APARTMENT RENTAL SYSTEM

DOCUMENTATION

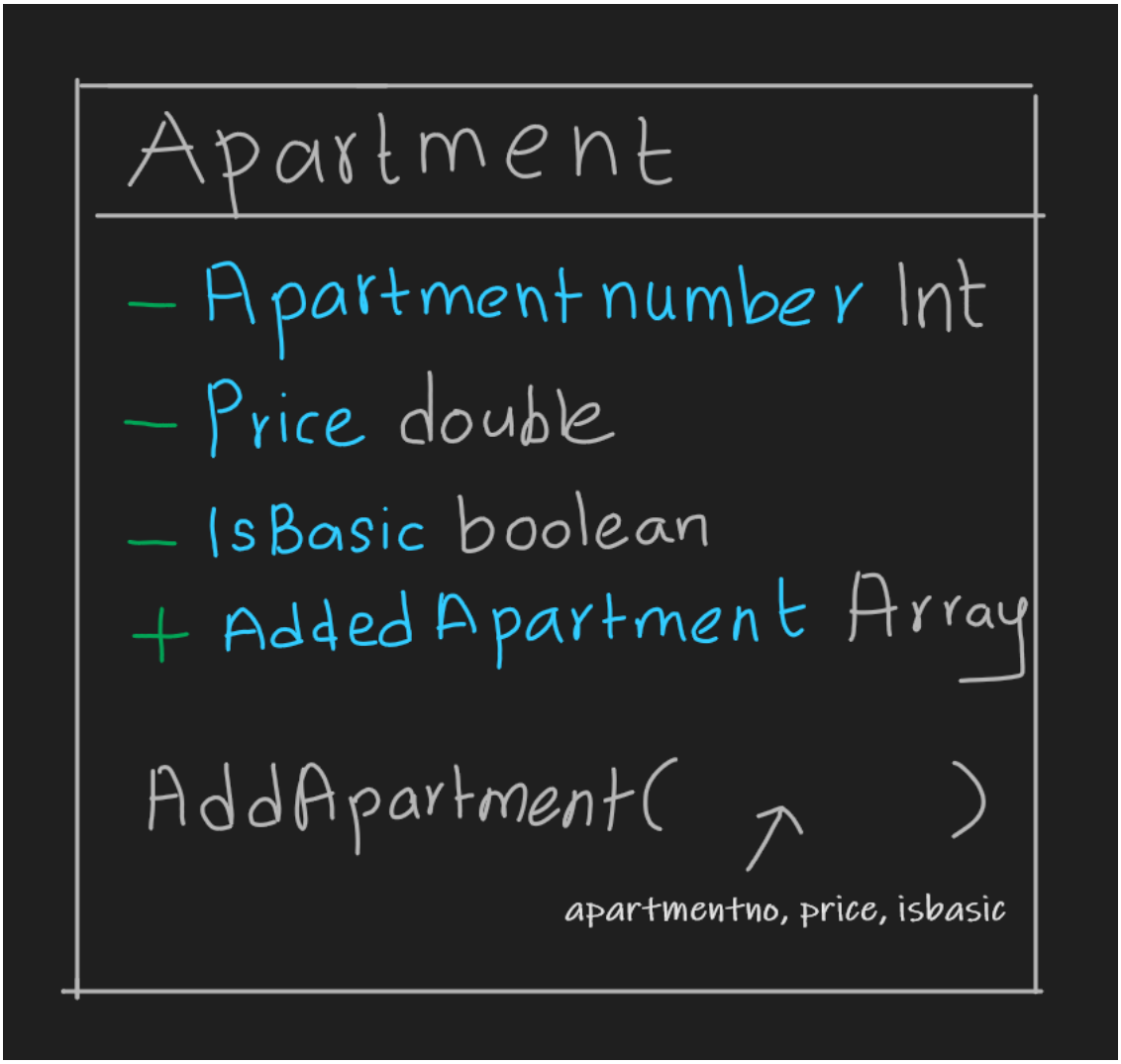
Yoosuf Yazak Hafiz | 076549

CPT205

Object Oriented Programming assignment

Contents

Class Diagram



(AddApartment method is a void method)

How it works

```
-------------------------
Apartment Renting System
-------------------------

Add an Apartment [1]

Delete an Apartment [2]

Apartment Details [3]
```

First there is a simple menu system for the user to interact with when you run the program, this will give you a choice of 3 to choose from

```
-------------------------
Apartment Renting System
-------------------------

Add an Apartment [1]

Delete an Apartment [2]

Apartment Details [3]


2

There are 0 apartments that can be deleted

There are no apartments for you to delete!
```

```
Add an Apartment [1]

Delete an Apartment [2]

Apartment Details [3]

3

There are 0 apartments that can be Viewed

There are no apartments for you to view!
```

