Programación 4

EXAMEN FEBRERO 2021

Debido a las restricciones impuestas por la pandemia COVID-19, el examen se realizó en modalidad virtual, según las pautas del documento anexo.

Problema 1 (35 puntos)

Se desea desarrollar siguiendo un proceso iterativo e incremental, un sistema de apoyo a la gestión de una biblioteca barrial.

Para empezar a funcionar se generó un stock básico compuesto por libros donados por los/as vecinos/as. A efectos de control, se desea que el sistema lleve registro de qué vecino/a donó cada libro y la fecha en la que lo hizo. De cada libro se conoce un número de inventario que lo identifica, su título, quien lo escribió, y la fecha de publicación. Además, cada libro pertenece a una categoría, por ejemplo: estudio, ficción, drama, poesía, etc. Para los/as vecinos/as se desea saber su nombre completo, cédula de identidad (que le identifica) y un número de teléfono.

Se pide:

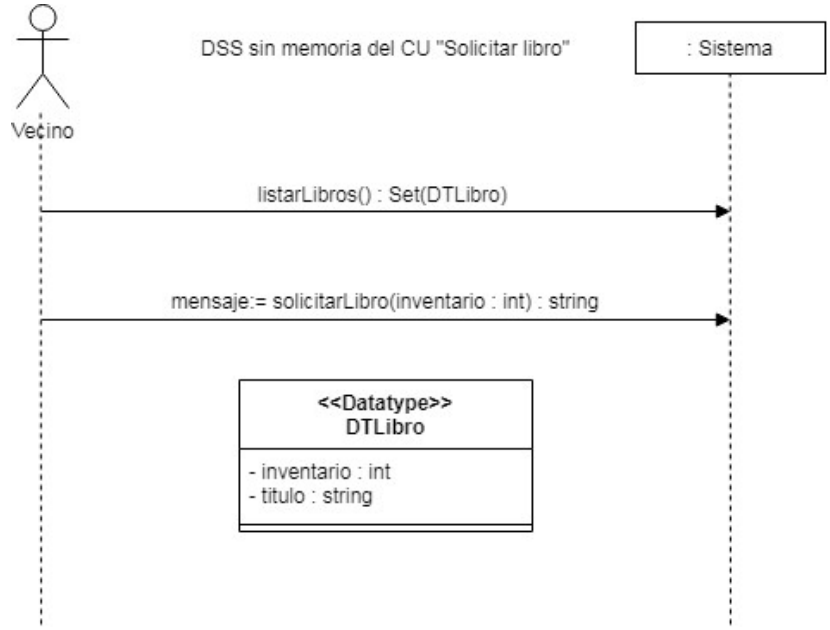
a) Realizar el Modelo de Dominio de la realidad planteada **sin incluir** restricciones no estructurales.

b) Realizar el Diagrama de Secuencia del Sistema (DSS) para el Caso de Uso "Alta de vecinos", indicando el uso de memoria del Sistema así como de Datatypes.

Caso de Uso	Alta de vecinos
Actor	Administrador
Descripción	El caso de uso comienza cuando un administrador desea agregar vecinos en el sistema. Para cada vecino/a, el sistema pide su nombre completo, cédula de identidad y un número de teléfono. Luego se da la opción de confirmar o cancelar; en caso de confirmar se da de alta el/la vecino/a con los datos ingresados. Finalmente, el sistema pregunta si desea seguir agregando o terminar el caso de uso.

En una siguiente fase se le presenta al equipo de desarrollo la descripción del CU "Solicitar libro" y su DSS.

Caso de Uso	Solicitar libro
Actor	Vecino
Descripción	El caso de uso comienza cuando un/a vecino/a desea solicitar un libro. Para ello el sistema lista todos los libros existentes (mostrando título y número de inventario) y el usuario selecciona uno de ellos. Luego, en caso de que el libro requerido se encuentre disponible se dará de alta un nuevo préstamo con fecha actual. Un libro se encuentra disponible si no existe un préstamo para él en el sistema. Luego de que un/a vecino/a devuelve un libro se elimina el préstamo. Si en cambio no está disponible el libro, se agregará el/la vecino/a a una lista de espera del mismo, y permanecerá en esta hasta que esté disponible. En cualquiera de los casos el sistema comunica un mensaje acorde.



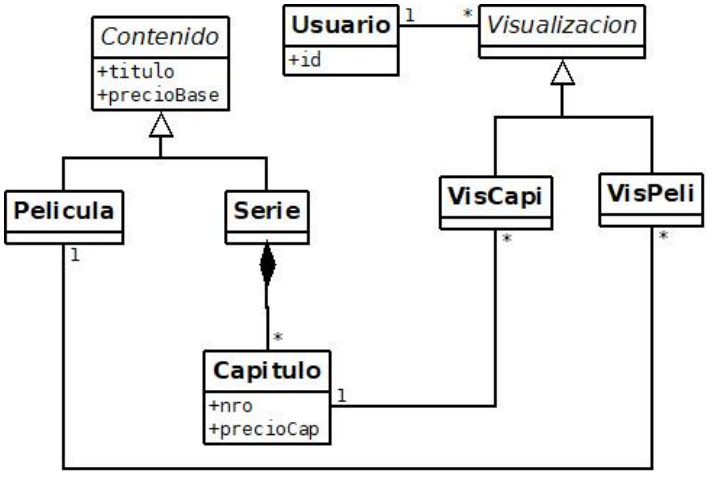
Se pide:

c) Realizar un nuevo Modelo de Dominio incluyendo, si corresponde, restricciones en lenguaje natural.

d) Escribir las pre y post condiciones de los contratos de todas las Operaciones del Sistema del DSS del CU "Solicitar libro".

