CONCORDIA UNIVERSITY

Deliverable #3 (Class Diagram)

Project Topic: Online Flight Reservation System

Team: 6 Ideas

Team Members:

Family Name	First Name	Student ID
Afrasiabi	Seyedeh Nazanin	40059181
Afrasiabi	Seyedeh Negar	40057810
Ahmed	Owais	40040018
Promwongsa	Nattakorn	40051020
Rasouli	Farinaz	40047293
Azadiabad	Siamak	26592856

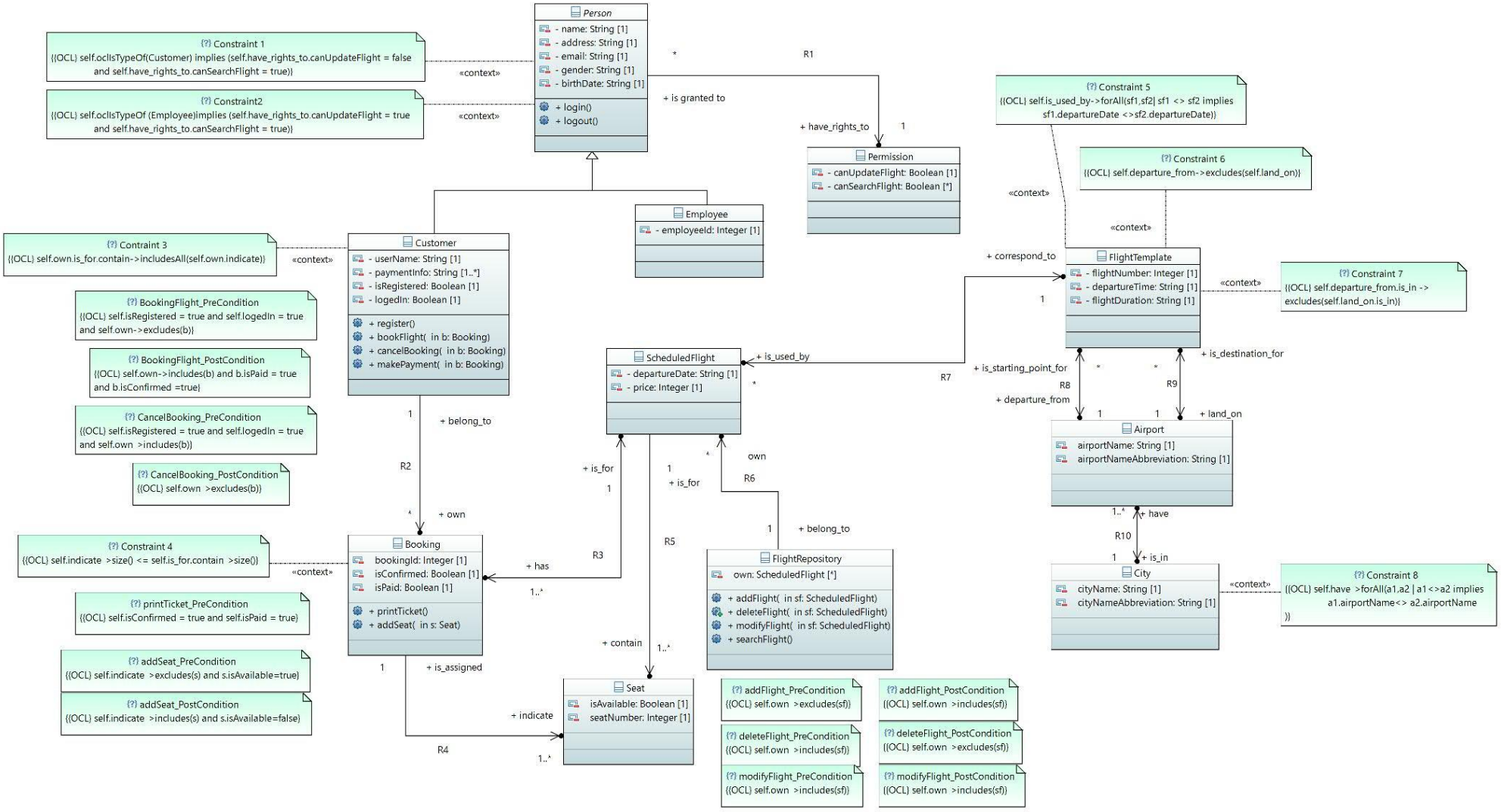
February 26, 2018

Model Driven Software Engineering Course

Table of Contents

1. Class Diagram and OCL Constraints in Papyrus

1.1 Class Diagram



2. Class Description

2.1 Person

This class is a generalization for Customer and Employee and it contains general information of a person.

