Declaration of Original Work for CE/CZ2002 Assignment

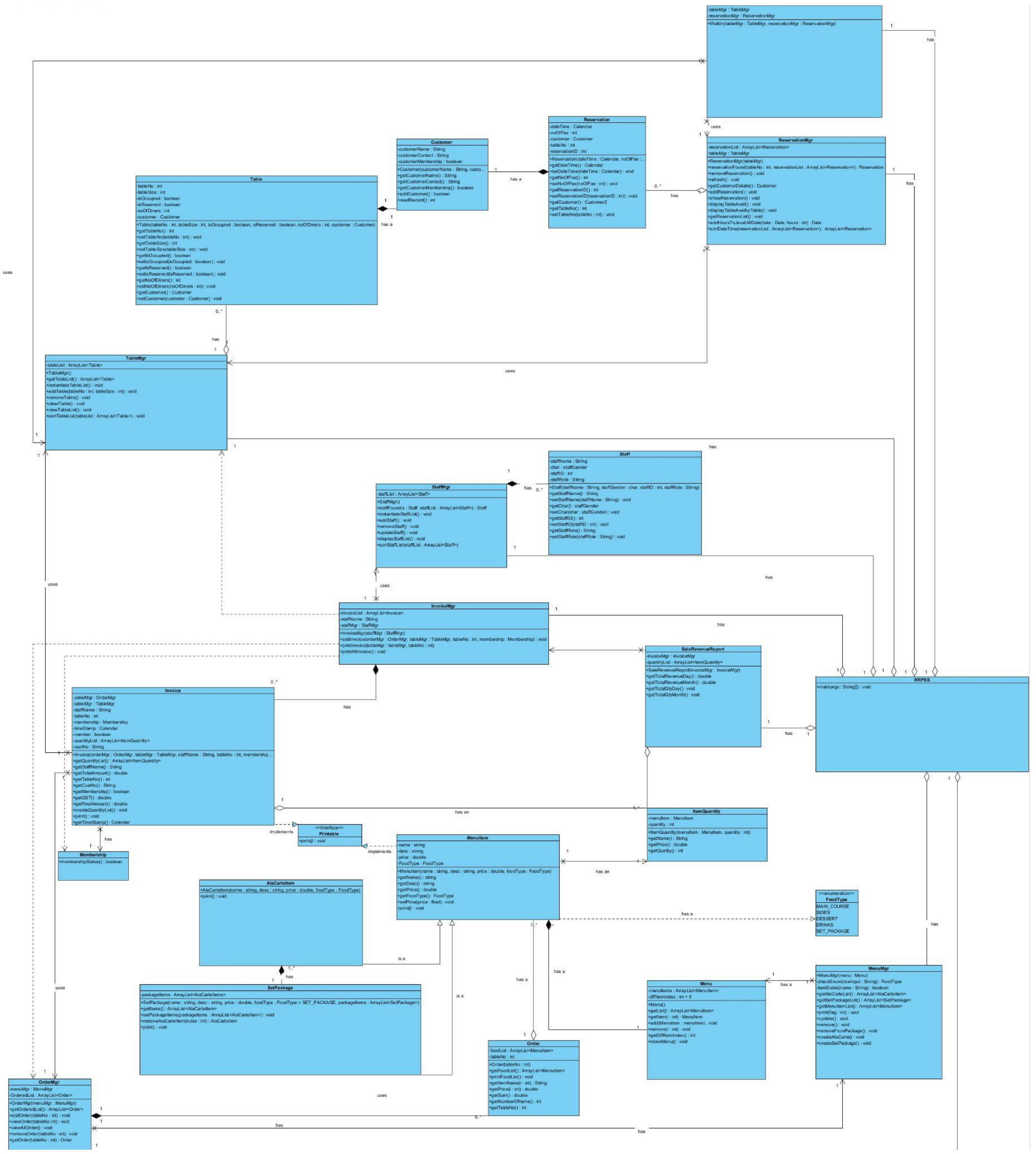
We hereby declare that the attached group assignment has been researched, undertaken, completed and submitted as a collective effort by the group members listed below.

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 (CZ2002)	Lab Group	Signature /Date
Ang Jun Yi, Keith	CZ2002	MACS	[illegible] 14/11/21
Celeste Sew Hui Ting (QIU HUI TING)	CZ2002	MACS	[illegible] 14/11/21
Say Yueyang, Symus	CZ2002	MACS	[illegible] 14/11/21
Su Xinhui	CZ2002	MACS	[illegible] 14/11/21
Wong Li Wen	CZ2002	MACS	[illegible] 14/11/21

