Team 2: Fedorov Konstantin | Sharipov Amirlan | Abdirashov Rakhman | Grigorenko Nikita

SSAD Assignment 4

Hotel Management Android Project
Implemented Design Pattern: Observer

The task of our project is to implement an application for a hotel manager, based on building a UML diagram and writing the code. The application should be able to, for example, book rooms, ask to clean, and so on...

Part 1: Explanation of the selected design and its usage

Why do we use this pattern?
The template of this pattern is ideal for our project, because the class Room manages the server, and sends it notifications that somewhere in the hotel it is necessary to perform some action. In turn, the observer sends the necessary notifications to one of the four expanding observers.

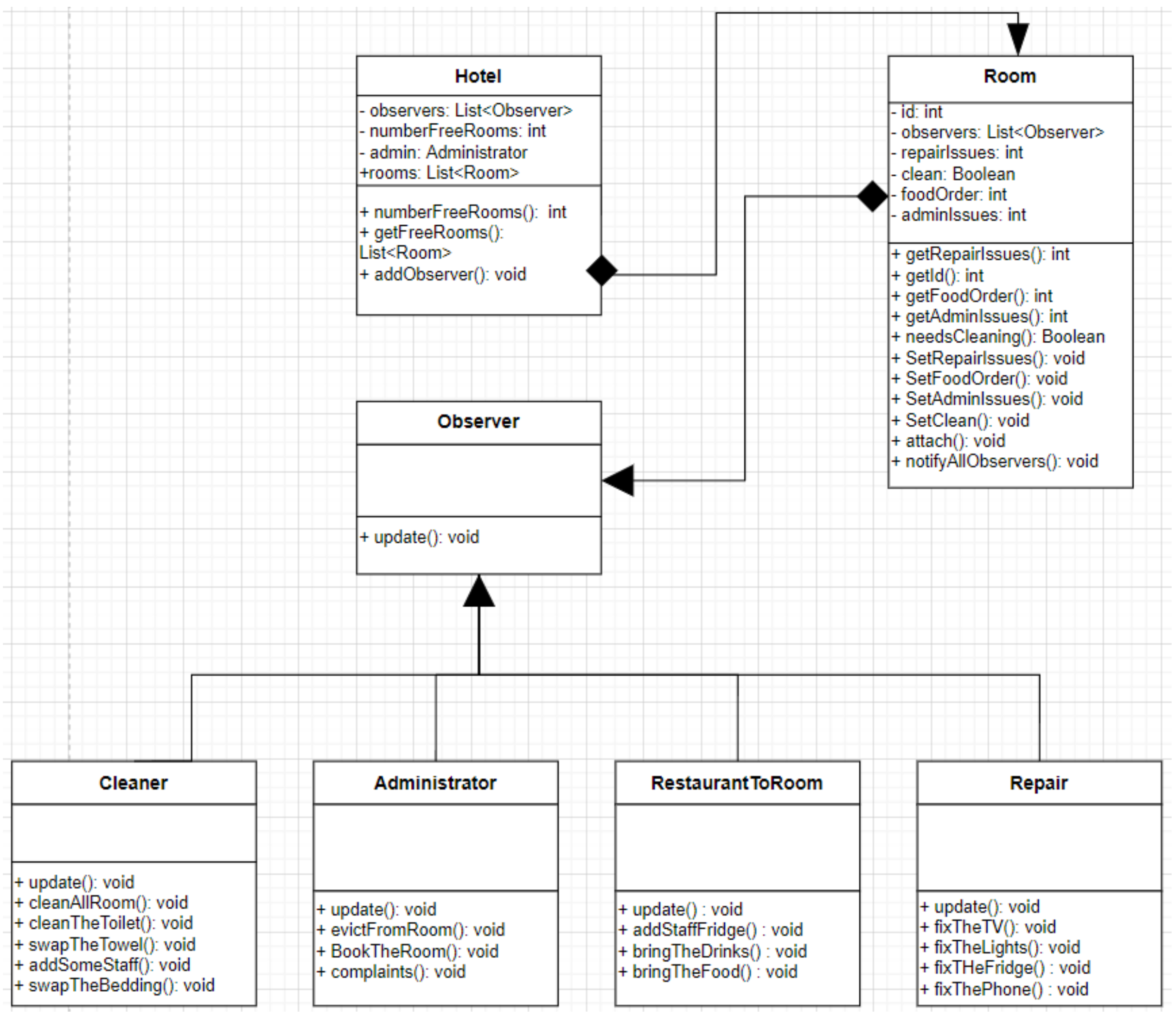
The implementation in our code/UML of the pattern.
The class Room is responsible for sending notifications from a certain room that some action needs to be performed.
An abstract class Observer is needed in order to update information about notifications and pass it to subservers that extend it.
Other observers like: Cleaner, Administrator, RestaurantToRoom, Repair perform actions on the specified notifications.

