Programación Avanzada

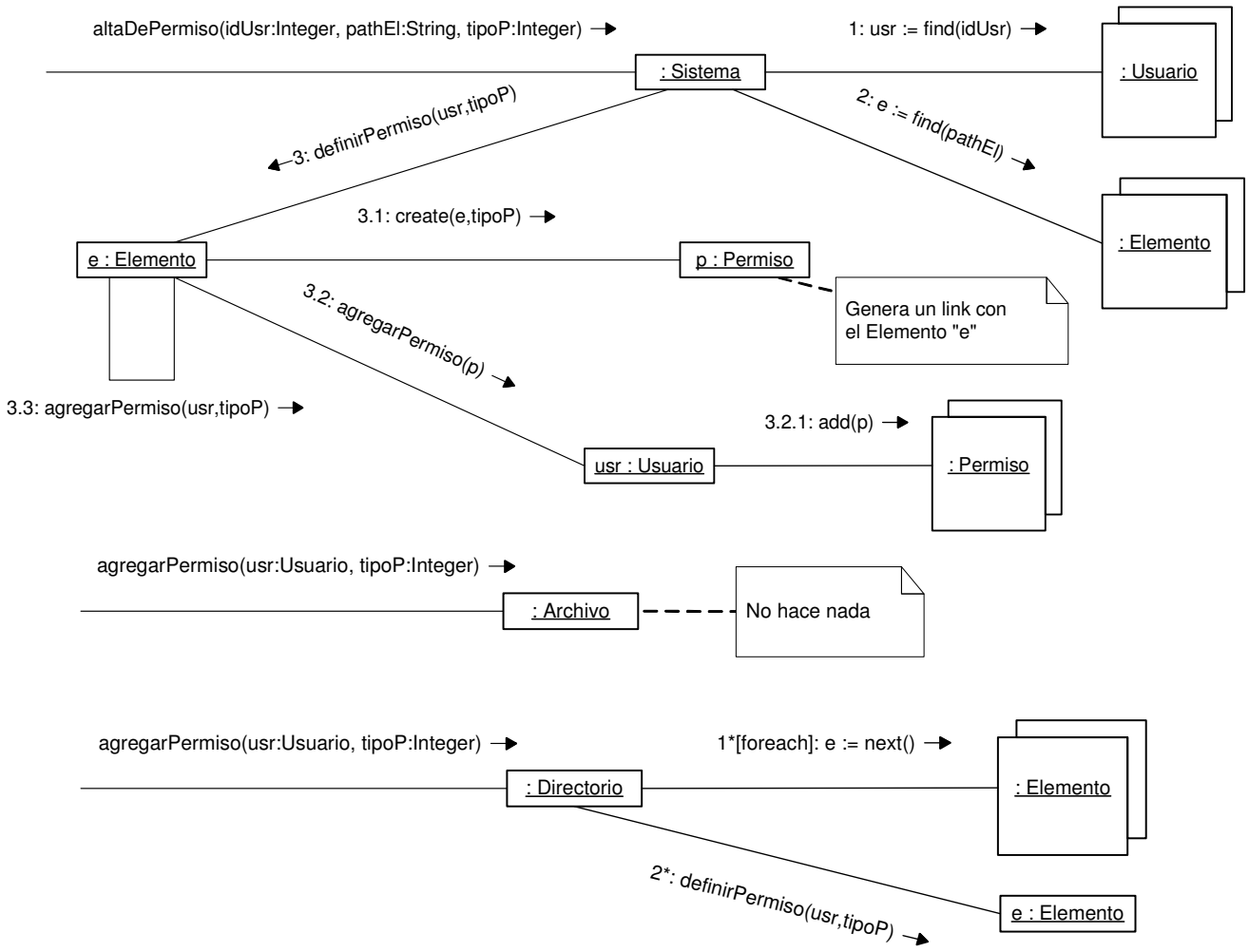
EXAMEN JULIO 2009

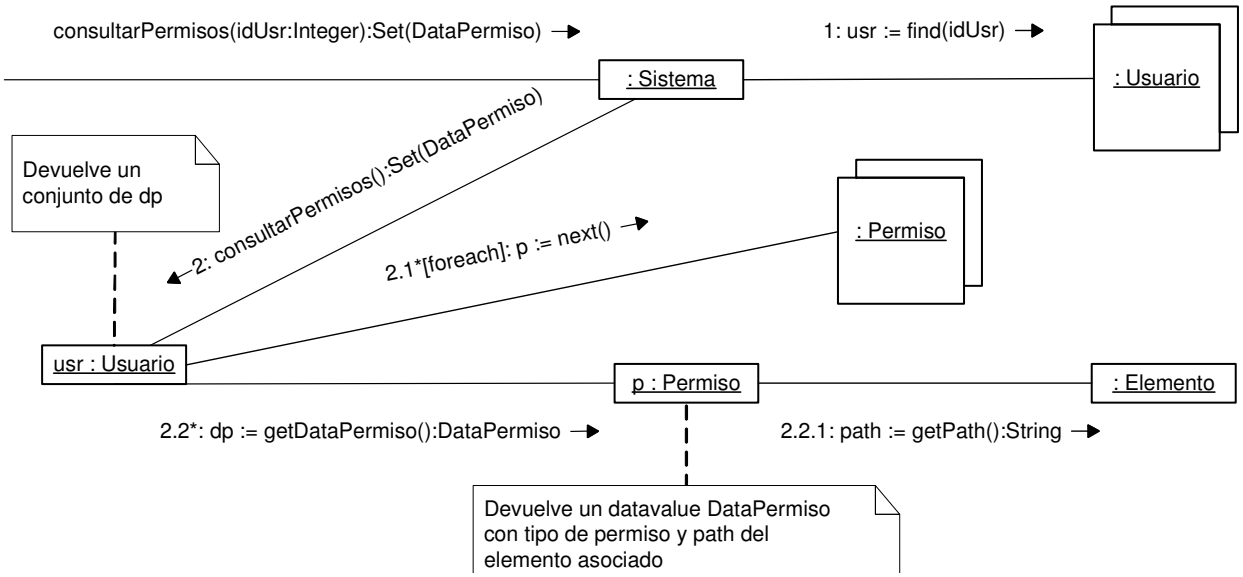
30/07/2009

SOLUCIÓN

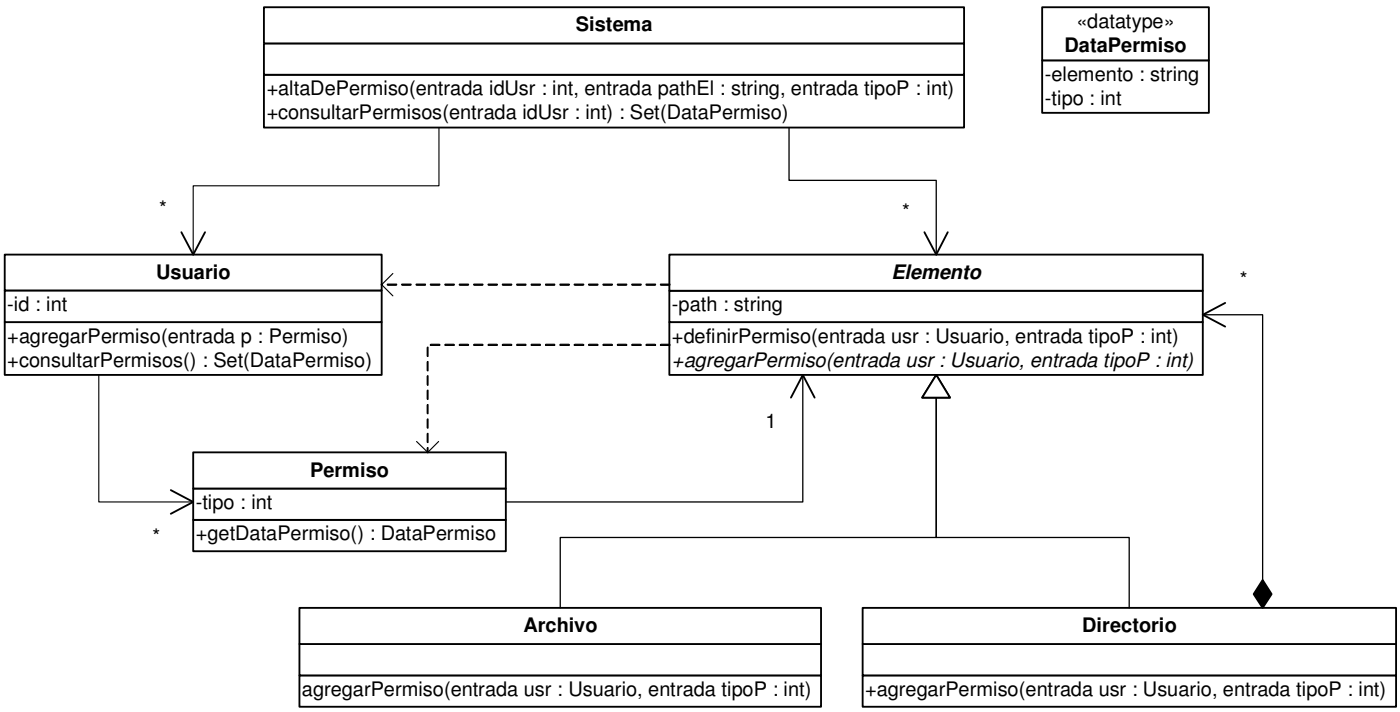
Problema 1

