OODP ASSIGNMENT REPORT(AY 2018/2019 SEMESTER 2)

CE/CZ2002: Object-Oriented Design & Programming

Building an OO Application

Group Members:

SCSE LOW CHEE KWAN

SCSE POH YING XUAN

SCSE SHEARMAN CHUA WEI JIE

SoH YONG WEISHENG, WILSON

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
LOW CHEE KWAN	CZ2002	FSP5	Kwan
POH YING XUAN	CZ2002	FSP5	PoL
SHEARMAN CHUA WEI JIE	CZ2002	FSP5	Shearman
YONG WEISHENG, WILSON	CZ2002	FSP5	

Design considerations

Our program is designed such that the basic components of the system are separated from the "business logic", which are defined in Manager classes. This duality can be seen, for instance, in the Order and OrderManager classes, the Reservation and ReservationManager classes, etc.

One advantage of using manager classes is that, while the basic component classes provide their own encapsulation over their internal state, managers provide an additional layer of encapsulation that grants control over how instances of the same components should be handled.

All manager classes are designed based on the singleton pattern, and contain aggregations of their respective components. Consequently, managers act as the "sole authority" for batching operations, such as writing/restoring component states to disk (or to any other medium of storage, such as databases). Managers also play the coordinating role between components. For example, ReservationManager keeps track of all Reservation instances, and it provides a factory function for creating Reservations that checks if the new Reservation would conflict (same time, same contact number etc.) with existing Reservations.

Finally, due to the frequent need to obtain user input, we have developed a UserInput helper package that "outsources" the typical read-eval-print loop used when collecting user inputs into simple method calls. This reduces the amount of clutter in "business code" by abstracting away mundane verification steps, for example, by only passing user input that satisfies a given range of integer values to the calling function.

Possible feature 1 - robust archiving of invoices using databases

Closed (paid) Orders can be written into a database to facilitate robust data querying and data analysis on customer preferences/consumption trends.

This is implementable by subclassing ReportManager and overriding the addOrder() method to commit incoming orders designated for archiving into the database. The overriding method would then invoke the inherited method using super() to allow the rest of the archiving process to proceed. This approach is conceptually similar to the interceptor pattern, and allows new behaviours to be added "invisibly" without affecting existing calling code, satisfying the Liskov Substitution Principle in the SOLID model.

An alternative approach would be to extend the Serializable interface as used by all classes that are intended to be serialized into binary objects. This interface would then extract attributes from each instance and commit them to the database. One major benefit of this would be that by satisfying the Interface Segregation Principle in the SOLID model, any class implementing Serializable would be compatible with database storage. However, this approach may introduce complexity as it may involve reflection techniques; also, not all Serializable components are intended for long-term storage (e.g. caching of unpaid Order states to disk is for session persistence, not for archiving purposes).

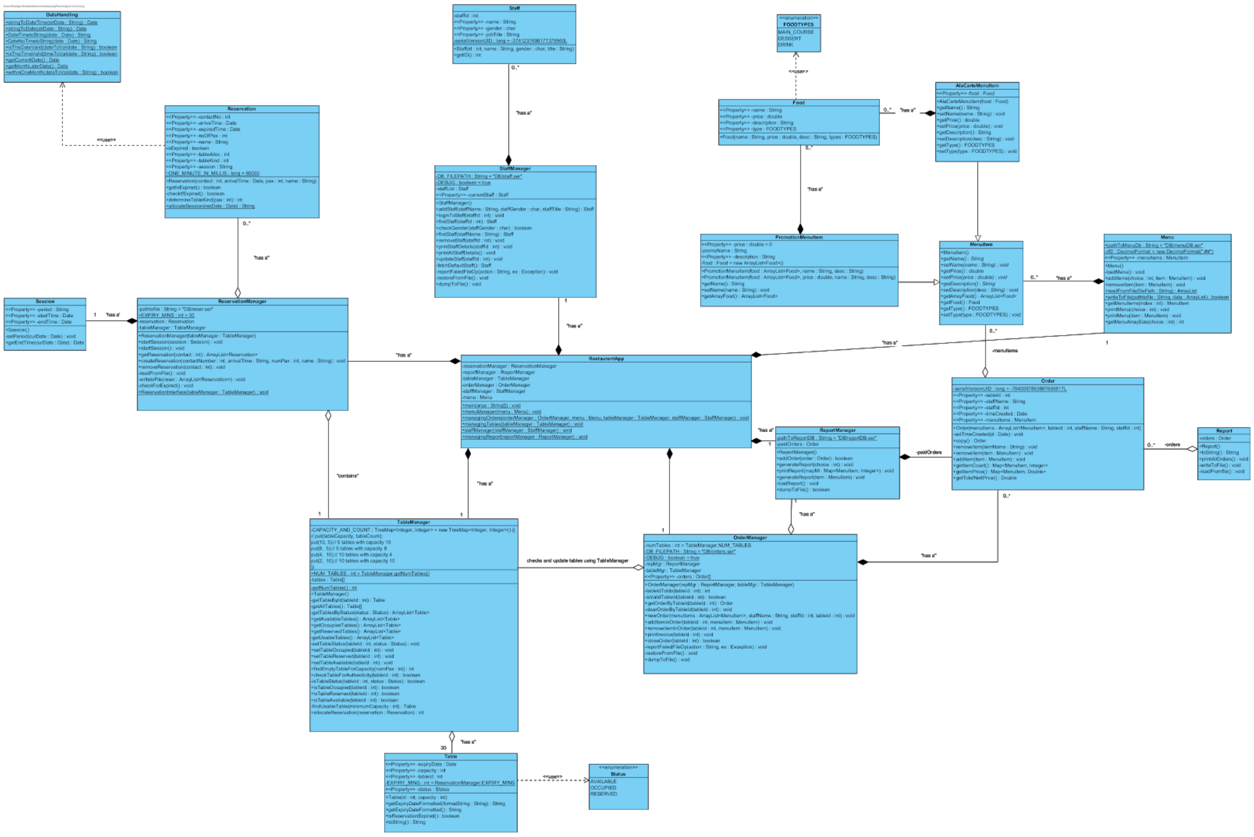
Possible feature 2 - smarter table management

Our current implementation of RRPSS makes it trivial for the number of tables at the restaurant to be updated in the system. However, as per the design specifications, tables are not allowed to be combined to serve larger groups of customers. While the current implementation tries to optimize around this constraint by trying to fill the smallest tables first, this arrangement leaves much to be desired. It would be better if a more flexible table-assignment algorithm could be implemented.

