UNIVERSITÉ Concordia UNIVERSITY

# SOEN 6441

**Project Part 2**

Pablo Arevalo Escobar
Kaling Swain

40081955
40226333

# UML and Design Pattern Analysis:

### JavaFX For User Interface:

We have created the user interface for our real estate application using JavaFX. JavaFX provides a rich set of UI controls and layouts that can help you create a visually appealing and user-friendly application. We used Scene Builder, a visual layout tool for designing JavaFX user interfaces, to create a custom UI that meets the specific requirements of your real estate project. We added various UI controls, such as text fields, labels, buttons, tables, and charts, to display property listings, property details, and search forms. Additionally, we used image view and list view controls to add property images and lists to your application.

### The Singleton Pattern:

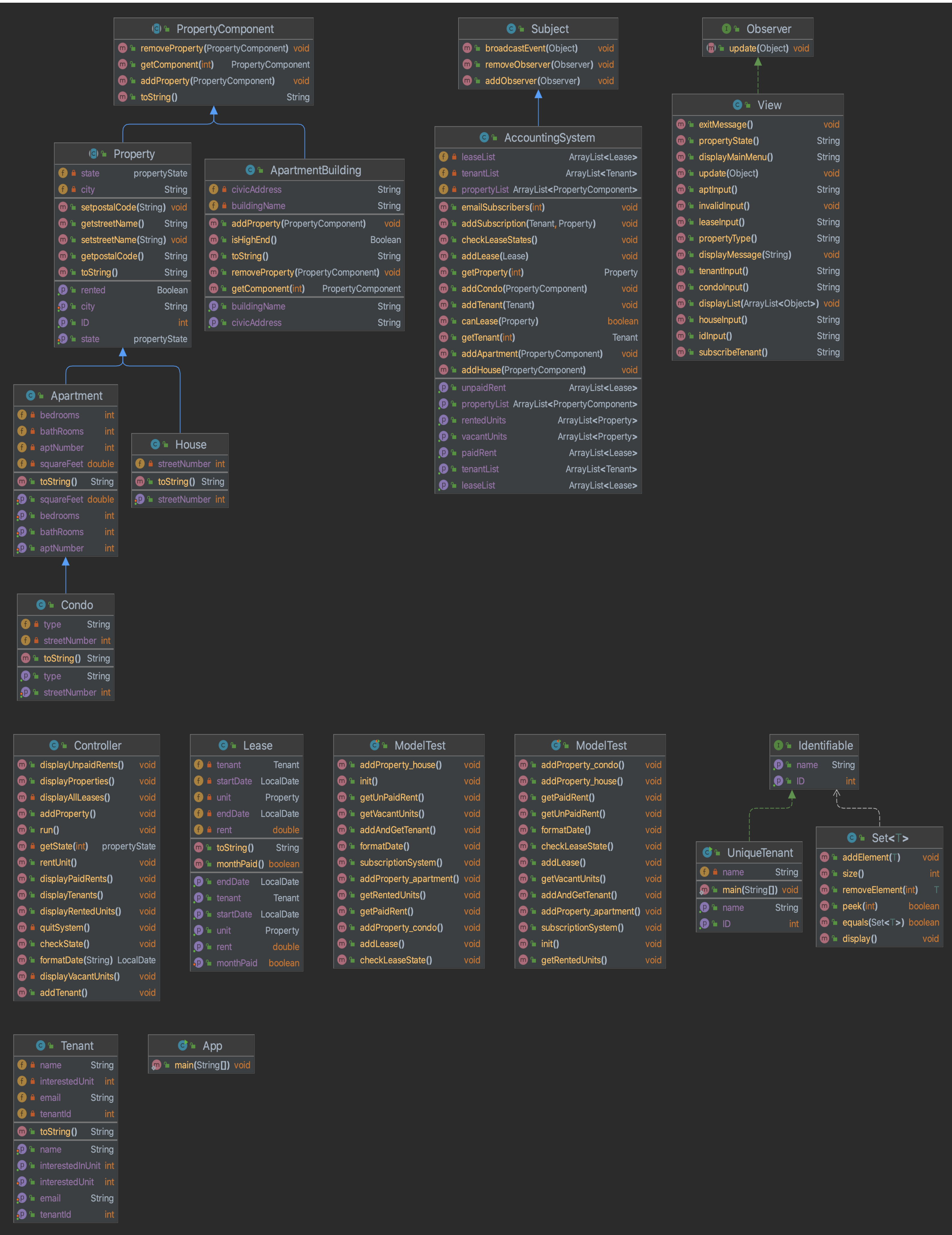
We later made certain parts of our class structure follow a singleton pattern. The singleton pattern is a design pattern that restricts the instantiation of a class to one object, effectively making it a "singleton". This means that there is only one instance of the class in the application, which is globally accessible and can be used by multiple components. Singletons are typically used in situations where it is important to ensure that there is only one instance of a class throughout the entire application, such as when there is a need to control the number of instances of a class, such as with a connection pool or a logger. It is mostly used when there is a need to ensure that all components in the application use the same instance of a class, such as with a configuration manager, because it reduces the overhead associated with creating multiple instances of a class.

### Generic Set Class:

Set class that implements the Identifiable interface and provides functionality to add, remove, check for containment, and display the elements of the set. Our Set class uses an ArrayList internally to store the elements of the set, and it enforces the uniqueness of elements by checking for duplicates before adding new elements.

We have also created a UniqueTenant class that implements the Identifiable interface and provides an ID and name for each tenant. We have used this class to demonstrate the usage of your Set class by creating two sets of UniqueTenant objects and performing operations such as adding, removing, and displaying elements, as well as checking for equality between the two sets. Overall, our Set class provides a simple and effective way to work with sets of objects that enforce uniqueness, and your UniqueTenant class demonstrates how this can be applied in practise.

