NANYANG TECHNOLOGICAL UNIVERSITY

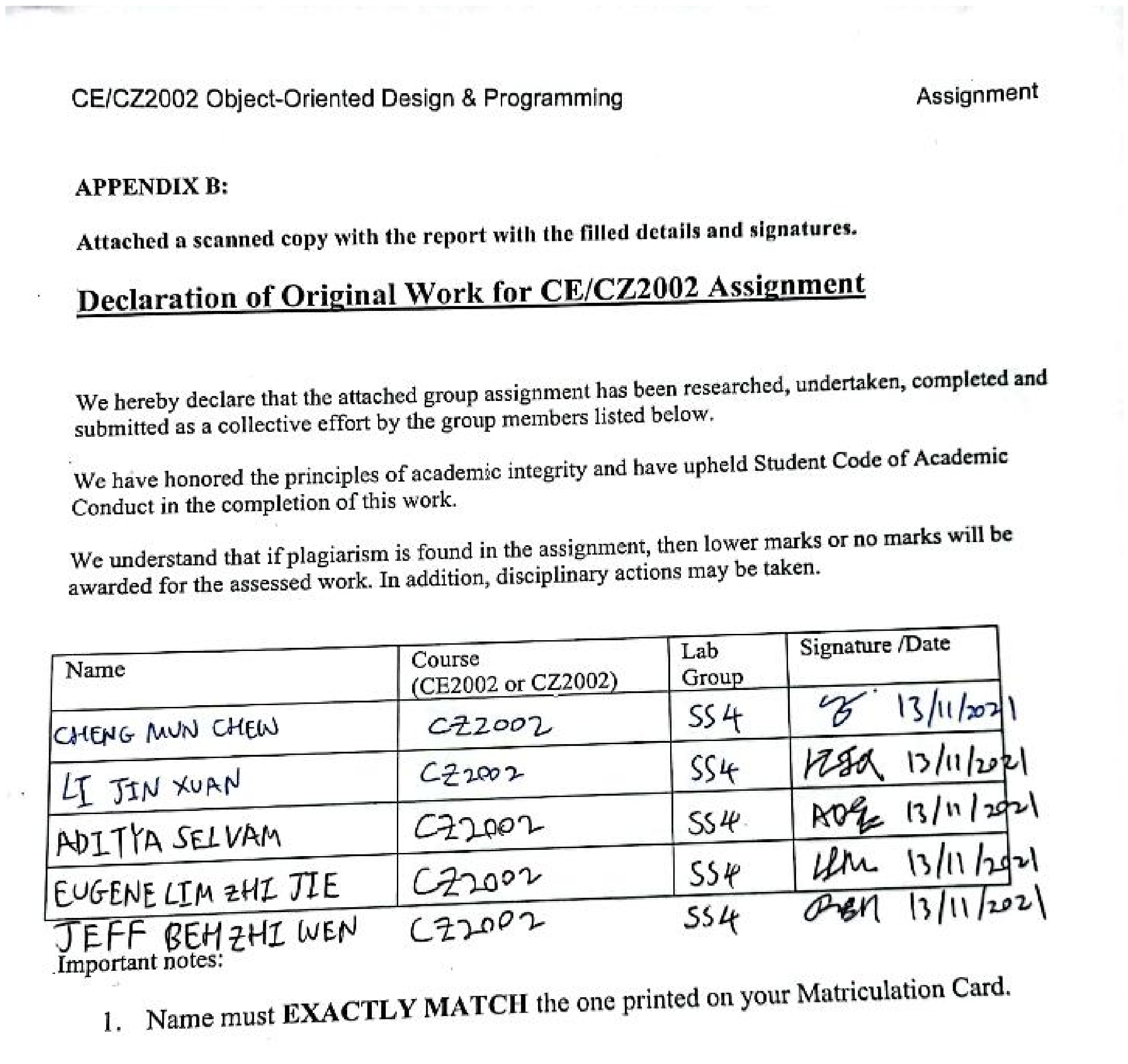
SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

ASSIGNMENT

CE/CZ2002: Object-Oriented Design & Programming
Building an OO Application

SS4 GROUP 5

CHENG MUN CHEW	U2020308K
LI JIN XUAN	U2022646F
ADITYA SELVAM	U2021126L
EUGENE LIM ZHI JIE	U2022192C
JEFF BEH ZHI WEN	U2021397D



APPENDIX B:

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

Declaration of Original Work for CE/CZ2002 Assignment

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

We have honored the principles of academic integrity and have upheld Student Code of Academic Conduct in the completion of this work.

