Programación 4

PARCIAL FINAL EDICIÓN 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 (25 puntos)

Se desea crear un sistema para un servicio de reproducción de música global. Los usuarios se registran con su email (único), contraseña y hasta 10 preferencias de géneros musicales. De cada canción se conoce su nombre, duración, fecha de publicación, género y autor. Los autores pueden ser solistas o bandas. En ambos casos, se conoce su género, nombre (que los identifica), fecha de inicio de actividad y, en el caso de ser bandas, la cantidad de miembros que la componen. Los usuarios pueden reproducir canciones tantas veces como deseen. Se registra la cantidad de veces que un usuario reproduce cada canción y la fecha de la última reproducción. Los usuarios pueden adquirir distintos tipos de suscripciones que se identifican con un nombre y tienen una duración expresada en meses. Cada usuario puede tener solamente una suscripción, de la cual interesa conocer sus fechas de comienzo y de expiración. La suscripción les da acceso a crear listas de reproducciones, cada lista puede tener hasta 100 canciones. Cada tipo de suscripción tiene un límite en la cantidad de listas que el usuario puede crear y una vez expirada la suscripción se descartan las listas.

Adicionalmente, se cuenta con la descripción del siguiente caso de uso:

Nombre	Alta de autor con canciones
Actor	Usuario
Descripción	El caso de uso comienza cuando un usuario desea agregar un autor con varias canciones asociadas. Para eso, primero se ingresa el nombre del autor y la fecha de inicio de actividad. Si es una banda, además se ingresa la cantidad de miembros. A continuación, se pide al sistema un listado de géneros y se selecciona uno de la lista o se agrega uno nuevo, en ambos casos dado por su nombre. El género seleccionado (o el agregado) quedará asociado al autor y a todas las canciones ingresadas. Finalmente, se ingresan varias canciones, cada una dada por su nombre, duración y fecha de publicación.

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.

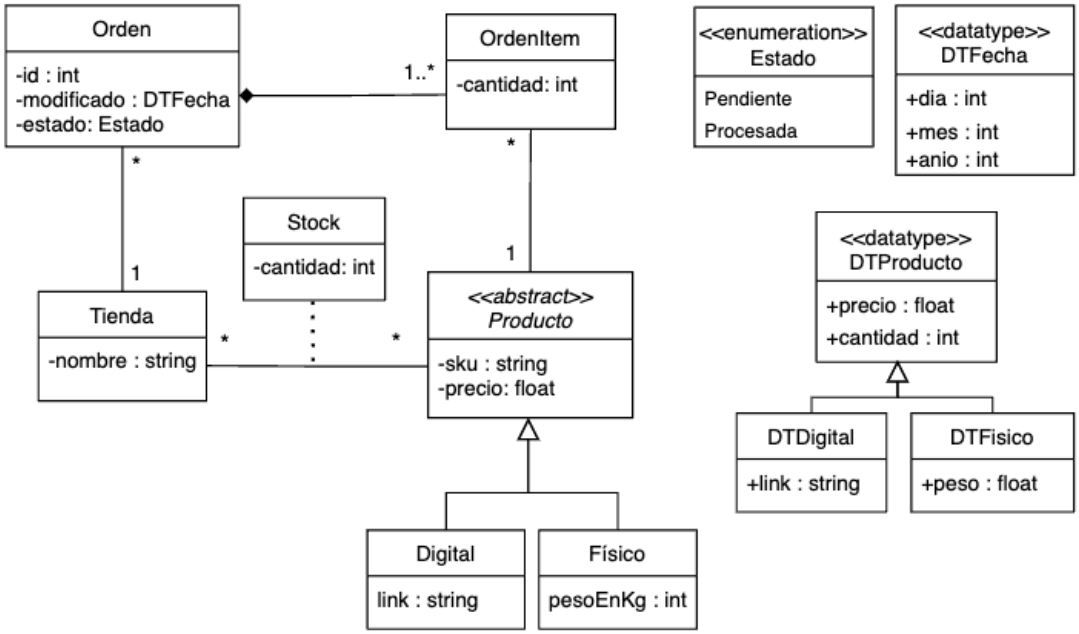
Problema 2 (30 puntos)

Se desea desarrollar una aplicación de comercio electrónico. Cada tienda tiene varios productos a la venta y existe un registro del stock de cada producto. Cada orden de venta puede contener varios ítems que detallan la cantidad solicitada de un producto. Al procesar una orden, se calcula el monto total de la misma. Para cada ítem, se toma la cantidad del producto solicitado y se multiplica por el precio del producto. Sin embargo, al existir dos tipos de productos, cada uno tiene un esquema de cálculo de precio diferenciado:

- Para los productos digitales, debido a los costos adicionales de infraestructura digital y licencias, se incrementa el precio de cada unidad en un 10% adicional al precio base.
- Para los productos físicos, se consideran los costos logísticos y de transporte, por lo que se añade una tarifa fija de $5 por cada kilogramo de peso del producto.

De esta manera, el monto total de una orden es la suma de los costos calculados para cada ítem, ajustados según el tipo de producto.

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



Restricciones:

- El atributo **sku** identifica al Producto.
- El atributo **id** identifica a la Orden.
- El atributo **nombre** identifica a la Tienda.
- Las órdenes de una tienda están compuestas de ítems de productos que vende la tienda.

