Programación 4

EXAMEN DICIEMBRE 2022 - SOLUCIÓN

Por favor siga las siguientes indicaciones:

- Escriba con lápiz y de un solo lado de las hojas
- Escriba su nombre y número de documento en todas las hojas que entregue
- Numere las hojas e indique el total de hojas en la primera de ellas
- Recuerde entregar su número de parcial junto al parcial
- Está prohibido el uso de computadoras, tabletas o teléfonos durante el parcial

Problema 1 (30 puntos)

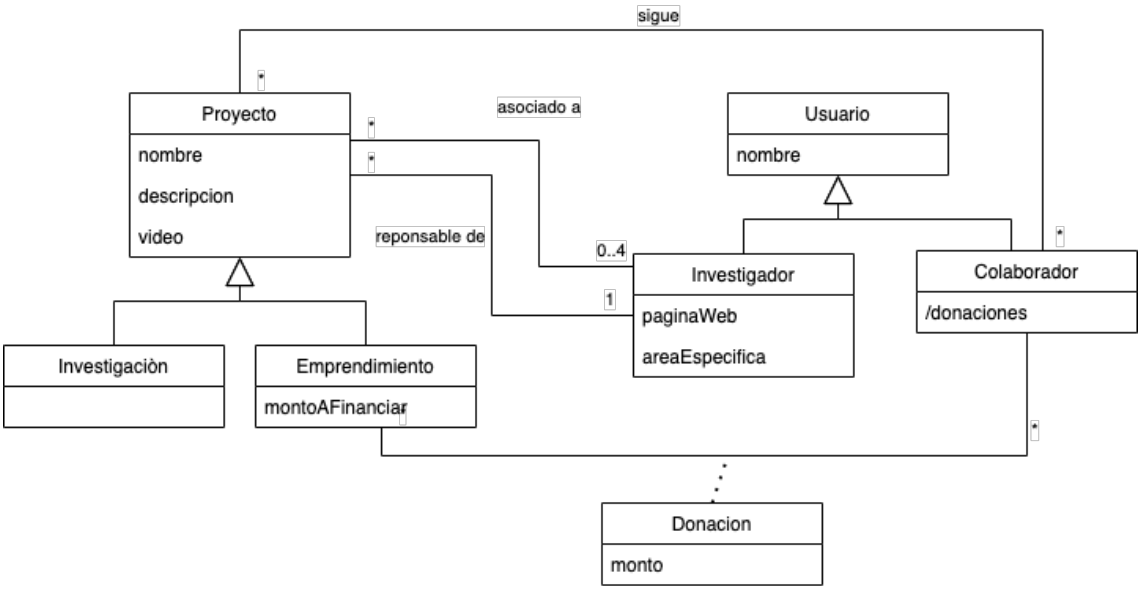
Se desea modelar un sistema para un portal de innovación en el cual los usuarios pueden cargar proyectos y tratar de conseguir financiamiento para llevarlos a cabo.

Se pide:

a) ¿Cuál es el propósito de la actividad de análisis?

El propósito es definir requerimientos describiendo de forma precisa cuál debe ser el comportamiento esperado del sistema. Se trabaja sobre un modelo de casos de uso viendo al sistema como una unidad y definiendo protocolos y estructura q caractericen el uso del sistema por parte de los actores en distintos escenarios.

