Exercise list of UML Interface Specification

Exercises: October 11, 2019

1. By using OCL, let specify the operation: `consigueDetallesCliente(in_cliente: IdentificadorCliente):DetallesCliente` of the interface `GestionClientes`, which class diagram is shown in figure below.

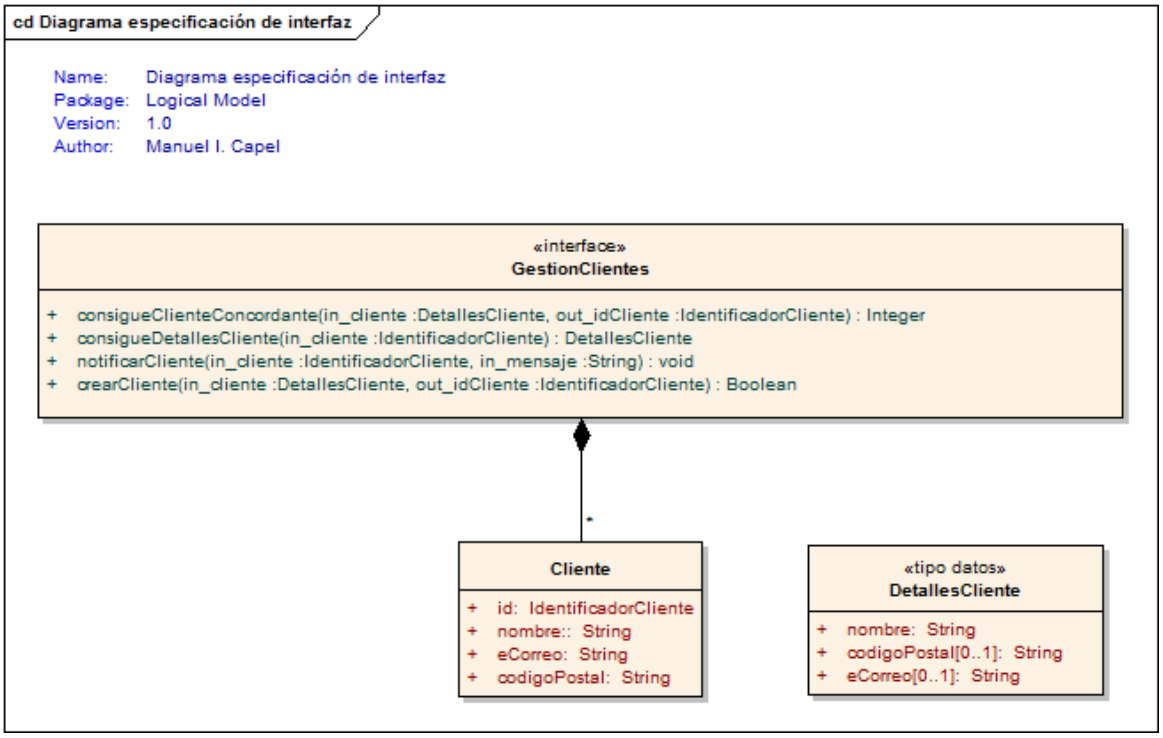

Figure 1: IIM ICustomerManagement

Note: OCL operations reference guide can be found at `https://wiki.eclipse.org/Acceleo/OCL_Operations_Reference`

2. We need to develop an *interface information model* (IIM) of a enrollment system for universities around the World. The name of the interface will be `IEnrollment`:
   - `getUniversity`(Imagine which parameters has to have this operation):`UniversityDetails`
   - `getCourse`(Imagine which parameters has to have this operation):`void`
   - `makeEnrollment`(Imagine which parameters has to have this operation):`enrollmentAcknowledge`

   Other entities in the asked IIM to be defined:

   - University
     - Name
     - Identifier
     - Shanghai ranking
     - Tuition fee per year
     - Any missing other that could be important too