We understand that if plagiarism is found in the assignment, then lower marks or no marks will be awarded for the assessed work. In addition, disciplinary actions may be taken.

Name	Course (CE2002 or CZ2002)	Lab Group	Signature /Date
CHENG MUN CHEW	CZ2002	SS4	[illegible] 13/11/2021
LI JIN XUAN	CZ2002	SS4	[illegible] 13/11/2021
ADITYA SELVAM	CZ2002	SS4	[illegible] 13/11/2021
EUGENE LIM ZHI JIE	CZ2002	SS4	[illegible] 13/11/2021
JEFF BEH ZHI WEN	CZ2002	SS4	[illegible] 13/11/2021

Important notes:

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

Design Considerations (Video Link: https://youtu.be/Ugf6nFzPSfI)

Our RRPSS application can be broken down into four main parts - Boundary, Entity, Controller and Database files.

Entity classes consist of nouns such as Table, Order, Customer, Reservation, Staff and Item. These classes are defined with their own unique set of attributes and methods to access and modify their variables, so that they can be instantiated for each unique instance.

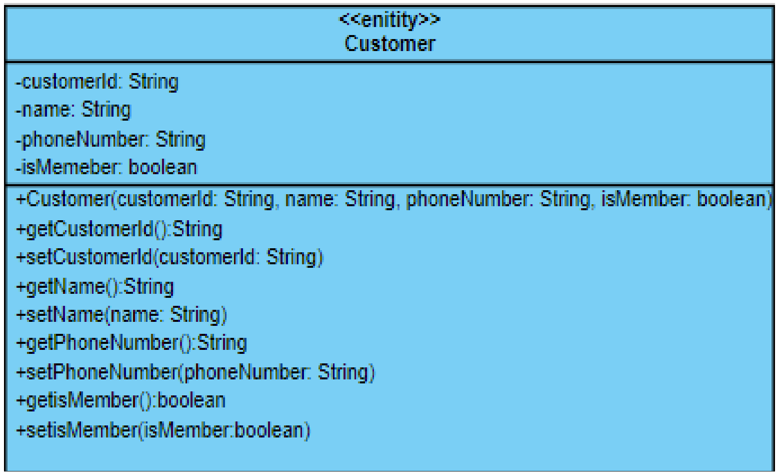
Controller classes contain the methods required to perform the functional requirements.. For example, TableController class has the updateTable() method which updates whether a table is vacant or reserved.

Database classes such as TableDB and MenuDB are inherited from the DB interface and used to process the data strings of each entity such that we can store data for future use. The ReadinFile class is used by the DB classes to read and write the data into the respective text files.

Boundary classes consist of the main RRPSSApp UI, OrderUI, MenuUI and expiredReservations. The RRPSSApp acts as the main interface of the application to allow the staff to navigate to which operation he wants to perform. OrderUI is the interface where the staff can choose to perform create/update/remove/view order operations, and print a specific order invoice. Similarly, MenuUI allows the staff to select create/update/remove menu and promotion items. For expiredReservations, it contains the method to remove past reservations automatically and set the tables that were previously reserved as vacant.

Encapsulation

1. The class variables of the entity classes are declared private so that the details of implementations are hidden from users, which helps to protect private data. For example, in the Customer entity class, we declared the customer's phoneNumber variable to be private to protect a customer's personal information.



2. In addition, we make use of public 'get' and 'set' methods to access and modify the values of these private class variables instead, as seen by the getPhoneNumber() and setPhoneNumber(String phoneNumber) methods.