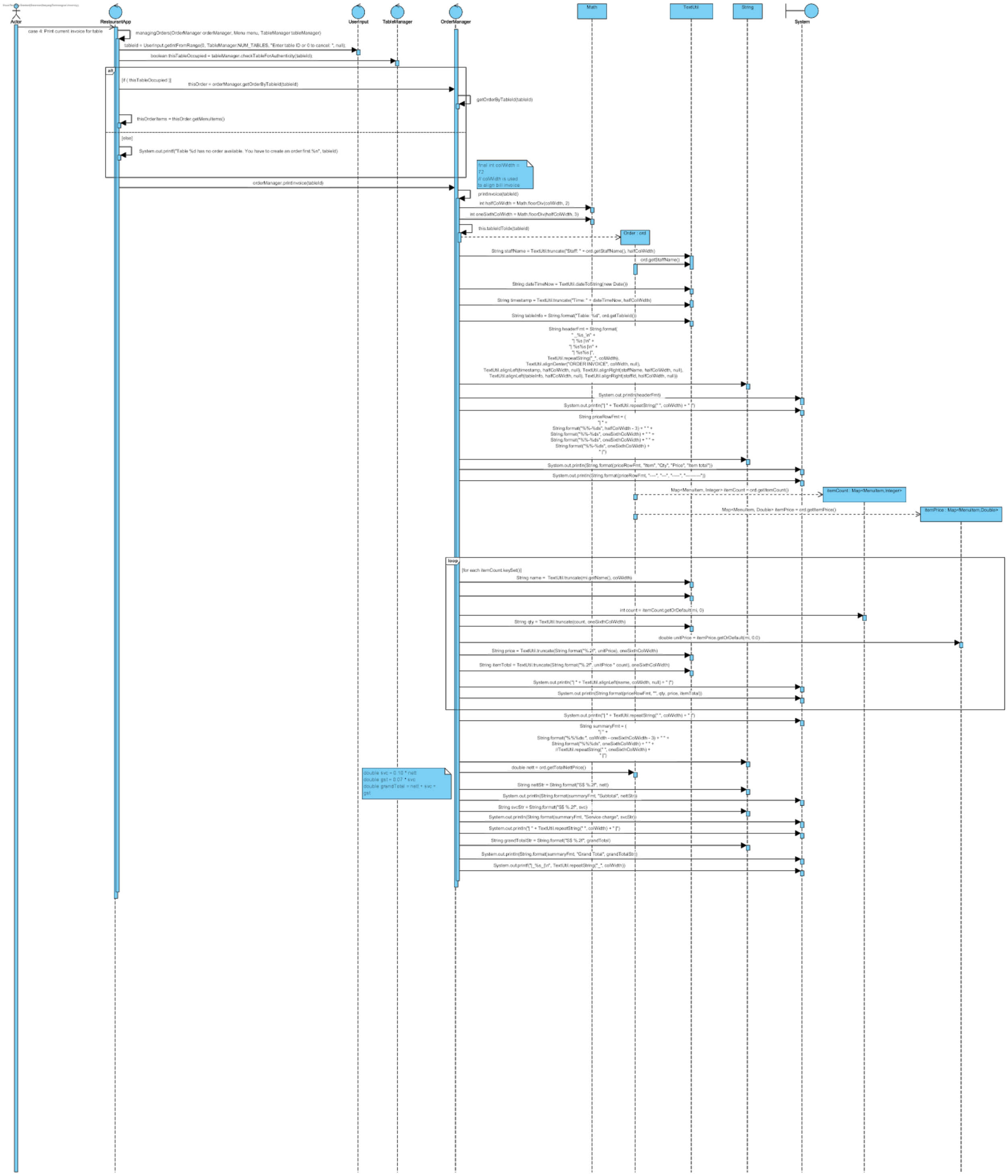
Based on the current implementation, the table-finding algorithm is encapsulated in the findEmptyTableForCapacity() method of TableManager. Therefore, this allows us to subclass and override this method to implement a new algorithm that may be as simple as naively combining available tables to fit a large group of customers. It could also be sophisticated, using graph theory to search for possible combinations of adjacent unoccupied tables.

In either case, the loose coupling between the public table-getting interface and the internal table-searching algorithm allows different approaches to table assignment that could lead to more efficient use of available tables.