UML Class Diagram


-tableMgr : TableMgr
-reservationMgr : ReservationMgr
+WalkIn(tableMgr : TableMgr, reservationMgr : ReservationMgr)
Reservation
-dateTime : Calendar
-noOfPax : int
-customer : Customer
-tableNo : int
-reservationID : int
+Reservation(dateTime : Calendar, noOfPax :...
+getDateTime() : Calendar
+setDateTime(dateTime : Calendar) : void
+getNoOfPax() : int
+setNoOfPax(noOfPax : int) : void
+getReservationID() : int
+setReservationID(reservationID : int) : void
+getCustomer() : Customer2
+getTableNo() : int
+setTableNo(tableNo : int) : void
ReservationMgr
-reservationList : ArrayList<Reservation>
-tableMgr : TableMgr
+ReservationMgr(tableMgr)
+reservationFound(tableNo : int, reservationList : ArrayList<Reservation>) : Reservation
+removeReservation() : void
+refresh() : void
+getCustomerDetails() : Customer
+addReservation() : void
+showReservation() : void
+displayTableAvail() : void
+displayTableAvailbyTable() : void
+getReservationList() : void
+addHoursToJavaUtilDate(date : Date, hours : int) : Date
+sortDateTime(reservationList : ArrayList<Reservation>) : ArrayList<Reservation>
Customer
-customerName : String
-customerContact : String
-customerMembership : boolean
+Customer(customerName : String, custo...
+getCustomerName() : String
+getCustomerContact() : String
+getCustomerMembership() : boolean
+addCustomer() : boolean
+readRecord() : int
Table
-tableNo : int
-tableSize : int
-isOccupied : boolean
-isReserved : boolean
-noOfDiners : int
-customer : Customer
+Table(tableNo : int, tableSize : int, isOccupied : boolean, isReserved : boolean, noOfDiners : int, customer : Customer)
+getTableNo() : int
+setTableNo(tableNo : int) : void
+getTableSize() : int
+setTableSize(tableSize : int) : void
+getIsOccupied() : boolean
+setIsOccupied(isOccupied : boolean) : void
+getIsReserved() : boolean
+setIsReserved(isReserved : boolean) : void
+getNoOfDiners() : int
+setNoOfDiners(noOfDiners : int) : void
+getCustomer() : Customer
+setCustomer(customer : Customer) : void
TableMgr
-tableList : ArrayList<Table>
+TableMgr()
+getTableList() : ArrayList<Table>
+instantiateTableList() : void
+addTable(tableNo : int, tableSize : int) : void
+removeTable() : void
+clearTable() : void
+viewTableList() : void
+sortTableList(tableList : ArrayList<Table>) : void
Staff
-staffName : String
-char : staffGender
-staffID : int
-staffRole : String
+Staff(staffName : String, staffGender : char, staffID : int, staffRole : String)
+getStaffName() : String
+setStaffName(staffName : String) : void
+getChar() : staffGender
+setChar(char : staffGender) : void
+getStaffID() : int
+setStaffID(staffID : int) : void
+getStaffRole() : String
+setStaffRole(staffRole : String) : void
StaffMgr
-staffList : ArrayList<Staff>
+StaffMgr()
+staffFound(s : Staff, staffList : ArrayList<Staff>) : Staff
+instantiateStaffList() : void
+addStaff() : void
+removeStaff() : void
+updateStaff() : void
+displayStaffList() : void
+sortStaffList(staffList : ArrayList<Staff>)
InvoiceMgr
-invoiceList : ArrayList<Invoice>
-staffName : String
-staffMgr : StaffMgr
+InvoiceMgr(staffMgr : StaffMgr)
+addInvoice(orderMgr : OrderMgr, tableMgr : TableMgr, tableNo : int, membership : Membership) : void
+printInvoice(tableMgr : tableMgr, tableNo : int)
+printAllInvoice() : void
SaleRevenueReport
-invoiceMgr : InvoiceMgr
-quantityList : ArrayList<ItemQuantity>
+SaleRevenueReport(invoiceMgr : InvoiceMgr)
+getTotalRevenueDay() : double
+getTotalRevenueMonth() : double
+getTotalQtyDay() : void
+getTotalQtyMonth() : void
RRPSS
+main(args : String[]) : void
Invoice
-orderMgr : OrderMgr
-tableMgr : TableMgr
-staffName : String
-tableNo : int
-membership : Membership
-timeStamp : Calendar
-member : boolean
-quantityList : ArrayList<ItemQuantity>
-custNo : String
+Invoice(orderMgr : OrderMgr, tableMgr : TableMgr, staffName : String, tableNo : int, membership ...
+getQuantityList() : ArrayList<ItemQuantity>
+getStaffName() : String
+getTotalAmount() : double
+getTableNo() : int
+getCustNo() : String
+getMembership() : boolean
+getGST() : double
+getFinalAmount() : double
+createQuantityList() : void
+print() : void
+getTimeStamp() : Calendar
Membership
+membershipStatus() : boolean
<<Interface>>
Printable
+print() : void
ItemQuantity
-menuItem : MenuItem
-quantity : int
+ItemQuantity(menuItem : MenuItem, quantity : int)
+getName() : String
+getPrice() : double
+getQuatity() : int
MenuItem
-name : string
-desc : string
-price : double
-FoodType : FoodType
+MenuItem(name : string, desc : string, price : double, foodType : FoodType)
+getName() : string
+getDesc() : string
+getPrice() : double
+getFoodType() : FoodType
+setPrice(price : float) : void
+print() : void
AlaCarteItem
+AlaCarteItem(name : string, desc : string, price : double, foodType : FoodType)
+print() : void
<<enumeration>>
FoodType
MAIN_COURSE
SIDES
DESSERT
DRINKS
SET_PACKAGE
SetPackage
-packageItems : ArrayList<AlaCarteItems>
+SetPackage(name : string, desc : string, price : double, foodType : FoodType = SET_PACKAGE, packageItems : ArrayList<SetPackage>)
+getItems() : ArrayList<AlaCarteItem>
+setPackageItems(packageItems : ArrayList<AlaCarteItem>) : void
+removeAlaCarteItem(index : int) : AlaCarteItem
+print() : void
Menu
-menuItems : ArrayList<MenuItem>
-diffItemIndex : int = 0
+Menu()
+getList() : ArrayList<MenuItem>
+getItem(i : int) : MenuItem
+add(MenuItem : menuItem) : void
+remove(i : int) : void
+getDiffItemIndex() : int
+closeMenu() : void
MenuMgr
+MenuMgr(menu : Menu)
-checkEnum(rawInput : String) : FoodType
-itemExists(name : String) : boolean
+getAlaCarteList() : ArrayList<AlaCarteItem>
+getSetPackageList() : ArrayList<SetPackage>
+getMenuItemList() : ArrayList<MenuItem>
+print(flag : int) : void
+update() : void
+remove() : void
+removeFromPackage() : void
+createAlaCarte() : void
+createSetPackage() : void
Order
-foodList : ArrayList<MenuItem>
-tableNo : int
+Order(tableNo : int)
+getFoodList() : ArrayList<MenuItem>
+printFoodList() : void
+getItemName(i : int) : String
+getPrice(i : int) : double
+getSum() : double
+getNumberOfItems() : int
+getTableNo() : int
OrderMgr
-menuMgr : MenuMgr
-OrderedList : ArrayList<Order>
+OrderMgr(menuMgr : MenuMgr)
+getOrderedList() : ArrayList<Order>
+addOrder(tableNo : int) : void
+viewOrder(tableNo int) : void
+viewAllOrder() : void
+removeOrder(tableNo : int) : void
+getOrder(tableNo : int) : Order
has
has a
has an
uses
is a
implements
1
0..*
1..*