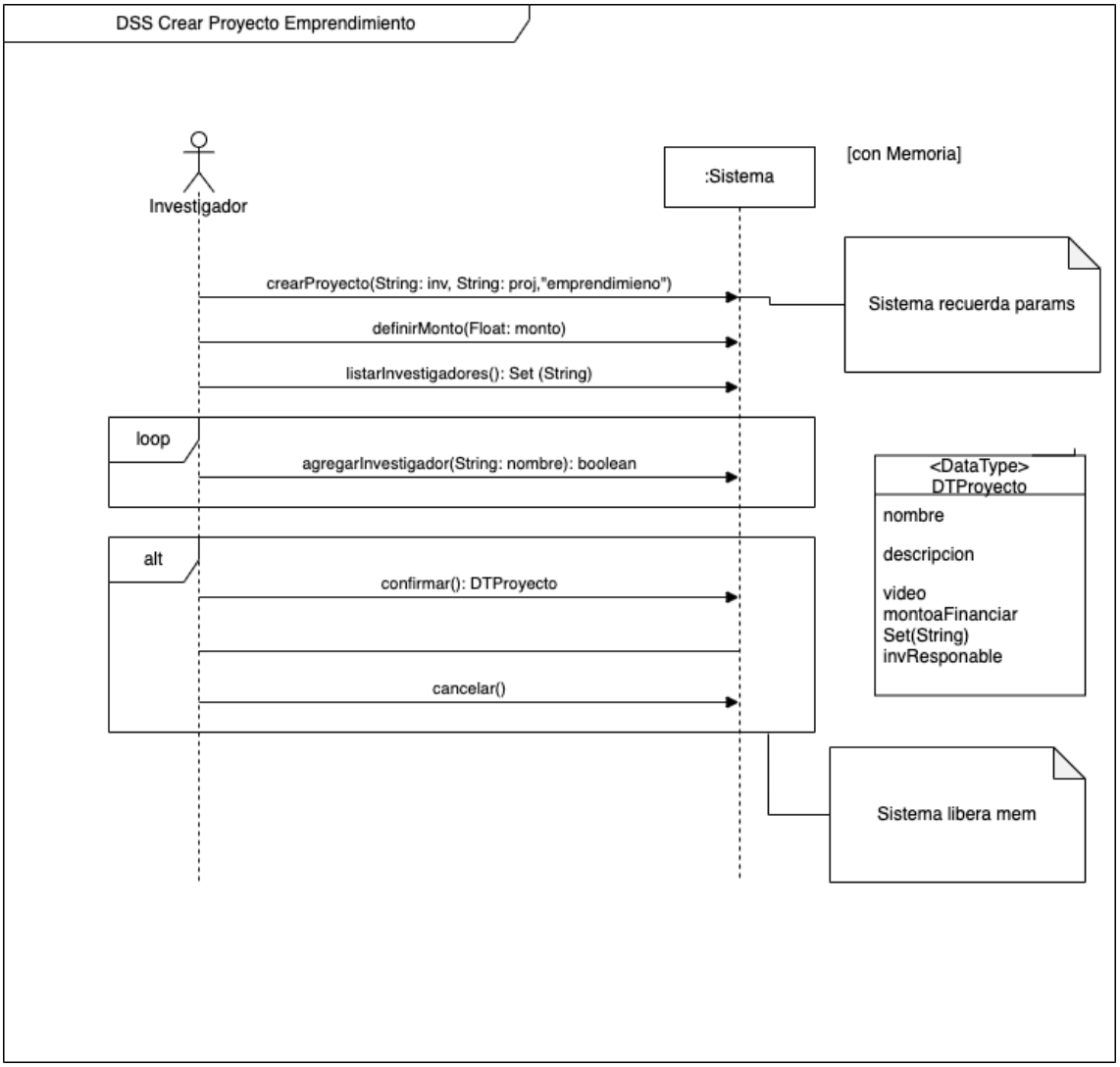
b) Realizar el Modelo de Dominio de la realidad planteada, incluyendo restricciones en lenguaje natural.



Restricciones:

- nombre identifica a Proyecto
- nombre identifica a Usuario
- donaciones son la suma de los montos de las donaciones realizadas para un <emprendimiento,colaborador>
- La suma de monto de donaciones de un proyecto de Emprendimiento no puede ser mayor que el montoAFinanciar.

c) Realizar un Diagrama de Secuencia del Sistema (DSS) para el Caso de Uso “Crear proyecto de emprendimiento” indicando el uso de memoria del Sistema y de datatypes, si corresponde. Utilice memoria para resolver este DSS.