a) Diagramas de Comunicación





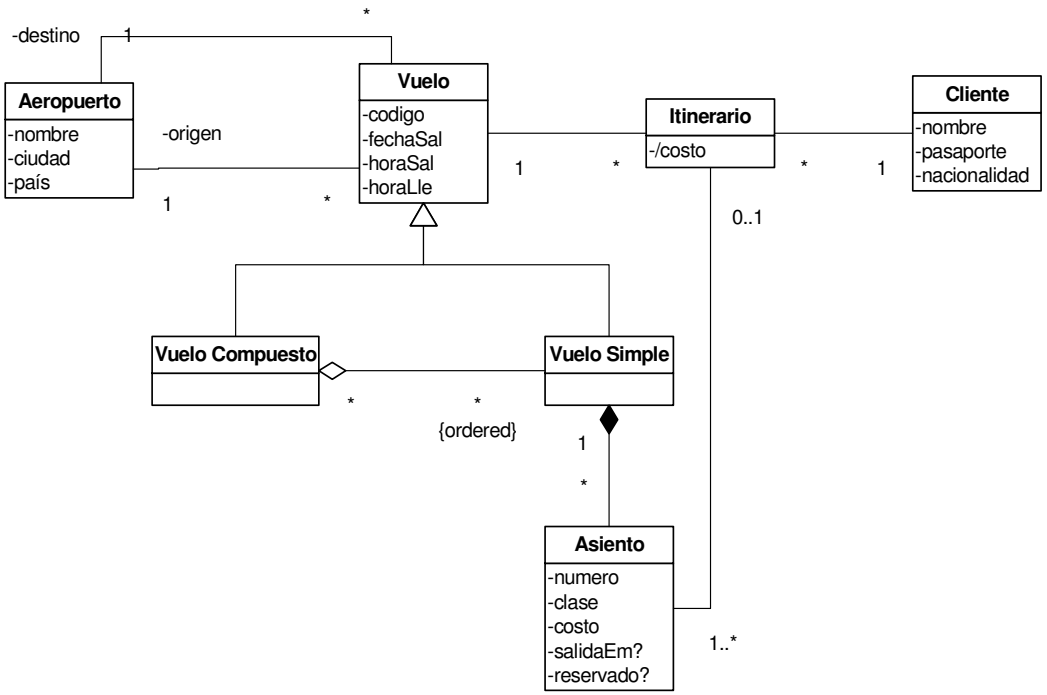
b) Diagrama de Clases de Diseño



Problema 2

Se desea desarrollar un prototipo de un sistema de reserva y compra de boletos por Internet para los vuelos de una aerolínea.

i. Realice el Modelo de Dominio de la realidad planteada (exclusivamente el diagrama de modelo de dominio UML y las restricciones en lenguaje natural).