<u>Design Considerations</u>

We designed our RRPSS to have multiple functionalities which are easy to use for the restaurant staff. Manager classes were made for each data class, e.g. MenuMgr for Menu, such that the data class contains all its relevant data whereas its manager contains all its operations. For example, when a staff wants to change the price of an item using the RRPSS, the RRPSS will call the MenuMgr to make changes to the price of the item by accessing the Menu. For smoother flow in our user interface, we used exception handling to make sure that the RRPSS does not terminate when the user enters an invalid input, but instead informs the user of the error and prompts the user to enter again instead. This prevents excessive re-running of the RRPSS which may cause the loss of previously entered data.

<u>Use of OO Concepts</u>

Abstraction:

The usage of abstract class and interface in our design allows us to hide unnecessary details from the users and only display the required information. For example, our MenuItem class is an abstract class that cannot be used to create objects. However, our AlaCarteItem and SetPackage classes do not need to know the inner details of this Menu Item to use it. By doing so, we are able to use the same information with no modification. We also implemented an interface which allows us to abstract the implementation of print completely.

Encapsulation/Information hiding:

In all our classes, our attributes data is hidden from other classes and can only be accessed through public methods of the object's class. This allows us to protect our object's private data from other classes and prevent any unauthorised access to them. It also promotes more flexibility to modify implemented code without affecting other code too much. For example, the customer's personal details can only be accessed through public methods in ReservationMgr and TableMgr, and staff details can only be accessed through public methods in StaffMgr and InvoiceMgr. Furthermore, in the use of private methods as in MenuMgr, the other objects do not need to know how the MenuMgr object checks for valid data, and is hence kept private to prevent modification.

Inheritance:

We have created classes without extensive duplication of code. Since both AlaCarteItem and SetPackage acquire all properties and behaviours of MenuItem, we can inherit them and reuse MenuItem code. This promotes code reusability and flexibility to override methods of base class.

Polymorphism:

Since both AlaCarteItem and SetPackage are of object MenuItem, when the Menu object is created, an ArrayList<MenuItem> is created, and both types of items are added to the array. The print method, which differs slightly for both MenuItem, is dynamically invoked when called. This facilitates the storage of information as (1) a single variable (i.e. the ArrayList) can be used to store both items, optimizing memory usage, and (2) methods can be reused across variables.

Design Principles

Single Responsibility Principle:

We ensured our classes fulfilled one single responsibility. For example, Table class is only responsible for storing the details of a table, and the TableMgr objects are only in charge of storing the ArrayLists of their corresponding objects and making modifications to it. Other examples of our SRP usage can be seen in other classes such as Order, Invoice, Customer, Staff and Reservation. This allows our application to be more maintainable with high cohesion.

Open/Closed Principle:

Abstraction is the key to open-closed principle. Our abstract MenuItem class is open for extension but closed for modification. If we want some modification to MenuItem, we can easily do that by extending MenuItem class without modifying MenuItem class itself. This can be achieved in AlacarteItem and SetPackage class where we extend the MenuItem class and can have methods for overriding.

Liskov Substitution Principle:

This principle states that objects of a superclass can be substituted with objects of its subclasses without disrupting the behaviour of our application. Our overridden method in AlaCarteItem and SetPackage subclasses accept the same input parameter as that of our MenuItem superclass. This means we are able to substitute between the usage of our superclass and subclasses objects without compromising the expected behaviour of our application.

Interface Segregation Principle:

Interface Segregation Principle states that many client specific interfaces are better than one general purpose interface. This means larger interfaces should be split into smaller ones. As we only have one interface in our application, this principle is not applicable to our design.