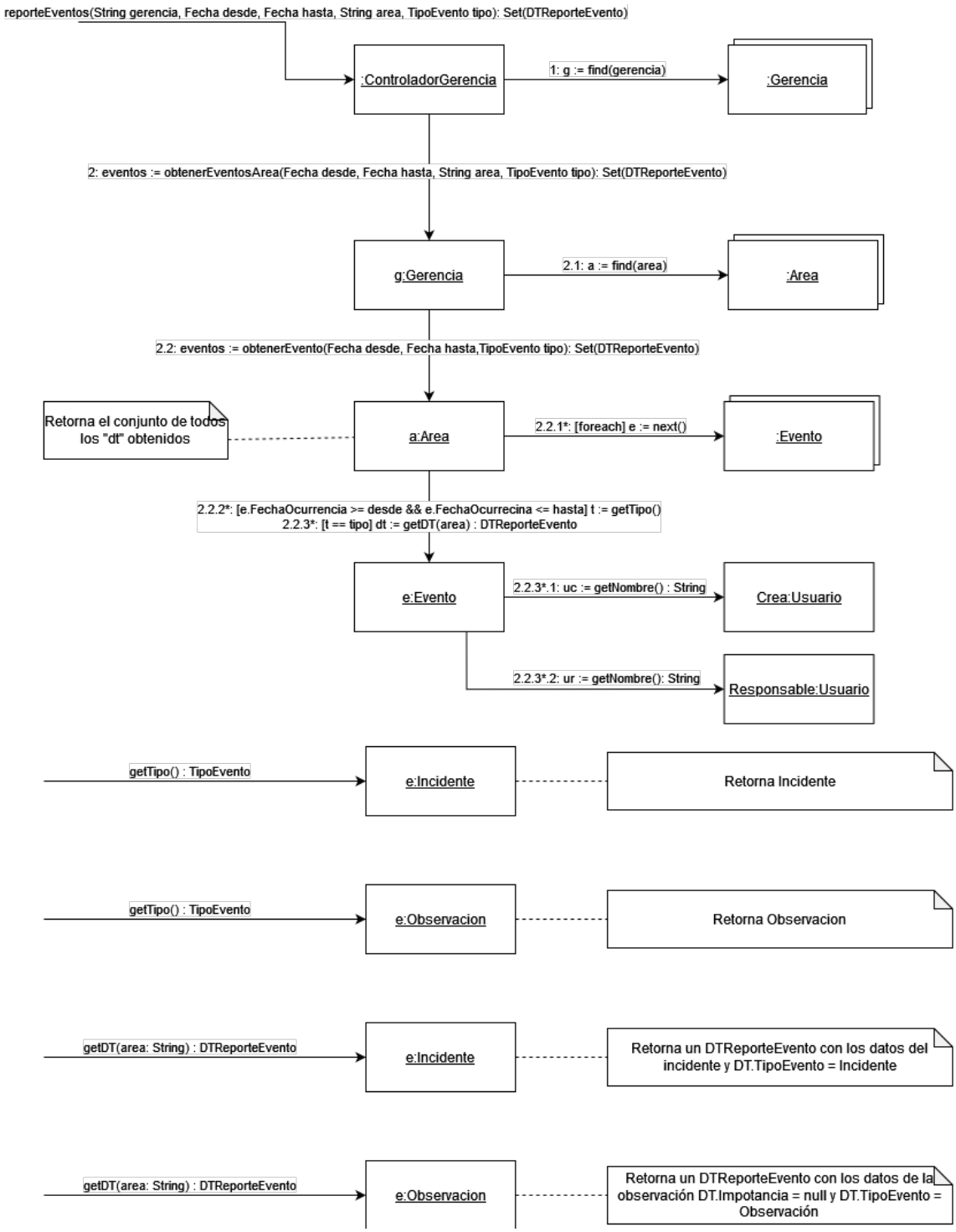


Problema 2 (35 puntos)

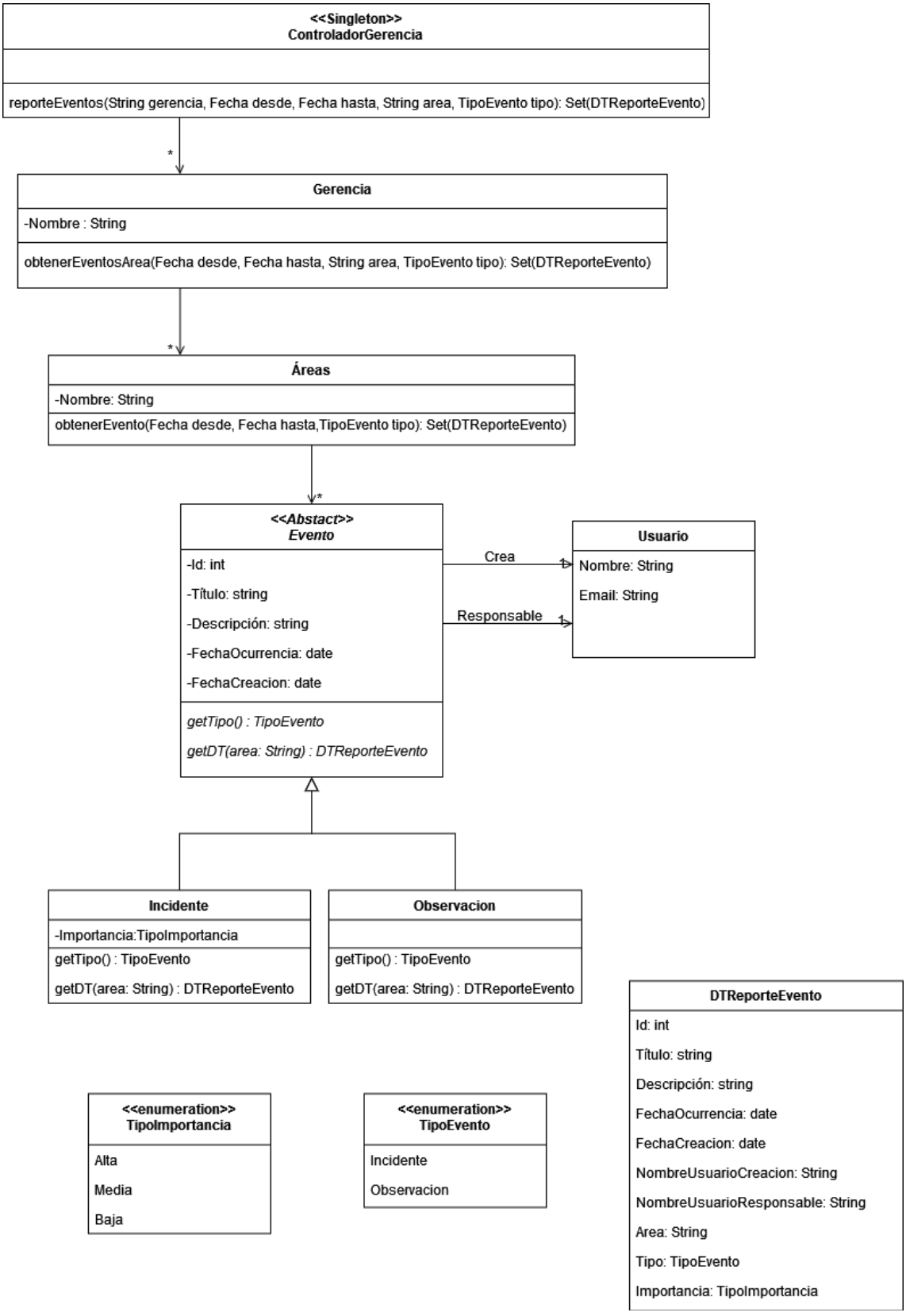
Se desea modelar el funcionamiento de una plataforma que permita la digitalización de eventos físicos sucedidos en una organización.

