Programación 4

EXAMEN JULIO 2012

Por favor siga las siguientes indicaciones:

- Escriba con lápiz.
- Escriba las hojas de un solo lado.
- 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 examen junto al examen.

Problema 1 (30 puntos)

La División de Laboratorios Veterinarios del Uruguay está interesada en contar con una aplicación de vademécum (catálogo especializado) de productos veterinarios. Para ello relevó la siguiente información:

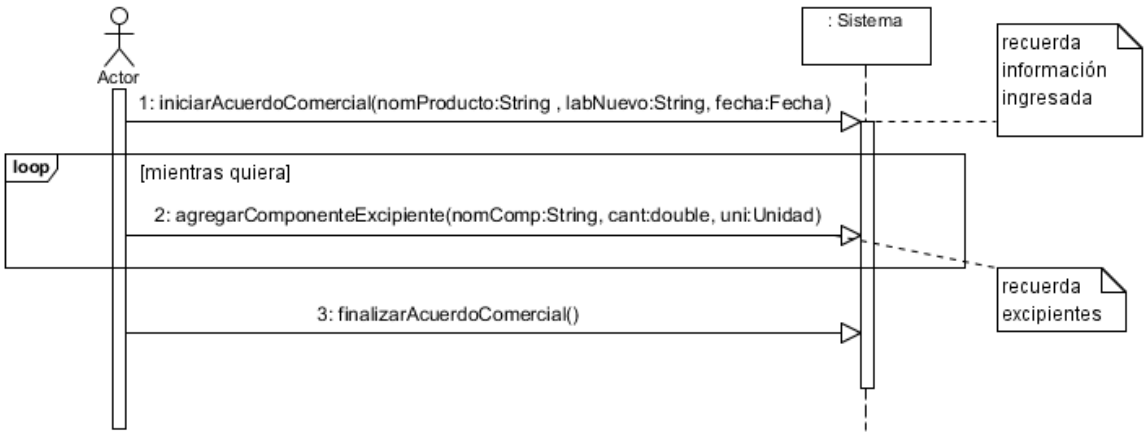
Un laboratorio elabora distintos productos veterinarios de forma exclusiva. Para elaborar productos, el laboratorio utiliza un conjunto de componentes autorizados, identificados por su nombre, y con un tipo (principio activo o excipiente). Un producto veterinario se identifica por su nombre comercial y posee una fórmula. La fórmula indica la cantidad de cada componente presente en el producto en una unidad determinada (miligramos, gramos, mililitros o litros). Los productos se clasifican como de consumo interno o exclusivo para exportar. En caso de ser exclusivo para exportar interesa saber los países a los que se exporta.

Existe la posibilidad de que un laboratorio firme un acuerdo comercial con otro para ser el nuevo elaborador de un producto veterinario. Se desea registrar la fecha del acuerdo comercial, el producto y los laboratorios participantes (actual y nuevo elaborador). Como parte del acuerdo se le permite al nuevo laboratorio agregar componentes excipientes a la fórmula del producto. Las condiciones del producto para realizar un acuerdo comercial son que sea para consumo interno y que no tenga otro acuerdo comercial.

Se requiere que la aplicación a desarrollar permita realizar búsquedas de productos veterinarios por nombre de laboratorio elaborador, nombre de país al que se exporta y nombre de un componente de la fórmula del producto.