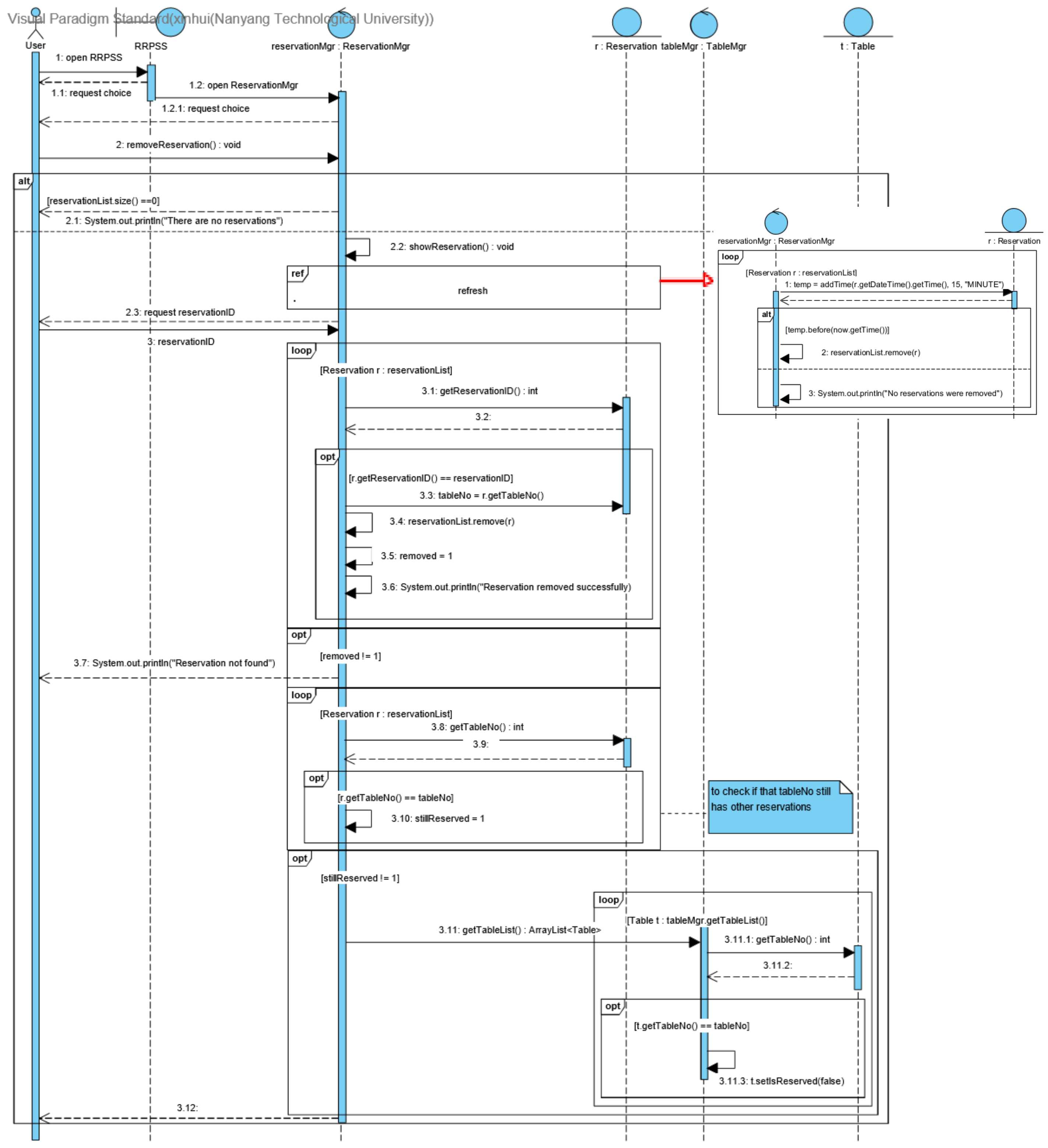
Dependency Inversion Principle:

High level modules should not depend upon low level modules. Both should depend upon abstractions. In our application, MenuItem and Invoice depend on printable interface. This printable interface helps us to abstract the implementation and reduce coupling between classes.

Assumptions

Our restaurant operates from 9am to 9pm and deals with both walk-ins and reservations. Customers can occupy reserved tables that are reserved at a later time, provided the reserved time is more than one hour away from current time. If the reserved table is within one hour of current time and all other tables are occupied, then no more walk-ins can be accepted. We assume that customers are only allowed to make reservations from 9am to 9pm with timing ending with either :00 or :30, and each reservation lasts one hour. Customers who made reservations are only allowed to enter 15 minutes before their reservation and are assumed to wait if they come earlier than their slot. If customers do not show up after 15 minutes of reservation time, their reservation slot will no longer be valid and the reserved table will be made available. Customers who have a membership card to the restaurant are entitled to a 10% discount. We assume that each customer's contact number is unique and used to identify membership status. We assume updates to change only the price of menu items. Invoice needs to be printed before sales revenue can be updated and reflected.

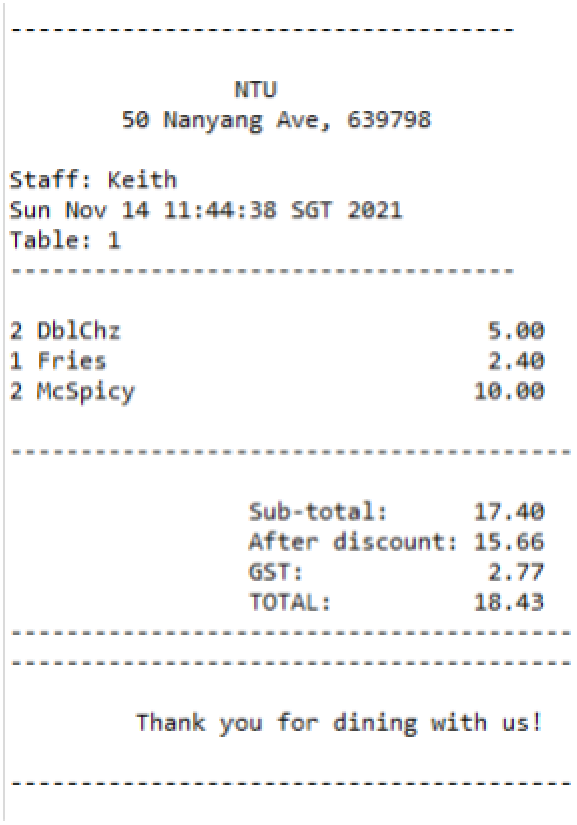
UML Sequence Diagram for Check/Remove Reservation Booking