Problema 2 (30 puntos)

Se desea modelar el funcionamiento de una plataforma online que ofrece películas y series, en la cual el pago se hace por visualización, en lugar de suscripción. Las series están compuestas por varios capítulos. Cada contenido (ya sea película o serie) tiene un precio base de visualización. En caso de ser una película, el costo de la visualización corresponde al precio base del contenido completo. En el caso de series, el costo de la visualización es la suma del precio del capítulo más el precio base de la serie (para cada capítulo diferente que se visualice de una misma serie, se debe sumar su precio base). Se asume que un usuario puede visualizar más de una vez una misma película o un mismo capítulo de una serie, debiendo pagar por cada vez. Para llevar un registro de la visualización de contenidos por parte de los usuarios, se realizó el siguiente Modelo de Dominio.



Se cuenta con los contratos de las siguientes operaciones del sistema:

agregarVisualizacion(idU : Integer; dv : DataVisualizacion)	
Descripción:	Agrega al sistema un registro de visualización de un contenido por parte de un usuario
Parámetros:	idU: Identificador del usuario dv: Información de la visualización
Precondiciones:	Existe en el sistema un Usuario identificado por idU Existe en el sistema un Contenido identificado por dv.getTitulo() Si dv.getTipo()=Serie, existe en el sistema un Capitulo de número igual a dv.getNroCap(), asociado a una Serie identificada por dv.getTitulo()
Poscondiciones:	Si dv.getTipo()=Pelicula, se crea una instancia vp de VisPeli y se crean un link entre vp y el Usuario identificado por idU y otro entre vp y la Película identificada por dv.getTitulo() Si dv.getTipo()=Serie, se crea una instancia vc de VisCapi y se crean un link entre vc y el usuario identificado por idU y otro entre vc y el Capitulo de número dv.getNroCap() asociado a la Serie identificada por dv.getTitulo()

calcularPago(idU : Integer) : Real	
Descripción:	Calcula el monto a pagar por concepto de visualizaciones de contenidos por parte de un usuario
Parámetros:	idU: Identificador del usuario
Precondiciones:	Existe en el sistema un Usuario identificado por idU
Poscondiciones:	El resultado corresponde a la suma de los costos de todas las visualizaciones del Usuario identificado por idU, donde el costo de una visualizacón se calcula según la descripción del primer párrafo del ejercicio.

Se pide:

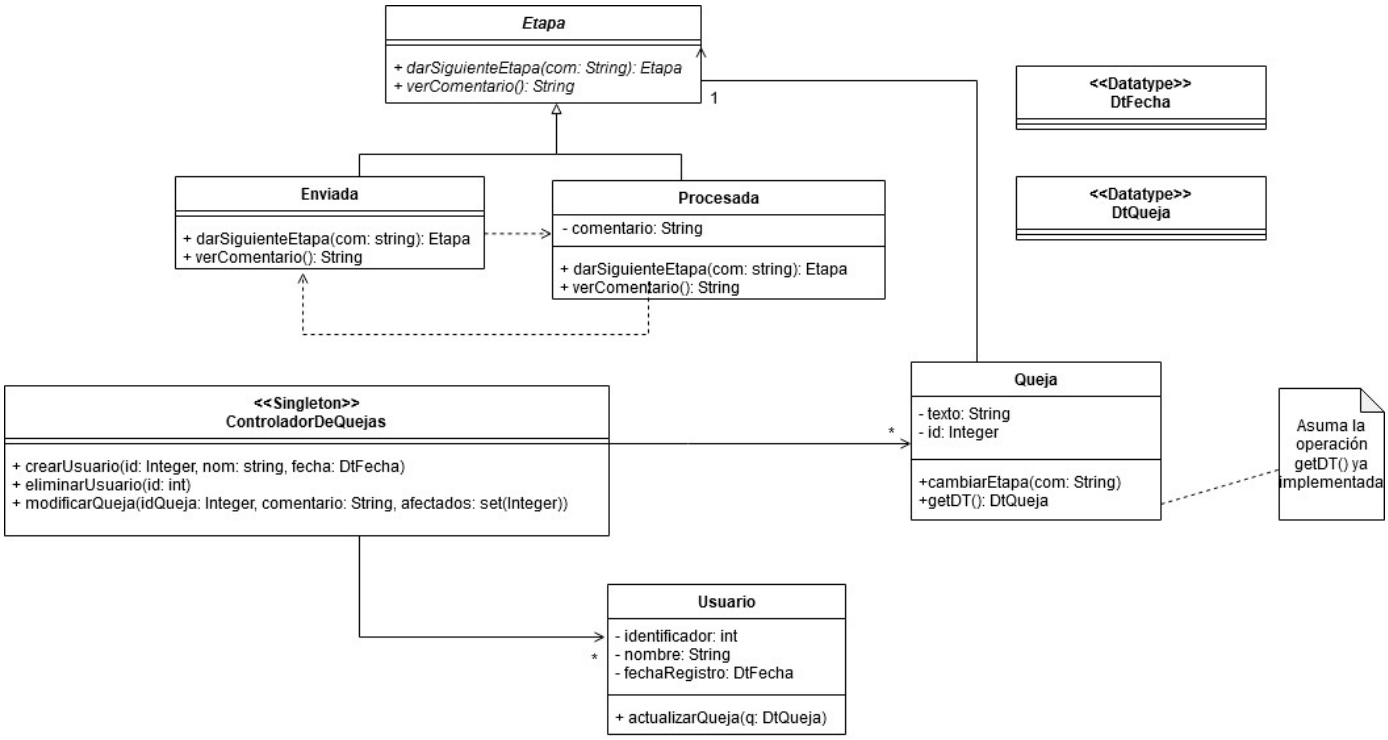
a) Realizar los Diagramas de Comunicación de las operaciones agregarVisualizacion y calcularPago, aplicando los criterios GRASP.

