NANYANG TECHNOLOGICAL UNIVERSITY

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

CE2002 OBJECT-ORIENTED DESIGN AND PROGRAMMING

AY21/22 Semester 1 Group Assignment

Restaurant Reservation and Point of Sales System (RRPSS)

Lab Group: SE4

Group Number: 5

Name of Group Members	Matriculation Number
Loh Zhi Heng	U2022581B
Lee Cheow Teng	U2020098A
Oh Zhi Hua	U2021091A
Hoang Minh Nhat	U2020852F

Date of Submission: 14th November 2021

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/CZ2002)**	**Lab Group**	**Signature/Date**
Loh Zhi Heng	CE2002	SE4	[illegible] 12/11/2021
Lee Cheow Teng	CE2002	SE4	[illegible] 12/11/2021
Oh Zhi Hua	CE2002	SE4	[illegible] 12/11/2021
Hoang Minh Nhat	CE2002	SE4	Nhat 12/11/2021

Table of Contents

Demonstration Link

https://youtu.be/_Kk1W9Kfxuc

1. Design Considerations

1.1 Approach Taken

For the project, we opted to use the **Model-View-Controller (MVC) Framework**, which is often used for web-based applications. We can segregate UI logic, business logic, and entity level logic using the framework. Additionally, the ability to develop in parallel, combined with the framework's support for **Test Driven Development**, allows us to test and build separate components simultaneously.

1.2 Principles Used

Inheritance

A promotion package is an **item** on the restaurant's menu, hence the Promotion class extends the Item class.
Since both **Staff** and **Customers** are derived from the **abstraction of People**, they both extend the People class.

Interface/Abstract Class

Item class implements **IItem**, which is an **interface** class that provides the prototype methods that Item and Promotion implements.
Staff and **Customer** classes extend **People** class, which is an **abstract** class that provides base methods for both classes to implement and refine.
PromotionController, **CategoryController**, and **TableController**, all implement **ISearch**, which is an **interface** class that provides the prototype method that all three controllers implement.

Method Overriding

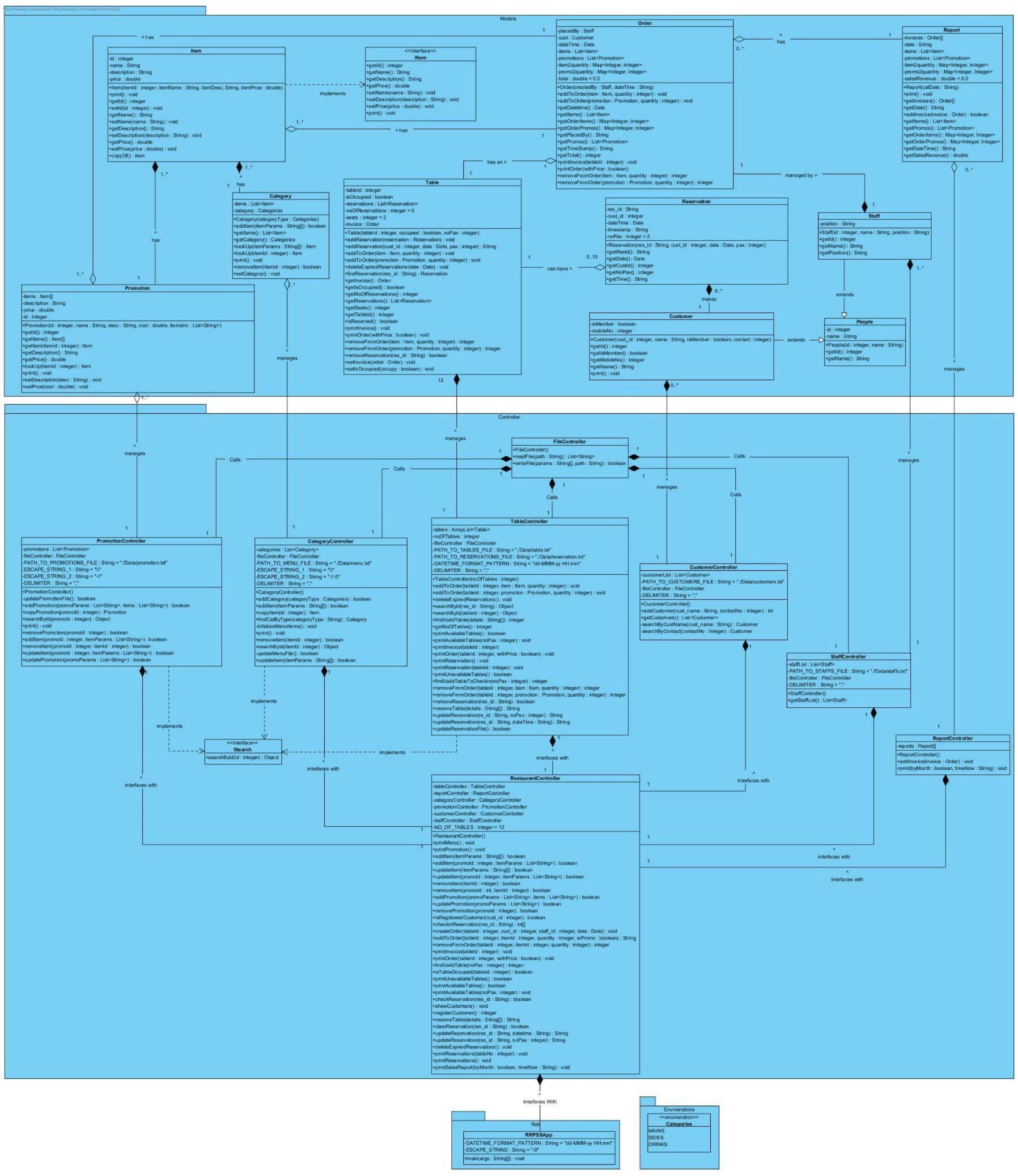
Since Promotion extends Item, it inherits methods of Item, which it can either **refine** or **reuse**.
The three controllers listed above implement **ISearch**, therefore they provide implementations for the method in ISearch.

