# Department of Computing

**SE-210: Software Design and Architecture**

**Class:** BESE-9AB

# Lab 03: Object Constraint Language (OCL)

**Date: October 28th , 2020**

# Lab Engineer: Sundas Dawood

# 

**Ahmed Hassan Ismail – BESE-9B - 237897**

# Lab 03: Object Constraint Language

### Introduction:

In this lab, students will learn how to constraints to UML class diagram according to a given scenario.

### Lab Objectives:

After the completion of this lab, students will be able to add constraints to UML class diagram.

### Tools:

* UML drawing tool

### Helping Material:

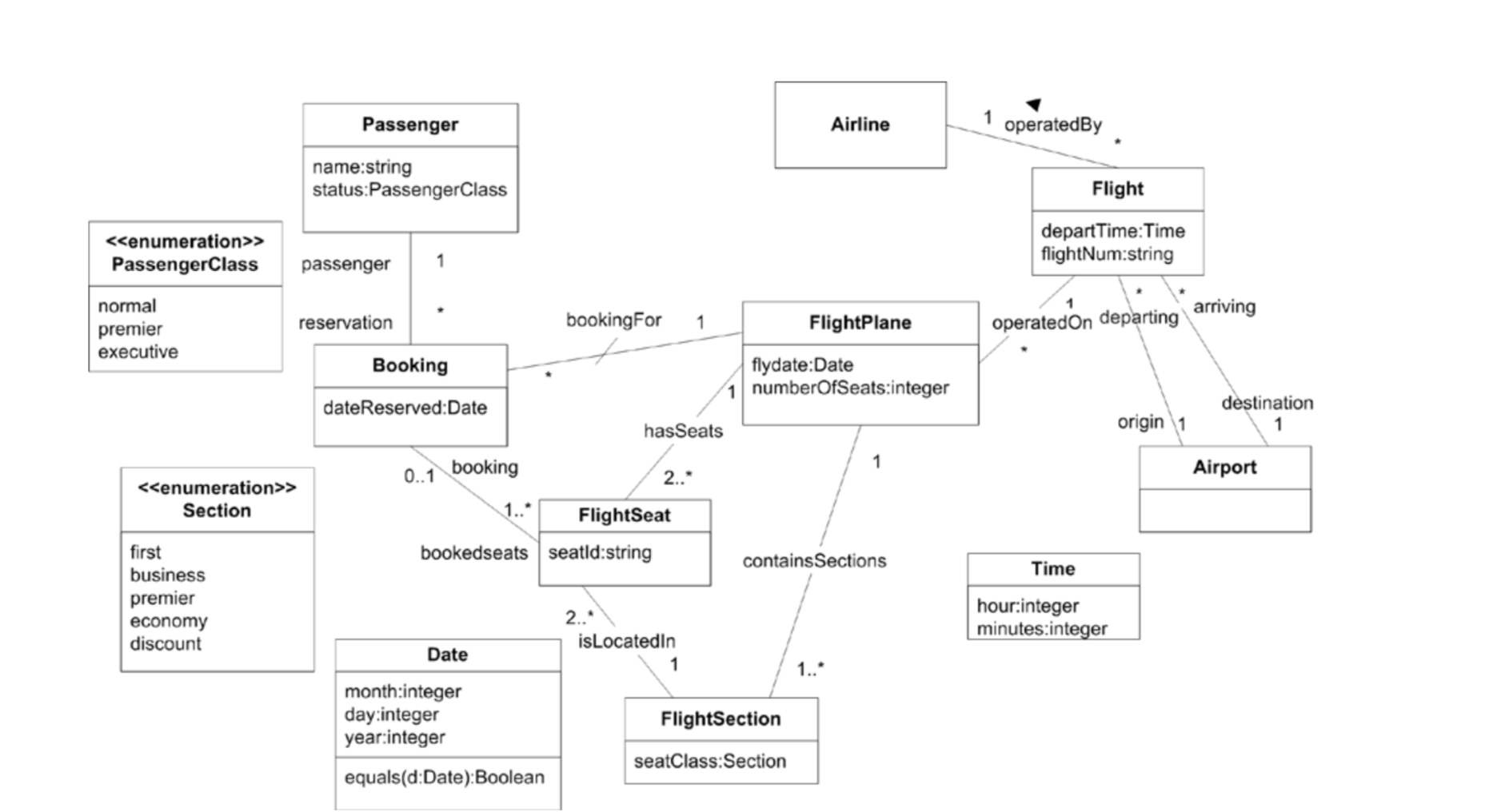
Helping material related to object constraint language can be downloaded from the LMS.