Se pide:

a) Realizar el Diagrama de Comunicación correspondiente a la operación especificada. Las fechas pueden ser comparadas con los operadores básico (<, >, =, >=, <=, etc.) como cualquier tipo primitivo.



b) Realizar el Diagrama de Clases de Diseño (DCD) resultante del diagrama anterior.



Problema 3 (35 puntos)

El Diagrama de Clases de Diseño de la figura muestra el diseño parcial de un sistema bancario.

Se pide:

a. Implementar en C++ el .h del enumerado `TipoTransacción`.

```
enum TipoTransaccion { Retiro, Deposito };
```

b. Implementar en C++ el .h de la clase `Hora` incluyendo constructor y destructor.

```
class Hora {
private:
    static Hora* instance;
    Hora();
public:
    static Hora* getInstance();
    ~Hora();
    time today();
    int year(time);
};
```

c. Implementar en C++ el .h de las clases `Cuenta` y `CajaAhorro` incluyendo constructor y destructor.

```
class Cuenta
{
private:
    int numero;
    double saldo;
    list<Transaccion *> transacciones;

public:
    Cuenta(int);
    virtual ~Cuenta();
    double deposito(double,Cliente*);
    double retiro(double,Cliente*);
    //Otra solución podría ser que retiro sea polimórfica
    //y que cada tipo de cuenta resuelva todo, pero habría
    //mucho código compartido

    virtual bool checkCuenta() = 0;
};

class CajaAhorro : public Cuenta
{
public:
    int movimientosAnuales;
public:
    CajaAhorro(int,int);
    ~CajaAhorro();
    bool checkCuenta();
};
```

d. Implementar en C++ los métodos de las operaciones `Cuenta::deposito` y `Cuenta::retiro`.

```
double Cuenta::deposito(double mon, Cliente* cl)
{
    saldo += mon;
    Transaccion *t = new Transaccion(Deposito, mon, cl, this);
    transacciones.push_front(t);
    return saldo;
}

double Cuenta::retiro(double mon, Cliente* cl)
{
    bool ok = this->checkCuenta();
    if (not ok)
        throw std::runtime_error("Operación incorrecta");

    if (saldo < mon)
        throw std::runtime_error("Saldo insuficiente");

    saldo -= mon;
    Transaccion *t = new Transaccion(Retiro, mon, cl, this);
    transacciones.push_front(t);
    return saldo;
}
```

e. Implementar en C++ el método de la operación `CajaAhorro::checkCuenta`.

```
bool CajaAhorro::checkCuenta()
{
    list<Transaccion *> trans = getTransacciones();

    Hora *ho = Hora::getInstance();
    time hoy = ho->today();
    int contador = 0;

    //colecto todas las transacciones de éste año
    for (list<Transaccion *>::iterator it = trans.begin();
            it != trans.end(); ++it)
        {
            int anio = ho->year((*it)->getFecha());
            if (anio == ho->year(hoy))
                contador++;
        }

    //controlo que aún pueda hacer movimientos éste año
    if (contador < movimientosAnuales)
        return true;
    else
        return false;
}
```