1.3 Assumptions Made

- There are 3 types of categories: Main, Sides, and Drinks.
    - New categories may be added, however not on the fly.
- There are 4 staff members in the restaurant with ID 0, 1, 2, and 3.
    - New staff members may be added via the data files.
- All of the tables are initially unoccupied when the restaurant (the app) is opened.
    - This is representative of the start of daily restaurant operations.
- Customer's ID required when checking into the restaurant, else must register a new customer ID. When a new customer is registered, his/her membership is initially false.
- Our policy of allocating tables based on the number of pax:
    - For 1 or 2 pax: Only allocates tables of 2 or 4 seats.
    - For 3 or 4 pax: Only allocates tables of 4 or 6 seats.
    - For 5 or 6 pax: Only allocates tables of 6 or 8 seats.
    - For 7 or 8 pax: Only allocates tables of 8 or 10 seats.
    - For 9 or 10 pax: Only allocates tables of 10 seats.
- When a customer checks in, a table cannot be allocated if it is currently occupied or is reserved in the next 15 minutes.
- A reservation can only be made at least 2 hours in advance; a table can only be allocated if it is not reserved within 2 hours before and after the time of the reservation.
    - Please see below for the reason why we set the buffer to 2 hours.
- A customer will take at most 2 hours to finish the meal.
- Current orders placed prior to the update will not be updated if an item or promotion package is modified in the midst of business hours. This is to avoid customers getting an unpleasant awakening about the item they ordered due to a price difference between time of ordering and the time of payment.
    - A copy of the item or promotion object will be made and added to the table's order attribute.
- A promotion package can have 1…* items. This is to facilitate groups of different sizes.
    - E.g: Set for 2, has 2 mains. Set for 4, has 4 mains.
- Order is being added to reports based on **date/time of creation** rather than **date/time of checkout**

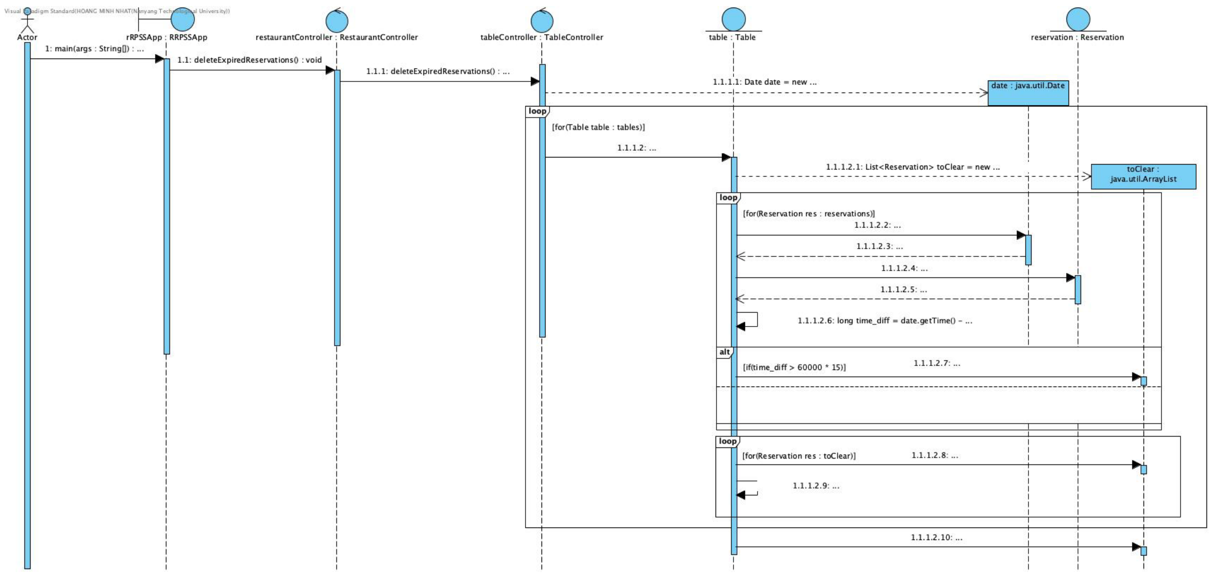
- Report generated is based on the current **day / month** request for generation being sent.
    - E.g. Request at 13th Nov
        - **Daily** will generate report for **13th Nov**
        - **Monthly** will generate report for **month of Nov**

