Programación 4

EXAMEN FEBRERO 2023

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. La biblioteca podrá prestar distintos tipos de objetos y comenzará, como se dijo antes, prestando libros y juegos de mesa. De cada objeto se conoce su nombre (que lo identifica), el año en el que se compró, su estado (esto es: nuevo, bien conservado, bastante usado, a reponer). Además, de los libros se conoce su autor y cantidad de páginas. Mientras que de los juegos de mesa se conoce la edad a partir de la cual se recomienda su uso y la cantidad de jugadores mínimos. De los niños se conoce su nombre (que lo identifica), edad, su dirección y un número de teléfono de contacto. Los niños pueden pedir prestados objetos y de cada prestamos se conoce la fecha de inicio y la fecha límite del préstamo (esto es, 15 días después de iniciado el préstamo). Cada niño puede tener a préstamo hasta tres objetos, sin devolver, simultáneamente. Un niño también puede calificar, por única vez, los libros que la biblioteca le haya prestado en algún momento, para eso puede ingresar un puntaje del 1 al 5 y un comentario corto.

Además, se cuenta con la descripción del siguiente Caso de Uso:

Caso de Uso	Registrar objetos a reponer
Actor	Bibliotecario
Descripción	El caso de uso comienza cuando un bibliotecario desea registrar objetos para que sean repuestos debido a que están en mal estado. Para esto ingresa el tipo de los objetos que registrará (libros o juegos). El sistema muestra la lista con los nombres de todos los objetos que tienen estado distinto que “a reponer” del tipo seleccionado antes. El bibliotecario ingresa de a uno a la vez los nombres de los objetos que desea marcar para su reposición. Cuando el bibliotecario termina de registrar nombres de objetos el sistema solicita que confirme o cancele el registro. En caso de que el bibliotecario acepte el sistema almacena la información y lista los nombres de los objetos registrados, indicando también si algún objeto está prestado el nombre del niño que lo tiene y su número de teléfono.

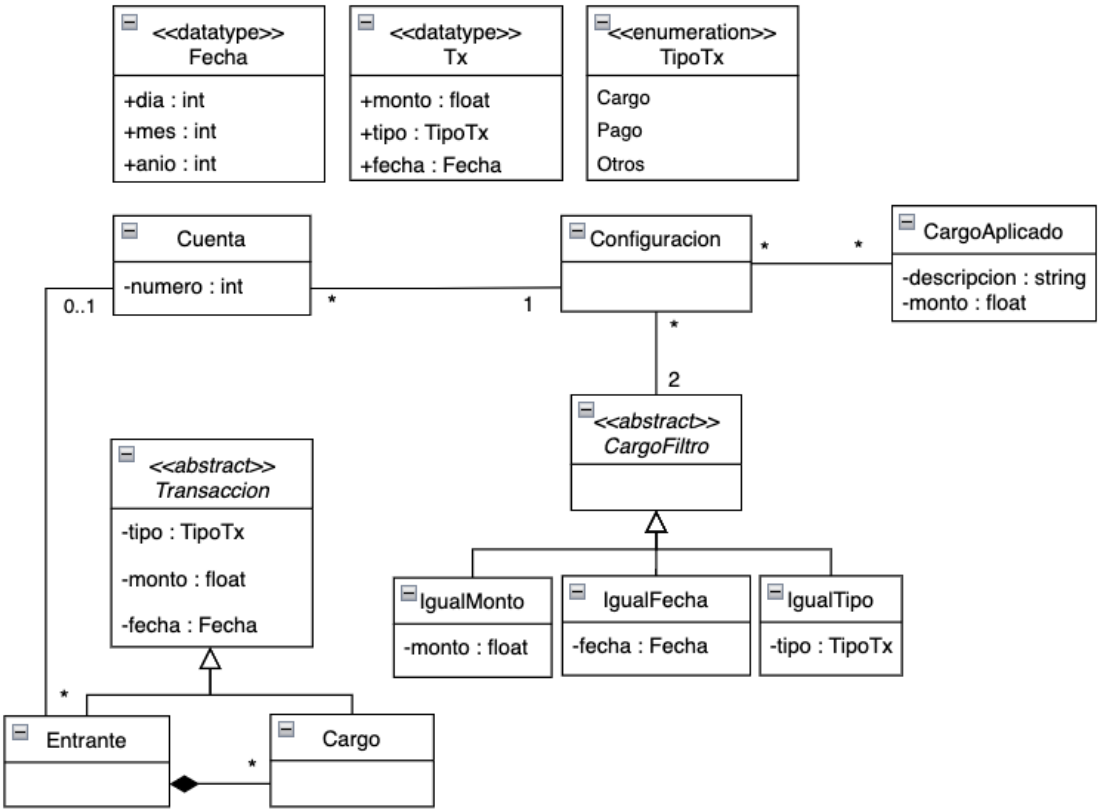
Se pide:

a) ¿Qué son las restricciones no estructurales o invariantes que a veces se adjuntan al Modelo de Dominio?
b) Realizar el Modelo de Dominio de la realidad planteada, incluyendo todas las restricciones que considere necesarias en lenguaje natural.
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. El sistema recibe las transacciones entrantes pertenecientes a cada cuenta (identificada por un número) y realiza el procesamiento de los cargos asociados, es decir, determina si corresponde cobrar cierto monto dependiendo de los atributos de una transacción. Para determinar si corresponde la aplicación de los cargos, existe para cada cuenta una configuración asociada a un par de filtros (`CargoFiltro`) que evalúan atributos de la transacción. Por ejemplo, si uno de los filtros que se encuentra asociado es el filtro `IgualMonto`, se debe verificar que el monto de la transacción coincida con el monto del filtro. Para poder aplicar los cargos, todos los filtros deben ser exitosos. Esto quiere decir que, con fallar al menos uno, no se deben aplicar los cargos. La configuración de cada cuenta tiene también asociados todos los cargos que pueden ser aplicados. Por cada cargo que se aplique sobre una transacción, se genera una nueva transacción de tipo `Cargo` (`TipoTx`) que se asocia a la transacción entrante, por ejemplo, de tipo Pago.

