

OOM Real Estate Housing Rental System

Phase I is due on 22/03/2020

Phase II is due on 19/04/2020

Soon Qatar will be hosting the 2022 world cup. Due to this, the government is encouraging companies to move from the traditional paper-based system to more advanced computerized systems. Hence, a company named OOM Real Estate in Qatar, wants to improve their manual system into a computerized system, to manage the rental of houses as there will be an influx of people coming to Qatar and it will be hard to manage all their assets manually. So, the OOM Real Estate company asked you to help them with the implementation of the system using Java.

OOM Real Estate has different types of houses. They have one bedroom, two bedrooms, and three bedrooms apartments.



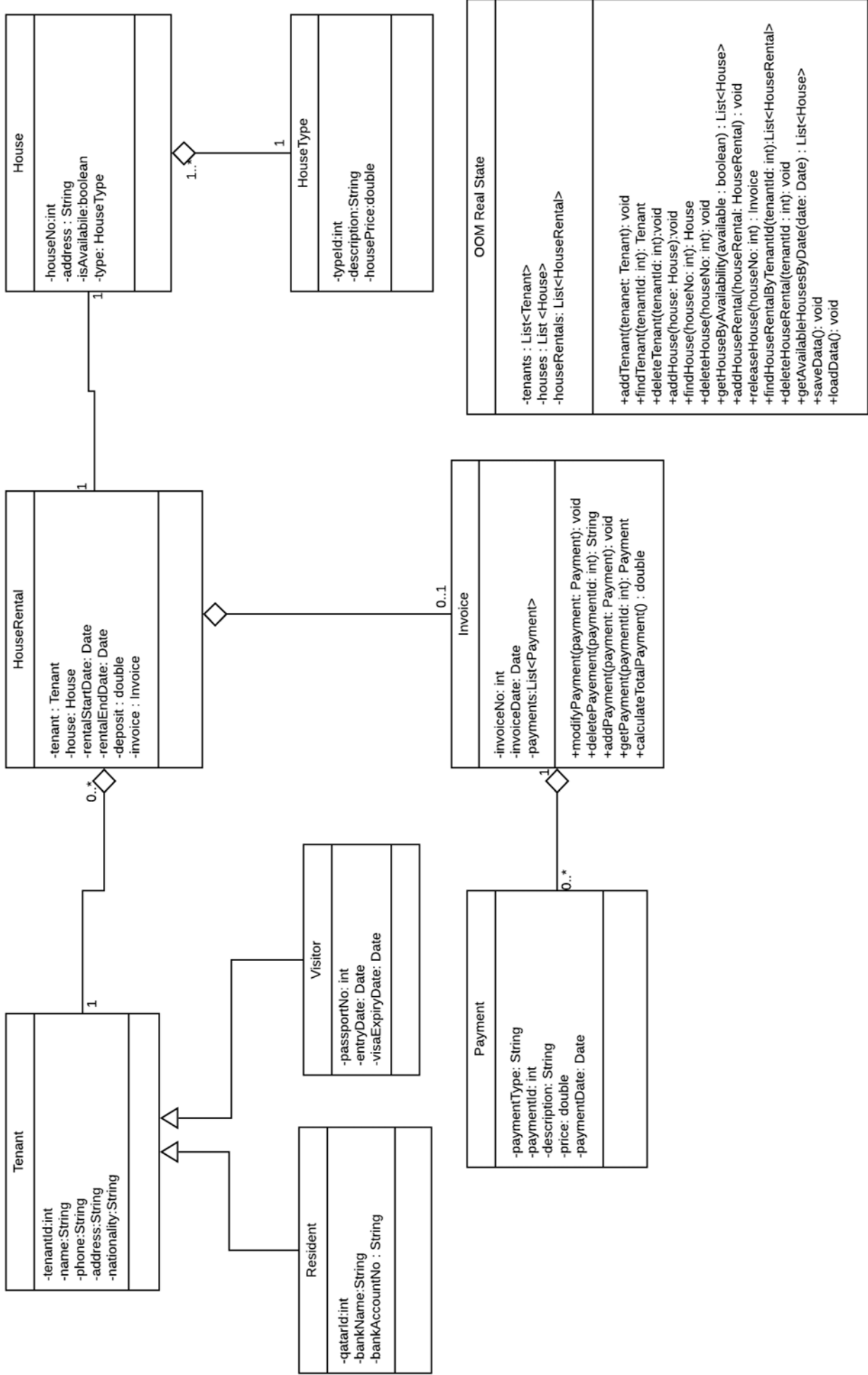
New customer or tenants' data are entered when the renting process starts. There are two types of tenants: Residents in Qatar who have the Qatari ID and Visitors. When a tenant rents a house, a rental object is created that records details of the rent, such as rent-start date, rent-end-date, house details, tenant details, and Invoice. The Payment class represents payments' details such as pay date; total amount paid, etc.

