**Raditya Surya Pratama**

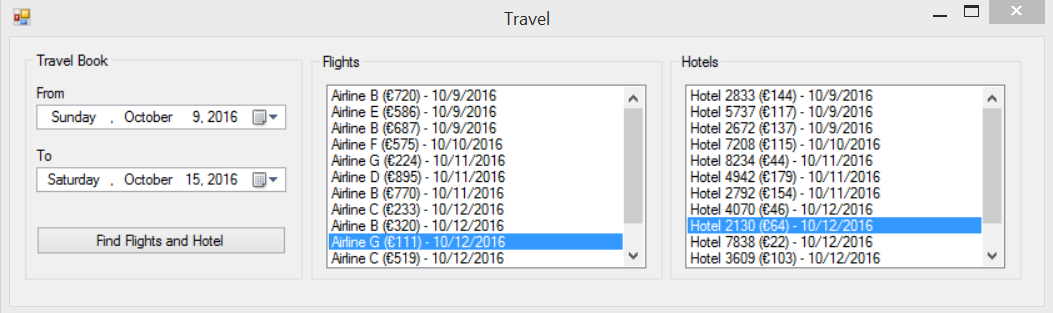
**Radu Alexandru Stoica**

Travel booking

Facade Pattern

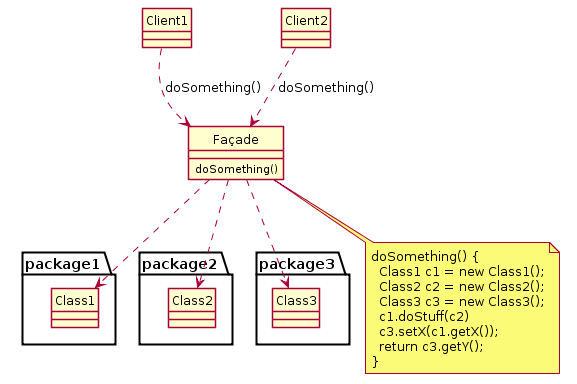
INTRODUCTION

The sixth assignment for DPR is to develop the second chosen design pattern, and we have chosen Facade Design Pattern. We made a simple travel flight & hotel search that allows the user to select a range of dates and then to display flights and hotels available within this date range. The TravelFacade class encapsulates the logic for the Client. Therefore, the Client can use the TravelFacade class in order to use the available subsystems.