Inheritance & Polymorphism

1. Order, reservation and table have different variables and types of data, the data are processed differently when read from and saved to their respective text files. Hence, we created an interface called 'DB.java', which contains abstract methods for the read and write (save) methods. We subsequently created the different subclasses like OrderDB and reservationDB that will realise the DB interface and implement the methods according to their data types. This displays the implementation of inheritance of the abstract methods from the 'DB.java' interface and the use of method overloading in each of the subclasses to achieve compile-time polymorphism when called. (Static binding.)

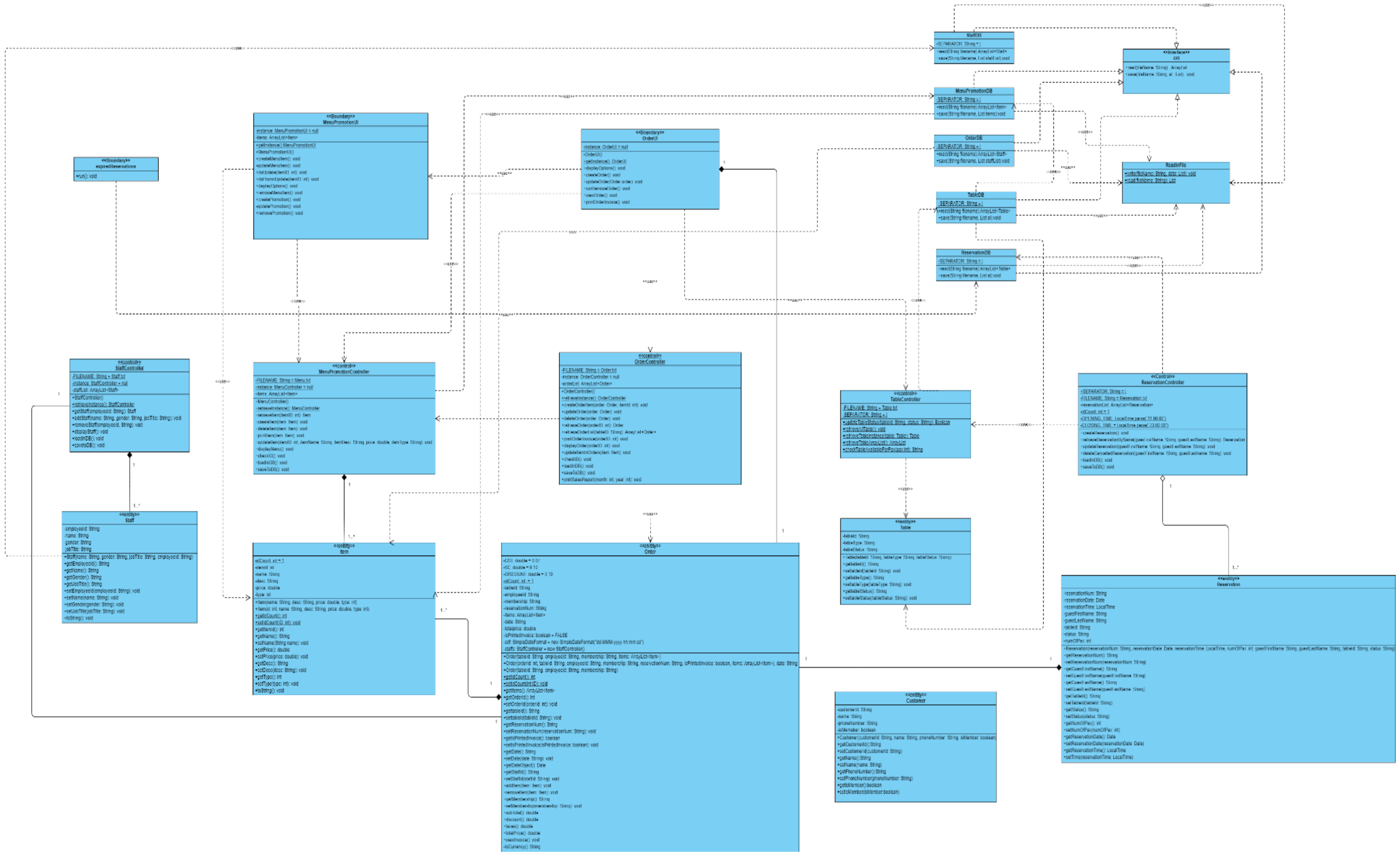
2. This DB.java interface also allows for extensibility in the future as new data can be processed by simply inheriting the DB class and overriding the methods to suit the new data type.
For example, suppose we create a new payment text file to store each payment data, we can create a paymentDB file and implement the DB interface.