Visual Paradigm Standard(xinhui(Nanyang Technological University))
User
RRPSS
reservationMgr : ReservationMgr
r : Reservation
tableMgr : TableMgr
t : Table
1: open RRPSS
1.1: request choice
1.2: open ReservationMgr
1.2.1: request choice
2: removeReservation() : void
alt
[reservationList.size() ==0]
2.1: System.out.println("There are no reservations")
2.2: showReservation() : void
ref
refresh
2.3: request reservationID
3: reservationID
loop
[Reservation r : reservationList]
3.1: getReservationID() : int
3.2:
opt
[r.getReservationID() == reservationID]
3.3: tableNo = r.getTableNo()
3.4: reservationList.remove(r)
3.5: removed = 1
3.6: System.out.println("Reservation removed successfully)
opt
[removed != 1]
3.7: System.out.println("Reservation not found")
loop
[Reservation r : reservationList]
3.8: getTableNo() : int
3.9:
opt
[r.getTableNo() == tableNo]
3.10: stillReserved = 1
to check if that tableNo still has other reservations
opt
[stillReserved != 1]
loop
[Table t : tableMgr.getTableList()]
3.11: getTableList() : ArrayList<Table>
3.11.1: getTableNo() : int
3.11.2:
opt
[t.getTableNo() == tableNo]
3.11.3: t.setIsReserved(false)
3.12:
reservationMgr : ReservationMgr
r : Reservation
loop
[Reservation r : reservationList]
1: temp = addTime(r.getDateTime().getTime(), 15, "MINUTE")
alt
[temp.before(now.getTime())]
2: reservationList.remove(r)
3: System.out.println("No reservations were removed")

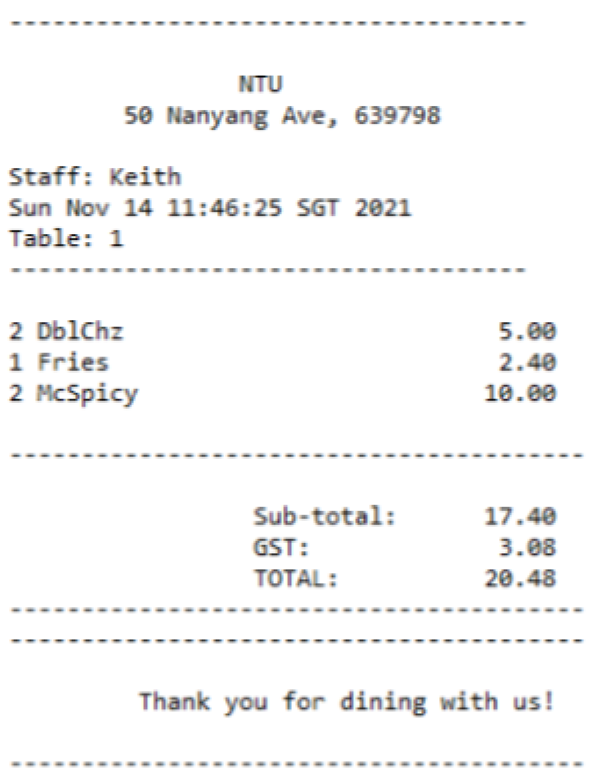
Test Cases: (not covered in demo)

- **Membership**



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

              NTU
      50 Nanyang Ave, 639798

Staff: Keith
Sun Nov 14 11:44:38 SGT 2021
Table: 1
------------------------------------

2 DblChz                         5.00
1 Fries                          2.40
2 McSpicy                       10.00

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

                 Sub-total:     17.40
                 After discount: 15.66
                 GST:            2.77
                 TOTAL:         18.43
----------------------------------------
----------------------------------------

         Thank you for dining with us!

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

Figure 1: Customer with membership card

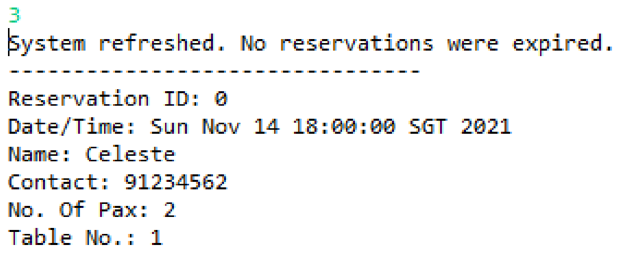


```
------------------------------------

              NTU
      50 Nanyang Ave, 639798

Staff: Keith
Sun Nov 14 11:46:25 SGT 2021
Table: 1
------------------------------------

2 DblChz                         5.00
1 Fries                          2.40
2 McSpicy                       10.00

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

                 Sub-total:     17.40
                 GST:            3.08
                 TOTAL:         20.48
----------------------------------------
----------------------------------------

         Thank you for dining with us!

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

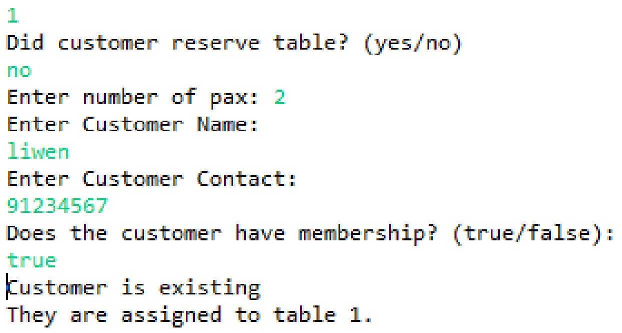
Figure 2: Customer with no membership card

- **Walk-ins and reservation**
    1. Customers with no reservation can still walk in if the reserved tables are more than 1 hour away from current time.
    2. (Assuming all tables currently occupied) Customers can still make reservations for a later time.
    3. Multiple reservations can be made for the same table provided there is no clash in time buffer.
    4. (Assuming all tables are reserved at the same time slot) Other customers cannot make reservations at this time slot.

Allow reserved table to be occupied before reservation timing