Class : Person <<Abstract>>				
Data	Name	Visibility	Type	Description
Attributes	name	Private	String	This attribute represents a Person' name
	address	Private	String	This attribute represents a Person's address
	email	Private	String	This attribute represents a Person's email
	gender	Private	String	This attribute represents a Person's gender
	birthDay	Private	String	This attribute represents a Person's birth day
Functions	login	Public	-	This function is for logging in into the system
	logout	Public	-	This function is for logging out of the system
Associations	R1	-	-	Each person has a permission.
	Specialization{disjoint, complete}	-	-	It is an abstract class and superclass of Passenger and Employee classes.

2.2 Customer

This class consists of information about customers. A customer must register to the system in order to have permit for booking and cancelling a ticket.

Class: Customer				
Data	Name	Visibility	Type	Description
	userName	Private	String	This attribute represents a passenger's user name
	paymentInfo	Private	String Array	This attribute represents a passenger's credit card information including credit card number, CVV and expire date.
	isRegistered	Private	Boolean	This attribute represents a status of registration
	logedIn	Private	Boolean	This attribute represents a status of logging in
Functions	register()	Public	-	This function is for registering in the system
	bookFlight()	Public	-	This function is for booking a flight
	cancelBooking()	Public	-	This function is for cancelling booking
	makePayment()	Public	-	This function is for making a payment
Associations	R2	-	-	Each customer can have zero to many bookings.

	Generalization from *Person*	-	-	It is a child class of Person class

2.3 Permission

This class indicates the permissions for searching and updating flights which can be granted to customers or Employees.

Class: Permission				
Data	Name	Visibility	Type	Description
Attributes	canUpdateFlight	Private	Boolean	This attribute represents a status of flight updating permission
	canSearchFlight	Private	Boolean	This attribute represents a status of flight searching permission
Functions	-	-	-	-
Associations	R1	-	-	A specific set of permissions can be assigned to one to many persons

2.4 Employee

This class contains information of employee.

Class: Employee				
Data	Name	Visibility	Type	Description
Attributes	employeeId	Private	Integer	This attribute represents employee ID
Functions	-	-	-	-
Associations	Generalization from *Person*	-	-	It is a child class of Person class

2.5 FlightRepository

This class contains list of all flights and all modifications for the flights can be done through this class.

Class: FlightRepository				
Data	Name	Visibility	Type	Description
Attributes	own	Private	scheduledFlight	This attribute represents a list of existing flights
Functions	searchFlight()	Public	-	This function is for searching existing flights
	addFlight()	Public	-	This function is for adding new flights

	deleteFlight()	Public	-	This function is for deleting existing flights
	modifyFlight()	Public	-	This function is for modifying existing flights
Associations	R6	-	-	Each flight repository owns all schedule flights.

2.6 ScheduledFlight

This class contains information about departure date and ticket price.

Class: ScheduledFlight				
Data	Name	Visibility	Type	Description
	departureDate	Private	String	This attribute represents departure date of a flight
	price	Private	Integer	This attribute represents flight's price
Functions	-	-	-	-
Associations	R3	-		Each schedule flight can have one to many bookings
	R5	-	-	Each schedule Flight contains one to many seats.
	R6	-	-	Each schedule flight belongs to one flight repository.
	R7	-	-	Each schedule flight corresponds to one flight template

2.7 Booking

This class provides information and functions for booking flights including flight confirmation and payment.

Class: Booking				
Data	Name	Visibility	Type	Description
Attributes	bookingId	Private	Integer	This attribute represents a booking ID
	isConfirmed	Private	Boolean	This attribute represents a status of booking confirmation
	isPaid	Private	Boolean	This attribute represents a status of payment
Functions	printTicket()	Public	-	This function is for printing a booking ticket
Associations	R2	-	-	Each booking belongs to one customer

	R3	-	-	Each booking is for only one scheduled flight
	R4	-	-	Each booking indicates one to many seats.