- College or Faculty
  - Name
  - Application dates range
  - Any missing other that could be important too
- Course
  - Name
  - Teaching dates
  - SeatsAvailableAtDate
  - Any missing other that could be important too
- Enrollment
  - Reference
  - CourseCode[1..n] courses
  - Any missing other that could be important too
- Student
  - Name
  - e-Mail
  - postCode
  - Any missing other that could be important too

3. By using OCL, let specify the operation `getStudentDetails(in_student: StudentIdentifier)` is an operation of the interface `StudentManagement`, which is included in a UML Interface Diagram to the one shown in figure below (interface class for general student management):

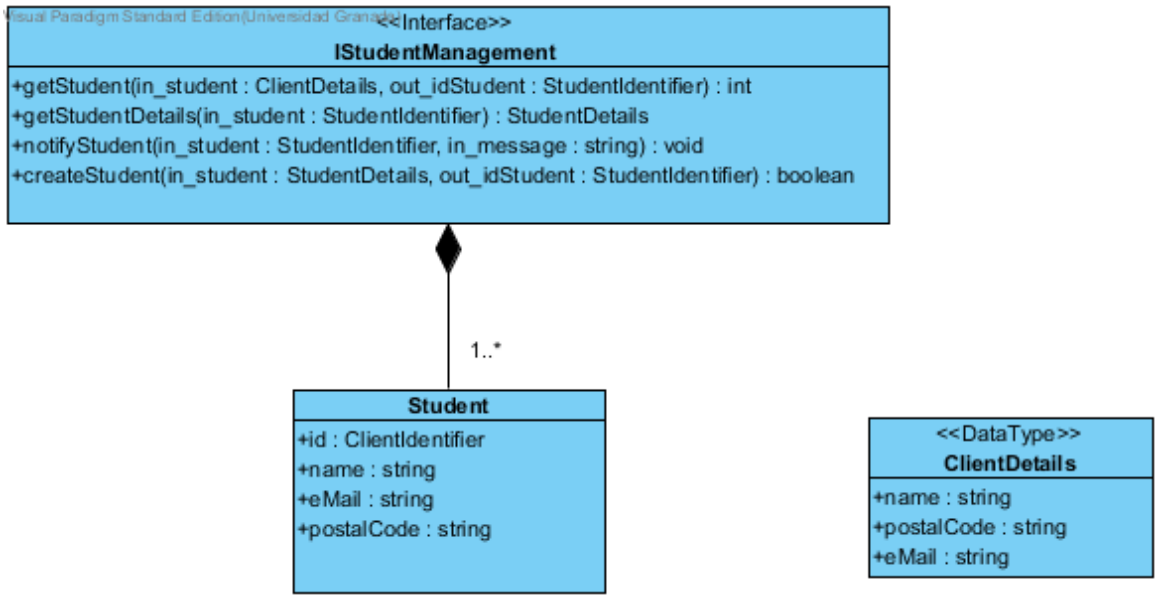

Figure 2: IIM IStudentManagement

4. Taking into account the Responsibility Diagram of Interface given by figure 3, specify with OCL the following invariants:
   (a) All flights objects must have a duration attribute that is less than 4 hours
   (b) The maximum number of passengers on a flight may not exceed 500
   (c) For every passenger, the age attribute must be greater than or equal to the class attribute `minAge`
   (d) The duration attribute of all flight objects must be equal to the difference between the arrival time and departure time attributes
   (e) The airport from which a flight is leaving must be different from the destination airport
   (f) For every flight, the name of the airport from which is leaving must be "Amsterdam"

