








CZ2002 Project Report

Declaration of Original Work 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	Lab Group	Signature/Date
Mukizhitha Rajkumar	CZ2002	SS6	 13/11/2021
Nataniel Chng Cheng Bin	CZ2002	SS6	 13/11/2021
Chang Tze Chuan	CZ2002	SS6	 13/11/2021
Ye Xin	CZ2002	SS6	 13/11/2021
Yap Wee Kiat	CZ2002	SS6	 13/11/2021

Declaration of Original Work CE/CZ2002 Assignment	1
Introduction	3
Approach Taken	3
Entity-Control-Boundary	3
Principles & OO Concepts Used	4
Single Responsibility Principle	4
Open-Closed Principle	4
Liskov Substitution Principle	4
Loose Coupling	5
Encapsulation	5
Interface Segregation Principle	5
Aggregation & Composition	5
Inheritance	6
UML Class Diagram	7
UML Sequence Diagram	8
Check Booking	8
Check/Remove Reservation	9
Assumptions Made	9
Test Cases	10
Link To Demo	11

Introduction

The objective of this assignment is to model, design and develop an Object-Oriented (OO) console-based application while applying relevant OO concepts taught in the course using the Java object oriented programming language.

The Restaurant Reservation and Point of Sale System (RRPSS) is an application to computerise the processes of making reservations, recording orders and display of sale records. This application is designed for restaurant staff.

Approach Taken

Entity-Control-Boundary

The RRPSS uses the entity-control-boundary stereotype class framework that separates the application into three logical components: entity, control and boundary.

The boundary communicates with the controller, and the controller interacts with the entities. The boundary handles user inputs and actions, while the controller manages the logic and execution of different functionalities of the application. The entity represents objects and persistent system data of the application.

The entities of our application are: MenuDB, SetPackage, MenuItem, ItemType, Staff, Order, RevenueDB, SaleRevenue, TableDB, Table, Reservation, ReservationDB. The controllers of our application are: OrderManager, MenuItemManager, TableManager, SetPackageManager, MenuManager, TableManager, ReservationManager. The boundaries of our application are: SaleRevenueApplication, OrderApplication, RRPSS, ReservationApplication

Principles & OO Concepts Used

Single Responsibility Principle

The Single Responsibility Principle (SRP) states that every class, function or module in an application should only have one responsibility. For example, the entity MenuDB is solely responsible for loading and updating changes of the database onto the application directly. The controller MenuItemManager is responsible for managing and manipulating the MenuItem object and all relevant data of that entity. This principle was applied to entities and ensures maintainability as it is easy to identify problem modules if needed.

Open-Closed Principle

The Open-Closed Principle (OCP) states that a module should be open for extension but closed for modification. In our application, the controllers have an association relationship with the entities which allows the use of the entity's methods and attributes without modifying the entity. For example, the MenuItemManager and SetPackageManager controllers are able to use the methods and attributes of the MenuDB entity without modifying it.

Liskov Substitution Principle

The Liskov Substitution Principle (LSP) states that objects of a base class should be substituted with objects of its subclasses without noticeable difference. In our application, the database files are converted into an array list of objects before being manipulated by the other classes of the system. Before closing the application, updates made to the array list are written to the database file using dedicated methods. Despite a change in data type, the core behaviour of database files and array lists are the same -- a collection of data. LSP is demonstrated because a user will not be able to notice this change in data format.

Loose Coupling

An effect of SRP and OCP is loose coupling, when elements of the system have minimal knowledge of internal details of other elements. They will be able to function as intended without much knowledge. For example, in the MenuDB class, there is no need for menu items to be associated with a particular set of orders. Instead, the Order class will record such information, which keeps coupling as low as possible.

Encapsulation

Encapsulation refers to keeping values and methods of an object hidden within its class to prevent illegal direct access by other classes. The entity attributes of our application are private, while getter and setter methods are public. This ensures that users of the class can get or set an attribute without actually knowing how the class works. For example, in the ReservationDB class, the database file is private. To view the contents of the database file, the ReservationManager class will have to use the getReservationList() method.

Interface Segregation Principle

The Interface Segregation Principle states that, “Many client specific interfaces are better than one general purpose interface.” The RRPSS segregates the running of a restaurant into separate interfaces for managing menu items, reservations, orders, providing invoices and generating sales reports. Additionally, the modules only inherit relevant classes to their responsibilities. This keeps the application simple to read, maintain and flexible for future updates.