When you first run the program, you won't be to use the choice 2 and 3, since there won't be any existing data to delete/view. The reason its showing two output messages are because, if there are apartments or any data, it will show the list of apartments that you can (interact) with to delete/view.

```
Enter Apartment Number:
1

Enter Apartment Price:
2500

Enter Apartment Type:
Basic [1]

Premium [2]

1

SUCCESSFULLY ADDED AN APARTMENT!


SUCCESSFULLY STORED THE ADDED APARTMENT!
```

When you choose the add an apartment choice, it will ask you for the apartment number and apartment price and the type of it.

If you add without any error or mistakes then it will “Successfully add” the apartment, I’ll explain why I’ve used two outputs here in the Code section.

```
Enter Apartment Number:
yes

PLEASE ENTER A VALID CHOICE! (enter your choice from the [])
```

If there was an error adding an apartment it will throw an error and repeat the questions

```
Add an Apartment [1]

Delete an Apartment [2]

Apartment Details [3]

3

There are 1 apartments that can be Viewed

Which apartment do you want to view?

Apartment: 1
```

Now you'd be able to use delete/view function, as you can see, we've already added one apartment from before, this view functionality gives you a better detail about the apartment you've added.

```
Which apartment do you want to view?

Apartment: 1

1
Apartment number: 1
Apartment price: 2500.0
Apartment type: Basic
```

As you can see it throws all the data you've input before while making the apartment.

```
There are 1 apartments that can be deleted

Which apartment do you want to delete?

Apartment: 1

1

SUCCESSFULLY REMOVED THE APARTMENT!
```

The same goes for delete functions as well. If there was an error it will throw an error statement, probably with how to fix it.

And with 3 functionalities that sums up my apartment renting system.

Code

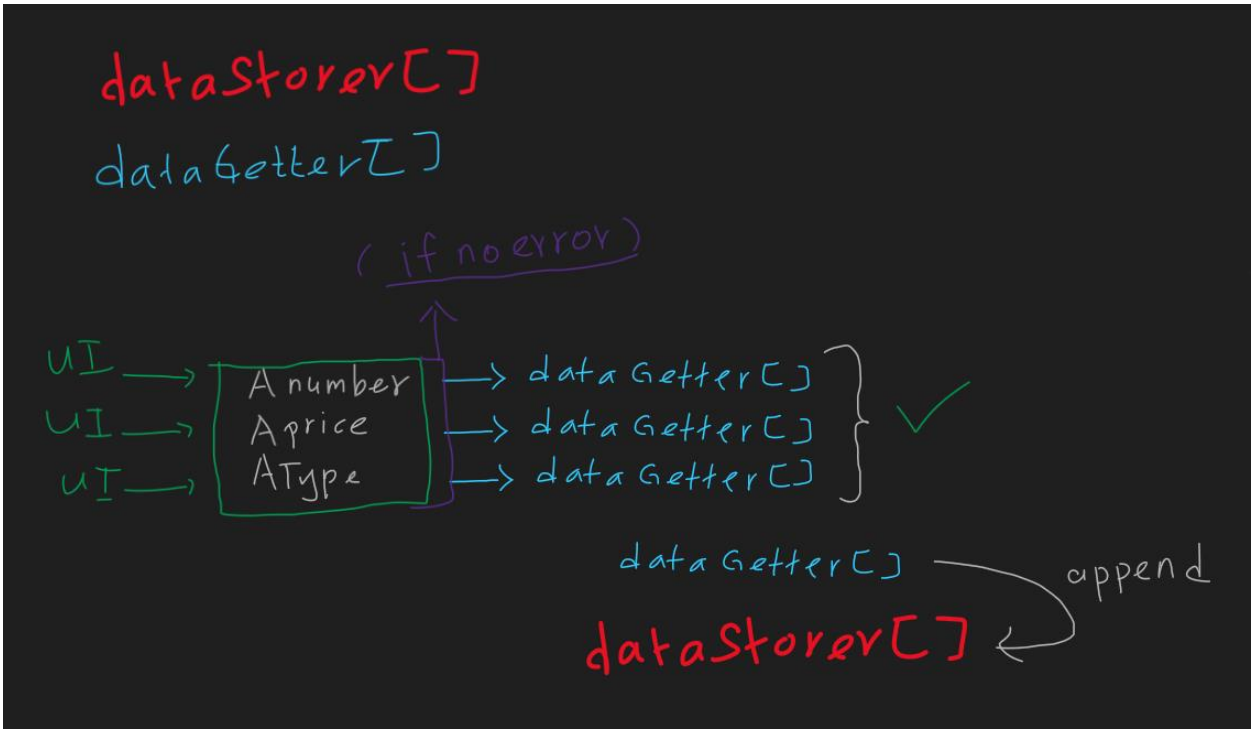
```
public class Apartment {
    protected int ApartmentNumber;
    protected double Price;
    protected boolean IsBasic;
    public List<String> AddedApartment = new ArrayList<String>();

    public void AddApartment(String apartmentNO, String price, String isBasic){

        AddedApartment.add(apartmentNO);
        AddedApartment.add(price);
        AddedApartment.add(isBasic);
    }
```