Class diagram


DateHandling
+stringToDateTime(strDate : String) : Date
+stringToDate(strDate : String) : Date
+DateTimetoString(date : Date) : String
+DateNoTimetoString(date : Date) : String
+isThisDateValid(dateToValidate : String) : boolean
+isThisTimeValid(timeToValidate : String) : boolean
+getCurrentDate() : Date
+getMonthLaterDate() : Date
+withinOneMonth(dateToValidate : String) : boolean
<<use>>
Reservation
<<Property>> -contactNo : int
<<Property>> -arriveTime : Date
<<Property>> -expiredTime : Date
<<Property>> -noOfPax : int
<<Property>> -name : String
-isExpired : boolean
<<Property>> -tableAlloc : int
<<Property>> -tableKind : int
<<Property>> -session : String
-ONE_MINUTE_IN_MILLIS : long = 60000
+Reservation(contact : int, arrivalTime : Date, pax : int, name : String)
+getIsExpired() : boolean
-checkIfExpired() : boolean
+determineTableKind(pax : int) : int
+allocateSession(resDate : Date) : String
0..*
"has a"
Session
<<Property>> -period : String
<<Property>> -startTime : Date
<<Property>> -endTime : Date
+Session()
-setPeriod(curDate : Date) : void
+getEndTime(curDate : Date) : Date
1
'has a'
ReservationManager
-pathtofile : String = "DB/reser.ser"
+EXPIRY_MINS : int = 30
-reservation : Reservation
-tableManager : TableManager
+ReservationManager(tableManager : TableManager)
+startSession(session : Session) : void
+startSession() : void
+getReservation(contact : int) : ArrayList<Reservation>
+createReservation(contactNumber : int, arrivalTime : String, numPax : int, name : String) : void
+removeReservation(contact : int) : void
-readFromFile() : void
-writetoFile(reser : ArrayList<Reservation>) : void
-checkForExpired() : void
+ReservationInterface(tableManager : TableManager) : void
"contains"
1
Staff
-staffId : int
<<Property>> -name : String
<<Property>> -gender : char
<<Property>> -jobTitle : String
-serialVersionUID : long = -3741237698171379560L
+Staff(id : int, name : String, gender : char, title : String)
+getID() : int
0..*
"has a"
StaffManager
-DB_FILEPATH : String = "DB/staff.ser"
-DEBUG : boolean = true
-staffList : Staff
<<Property>> -currentStaff : Staff
+StaffManager()
+addStaff(staffName : String, staffGender : char, staffTitle : String) : Staff
+loginToStaff(staffId : int) : void
+findStaff(staffId : int) : Staff
+checkGender(staffGender : char) : boolean
+findStaff(staffName : String) : Staff
+removeStaff(staffId : int) : void
+printStaffDetails(staffId : int) : void
+printAllStaffDetails() : void
+updateStaff(staffId : int) : void
-fetchDefaultStaff() : Staff
-reportFailedFileOp(action : String, ex : Exception) : void
-restoreFromFile() : void
+dumpToFile() : void
1
"has a"
"has a"
RestaurantApp
-reservationManager : ReservationManager
-reportManager : :ReportManager
-tableManager : TableManager
-orderManager : OrderManager
-staffManager : StaffManager
-menu : Menu
+main(args : String[]) : void
+menuManager(menu : Menu) : void
+managingOrders(orderManager : OrderManager, menu : Menu, tableManager : TableManager, staffManager : StaffManager) : void
+managingTables(tableManager : TableManager) : void
+staffManager(staffManager : StaffManager) : void
+managingReport(reportManager : ReportManager) : void
"has a"
1
<<enumeration>>
FOODTYPES
MAIN_COURSE
DESSERT
DRINK
<<use>>
Food
<<Property>> -name : String
<<Property>> -price : double
<<Property>> -description : String
<<Property>> -type : FOODTYPES
+Food(name : String, price : double, desc : String, types : FOODTYPES)
0..*
"has a"
AlaCarteMenuItem
<<Property>> -food : Food
+AlaCarteMenuItem(food : Food)
+getName() : String
+setName(name : String) : void
+getPrice() : double
+setPrice(price : double) : void
+getDescription() : String
+setDescription(desc : String) : void
+getType() : FOODTYPES
+setType(type : FOODTYPES) : void
0..*
"has a"
PromotionMenuItem
<<Property>> -price : double = 0
-promoName : String
<<Property>> -description : String
-food : Food = new ArrayList<Food>()
+PromotionMenuItem(food : ArrayList<Food>, name : String, desc : String)
+PromotionMenuItem(food : ArrayList<Food>, price : double, name : String, desc : String)
+getName() : String
+setName(name : String) : void
+getArrayFood() : ArrayList<Food>
MenuItem
+MenuItem()
+getName() : String
+setName(name : String) : void
+getPrice() : double
+setPrice(price : double) : void
+getDescription() : String
+setDescription(desc : String) : void
+getArrayFood() : ArrayList<Food>
+getFood() : Food
+getType() : FOODTYPES
+setType(type : FOODTYPES) : void
0..*
"has a"
Menu
+pathToMenuDb : String = "DB/menuDB.ser"
-df2 : DecimalFormat = new DecimalFormat(".##")
<<Property>> -menuItems : MenuItem
+Menu()
-loadMenu() : void
+addItems(choice : int, item : MenuItem) : void
+removeItem(item : MenuItem) : void
+readFromFile(filePath : String) : ArrayList
+writeToFile(pathtofile : String, data : ArrayList) : boolean
+getMenuItems(index : int) : MenuItem
+printMenu(choice : int) : void
+printMenu(item : MenuItem) : void
+getMenuArraySize(choice : int) : int
"has a"
1
0..*
-menuItems
Order
-serialVersionUID : long = -7040097863887693817L
<<Property>> -tableId : int
<<Property>> -staffName : String
<<Property>> -staffId : int
<<Property>> -timeCreated : Date
<<Property>> -menuItems : MenuItem
+Order(menuItems : ArrayList<MenuItem>, tableId : int, staffName : String, staffId : int)
-setTimeCreated(d : Date) : void
+copy() : Order
+removeItem(itemName : String) : void
+removeItem(item : MenuItem) : void
+addItem(item : MenuItem) : void
+getItemCount() : Map<MenuItem, Integer>
+getItemPrice() : Map<MenuItem, Double>
+getTotalNettPrice() : Double
0..*
-orders
Report
-orders : Order
+Report()
+toString() : String
+printAllOrders() : void
-writeToFile() : void
-readFromFile() : void
"has a"
1
ReportManager
-pathToReportDB : String = "DB/reportDB.ser"
-paidOrders : Order
+ReportManager()
+addOrder(order : Order) : boolean
+generateReport(choice : int) : void
+printReport(mapMi : Map<MenuItem, Integer>) : void
+generateReport(item : MenuItem) : void
-loadReport() : void
+dumpToFile() : boolean
-paidOrders
1
"has a"
0..*
"has a"
"has a"
1
1
TableManager
-CAPACITY_AND_COUNT : TreeMap<Integer, Integer> = new TreeMap<Integer, Integer>() {{
// put(tableCapacity, tableCount);
put(10, 5);// 5 tables with capacity 10
put(8, 5);// 5 tables with capacity 8
put(4, 10);// 10 tables with capacity 4
put(2, 10);// 10 tables with capacity 10
}}
+NUM_TABLES : int = TableManager.getNumTables()
-tables : Table[]
-getNumTables() : int
+TableManager()
-getTableById(tableId : int) : Table
-getAllTables() : Table[]
-getTablesByStatus(status : Status) : ArrayList<Table>
+getAvailableTables() : ArrayList<Table>
+getOccupiedTables() : ArrayList<Table>
+getReservedTables() : ArrayList<Table>
-getUsableTables() : ArrayList<Table>
-setTableStatus(tableId : int, status : Status) : void
+setTableOccupied(tableId : int) : void
+setTableReserved(tableId : int) : void
+setTableAvailable(tableId : int) : void
+findEmptyTableForCapacity(numPax : int) : int
+checkTableForAuthenticity(tableId : int) : boolean
-isTableStatus(tableId : int, status : Status) : boolean
+isTableOccupied(tableId : int) : boolean
+isTableReserved(tableId : int) : boolean
+isTableAvailable(tableId : int) : boolean
-findUsableTable(minimumCapacity : int) : Table
+allocateReservation(reservation : Reservation) : int
checks and update tables using TableManager
1
OrderManager
-numTables : int = TableManager.NUM_TABLES
-DB_FILEPATH : String = "DB/orders.ser"
-DEBUG : boolean = true
-repMgr : ReportManager
-tableMgr : TableManager
<<Property>> -orders : Order[]
+OrderManager(repMgr : ReportManager, tableMgr : TableManager)
-tableIdToIdx(tableId : int) : int
-isValidTableId(tableId : int) : boolean
+getOrderByTableId(tableId : int) : Order
-clearOrderByTableId(tableId : int) : void
+newOrder(menuItems : ArrayList<MenuItem>, staffName : String, staffId : int, tableId : int) : void
+addItemInOrder(tableId : int, menuItem : MenuItem) : void
+removeItemInOrder(tableId : int, menuItem : MenuItem) : void
+printInvoice(tableId : int) : void
+closeOrder(tableId : int) : boolean
-reportFailedFileOp(action : String, ex : Exception) : void
-restoreFromFile() : void
+dumpToFile() : void
"has a"
30
Table
<<Property>> -expiryDate : Date
<<Property>> -capacity : int
<<Property>> -tableId : int
-EXPIRY_MINS : int = ReservationManager.EXPIRY_MINS
<<Property>> -status : Status
+Table(id : int, capacity : int)
+getExpiryDateFormatted(formatString : String) : String
+getExpiryDateFormatted() : String
+isReservationExpired() : boolean
+toString() : String
<<use>>
<<enumeration>>
Status
AVAILABLE
OCCUPIED
RESERVED