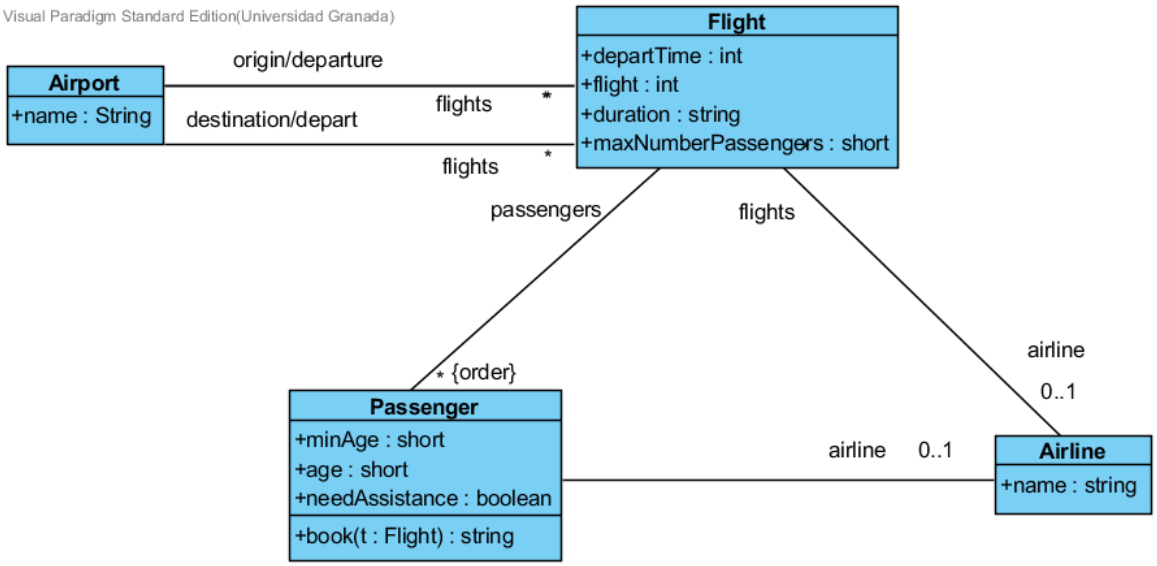

Figure 3: RDI of flights

5. Specify the operation `IHotel::MgtmakeReservation` by using OCL and the operator listed in table below: `exits, select, asSequence, first`:

   Note: in the precondtion it's necessary to make sure that `id_student` represents a valid student identifier and, in the postcondition, that the returned details by the operation match with the student details whose identifier is `in_student`

OCL operator	**meaning**
exists	existential quantifier(predicate)
select	select an element from a range, fulfilling a condition given
in	comes before the specification of the postcondition result
asSequence	Returns a Sequence containing all elements of self. Element ordering is preserved when possible
first	Returns the first element of self

6. Factorize the common information elements of interfaces: IMakeReservation and `ITakeUpReservation` and put them in a new interface: `IReservationSystem`. Then, the interfaces `IMakeReservation` and `ITakeUpReservation` inherit from `IReservationSystem`.

7. Make the class diagrams of the interfaces `IReservationSystem` and rebuild the `IMakeReservation` one.

8. Write the correct code of a Maven component that performs the *managed bean* `MensajeBean` injection into the *bean* `HolaMundo2`, such as when the second one gets executed yields the following output:

   - When the page `http://localhost:8080/holamundo2/` is opened, we get the following mesage: `Hola a todo el Mundo y parte del extranjero!`
   - The console of our IDE will show: `Hola Mundo-2 ha comenzado!  Nada aun!`

```java
import java.io.Serializable;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.ManagedProperty;
import javax.faces.bean.RequestScoped;

//Bean management instructions and the scope of the managed ben have been omitted
public class HolaMundo2 implements Serializable {

        //The annotation used to program a property dependeny injection is missed

        private MensajeBean mensajeBean;
        private String mensaje="Nada aun!";

        public HolaMundo2(){
                System.out.println("Hola Mundo-2 ha comenzado!");
                System.out.println(mensaje);
        }
//Complete the following code so that the program runs correctly
}
import java.io.Serializable;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.ManagedProperty;
import javax.faces.bean.RequestScoped;

//Bean management instructions and the scope of the managed ben have been omitted
public class MensajeBean implements Serializable{
  private String mensaje= "Hola a todo el Mundo y parte del extranjero!";
//Complete the following code so that the program runs correctly
}
```