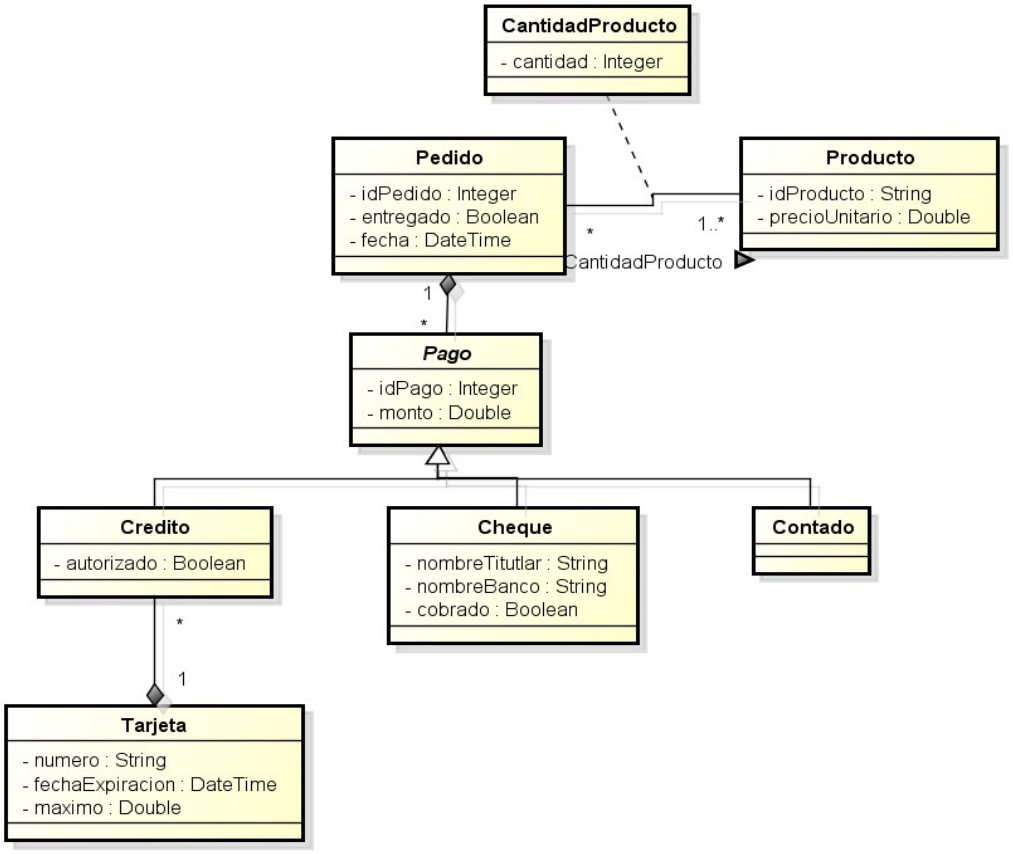
Se pide:

1. Modelar la realidad planteada mediante un Diagrama de Modelo de Dominio UML.
2. Expresar todas las restricciones del modelo en **lenguaje natural**.
3. Expresar las pre- y post-condiciones, en lenguaje natural, de los contratos de las operaciones del siguiente Diagrama de Secuencia del Sistema (DSS) con memoria, correspondiente a la creación de un acuerdo comercial:


Actor
: Sistema
recuerda información ingresada
1: iniciarAcuerdoComercial(nomProducto:String , labNuevo:String, fecha:Fecha)
loop
[mientras quiera]
2: agregarComponenteExcipiente(nomComp:String, cant:double, uni:Unidad)
recuerda excipientes
3: finalizarAcuerdoComercial()

Problema 2 (35 puntos)

Se desea diseñar un sistema de pedidos partiendo del siguiente diagrama de modelo de dominio.



i. Realizar el Diagrama de Comunicación completo de las siguientes operaciones del sistema. Indicar claramente los parámetros y el tipo del resultado de todas las operaciones involucradas en su solución. No se permite agregar atributos que puedan ser calculados por la información ya brindada en el modelo de dominio. Se pueden agregar DataTypes en caso de que lo requiera.

Operación	`estaPagoPedido(id : Integer) : Boolean`
Descripción	Verifica si el pedido cuenta con pagos suficientes para cubrir el costo de los productos.
Pre y Post	Pre: Existe una instancia de `Pedido` con `idPedido = id`. Post: Se retorna true si y sólo si la sumatoria de los montos de los pagos realizados es mayor o igual a la sumatoria de cantidades por precio unitario de los productos. En caso de `Crédito` además se requiere que este autorizado y en caso de `Cheque` se requiere además que este cobrado.