2.8 Seat

This class contains seats information such as seat number and seat availability

Class: Seat				
Data	Name	Visibility	Type	Description
Attributes	seatNumber	Private	Integer	This attribute represents a seat number
	isAvailable	Private	Boolean	This attribute represents a status of seat availability
Functions	-	-	-	-
Associations	R4	-	-	Each seat is assigned to only one booking
	R5	-	-	Each seat is for only one scheduled flight

2.9 FlightTemplate

For all flights in different date and same start point, end point, and time of departure, the flight number and departure time is same. This class contains this template information.

Class: FlightTemplate				
Data	Name	Visibility	Type	Description
Attributes	flightNumber	Private	Integer	This attribute represents flight number
	departureTime	Private	String	This attribute represents departure time
Functions	-	-	-	-
Associations	R7	-	-	Each flight template is used by zero to many scheduled flights
	R8	-		Each flight template departs from only one airport
	R9	-	-	Each flight lands on only one airport

2.10 Airport

This class represents airports of the flight.

Class: Airport				
Data	Name	Visibility	Type	Description
Attributes	airportName	Private	String	This attribute represents an airport name
	airportNameAbbreviation	Private	String	This attribute represents an airport abbreviation

Functions	-	-	-	-
Associations	R8	-	-	Each airport can be a starting point of zero to many flight templates
	R9	-	-	Each airport can be a destination of zero to many flight templates
	R10	-	-	Each airport is located in only one city.

2.11 City

This class represents cities of each airport.

Class: City				
Data	Name	Visibility	Type	Description
Attributes	cityName	Private	String	This attribute represents an airport name
	cityNameAbbreviation	Private	String	This attribute represents an airport abbreviation
Functions	-	-	-	-
Associations	R10	-	-	Each city can have one to many airports
	R11	-	-	-
	R12	-	-	-

3. OCL constraints

3.1 Invariants

3.1.1 Constraint 1

Description	Customers can search flights, but cannot update the flights.
OCL Constraint	**Context** Person **inv: self**.*oclIsTypeOf*(*Customer*) **implies** (**self**.*have_rights_to*.*canUpdateFlight* = **false and** **self**.*have_rights_to*.*canSearchFlight* = **true**)
Alternative Constraint	**Context** Person **inv: self**.*oclIsTypeOf*(*Customer*) **implies** (**self**.*R1*.*canUpdateFlight* = **false and self**.*R1*.*canSearchFlight* = **true**)

3.1.2 Constraint 2

Description	Employees must be able to search and update flights
OCL Constraint	**Context** Person **inv: self**.*oclIsTypeOf*(*Employee*) **implies** (**self**.*have_rights_to*.*canUpdateFlight* = **true and self**.*have_rights_to*.*canSearchFlight* = **true**)
Alternative Constraint	**Context** Person **inv: self**.*oclIsTypeOf*(*Employee*) **implies** (**self**.*R1*.*canUpdateFlight* = **true and self**.*R1*.*canSearchFlight* = **true**)

3.1.3 Constraint 3

Description	Customer can have only seats that belongs to a flight that he/she books only
OCL Constraint	**Context** Customer **inv: self**.*own*.*is_for*.*contain*->*includesAll*(**self**.*own*.*indicate*)
Alternative Constraint	**Context** Customer **inv: self**.*R2*.*R3*.*R5*->*includesAll*(**self**.*R4*)

3.1.4 Constraint 4

Description	Each booking should not have seats more than flight seats
OCL Constraint	**Context** Booking **inv: self**.*indicate*->*size*() <= **self**.*is_for*.*contain*->*size*()
Alternative Constraint	**Context** Booking **inv: self**.*R4* ->*size*() <= **self**.*R3*.*R5*->*size*()

3.1.5 Constraint 5

Description	Each flight template cannot have schedule flights which have the same departure date
OCL Constraint	**Context** FlightTemplate **inv: self**.*is_used_by*->*forAll*(sf1,sf2| *sf1* <> *sf2* **implies** *sf1*.*departureDate* <>*sf2*.*departureDate*)
Alternative Constraint	**Context** FlightTemplate **inv: self**.*R7*->*forAll*(sf1,sf2| *sf1* <> *sf2* **implies** *sf1*.*departureDate* <>*sf2*.*departureDate*)

3.1.6 Constraint 6

Description	For the same flight template, source and destination airports cannot be the same
OCL Constraint	**Context** FlightTemplate **inv: self**.*departure_from*->*excludes*(**self**.*land_on*)

Alternative Constraint	**Context** FlightTemplate **inv: self**.*R8->excludes*(**self**.R9)