```
3
System refreshed. No reservations were expired.
--------------------------------
Reservation ID: 0
Date/Time: Sun Nov 14 18:00:00 SGT 2021
Name: Celeste
Contact: 91234562
No. Of Pax: 2
Table No.: 1
```

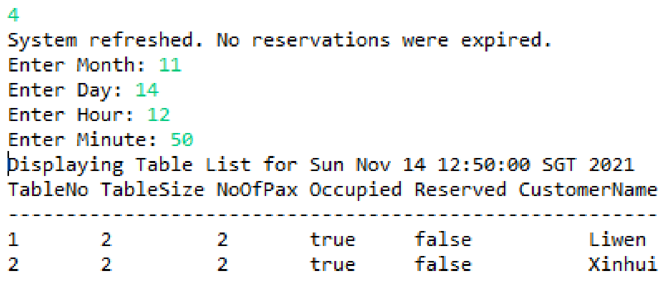


```
1
Did customer reserve table? (yes/no)
no
Enter number of pax: 2
Enter Customer Name:
liwen
Enter Customer Contact:
91234567
Does the customer have membership? (true/false):
true
Customer is existing
They are assigned to table 1.
```

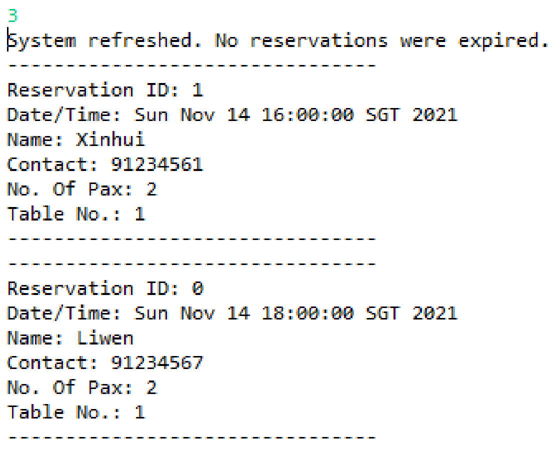
Figure 3: Reservation made at 6pm assigned to Table 1

Figure 4: Walk In assigned to Table 1

Allow reservations at a later time (Using **two tables** as example)



```
4
System refreshed. No reservations were expired.
Enter Month: 11
Enter Day: 14
Enter Hour: 12
Enter Minute: 50
Displaying Table List for Sun Nov 14 12:50:00 SGT 2021
TableNo TableSize NoOfPax Occupied Reserved CustomerName
--------------------------------------------------------
1       2         2       true     false          Liwen
2       2         2       true     false          Xinhui
```

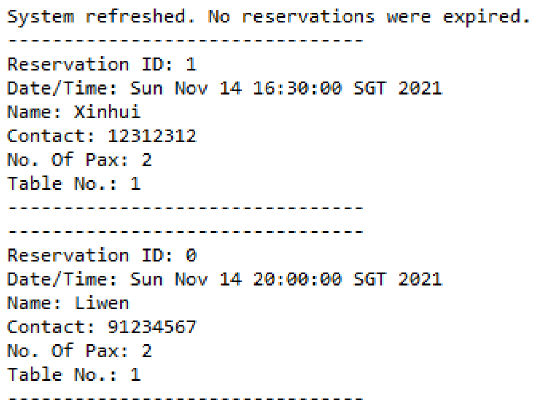


```
3
System refreshed. No reservations were expired.
-------------------------------
Reservation ID: 1
Date/Time: Sun Nov 14 16:00:00 SGT 2021
Name: Xinhui
Contact: 91234561
No. Of Pax: 2
Table No.: 1
-------------------------------
-------------------------------
Reservation ID: 0
Date/Time: Sun Nov 14 18:00:00 SGT 2021
Name: Liwen
Contact: 91234567
No. Of Pax: 2
Table No.: 1
-------------------------------
```

Figure 5: All tables occupied

Figure 6: Reservation at a later time

Multiple reservations for same table

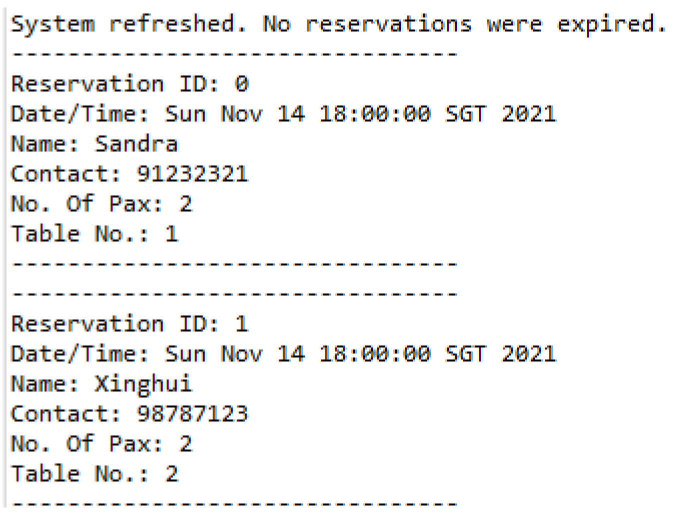