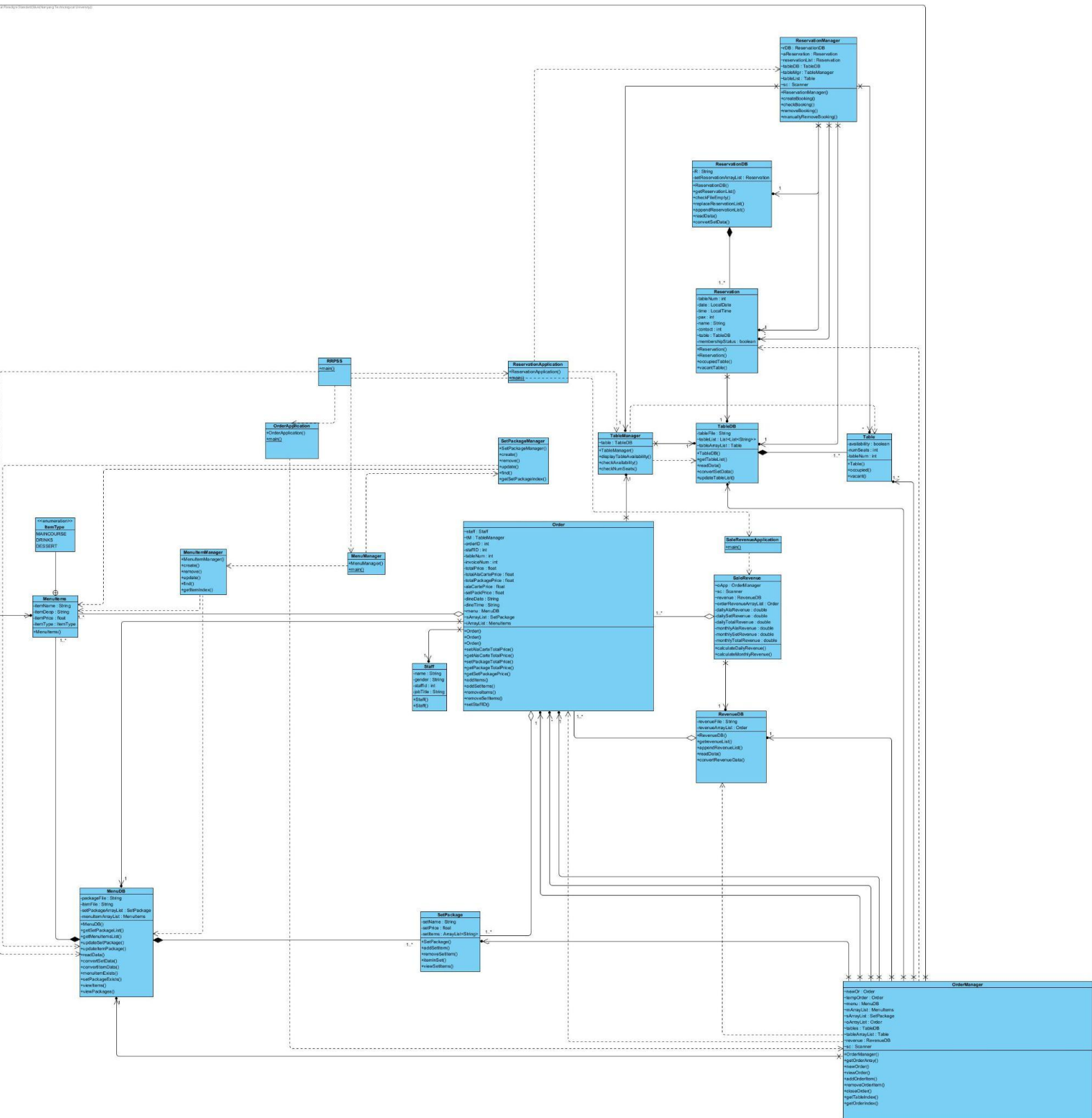
Aggregation & Composition

In the RRPSS, there are composition and aggregation relationships between classes. For example, the MenuDB class is composed of MenuItem and SetPackage objects. Meanwhile, the Order class is an aggregation of MenuItem, as MenuItem are added to Order via reference. There is a whole-part relationship between MenuDB, MenuItem and Order. A menu cannot exist without its menu items, and an order cannot be made without having at least one menu item.