Chapter 9 of the uploaded OCL specifications

### Lab Tasks

### Task 1

This week’s task is to *add OCL constraints* of the below class diagram.



The above diagram describes a system that maintains information about flights (instance of Flight) and bookings (instances of Booking). Passengers are booked on flights and are assigned seats (instances of FlightSeat) at the time of booking. A flight is operated by a plane (instance of FlightPlane) on a particular date.

1. Specify the following invariants using the OCL:
   * For a given flight, the originating airport must not be the same as the destination  airport.

Answer: context Flight

inv: self.origin <> self.destination

* + If a flight plane has more than 300 seats then each of its flight sections must have  at least 20 seats and at most 100 seats.

Answer: context FlightPlane

inv: if self.numberOfSeats > 300 then self.FlightSection -> forAll(fs | fs.FlightSeat -> size() > 19 and fs.FlightSeat -> size() <101)

1. Define new query operations, using the OCL, that provide the following information from the class diagram:

* The set of bookings for an airline on a specified date.

Answer: context Airline def: FindBooking(d:Date):Set(Booking) = self.Flight.FlightPlane -> select(f | f.flydate.equals(d)) -> collect(Booking

* The flight sections on a flight that have seats that have not been booked.

Answer: context Flight def: OpenSection():Set(FlightSection) = self.FlightSection -> select(fs | fs.FlightSeat -> exists(s | s.Booking->isEmpty())

* The flight sections on a flight that have all seats booked.

Answer: context Flight def: ClosedSection():Set(FlightSection) = self.FlightPlane.FlightSection -> select(fs | fs.FlightSeat -> forAll(s | s.Booking->notEmpty()))

3. Modify the above class diagram to reflect the following requirements (include additional OCL constraints if needed)

* A flight attendant can be assigned to at most one flight section. Each section has  one attendant that is the supervisor for the section. A supervisor must have at least 10 years of service. If there are greater than 200 seats in a section then at least 4 attendants must be assigned to the section.

Answer: context FlightSection inv: self.supervisor.yearsOfService > 9 context FlightPlane inv: if self.FlightSeat -> size() > 200 then self.attendant -> size() > 3

* A flight can be associated with more than one airline (code sharing). The flight is still operated by one airline, but it can be associated with more than one airline. An airline cannot co-share a flight with itself.

Answer: context Flight inv: self.code-shared -> excludes(self.operated-by)

* A flight can have many “legs”, that is, it can have more than one destination (each destination is different). Each leg has an origin and a destination, where the origin of a leg other than the first leg is the destination of the previous leg.

Answer: context Flight

inv: self.Leg -> subsequence(1, self.Leg->size()-1) -> forAll (lg | lg.dest = (self.Leg ->at(indexOf(lg)+1)).origin)

**Answer:**

|  |
| --- |
| Solution |
| Task 1  I have solved the tasks above in below the questions itself. |

### Deliverables

Compile a single word document by filling in the solution part and submit this Word file on LMS. This lab grading policy is as follows: The lab is graded between 0 to 10 marks. The submitted solution can get a maximum of 5 marks. At the end of each lab, there will be a lab quiz related to the tasks. The quiz has a weightage of 5 marks. Insert the solution/answer in this document. You must show the implementation of the tasks in Rational Rose, along with your completed Word document to get your work graded. You must also submit this Word document on the LMS. In case of any problems with submissions on LMS, submit your Lab assignments by emailing it to Sundas Dawood <sundas.dawood@seecs.edu.pk>