

**NANYANG
TECHNOLOGICAL
UNIVERSITY**
SINGAPORE

LAB GROUP: SSP1

GROUP: 2

CZ2002: Object-Oriented Design & Programming

Assignment : Restaurant Reservation and Point of Sale System (RRPSS)

2021/2022 SEMESTER 1

by

Group Members		
S/N	Matriculation No.	Student Name
1.	U2021918D	Liang Zhiheng
2.	U2021547H	Shi Jiancheng
3.	U2022775F	Mak Qin Xiang
4.	U2021730A	Alexander Ng
5.	U2022732H	Marcus Teh



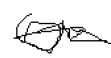


APPENDIX B:

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	Lab Group	Signature/Date
Liang Zhiheng	CZ2002	SSP1	
Shi Jiancheng	CZ2002	SSP1	
Mak Qin Xiang	CZ2002	SSP1	
Alexander Ng	CZ2002	SSP1	
Marcus Teh	CZ2002	SSP1	

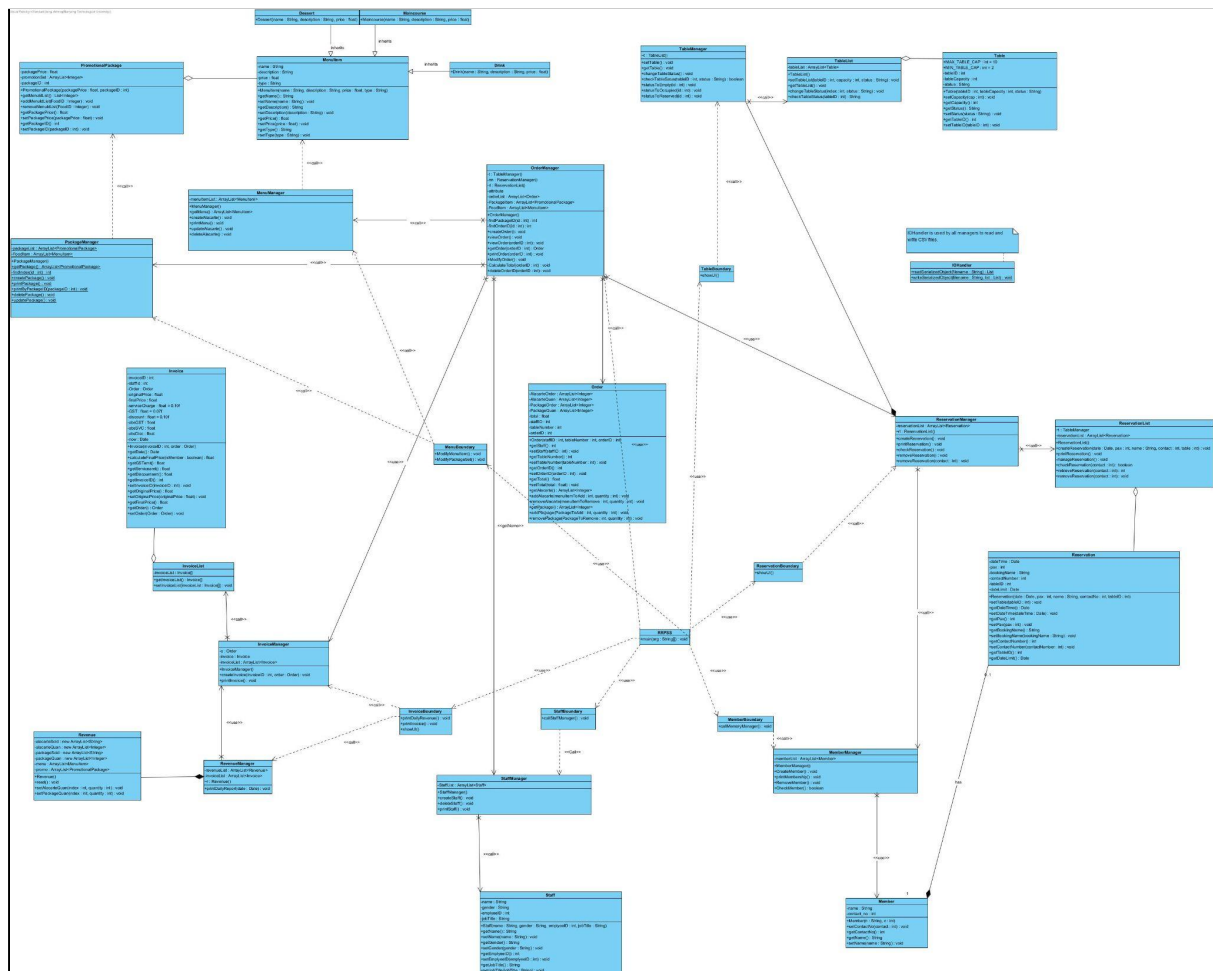
Assumptions Made	4
Detailed UML Class Diagram (If too blurry, actual vpp file is outside of report)	4
Design considerations	4
Solid Approach	5
Single Responsibility Principle	5
Open-Closed Principle	5
Liskov Substitution Principle	5
Dependency Injection Principle	5
Object-oriented Principles	6
Encapsulation	6
Abstraction	6
Inheritance	6
UML Sequence Diagram (Actual File is outside of report)	7
Testing	8
Reservation Manager	8
Member Manager	9
Staff Manager	9
Table Manager	10
Video Demonstration	11