2. UML Class Diagram



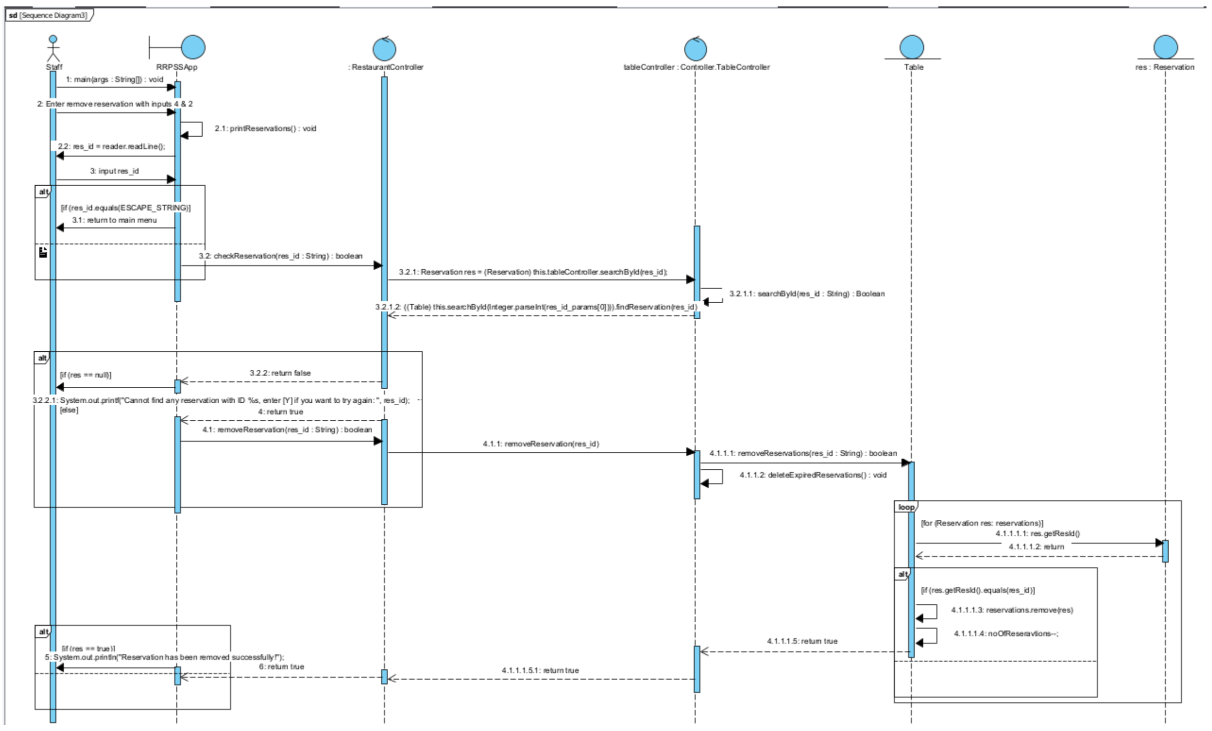
3. UML Sequence Diagram

3.1 Deleting Expired Reservations



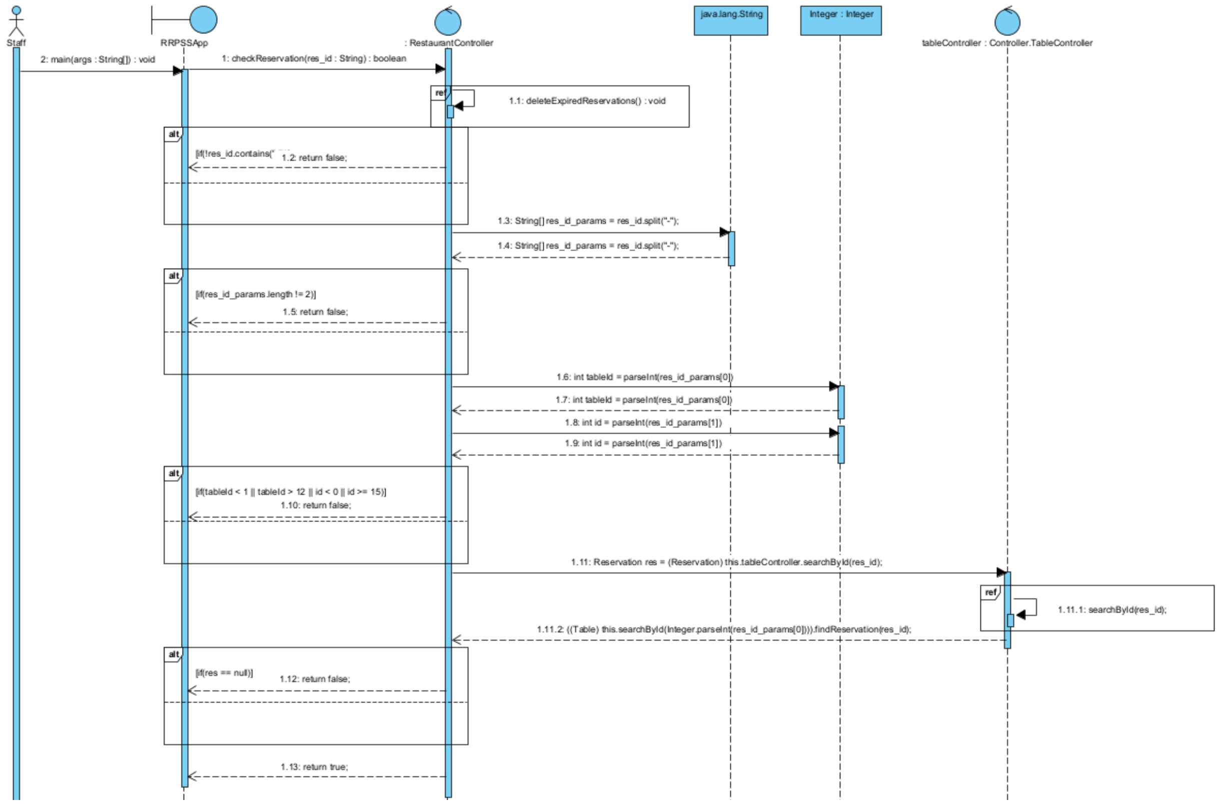
Visual Paradigm Standard(HOANG MINH NHAT(Nanyang Technological University))
Actor
rRPSSApp : RRPSSApp
restaurantController : RestaurantController
tableController : TableController
table : Table
reservation : Reservation
1: main(args : String[]) : ...
1.1: deleteExpiredReservations() : void
1.1.1: deleteExpiredReservations() : ...
1.1.1.1: Date date = new ...
date : java.util.Date
loop
[for(Table table : tables)]
1.1.1.2: ...
1.1.1.2.1: List<Reservation> toClear = new ...
toClear : java.util.ArrayList
loop
[for(Reservation res : reservations)]
1.1.1.2.2: ...
1.1.1.2.3: ...
1.1.1.2.4: ...
1.1.1.2.5: ...
1.1.1.2.6: long time_diff = date.getTime() - ...
alt
[if(time_diff > 60000 * 15)]
1.1.1.2.7: ...
loop
[for(Reservation res : toClear)]
1.1.1.2.8: ...
1.1.1.2.9: ...
1.1.1.2.10: ...

