Programación 4

EXAMEN FEBRERO 2023 - 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)

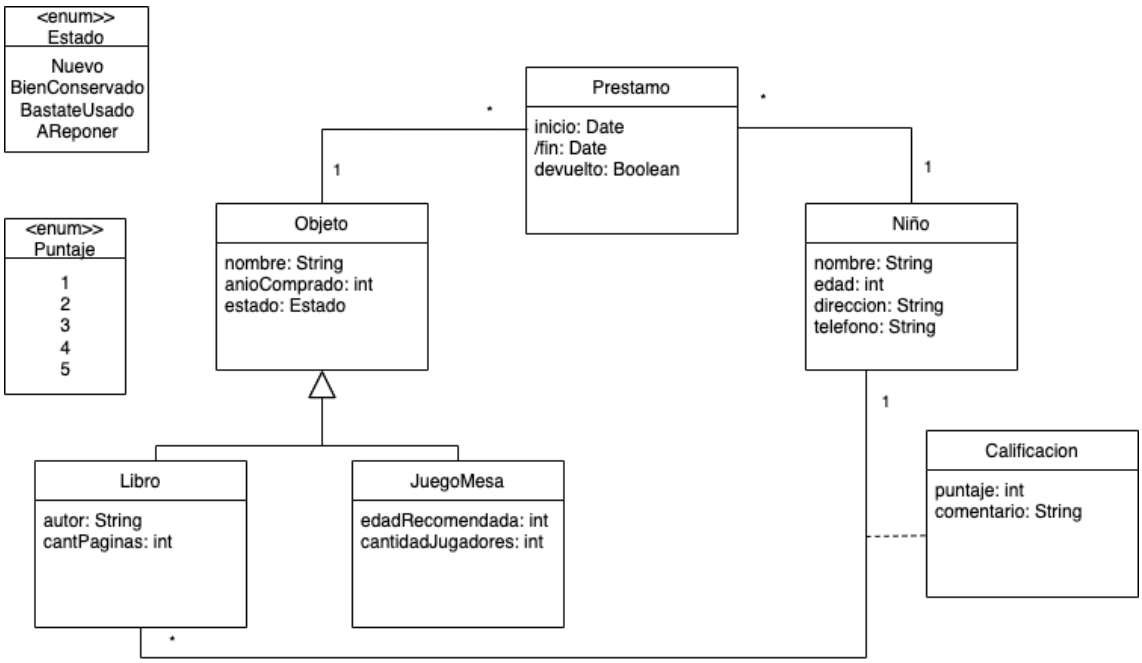
Se quiere modelar un sistema para una biblioteca comunitaria en la cual los niños pueden pedir prestado libros o juegos de mesa.

Se pide:

a) ¿Qué son las restricciones no estructurales o invariantes que a veces se adjuntan al Modelo de Dominio?

Una restricción no estructural, o invariante, es un predicado que expresa una condición sobre los elementos del Modelo de Dominio y que siempre debe ser verdadero. Estas restricciones se expresan por fuera del Modelo de Dominio, por eso su nombre de "no estructurales". Cuando un invariante es evaluado contra una cierta configuración de objetos y devuelve falso, significa que la configuración de objetos no es válida.

b) Realizar el Modelo de Dominio de la realidad planteada, incluyendo todas las restricciones que considere necesarias en lenguaje natural.



