OOPD Project 2, Final Submission

FLIGHT RESERVATION SYSTEM using OOPS Principles

Name: INDRANIL MAJUMDAR , Roll: MT20129
Name: SHIVAM SHARMA , Roll: MT20121

CONTENTS:--

1) Problem Description

2) Solution

- A) Identification of Initial OO Design
- B) OO Design resulting from SRP, OCP and LSP
- C) Implementation
- D) Demo and Viva**

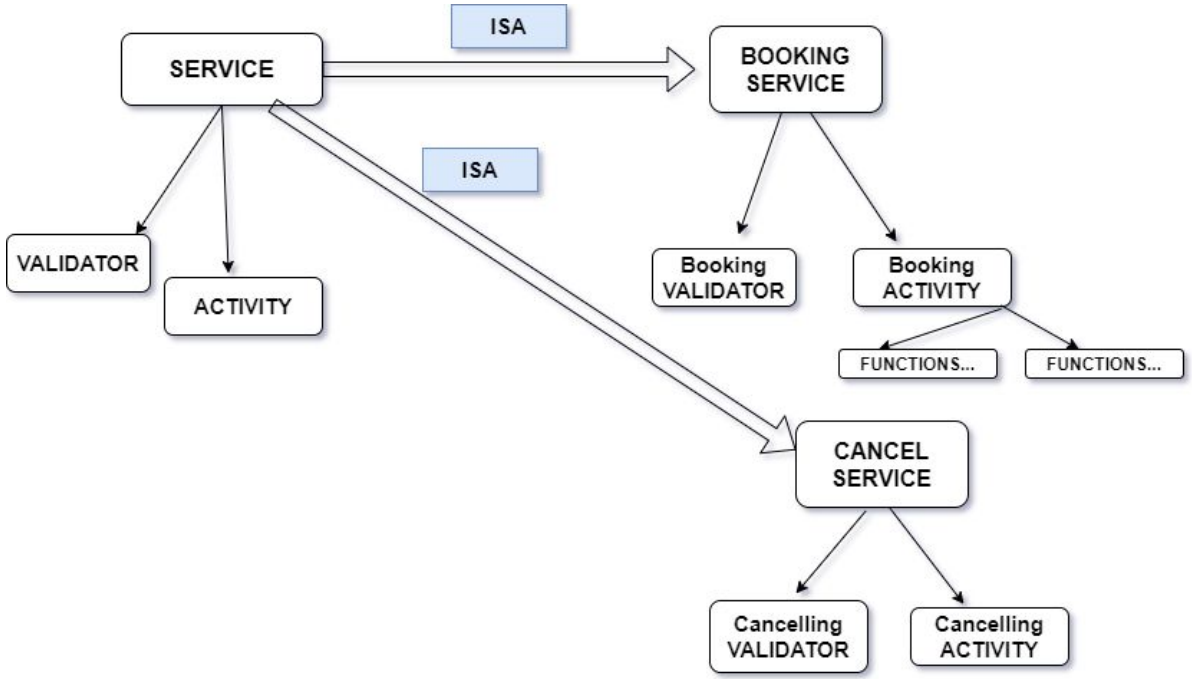
Project Description

QUESTION NO. 6) A flight reservation system for the internet is to be built. Clients enter the passenger name, starting airport and the destination airport. They choose the class of travel, economy, business, or first class. They have to enter the date of travel. The system does the booking unconditionally. Payment is accepted by credit card. We can abandon the transaction before payment is invoked. It is possible to cancel a booking on payment of penalty. Or to change the date of travel.

A. **Identification of initial OO design from the problem statement (suitably expanded to build a realistic project as was done for the first project)**
B. **OO design resulting from SRP, OCP, and LSP**
C. **Implementation**

A) Identification of initial OO Design

The four principles of **object**-oriented programming are **encapsulation**, **abstraction**, **inheritance**, and **polymorphism**. All classes have their states, attributes private from other non-child classes (**encapsulation**). The **abstraction** is seen in places where the internal working of the methods hidden from the outside world and return the appropriate value.



B) OO Design resulting from SRP, OCP & LSP

SRP is followed since we have different classes meant to do different tasks and all work independently of each other. (Eg. BookingValidator ➜ only validates , BookingActivity ➜ Tasks to update DB.)

OCP followed since to add features to existing Flight System we don't necessarily change existing code and just add new modules. (Eg, Add changes to child classes like FlightClassValidator, child class of BookingValidator).

LSP defines that objects of a superclass shall be replaceable with objects of its subclasses without breaking the application. That requires the objects of your subclasses to behave in the same way as the objects of your superclass.