I have created the blueprint for Apartment with protected variables of apartment number and its price and a Boolean which is apartment type, its Boolean since it will either basic or premium, so if isBasic is false then it's a premium apartment and so on.

```
public class MainClass {
    public static void main(String args[]){

        Apartment manager = new Apartment();

        int ApartmentCounter = 0;

        ArrayList<String> datacheat = new ArrayList<String>();

        List DataStorer = new ArrayList();
```

These are the main function variables, an apartment counter which keeps on getting incremented whenever you have successfully added an apartment, and I have declared a List as a new object or ArrayList(). Manager is the object from the apartment class I showed you before.

The reason I used 2 arrays are because when the user input first, I want all the input data in one place, which can be arranged if I use an array to store and keep it as it is. So, what I've done to tackle this was using one Array for moving the data to another array as an array, so this data can be stored in the permanent data holder array as one since it's a list itself. When the data holder array have been appended then the data getter array will be cleared for the next data transfer, that's why there are two outputs after adding an apartment (Here is an illustration)



As you have noticed before that I take in data as strings from the array method I've declared in the apartment class before, so I need to convert all the data all the data was added successfully with no error, until then it will keep on looping with hints to get the right input from the user.

```
String ApartmentNumber = Integer.toString(manager.ApartmentNumber);
String ApartmentPrice = Double.toString(manager.Price);

//adding data to the dataGetter array
manager.AddApartment(ApartmentNumber, ApartmentPrice, "Basic");
```

The reason why I have hardcoded "Basic" in the array method is because I'm calling this in the if statement when its true if the user have chosen basic as their apartment, it will be the exact same with premium as well, after adding into the array in the object manager method, this data then

would be stored in the data holder as an array. (Here is a code with the if statement of apartment type)

```
Scanner choiceGetter2 = new Scanner(System.in);
int choice2 = choiceGetter2.nextInt();
if(choice2 == 1){
    manager.IsBasic = true;

    String ApartmentNumber = Integer.toString(manager.ApartmentNumber);
    String ApartmentPrice = Double.toString(manager.Price);

    //adding data to the dataGetter array
    manager.AddApartment(ApartmentNumber, ApartmentPrice, "Basic");

    System.out.println("\nSUCCESSFULLY ADDED AN APARTMENT!\n");


    //adding data to the storeArray

    DataStorer.add(manager.AddedApartment);
    manager.AddedApartment.clear();
    System.out.println("\nSUCCESSFULLY STORED THE ADDED APARTMENT!\n\n");


    ApartmentCounter++;

}
else if(choice2 == 2){
```

Else if choice is 2 then it's the exact same code but it's a premium apartment.

```
System.out.println("\nThere are "+ApartmentCounter+" apartments that can be deleted\n");
if(ApartmentCounter!=0){
    System.out.println("Which apartment do you want to delete?\n");
        for(int i = 1;i<=ApartmentCounter;i++){
            System.out.println("Apartment: "+i+"\n");
        }
```

The code above is validation to check if there are apartments to delete or not if there are then it will iterate through the range of 1 and how many apartments have been added, and that's how you form the added apartment list

```
Scanner DeleteChoiceGetter = new Scanner(System.in);

int DeleteChoice = DeleteChoiceGetter.nextInt();

int remover = DeleteChoice -1;

DataStorer.remove(remover);

for(int i =0; i<2; i++){
    datacheat.remove(remover);
}
```

To make this user friendly, I made the program to start from 1, due to this reason I have to -1 the input of the user and store it as a variable and use the .remove() function by passing the created variable to remove the added apartment, this would be very easy since elements are now arrays itself (the code above is how I use it)

The exact same concept for viewing apartment details except instead of deleting you iterate through it to get specific elements and output the value as a string.

And that pretty much sums up the entire code.

// end of documentation