3.2 Remove/Cancel An Existing Reservation

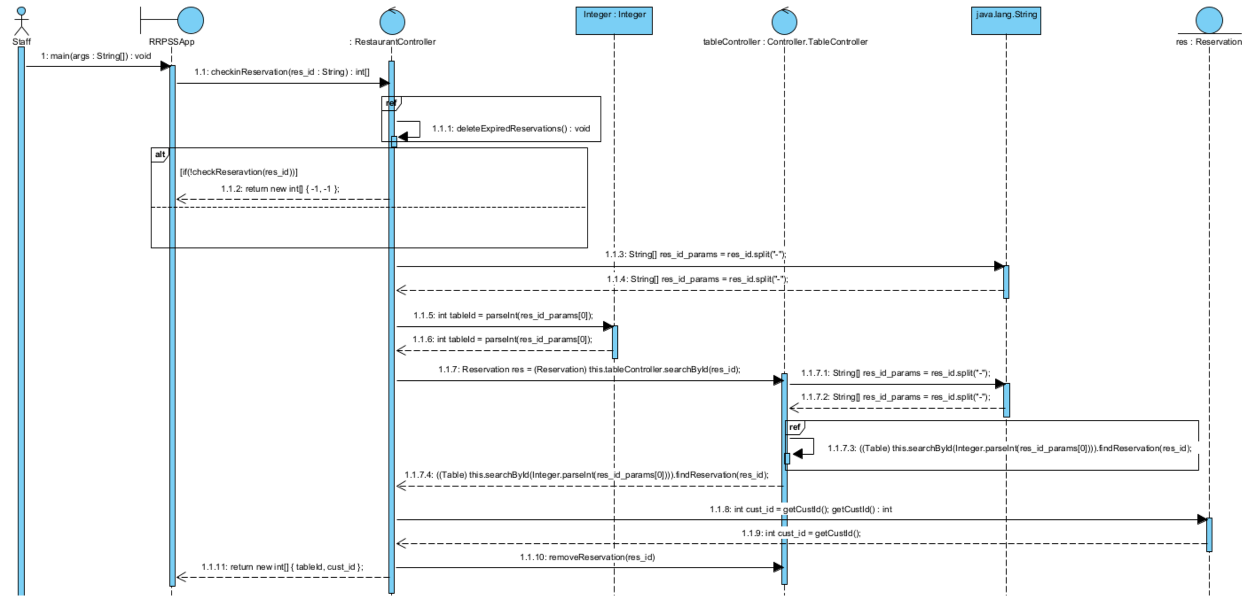


sd [Sequence Diagram3]
Staff
RRPSSApp
: RestaurantController
tableController : Controller.TableController
Table
res : Reservation
1: main(args : String[]) : void
2: Enter remove reservation with inputs 4 & 2
2.1: printReservations() : void
2.2: res_id = reader.readLine();
3: input res_id
alt
[if (res_id.equals(ESCAPE_STRING)]
3.1: return to main menu
3.2: checkReservation(res_id : String) : boolean
3.2.1: Reservation res = (Reservation) this.tableController.searchById(res_id);
3.2.1.1: searchById(res_id : String) : Boolean
3.2.1.2: ((Table) this.searchById(Integer.parseInt(res_id_params[0]))).findReservation(res_id)
alt
[if (res == null)]
3.2.2: return false
3.2.2.1: System.out.printf("Cannot find any reservation with ID %s, enter [Y] if you want to try again: ", res_id);
[else]
4: return true
4.1: removeReservation(res_id : String) : boolean
4.1.1: removeReservation(res_id)
4.1.1.1: removeReservations(res_id : String) : boolean
4.1.1.2: deleteExpiredReservations() : void
loop
[for (Reservation res: reservations)]
4.1.1.1.1: res.getResId()
4.1.1.1.2: return
alt
[if (res.getResId().equals(res_id)]
4.1.1.1.3: reservations.remove(res)
4.1.1.1.4: noOfReseravtions--;
4.1.1.1.5: return true
4.1.1.1.5.1: return true
alt
[if (res == true)]
5: System.out.println("Reservation has been removed successfully!");
6: return true

3.3 Check Reservation


Staff
RRPSSApp
: RestaurantController
java.lang.String
Integer : Integer
tableController : Controller.TableController
2: main(args : String[]) : void
1: checkReservation(res_id : String) : boolean
ref
1.1: deleteExpiredReservations() : void
alt
[if(!res_id.contains("
1.2: return false;
1.3: String[] res_id_params = res_id.split("-");
1.4: String[] res_id_params = res_id.split("-");
alt
[if(res_id_params.length != 2)]
1.5: return false;
1.6: int tableId = parseInt(res_id_params[0])
1.7: int tableId = parseInt(res_id_params[0])
1.8: int id = parseInt(res_id_params[1])
1.9: int id = parseInt(res_id_params[1])
alt
[if(tableId < 1 || tableId > 12 || id < 0 || id >= 15)]
1.10: return false;
1.11: Reservation res = (Reservation) this.tableController.searchById(res_id);
ref
1.11.1: searchById(res_id);
1.11.2: ((Table) this.searchById(Integer.parseInt(res_id_params[0]))).findReservation(res_id);
alt
[if(res == null)]
1.12: return false;
1.13: return true;

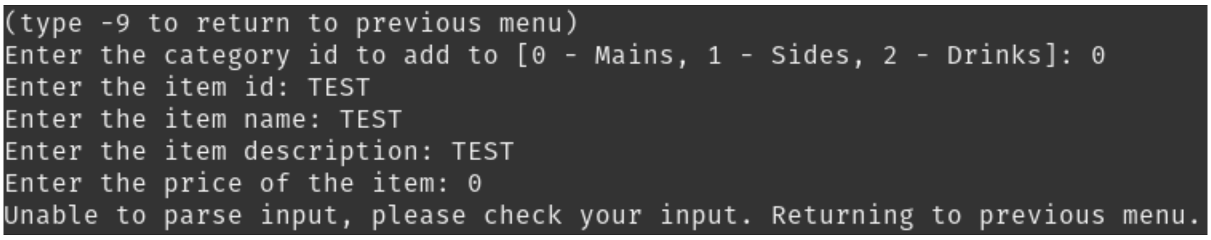
3.4 Check-in with Reservation


Staff
RRPSSApp
: RestaurantController
Integer : Integer
tableController : Controller.TableController
java.lang.String
res : Reservation
1: main(args : String[]) : void
1.1: checkinReservation(res_id : String) : int[]
ref
1.1.1: deleteExpiredReservations() : void
alt
[if(!checkReseravtion(res_id))]
1.1.2: return new int[] { -1, -1 };
1.1.3: String[] res_id_params = res_id.split("-");
1.1.4: String[] res_id_params = res_id.split("-");
1.1.5: int tableId = parseInt(res_id_params[0]);
1.1.6: int tableId = parseInt(res_id_params[0]);
1.1.7: Reservation res = (Reservation) this.tableController.searchById(res_id);
1.1.7.1: String[] res_id_params = res_id.split("-");
1.1.7.2: String[] res_id_params = res_id.split("-");
ref
1.1.7.3: ((Table) this.searchById(Integer.parseInt(res_id_params[0]))).findReservation(res_id);
1.1.7.4: ((Table) this.searchById(Integer.parseInt(res_id_params[0]))).findReservation(res_id);
1.1.8: int cust_id = getCustId(); getCustId() : int
1.1.9: int cust_id = getCustId();
1.1.10: removeReservation(res_id)
1.1.11: return new int[] { tableId, cust_id };