b) Realizar el Diagrama de Clases de Diseño (DCD) resultante, incluyendo controladores, interfases y datatypes.

Nota: No incorporar setters ni getters al DCD. Incorporar en el DCD sólo los constructores o destructores que se utilicen en los diagramas de comunicación. No incorporar al DCD las operaciones de colecciones.

Problema 3 (35 puntos)

Se desea desarrollar un sistema para la gestión de quejas en una empresa por parte de los usuarios. El sistema permite crear y eliminar usuarios. Una queja cuenta con dos etapas: inicialmente al ser creada comienza como enviada y posteriormente cuando un empleado la revisa pasa a estar procesada con un comentario que responde a la misma. En caso de que haya una apelación, la queja podría volver a la etapa anterior. Los usuarios guardan información local de las quejas en las que han participado y esta debe mantenerse actualizada. El siguiente Diagrama de Clases modela este sistema.



El comportamiento de las operaciones se describe a continuación:

1. ControladorDeQuejas
   - crearUsuario(id: Integer, nom: String, fecha: DtFecha): crea un usuario con identificador *id*, nombre *nom* y fecha de registro *fecha*.
   - eliminarUsuario(id: Integer): elimina el usuario con el identificador *id* del sistema.
   - modificarQueja(idQueja: Integer, comentario: string, afectados set(Integer)): modifica la etapa en la que se encuentra la queja agregando el comentario *com*. Actualiza la información local de la queja que mantienen los usuarios cuyo identificador se encuentra en *afectados*.
2. Usuario
   - actualizarQueja(q: DtQueja): Actualiza la información local que el usuario mantiene sobre la queja.
3. Queja
   - cambiarEtapa(com: String): actualiza la etapa en la que se encuentra la queja con el comentario *com*.
   - GetDT(): Retorna un DtQueja con todos los datos de la queja. No es necesario implementarla

4. Etapa
    - darSiguienteEtapa(com: String):
        1. Enviada: Retorna una instancia de la clase Procesada. El parámetro *com* es ignorado.
        2. Procesada: Retorna una instancia de la clase Enviada, *com* es guardado en el atributo comentario.
    - verComentario():
        1. Enviada: Retorna el mensaje genérico "Enviada: comentario en breve".
        2. Procesada: Retorna el comentario dejado al actualizar la etapa de la queja.

Se pide:

a) Analice el diagrama y fundamente si se ha(n) aplicado algún(os) patrón(es) de diseño. En caso de que sí identifique cuál, explique el problema que resuelve y qué clase cumple qué rol.

b) Implemente los siguientes archivos.

H	CPP
ControladorDeQuejas	ControladorDeQuejas
Etapa	Etapa
Enviada	Enviada
Queja	Queja (recuerde que getDT() se puede asumir implementada)

Dispone de Usuario.h ya implementada

```
//Usuario.h
class Usuario {
  private:
    int id;
    string nombre;
    DtFecha fechaRegistro;
    set<DtQueja> misQuejas;
  public:
    Usuario(int, string, DtFecha&);
    actualizarQueja(DtQueja&);
}
```

Considere:

- Es posible utilizar map, set y vector de la librería STL.
- Puede suponer la existencia de la interface ICollectible e implementaciones de las interfases ICollection, IKey, IDictionary e IIterator según sea necesario.
- **No** es necesario implementar setters y getters de las clases pero **sí** debe implementar constructores y destructores.
- **No** es necesario incluir directivas al precompilador.
- Los datatypes DtFecha y DtQueja se asumen ya implementados.
- Su implementación debe hacer un correcto manejo de memoria.