Programación Avanzada

SOLUCIÓN SEGUNDO PARCIAL 2015

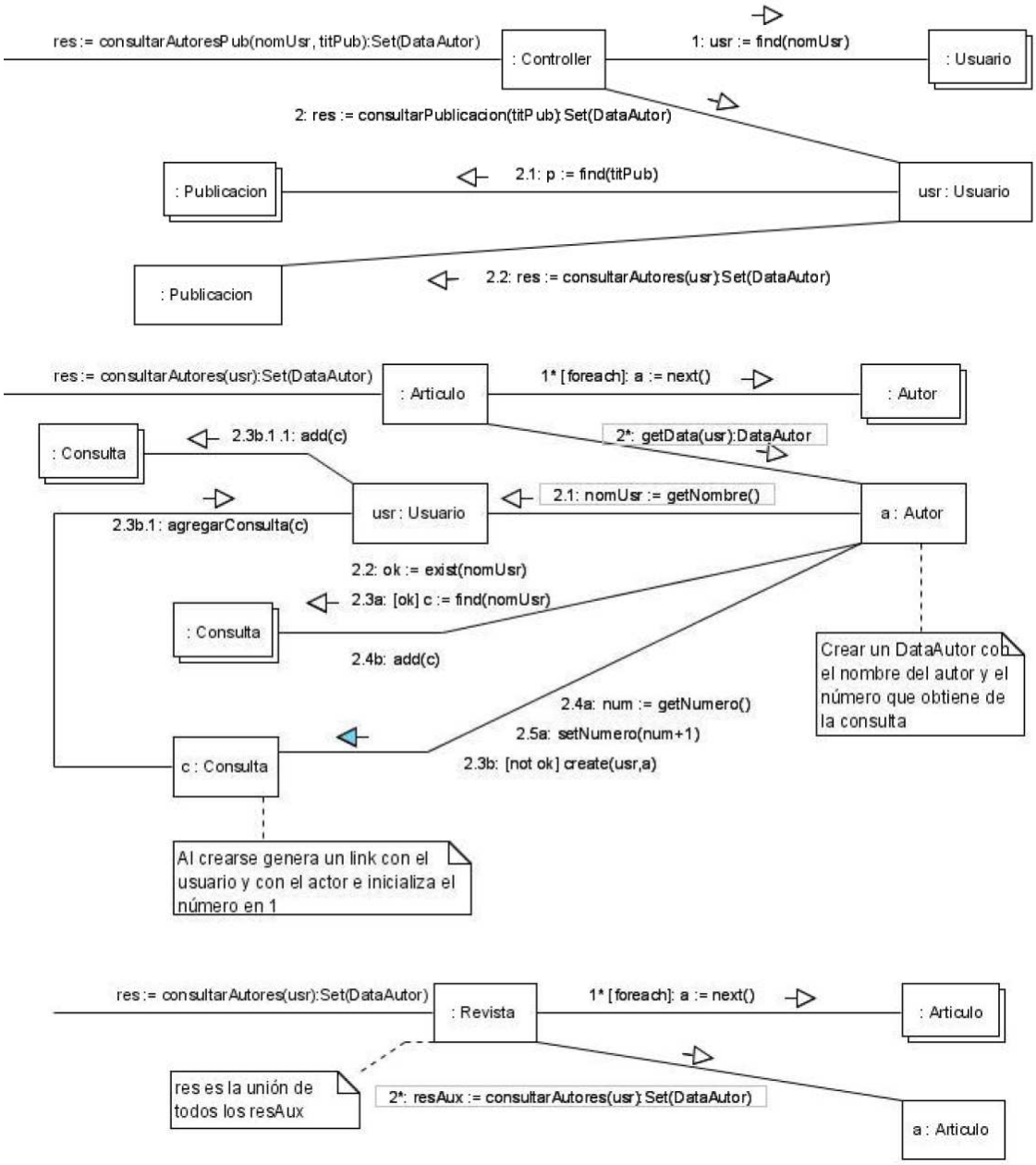
Problema 1 (35 puntos)

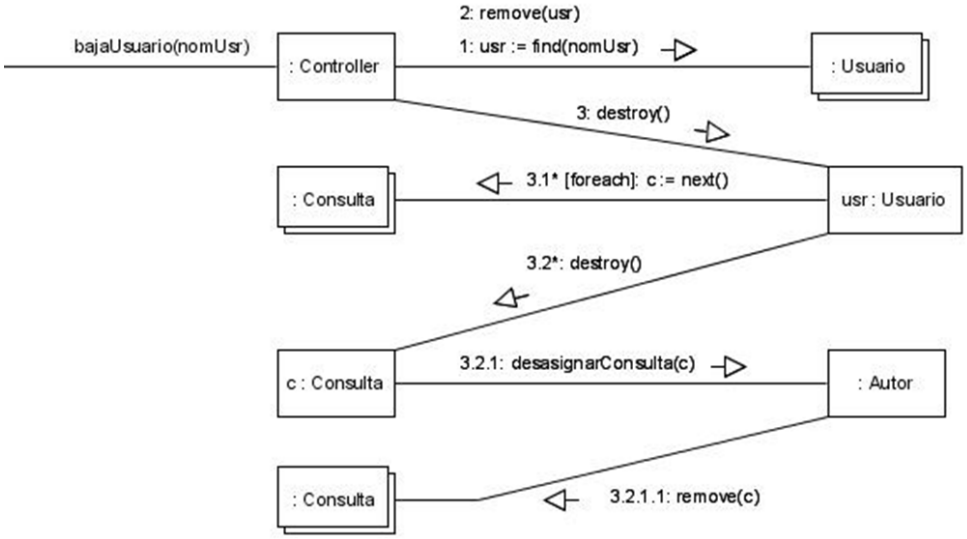
i) Un controlador es una clase que implementa las operaciones del sistema.

ii) Controladores de Fachada: contiene todas las operaciones del sistema.
Controlador de Caso de Uso: contiene todas las operaciones de un mismo caso de uso.
Controlador de Mini Fachada: contiene operaciones de varios casos de uso relacionados.