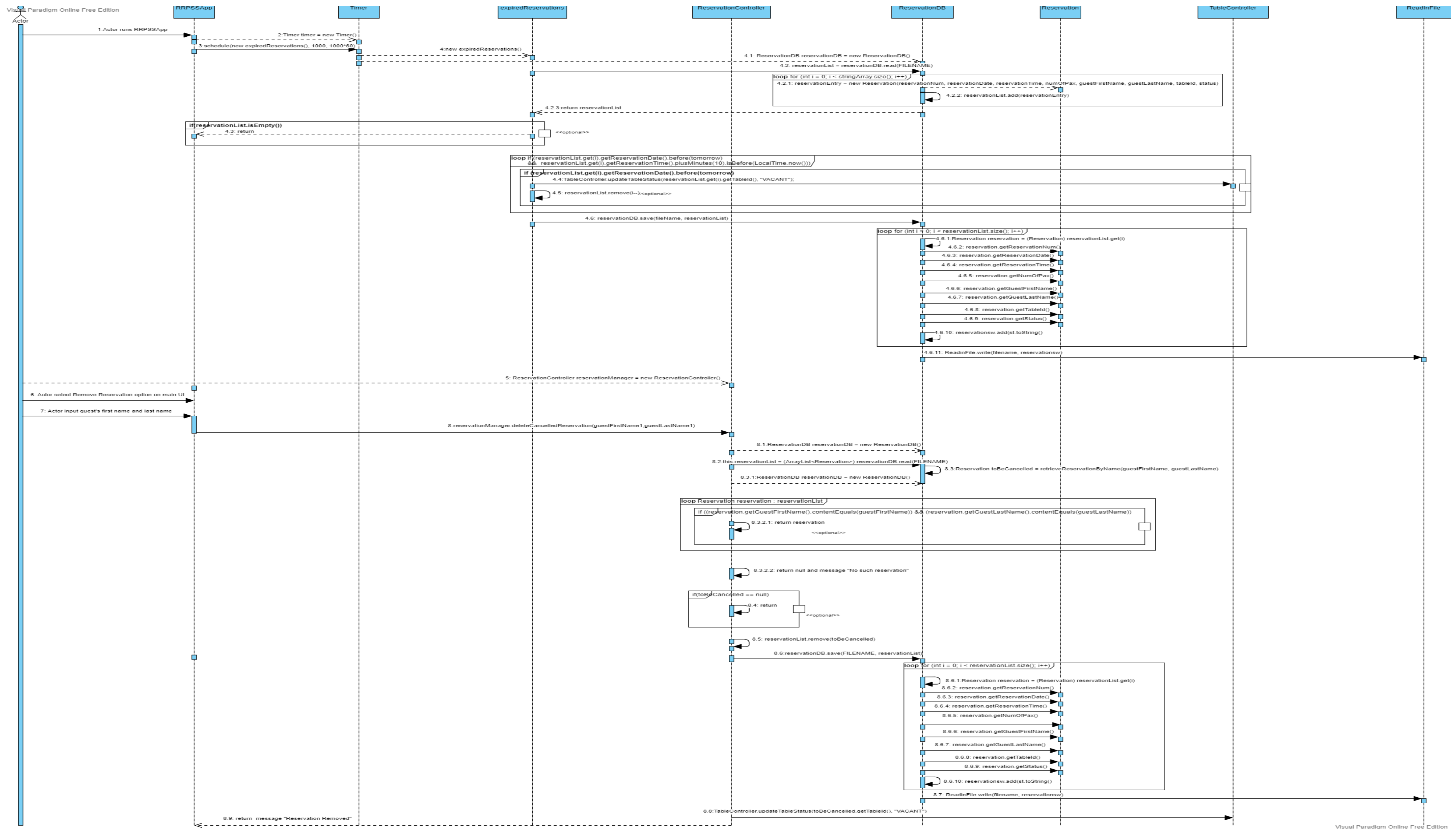
3. We used an instance of run-time polymorphism, when making use of the NumberFormat and DecimalFormat class to make sure the currency is in Singapore dollars in the toCurrency() method. This is achieved through explicit downcasting of the superclass; NumberFormat to the subclass; DecimalFormat. Afterwards, implementing a method in NumberFormat which will be overridden by the method in DecimalFormat (This will be bound during run-time.)