Renting Process:

Tenant details (see the class diagram) must be recorded. For resident tenants, the housing system needs to record additional details such as bank information and the tenant's Qatari ID. For visitors, the passport number, visa entry, and expiry dates are recorded. Tenants must pay a deposit amount equal to 50% of house price they are renting, which will be refunded when they release from the housing in good condition. When the tenant releases the house any damages are evaluated and the cost of repairs is deducted from the deposit amount. The remaining deposit is returned to the tenant.

The following is a brief class diagram that describes classes in this system and their relationships.

House Rental System



Project Phase I requirements

Phase I is due on 22/03/2020

1. Implement the entity classes for OOM Real State System as shown in Figure 1.

Invoice	Explanation
+modifyPayment(payment: Payment): void	Update payment information.
+deletePayment(paymentId: int): String	Delete a payment.
+addPayment(payment: Payment): void	Add a new payment to the system.
+getPayment(paymentId: int): Payment	Search for a specific tenant by their id
+calculateTotalPayment(): double	Calculate total payment for the invoice.

2. Implement the OOM Real State class as described below.

OOM Real State System	Explanation
+addTenant(tenant: Tenant): void	Add a new tenant to the system (resident or visitor).
+findTenant(tenantId: int): Tenant	Search for a specific tenant by their id.
+deleteTenant(tenantId: int): void	Delete for a specific tenant by their id.
+addHouse(house: House): void	Add a new House to the system (1 bedroom, 2 bedrooms or 3 bedrooms)
+findHouse(houseNo)	Search for a specific house by its number.
+deleteHouse(houseNo)	Delete for a specific house by its number.
+getHouseByAvailability(available: boolean) : List<House>	Get all available houses.
+releaseHouse(houseNo: int): invoice	When a tenant leaves the house, the house status is updated to available. The invoice of that house is returned
+findHouseRentalByTenantId(tenantId: int): List<HouseRental>	Returns all the houses rented by a specific tenant
+deleteHouseRental(tenantId : int): void	Deletes specific houseRental by specific tenantId.
+getAvailableHousesByDate(date: Date): List<House>	Return all houses are available after a specific date
+saveData(): void	Save data to file
+loadData(): void	Load data from file

3. Add an App class having the main method to test all the methods of the application. In phase I, any data needed should be hardcoded. The app should display the results to the console. There is no need to ask the user to enter any data during phase I.

Project Phase II requirements
Phase II is due on 19/04/2020

Your system should have a GUI interface to interact with system users and should use files to save data permanently. (*This requirement is to be finished in Phase II*)

Below are some of the screenshots for the GUI you suppose to create in phase II of the project. The following screenshot are guidelines of how the system should look like. And they are the minimal requirements. **However, if some of you would like to improve the design or implement more functionalities, then you are free to do so.**

Main UI Window

This window is loaded when you first start the application.

