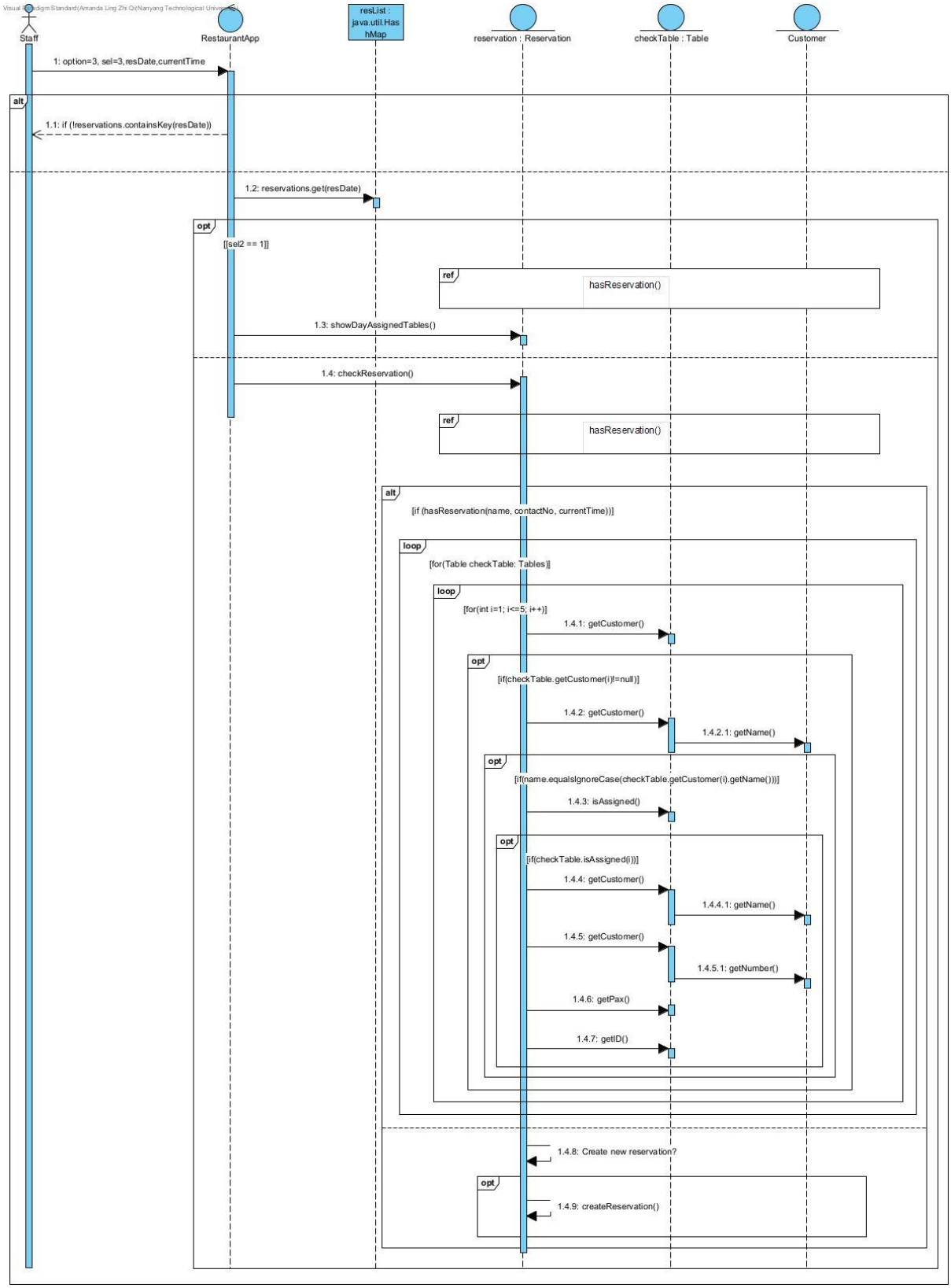
4.2.1 HasReservation



4.2.2 Remove Reservation



4.2.3 Check reservation



For a clearer image, please click on the link below:

https://drive.google.com/drive/folders/18ml7IyagImFJwox-ZkRlyPoR4nSheZyj?usp=sharing

5. TEST CASES

1) Exception handling

User enters a string and an invalid number:

```
Please Enter Today's Date (in format yyyy-MM-dd):
string_value_instead_of_date

Please Enter A Valid date in the following format yyyy-MM-dd
999

Please Enter A Valid date in the following format yyyy-MM-dd
2021-11-11
date: 2021-11-11
```

Item No. out of range: Drink menu

```
-------------------------------------------------------------------------------------------------
No Name                         Description
-------------------------------------------------------------------------------------------------
1  Coke Float                   Coke topped with vanilla ice-cream
2  Iced Vanilla Latte           Vanilla coffee with lots of milk
3  Singapore Sling              Gin-based sling cocktail
4  Apple Cider                  Sparkling apple drink
5  Fizzy Blueberry              Sparkling blueberry drink
6  Caramel macchiato            Espresso coffee drink w caramel & cream
7  Orange juice                 Fresh-made orange juice
-------------------------------------------------------------------------------------------------
*** Select the item you wish to update ***
-5

Menu Item out of range please enter a valid index
99
Menu Item out of range please enter a valid index
abc
*** Please enter a valid number ***
5
*** Select your choice ***
1. Name
2. Description
```