Low Coupling and High Cohesion

1. Our class diagram is considered quite sparse with minimal links to one another. Hence, the relationships between the classes are loosely coupled, which makes modifications to one class less likely to impact other classes. Thus will not cause a ripple effect on the other classes.

2. Classes like Order and Customer are designed such that they can easily be reused in other system designs. For example, Order and Customer are independent class entities that can also be used in an online e-commerce application. This can be achieved simply by implementing minimal modifications to the class attributes and methods.


StaffDB
<<Interface>>
DB
MenuPromotionDB
OrderDB
ReadInFile
TableDB
ReservationDB
<<Boundary>>
MenuPromotionUI
<<Boundary>>
OrderUI
<<Boundary>>
expiredReservations
<<control>>
StaffController
<<control>>
MenuPromotionController
<<control>>
OrderController
<<control>>
TableController
<<Control>>
ReservationController
<<entity>>
Staff
<<entity>>
Item
<<entity>>
Order
<<entity>>
Table
<<entity>>
Reservation
<<entity>>
Customer
<<use>>
1
1..*


Visual Paradigm Online Free Edition
Actor
RRPSSApp
Timer
expiredReservations
ReservationController
ReservationDB
Reservation
TableController
ReadInFile
1:Actor runs RRPSSApp
2:Timer timer = new Timer()
3:schedule(new expiredReservations(), 1000, 1000*60)
4:new expiredReservations()
4.1: ReservationDB reservationDB = new ReservationDB()
4.2: reservationList = reservationDB.read(FILENAME)
loop for (int i = 0; i < stringArray.size(); i++)
4.2.1: reservationEntry = new Reservation(reservationNum, reservationDate, reservationTime, numOfPax, guestFirstName, guestLastName, tableId, status)
4.2.2: reservationList.add(reservationEntry)
4.2.3:return reservationList
if(reservationList.isEmpty())
4.3: return
<<optional>>
loop if (reservationList.get(i).getReservationDate().before(tomorrow)
&& reservationList.get(i).getReservationTime().plusMinutes(10).isBefore(LocalTime.now()))
if (reservationList.get(i).getReservationDate().before(tomorrow)
4.4:TableController.updateTableStatus(reservationList.get(i).getTableId(), "VACANT");
4.5: reservationList.remove(i--)
<<optional>>
4.6: reservationDB.save(fileName, reservationList)
loop for (int i = 0; i < reservationList.size(); i++)
4.6.1:Reservation reservation = (Reservation) reservationList.get(i)
4.6.2: reservation.getReservationNum()
4.6.3: reservation.getReservationDate()
4.6.4: reservation.getReservationTime()
4.6.5: reservation.getNumOfPax()
4.6.6: reservation.getGuestFirstName()
4.6.7: reservation.getGuestLastName()
4.6.8: reservation.getTableId()
4.6.9: reservation.getStatus()
4.6.10: reservationsw.add(st.toString()
4.6.11: ReadinFile.write(filename, reservationsw)
5: ReservationController reservationManager = new ReservationController()
6: Actor select Remove Reservation option on main UI
7: Actor input guest's first name and last name
8:reservationManager.deleteCancelledReservation(guestFirstName1,guestLastName1)
8.1:ReservationDB reservationDB = new ReservationDB()
8.2:this.reservationList = (ArrayList<Reservation>) reservationDB.read(FILENAME)
8.3:Reservation toBeCancelled = retrieveReservationByName(guestFirstName, guestLastName)
8.3.1:ReservationDB reservationDB = new ReservationDB()
loop Reservation reservation : reservationList
if ((reservation.getGuestFirstName().contentEquals(guestFirstName)) && (reservation.getGuestLastName().contentEquals(guestLastName))
8.3.2.1: return reservation
<<optional>>
8.3.2.2: return null and message "No such reservation"
if(toBeCancelled == null)
8.4: return
<<optional>>
8.5: reservationList.remove(toBeCancelled)
8.6:reservationDB.save(FILENAME, reservationList)
loop for (int i = 0; i < reservationList.size(); i++)
8.6.1:Reservation reservation = (Reservation) reservationList.get(i)
8.6.2: reservation.getReservationNum()
8.6.3: reservation.getReservationDate()
8.6.4: reservation.getReservationTime()
8.6.5: reservation.getNumOfPax()
8.6.6: reservation.getGuestFirstName()
8.6.7: reservation.getGuestLastName()
8.6.8: reservation.getTableId()
8.6.9: reservation.getStatus()
8.6.10: reservationsw.add(st.toString()
8.7: ReadinFile.write(filename, reservationsw)
8.8:TableController.updateTableStatus(toBeCancelled.getTableId(), "VACANT")
8.9: return message "Reservation Removed"
Visual Paradigm Online Free Edition