# APPLICATION UML DIAGRAM AFTER JAVAFX INTEGRATION



PropertyComponent
- removeProperty(PropertyComponent) void
- getComponent(int) PropertyComponent
- addProperty(PropertyComponent) void
- toString() String

Subject
- broadcastEvent(Object) void
- removeObserver(Observer) void
- addObserver(Observer) void

Observer
- update(Object) void

Property
- state propertyState
- city String
- setpostalCode(String) void
- getstreetName() String
- setstreetName(String) void
- getpostalCode() String
- toString() String
- rented Boolean
- city String
- ID int
- state propertyState

ApartmentBuilding
- civicAddress String
- buildingName String
- addProperty(PropertyComponent) void
- isHighEnd() Boolean
- toString() String
- removeProperty(PropertyComponent) void
- getComponent(int) PropertyComponent
- buildingName String
- civicAddress String

AccountingSystem
- leaseList ArrayList<Lease>
- tenantList ArrayList<Tenant>
- propertyList ArrayList<PropertyComponent>
- emailSubscribers(int) void
- addSubscription(Tenant, Property) void
- checkLeaseStates() void
- addLease(Lease) void
- getProperty(int) Property
- addCondo(PropertyComponent) void
- addTenant(Tenant) void
- canLease(Property) boolean
- getTenant(int) Tenant
- addApartment(PropertyComponent) void
- addHouse(PropertyComponent) void
- unpaidRent ArrayList<Lease>
- propertyList ArrayList<PropertyComponent>
- rentedUnits ArrayList<Property>
- vacantUnits ArrayList<Property>
- paidRent ArrayList<Lease>
- tenantList ArrayList<Tenant>
- leaseList ArrayList<Lease>

View
- exitMessage() void
- propertyState() String
- displayMainMenu() String
- update(Object) void
- aptInput() String
- invalidInput() void
- leaseInput() String
- propertyType() String
- displayMessage(String) void
- tenantInput() String
- condoInput() String
- displayList(ArrayList<Object>) void
- houseInput() String
- idInput() String
- subscribeTenant() String

Apartment
- bedrooms int
- bathRooms int
- aptNumber int
- squareFeet double
- toString() String
- squareFeet double
- bedrooms int
- bathRooms int
- aptNumber int

House
- streetNumber int
- toString() String
- streetNumber int

Condo
- type String
- streetNumber int
- toString() String
- type String
- streetNumber int

Controller
- displayUnpaidRents() void
- displayProperties() void
- displayAllLeases() void
- addProperty() void
- run() void
- getState(int) propertyState
- rentUnit() void
- displayPaidRents() void
- displayTenants() void
- displayRentedUnits() void
- quitSystem() void
- checkState() void
- formatDate(String) LocalDate
- displayVacantUnits() void
- addTenant() void

Lease
- tenant Tenant
- startDate LocalDate
- unit Property
- endDate LocalDate
- rent double
- toString() String
- monthPaid() boolean
- endDate LocalDate
- tenant Tenant
- startDate LocalDate
- unit Property
- rent double
- monthPaid boolean

ModelTest
- addProperty_house() void
- init() void
- getUnPaidRent() void
- getVacantUnits() void
- addAndGetTenant() void
- formatDate() void
- subscriptionSystem() void
- addProperty_apartment() void
- getRentedUnits() void
- getPaidRent() void
- addProperty_condo() void
- addLease() void
- checkLeaseState() void

ModelTest
- addProperty_condo() void
- addProperty_house() void
- getPaidRent() void
- getUnPaidRent() void
- formatDate() void
- checkLeaseState() void
- addLease() void
- getVacantUnits() void
- addAndGetTenant() void
- addProperty_apartment() void
- subscriptionSystem() void
- init() void
- getRentedUnits() void

Identifiable
- name String
- ID int

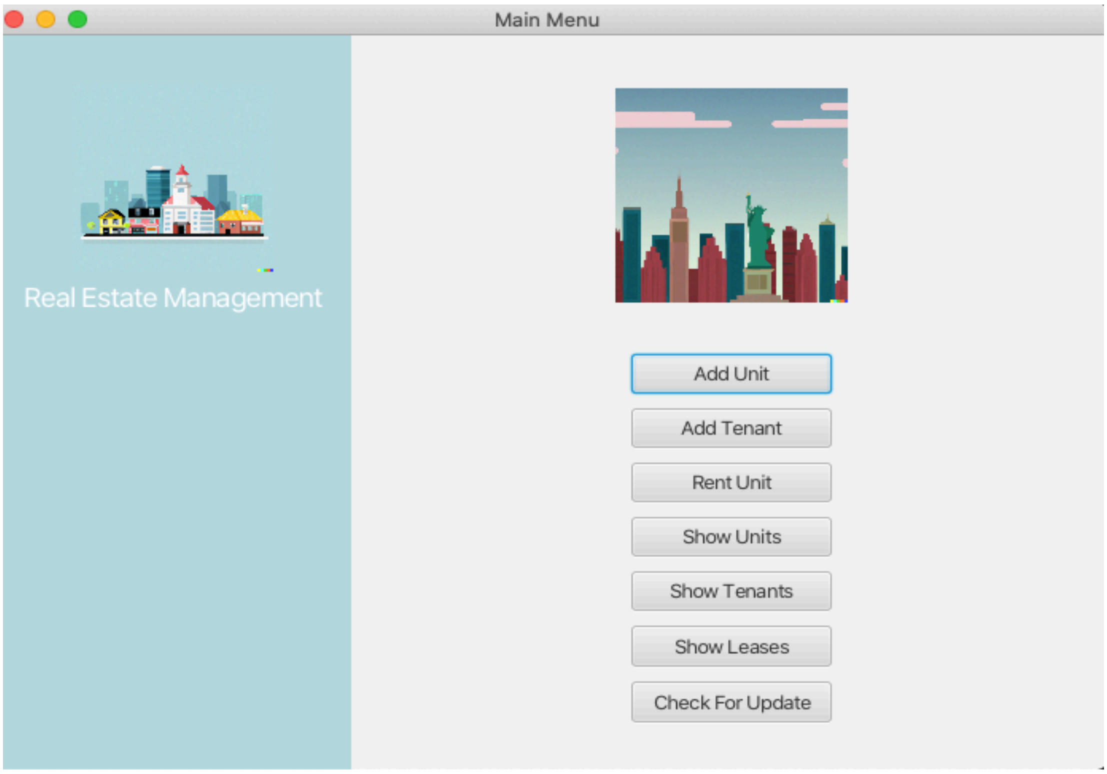
UniqueTenant
- name String
- main(String[]) void
- name String
- ID int