Una segunda opción sería considerar el Itinerario como una clase de asociación entre Cliente y Vuelo (lo que asume que un cliente no tiene más de un itinerario para un mismo vuelo).

inv: --El atributo Nombre identifica al Aeropuerto

inv: --El atributo Codigo identifica al Vuelo

inv: -- El aeropuerto de inicio de todo vuelo compuesto es el mismo que el aeropuerto de inicio del primer vuelo que lo compone.

inv: -- El aeropuerto de destino de todo vuelo compuesto es el mismo que el aeropuerto de destino del último vuelo que lo compone.

inv: -- El número de asiento no se repite en un vuelo.

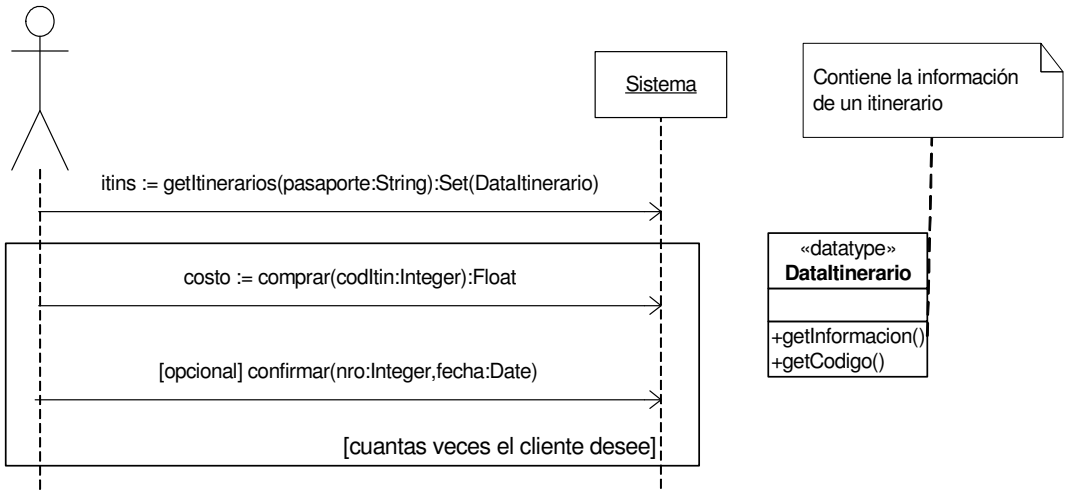
inv: --El atributo Pasaporte identifica al Cliente

inv: --Si un asiento está en un itinerario entonces está reservado

inv: -- El precio de un itinerario es la suma de los costos de los asientos que registra.

inv: -- Los asientos de un itinerario forman parte del vuelo del itinerario

ii. Realice el diagrama de secuencia del sistema del caso de uso Compra de Ticket a Partir de un Itinerario.



Problema 3

a)

1	`void main(){`	
2	`Clase c1;`	`Constructor por defecto`
3	`Clase c2(3,2.54);`	`Constructor comun`
4	`Clase c3=c2;`	`Constructor copia`

5	`c1=c2+c3;`	`Constructor por copia (parámetro c3)`
		`Operator+`
		`Operator=`
		`Constructor por copia (return de +)`
		`Destructor (del return del +)`
		`Destructor (del parámetro c3)`

6	`c1.desplegar();`	`Método de Desplegar`

7	`}`	`Destructor (de c3)`
		`Destructor (de c2)`
		`Destructor (de c1)`

b)

i. El objetivo del patrón Singleton es asegurar que una clase tenga una sola instancia y proveer un acceso global a ella.

ii. De un ejemplo en *c++* (archivos de cabecera y de implementación) de una clase en la cual se aplique dicho patrón. No es necesario incluir directivas al preprocesador.

```
//MiSingleton.hh

#ifndef MISINGLETON_HH
#define MISINGLETON_HH

class MiSingleton {
private:
      static MiSingleton * instancia;
      MiSingleton();
public:
      static MiSingleton * getInstancia();
      void operacion();
};

#endif
```

```
//MiSingleton.cc

#include <stdio.h>
#include <iostream>
#include "MiSingleton.hh"

using namespace std;

MiSingleton* MiSingleton::instancia = NULL;

MiSingleton::MiSingleton() {
      cout << "constructor de MiSingleton\n";
}

MiSingleton * MiSingleton::getInstancia() {
   if (instancia == NULL)
      instancia = new MiSingleton();
   return instancia;
}

void MiSingleton::operacion() {
      cout << "operacion()\n";
}
```