```
System refreshed. No reservations were expired.
-------------------------------
Reservation ID: 1
Date/Time: Sun Nov 14 16:30:00 SGT 2021
Name: Xinhui
Contact: 12312312
No. Of Pax: 2
Table No.: 1
-------------------------------
-------------------------------
Reservation ID: 0
Date/Time: Sun Nov 14 20:00:00 SGT 2021
Name: Liwen
Contact: 91234567
No. Of Pax: 2
Table No.: 1
-------------------------------
```

Figure 7: Reservations assigned to same table for different timings

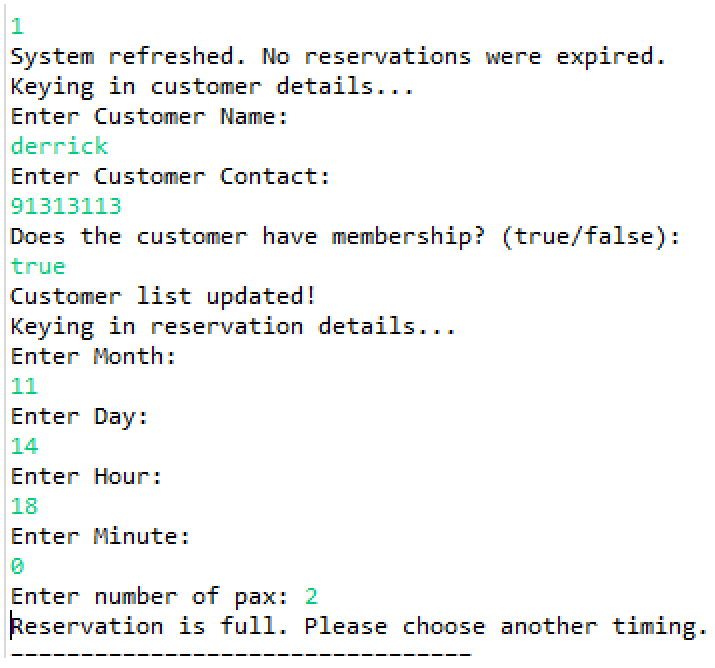
Do not allow more reservation if at that time slot reservations are full

(Using **two** tables as example)



```
System refreshed. No reservations were expired.
--------------------------------
Reservation ID: 0
Date/Time: Sun Nov 14 18:00:00 SGT 2021
Name: Sandra
Contact: 91232321
No. Of Pax: 2
Table No.: 1
--------------------------------
--------------------------------
Reservation ID: 1
Date/Time: Sun Nov 14 18:00:00 SGT 2021
Name: Xinghui
Contact: 98787123
No. Of Pax: 2
Table No.: 2
--------------------------------
```

Figure 8: All tables reserved at 6pm

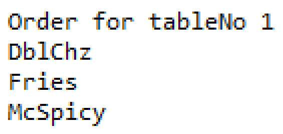


```
1
System refreshed. No reservations were expired.
Keying in customer details...
Enter Customer Name:
derrick
Enter Customer Contact:
91313113
Does the customer have membership? (true/false):
true
Customer list updated!
Keying in reservation details...
Enter Month:
11
Enter Day:
14
Enter Hour:
18
Enter Minute:
0
Enter number of pax: 2
Reservation is full. Please choose another timing.
---------------------------------
```

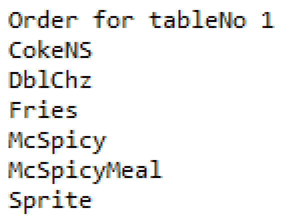
Figure 9: Reservation at 6pm where all tables are fully reserved

- **Order**

Adding more items to same order



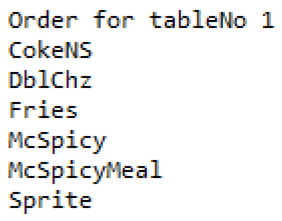
```
Order for tableNo 1
DblChz
Fries
McSpicy
```



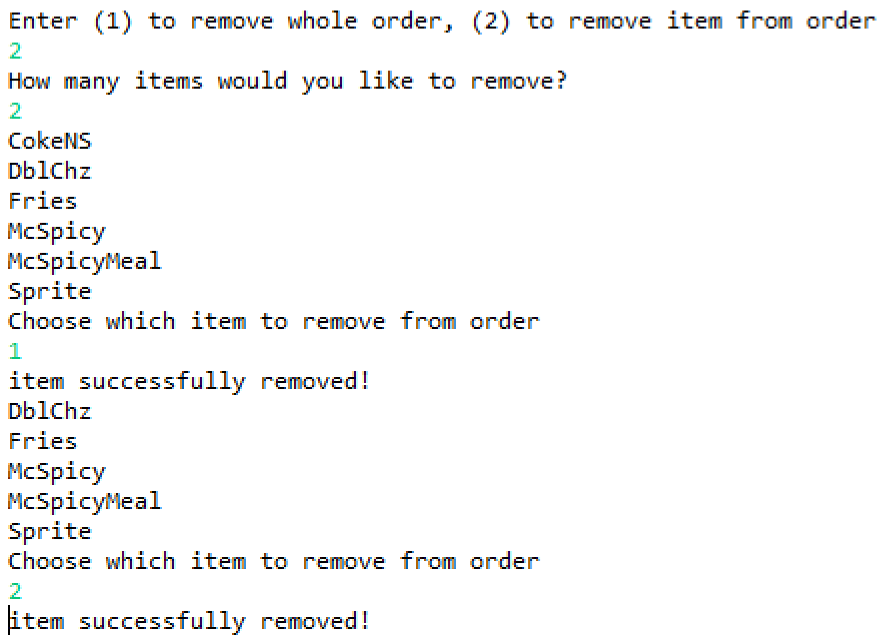
```
Order for tableNo 1
CokeNS
DblChz
Fries
McSpicy
McSpicyMeal
Sprite
```

Figure 10: More order items in order

Removing items from order

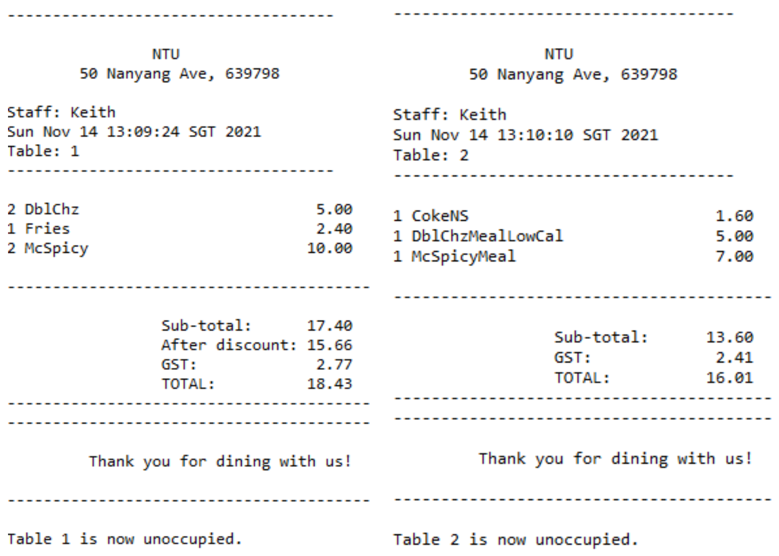


