

NANYANG TECHNOLOGICAL UNIVERSITY

NANYANG BUSINESS SCHOOL

CZ2002 OBJECT-ORIENTED PROGRAMMING AND DESIGN



Restaurant Reservation and Point-of-Service System

Group Project Report

Project Submitted by:

Team 1

Tan Yu Ling (U2022392H)

Teresa Zhang Han Yu (U2022886C)

Venkataraman Sidhaarth (U2021808J)

Tan Shu Hua, Samantha (U2021180J)

Shenal Devinda Bandara Rajaguru (U2023670B)

APPENDIX B:




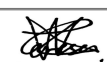
Attached a scanned copy with the report with the filled details and signatures.

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 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
Venkataraman Sidhaarth	CZ2002	SDDP3	
Tan Shu Hua, Samantha	CZ2002	SDDP3	
Tan Yu Ling	CZ2002	SDDP3	
Teresa Zhang Han Yu	CZ2002	SDDP3	

Shenal Devinda Bandara Rajaguru
Important notes:

CZ2002

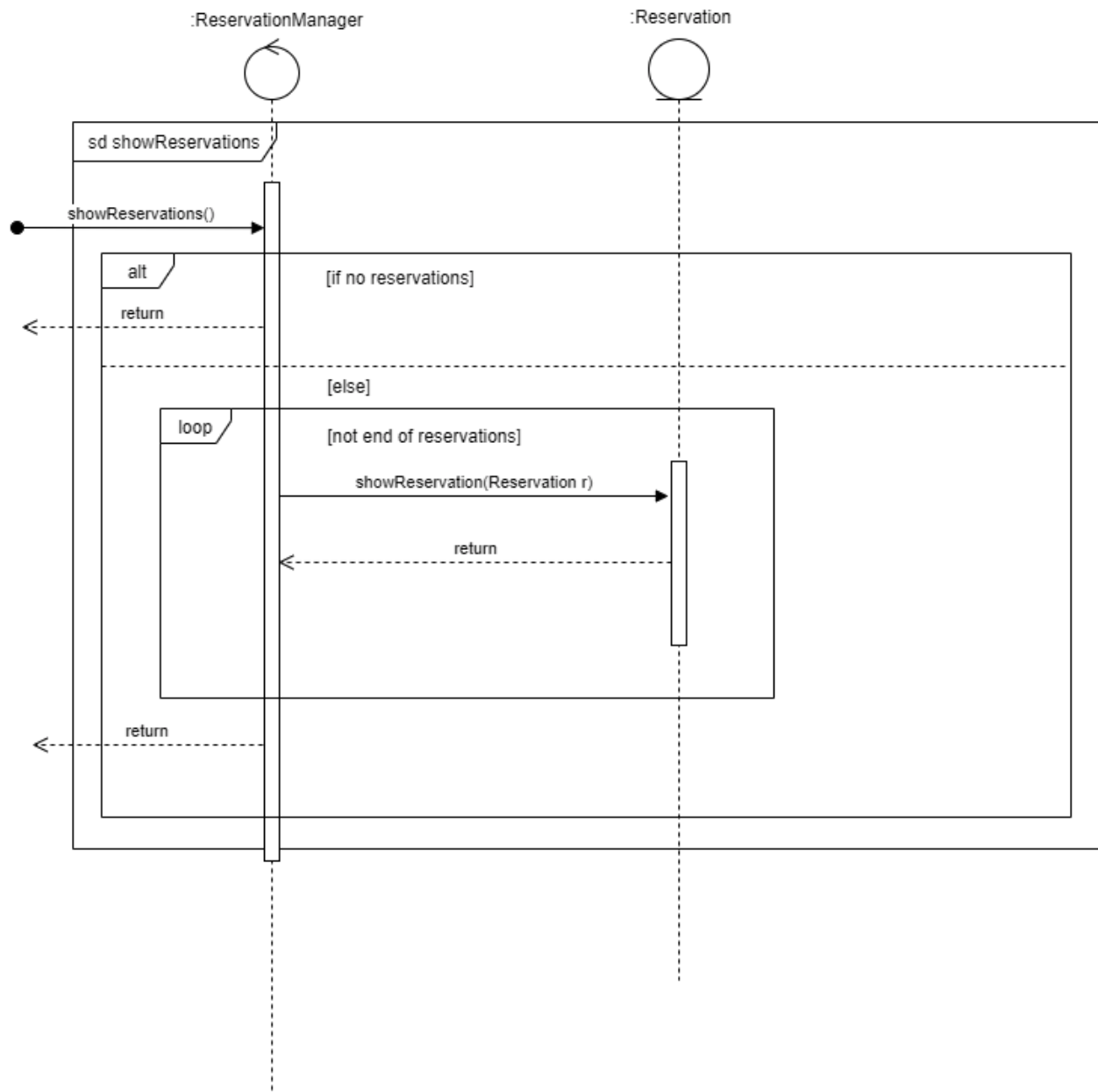
SDDP3



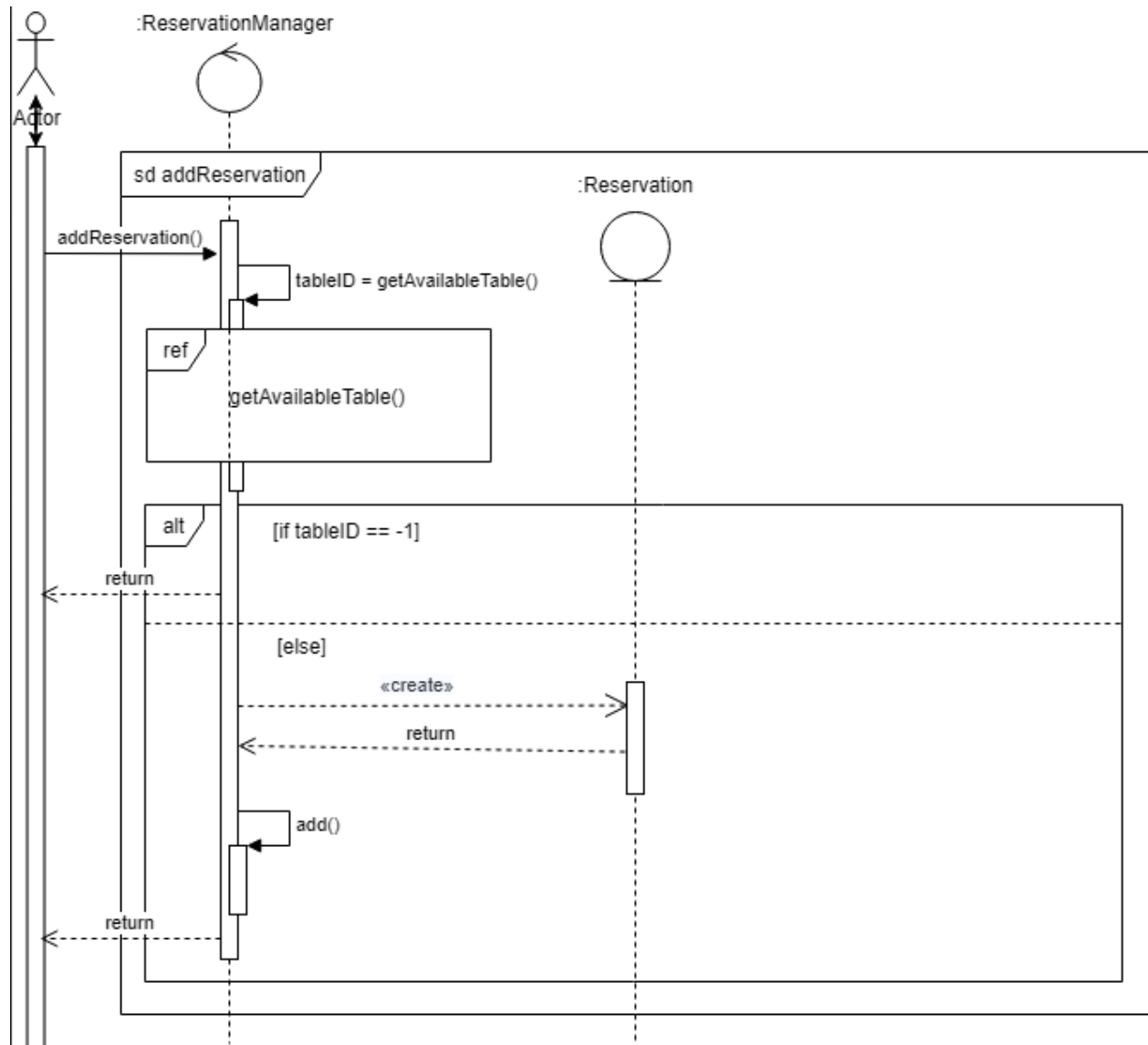
1. Name must **EXACTLY MATCH** the one printed on your Matriculation Card.