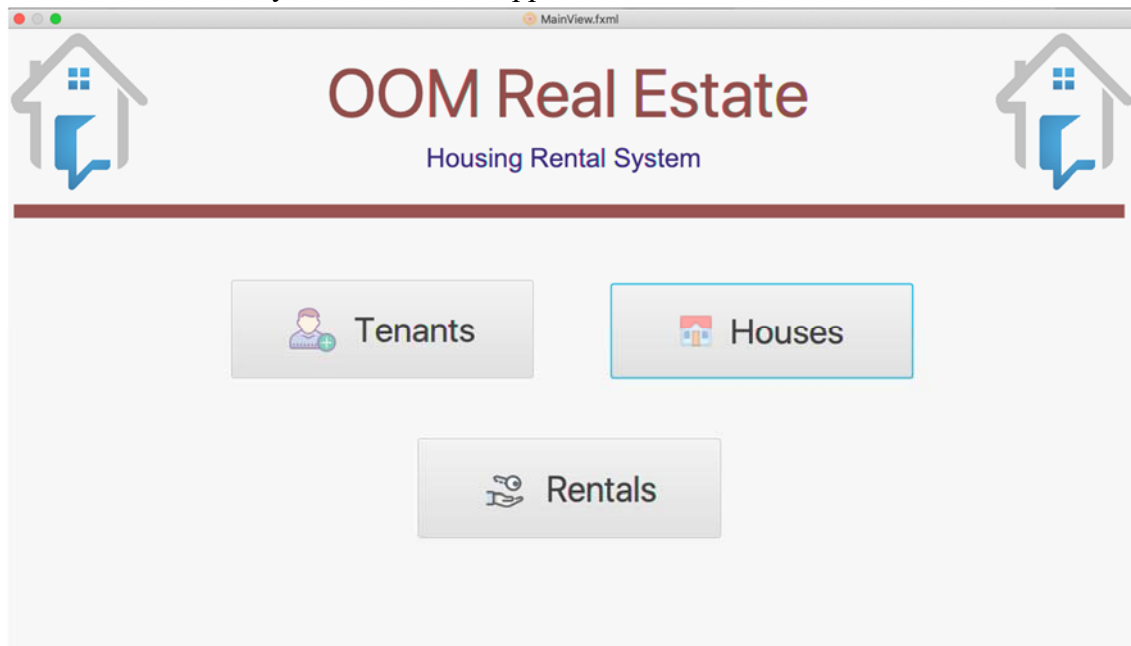


Figure 1

Main House Window

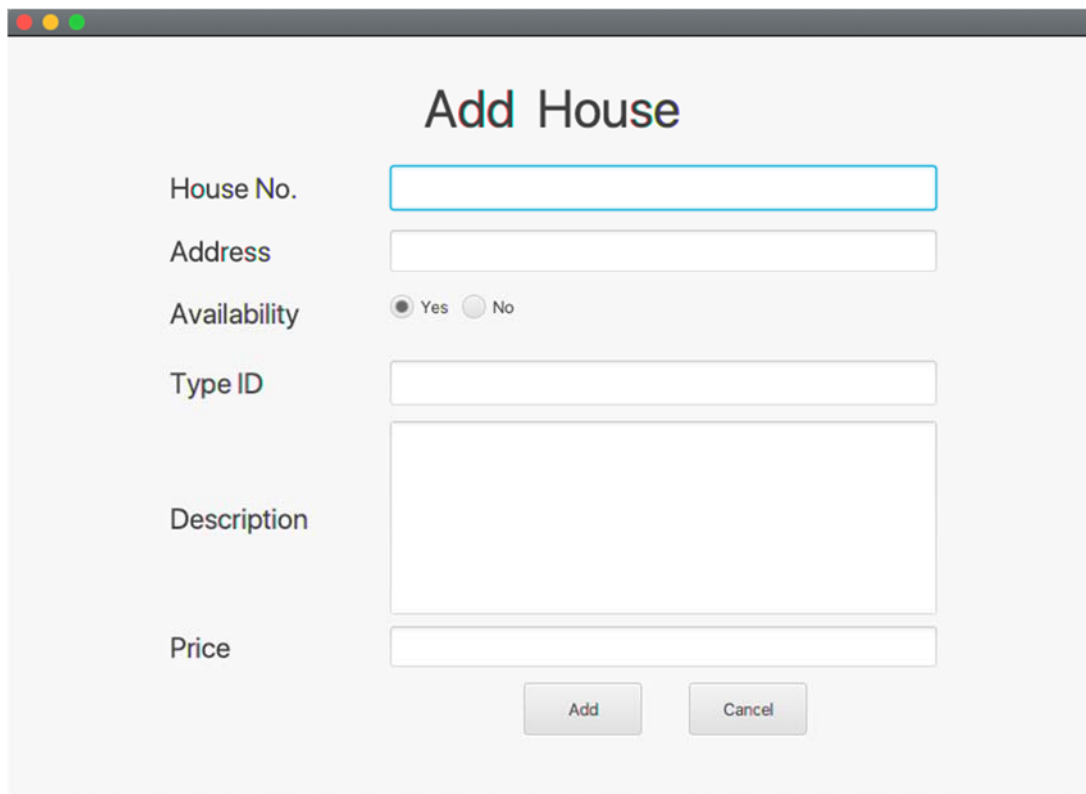
This window is shown when the user clicks on the House Button in the MainView [Fig.1].