Set<T>
- addElement(T) void
- size() int
- removeElement(int) T
- peek(int) boolean
- equals(Set<T>) boolean
- display() void

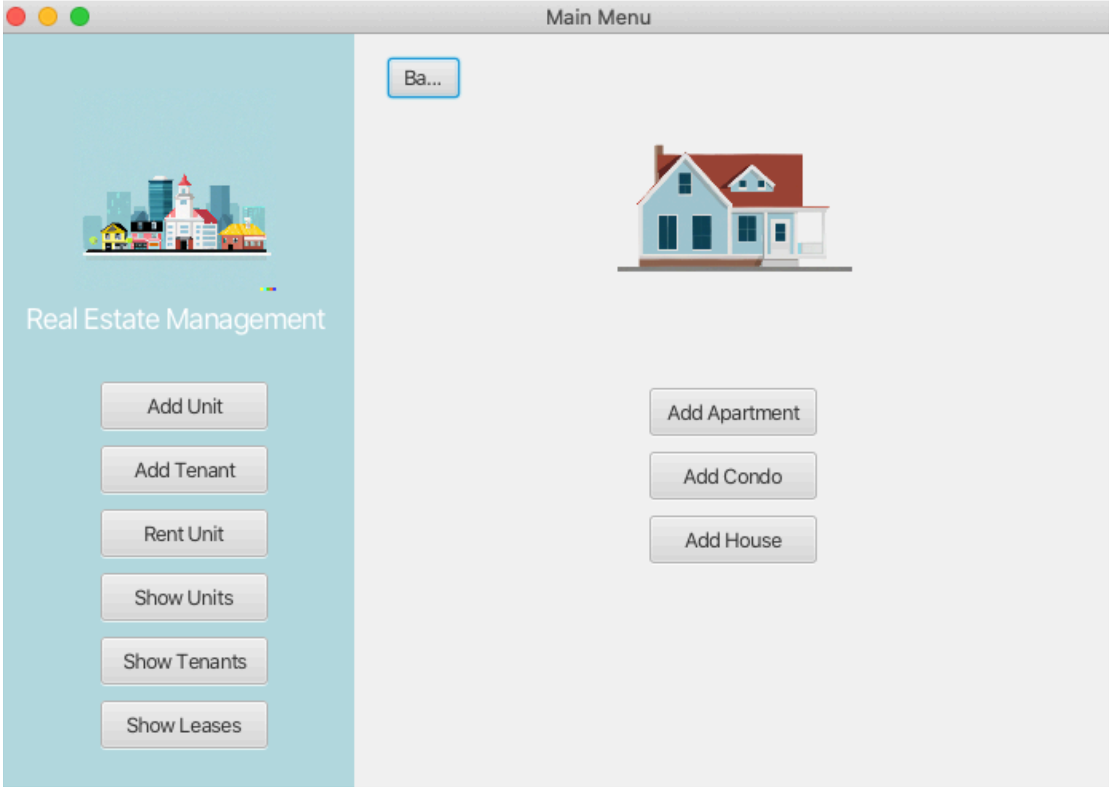
Tenant
- name String
- interestedUnit int
- email String
- tenantId int
- toString() String
- name String
- interestedInUnit int
- interestedUnit int
- email String
- tenantId int

App
- main(String[]) void

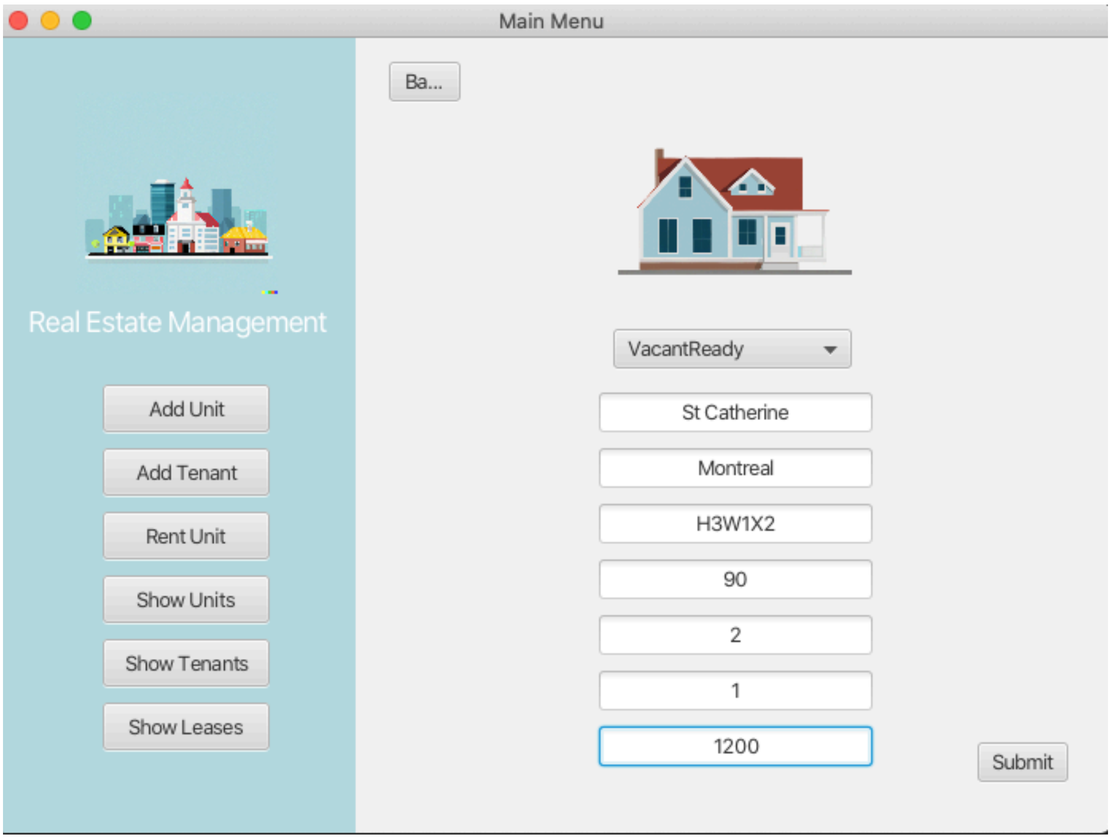
## Main window:



## Add Unit window:



## Add Apartment Window:



VacantReady

St Catherine

Montreal

H3W1X2

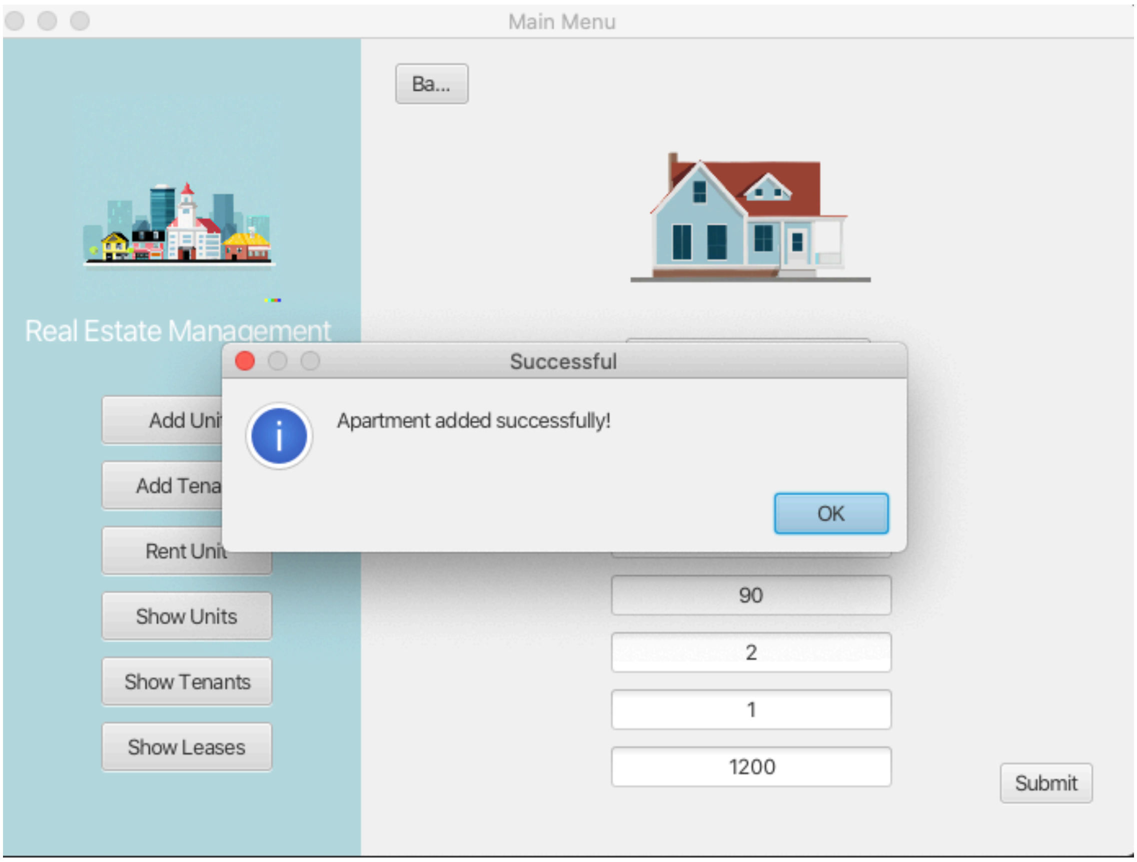
90

2

1

1200

## Successful Alert window:



Apartment added successfully!