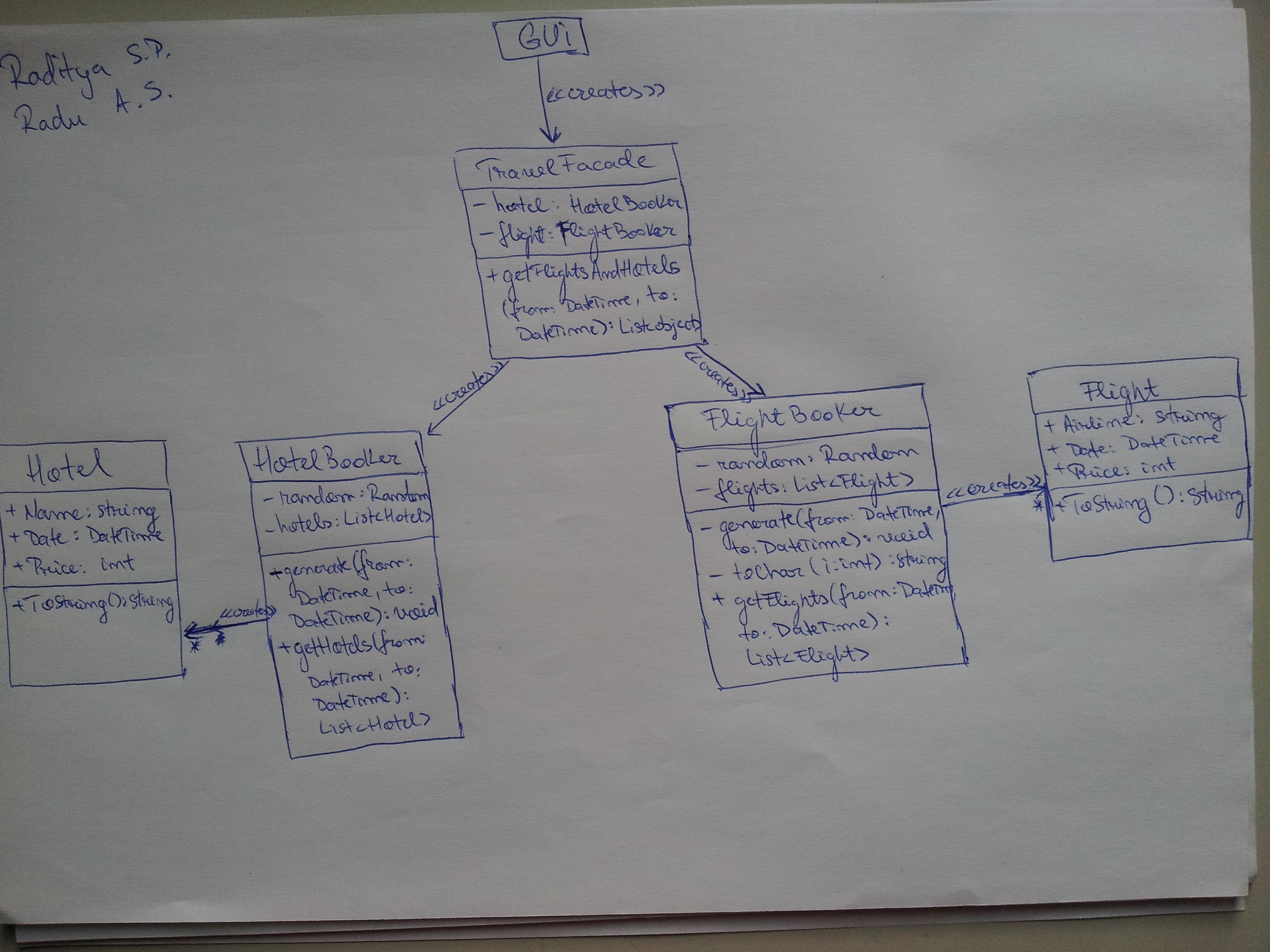
FACADE PATTERN

The Facade pattern encapsulates the complex system, allowing the user to control it using only a single interface object. In this way, the pattern promotes the decoupling the subsystems from the client. Also, the facade interface being the only access point for the client, it will also limit the access to the features of the subsystems as needed. The facade should act as a facilitator and not as a “god” object. Overall, the main goal of this pattern is to make it easier for the client to access different subsystems.



Source: <https://en.wikipedia.org/wiki/Facade_pattern>

UML DIAGRAM



REUSABILITY

The reusability of our code is represented by the TravelFacade class, which can be used by any client without the need to create objects for each subsystem.

EXTENSIBILITY

The extensibility of our code is represented by the implementation of the TravelFacade class. If a new subsystem must be added, the TravelFacade can add that new subsystem and the Client can use it by still using the same TravelFacade object as before.

MAINTAINABILITY

For the maintainability matters in the application, the classes should be easily changed or updated. In this case, the maintainability is supported by the fact that the only class used by the Client is the facade object. Therefore, if something changes, the new or updated object will still be a subsystem of the facade object. Thus, the Client doesn’t need to create that object only the facade does. Also, the subsystems are not related to each other which makes the maintainability easier. Therefore, changes are having a lower impact over the whole implementation.

PATTERN DOWNSIDE

There are a couple of disadvantages of using the Facade pattern:

* It does not prevent sophisticated clients from accessing the underlying classes.
* Note that Façade does not add any functionality, it just simplifies interfaces.

Source: http://arun-ts.blogspot.nl/2010/11/facade-design-pattern-structural.html

UNIT TEST

The unit test is included in the project solution. The unit test covers the type of the context object that created is implementing the IState interface, and when the state is change the Context object should change into StateDebosit or StateWithdraw object type.