Assumptions Made

Below are the assumptions taken into consideration when writing the code for the project.

- When the restaurant is fully booked in the timeslot, no more reservations can be made.
- When a table has been reserved it cannot be reserved again at the same time slot.
- The table reserved for a customer can fit the entire group of customers.
- If we were to remove a menu item, we assume that it is not in any promotional packages.
- The removal of menu items in order will be maximally be how many quantity of it is in the order

Detailed UML Class Diagram (If too blurry, actual vpp file is outside of report)



Design considerations

Our team avoided the symptoms of rotting design which are rigidity, fragility and immobility when making our application, the Restaurant Reservation and Point of Sale System (RRPSS). Designed the RRPSS application where we aimed to have a design with reusability , extensibility and maintainability in mind.

Solid Approach

Used SOLID design principle as much as possible as a guide when designing the RRPSS. This is to reduce dependencies so that any changes in one area of software will have minimal impact on the other parts.

Single Responsibility Principle

This principle ensures that there is no ‘God’ class where each class will only have one responsibility for one part of RRPSS. We have split the code into several controller classes which manage each different aspect of the restaurant. For our RRPSS, we have done so for example the “MenuManager” which will only be responsible for its menu functions such as creation , deletion and removal. Likewise for the other classes in our class diagram, it will only handle responsibilities pertaining to its class.

Open-Closed Principle

If the application is able to extend a class’s behavior without modifying it, it complies with this principle. This means that the class behaviour can be extended but the source code is set and cannot be changed. Our RRPSS follows this principle in the case of the “MenuItems” class. It is a superclass and different food types can be inherited from the “MenuItems” class. For instance “Drinks”, “Dessert” and “MainCourse”. Should the restaurant decide to sell alcohol, it can simply create a subclass “Alcohol” which inherits the methods from the “MenuItems” class.

Liskov Substitution Principle

The principle defines that objects of a superclass shall be replaceable with objects of its subclasses without breaking the application. We do not defy this principle in our code. For example, in the class “Drink” that inherits from MenuItem, the drink class behaves as expected of MenuItem and does not expect more from the user , or provide less to the user.

Dependency Injection Principle

This principle states that high level modules should not depend upon low level modules , both should depend on abstractions. Abstractions should not depend on details, details should depend upon abstractions. We followed this principle by implementing interfaces, to decrease the dependence of higher and lower level classes. This can be found in all our classes that we require to be serializable, which are mostly the entity classes that contain our data. Such examples are the MenuItem class and the Member class. For these classes, we implemented an interface Serializable which contains a method that allows the class to serialize the object data into a CSV String readable format. This method will be called by all the classes that interact with our CSV data files, and these classes can easily read data from the CSV file to an ArrayList, or write data from the ArrayList to the CSV file.