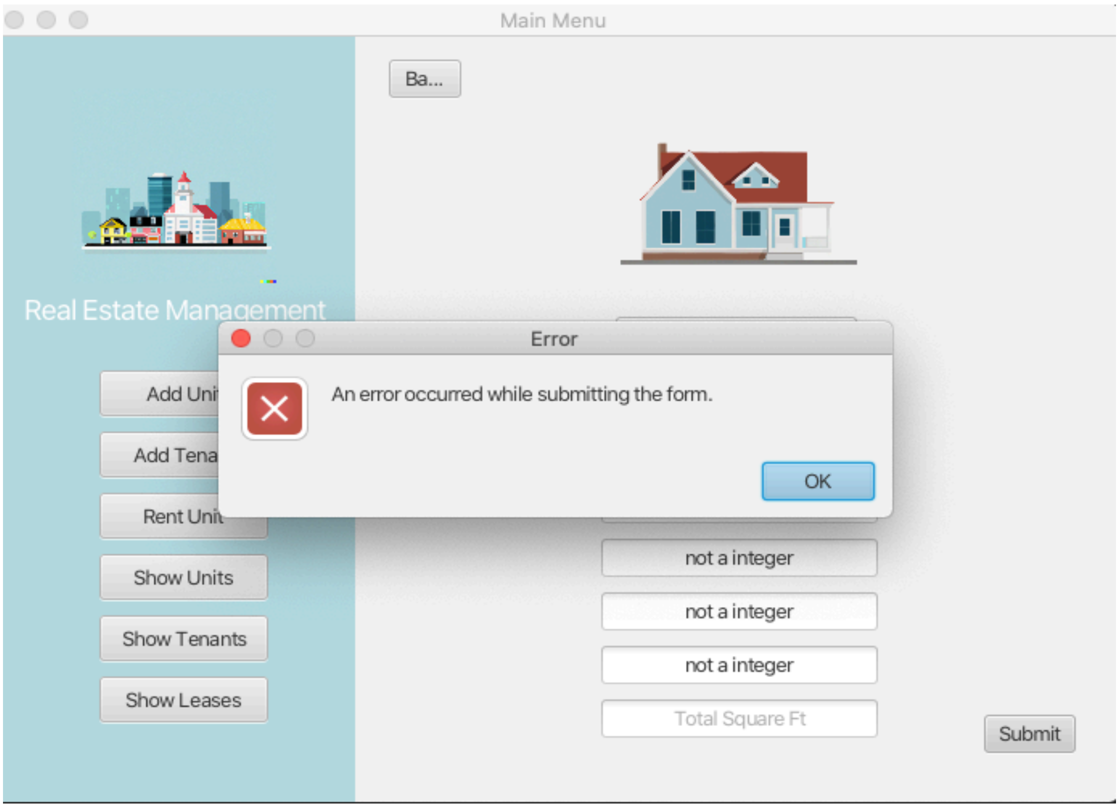
90

2

1

1200

**Fail Alert window:**



Error

An error occurred while submitting the form.