4. Extra Test Cases

4.1 General

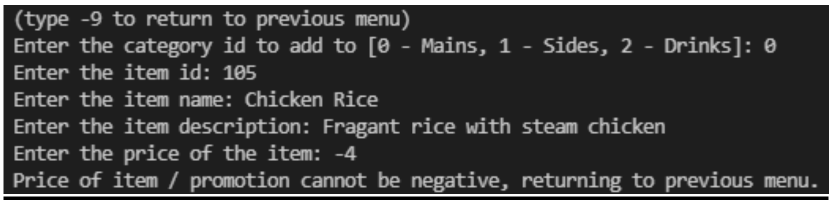
Test Scenario 4.1.1: Input format mismatch



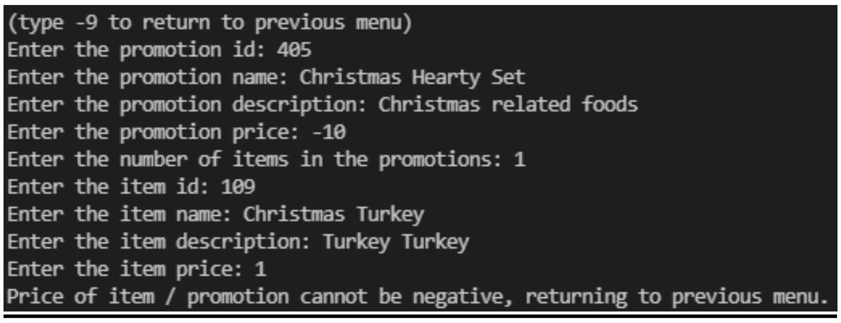
```
(type -9 to return to previous menu)
Enter the category id to add to [0 - Mains, 1 - Sides, 2 - Drinks]: 0
Enter the item id: TEST
Enter the item name: TEST
Enter the item description: TEST
Enter the price of the item: 0
Unable to parse input, please check your input. Returning to previous menu.
```

4.2 Menu / Promotion Queries

Test Scenario 4.2.1: Add Item to menu / Promotion with negative price

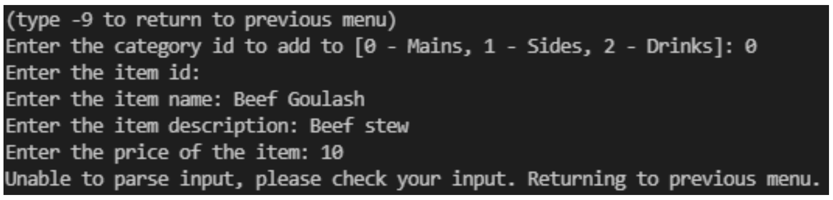


```
(type -9 to return to previous menu)
Enter the category id to add to [0 - Mains, 1 - Sides, 2 - Drinks]: 0
Enter the item id: 105
Enter the item name: Chicken Rice
Enter the item description: Fragant rice with steam chicken
Enter the price of the item: -4
Price of item / promotion cannot be negative, returning to previous menu.
```

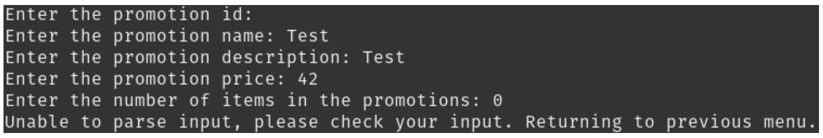


```
(type -9 to return to previous menu)
Enter the promotion id: 405
Enter the promotion name: Christmas Hearty Set
Enter the promotion description: Christmas related foods
Enter the promotion price: -10
Enter the number of items in the promotions: 1
Enter the item id: 109
Enter the item name: Christmas Turkey
Enter the item description: Turkey Turkey
Enter the item price: 1
Price of item / promotion cannot be negative, returning to previous menu.
```

Test Scenario 4.2.2: Add Item to menu / Promotion without ID

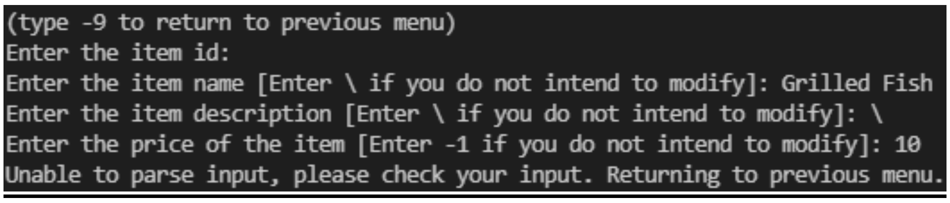


```
(type -9 to return to previous menu)
Enter the category id to add to [0 - Mains, 1 - Sides, 2 - Drinks]: 0
Enter the item id:
Enter the item name: Beef Goulash
Enter the item description: Beef stew
Enter the price of the item: 10
Unable to parse input, please check your input. Returning to previous menu.
```

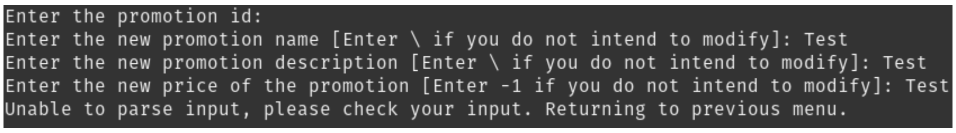


```
Enter the promotion id:
Enter the promotion name: Test
Enter the promotion description: Test
Enter the promotion price: 42
Enter the number of items in the promotions: 0
Unable to parse input, please check your input. Returning to previous menu.
```