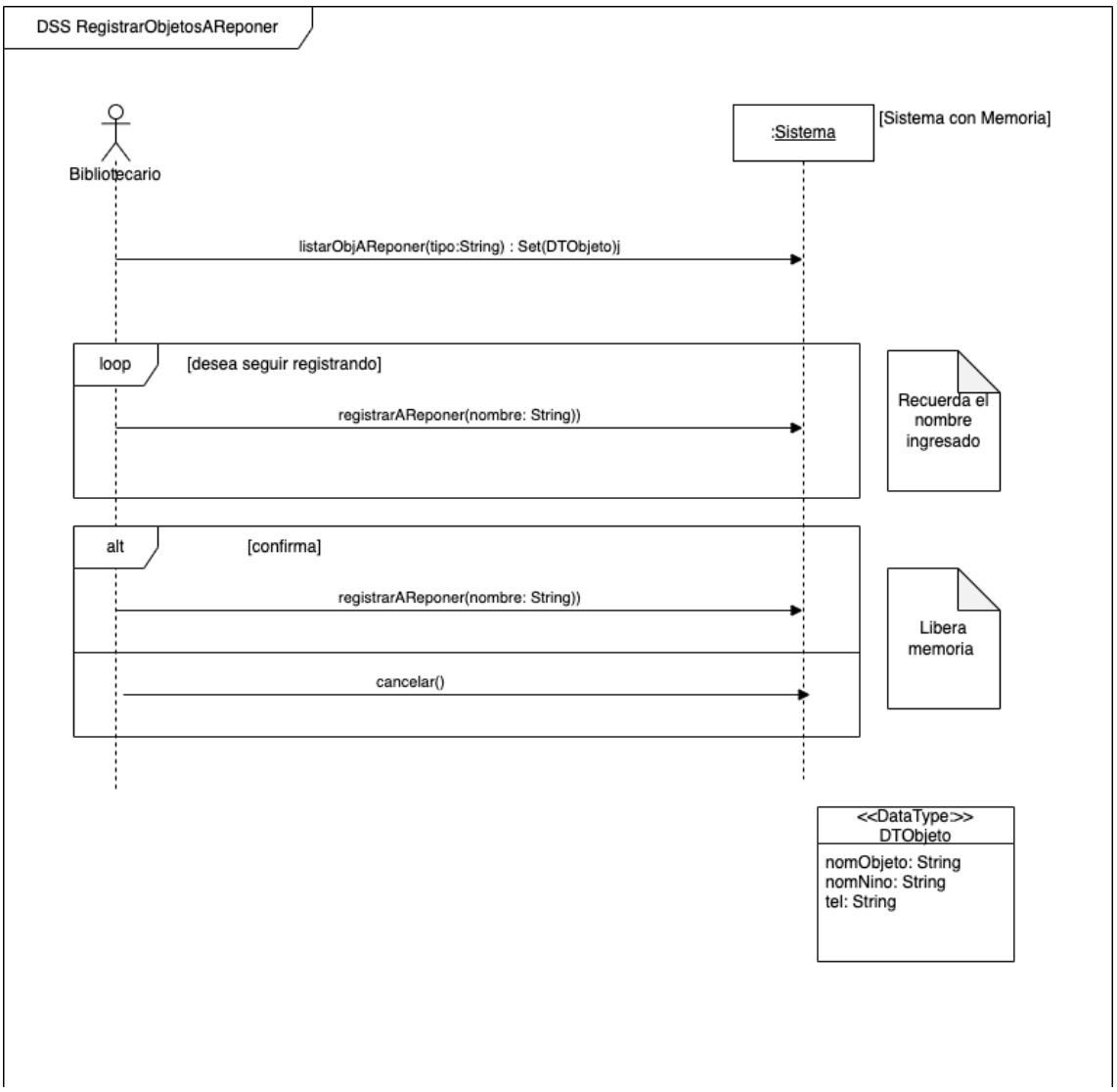
Restricciones:

Identidad
- Nombre de Objeto es único.
- Nombre de Niño es único.

Integridad Circular
- Una instancia de Niño tiene un link “Calificación” con una instancia de Libro, solo si tiene también un link “Prestamo” con la misma instancia de Libro.

Regla de Negocio
- Una instancia de Niño tiene a lo sumo tres links “Prestamo” con el atributo Devuelto=False.
- El atributo fecha de fin de un Prestamo corresponde al valor del atributo fecha de inicio más 15 días.

c) Realizar un Diagrama de Secuencia del Sistema (DSS) para el Caso de Uso “Registrar objeto a reponer”. Indique el uso de memoria del Sistema y de datatypes, si corresponde. Utilice memoria para resolver este DSS.