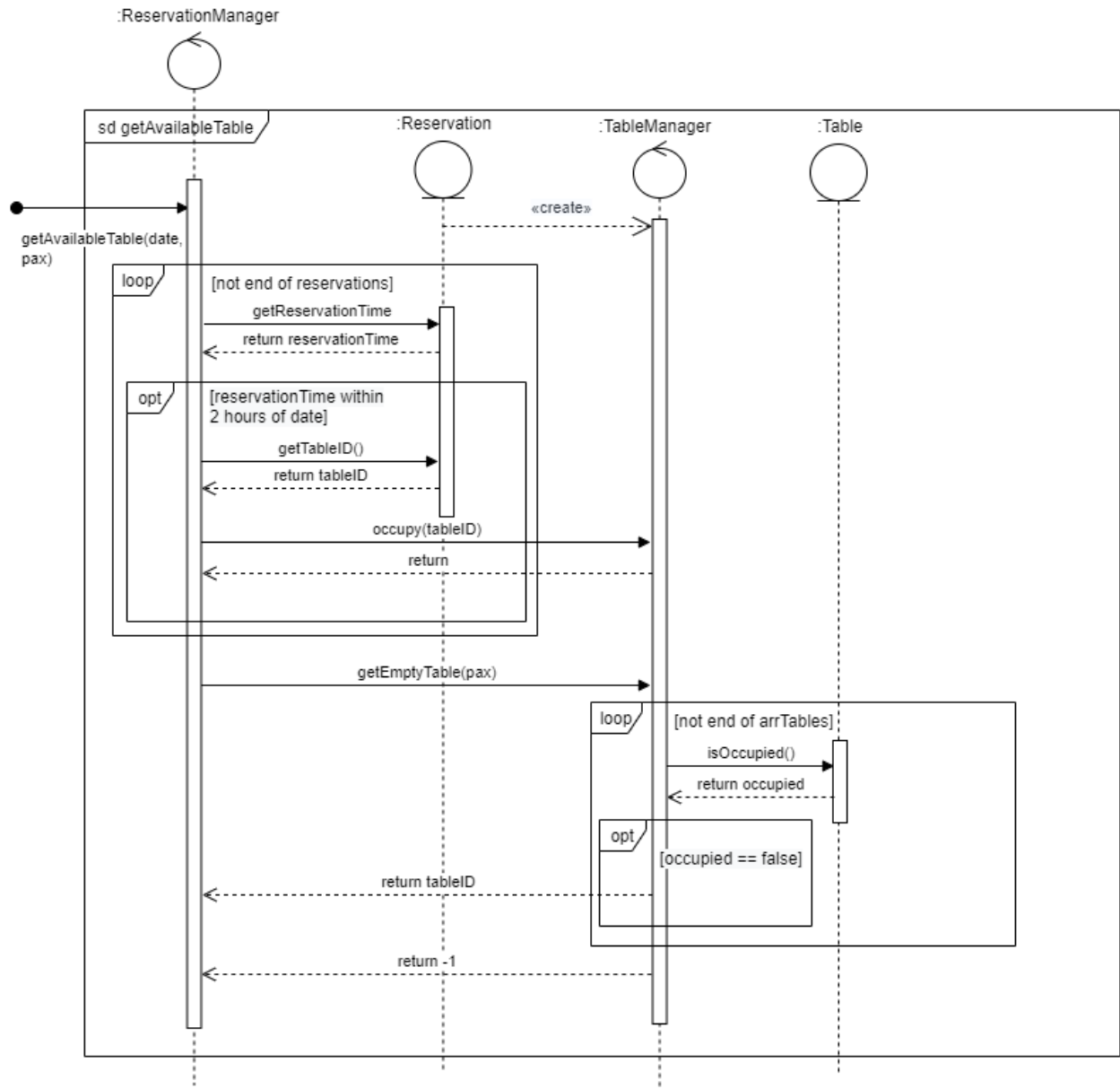
UML Sequence Diagrams

Show reservation:

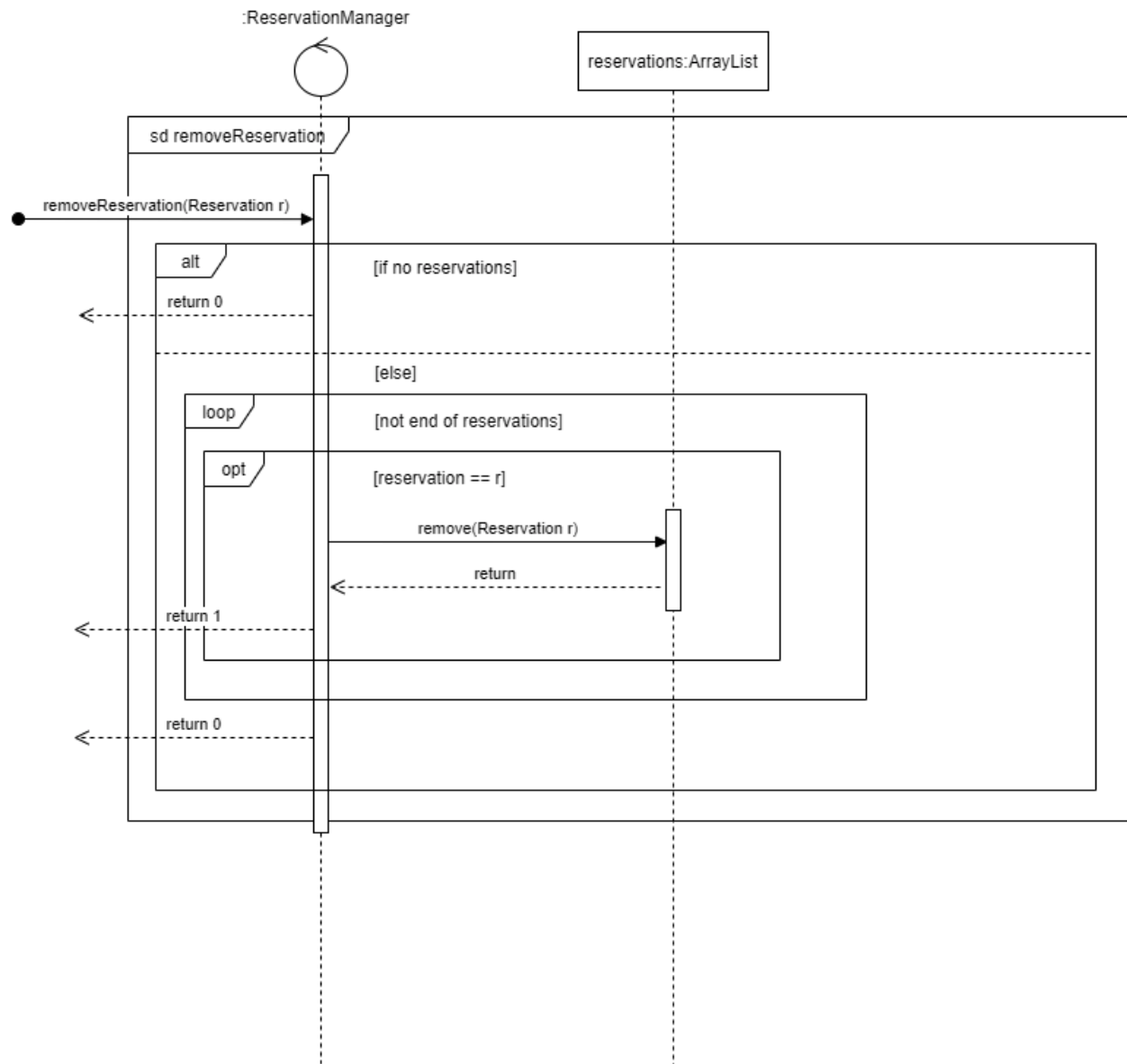


Add Reservation:

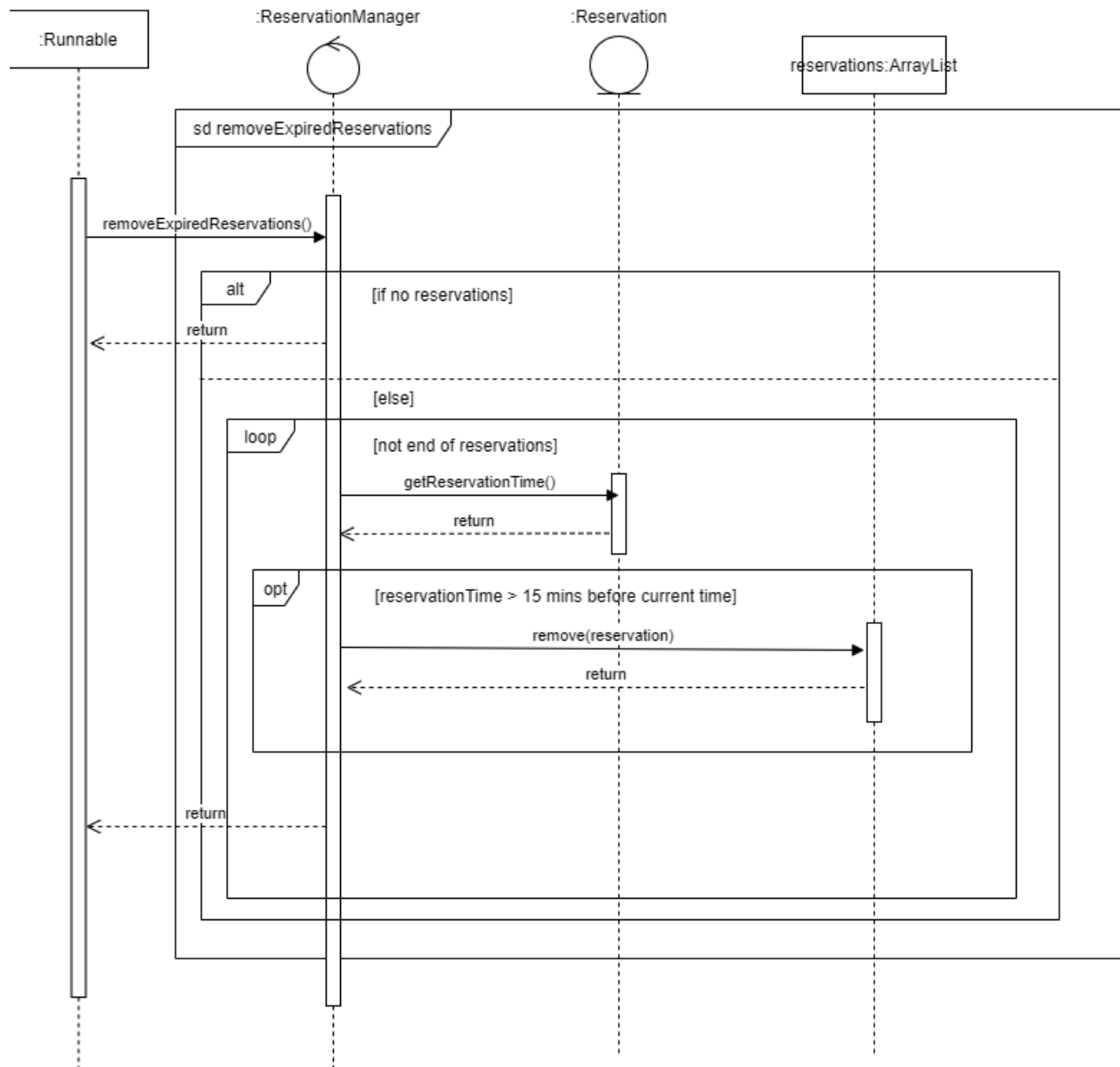