House No	Adress	Description	Price	Availability
60	Barwa	Villa	6000.0	true
61	Barwa	Villa	6500.0	true
62	Barwa	Villa	7000.0	true
60	Barwa	Apartment	4000.0	true
61	Barwa	Apartment	3000.0	true
62	Barwa	Apartment	3500.0	true

Figure 2

Add House Window

This window is shown when the user clicks on the Add Button inside the House View shown [Fig.2]

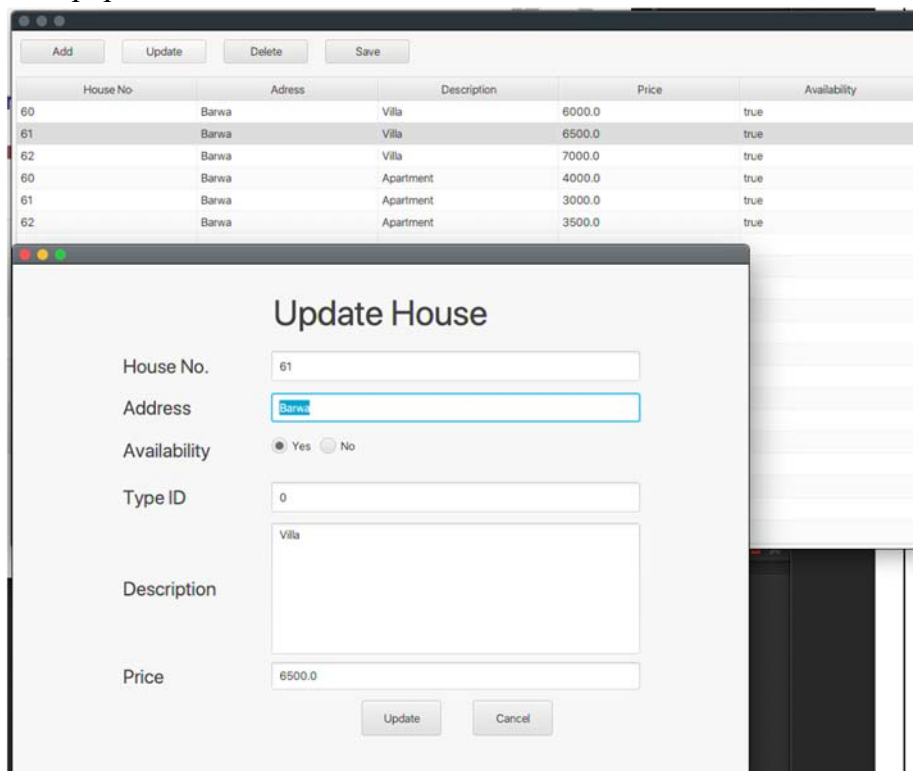


The 'Add House' window is a light gray dialog box with a title bar containing red, yellow, and green window control buttons. The title 'Add House' is centered at the top. Below the title, there are several input fields: 'House No.' (a single-line text box), 'Address' (a single-line text box), 'Availability' (radio buttons for 'Yes' and 'No', with 'Yes' selected), 'Type ID' (a single-line text box), 'Description' (a multi-line text area), and 'Price' (a single-line text box). At the bottom right, there are two buttons: 'Add' and 'Cancel'.