No existing order (Exception handling)

```
----------------------------------------
       WELCOME TO FUSION BISTRO
----------------------------------------
1. Create/Update/Remove/View menu
2. Create/Update/View order
3. Create/check/remove reservation item
4. Print order invoice
5. Print sale revenue report
6. Quit
----------------------------------------

*** Please enter an option ***
2

-----------------
1. Create Order
2. Update Order
3. View Order
4. Quit
-----------------

*** Please enter an option ***
2


***** Order IDs in system: *****
Order ID:     TableID:     Customer Name:
1             2            no_reservation

Please enter the ID of the order you want to view (Enter 0 to quit)
5

Error: an Order ID that does not exist was entered
```

No existing order (Error handling)

```
----------------------------------------
        WELCOME TO FUSION BISTRO
----------------------------------------
1. Create/Update/Remove/View menu
2. Create/Update/View order
3. Create/check/remove reservation item
4. Print order invoice
5. Print sale revenue report
6. Quit
----------------------------------------

*** Please enter an option ***
4


***** Order IDs in system: *****
Order ID:     TableID:     Customer Name:
1             2            no_reservation

Please enter the ID of the order you want to view (Enter 0 to quit)
3

Error: an Order ID that does not exist was entered
```

Corresponds to the order made by no_reversation walk in customer at the start

2) Check Reservation

a) Full reservation

```
###### LIST OF EMPTY TABLES (BY TIME) ######
Date: 2021-12-12        Time Slot: 18.00-20.00
-------------------------------------------------
Table ID        Pax
-------------------------------------------------
1               2
2               4
3               6
4               8
-----------------------------------------------

Please enter number of people at the table:
10

-------------------------------------------------------------------------------
CREATE RESERVATION: UNSUCCESSFUL
Error Message: All tables are full or they do not meet your seating capacity.
-------------------------------------------------------------------------------
```

Pax 10 is not available

Pax chosen is 10, however the table for pax 10 is already occupied

3) Update and remove promotional package

Update drinks for promotional set 5:

```
Set number : 4
Name of set: set A
Set includes: Le Fusion Burger, Singapore Sling, Tiramisu
Price of promo set: 22.24

Set number : 5
Name of set: Sample Set
Set includes: Le Fusion Burger, Coke Float, Le Fusion Cho
Price of promo set: 20.24

------------------------------------------------------------
*** Select the item you wish to update ***
5
```

```
*** Select your choice ***
1. Name
2. MainCourse
3. Drinks
4. Dessert
5. Price
6. Quit
3

------------------------------------------------------------------------------
No Name                         Description
------------------------------------------------------------------------------
1  Coke Float                   Coke topped with vanilla ice-cream
2  Iced Vanilla Latte           Vanilla coffee with lots of milk
3  Singapore Sling              Gin-based sling cocktail
4  Apple Cider                  Sparkling apple drink
5  Fizzy Blueberry              Sparkling blueberry drink
6  Caramel macchiato            Espresso coffee drink w caramel & cream
7  Orange juice                 Fresh-made orange juice
------------------------------------------------------------------------------
*** Select the item you wish to include ***
7
```

Verification of updated promotional set 5:

```
Set number : 4
Name of set: set A
Set includes: Le Fusion Burger, Singapore Sling, Tiramis
Price of promo set: 22.24

Set number : 5
Name of set: Sample Set
Set includes: Le Fusion Burger, Orange juice, Le Fusion
Price of promo set: 22.24

------------------------------------------------------------


------------------------------------------
```

Item successfully changed

Removing promotion set 5: Sample set

```
Set number : 3
Name of set: Fusion Bistro Promo Set 3
Set includes: Assorted Sushi Platter, Fizzy Blueberry, Waffles Delight
Price of promo set: 29.92

Set number : 4
Name of set: set A
Set includes: Le Fusion Burger, Singapore Sling, Tiramisu cake
Price of promo set: 22.24

Set number : 5
Name of set: Sample Set
Set includes: Le Fusion Burger, Orange juice, Le Fusion Chocolate Fondue
Price of promo set: 22.24

------------------------------------------------------------------------------

*** Select the item you wish to remove ***
5
```

Verification that item has been successfully deleted:

```
------------------------------------------------------------
Set number : 1
Name of set: Fusion Bistro Promo Set 1
Set includes: Le Fusion Burger, Coke Float, Tiramisu cake
Price of promo set: 15.84

Set number : 2
Name of set: Fusion Bistro Promo Set 2
Set includes: Crispy Pork Roulade, Apple Cider, Fiery Volc
Price of promo set: 26.32

Set number : 3
Name of set: Fusion Bistro Promo Set 3
Set includes: Assorted Sushi Platter, Fizzy Blueberry, Waf
Price of promo set: 29.92

Set number : 4
Name of set: set A
Set includes: Le Fusion Burger, Singapore Sling, Tiramisu
Price of promo set: 22.24

------------------------------------------------------------
```

Item successfully removed

Link to video: https://www.youtube.com/watch?v=cib7P7yLUC4