Se pide:

a. Realizar el Diagrama de Comunicación correspondiente a la siguiente operación del sistema, incluyendo visibilidades:

registrarProductos(nt : string, dtp : Set(DTProducto))	
Descripción	Registra un conjunto de productos en la tienda.
Parámetros	nt: nombre de la tienda. dtp: datos de los productos a registrar.
Precondiciones	• Existe una instancia t:Tienda con t.nombre = nt
Postcondiciones	Para cada datavalue dp:DTProducto en el conjunto dtp: • Se crea una instancia p:Producto con los datos indicados en dp, según su tipo (DTDigital o DTFisico) • Se crea una instancia s:Stock con s.cantidad = dp.cantidad • Se crea un link entre p:Producto y s:Stock • Se crea un link entre t:Tienda y s:Stock

NOTA: Asuma que el controlador tiene una colección de todas las instancias de Tienda y una operación darNuevoSKU():string que proporciona un ID para un nuevo producto.

b. Realizar el Diagrama de Comunicación correspondiente a la siguiente operación del sistema, incluyendo visibilidades:

procesarOrdenes():Set(DTOrden)	
Descripción	Procesa las órdenes de compra pendientes y retorna la información de cada orden procesada.
Parámetros	No tiene.
Precondiciones	No tiene.
Postcondiciones	• Para cada instancia o:Orden con o.estado = "Pendiente" o Se actualiza el atributo o.modificado con la fecha actual del sistema. o Se crea un dto:DTOrden con dto.id = o.id y dto.monto = monto total de la orden. • Se retorna el conjunto de elementos dto.

NOTA: El controlador posee la operación getFechaActual():DTFecha que retorna la fecha actual del sistema.

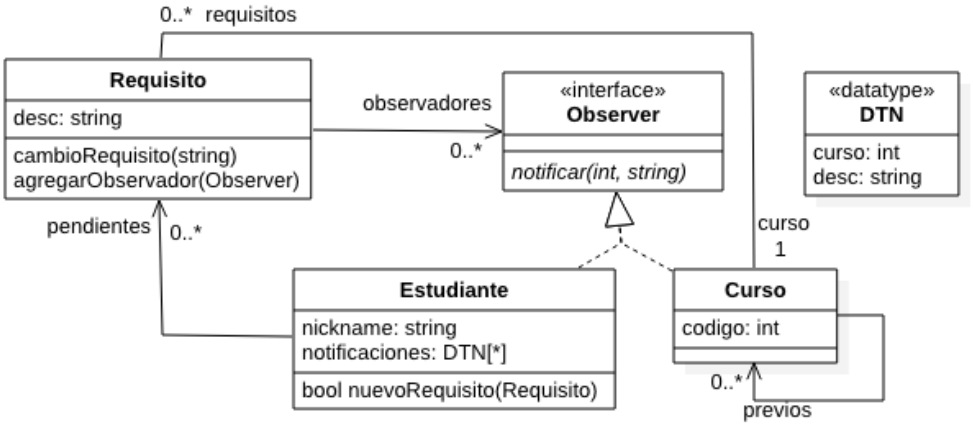
c. Realizar un único Diagrama de Clases de Diseño (DCD) resultante de las partes a) y b).

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

<u>Problema 3</u> (30 puntos)

El Diagrama de Clases de Diseño de la figura muestra el diseño parcial de un sistema de gestión de cursos de una universidad. Un Curso (identificado por un codigo) tiene un conjunto de Requisito para ser aprobado; cada requisito tiene una descripción (desc). Además, puede tener varios cursos previos que se requieren para ser cursado. Un Estudiante (identificado por un nickname) posee un conjunto de requisitos pendientes que provienen de los cursos a los que se inscribe, debiendo haber cumplido con todos los requisitos de los cursos previos. En la figura se observa la aplicación del patrón Observer que permite que tanto estudiantes

como cursos sean notificados de cambios en los requisitos. En particular, los estudiantes almacenan una lista de las `notificaciones` recibidas en un datatype `DTN`.



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

`void Requisito::cambioRequisito(string descripcion);`
Modifica la **`descripcion`** de un requisito y notifica a los observadores del cambio.

`void Requisito::agregarObservador(Observer * observador);`
Agrega al **`observador`** a la lista de observadores.

`void Observer::notificar(int curso, string descripcion);`
Permite notificar a un observador de un cambio en la **`descripcion`** de un requisito de un **`curso`**. En el caso de un estudiante, almacena la notificación utilizando el datatype `DTN`.

`void Estudiante::nuevoRequisito(Requisito * requisito);`
Incluye un nuevo **`requisito`** a su conjunto de requisitos pendientes. Si en el conjunto de requisitos pendientes del estudiante existe alguno de un curso previo al curso al cual pertenece el nuevo requisito (pasado por parámetro), existe un error, ya que debe aprobar primero los requisitos de todos los cursos previos (no se podrá inscribir al nuevo curso). En este caso, se lanza la excepción `std::runtime_error`. En caso contrario, se vincula al estudiante con el nuevo requisito y se agrega al estudiante a la lista de observadores del requisito.

Se pide:

a. Implementar en C++ el .h de la interfaz `Observer`.
b. Implementar en C++ el .h de la clase `Estudiante`.
c. Implementar en C++ el método de la operación `Requisito::cambioRequisito`.
d. Implementar en C++ el método de la operación `Estudiante::notificar`.
e. Implementar en C++ el método de la operación `Estudiante::nuevoRequisito`.

Observaciones:

- NO se pueden agregar más operaciones de las definidas en el diseño anterior.
- Utilice colecciones paramétricas (contenedores STL) y la clase std::string.
- No incluya directivas al precompilador (#include, etc).
- No es necesario implementar setters y getters adicionales de las clases; puede asumir su existencia si se requieren en algún método.