Figure 3

Update House Window

This window is shown when the user clicks on the Update Button inside the House View shown [Fig.2]. You will be required to populate the selected house info into a form to allow the user to edit.



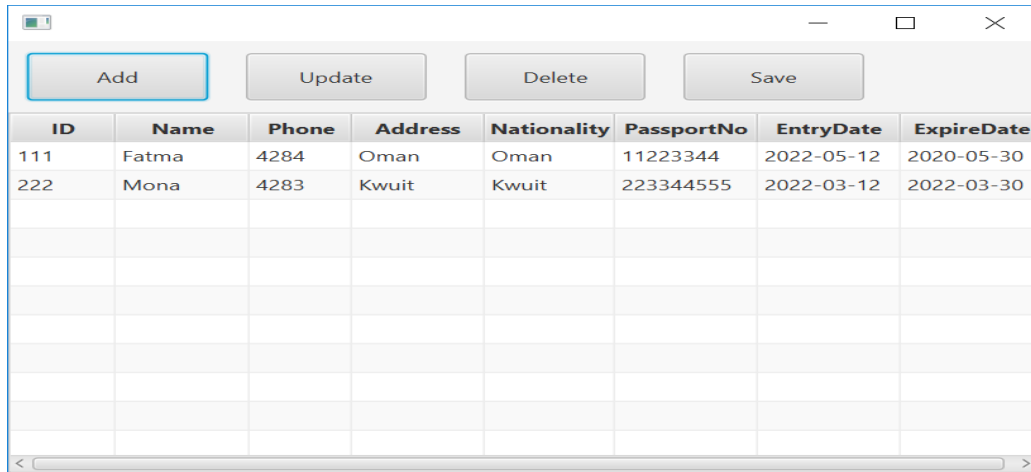
The 'Update House' window is a light gray dialog box with a title bar containing red, yellow, and green window control buttons. The title 'Update House' is centered at the top. Below the title, there are several input fields: 'House No.' (a single-line text box with the value '61'), 'Address' (a single-line text box with the value 'Barwa'), 'Availability' (radio buttons for 'Yes' and 'No', with 'Yes' selected), 'Type ID' (a single-line text box with the value '0'), 'Description' (a multi-line text area with the value 'Villa'), and 'Price' (a single-line text box with the value '6500.0'). At the bottom right, there are two buttons: 'Update' and 'Cancel'.

House No.	Address	Description	Price	Availability
60	Barwa	Villa	6000.0	true
61	Barwa	Villa	6500.0	true
62	Barwa	Villa	7000.0	true
60	Barwa	Apartment	4000.0	true
61	Barwa	Apartment	3000.0	true
62	Barwa	Apartment	3500.0	true

Figure 4

Main Tenant Window

This window is shown when the user clicks on the Tenant Button inside the Main View shown [Fig.1]

