Name- Rana Manav Singh

Eecs Username – rms99

Student ID – 217473125

During the implementation of the project, have your strictly follow your initial design? If not, what are the problems in your initial design?

No, I have not followed my design strictly. Rather I have switched my whole initial design from Abstract Factory design pattern to factory design pattern.

This is because -

1. The previous design chosen was abstract factory design, but the project was getting too much complicated as the there were too many methods and every method had a special signature and input. Thus, instead of making so many interfaces I just made to backend classes.
2. We very well know that a platform's dependencies must be encapsulated if an application is to be portable. These "platforms" could include things like a windowing system, an operating system, a database, and so on. But in abstract class at time or you can say too often, this encapsulation isn't planned of time, and a swarm of #ifdef case statements with options for all currently supported platforms spring up all over the code . Thus, the already made interfaces will have to be changed thoroughly. But with factory method I just have to change the 2 backend classes.
3. Thus, by simply switching from abstract factory design to factory design, I was easily able to handle so many methods, GUI Classes, and databases.
4. The design I chose this time is factory method. I simply created 3 different factories one for **ADMIN,** one for **CUSTOMER,** one for **PARKING OFFICER.**

5. All these three factories have separate user databases i.e. one database for ADMIN, one for CUSTOMER, one for PARKING OFFICER.
6. But the only database they handle commonly is the parking database.
7. The GUI classes take the input from the respective user and passes it to the required method in either one of the 2 backend classes.
8. Then the action is simply performed.

Please comment the difference between the class diagrams of your final implementation and the initial design in the midterm (If applicable).

1. The most basic difference is I used two databases in previous UML (one for users and one for parking spots). But the project was way complex than anticipated thus I had to change from 2 databases to four.

   These four databases are –

   a. Customer database – It handles the data of the customer this includes First name, Last name, email address and password. Neither admin or parking officer can open this database.
   b. Admin database- simply contains the usernames and passwords of all the system admins. No even can edit this database until I am called.
   c. Parking officer database- this contains the First name, Last name, Email address and Unique ID (That is the password) of the parking officer. No one can edit this database except for admin.
   d. Parking spots- this contains all the details of a booked parking spot. It has the spot number, time of expiry, date of booking, payment status, card used, email of the person who booked

it, vehicle number of the car standing at the spot, booking type whether manual or online and the unique Booking ID.

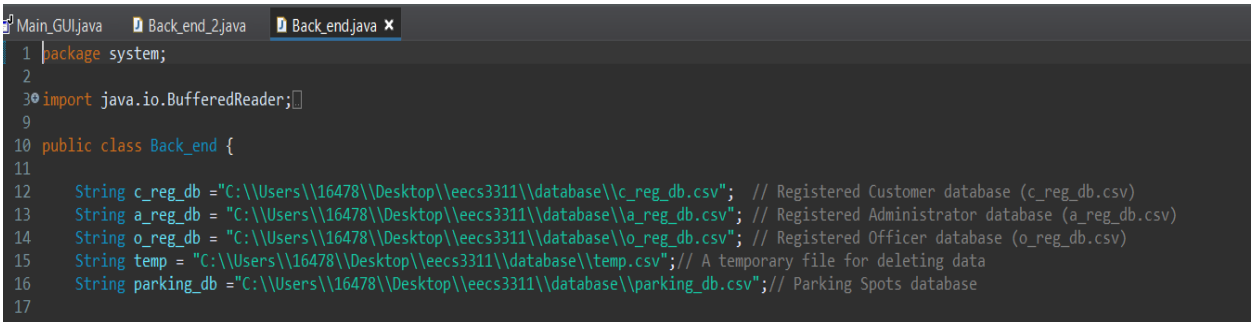
2. Instead of making interfaces I have made two backend classes that validate and update the database. Rest all the classes just take input and send it to the desired methods two backend classes who do the desired need.
3. Instead of calling the payment class again and again I have made the code in such a way that after selecting 1 spot you are asked if you want more spots. If you choose zero, then you directly go the to the payment page and pay the total. But if you choose to suppose 1 or 2 then the textfield below appears where you fill in the required info then you got to the payment page. Thus, either you book 1 spot or 2 spot or 3 spot you pay in one go.
4. So, the differences basically are to sum up-
    a. The interfaces have been removed and now we have two backend classes.
    b. Before the GUI class had specific backend code. But now that has been shifted to the two proper backend classes.
    c. Before all the classes were updating the database or using database for validation but now only the methods in the two backend classes are doing so.