OK

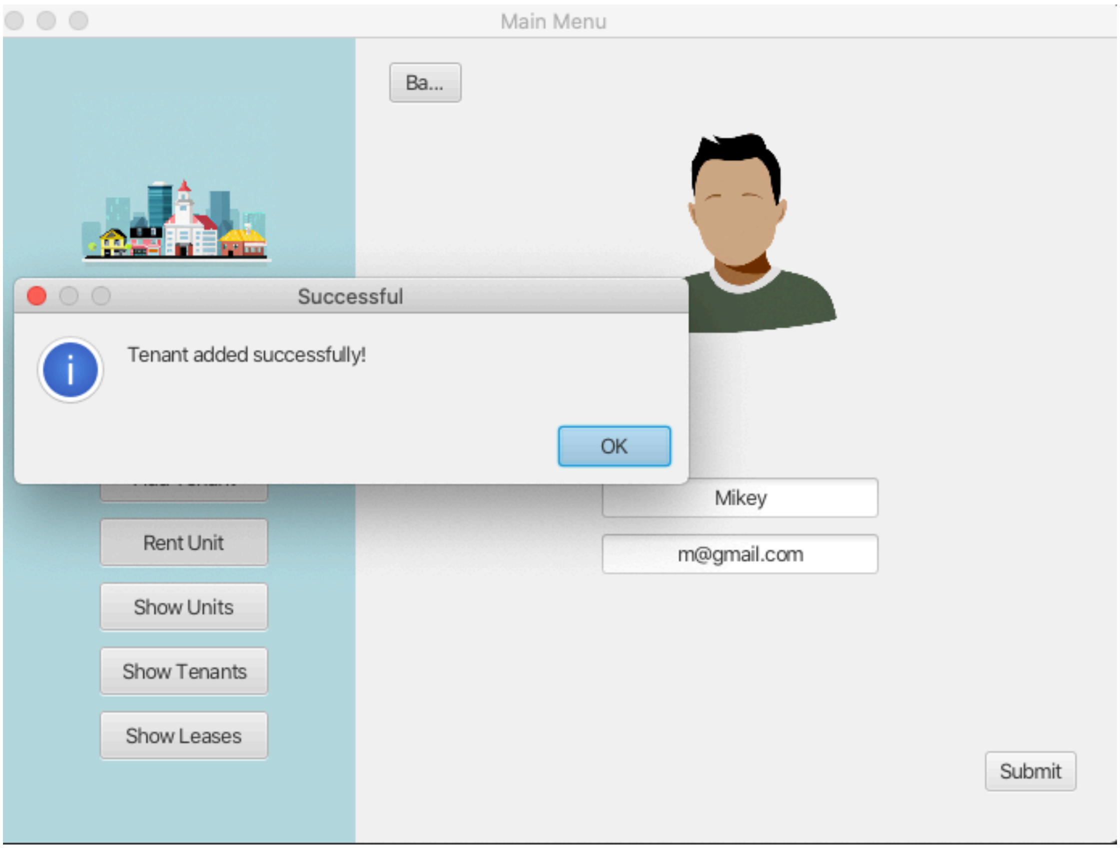
not a integer

not a integer

not a integer

Total Square Ft

**Add Tenant Window:**



Successful

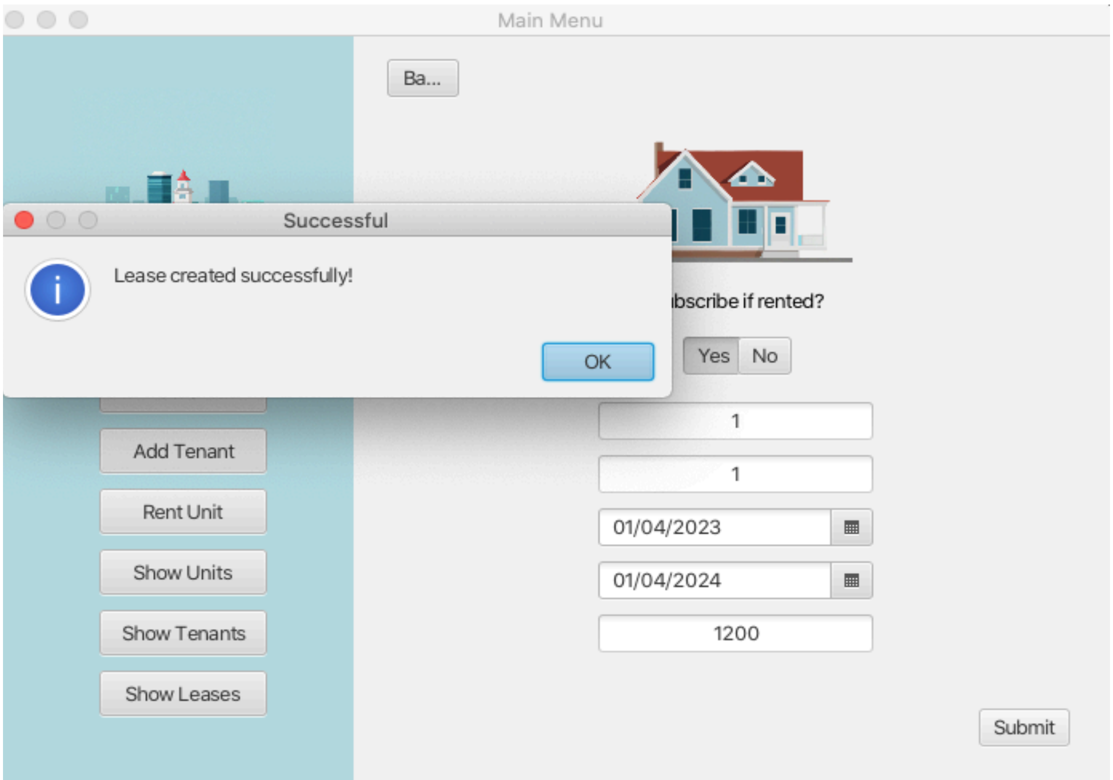
Tenant added successfully!

OK

Mikey

m@gmail.com

**Create Lease Window:**



Lease created successfully!

ıbscribe if rented?