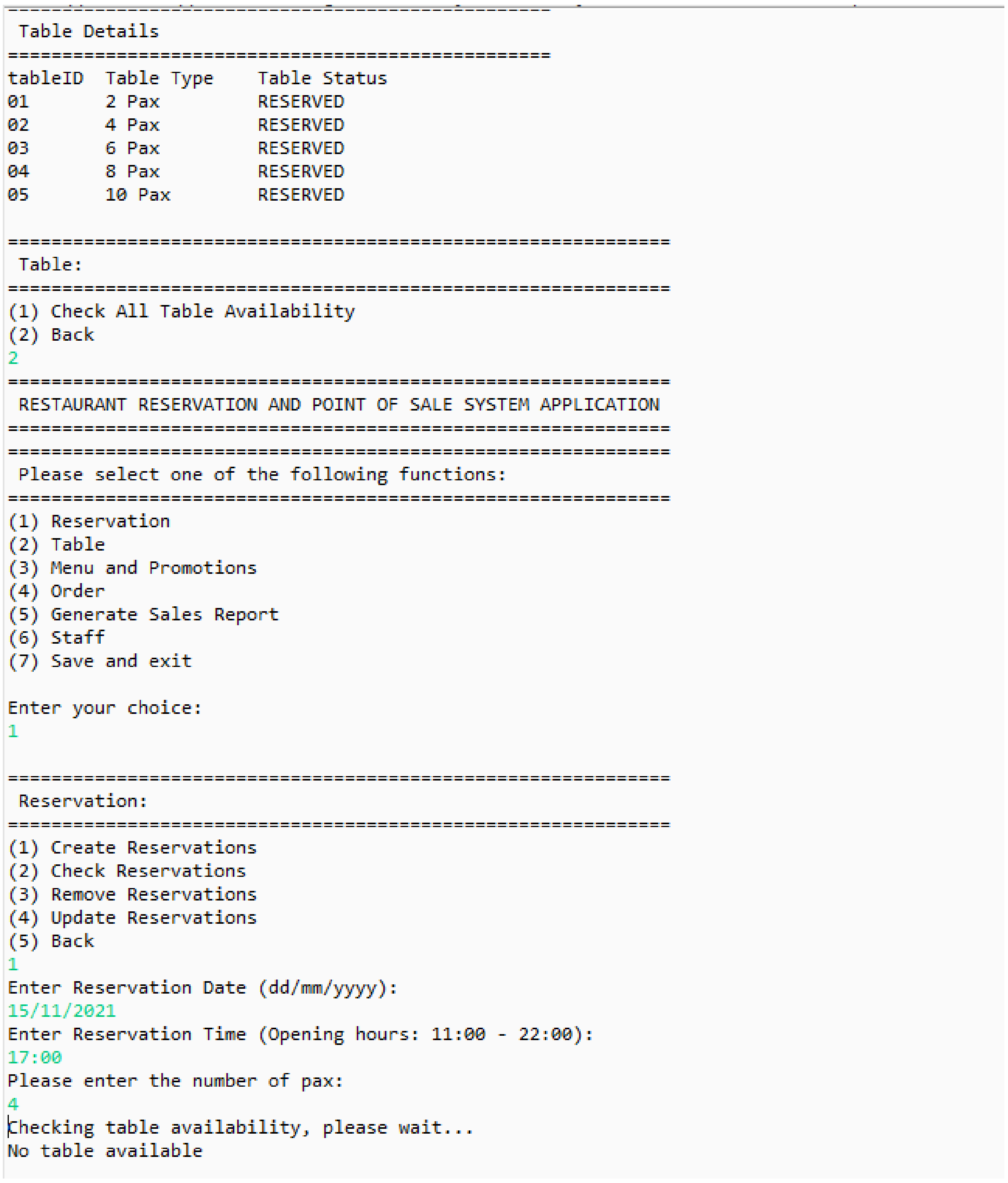
Notes:

1. Our sequence diagram ignored some of the java libraries involved such as SimpleDateFormat and Scanner to focus primarily on the interactions between the class stereotypes that we implemented.

2. The first half (1-5) of the sequence diagram shows the sequence of automatically removing an expired reservation upon running the application, while the second half (5-8) is the sequence of removing a reservation manually by a user.

Additional Test Cases and Results

1. Full Reservation



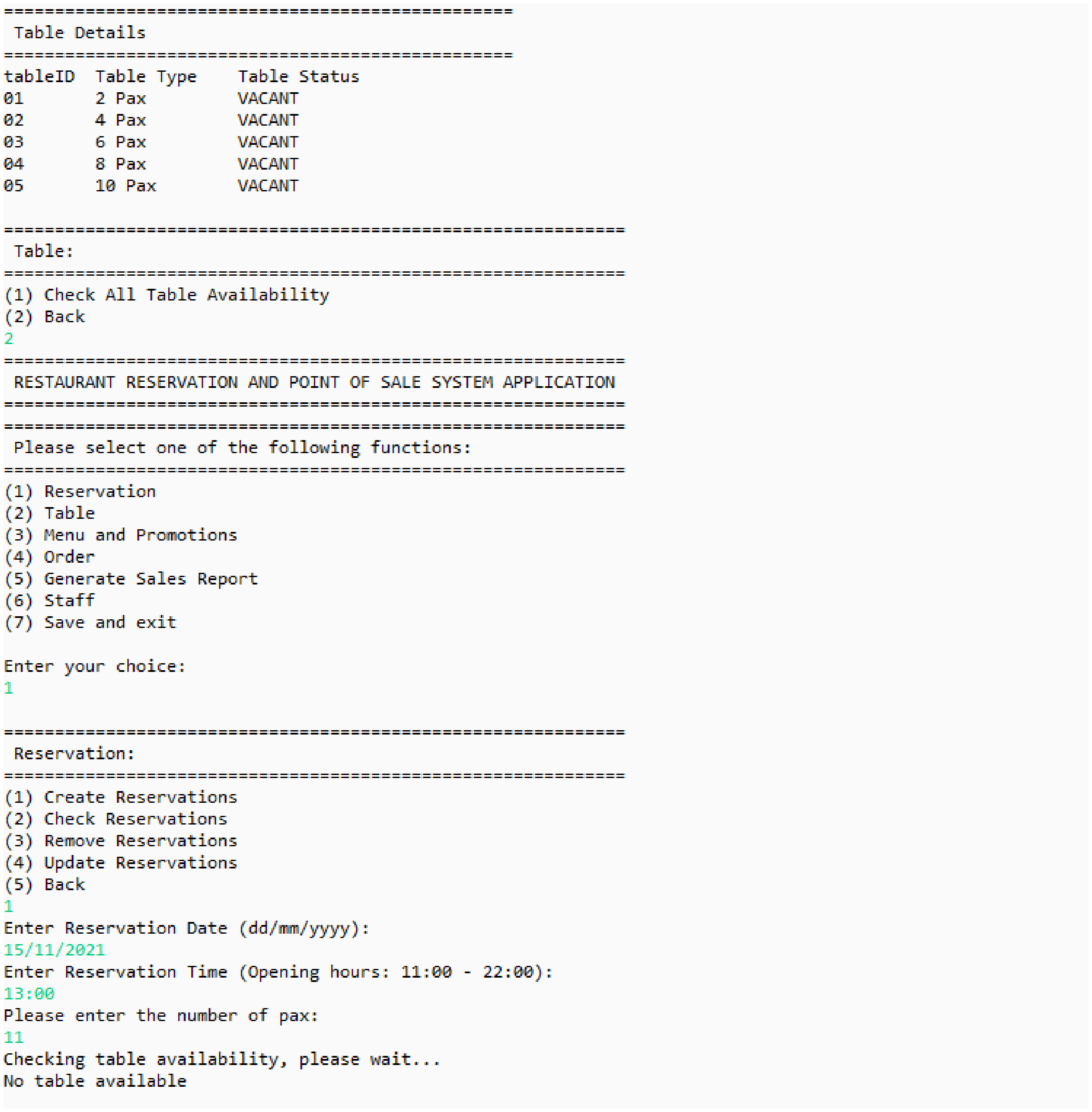
```
 Table Details
==================================================
tableID  Table Type    Table Status
01       2 Pax         RESERVED
02       4 Pax         RESERVED
03       6 Pax         RESERVED
04       8 Pax         RESERVED
05       10 Pax        RESERVED

=============================================================
 Table:
=============================================================
(1) Check All Table Availability
(2) Back
2
=============================================================
 RESTAURANT RESERVATION AND POINT OF SALE SYSTEM APPLICATION
=============================================================
=============================================================
 Please select one of the following functions:
=============================================================
(1) Reservation
(2) Table
(3) Menu and Promotions
(4) Order
(5) Generate Sales Report
(6) Staff
(7) Save and exit

Enter your choice:
1

=============================================================
 Reservation:
=============================================================
(1) Create Reservations
(2) Check Reservations
(3) Remove Reservations
(4) Update Reservations