Object-oriented Principles

Encapsulation

Encapsulation is the binding of data and functions which operate on that data into a single class. It hides the private details of a class from the others and only exposes the functionality that is interfacing with it.

The concept of encapsulation is done by keeping the attributes in private so it cannot be accessed by other functions while the get and set methods are kept global to access the private data.

Abstraction

Abstraction is the main concept of object oriented programming. It involves hiding functions and attributes that the user does not need or know how to use which creates a more user friendly interface.

In our project, the user only needs to key in the choices on whether to update or delete or add items to the order, menu and packages etc. The user does not need to know that these functions actually create or delete objects from an ArrayList which is written to or read from a CSV file using methods from the Serializable interface.

Inheritance

Inheritance is a concept that acquires the properties from one class to other classes. The class that inherits the properties is known as the sub-class or the child class. The class from which the properties are inherited is known as the superclass or the parent class.

This concept is seen in our project under the Menu class where its subclass are the drinks , dessert and main course where it inherits attributes and methods from another class.

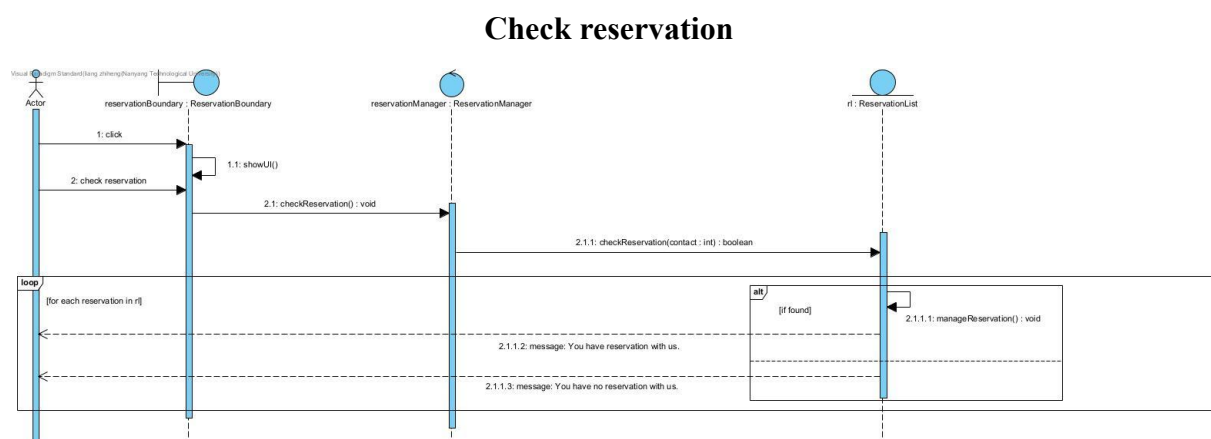
Overloading

Method overloading is a feature of Java in which a class has more than one method of the same name and their parameters are different. For example, for ReservationManager, we have two Method RemoveReservation, one method does not take in any parameters while the others take in an Integer parameter Contact. Furthermore, the OrderManager viewOrder Method is also overloading.