The program code is completely based on the diagram made and completely does what is described on it.

The Observer pattern template is used as the basis of our hotel model. All the code is written with the support of this pattern.

Team 2: Fedorov Konstantin | Sharipov Amirlan | Abdirashov Rakhman | Grigorenko Nikita

Part 2: Explanation of the UML/Code



Java class Hotel contains the following attributes:

- ☐ rooms: list of all rooms in hotel
- ☐ observers: list of all observers
- ☐ numberFreeRooms: integer number of free rooms in hotel
- ☐ admin: Administrator class of Administrator

Methods:

- ☐ numberFreeRooms(): return of free rooms
- ☐ getFreeRooms(): return list of free rooms in hotel
- ☐ addObserver(): add another observer to the hotel

Java class Room contains the following attributes:

- ☐ id: id of the room
- ☐ observers: list of all observers of the hotel

- ☐ repairIssues: code of repair zone(which element in the room is needed to repair)
- ☐ clean: if the room is dirty - true, otherwise - false.
- ☐ foodOrder: code of the service in food(which food or drink in the room is needed to be delivered)
- ☐ adminIssues: code of the admin work(which work admin should to do)

Methods:

1. Getters:
   - ☐ getRepairIssues(): get info about private repairIssues
   - ☐ getId(): get Id from private
   - ☐ getFoodOrder(): get code of the service in food from private field
   - ☐ getAdminIssues(): get code of the admin work from private field
   - ☐ needsCleaning(): get info about clean or not
2. Setters:
   - ☐ SetRepairIssues(): set info about private repairIssues
   - ☐ SetFoodOrder(): set code of the service in food from private field
   - ☐ SetAdminIssues(): set code of the admin work from private field
   - ☐ SetClean(): set info about clean or not
3. Other:
   - ☐ attach(): attaching to the observer room
   - ☐ notifyAllObservers(): to notify the observers of a change in the rooms

Abstract class Observer contains only one method:

- ☐ update(): needs to update information about the room and call observers.

Java class Cleaner contains the following methods:

- ☐ update(): update information about the work done
- ☐ cleanAllRoom(): clean up and clean the whole room
- ☐ cleanTheToilet(): clean up only toilet

- ☐ swapTheTowel(): change towels in the room
- ☐ addSomeStaff(): restore the amount of soap, paper, etc.
- ☐ swapTheBedding(): change the bed linen on the bed

Java class Administrator contains the following methods:

- ☐ update(): update information about the work done
- ☐ evictFromRoom(): eviction from the room (the room is free)
- ☐ BookTheRoom(): book a room
- ☐ complaints(): fix complaints

Java class RestaurantToRoom contains the following methods:

- ☐ update(): update information about the work done
- ☐ addStaffFridge(): add some products into the fridge
- ☐ bringTheDrinks(): add some drinks into the room
- ☐ bringTheFood(): add some food into the room

Java class Repair contains the following methods:

- ☐ update(): update information about the work done
- ☐ fixTheTV(): repair a TV in the room
- ☐ fixTheLights(): repair a Lights in the room
- ☐ fixTheFridge(): repair a Fridge in the room
- ☐ fixThePhone(): repair a Phone in the room

The UML diagram is constructed so that all classes are interconnected and competently build the functionality of the working application. Also, each class is sufficiently implemented and has at least three methods available (except the observer). The UML diagram, along with the code and report, fully show and make it clear how the application works.

- ❖ Link To The UML: https://drive.google.com/file/d/186a7AJLKrO9UMT8n4VVNYGpj6oyIzvdb/view

<u>And here came the end of both parts of the report. Thanks for reading!</u>