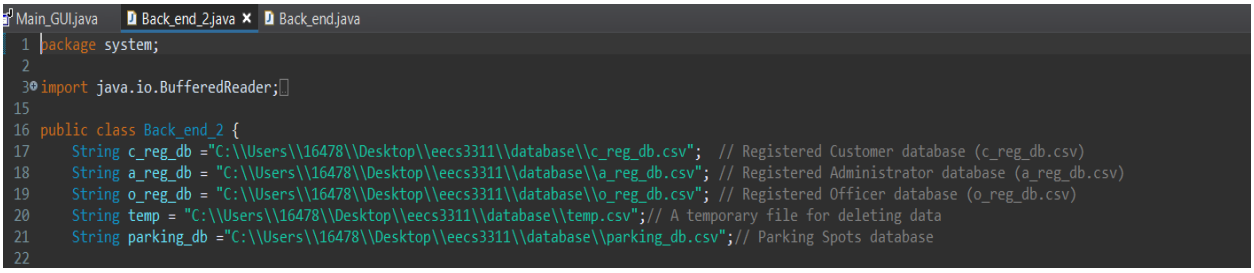
Walk Through of the application

Click here to go the video.

https://youtu.be/SUYLIIH4rQ0 (you can copy paste it as well).

I am also putting the video under the report folder.





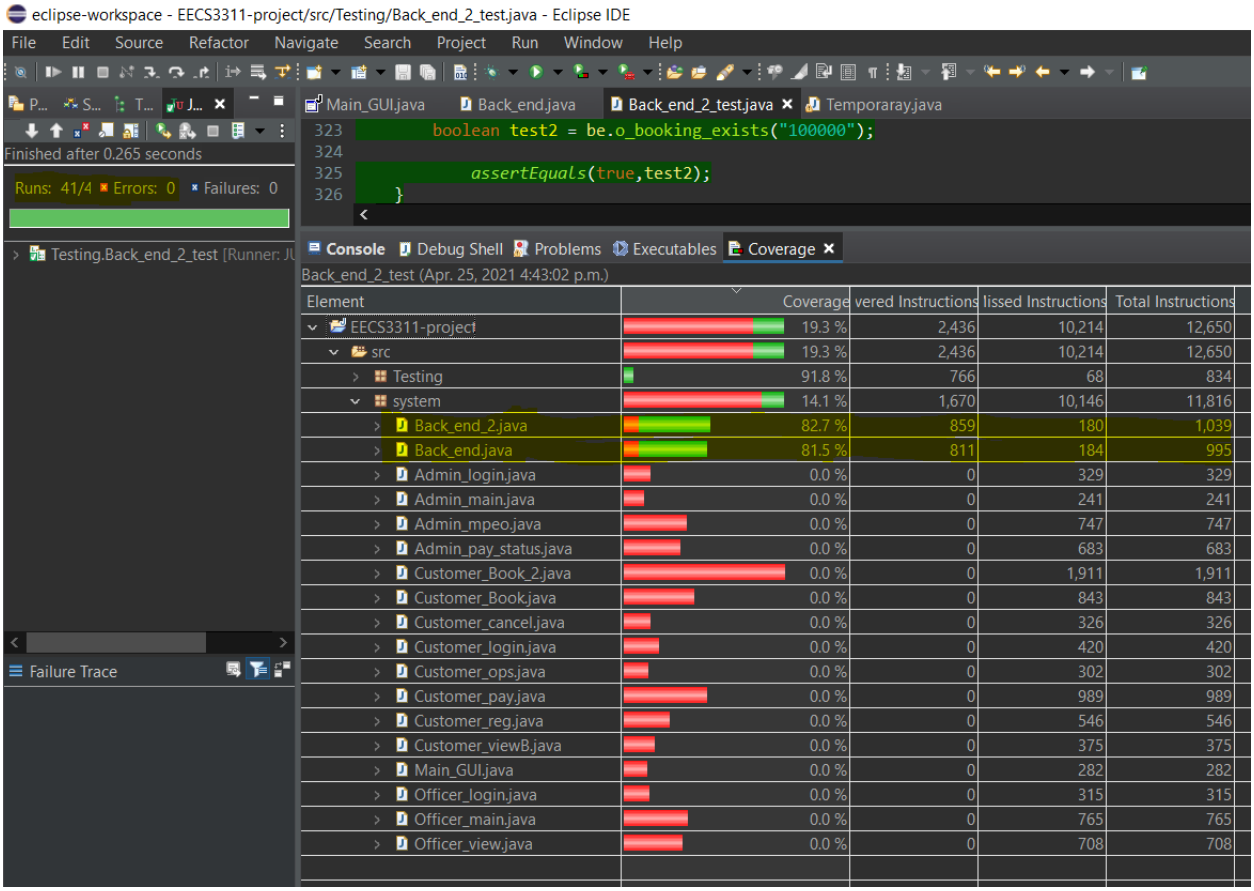
There are total 4 databases in data base folder and these are the paths you will have to edit. These are available on the top of Back_end class and Back_end_2 class. The temp.csv file will be created by the system on its own but make sure the path leads to the same folder as the other 4 databases.

Click you will get some valid credits

Testing Work-

My NON-GUI Classes are back_end and back_end_2.

I have reached 80% test coverage for both.



I am attaching the image separately in the folder as well.