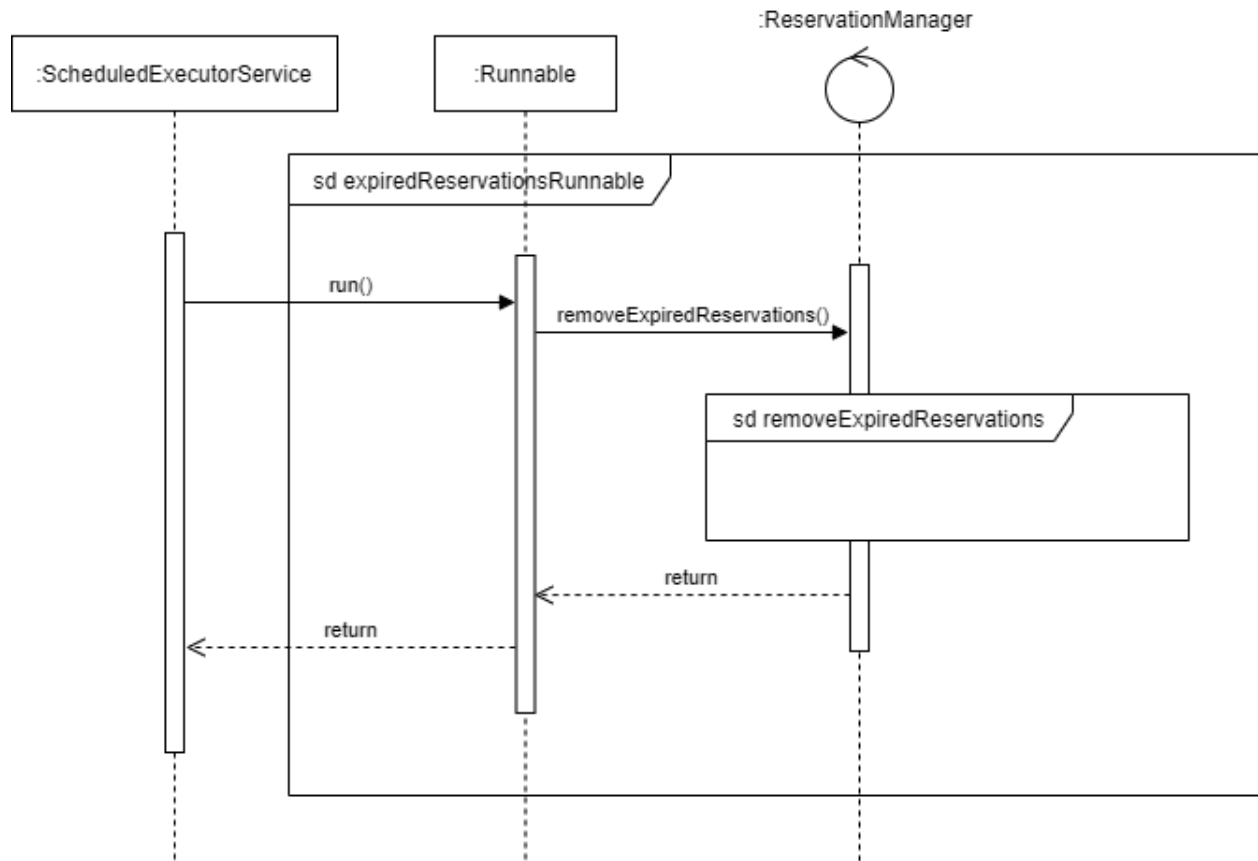
Remove Reservation:



Remove Expired Reservation:



Expired Reservation Runnable:



Design Considerations and Use of OO Concepts

1. Introduction

The purpose of this project is to create a Restaurant Reservation and Point of Sale Service for the use of Restaurant staff. In the creation of this system, object-oriented programming design concepts were used. The advantage of object-oriented programming is its ability to develop a system that is reusable, extensible, with high maintainability. To this end, we strove to apply the SOLID design principles in our application by making use of the OOP features of abstraction, encapsulation and more.

2. Design Architecture and Patterns

In developing our application, we utilised the Entity-Control-Boundary pattern in our design. Entity classes include Product, PromoProduct, Order, Reservation, and more. They are responsible for storing information necessary for the application. The Boundary class is our RestaurantApp class, the main function that users interact with. Control classes include OrderList, RestaurantManager, Menu, and TableManager. They serve as the go-between of boundaries and entities, and implement much of the higher-level logic.

3. Design Principles

1. Single Responsibility Principle

The Single Responsibility Principle states that each class should have only one responsibility. This is similar to the concept of having cohesive classes, where there are clear boundaries to what each class does. Each class was categorised into one of the three class stereotypes of Entity-Control-Boundary Pattern. Each class also has a clear role as to what function it plays in the context of the system. For example, Product is an entity class that stores the name, description, price and more of menu items. OrderList is a control class that facilitates the creation, update and removal of orders between user and entities. Although in real life an order would have to keep track of the products it includes, in our system the storage of product details remains only in instances of Product. OrderList does not include the details of how Product works. Thus, should there be a need to change either one of these classes, the other class will remain largely unaffected.

2. Open-Closed Principle

The Open-Closed Principle states that classes should be open for extension but closed for modification. This principle can be seen in the product part of the system. Classes Product and PromoProduct, for menu products and promotional products respectively, both extend the Item interface. If a new category of Item is created in the future, the system can be extended simply by creating this new class and implementing Item. This way, the effect on dependencies upon Product and PromoProduct would be minimised, while the new Item type can be integrated with Product and PromoProduct more easily.

3. Interface Segregation Principle

This principle encourages developers to create specific interfaces that play a single role, rather than one large, general-purpose interface. This project makes use of the Indexable interface. This interface includes two methods, getIndexbyID and generateID. It is only used for automating the process of generating a new ID. Indexable has been implemented into classes such as Menu and ReservationManager, where new IDs must be created whenever a new product or a new reservation is made. This interface simplifies the code and development process of these classes. Of course, this is only possible because the id generation function needed by both Menu and ReservationManager are identical. The function simply checks for and generates an unused ID. Otherwise, differing requirements in Menu could lead to changes in Indexable and ReservationManager, causing a larger problem.

4. Dependency Injection Principle

In essence, this principle necessitates that higher level modules should implement higher level logic based on abstractions from lower level modules. These abstractions should then determine how the detailed logic in lower level modules should be implemented. For instance, in the ReservationManager class, an available table must be found before a reservation can be created. The retrieval of an empty table is abstracted to a dependency on the TableManager class. The TableManager dependency is injected when ReservationManager instantiates TableManager class. Given a booking time and the number of people, TableManager implements the logic to return an available table ID. By injecting this dependency, ReservationManager does not have to contend with lower level information such as the availability and capacity of tables.

4. Assumptions

1. Reservations have a grace period of 15 minutes
2. Reservations can only be made 1 year in advance
3. Members carry a membership card that is checked manually by staff
4. Promotional Products cannot have a validity date that starts before current date
5. Promotional Products are valid up to only 1 year ahead
6. Only staff have access to the app. Hence, the validity of StaffID is not checked.

Link to Presentation Video

https://youtu.be/_iDRh0i3fNQ