3.1.7 Constraint 7

Description	For the same flight template, source and destination cities cannot be the same
OCL Constraint	**Context** FlightTemplate **inv: self**.*departure_from.is_in -> excludes*(**self**.*land_on.is_in*)
Alternative Constraint	**Context** FlightTemplate **inv: self**.*R8.R10 -> excludes*(**self**.*R9.R10*)

3.1.8 Constraint 8

Description	Each city cannot have more one airport with the same name
OCL Constraint	**Context** City **inv: self**.*have->forAll*(a1,a2 | *a1<>a2* **implies** *a1.airportName<> a2.airportName*)
Alternative Constraint	**Context** City **inv: self**.*R10->forAll*(a1,a2 | *a1<>a2* **implies** *a1.airportName<> a2.airportName*)

3.2 Pre-Condition and Post-Condition

3.2.1 Constraint 1

Description	Customers can book a flight once he/she already registered and logged in into the system
OCL Constraint	**Context** Customer :: bookFlight(b : Booking) **Pre: self**.*isRegistered* = **true and self**.*logedIn* = **true and self**.*own->excludes*(*b*) **Post: self**.*own->includes*(*b*) **and** *b.isPaid* = **true and** *b.isConfirmed* =**true**
Alternative Constraint	**Context** Customer :: bookFlight(b : Booking) **Pre: self**.*isRegistered* = **true and self**.*logedIn* = **true and self**.*R2->excludes*(*b*) **Post: self**.*R2->includes*(*b*) **and** *b.isPaid* = **true and** *b.isConfirmed* =**true**

3.2.2 Constraint 2

Description	The booking can only be cancelled if it already exists
OCL Constraint	**Context** Customer :: cancelFlight(b : Booking) **Pre: self**.*isRegistered* = **true and self**.*logedIn* = **true and self**.*own->includes*(*b*) **Post: self**.*R2->excludes*(*b*)
Alternative Constraint	**Context** Customer :: cancelFlight(b : Booking) **Pre: self**.*isRegistered* = **true and self**.*logedIn* = **true and self**.*R2->includes*(*b*) **Post: self**.*R2->excludes*(*b*)

3.2.3 Constraint 3

Description	The booking can only be printed if it is already paid and confirmed
OCL Constraint	**Context** Booking :: printTicket() **Pre: self**.*isConfirmed* = **true and self**.*isPaid* = **true**
Alternative Constraint	**------**

3.2.4 Constraint 4

Description	The seat can only be added if it is available
OCL Constraint	**Context** Booking :: addSeat(s : Seat) **Pre: self**.*indicate->excludes*(*s*) **and** *s.isAvailable*=**true** **Post: self**.*indicate->includes*(*s*) **and** *s.isAvailable*=**false**
Alternative Constraint	**Context** Booking :: addSeat(s : Seat) **Pre: self**.*R4->excludes*(*s*) **and** *s.isAvailable*=**true** **Post: self**.*R4->includes*(*s*) **and** *s.isAvailable*=**false**

3.2.5 Constraint 5

Description	The scheduled flight can only be added if it does not exist
OCL Constraint	**Context** FlightRepository :: addFlight(sf : ScheduledFlight) **Pre: self**.*own->excludes*(*sf*) **Post: self**.*own->includes*(*sf*)
Alternative Constraint	**Context** FlightRepository :: addFlight(sf : ScheduledFlight) **Pre: self**.*R6->excludes*(*sf*) **Post: self**.*R6->includes*(*sf*)

3.2.6 Constraint 6

Description	The scheduled flight can only be deleted if it already exists
OCL Constraint	**Context** FlightRepository :: deleteFlight(sf : ScheduledFlight) **Pre: self**.*own->includes*(*sf*) **Post: self**.*own->excludes*(*sf*)
Alternative Constraint	**Context** FlightRepository :: deleteFlight(sf : ScheduledFlight) **Pre: self**.*R6->includes*(*sf*) **Post: self**.*R6->excludes*(*sf*)

3.2.7 Constraint 7

Description	The scheduled flight can only modified if it already exists
OCL Constraint	**Context** FlightRepository :: deleteFlight(sf : ScheduledFlight) **Pre: self**.*own->includes*(*sf*) **Post: self**.*own-> includes* (*sf*)
Alternative Constraint	**Context** FlightRepository :: deleteFlight(sf : ScheduledFlight) **Pre: self**.*R6->includes*(*sf*) **Post: self**.*R6-> includes*(*sf*)