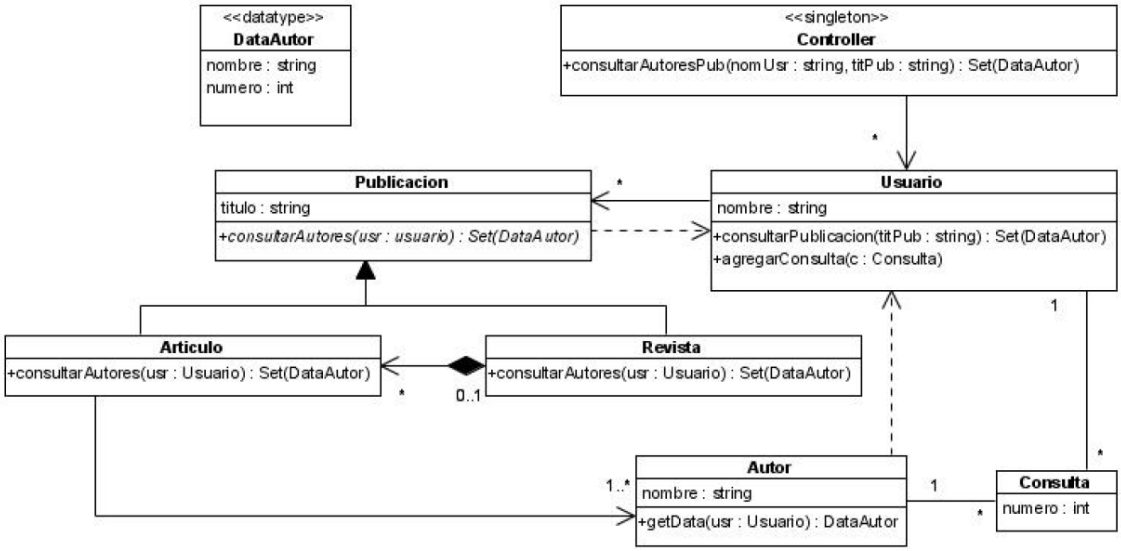
iii) Las interfaces del sistema contienen las operaciones del sistema, las cuales son implementadas por los controladores.

b) i) Diagramas de Comunicación





ii) Diagrama de Clases de Diseño



Problema 2 (25 puntos)

a) Implementar completamente en C++ una clase *singleton* denominada A, que tiene un atributo privado de tipo int, denominado *dato*.

```
class A {
   private:
            int dato;
            static A *instance;
            A();
    public:
            static A *getInstance();
            int getDato();
            void setDato(int);
}
```

```
A *A::instance = NULL;

A* A::getInstance() {
   if (instance == NULL)
         instance = new A;
   return instance;
}

int A::getDato() {
   return dato;
}

void A::setDato(int d) {
   dato = d;
}
```

b) i) Implementar las declaraciones en C++ de las clases *Prestamo* y *Sala*, y las de los datatypes *DataPrestamo* y *DataSala*.

```
class Prestamo {
	private:
		int codigo;
		ItemPrestable * ip;
	public:
		Prestamo(int, ItemPrestable *);
		virtual ~Prestamo();
		int darCodigo();
		DataItemPrestable * darInfoItem();
		virtual DataPrestamo * darInfoPrestamo()=0;
};

class Sala: public Prestamo {
	private:
		int horaRetiro;
	public:
		Sala(int, ItemPrestable *, int);
		int darHoraRetiro();
		DataPrestamo * darInfoPrestamo();
};

class DataPrestamo {
	private:
		int codigo;
		DataItemPrestable * dip;
	public:
		DataPrestamo(int, DataItemPrestable *);
		virtual ~DataPrestamo();
		int darCodigo();
		DataItemPrestable * darInfoItem();
};

class DataSala : public DataPrestamo {
	private:
		int horaRetiro;
	public:
		DataSala(int, DataItemPrestable *, int);
		int darHoraRetiro();
}
```

ii) Implementar los constructores y destructores de las clases *Prestamo* y *Sala*, y de los datatypes *DataPrestamo* y *DataSala*, e implementar la operación *darInfoPrestamo* en la clase *Sala*.

```
Prestamo::Prestamo(int c, ItemPrestable * i) {
	codigo = c;
	ip = i;
}

Prestamo::~Prestamo() {
}

Sala::Sala(int c, ItemPrestable * i, int h): Prestamo(c, i) {
	horaRetiro = h;
}
```

```
DataPrestamo * Sala::darInfoPrestamo() {
        return new DataSala(darCodigo(), darInfoItem(), horaRetiro);
}

DataPrestamo::DataPrestamo(int c, DataItemPrestable * i) {
        codigo = c;
        dip = i;
}

DataPrestamo::~DataPrestamo() {
       delete dip;
}

DataSala::DataSala(int c, DataItemPrestable * i, int h): DataPrestamo(c, i) {
       horaRetiro = h;
}
```

iii) Implementar en C++ la operación *darPrestamos*.

```
void Sistema::darPrestamos(ICollection *result) {
        IIterator * it = prestamos->getIterator();
        Prestamo * p;
        while (it->hasCurrent()) {
                p = it->getCurrent());
                result->add(p->darInfoPrestamo());
                it->next();
        }
        delete it;
        return result;
}
```