Yes No

1

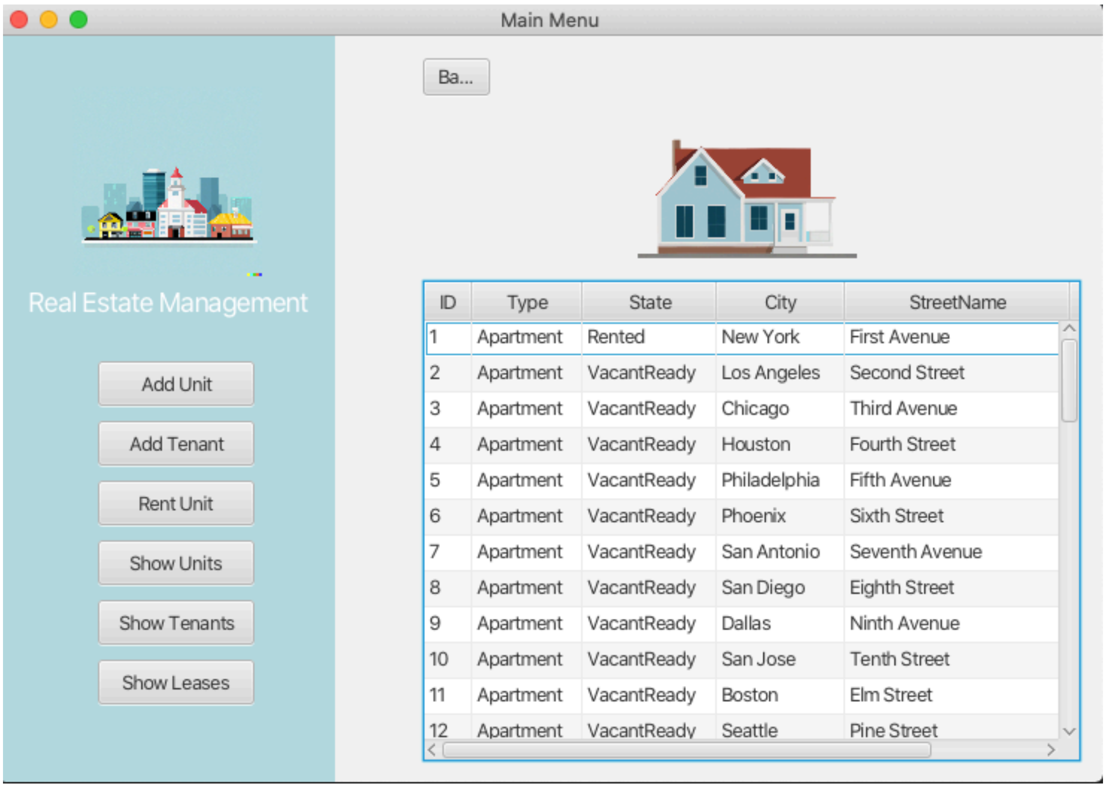
1

01/04/2023

01/04/2024

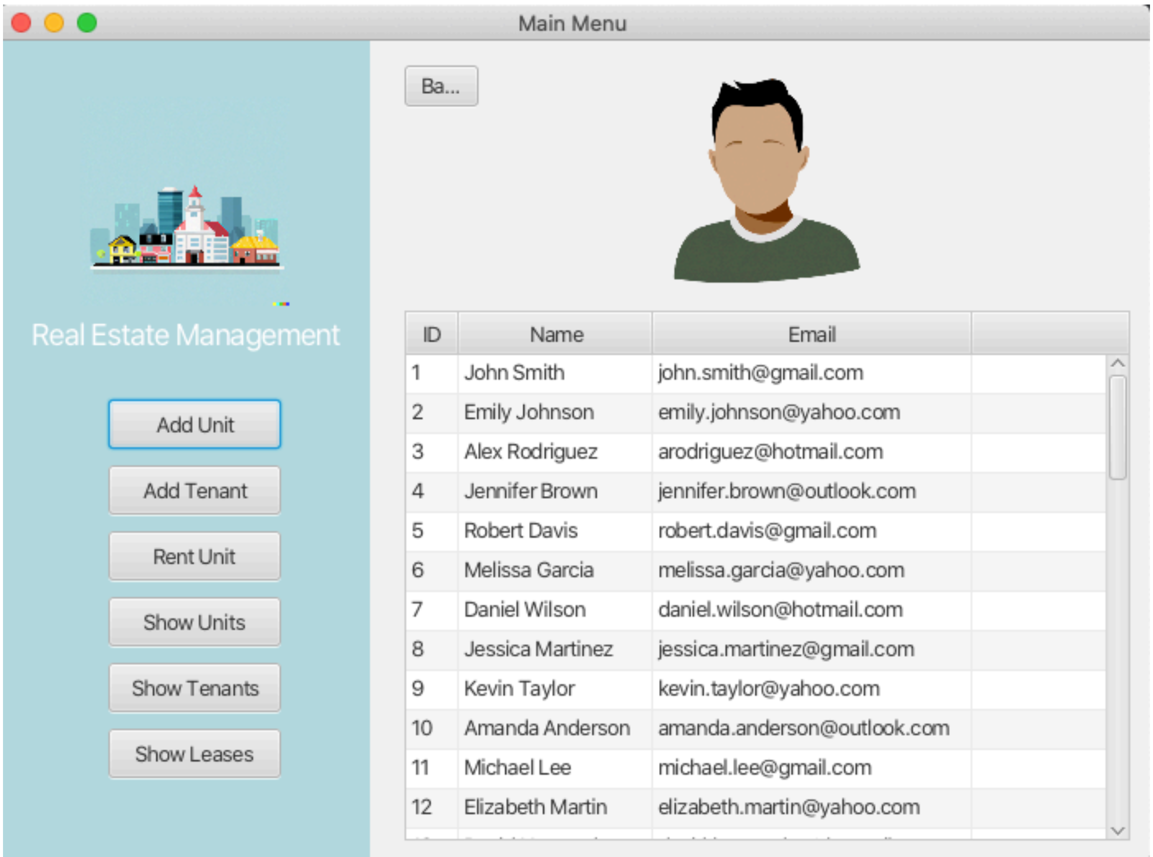
1200

**Show Units window:**



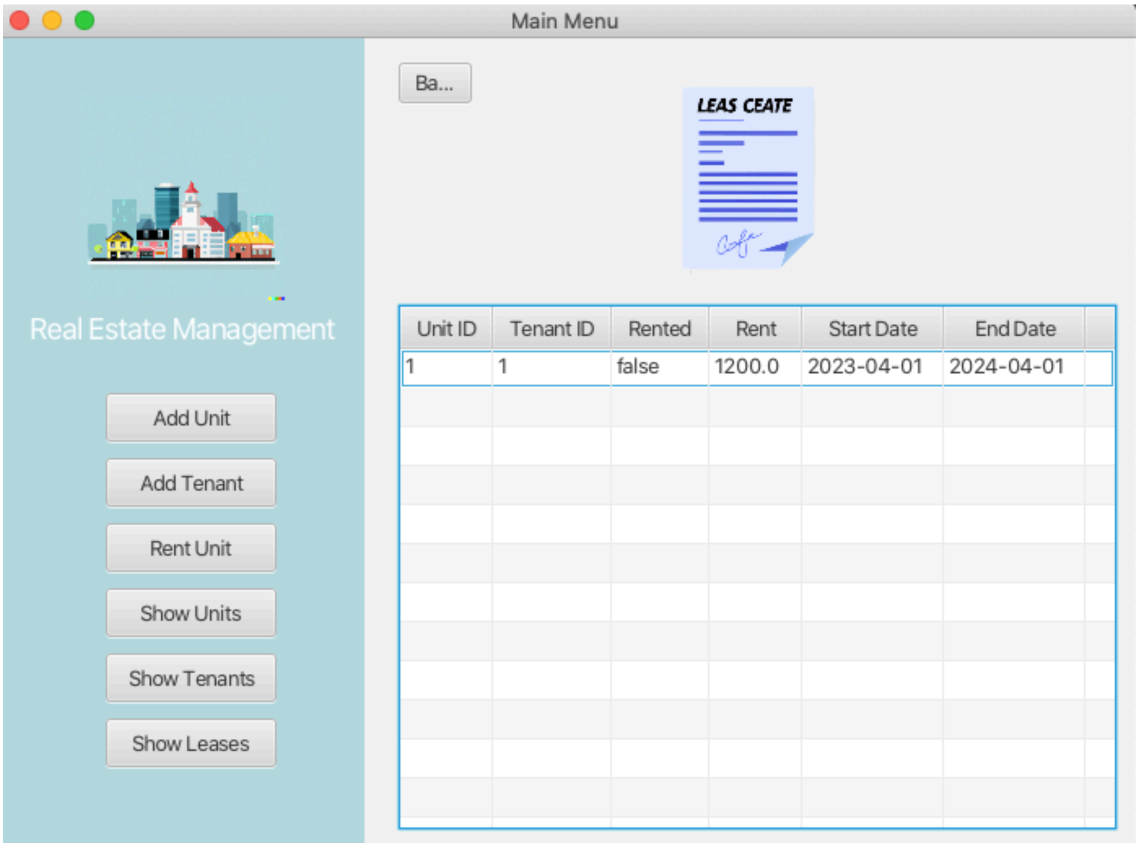
| ID | Type | State | City | StreetName |
|---|---|---|---|---|
| 1 | Apartment | Rented | New York | First Avenue |
| 2 | Apartment | VacantReady | Los Angeles | Second Street |
| 3 | Apartment | VacantReady | Chicago | Third Avenue |
| 4 | Apartment | VacantReady | Houston | Fourth Street |
| 5 | Apartment | VacantReady | Philadelphia | Fifth Avenue |
| 6 | Apartment | VacantReady | Phoenix | Sixth Street |
| 7 | Apartment | VacantReady | San Antonio | Seventh Avenue |
| 8 | Apartment | VacantReady | San Diego | Eighth Street |
| 9 | Apartment | VacantReady | Dallas | Ninth Avenue |
| 10 | Apartment | VacantReady | San Jose | Tenth Street |
| 11 | Apartment | VacantReady | Boston | Elm Street |
| 12 | Apartment | VacantReady | Seattle | Pine Street |