Sequence diagram


Actor
RestaurantApp
UserInput
TableManager
OrderManager
Math
TextUtil
String
System
case 4: Print current invoice for table
managingOrders(OrderManager orderManager, Menu menu, TableManager tableManager)
tableId = UserInput.getIntFromRange(0, TableManager.NUM_TABLES, "Enter table ID or 0 to cancel: ", null);
boolean thisTableOccupied = tableManager.checkTableForAuthenticity(tableId);
alt
[if (thisTableOccupied)]
thisOrder = orderManager.getOrderByTableId(tableId)
getOrderByTableId(tableId)
thisOrderItems = thisOrder.getMenuItems()
[else]
System.out.printf("Table %d has no order available. You have to create an order first.%n", tableId)
final int colWidth = 72 // colWidth is used to align bill invoice
orderManager.printInvoice(tableId)
printInvoice(tableId)
int halfColWidth = Math.floorDiv(colWidth, 2)
int oneSixthColWidth = Math.floorDiv(halfColWidth, 3)
this.tableIdToIdx(tableId)
Order : ord
String staffName = TextUtil.truncate("Staff: " + ord.getStaffName(), halfColWidth)
ord.getStaffName()
String dateTimeNow = TextUtil.dateToString(new Date())
String timestamp = TextUtil.truncate("Time: " + dateTimeNow, halfColWidth)
String tableInfo = String.format("Table: %d", ord.getTableId())
String headerFmt = String.format("_%s_\n" + "| %s |\n" + "| %s%s |\n" + "| %s%s |", TextUtil.repeatString("_", colWidth), TextUtil.alignCenter("ORDER INVOICE", colWidth, null), TextUtil.alignLeft(timestamp, halfColWidth, null), TextUtil.alignRight(staffName, halfColWidth, null), TextUtil.alignLeft(tableInfo, halfColWidth, null), TextUtil.alignRight(staffId, halfColWidth, null))
System.out.println(headerFmt)
System.out.println("| " + TextUtil.repeatString(" ", colWidth) + " |")
String priceRowFmt = ("| " + String.format("%%-%ds", halfColWidth - 3) + " " + String.format("%%-%ds", oneSixthColWidth) + " " + String.format("%%-%ds", oneSixthColWidth) + " " + String.format("%%-%ds", oneSixthColWidth) + " |")
System.out.println(String.format(priceRowFmt, "Item", "Qty", "Price", "Item total"))
System.out.println(String.format(priceRowFmt, "----", "---", "-----", "----------"))
Map<MenuItem, Integer> itemCount = ord.getItemCount()
itemCount : Map<MenuItem,Integer>
Map<MenuItem, Double> itemPrice = ord.getItemPrice()
itemPrice : Map<MenuItem,Double>
loop
[for each itemCount.keySet()]
String name = TextUtil.truncate(mi.getName(), colWidth)
int count = itemCount.getOrDefault(mi, 0)
String qty = TextUtil.truncate(count, oneSixthColWidth)
double unitPrice = itemPrice.getOrDefault(mi, 0.0)
String price = TextUtil.truncate(String.format("%.2f", unitPrice), oneSixthColWidth)
String itemTotal = TextUtil.truncate(String.format("%.2f", unitPrice * count), oneSixthColWidth)
System.out.println("| " + TextUtil.alignLeft(name, colWidth, null) + " |")
System.out.println(String.format(priceRowFmt, "", qty, price, itemTotal))
System.out.println("| " + TextUtil.repeatString(" ", colWidth) + " |")
String summaryFmt = ("| " + String.format("%%%ds:", colWidth - oneSixthColWidth - 3) + " " + String.format("%%%ds", oneSixthColWidth) + " " + //TextUtil.repeatString(" ", oneSixthColWidth) + " |")
double nett = ord.getTotalNettPrice()
double svc = 0.10 * nett double gst = 0.07 * svc double grandTotal = nett + svc + gst
String nettStr = String.format("S$ %.2f", nett)
System.out.println(String.format(summaryFmt, "Subtotal", nettStr))
String svcStr = String.format("S$ %.2f", svc)
System.out.println(String.format(summaryFmt, "Service charge", svcStr))
System.out.println("| " + TextUtil.repeatString(" ", colWidth) + " |")
String grandTotalStr = String.format("S$ %.2f", grandTotal)
System.out.println(String.format(summaryFmt, "Grand Total", grandTotalStr))
System.out.printf("|_%s_|\n", TextUtil.repeatString("_", colWidth))