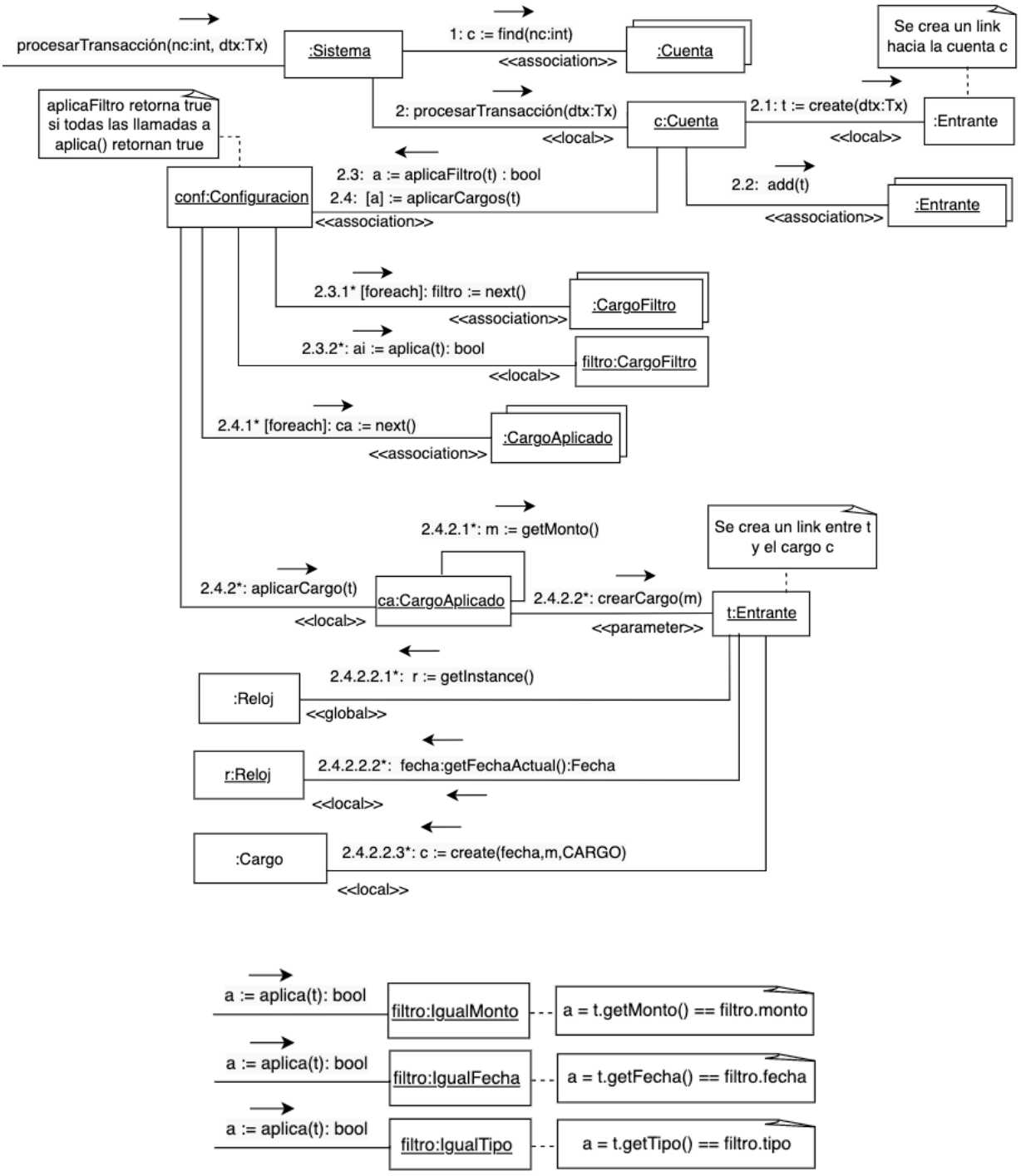


Problema 2 (35 puntos)

