Programación 4

EXAMEN JULIO 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 desea crear un sistema para registrar las consultas de mascotas de una veterinaria. Las mascotas se registran con un número de socio (único), nombre y edad. Además, para los perros y gatos se conoce su peso en kilogramos y para las serpientes su largo en centímetros. Cada mascota está asociada a uno o más dueños, de los que se conoce su nombre (que los identifica), su dirección y su número de teléfono. La veterinaria ofrece planes de membresía que tiene un nombre (único) y un costo. Cada mascota tiene una membresía única a un plan de la veterinaria. De esta membresía se conoce la fecha de inicio.

Una consulta en la veterinaria surge en el marco de la membresía de una mascota (no se registran consultas de mascotas que no tienen una membresía). Las consultas tienen una fecha y un texto que indica el motivo de la consulta y son realizadas por uno de los dueños de la mascota atendida. Cada consulta puede derivar en uno o más tratamientos (que pueden ser medicamentos o internaciones), los cuales incluyen un texto con indicaciones. En caso de que el tratamiento sea un medicamento, se incluye su dosificación, mientras que las internaciones incluyen el número de cama en la cual la mascota estará internada.

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

Caso de Uso	Registrar consulta
Actor	Usuario
Descripción	El caso de uso comienza cuando un usuario desea registrar una consulta de una mascota. Primero, si la mascota no está registrada en el sistema, la registra junto con su dueño. Para eso, indica los datos del dueño, nombre de la mascota, edad y tipo (perro, gato o serpiente), a lo que el Sistema devuelve un nuevo número de socio. Si la mascota ya está registrada, el usuario sólo ingresa el número de socio correspondiente. Si la mascota, por cualquier razón, no tiene una membresía activa, el usuario debe registrarle una. Para eso, el Sistema informa los planes disponibles y el usuario indica el nombre del plan deseado, a lo que el Sistema registra una nueva membresía en la fecha actual. Finalmente, el usuario indica el nombre del dueño que trae a la mascota a la consulta y el texto indicando el motivo de la consulta, a lo que el Sistema registra la consulta en la fecha actual.

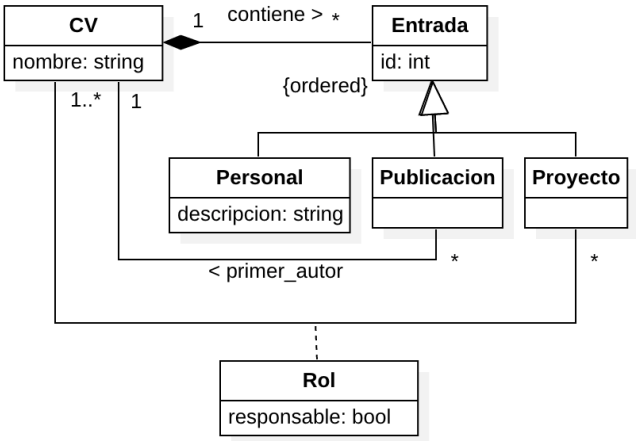
Se pide:

a) Realizar el Modelo de Dominio de la realidad planteada, incluyendo todas las restricciones que considere necesarias en lenguaje natural.
b) Realizar un Diagrama de Secuencia del Sistema (DSS) para el Caso de Uso. Indique el uso de memoria del Sistema y de datatypes, si corresponde.

Problema 2 (35 puntos)

Se está desarrollando un sistema de gestión de currículum para investigadores. Cada currículum (CV) se identifica por el nombre del investigador y está compuesto por un conjunto de entradas que se ordenan según un identificador único dentro del CV. Las entradas del CV pueden ser una descripción de información personal, información sobre una publicación científica o de un proyecto en las cuales participe el investigador. Para cada publicación se indica su primer autor. Por su parte, para un proyecto se conoce la lista de participantes, entre los cuales hay un solo responsable, indicado por el atributo correspondiente. Un investigador se dice *principal* en una publicación si es el primer autor, y en un proyecto si es su responsable. Además, un investigador es siempre el principal en sus propias entradas personales.

Considere el siguiente modelo de dominio parcial y contratos para el sistema descrito.



