Middle East Technical University
Northern Cyprus Campus

CNG 443: Intr. to Object-Oriented Programming Languages and Systems
Assignment 1: RestManApp

Date handed-out: 30 October 2020, Friday
Date submission due: 13 November 2020, 23:55 (Istanbul time)

Learning Outcomes

On successful completion of this assignment, a student will:

- Have written a class suitable for instantiation.
- Have written a Javadoc comments for it.
- Have written code which creates and manipulates instances of this class.
- Have begun to appreciate the usefulness of reusing code, even within one class.
- Have developed a class based on use of an array as a means of storing a collection of objects.
- Have designed and written reusable methods for adding, searching for and deleting objects to/in/from the collection.
- Have defined and implemented appropriate test program to check the operation of the collection class.
- Have used a UML class diagram to implement an application.

Requirements

This assignment is about creating a small Java application for a restaurant to manage the staff, customer, bookings and order of a customer. In particular, it aims to maintain information about the customers, staff working at the restaurant, customers' bookings, and their orders. The figure below shows a summary class diagram for this application.

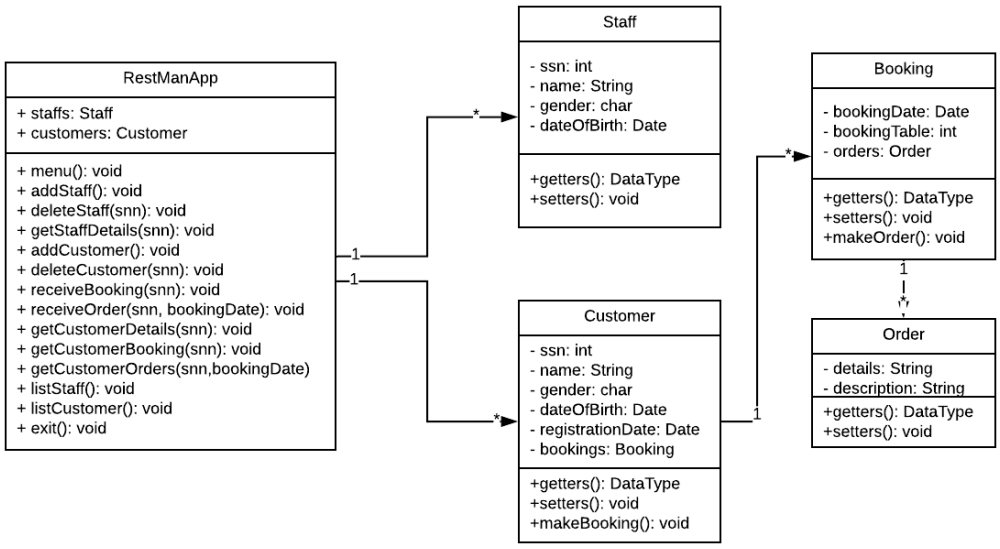

Figure 1 RestMan Application Class Diagram

The overall requirements are based on this class diagram, which is also summarised below:

- The main application called ResManApp will be used to maintain information about staff and customers. ResManApp will also have the main method and will provide the overall interaction with the application. Therefore, this class should include the main method where an instance of this class is constructed and the menu of commands is displayed to the user. Since we have not yet covered Graphical User Interfaces (GUI) in this course, you need to implement it as a command-line application. The required methods are as follows:
    - ***void menu()***: This method will display the interaction menu to the user;
    - ***void addStaff()***: This method will add new staff to the list of Staff maintained;
    - ***void deleteStaff(int snn)***: This method will read an snn number of a Staff, and delete the corresponding staff object. If the snn number does not exist, the program should provide an appropriate error message.
    - ***void getStaffDetails(int snn)***: Given an snn number, this method will display the Staff details. If the snn number does not exist, the program should provide an appropriate error message.
    - ***void addCustomer()***: This method will add a customer.
    - ***void deleteCustomer(int snn)***: Given an snn number, this method will delete a customer. If the snn number does not exist, the program should provide an appropriate error message.
    - ***void receiveBooking(int snn)***: Given an snn number, this method will take the customer's booking information. If the snn number does not exist, the method should provide an appropriate error message.
    - ***void receiveOrder(int snn, Date bookingDate)***: Given an snn number and a booking date that the customer has already provided, this method will take the customer's orders. If the snn number or the date does not exist, the method should provide an appropriate error message.
    - ***void getCustomerDetails(int snn)***: Given an snn number, this method will display the customer details. If the snn number does not exist, the method should provide an appropriate error message.
    - ***void getCustomerBooking(int snn)***: Given an snn number, this method will display a customer's bookings. If the snn number does not exist, the program should provide an appropriate error message.
    - ***void getCustomerOrders(int snn, Date bookingDate)***: Given an snn number and the booking date, this method will display a customer's orders. If the snn number or the booking date does not exist, the program should provide an appropriate error message.
    - ***void listStaff()***: This method will list all the staff.
    - ***void listCustomer()***: This method will list all the customer.
    - ***void exit()***: This method should terminate the program.