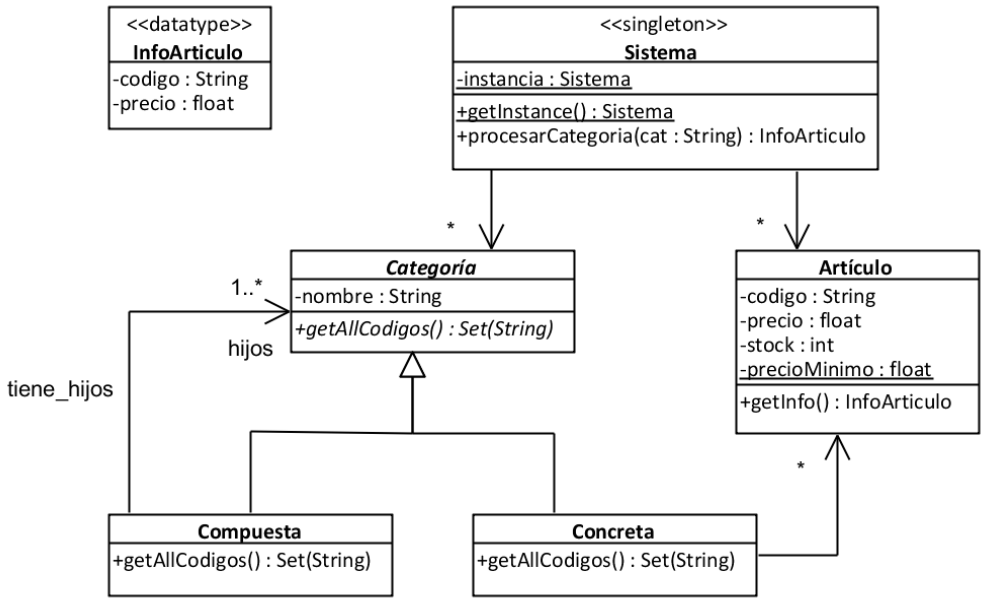
Operación	`anexarCreditoPedido(idPed : Integer, montoCred: Double,numTarj: String):Boolean`
Descripción	Se agrega a un Crédito al Pedido. En caso que sea posible se autoriza el pago con esa tarjeta de crédito. Se retorna true si y sólo si el pago fue autorizado.

Pre y Post	Pre: Existe una instancia de `Pedido` con `idPedido = idPed` y no entregado. Pre: Existe una instancia de `Tarjeta` con `numero = numTarj`. Post: Se crea una instancia de `Crédito` con `idPago` igual a 0 en caso que no tenga pagos el `Pedido` con `idPedido = idPed`, sino `idPago` será el máximo de los `idPagos` que tiene ese `Pedido` más 1. Post: A la nueva instancia de `Crédito` se le asigna al atributo monto el parámetro `montoCred`. Post: Se crea un link entre el `Pedido` con `idPedido = idPed` y la nueva instancia de `Crédito`. Post: Se crea un link entre la nueva instancia de `Crédito` y la `Tarjeta` con `numero = numTarj`. Post: En caso que la fecha del `Pedido` con `idPedido = idPed` sea anterior a la `fechaExpiración` de la `Tarjeta` con `numero = numTarj` y la sumatoria de los montos de los créditos otorgados (autorizados) de esa tarjeta más el monto del nuevo `Crédito` no supere `maximo` se asignará en true el atributo autorizado del `Crédito`. En caso contrario se asignará autorizado con false. Post: El valor de retorno será true si y sólo si el `Crédito` creado ha sido autorizado.

ii. Realizar el Diagrama de Clases de Diseño (DCD) correspondiente para las operaciones de i.