(5) Back
1
Enter Reservation Date (dd/mm/yyyy):
15/11/2021
Enter Reservation Time (Opening hours: 11:00 - 22:00):
17:00
Please enter the number of pax:
4
Checking table availability, please wait...
No table available
```

2. Number of pax > all table sizes



```
==================================================
 Table Details
==================================================
tableID  Table Type    Table Status
01       2 Pax         VACANT
02       4 Pax         VACANT
03       6 Pax         VACANT
04       8 Pax         VACANT
05       10 Pax        VACANT

=============================================================
 Table:
=============================================================
(1) Check All Table Availability
(2) Back
2
=============================================================
 RESTAURANT RESERVATION AND POINT OF SALE SYSTEM APPLICATION
=============================================================
=============================================================
 Please select one of the following functions:
=============================================================
(1) Reservation
(2) Table
(3) Menu and Promotions
(4) Order
(5) Generate Sales Report
(6) Staff
(7) Save and exit

Enter your choice:
1

=============================================================
 Reservation:
=============================================================
(1) Create Reservations
(2) Check Reservations
(3) Remove Reservations
(4) Update Reservations
(5) Back
1
Enter Reservation Date (dd/mm/yyyy):
15/11/2021
Enter Reservation Time (Opening hours: 11:00 - 22:00):
13:00
Please enter the number of pax:
11
Checking table availability, please wait...
No table available
```