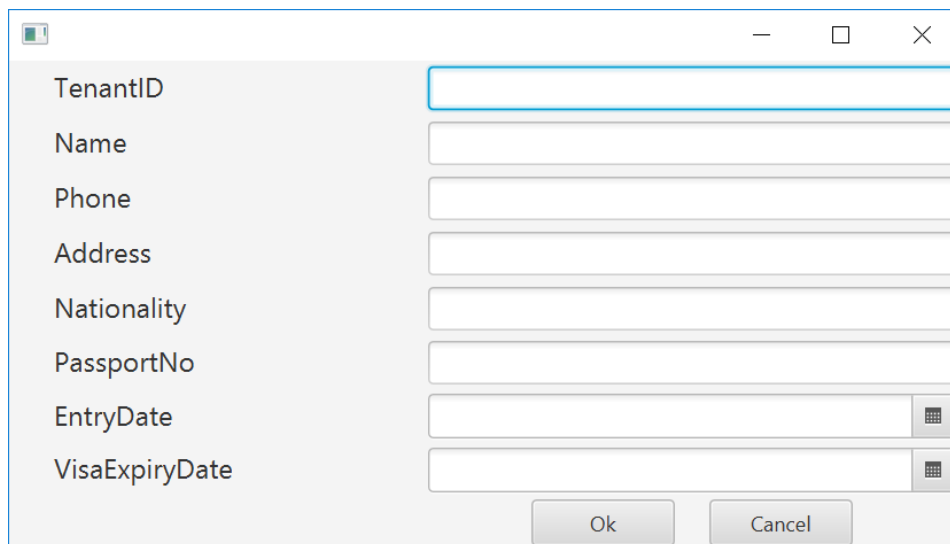
A screenshot of a software window titled 'Main Tenant Window'. At the top, there are four buttons: 'Add', 'Update', 'Delete', and 'Save'. Below the buttons is a table with 8 columns: ID, Name, Phone, Address, Nationality, PassportNo, EntryDate, and ExpireDate. The table contains two rows of data. The first row has ID 111, Name Fatma, Phone 4284, Address Oman, Nationality Oman, PassportNo 11223344, EntryDate 2022-05-12, and ExpireDate 2020-05-30. The second row has ID 222, Name Mona, Phone 4283, Address Kwuit, Nationality Kwuit, PassportNo 223344555, EntryDate 2022-03-12, and ExpireDate 2022-03-30. There are several empty rows below the data. The window has a standard Windows-style title bar with minimize, maximize, and close buttons.

ID	Name	Phone	Address	Nationality	PassportNo	EntryDate	ExpireDate
111	Fatma	4284	Oman	Oman	11223344	2022-05-12	2020-05-30
222	Mona	4283	Kwuit	Kwuit	223344555	2022-03-12	2022-03-30

Figure 5

Add Tenant Window

This window is shown when the user clicks on the Add Button inside the Tenant View shown [Fig.5]

A screenshot of a software window titled 'Add Tenant Window'. It contains a list of labels on the left and corresponding input fields on the right. The labels are: TenantID, Name, Phone, Address, Nationality, PassportNo, EntryDate, and VisaExpiryDate. The input fields are text boxes for the first six labels and date pickers for the last two. At the bottom right, there are 'Ok' and 'Cancel' buttons. The window has a standard Windows-style title bar.

TenantID

Name

Phone

Address

Nationality

PassportNo

EntryDate

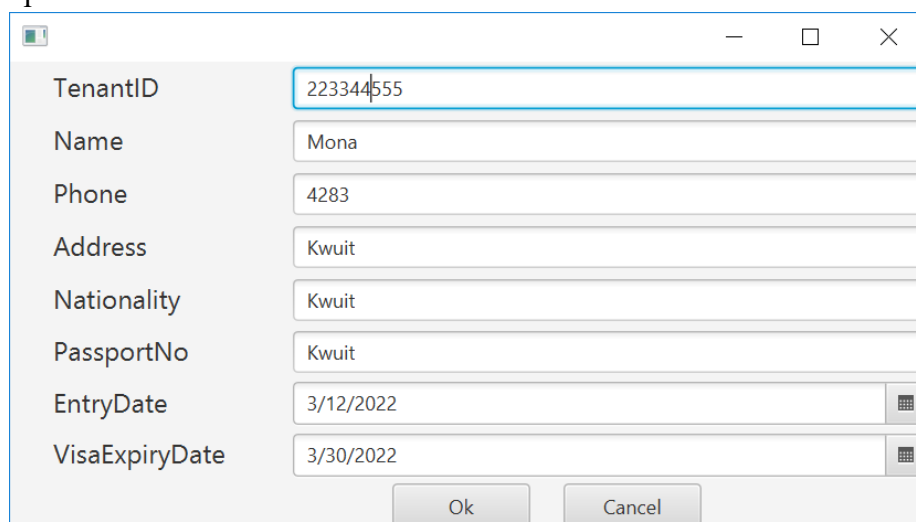
VisaExpiryDate

Ok Cancel

Figure 6

Update Tenant Window

This window is shown when the user clicks on the Update Button inside the Tenant View shown [Fig.5]. It is similar to the Update House.