Inheritance

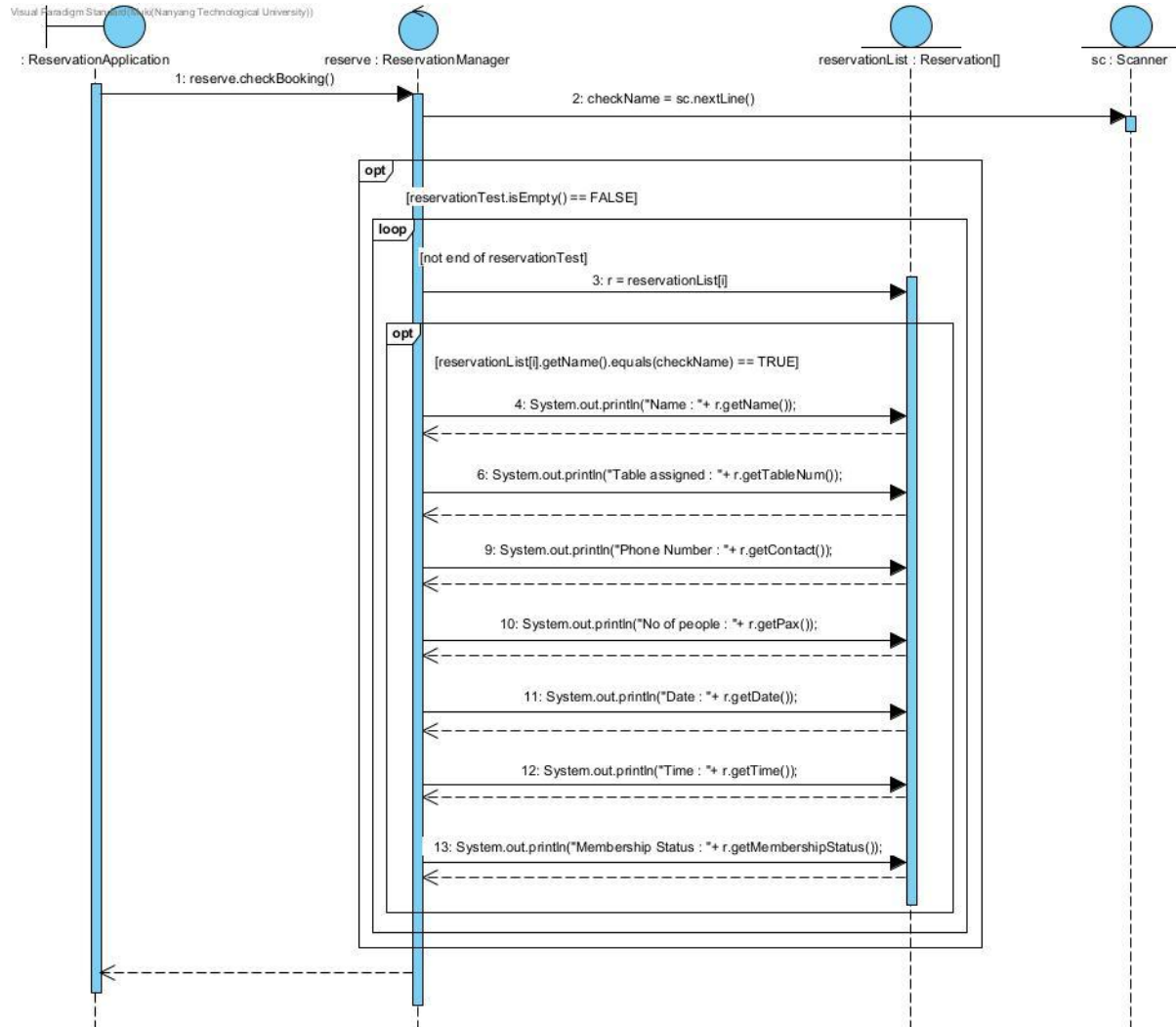
In our application, subclasses are able to use the attributes and methods of superclasses. For instance, the SaleRevenue class uses the methods from RevenueDB to manipulate the data entries indirectly. The OrderManager class also makes use of RevenueDB's methods and attributes. This concept is useful as code does not have to be duplicated to be used among different classes. Additionally, it ties in closely with encapsulation as the original database files are not accessed directly by other classes.