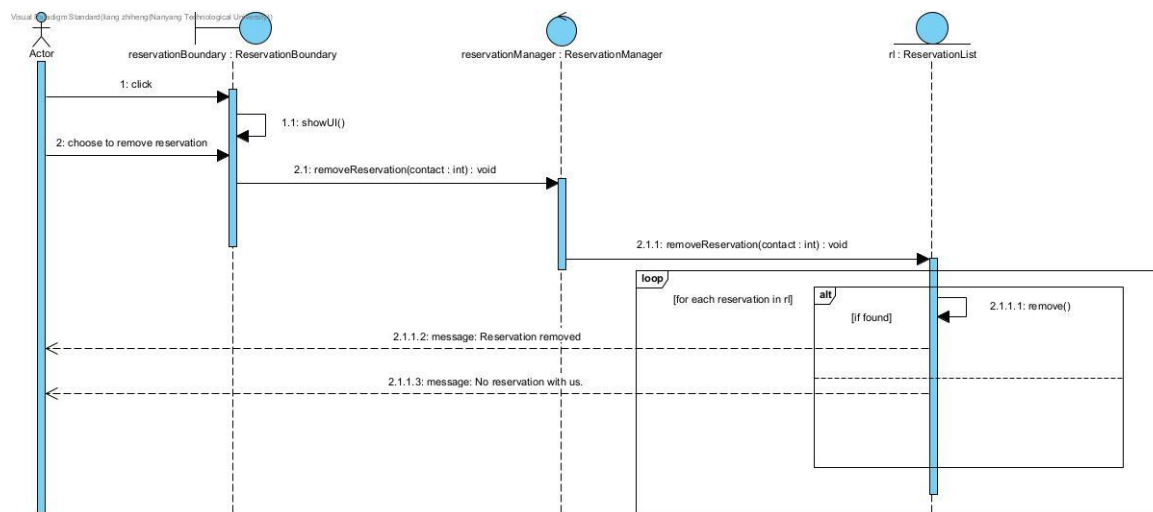
Overriding

Overriding is a feature that allows a subclass or child class to provide a specific implementation of a method that is already provided by one of its superclasses or parent classes. This technique is applied to parent class MenuItem and its subclasses Maincourse, Drink and Dessert.

UML Sequence Diagram (Actual vpp File is outside of report)



Remove reservation



Testing

Reservation Manager

Test case	Expected Outcome	Screen Capture of outcome
User makes a reservation, but the restaurant is fully booked.	<p>We will tell the user that there are “no more empty tables. Please choose another date.”</p> <p>Users can decide to make a reservation on another day.</p>	<pre> 1. View menu items 2. View promotion package 3. Create order 4. View order 5. Add/Remove order item/s to/from order 6. Create reservation booking 7. Check reservation booking 8. Remove reservation booking 9. Check table availability 10. Print order invoice 11. Exit 12. Manager Function Enter your choice: 6 Create reservation booking Enter date & time to reserve (dd-mm-yyyy HH:mm): 04-11-2021 11:00 Enter number of people: 2 Reserved by: Jack Enter contact number: 123 Enter a table number: 123 Table ID Status Capacity 2 reserved 2 3 reserved 2 9 occupied 4 4 occupied 10 No more empty tables. Please choose another date </pre>
Auto remove the expired reservation	<p>Remove reservation 15 minutes after the booked timing.</p> <p>In our screen capture Jack made a reservation for Sat 13 Nov at 11.20am. At 11.35am, if we want to find the reservation made by Jack, it shows that Jack does not have</p>	<pre> RRPQS (1) [Java Application] D:\Work folder\Java\bin\javaw.exe (14 Nov 2021, 8:42:08 pm) Table Reserved by Date Contact number 1 Jack Sat Nov 13 11:20:00 SGT 2021 123 reservation created. 1. View menu items 2. View promotion package 3. Create order 4. View order 5. Add/Remove order item/s to/from order 6. Create reservation booking 7. Check reservation booking 8. Remove reservation booking 9. Check table availability 10. Print order invoice 11. Exit 12. Manager Function Enter your choice: 7 Check reservation booking Enter your contact number for checking: 123 You don't have reservation with us. </pre>

	a reservation made.	
--	---------------------	--

Member Manager

Test case	Expected Outcome	Screen Capture of outcome
Creation of new member	New customer is added. Our test case will add a new member jack with his contact of 789.	<pre> 1.Create Member 2.Remove Member 3.Check Member 4.print membership 5.Exit 1 Enter member name: jack Enter member contact: 789 1.Create Member 2.Remove Member 3.Check Member 4.print membership 5.Exit 4 Name Contact sean 123 abc 456 4 Name Contact sean 123 abc 456 jack 789 </pre>
Deletion of member	Member is deleted. Our test case will delete the member with the contact 789 who is jack.	<pre> 1.Create Member 2.Remove Member 3.Check Member 4.print membership 5.Exit 4 Name Contact sean 123 abc 456 jack 789 1.Create Member 2.Remove Member 3.Check Member 4.print membership 5.Exit 2 Enter your contact number: 789 Member removed 1.Create Member 2.Remove Member 3.Check Member 4.print membership 5.Exit 4 Name Contact sean 123 abc 456 </pre>