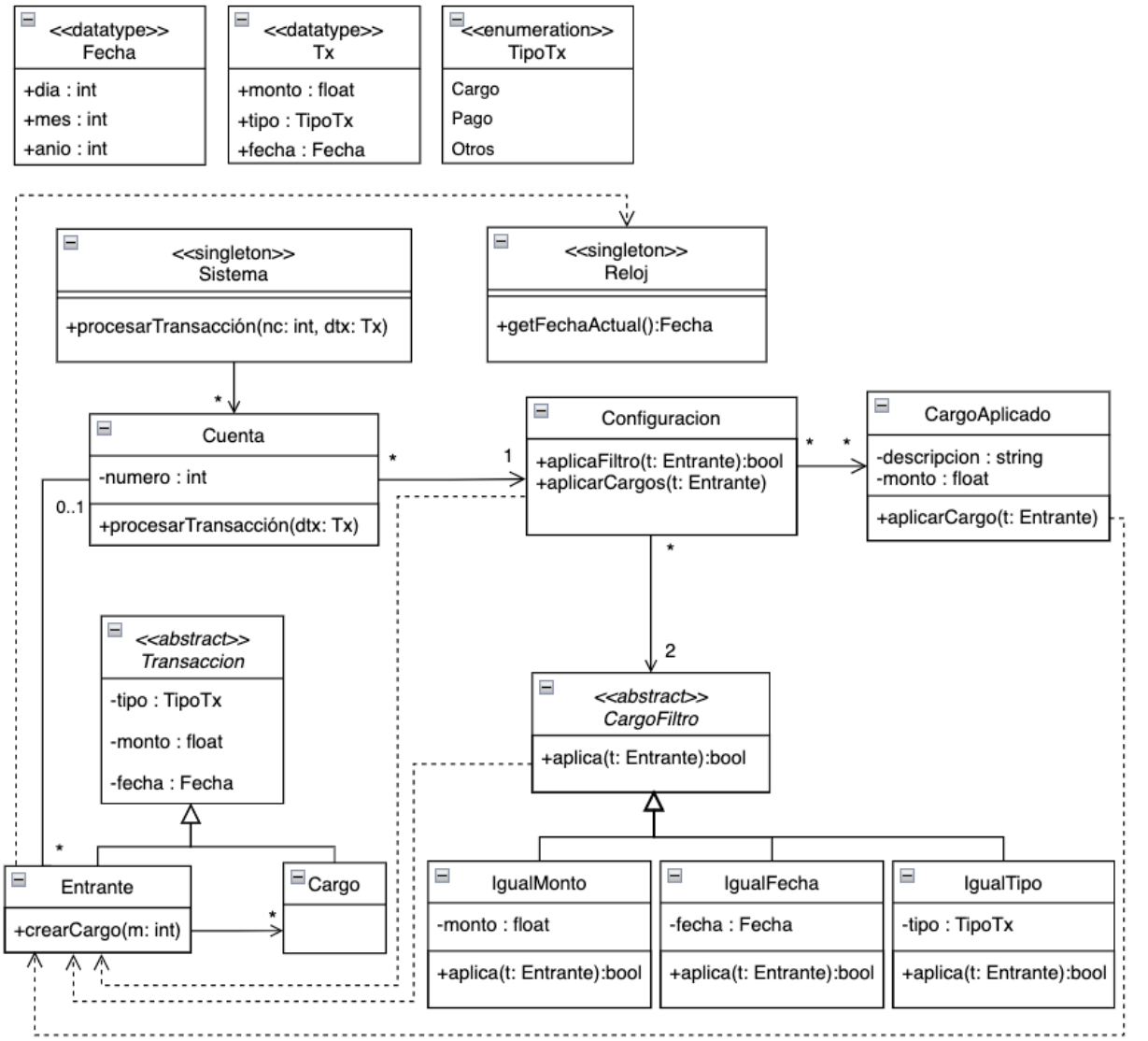
Se desea implementar un sistema de procesamiento de transacciones asociadas a cuentas bancarias.

Se pide:

a) Realizar el Diagrama de Comunicación correspondiente a la operación del sistema descrita, incluyendo visibilidades.



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



Problema 3 (35 puntos)

El Diagrama de Clases de Diseño de la figura muestra el diseño parcial de un sistema de control remoto y de su interfaz con dispositivos electrónicos.

Se pide:

a) Implementar en C++ el .h de las clases `Control`.

```
class Control
{
private:
    Dispositivo * dispositivo;

public:
    Control();
    ~Control();

    void bajarVolumen();
    Dispositivo * getDispositivo();
};
```

b) Implementar en C++ el .h de la interfaz `Dispositivo`.

```
class Dispositivo
{
public:
    virtual ~Dispositivo();

    virtual void apagar() = 0;
    virtual bool estaPrendido() = 0;
    virtual int obtenerVolumen() = 0;
    virtual void bajarVolumen(int) = 0;
};
```

c) Implementar en C++ el .h de la clase `TV`.

```
class TV : public Dispositivo
{
private:
    bool encendido;
    int volumen;
    list<Parlante *> parlantes;

public:
    TV();
    ~TV();

    void apagar();
    bool estaPrendido();
    int obtenerVolumen();
    void bajarVolumen(int);
};
```

d) Implementar en C++ el método de la operación `ControlAvanzado::silenciar`.

```
void ControlAvanzado::silenciar()
{
    Dispositivo * dis = getDispositivo();

    if (dis->estaPrendido())
    {
        int volumen = dis->obtenerVolumen();
        dis->bajarVolumen(volumen);
    }
}
```

e) Implementar en C++ el método de la operación `TV::bajarVolumen`.

```
void TV::bajarVolumen(int vol)
{
    list<Parlante *>::iterator it;

    if (volumen >= vol)
    {
        volumen -= vol;

        for (it = parlantes.begin(); it != parlantes.end(); ++it)
            (* it)->bajarVolumen(vol);
    }
    else{
        volumen = 0;
        apagar();

        for (it = parlantes.begin(); it != parlantes.end(); ++it){
            try
            {
                (* it)->standby();
            }
            catch(const std::exception& e)
            {
                (* it)->apagar();
            }
        }
    }
}
```