Considere el siguiente modelo de dominio para el sistema descrito anteriormente.



Se cuenta también con el siguiente contrato.

`procesarTransaccion(nc : int, dtx : Tx)`	
Descripción	Se procesa una nueva transacción que ingresa en el sistema.
Parámetros	`nc`: Identificador de la cuenta asociada a la transacción. `dtx`: Datos de la transacción.
Precondiciones	• Existe en el sistema una Cuenta identificada con el numero "`nc`"
Postcondiciones	• Se crea una instancia de `Entrante` "`e`" con los datos indicados en `dtx`. • Se crea un link entre la instancia de Entrante "`e`" y la instancia de Cuenta identificada con el numero "`nc`".

	• En caso de corresponder aplicar los cargos (si el filtro fue exitoso), por cada instancia de `CargoAplicado` asociado a la `Cuenta`, se crea una instancia de `Cargo` con tipo "`Cargo`", monto el monto calculado con los datos de la instancia `CargoAplicado` y con fecha la fecha actual del sistema. • Se crea un link entre la instancia de Entrante "`e`" y cada instancia `Cargo` creada

Se pide:

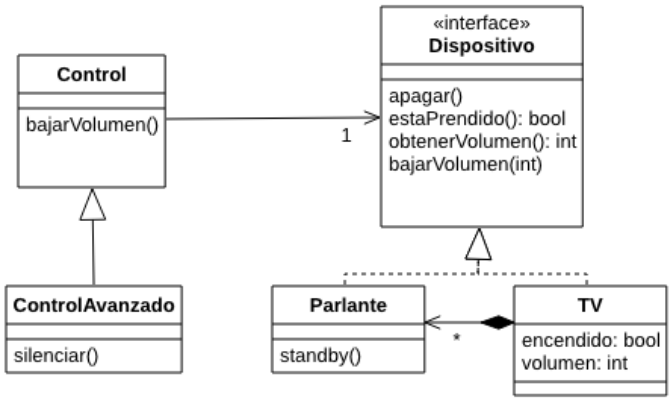
a) Realizar el Diagrama de Comunicación correspondiente a la operación del sistema descrita, incluyendo visibilidades.
b) Realizar el Diagrama de Clases de Diseño (DCD) resultante.

Observaciones:

- La clase Singleton `Reloj` cuenta la operación `getFechaActual():Fecha` que retorna la fecha actual del sistema.

<u>Problema 3</u> (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. Un `Control` declara una referencia que lo vincula con un `Dispositivo`, lo que se define como una interfaz para permitir utilizar el mismo control para diferentes tipos de dispositivos. El control permite bajar el volumen de a un nivel a la vez, en tanto los dispositivos ofrecen funciones más básicas que permiten manipularlos, por ejemplo, apagarlos, consultar si están prendidos, obtener su nivel de volumen y bajar su volumen de a varios niveles a la vez. Se define un `ControlAvanzado` que tiene una funcionalidad especial de silenciar un dispositivo encendido. Existen dos tipos de dispositivos: `Parlante`, que puede ponerse en pausa en lugar de apagarse completamente, y `TV`, que se puede conectar con múltiples parlantes.



A continuación, se describen el comportamiento de las principales operaciones definidas.

`void ControlAvanzado::silenciar();`
Si el dispositivo vinculado se encuentra encendido, reduce a cero su volúmen.

`void Dispositivo::apagar();`
Apaga el dispositivo.

`bool Dispositivo::estaPrendido();`
Indica si el dispositivo se encuentra encendido o no. En el caso de una **`TV`**, el atributo **`encendido`**.

`int Dispositivo::obtenerVolumen();`
Retorna el nivel de volúmen que tiene un dispositivo. En el caso de una **`TV`**, el atributo **`volumen`**.

`void Dispositivo::bajarVolumen(int vol);`
Reduce el nivel de volumen tantos niveles como indica el parámetro **`vol`**. En el caso de una **`TV`**, si **`vol`** supera el nivel de volumen actual, se apaga. Además, en ese mismo caso, intenta poner a todos los parlantes en pausa y, si no puede pausar alguno, también lo apaga. El volumen actual de la TV y sus parlantes, es siempre el mismo.

`void Parlante::standby();`
Pone en pausa el parlante. En caso de que se genere algún problema con la pausa, lanza una excepción **`std::exception`**.

Se pide:

a) Implementar en C++ el .h de las clases `Control`.
b) Implementar en C++ el .h de la interfaz `Dispositivo`.
c) Implementar en C++ el .h de la clase `TV`.
d) Implementar en C++ el método de la operación `ControlAvanzado::silenciar`.
e) Implementar en C++ el método de la operación `TV::bajarVolumen`.

Observaciones:

- NO se pueden agregar más operaciones de las definidas en el diseño anterior.
- Puede utilizar colecciones genéricas (realizaciones de `IDictionary` e `ICollection`) o paramétricas (contenedores STL).
- No incluir directivas al precompilador (#include, etc).
- No es necesario implementar setters y getters adicionales de las clases, pero puede asumir su existencia si se requieren en algún método.
- La implementación debe hacer un correcto manejo de memoria.