Testing

On starting the Restaurant Reservation and Point of Sale System (RRPSS)

On starting up the RRPSS, you will see the following main application:

```
Welcome,now starting PM session

Starting Restaurant App RRPSS

[Main menu]
Which do you wish to access?
1. Menu Manager
2. Table Manager
3. Order Manager
4. Reservation Manager
5. Report Manager
6. Staff Manager

Enter 0 to cancel
Please enter an integer between 0 and 6 (inclusive):
```

- The main application allows the user to select the different “Manager” to access different functions related to the particular manager. (e.g. to Create menu item, user will select 1.Menu Manager > 5. Update Item from Menu Manager)

Selecting Menu Manager :

```
[Main menu]
Which do you wish to access?
1. Menu Manager
2. Table Manager
3. Order Manager
4. Reservation Manager
5. Report Manager
6. Staff Manager

Enter 0 to cancel
Please enter an integer between 0 and 6 (inclusive): 1
```

Menu Manager (creating new Menu Item):

```
What do you wish to do?
1. Print Menu
2. Add Ala Carte Item
3. Remove Item from Menu
4. Add Promotional Item
5. Update Item

ENTER 0 TO QUIT

Please enter an integer between 0 and 5 (inclusive):2
What is the Name of your food?Teriyaki Chicken Burger
What is the price of Teriyaki Chicken Burger?3.5
What is the description of Teriyaki Chicken Burger?Teriyaki samurai goodness
What is the type for Teriyaki Chicken Burger?
1-MAIN COURSE, 2-DRINK, 3-DESSERT1
```

Expected Output:

```
Added Ala carte item
Ala Carte
Item:Teriyaki Chicken Burger (Type: MAIN_COURSE)
Price:$3.5
Description:Teriyaki samurai goodness
```

Menu Manager (creating new Promotional Item):

```
What do you wish to do?
1. Print Menu
2. Add Ala Carte Item
3. Remove Item from Menu
4. Add Promotional Item
5. Update Item

ENTER 0 TO QUIT

Please enter an integer between 0 and 5 (inclusive):4
```

Ala carte items will be displayed for you to choose to add into promotional set:

```
Here is your menu:
1. Ala Carte
Item:Chicken Burger (Type: MAIN_COURSE)
Price:$2.0
Description:Freshly fried chicken

2. Ala Carte
Item:Cheese Burger (Type: MAIN_COURSE)
Price:$2.0
Description:The usual

3. Ala Carte
Item:Double Cheese Burger (Type: MAIN_COURSE)
Price:$3.5
Description:Double the trouble, double the satisfaction

4. Ala Carte
Item:Fish Burger (Type: MAIN_COURSE)
Price:$2.5
Description:Deep fried fish

5. Ala Carte
Item:Grilled Burger (Type: MAIN_COURSE)
Price:$3.0
Description:Freshly grilled chicken/beef

6. Ala Carte
Item:Coke (Type: DRINK)
Price:$1.7
Description:Just Coke

7. Ala Carte
Item:Pepsi (Type: DRINK)
Price:$1.5
Description:Hate Coke, Love Pepsi

8. Ala Carte
Item:Mountain Dew (Type: DRINK)
Price:$1.8
Description:Love it

9. Ala Carte
Item:Water (Type: DRINK)
Price:$1.0
Description:H2O

10. Ala Carte
Item:Chocolate Cake (Type: DESSERT)
Price:$2.5
Description:Mouth watering chocolate

11. Ala Carte
Item:Durian Cake (Type: DESSERT)
Price:$2.5
Description:King of all fruits

12. Ala Carte
Item:Cheese Cake (Type: DESSERT)
Price:$2.5
Description:Just Cheese

Which Menu Item do you want to include?
0 TO CONFIRM, -1 TO QUIT

Please enter an integer between -1 and 12 (inclusive): 2
```