A screenshot of a software window titled 'Update Tenant Window'. It contains a list of labels on the left and corresponding input fields on the right. The labels are: TenantID, Name, Phone, Address, Nationality, PassportNo, EntryDate, and VisaExpiryDate. The input fields are text boxes for the first six labels and date pickers for the last two. The TenantID field contains the value '223344555'. The Name field contains 'Mona', Phone contains '4283', Address contains 'Kwuit', Nationality contains 'Kwuit', and PassportNo contains 'Kwuit'. The EntryDate field contains '3/12/2022' and the VisaExpiryDate field contains '3/30/2022'. At the bottom right, there are 'Ok' and 'Cancel' buttons. The window has a standard Windows-style title bar.

TenantID

Name

Phone

Address

Nationality

PassportNo

EntryDate

VisaExpiryDate

Ok Cancel

Figure 7

Rentals View

This window is shown when the user clicks on the Rental Button inside the Main View shown [Fig.1]. The following are the steps you need to implement.

1. You should populate the **tenant id** and **house numbers** in the two drops down elements [Tenant, House].
2. When the user selects a tenant id, house no, rent start and end dates, and presses on Rent. Then you need to use this information to create a rental object.
Hint: before creating the rental object, use the **findHouse** and **findTenant** methods you created in phase 1, to get all the house and tenant objects .
3. Create a Rental Object using that information.
4. Display the properties of the rental object, as shown below.

Hint: the color code for the top part is “#6ec2de”. You are free to use any color.

The screenshot shows a window titled "RentalView.fxml" with a light blue header bar. Below the header, there are two rows of input fields. The first row contains a "Tenant" dropdown menu, a "Rent Start Date" text field with a calendar icon, and a "Rent" button. The second row contains a "House" dropdown menu, a "Rent End Date" text field with a calendar icon, and the same "Rent" button. Below these fields is a table with six columns: "RentalStartDate", "RentalEndDate", "Deposit:", "Invoice No", "Invoice Date", and "Total". The table body is currently empty.

Instructions for Phase 2:

1. All the instruction in phase I apply here too, as this is instructions are just a continuation of the previous phase.
2. Please start immediately and plan your time so that you finish within the project period. **Submissions after the due date will incur a 25% penalty for every day or part of a day.**
3. The name of the member who created a class must be written as the author using Javadoc comments. Also, add the date of the creation of the class.
4. Each group should submit one softcopy of their work by uploading an archive of their project folder to the course website on Blackboard under your group and also update their Phase I report by adding the phase II GUI.
5. Each member MUST submit a one-page confidential report (called the self-report) describing his/her exact contribution to the project and explaining the advantages and disadvantages of working in teams, based on your current experience with this project.

Evaluation criteria:

The breakdown of the project grade is as follows:

- 30% for the functionalities required above, and the report that shows the code and the results of all the test cases. This includes correct indentation and formatting of your code, use of a meaningful variable name, abiding by Java naming conventions etc. (Phase I and II)
- 20% for authoring and producing the Javadoc documentation (Phase I)
- 25% for adding a GUI to the application. (Phase II)
- 15% File management (saving, loading, updating) (Phase II)
- 10% for the presentation of the work (Phase II)
- **Copying and/or plagiarism (-100%)**
- **The program does not compile (-100%)**
- **If you are not attend the discussion (-100)**
- **It is the right of the instructor to use any way of testing the student in the discussion and demo session, and according to that in some cases (100%) graded project may be down to (-100%) graded project.**