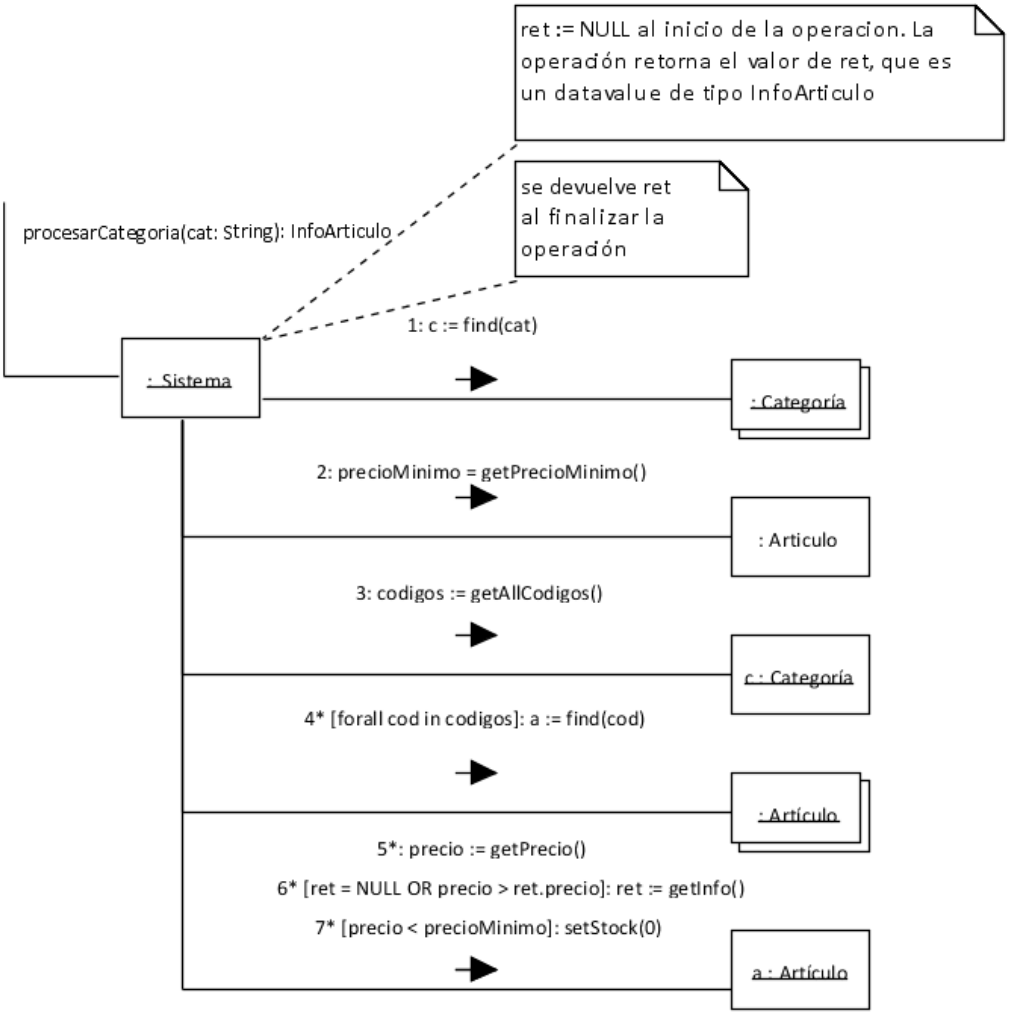
Problema 3 (35 puntos)

Se desea implementar un sistema que gestiona el inventario de artículos de una tienda. Los artículos tienen un código, precio y se conoce su stock. Además, los artículos se catalogan en distintas categorías que pueden estar anidadas dentro de otras categorías. A continuación se presenta un diagrama de clases de diseño parcial de la solución diseñada.