Start adding items to the promotional set meal:

```
Please enter an integer between -1 and 12 (inclusive): 2

Cheese Burger has been added

Which Menu Item do you want to include?
0 TO CONFIRM, -1 TO QUIT

Please enter an integer between -1 and 12 (inclusive): 6

Coke has been added

Which Menu Item do you want to include?
0 TO CONFIRM, -1 TO QUIT

Please enter an integer between -1 and 12 (inclusive): 10

Chocolate Cake has been added

Which Menu Item do you want to include?
0 TO CONFIRM, -1 TO QUIT

Please enter an integer between -1 and 12 (inclusive): 0
What is the Name of your Promotion?Cheezy combo
What is the price of the promotion? 0 for price combined5
What is the description of the promotion?cheese burger and chocolate cake meal
```

Expected Output:

```
Added promotion item
Promotion Set
Promotion Name: Cheezy combo
Item:Cheese Burger (Type: MAIN_COURSE)
Item:Coke (Type: DRINK)
Item:Chocolate Cake (Type: DESSERT)
Price:$5.0
Description:cheese burger and chocolate cake meal
```

Menu Manager (Removing Menu/Promotional Item):

```
What do you wish to do?
1. Print Menu
2. Add Ala Carte Item
3. Remove Item from Menu
4. Add Promotional Item
5. Update Item

ENTER 0 TO QUIT

Please enter an integer between 0 and 5 (inclusive):3

Here is your menu:
1. Ala Carte
Item:Chicken Burger (Type: MAIN_COURSE)
Price:$2.0
Description:Freshly fried chicken
                                                                .
                                                                .
                                                                .
15. Promotion Set
Promotion Name: Cheezy combo
Item:Cheese Burger (Type: MAIN_COURSE)
Item:Coke (Type: DRINK)
Item:Chocolate Cake (Type: DESSERT)
Price:$5.0
Description:cheese burger and chocolate cake meal


What item do you want to remove?
ENTER 0 TO QUIT
Please enter an integer between 0 and 15 (inclusive): 15
```

Expected Output:

```
Removed Menu item
Promotion Set
Promotion Name: Cheezy combo
Item:Cheese Burger (Type: MAIN_COURSE)
Item:Coke (Type: DRINK)
```

```
Item:Chocolate Cake (Type: DESSERT)
Price:$5.0
Description:cheese burger and chocolate cake meal
```

Menu Manager (updating Promotional Item):

```
What do you wish to do?
1. Print Menu
2. Add Ala Carte Item
3. Remove Item from Menu
4. Add Promotional Item
5. Update Item

ENTER 0 TO QUIT

Please enter an integer between 0 and 5 (inclusive):5
Here is your menu:
1. Ala Carte
Item:Chicken Burger (Type: MAIN_COURSE)
Price:$2.0
Description:Freshly fried chicken
                                                           .
                                                           .
                                                           .
15. Promotion Set
Promotion Name: Cheezy combo
Item:Cheese Burger (Type: MAIN_COURSE)
Item:Coke (Type: DRINK)
Item:Chocolate Cake (Type: DESSERT)
Price:$5.0
Description:cheese burger and chocolate cake meal

What item do you want to remove?
ENTER 0 TO QUIT
Please enter an integer between 0 and 14 (inclusive): 14

What do you wish to update?
1 .promotion name
2. Description
3. Price
Enter 0 to cancel

Please enter an integer between 0 and 3 (inclusive): 3
What is the new price?5.5
```

Expected Output:

```
Updated promotion item
Promotion Set
Promotion Name: Cheese buddy set
Item:Cheese Burger (Type: MAIN_COURSE)
Item:Double Cheese Burger (Type: MAIN_COURSE)
Item:Coke (Type: DRINK)
Item:Pepsi (Type: DRINK)
Item:Durian Cake (Type: DESSERT)
Item:Cheese Cake (Type: DESSERT)
Price:$5.5
Description:Easy Combo
```

Menu Manager (updating Menu Item):

```
What do you wish to do?
1. Print Menu
2. Add Ala Carte Item
3. Remove Item from Menu
4. Add Promotional Item
5. Update Item

ENTER 0 TO QUIT

Please enter an integer between 0 and 5 (inclusive): 5
Here is your menu:
1. Ala Carte
Item:Teriyaki Chicken Burger (Type: MAIN_COURSE)
Price:$3.5
Description:Teriyaki samurai goodness
                                                           .
                                                           .
                                                           .
```

.

```
Which Menu Item do you want to update?
0 to quit

Please enter an integer between 0 and 15 (inclusive): 1
What do you wish to update?
1 .Product name
2. Description
3. Price
4. Food type
Enter 0 to cancel

Please enter an integer between 0 and 4 (inclusive): 1
What is the updated item name?Teriyaki chicken cutlet Burger
```

Expected Output:

```
Updated ala carte item
Ala Carte
Item:Teriyaki chicken cutlet Burger (Type: MAIN_COURSE)
Price:$3.5
Description:Teriyaki samurai goodness
```

Selecting Order Manager :

```
[Main menu]
Which do you wish to access?
1. Menu Manager
2. Table Manager
3. Order Manager
4. Reservation Manager
5. Report Manager
6. Staff Manager

Enter 0 to cancel
Please enter an integer between 0 and 6 (inclusive): 3
```

Order Manager (creating Order):

```
[Order management]
1. Create order for table
2. Add items to order
3. Remove items from order
4. Print current invoice for table
5. Close order
Enter 0 to return to main menu
Please enter an integer between 0 and 5 (inclusive): 1

Creating order...
  Enter number of pax (or 0 to cancel): 4
  Enter tableId if customer has reservation, or 0 if no reservation: 0
```

Expected Output:

```
Found table 11.
(DEBUG) OrderManager: Dumped to DB/orders.ser
Order for table 11 (4 pax) created successfully. You can now add items to this order.
Returning to order management menu.
```