Test Scenario 4.2.3: Update Item in menu / Promotion without ID

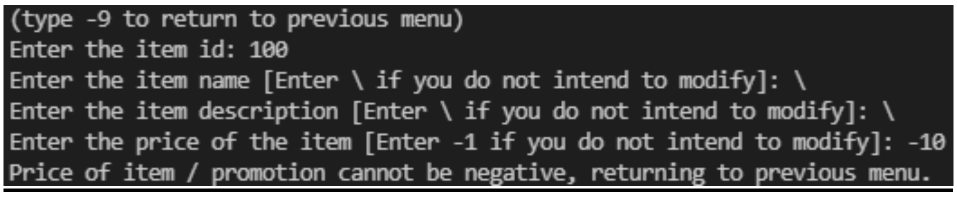


```
(type -9 to return to previous menu)
Enter the item id:
Enter the item name [Enter \ if you do not intend to modify]: Grilled Fish
Enter the item description [Enter \ if you do not intend to modify]: \
Enter the price of the item [Enter -1 if you do not intend to modify]: 10
Unable to parse input, please check your input. Returning to previous menu.
```

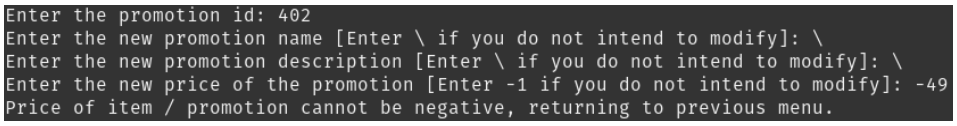


```
Enter the promotion id:
Enter the new promotion name [Enter \ if you do not intend to modify]: Test
Enter the new promotion description [Enter \ if you do not intend to modify]: Test
Enter the new price of the promotion [Enter -1 if you do not intend to modify]: Test
Unable to parse input, please check your input. Returning to previous menu.
```

<u>Test Scenario 4.2.4:</u> Update Item in menu / Promotion in menu with negative price

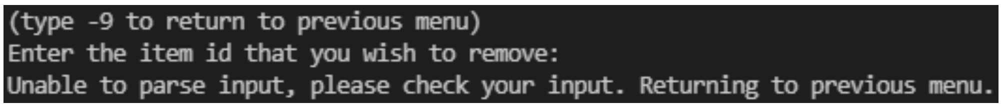


```
(type -9 to return to previous menu)
Enter the item id: 100
Enter the item name [Enter \ if you do not intend to modify]: \
Enter the item description [Enter \ if you do not intend to modify]: \
Enter the price of the item [Enter -1 if you do not intend to modify]: -10
Price of item / promotion cannot be negative, returning to previous menu.
```

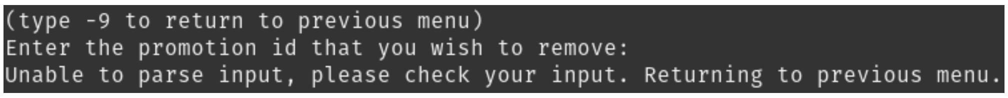


```
Enter the promotion id: 402
Enter the new promotion name [Enter \ if you do not intend to modify]: \
Enter the new promotion description [Enter \ if you do not intend to modify]: \
Enter the new price of the promotion [Enter -1 if you do not intend to modify]: -49
Price of item / promotion cannot be negative, returning to previous menu.
```

<u>Test Scenario 4.2.5:</u> Remove Item from menu / Promotion without ID



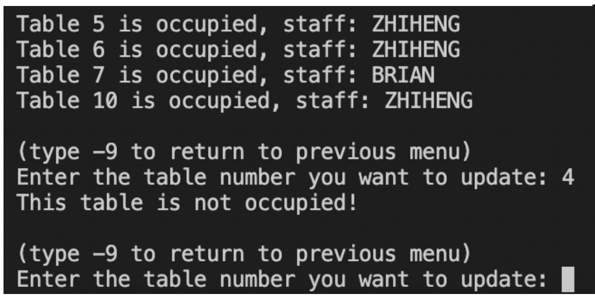
```
(type -9 to return to previous menu)
Enter the item id that you wish to remove:
Unable to parse input, please check your input. Returning to previous menu.
```



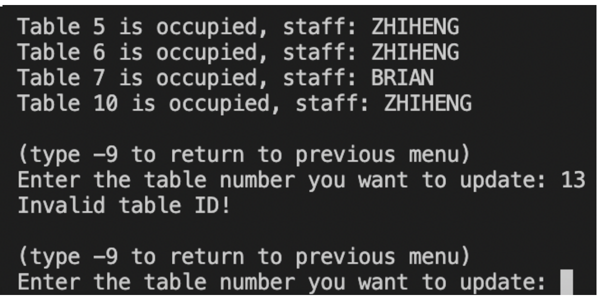
```
(type -9 to return to previous menu)
Enter the promotion id that you wish to remove:
Unable to parse input, please check your input. Returning to previous menu.
```

4.3 <u>Order Queries</u>

<u>Test Scenario 4.3.1:</u> Update a current order of an unoccupied/invalid table.

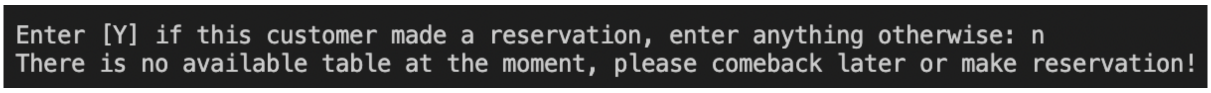


```
Table 5 is occupied, staff: ZHIHENG
Table 6 is occupied, staff: ZHIHENG
Table 7 is occupied, staff: BRIAN
Table 10 is occupied, staff: ZHIHENG

(type -9 to return to previous menu)
Enter the table number you want to update: 4
This table is not occupied!

(type -9 to return to previous menu)
Enter the table number you want to update: 
```