**Show Tenants Window:**



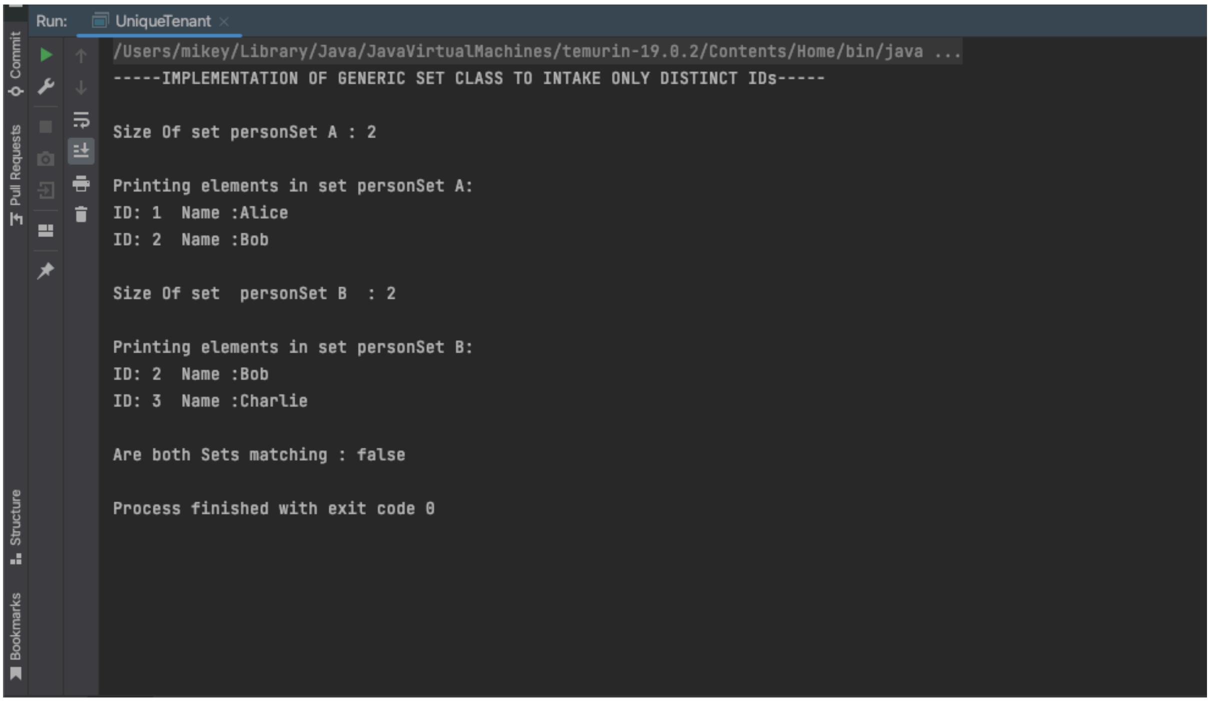
| ID | Name | Email |
|---|---|---|
| 1 | John Smith | john.smith@gmail.com |
| 2 | Emily Johnson | emily.johnson@yahoo.com |
| 3 | Alex Rodriguez | arodriguez@hotmail.com |
| 4 | Jennifer Brown | jennifer.brown@outlook.com |
| 5 | Robert Davis | robert.davis@gmail.com |
| 6 | Melissa Garcia | melissa.garcia@yahoo.com |
| 7 | Daniel Wilson | daniel.wilson@hotmail.com |
| 8 | Jessica Martinez | jessica.martinez@gmail.com |
| 9 | Kevin Taylor | kevin.taylor@yahoo.com |
| 10 | Amanda Anderson | amanda.anderson@outlook.com |
| 11 | Michael Lee | michael.lee@gmail.com |
| 12 | Elizabeth Martin | elizabeth.martin@yahoo.com |

**Show Lease window:**



| Unit ID | Tenant ID | Rented | Rent | Start Date | End Date |
|---|---|---|---|---|---|
| 1 | 1 | false | 1200.0 | 2023-04-01 | 2024-04-01 |

## GENERIC SET CLASS FUNCTIONALITY OUTPUT:



```
/Users/mikey/Library/Java/JavaVirtualMachines/temurin-19.0.2/Contents/Home/bin/java ...
-----IMPLEMENTATION OF GENERIC SET CLASS TO INTAKE ONLY DISTINCT IDs-----

Size Of set personSet A : 2

Printing elements in set personSet A:
ID: 1  Name :Alice
ID: 2  Name :Bob

Size Of set  personSet B  : 2

Printing elements in set personSet B:
ID: 2  Name :Bob
ID: 3  Name :Charlie

Are both Sets matching : false

Process finished with exit code 0
```