Order Manager (Add item/s to order):

```
[Order management]
1. Create order for table
2. Add items to order
3. Remove items from order
4. Print current invoice for table
5. Close order
Enter 0 to return to main menu
Please enter an integer between 0 and 5 (inclusive): 2

Enter table ID or 0 to cancel: 11
Here is your menu:
1. Ala Carte
Item:Teriyaki chicken cutlet Burger (Type: MAIN_COURSE)
Price:$3.5
Description:Teriyaki samurai goodness
```

```
2. Ala Carte
Item:Chicken Burger (Type: MAIN_COURSE)
Price:$2.0
Description:Freshly fried chicken
                                                                .
                                                                .
                                                                .
14. Promotion Set
Promotion Name: Chicken Burger set
Item:Chicken Burger (Type: MAIN_COURSE)
Item:Pepsi (Type: DRINK)
Item:Durian Cake (Type: DESSERT)
Price:$6.0
Description:Easy Combo

15. Promotion Set
Promotion Name: Cheese buddy set
Item:Cheese Burger (Type: MAIN_COURSE)
Item:Double Cheese Burger (Type: MAIN_COURSE)
Item:Coke (Type: DRINK)
Item:Pepsi (Type: DRINK)
Item:Durian Cake (Type: DESSERT)
Item:Cheese Cake (Type: DESSERT)
Price:$5.5
Description:Easy Combo

Currently 0 items in order.
What item do you want add to the order for table 11?
(Enter 0 to finish adding)
14
(DEBUG) OrderManager: Dumped to DB/orders.ser

Currently 1 item in order.
What item do you want add to the order for table 11?
(Enter 0 to finish adding)
15
(DEBUG) OrderManager: Dumped to DB/orders.ser

Currently 2 items in order.
What item do you want add to the order for table 11?
(Enter 0 to finish adding)
0
2 items added to order for table 11.

Returning to main menu.
```

Order Manager (Remove item/s from order):

```
[Order management]
1. Create order for table
2. Add items to order
3. Remove items from order
4. Print current invoice for table
5. Close order
Enter 0 to return to main menu
Please enter an integer between 0 and 5 (inclusive): 3


Enter table ID or 0 to cancel: 11
1: Promotion Set
Promotion Name: Chicken Burger set
Item:Chicken Burger (Type: MAIN_COURSE)
Item:Pepsi (Type: DRINK)
Item:Durian Cake (Type: DESSERT)
Price:$6.0
Description:Easy Combo

2: Promotion Set
Promotion Name: Cheese buddy set
Item:Cheese Burger (Type: MAIN_COURSE)
Item:Double Cheese Burger (Type: MAIN_COURSE)
Item:Coke (Type: DRINK)
Item:Pepsi (Type: DRINK)
Item:Durian Cake (Type: DESSERT)
Item:Cheese Cake (Type: DESSERT)
Price:$5.5
Description:Easy Combo
```

```
Currently 2 items in order.
Which item do you want remove from the order for table 11?
(Enter 1 to 2, or 0 to finish)
1
(DEBUG) OrderManager: Dumped to DB/orders.ser
Item removed from order: 1. Chicken Burger set
1: Promotion Set
Promotion Name: Cheese buddy set
Item:Cheese Burger (Type: MAIN_COURSE)
Item:Double Cheese Burger (Type: MAIN_COURSE)
Item:Coke (Type: DRINK)
Item:Pepsi (Type: DRINK)
Item:Durian Cake (Type: DESSERT)
Item:Cheese Cake (Type: DESSERT)
Price:$5.5
Description:Easy Combo

Currently 1 item in order.
Which item do you want remove from the order for table 11?
(Enter 1 to 1, or 0 to finish)
0
1 item removed from order for table 11.

Returning to main menu.
```

Order Manager (Print Bill Invoice):

```
[Order management]
1. Create order for table
2. Add items to order
3. Remove items from order
4. Print current invoice for table
5. Close order
Enter 0 to return to main menu
Please enter an integer between 0 and 5 (inclusive): 4

Enter table ID or 0 to cancel: 11
```

Expected Output:

```
 __________________________________________________________________________
|                              ORDER INVOICE                               |
| Time: 19-Apr-19-17:35                                     Staff: cashier |
| Table: 11                                                    Staff ID: 2 |
|                                                                          |
| Item                              Qty          Price        Item total   |
| ----                              ---          -----        ----------   |
| Cheese buddy set                                                         |
|                                   1            5.50         5.50         |
|                                                                          |
|                                                  Subtotal:      S$ 5.50  |
|                                            Service charge:      S$ 0.55  |
|                                                       GST:      S$ 0.04  |
|                                                                          |
|                                               Grand Total:      S$ 6.09  |
|__________________________________________________________________________|

Returning to order management menu.
```

Selecting Reservation Manager :

```
[Main menu]
Which do you wish to access?
1. Menu Manager
2. Table Manager
3. Order Manager
4. Reservation Manager
5. Report Manager
6. Staff Manager

Enter 0 to cancel
Please enter an integer between 0 and 6 (inclusive): 4
```

Reservation Manager (Create reservation booking):

```
(1) Book a reservation
(2) Cancel a Reservation
(3) Check Reservation
(4) Exit ReservationManager
```

```
Enter the number of your choice: 1
Assigning Reservation ..
Please enter Contact No: 90838814
Please enter date(dd/MM/yyyy) between 19/04/2019 and 19/05/2019: 20/04/2019
Please enter a time(HH:mm) between AM Session(11:00 to 15:00) or PM Session(18:00 to 22:00): 11:00
Please enter Number of pax: 5
Please enter Name: Ong
```

Expected Output:

```
Contact number: 90838814
Reservation at: Sat 2019.04.20 at 11:00:00 Session AM for 5
For: Ong
Expiring at: Sat Apr 20 11:15:00 SGT 2019
```