- The given class diagram has all the fields and methods needed, so please follow the diagram. If you need extra fields, you can but please make sure that you update your class diagram.

- Since you did not learn how to make your class persistent or use a database, you will lose data every time you run your application. Therefore, you need to create some objects before you start your application. Your application needs to start with 3 Staff objects, 3 Customer objects, with each Customer having one Booking and for each Booking one Order object. To create this data, you need to create a class which is called *PopulateData* that can be used to populate your application with these initial data.
- Once you complete your implementation, fully update the UML class diagram and submit it as well. Original UML diagram was created with LucidChart. You can use that or any other tool to create your updated UML diagram (e.g. Draw.io (www.draw.io), LucidChart (www.lucidchart.com/), Visio, etc.). This assignment also has an attachment that is the Visio version of this diagram so that you can import it to a tool and edit it.
    - The original diagram is also available here: https://lucid.app/documents/view/97104653-d1ca-49c1-a1b7-ffe61144def9. You can open this link, and create a duplicate to edit it (File→duplicate option).

Environment: As a development environment, you can use any IDE you like but you are strongly recommended to use **Intellij** (https://www.jetbrains.com/idea/) or **Eclipse** (http://www.eclipse.org/).

Submission: You need to submit the following:

1. Store all your Java classes (.java files) and also updated UML class diagram (.png format) in a single folder, ZIP the folder, and submit this ZIP file to ODTUCLASS.
2. Create a Jar file that can be used to execute your code. Please see the instructions below. Package all classes into a package called ResManApp.jar. With this package one should be able to run your application by just typing "java –jar ResManApp.jar".

Extra Requirements:

Some additional requirements are listed below:

- We have not yet covered how to use a Database or make objects persistent in this course. Therefore, this assignment maintains objects such as staff and customers in arrayLists.
- We have not yet covered Graphical User Interfaces (GUI) in this course. So please provide a command-line interaction (CLI).
- For each class, please decide what kind of constructors are required, the access types of methods and fields. If you use private fields, make sure that you provide accessor and mutator methods. For each class, you need to do constructor overloading and provide at least two constructors.
- You should use the Date class provided in java.utils. In order to read the date from the user, read a string with the "dd/mm/yyyy" format, in which dd, mm and yyyy represent the day, month and the year, respectively. Study the "Parsing Strings into Dates" section provided in:
    - https://www.tutorialspoint.com/java/java_date_time.html

- Pay attention to the overall design, layout and presentation of your code.
- Make sure that all your methods and fields are commented on properly, including JavaDoc commands.

Assessment Criteria

This assignment will be marked as follows:

Aspect	**Marks (Total 100)**
All classes are implemented	25
Constructors are properly implemented for all classes. There should be at least two different constructors for each class.	10
All methods are implemented	25
Command line interaction and menu	10
Initial data population is done (3 Customer, 3 Staff, 3 Booking and 3 Order objects)	10
Package	10
PopulateData Class	5
UML Class Diagram	5

For each of the items above, we will also use the following grading criteria:

Half working	%20
Fully working	%20
Appropriate reuse of other code	%10
Good Javadoc comments	%10
Good quality code	%40