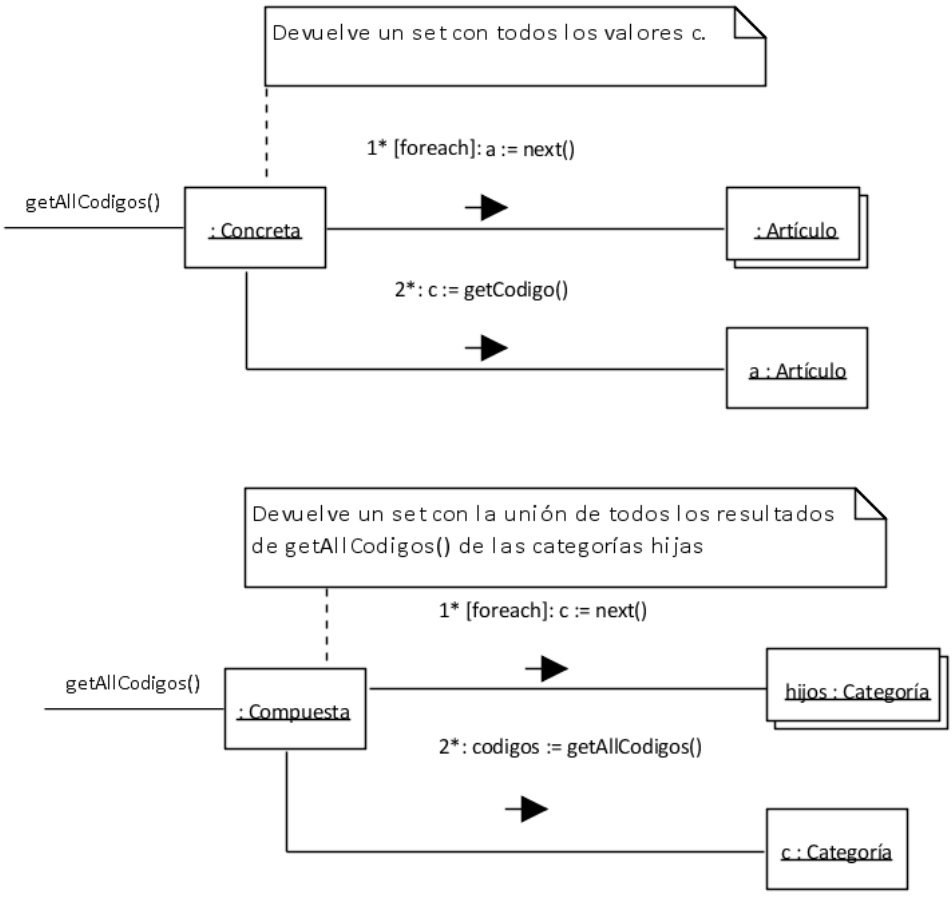
La operación `procesarCategoría(cat: String)` reduce a 0 el stock de todos los artículos de la categoría `cat` cuyo precio es menor a `Articulo::precioMinimo` y devuelve la

información del artículo de precio mayor que esté asociado a dicha categoría usando la operación `Categoria::getAllCodigos()`. Se tiene el siguiente diagrama de comunicación para la operación `procesarCategoría()`.



La operación `procesarCategoría()` tiene como precondición que exista una categoría de nombre igual a cat en el sistema y en caso contrario lanza una excepción de tipo `std::invalid_argument`.

La operación abstracta `getAllCodigos()` devuelve un conjunto con todos los códigos de los artículos asociados a la misma categoría (si es `Concreta`) o asociados recursivamente a través de la asociación `tiene_hijos` (si es `Compuesta`). A continuación los diagramas de comunicación correspondientes a dicha operación.



Se pide:

i. Implementar en C++ los .h de las clases Sistema, Artículo, Categoría y Compuesta.

ii. Implementar en C++ los .cpp de las clases Sistema y Compuesta.

Notas:

- Deben verificarse las precondiciones de las operaciones.
- No debe incluirse get y set de atributos.
- Suponga que se tiene una implementación de la clase String que implementa ICollectible e IKey.
- Suponga que se tiene una implementación de ICollection llamada Col e IDictionary llamada Dict.
- No incluya constructores ni destructores de las clases.
- No incluya las directivas del preprocesador.