© Pinyang Sun and Wuyang Yang, Tsinghua University

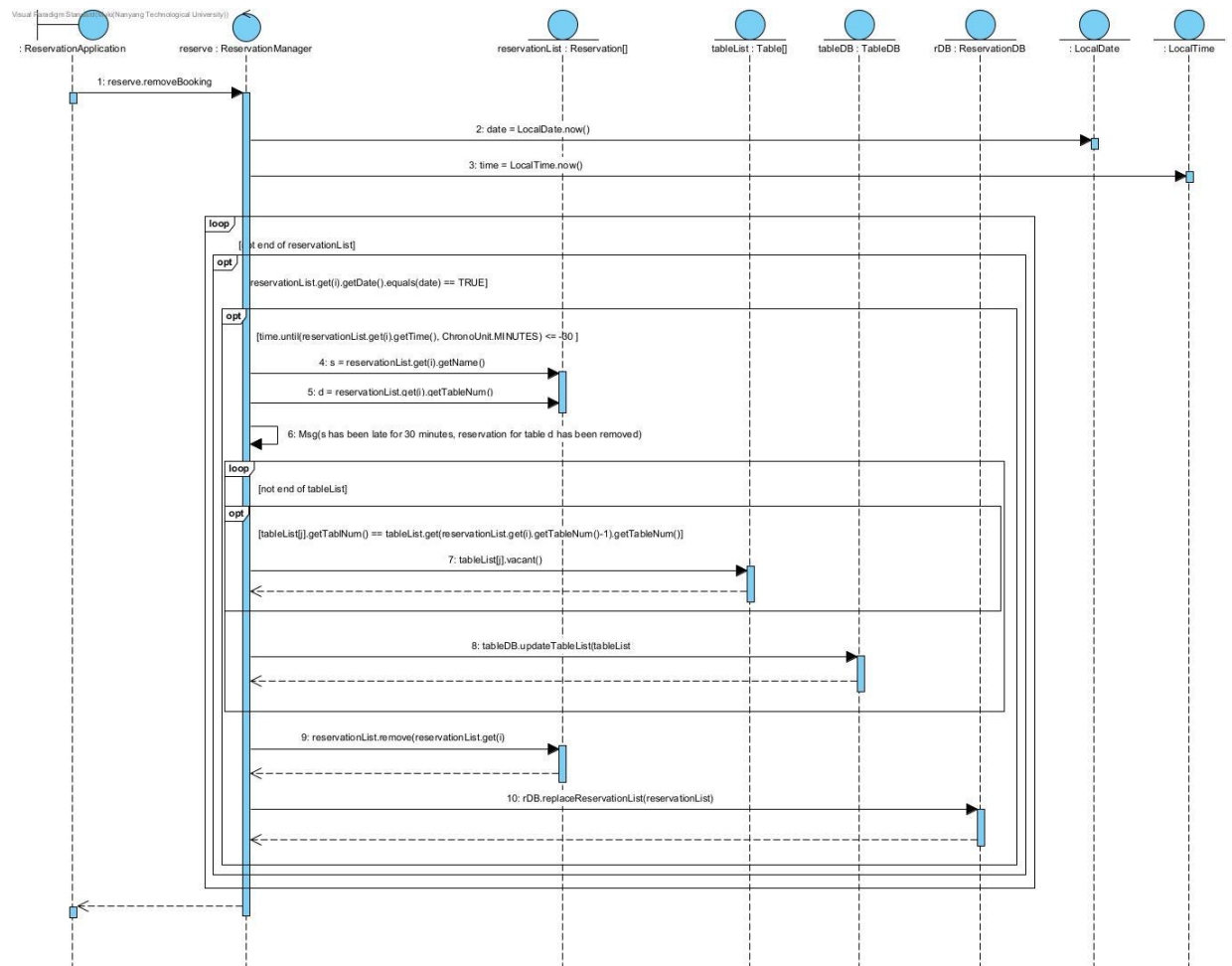


UML Sequence Diagram

Check Booking



Check/Remove Reservation



Assumptions Made

1. We assume that there is no need to change the staff of the restaurant
2. The restaurant does not accept walk-ins, only reservations are accepted
3. Customers of the same group cannot be seated in separate tables
4. At any point, only one reservation can be created under the customer's name
5. Once the invoice is printed, the order is closed and customers can no longer add to order.
6. Sale Revenue consists solely of the total revenue generated from the sale items non-inclusive of GST and other charges.
7. We assume that the databases' file path names and location don't change.
8. We assume that the customer makes payment immediately after the order invoice is printed

Test Cases

All tables are reserved

```
=====
1: Manage Menu
2: Manage Orders
3: Manage Reservation
4: Generate Sales Report
5: Exit
=====
Enter option:
3

=====
1: Place Reservations
2: Remove Reservations
3: Check Reservations
4: View table availability
5: Back
=====
Enter option:
1

Fully reserved! Unable to create new reservations.
=====
1: Place Reservations
2: Remove Reservations
3: Check Reservations
4: View table availability
5: Back
=====
Enter option:
4
|
Table:    Seats:    Available?
1         2         NO
2         2         NO
3         4         NO
4         4         NO
5         6         NO
6         6         NO
7         8         NO
8         8         NO
9         10        NO
10        10        NO
```

Calculate monthly
revenue

```
=====
1: Calculate daily revenue
2: Calculate monthly revenue
3: Back
=====
Enter option:
2

Enter Year(yyyy):
2021
Enter Month(MM):
11

Month: 11
AlaCarte Revenue: 135.9
Set revenue: 107.0
Total revenue: 242.9
```

Link To Demo

<https://youtu.be/dhw4P48OGxg>