`crearCV(nom : string, desc: string)`	
Descripción	Crea un CV para un investigador con una descripción personal inicial.
Parámetros	`nom`: nombre del investigador `desc`: descripción personal inicial de investigador
Precondiciones	• No existe una instancia `c:CV` con `c.nombre = nom`
Postcondiciones	• Se crea una instancia `c:CV` con `c.nombre = nom` • Se crea una instancia `p:Personal` con `p.descripcion = desc` y `p.id = 1` • Se crea un link `contiene` entre `c` y `p`

`consultarPrincipales(nom : string):Set(DTR)`	
Descripción	Consulta los investigadores principales de las entradas de un CV.
Parámetros	`nom`: nombre del investigador
Precondiciones	• Existe una instancia `c:CV` con `c.nombre = nom`
Postcondiciones	• Para cada instancia `pe:Personal` con un link `contiene` con `c`, se genera un datavalue `dt:DTR` con `dt.id = pe.id`, `dt.tipo = Personal` y `dt.principal = c.nombre`. • Para cada instancia `pu:Publicacion` con un link `contiene` con `c` y con un link `primer_autor` con una instancia `cv:CV`, se genera un datavalue `dt:DTR` con `dt.id = pu.id`, `dt.tipo = Publicacion` y `dt.principal= cv.nombre`. • Para cada instancia `pr:Proyecto` con un link `contiene` con `c` y con un link a un `r1:Rol` tal que `r1.responsable = true`, se genera un datavalue `dt:DTR` con `dt.id = pr.id`, `dt.tipo = Proyecto` y

	`dt.principal= r1.cv.nombre.` • Se retorna el conjunto de `dt` generados.

Nota: Asumir que el Sistema posee una colección global de CV.

Se pide:

a) Realizar el Diagrama de Comunicación correspondiente a la operación del sistema `crearCV(nom : string, desc: string).`

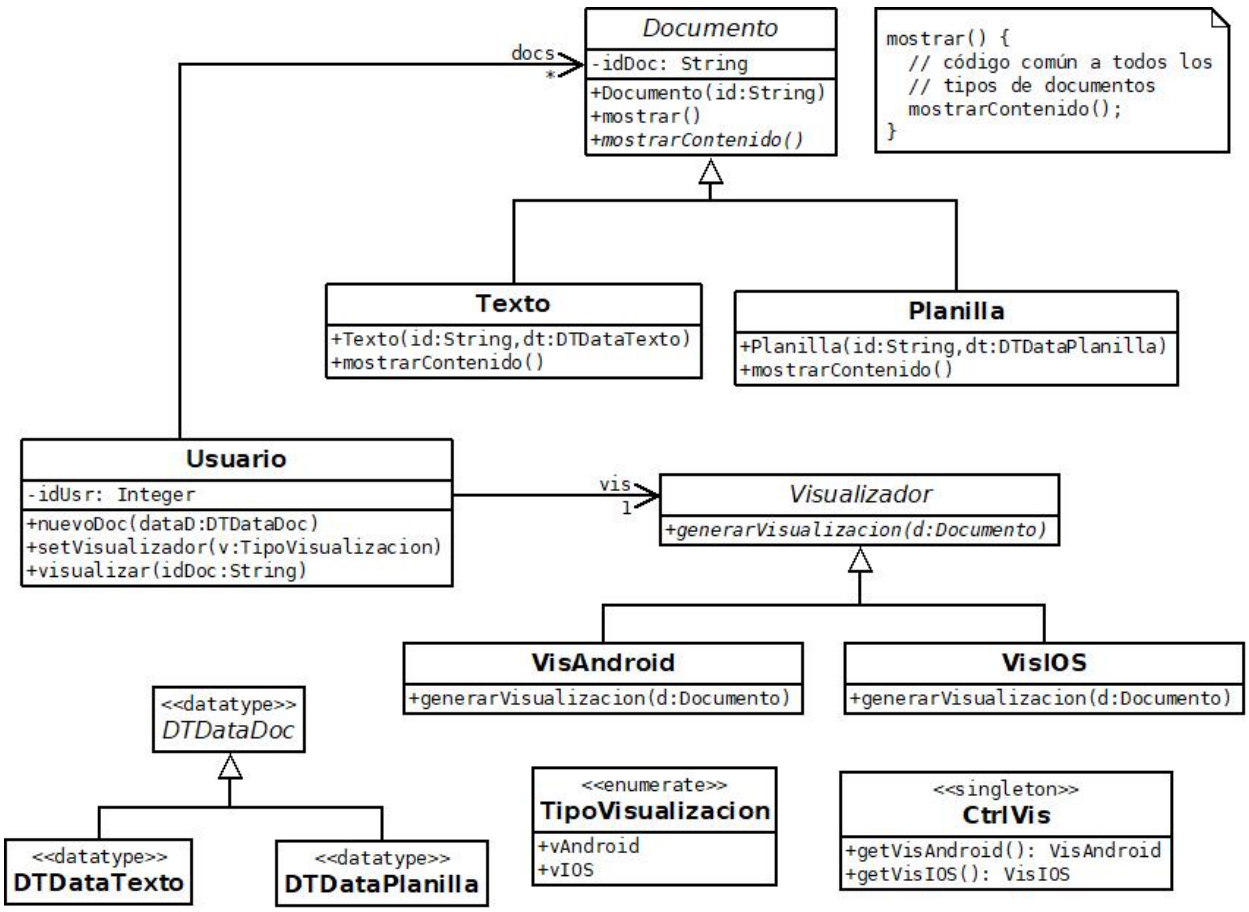
b) Realizar el Diagrama de Comunicación correspondiente a la operación del sistema `consultarPrincipales(nom : string):Set(DTR).`

