Middle East Technical University
Northern Cyprus Campus

CNG 443: Intr. to Object-Oriented Programming Languages and Systems
Assignment 2: RestManApp -- Restructuring

Date handed-out: 18 November, 2020
Date submission due: 4 December 2020, 23:55 (Istanbul time)

Learning Outcomes

On successful completion of this assignment, a student will:

- Have used an UML class diagram to implement an application.
- Have practiced class hierarchy and the relevant design and implementation decisions.
- Have learnt how to maintain different types of objects.
- Have practiced and used abstract classes, and interfaces.
- Have learnt how to create a package for an application.
- Have also practiced Exception handling in Java.
- Have learnt how to draw a UML sequence diagram.

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. In overall, the idea is similar to the previous assignment but the class diagram has been significantly changed. The figure below shows a summary class diagram for this application.

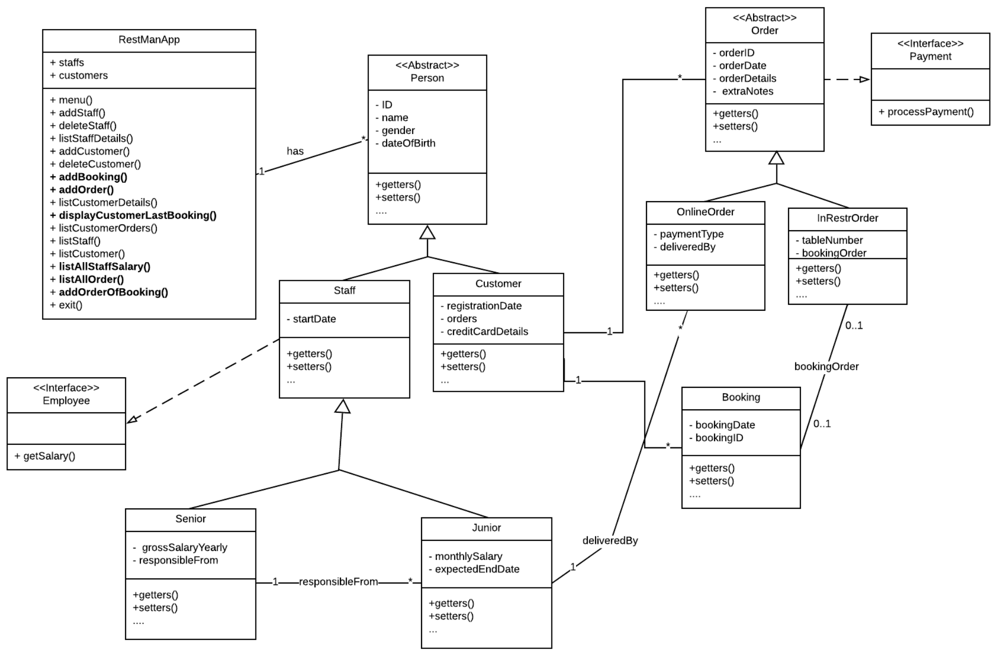

Figure 1 Updated RestMan Application Class Diagram

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. Please note that the ones which are bold are the new addition compared to the previous version.
    - *menu()*: This method will display the interaction menu to the user;
    - *addStaff()*: This method will add new staff to the list of Staff maintained;
    - *deleteStaff()*: This method will be used to delete a staff.
    - *listStaffDetails()*: This method will display the details of a staff (In the previous assignment, this was referred as getStaffDetails).
    - *addCustomer()*: This method will add a customer.
    - *deleteCustomer()*: This method will delete a customer.
    - ***addBooking()*****:** This method will receive the details of a customer and create a booking for that customer. The method needs to also get the relevant booking details.
    - *listCustomerDetails()*: This method will display a customer details (In the previous assignment, this was called getCustomerDetails).
    - ***displayCustomerLastBooking***(): This method will display the most recent booking done by the customer.
    - *listCustomerOrders()*: This method will display the orders made by a particular customer (In the previous assignment, this was referred as getCustomerOrder).
    - *listStaff()*: This method will list all the staff.
    - *listCustomer()*: This method will list all the customer.
    - *exit()*: This method should terminate the program.
    - ***addOrder()*****:** This method will get the details of a customer and an order, and will create an order for that customer. In this case, the order type has to be specified, if it is online then an online order should be created. If it is being done within the restaurant, then it should be an InRestrOrder.
    - ***listAllSaffSalary()*****:** This method will display the salary details of all staff in the application. Depending on whether the staff is junior or senior, the salary calculation will be done via ***getSalary()*** method specified in the interface as follows:
        - If the staff is a seniorthen the monthly salary is calculated as follows: for every year who has been a staff (currentDate-StartDate to find the number of years) who gets %10 increment on the grossSalaryYearly. This will be divided by 12. For example, if the employee is working for 2 years and the grossSalaryYearly is 12000, then he will have a salary of 1200

per month. This is because he received 1200 for the first year and 1200 for the second year as increments.
        - If the Staff is a junior, then the monthlySalary value will be displayed.
    - ***listAllOrder():*** This method will display all online and in restaurant orders that have been made. This will display all the orders made within the rest. and their associated bookings (if it exists). Please note that an in restaurant order can be made without a booking.
    - ***addOrderOfBooking()*:** This method will get the details of a customer and a booking, and will create an order for that customer and booking.
- In the methods list above, the following naming convention used. Add means you add data, delete means you delete data and list means you display data.
- In this assignment, you need to do exception handling. Please make sure that all the checked exception types are handled in your methods.
- 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.
- Payment interface has a method called "***processPayment()***". Since in this assignment, we are not actually going to process payments, you just need to implement it in a simple way to display a message on the screen. That should be enough for this assignment.
- In this assignment, the customers can make a booking. Each booking is identified with a unique ID. You can use a counter to ensure that the bookingID is unique. Similarly, the orders are also maintained with a unique ID. Similar approach can be used.
- 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/invitations/accept/47b222e2-9039-4624-a28a-59979e242c72
    - 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. Organise your source code to be included in a package called "assignment" and under that package split your concrete classes into one package called "core" and "support". In "core" folder put your main concrete classes and in your "support" folder add interfaces and abstract classes.
2. Store your "assignment" package and UML diagrams (.png format) in a single folder, ZIP the folder, and submit this ZIP file to ODTUCLASS.
3. 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.
- Additionally, provide a **UML sequence diagram** for ***addOrderofBooking(*** () in your application and also discuss if it is possible to simplify this sequence diagram by making some changes to the given class diagram.

Assessment Criteria

This assignment will be marked as follows:

Aspect	**Marks (Total 100)**
All classes are implemented	10
All class hierarchies are implemented	10
All interfaces are implemented and used	10
For all classes constructors are properly implemented	10
For all classes all required data fields are implemented	10
For all classes all required methods are implemented	10
All methods in the RestManApp are implemented	25

Package Structure and Jar for Invoking the application	5
UML Sequence diagram	5
Exception Handling is done	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