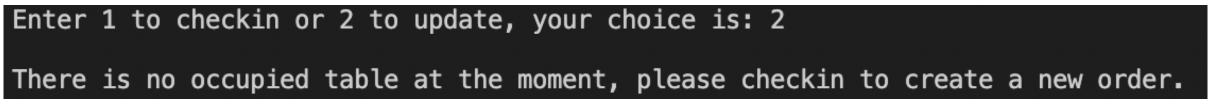
```
Table 5 is occupied, staff: ZHIHENG
Table 6 is occupied, staff: ZHIHENG
Table 7 is occupied, staff: BRIAN
Table 10 is occupied, staff: ZHIHENG

(type -9 to return to previous menu)
Enter the table number you want to update: 13
Invalid table ID!

(type -9 to return to previous menu)
Enter the table number you want to update: 
```

<u>Test Scenario 4.3.2:</u> Walk-in dining but all of the tables are either reserved or occupied.



```
Enter [Y] if this customer made a reservation, enter anything otherwise: n
There is no available table at the moment, please comeback later or make reservation!
```

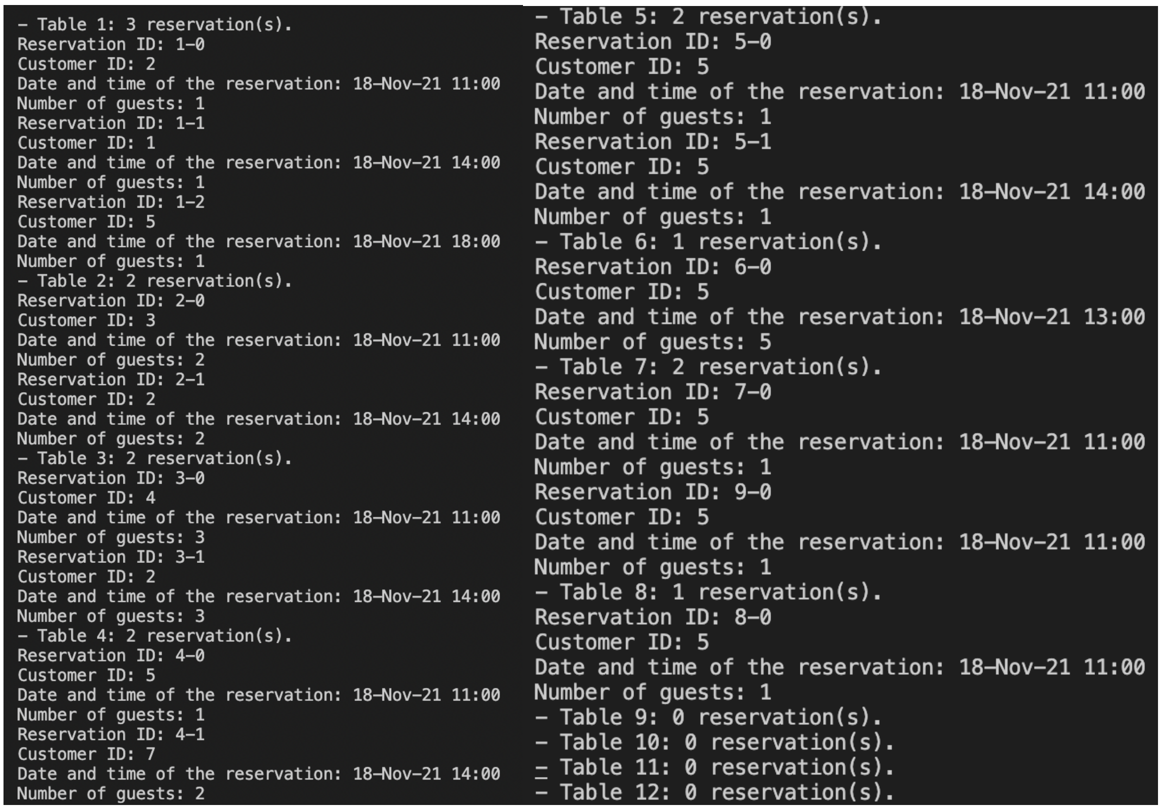
<u>Test Scenario 4.3.2:</u> Try to update a current order but all of the tables are unoccupied.



```
Enter 1 to checkin or 2 to update, your choice is: 2

There is no occupied table at the moment, please checkin to create a new order.
```

4.4 Reservation Queries

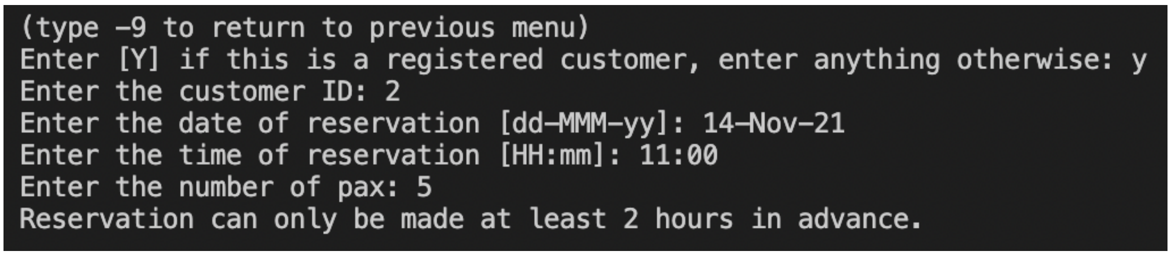
List of reservations for all of the tests below:



```
- Table 1: 3 reservation(s).
Reservation ID: 1-0
Customer ID: 2
Date and time of the reservation: 18-Nov-21 11:00
Number of guests: 1
Reservation ID: 1-1
Customer ID: 1
Date and time of the reservation: 18-Nov-21 14:00
Number of guests: 1
Reservation ID: 1-2
Customer ID: 5
Date and time of the reservation: 18-Nov-21 18:00
Number of guests: 1
- Table 2: 2 reservation(s).
Reservation ID: 2-0
Customer ID: 3
Date and time of the reservation: 18-Nov-21 11:00
Number of guests: 2
Reservation ID: 2-1
Customer ID: 2
Date and time of the reservation: 18-Nov-21 14:00
Number of guests: 2
- Table 3: 2 reservation(s).
Reservation ID: 3-0
Customer ID: 4
Date and time of the reservation: 18-Nov-21 11:00
Number of guests: 3
Reservation ID: 3-1
Customer ID: 2
Date and time of the reservation: 18-Nov-21 14:00
Number of guests: 3
- Table 4: 2 reservation(s).
Reservation ID: 4-0
Customer ID: 5
Date and time of the reservation: 18-Nov-21 11:00
Number of guests: 1
Reservation ID: 4-1
Customer ID: 7
Date and time of the reservation: 18-Nov-21 14:00
Number of guests: 2
```

