Memo

Use Cases

Use cases -> Muntaser_Khan_Project_UseCases.pdf

UML Diagram

Class diagram -> Muntaser_Khan_Project_ClassDiagram.pdf

Testing

Testing folder ->Hostel\src\test\functionalTest

Source Code

Project main source code folder ->Hostel\src\ unit testing code folder ->Hostel\src\test\unitTest

Readme

Readme file -> README.txt

Memo

Status of my project

My project has been completed in respect of loading files, searching available beds and adding or searching users. It implemented main command line functionalities required by the owner of Hostel 21. These functionalities are:

1. Owner can add a user to the system or a user can register in the system. User information is stored in file.
2. Owner can load any xml files to get the hostels into the system.
3. Owner can search a user by e-mail address
4. Customers can search (though they can't book) beds in hostel rooms by dates using data loaded from the files. Search information is stored in files.
5. The source code is built in a way that customers cannot see other customers' or the owner's personal information, such as orders and accounts.

There are some details that I failed to improve. Owner of the hostel can't view, cancel and add bookings. There is no financial report of sales available.

Time spent

1. **Number of hours I needed to get the code working**

The time I worked on coding is about 6 days, 8-10 hours for each day. So the total number of hours is around 60 hours.

2. **Number of hours spent preparing your submission**

After making the soure code work through command line interface, I spent about 10 hours for writing files and finding testing code coverage, cyclomatic complexity, etc.

3. **A list of challenges faced while working on this assignment**

a) Parsing XML file was challenging, as there was node inside a node. Didn't have too much experience using xml parser so had to struggle through it

b) There are 4 entities in this project and they are hostels, bookings, customers and owner. I had to find out how to gather the relations of these entities and which design pattern I should follow. I had to study different design patterns, as I am new to the differences between them. Firstly, I created a framework to support my value object layer. Then some methods are added for the use by application layer, such as create method using for creating a booking main record and several relative detail records about it. So, booking though wasn't successfully implemented in main file, the structure has been built with a good design pattern by breaking down codes.

c) Writing the script file build.sh also was a new experience.

d) I wasn't using MySQL for the project to make it more complicated. So, I had to find a way to solve how to read and write data in files in Java project without SQL. For the data layer, although I don't use any database software, I simulate some functions like database. In my project, I extract some common functions to construct a framework. It has three sub-layers: a database table (HostelModel, HostelQueryModel etc), four basic functions: creating, deleting, updating and inquiring(Order, OrderFactory, OrderImpl etc.) and the other one for more complexity and specific purpose. All the data is stored on hard disk using java serializable method. Every time the application wants some data, it needs to read relative files from the disk. And every time it wants to change some data, it needs to write it back on the disk too.
e) Finishing the man file h21.java of command line interface was challenging as the command options were getting nested because there were a lot of if statements. Had to spend a lot of time fixing if statements by debugging through the code. Again, less experience with building command line interface was an issue
f) I had to figure out how should the data communicate in command line interface or any interface, such as reading and writing data on the interface. I did not use data access layer(dal) to communicate date directly. I built logical layer and application layer. After the accomplishment of data layer, the application is in the run. Logically the application is divided into three parts: booking, hostel and user. There is always a huge gap between application layer and data access layer, changing data from a type to another type is normal and usually happening. Thus when I write codes about application layer, I continuously set up logical layer to fulfill the need caused by the gap. Though the booking was implemented in the main file, the interface and factory files have been set up perfectly to implement in future.
g) I had to protect user/customer information. I needed to figure out how to authorize permission to whom for which command. For example, only admin can load xml files. So in application layer, I create a login interface to record password and username, they are used to control view permission on user interface. But I didn't get time to implement in the main file

4. **A list of recommendations for how to make the assignment better, to the possible benefit of future generations**

a) People who have a birthday on that day of booking will receive a discount.
b) Hold fun events in holiday or vacation and inform customer by email.
c) Introduce extra functionalities, such as booking for swimming pool use.

Junit Testing Code Coverage : 90%
Lines of Code: 1970
Lines of Code for Junit Testing: 510
Cyclomatic complexity: 1.9
Building and Running step
1. Export Eclipse project "Hostel" into the IDE and go to the default package(src folder, not to src/Hostel21).
2. Run h21.java for getting started with proper arguments
3. If you are running in command line, go to Hostel/src in terminal and run the script build.sh to get started. It will ask if you want to find the command line interface usage. Enter and run java h21 to find the possible command options.
The possible examples of command you can implement are:
java h21 admin load file1.xml
java h21 search --city "San Francisco" --start_date 20140701 --end_date 20140704
java h21 user
java h21 search
java h21 user add --first_name mun --last_name khan --email mkhan12@iit.edu
java h21 search -- email mkhan12@iit.edu

4. If you want to run the admin command option, use owner's username and password to load files
5. Run unitTest folder for unit-testing.
6. Run functionaltest folder for functionality testing