If fully booked for Session Expected Output:

```
1) Book a reservation
(2) Cancel a Reservation
(3) Check Reservation
(4) Exit ReservationManager

           Enter the number of your choice: 1
           Assigning Reservation ..
           Please enter Contact No: 90367190
           Please enter date(dd/MM/yyyy) between 19/04/2019 and 19/05/2019: 19/04/2019
           Please enter a time(HH:mm) between AM Session(11:00 to 15:00) or PM Session(18:00 to 22:00): 21:30
           Please enter Number of pax: 10
           Please enter Name: Gavin
           Fully booked for this session!
```

Reservation Manager (Check reservation booking):

```
(1) Book a reservation
(2) Cancel a Reservation
(3) Check Reservation
(4) Exit ReservationManager

           Enter the number of your choice: 3
           Please enter contact number to check Reservation: 90838814
```

Expected Output:

```
           Contact number: 90838814
           Reservation at: Sat 2019.04.20 at 11:00:00 Session AM for 5
           For: Ong
           Expiring at: Sat Apr 20 11:15:00 SGT 2019
```

Reservation Manager (Remove reservation booking):

```
(1) Book a reservation
(2) Cancel a Reservation
(3) Check Reservation
(4) Exit ReservationManager

           Enter the number of your choice: 2
           Please enter contact number to remove Reservation: 90838814
```

Expected Output:

```
           Removing Reservation:
           Contact number: 90838814
           Reservation at: Sat 2019.04.20 at 11:00:00 Session AM for 5
           For: Ong
```

Selecting Table Manager :

```
[Main menu]
Which do you wish to access?
1. Menu Manager
2. Table Manager
3. Order Manager
4. Reservation Manager
5. Report Manager
6. Staff Manager

Enter 0 to cancel
Please enter an integer between 0 and 6 (inclusive): 2
```

Table Manager (Check table availability):

```
[Table management]
Which do you wish to do?
```

```
1. Find available table for group
2. List occupied tables only
3. List reserved tables only
4. List available tables only
5. List all tables
Enter 0 to return to main menu
Please enter an integer between 0 and 5 (inclusive): 4
```

Expected Output:

```
Table[id=1, status=AVAILABLE, capacity=2, expiryDate=null]
Table[id=2, status=AVAILABLE, capacity=2, expiryDate=null]
Table[id=3, status=AVAILABLE, capacity=2, expiryDate=null]
Table[id=4, status=AVAILABLE, capacity=2, expiryDate=null]
Table[id=5, status=AVAILABLE, capacity=2, expiryDate=null]
Table[id=6, status=AVAILABLE, capacity=2, expiryDate=null]
Table[id=7, status=AVAILABLE, capacity=2, expiryDate=null]
Table[id=8, status=AVAILABLE, capacity=2, expiryDate=null]
Table[id=9, status=AVAILABLE, capacity=2, expiryDate=null]
Table[id=10, status=AVAILABLE, capacity=2, expiryDate=null]
Table[id=12, status=AVAILABLE, capacity=4, expiryDate=null]
Table[id=13, status=AVAILABLE, capacity=4, expiryDate=null]
Table[id=14, status=AVAILABLE, capacity=4, expiryDate=null]
Table[id=15, status=AVAILABLE, capacity=4, expiryDate=null]
Table[id=16, status=AVAILABLE, capacity=4, expiryDate=null]
Table[id=17, status=AVAILABLE, capacity=4, expiryDate=null]
Table[id=18, status=AVAILABLE, capacity=4, expiryDate=null]
Table[id=19, status=AVAILABLE, capacity=4, expiryDate=null]
Table[id=20, status=AVAILABLE, capacity=4, expiryDate=null]
Table[id=21, status=AVAILABLE, capacity=8, expiryDate=null]
Table[id=22, status=AVAILABLE, capacity=8, expiryDate=null]
Table[id=23, status=AVAILABLE, capacity=8, expiryDate=null]
Table[id=24, status=AVAILABLE, capacity=8, expiryDate=null]
Table[id=25, status=AVAILABLE, capacity=8, expiryDate=null]
Table[id=27, status=AVAILABLE, capacity=10, expiryDate=null]
Table[id=28, status=AVAILABLE, capacity=10, expiryDate=null]
Table[id=29, status=AVAILABLE, capacity=10, expiryDate=null]
Table[id=30, status=AVAILABLE, capacity=10, expiryDate=null]

28 available tables in total.

Returning to table management menu.
```

Selecting Report Manager :

```
[Main menu]
Which do you wish to access?
1. Menu Manager
2. Table Manager
3. Order Manager
4. Reservation Manager
5. Report Manager
6. Staff Manager

Enter 0 to cancel
Please enter an integer between 0 and 6 (inclusive): 5
```

Report Manager (Print sale revenue report by period):

```
[Report management]
Which do you wish to do?
1. Generate Revenue Report For a Day
2. Generate Revenue Report For a period
3. Generate Revenue Report For a Menu Item
Enter 0 to return to main menu
Please enter an integer between 0 and 3 (inclusive): 2

Please enter initial date with 'today/tomorrow/tmr/ytd/yesterday' or a custom date (dd/mm or dd/mm/yyyy):
today
Please enter final date with 'today/tomorrow/tmr/ytd/yesterday' or a custom date (dd/mm or dd/mm/yyyy):
Today
```

Expected Output:

```
 __________________________________________________________________________
|                              REVENUE REPORT                              |
|                                                                          |
| Item                              Qty          Price        Item total   |
| ----                              ---          -----        ----------   |
| Grilled Burger                                                           |
|                                   2            3.00         6.00         |
| Double Cheese Burger                                                     |
|                                   2            3.50         7.00         |
| Chicken Burger                                                           |
|                                   2            2.00         4.00         |
| Fish Burger                                                              |
|                                   2            2.50         5.00         |
| Cheese Burger                                                            |
|                                   2            2.00         4.00         |
|                                                                          |
|                                             Total Revenue:     S$ 26.00  |
|__________________________________________________________________________|
```