c) Realizar el Diagrama de Clases de Diseño (DCD) resultante. No incorporar setters, getters ni operaciones de colecciones, e incorporar sólo los constructores y destructores que se utilicen en los diagramas de comunicación.

<u>Problema 3</u> (35 puntos)

Se está construyendo un sistema para la gestión de documentos (textos y planillas) que permite generar visualizaciones de estos documentos en diferentes plataformas móviles, por ejemplo, IOS y Android. En la figura se muestra un diseño parcial, que tuvo en cuenta las siguientes consideraciones:

1. Mostrar un documento implica realizar algunas acciones comunes a todos los tipos de documento, que se estructuran en un algoritmo general, y realizar otras específicas a cada tipo de documento que no afectan la estructura general del algoritmo.
2. Existen diferentes algoritmos de visualización dependientes de una plataforma, los cuales deben ser fácilmente intercambiables.



Además, se cuenta con las descripciones de los métodos de las siguientes operaciones:

- `Usuario::nuevoDoc`: Instancia un nuevo documento concreto según el tipo del datavalue recibido por parámetro y lo inicializa (a partir del datavalue concreto) utilizando su constructor. Setea el identificador del documento utilizando la operación `Utils::getNuevoIdDoc()` y lo agrega a la colección de documentos del usuario.
- `Usuario::setVisualizador`: Setea el visualizador del usuario en base al parámetro recibido. Los visualizadores de diferentes tipos se obtienen a partir de la clase `CtrlVis`.
- `Usuario::visualizar`: Recupera el documento a partir del parámetro, desde la colección de documentos del usuario. Luego genera la visualización utilizando el visualizador seteado.

Se pide:

a) Identifique qué patrones se utilizan en este diseño, exceptuando el patrón Singleton. Para cada patrón identificado, indicar las clases participantes y sus roles.
b) Implementar en C++ el .h de la clase `Usuario`. Incluir el(los) constructor(es) que sea(n) necesario(s). No incluir destructor ni operaciones get/set.
c) Implementar en C++ las operaciones `Usuario::nuevoDoc`, `Usuario::setVisualizador` y `Usuario::visualizar`.

Observaciones:

- La descripción de la operación `Documento::mostrar` se presenta a los efectos de la comprensión del ejercicio y a la realización de la solución de parte del mismo. Notar que no se pide su implementación, ni tampoco su invocación desde las operaciones que se pide implementar.
- Utilice colecciones paramétricas (contenedores STL) y la clase `std::string`.
- No incluya directivas al precompilador (`#include`, etc).