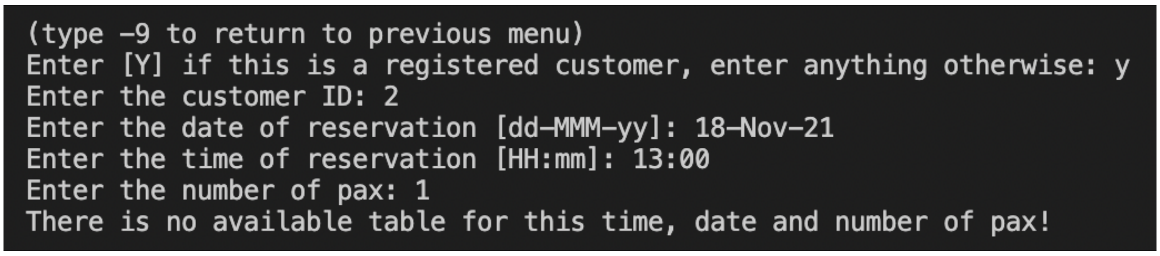
```
- Table 5: 2 reservation(s).
Reservation ID: 5-0
Customer ID: 5
Date and time of the reservation: 18-Nov-21 11:00
Number of guests: 1
Reservation ID: 5-1
Customer ID: 5
Date and time of the reservation: 18-Nov-21 14:00
Number of guests: 1
- Table 6: 1 reservation(s).
Reservation ID: 6-0
Customer ID: 5
Date and time of the reservation: 18-Nov-21 13:00
Number of guests: 5
- Table 7: 2 reservation(s).
Reservation ID: 7-0
Customer ID: 5
Date and time of the reservation: 18-Nov-21 11:00
Number of guests: 1
Reservation ID: 9-0
Customer ID: 5
Date and time of the reservation: 18-Nov-21 11:00
Number of guests: 1
- Table 8: 1 reservation(s).
Reservation ID: 8-0
Customer ID: 5
Date and time of the reservation: 18-Nov-21 11:00
Number of guests: 1
- Table 9: 0 reservation(s).
- Table 10: 0 reservation(s).
- Table 11: 0 reservation(s).
- Table 12: 0 reservation(s).
```

<u>Test Scenario 4.4.1:</u> Input a time that is not at least 2 hours upon the current time, regardless of the number of pax.



```
(type -9 to return to previous menu)
Enter [Y] if this is a registered customer, enter anything otherwise: y
Enter the customer ID: 2
Enter the date of reservation [dd-MMM-yy]: 14-Nov-21
Enter the time of reservation [HH:mm]: 11:00
Enter the number of pax: 5
Reservation can only be made at least 2 hours in advance.
```

<u>Test Scenario 4.4.2:</u> Input a time that is at least 2 hours, regardless of the number of pax, but there is no available table within 2 hours before and after the desired time.



```
(type -9 to return to previous menu)
Enter [Y] if this is a registered customer, enter anything otherwise: y
Enter the customer ID: 2
Enter the date of reservation [dd-MMM-yy]: 18-Nov-21
Enter the time of reservation [HH:mm]: 13:00
Enter the number of pax: 1
There is no available table for this time, date and number of pax!
```

<u>Test Scenario 4.4.3:</u> Update a reservation with a time that is not at least 2 hours upon the current time.

Input: reservation ID (8-0), new time date (14-Nov-21 11:00).

Result: the same as the Test Scenario 4.4.1 and the reservation will remain unchanged.

<u>Test Scenario 4.4.4:</u> Update a reservation time that is at least 2 hours upon the current time, but there is no available table within 2 hours before or after the desired time.

Input: reservation ID (8-0), new time date (18-Nov-21 13:00).

Result: the same as the Test Scenario 4.4.2 and the reservation will remain unchanged.

<u>Test Scenario 4.4.5:</u> Update a reservation with a new number of pax, which may allocate another table and may result in a clash.

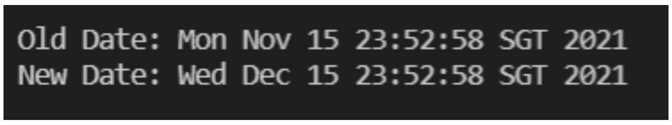
Input: reservation ID: 8-0, new number of pax: 2

Result: the same as the Test Scenario 4.4.2 and the reservation will remain unchanged.

4.5 Print Sales Report Queries

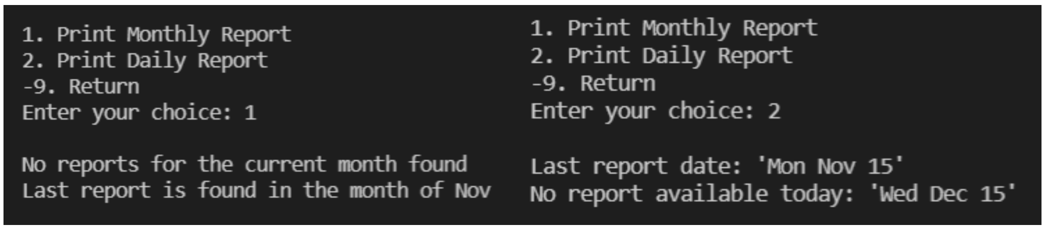
<u>Test Scenario:</u> Report generation for orders with differing creation and checkout date.

1. Create order (Any generic order works).
2. Shift time by 1 month & checkout order.



```
Old Date: Mon Nov 15 23:52:58 SGT 2021
New Date: Wed Dec 15 23:52:58 SGT 2021
```

3. Print daily or monthly report.



```
1. Print Monthly Report
2. Print Daily Report
-9. Return
Enter your choice: 1

No reports for the current month found
Last report is found in the month of Nov
```

```
1. Print Monthly Report
2. Print Daily Report
-9. Return
Enter your choice: 2

Last report date: 'Mon Nov 15'
No report available today: 'Wed Dec 15'
```