Staff Manager

Test case	Expected Outcome	Screen Capture of outcome																
Creation of new staff	<p>New staff is added.</p> <p>A new staff named Cherry, F gender with an employee Id 3 is added.</p>	<pre>1.Create Staff 2.Delete Staff 3.Print Staff 4.Exit 3</pre> <table><tr><th>Staff ID</th><th>Name</th><th>Gender</th><th>Job Title</th></tr><tr><td>1</td><td>Apple</td><td>m</td><td>manager</td></tr></table> <table><tr><th>Staff ID</th><th>Name</th><th>Gender</th><th>Job Title</th></tr><tr><td>2</td><td>Banana</td><td>m</td><td>waiter</td></tr></table>	Staff ID	Name	Gender	Job Title	1	Apple	m	manager	Staff ID	Name	Gender	Job Title	2	Banana	m	waiter
Staff ID	Name	Gender	Job Title															
1	Apple	m	manager															
Staff ID	Name	Gender	Job Title															
2	Banana	m	waiter															

		<pre> 1. Create/Remove/Print Staff 2. Create/Update/Remove MenuItem 3. Create/Update/Delete promotion package 4. Print daily sale revenue report 5. Handle member 6. Handle Table 7. Exit Enter your choice: 1 1.Create Staff 2.Delete Staff 3.Print Staff 4.Exit 1 Name : Cherry Gender : f Job title : slave Emp Id : 3 Staff Cherry Added Successfully 1.Create Staff 2.Delete Staff 3.Print Staff 4.Exit 3 Staff ID Name Gender Job Title ----- 1 Apple m manager 2 Banana m waiter 3 Cherry f slave </pre>
Deletion of staff	<p>Staff is deleted.</p> <p>Employee ID 3 (cherry) is removed from staff list.</p>	<pre> 1.Create Staff 2.Delete Staff 3.Print Staff 4.Exit 3 Staff ID Name Gender Job Title ----- 1 Apple m manager 2 Banana m waiter 3 Cherry f slave 1.Create Staff 2.Delete Staff 3.Print Staff 4.Exit 2 Enter Emp ID: 3 Deleting Emp ID 3 Emp ID 3 Removed Successfully Enter your choice: 1 1.Create Staff 2.Delete Staff 3.Print Staff 4.Exit 3 Staff ID Name Gender Job Title ----- 1 Apple m manager 2 Banana m waiter </pre>

Table Manager

Test case	Expected Outcome	Screen Capture of outcome
Create new tables	<p>New table created.</p> <p>We created a new table (table id 5) with a capacity of 6 .</p>	<pre> 1: create table 2: display table 3: change table status 0:Exit 2 Table ID Status Capacity 1 empty 2 2 reserved 2 3 occupied 4 4 occupied 10 </pre>

		<pre> 1: create table 2: display table 3: change table status 0:Exit 1 Enter the table ID: 5 Enter the capacity: 6 table created. 1: create table 2: display table 3: change table status 0:Exit 2 Table ID Status Capacity 1 empty 2 2 reserved 2 3 occupied 4 4 occupied 10 5 empty 6 </pre>
Change of table status	<p>Table status can be changed to either reserved/empty/occupied.</p> <p>For our test case, we changed the table status of table id 5 from empty to reserved.</p>	<pre> Table ID Status Capacity 1 empty 2 2 reserved 2 3 occupied 4 4 occupied 10 5 empty 6 1: create table 2: display table 3: change table status 0:Exit 3 Enter the table number: 5 1: change table to empty 2: change table to occupied 3: change table to reserved 3 1: create table 2: display table 3: change table status 0:Exit 2 Table ID Status Capacity 1 empty 2 2 reserved 2 3 occupied 4 4 occupied 10 5 reserved 6 1: create table 2: display table 3: change table status 0:Exit </pre>

Video Demonstration

Link : <https://youtu.be/ha06fcq63a0>