```
Order for tableNo 1
CokeNS
DblChz
Fries
McSpicy
McSpicyMeal
Sprite
```



```
Enter (1) to remove whole order, (2) to remove item from order
2
How many items would you like to remove?
2
CokeNS
DblChz
Fries
McSpicy
McSpicyMeal
Sprite
Choose which item to remove from order
1
item successfully removed!
DblChz
Fries
McSpicy
McSpicyMeal
Sprite
Choose which item to remove from order
2
item successfully removed!
```

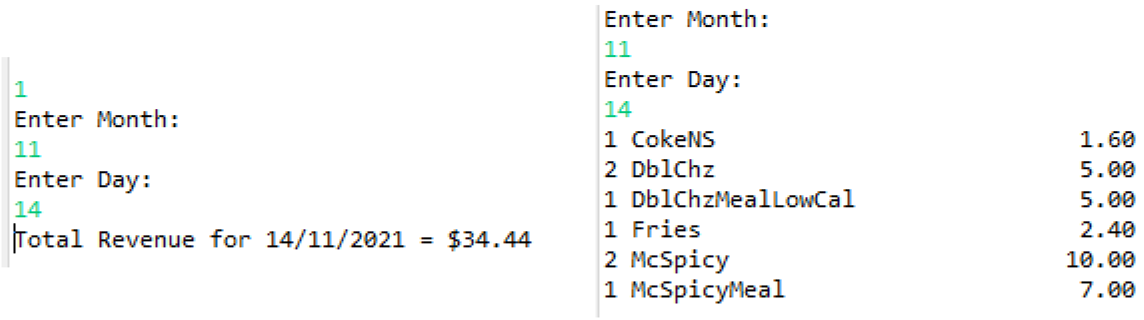
Figure 11: Removing items from order



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

                NTU                                         NTU
        50 Nanyang Ave, 639798                      50 Nanyang Ave, 639798

Staff: Keith                                Staff: Keith
Sun Nov 14 13:09:24 SGT 2021                Sun Nov 14 13:10:10 SGT 2021
Table: 1                                    Table: 2
------------------------------------        ------------------------------------

2 DblChz                          5.00      1 CokeNS                          1.60
1 Fries                           2.40      1 DblChzMealLowCal                5.00
2 McSpicy                        10.00      1 McSpicyMeal                     7.00

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

                 Sub-total:      17.40                        Sub-total:      13.60
                 After discount: 15.66                        GST:             2.41
                 GST:             2.77                        TOTAL:          16.01
                 TOTAL:          18.43      ----------------------------------------
----------------------------------------    ----------------------------------------
----------------------------------------
                                                 Thank you for dining with us!
         Thank you for dining with us!

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

Table 1 is now unoccupied.                  Table 2 is now unoccupied.
```

Figure 12: Two invoices for two different orders

- **Sales revenue report for the day (example two orders above)**



```
1
Enter Month:
11
Enter Day:
14
Total Revenue for 14/11/2021 = $34.44
```

```
Enter Month:
11
Enter Day:
14
1 CokeNS                          1.60
2 DblChz                          5.00
1 DblChzMealLowCal                5.00
1 Fries                           2.40
2 McSpicy                        10.00
1 McSpicyMeal                     7.00
```

Figure 13: Total Revenue (left), Revenue breakdown per item (right)

YOUTUBE LINK TO VIDEO: https://youtu.be/bcFLmTTZhU8