B) OO Design resulting from SRP, OCP & LSP

The **list of classes** are as follows, all of which have their respective subclasses and methods (all are derived from SERVICE class, and hence have VALIDATOR and ACTIVITY and TASK subclasses) :--

i)BookingService

ii)Cancel

iii)NewCreditCard

iv)UpdateDate

v)Payment

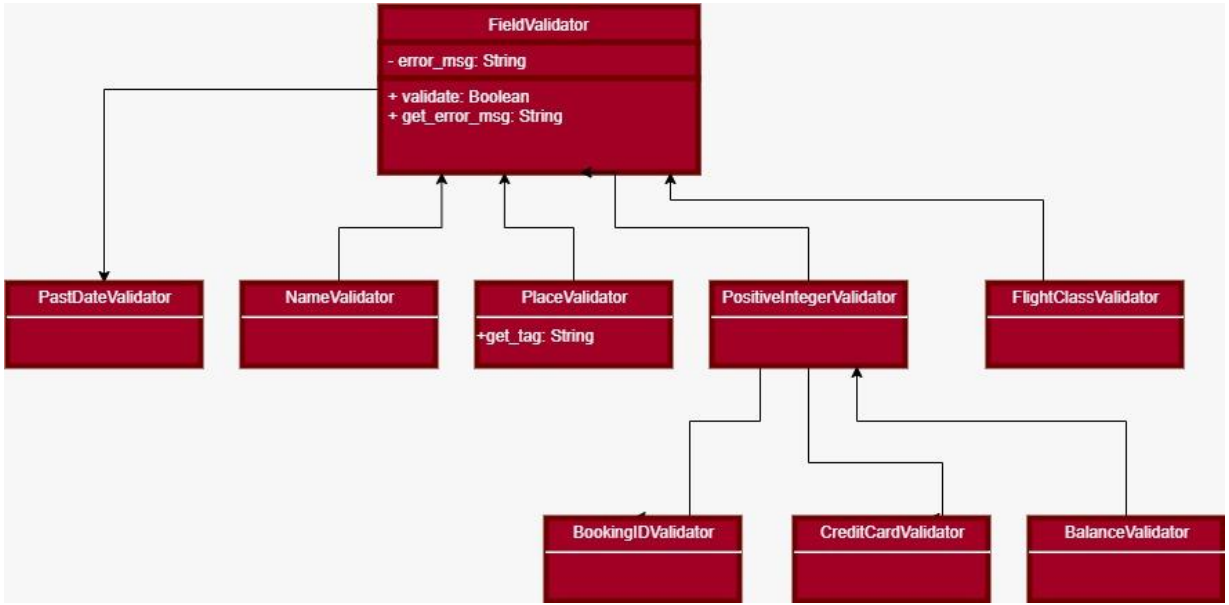
vi)AddOn

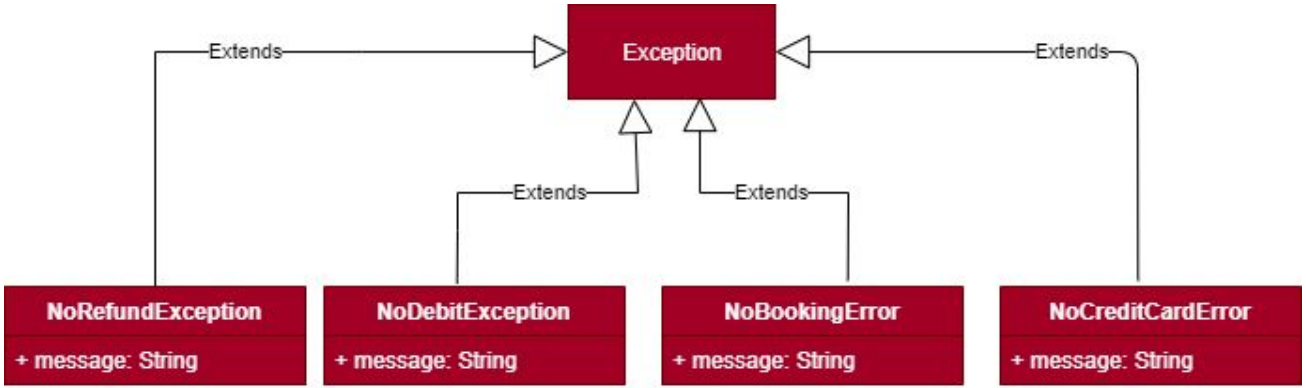
How SOLID principles are followed in the OO Design

- The complex functionalities of the Flight Reservation application are implemented using OOP and to follow SOLID the logic is divided into subclasses through which we have followed the ***simple responsibility principle*.**
- Any specific behaviour is implemented by extending the class and making the changes in the new subclass created thus following the ***open closed principle***.
- Abstract classes are created which are implemented using the subclasses following the ***liskov substitution principle.***

Flow Diagram

Click : [OPEN CODE FLOW DIAGRAM]





C) Implementation

All the functionalities are available as APIs

a) **Booking Functionality :** After the call to flight booking the input fields are Validated. If the validation fails, returns an error. Else perform the database operations to save the booking. Returns the booking id.

b) **Cancellation of Booking:** After validating the booking id, cancels the booking and flight my removing the field from database. If the booking is confirmed refunds appropriate amount into the credit card that was earlier used to make the payment.

c) **Confirm Payment:** Validates and confirms the input booking ID & credit card no. Returns as message the amount debited.

d) **Change Date of travel**: Takes as input the booking id and the new date. Validates the date and change the date in database. Additionally removes previous flight from the database if the user was the only passenger in the previous flight.

e) **Add on Facility**:** Input is the booking id and the credit card to be used for payment. Deduct 500 from the given card and notify the same in the success message. Update the database to include the facility.

f) **Booking Details**: Takes as input the booking id and if the validation is successful returns the value in booking table to the user. Details returned are == name, source, destination, date of travel, class, booking status, credit card details, add on facility, flight id from database.

Thank You for listening so patiently!!