Tecnólogo Informático – San José
Programación Avanzada
Examen Julio 2016

- *Completar TODAS las hojas con el nombre y el número de cédula.*
- *Escriba las hojas de un solo lado.*
- *No se puede utilizar material de ningún tipo.* ***Apagar celulares.***
- *Solo se contestan dudas acerca de la letra de los ejercicios.*
- *El examen dura 2 horas y media.*

Ejercicio 1 (30 puntos)

a. Indique brevemente la relación entre los siguientes elementos: Caso de Uso, Escenario, Operación del Sistema, Contrato y Diagrama de Secuencia del Sistema.

b. Se desea construir un sistema de gestión con información sobre ómnibus (buses), compañías de buses así como los horarios y lugares de partida y destino de los servicios que ofrecen las compañías consultables mediante la web. En este sistema *no se mantiene información histórica*.

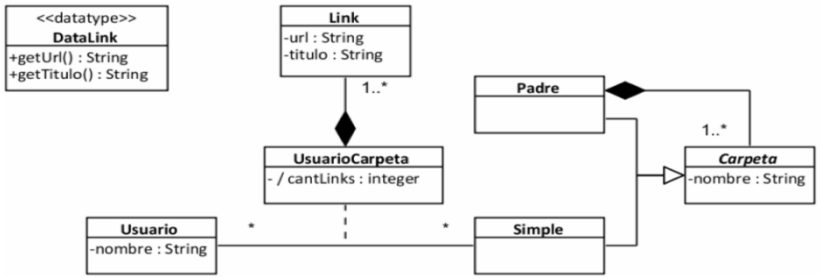
Los buses se identifican por el nombre del fabricante, su modelo y el nro de fabricación. Se conoce también la fecha de fabricación y su precio. En este sistema se desea contar con información tanto de buses usados como de los nuevos que los fabricantes van teniendo a disposición. De un bus usado interesa saber la fecha estimada en la que quedará en desuso. A su vez, de un bus nuevo interesa los años de garantía. Se cuenta con la información de los buses más allá de que tengan alguna relación con las compañías de buses (están en exhibición).
Las compañías de buses se identifican por un nombre y el país. De ellas también se conoce su dirección, teléfono y url de su página web. Una compañía puede estar interesada en los buses y/o puede haberlos comprado. Cuando la compañía compra un bus se tiene como parte de la transacción el precio y la fecha de la compra. La fecha en que realiza una compra debe ser posterior a la fecha de fabricación y en el caso de que el bus sea usado deber ser además inferior a la fecha estimada de desuso. Es política de todas las compañías que cuando compran un bus usado, en la misma transacción comercial, también definen el interés por un bus nuevo del mismo fabricante que el usado (si es que hay disponible). Que una compañía haya definido su interés por un bus no significa que quede reservado para ella; otra compañía también puede manifestar su interés por el mismo bus. En cambio la compra es realizada por una sola compañía.
Los servicios que brindan las compañías están dados por los horarios y precios de transportar pasajeros de un lugar a otro. Una posible presentación en la web de esta información es mediante una tabla con líneas donde cada línea indica dos terminales de buses, el origen y destino (que deben ser diferentes), junto con el día de la semana de salida, la hora de salida, su duración en horas, el precio en dólares y la compañía que realiza ese servicio. Ejemplo de una línea es MVD, Floripa, sábado, 14:00, 15hrs, 250 U$S, TTL. Cada compañía ofrece desde una misma terminal a lo sumo 2 servicios diarios a una misma terminal destino. Todo servicio debe ser cubierto al menos por una compañía. Una terminal se identifica por el país, la ciudad y el nombre de la terminal. Se conoce además su dirección.

Se pide: Modelo de Dominio, con restricciones en lenguaje natural.

Ejercicio 2 (35 puntos)

Se desea construir un sistema para un sitio web de almacenamiento de enlaces donde los usuarios pueden compartir *Links* interesantes a páginas web y almacenarlos en diferentes carpetas que son visibles desde todo el sitio. Las carpetas son compartidas y los usuarios pueden subir links en las carpetas independientemente de que otro usuario haya subido algún link en ella. Las carpetas forman una estructura arborescente de manera similar a un sistema de archivos. Hay dos tipos de carpetas: las *Carpetas Simples* que son las que contienen los Links y las carpetas *Padre* que agregan a un conjunto de carpetas.

De cada *Link* se conoce la URL a la página web (que lo identifica dentro de la carpeta) y su título. De los usuarios se conoce el nombre que los identifica. Es de interés saber para cada usuario cuantos links en total subió a cada carpeta. Los links pueden estar solamente en Carpetas Simples. De una carpeta se conoce su nombre que la identifica.
De acuerdo a los requerimientos el equipo de analistas realizó un modelo de dominio que se deberá respetar durante el diseño.



La intención es que el equipo que usted integra diseñe la siguiente operación del sistema.

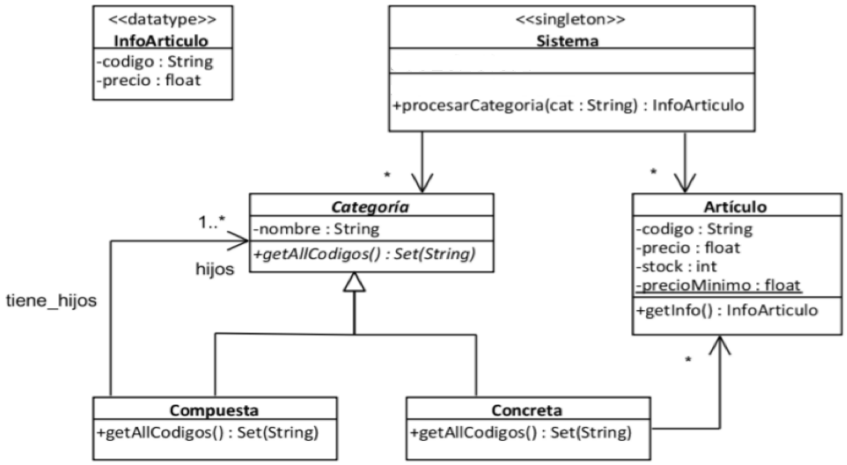
`getLinks(usr: String, N: integer, carp: String):Set(DataLink)`	
Descripción	Obtiene los datos de los links subidos en Carpetas Simples contenidas en la carpeta de nombre `carp` por el usuario de nombre `usr`. Sólo se consideran los links subidos en Carpetas Simples donde el usuario haya subido al menos `N` links. Si la carpeta `carp` es Simple sólo se busca en esa carpeta. De lo contrario se busca en las **todas las** carpetas hijas de `carp`.
Precondiciones	Existe una instancia de Usuario con nombre `usr` Existe una instancia de Carpeta (Simple o Padre) de nombre `carp`
Postcondiciones	Devuelve un Set con data values de DataLink que correspondan a los Links que están asociados a una instancia de UsuarioCarpeta `uc` que cumple que • `uc` está asociada al usuario de nombre `usr` • `uc.cantLinks ≥ N` • `uc` está asociada con una instancia de carpeta Simple de nombre `carp` o `uc` está asociada con una instancia de carpeta Simple que es hija de una carpeta de nombre `carp`.

Se pide

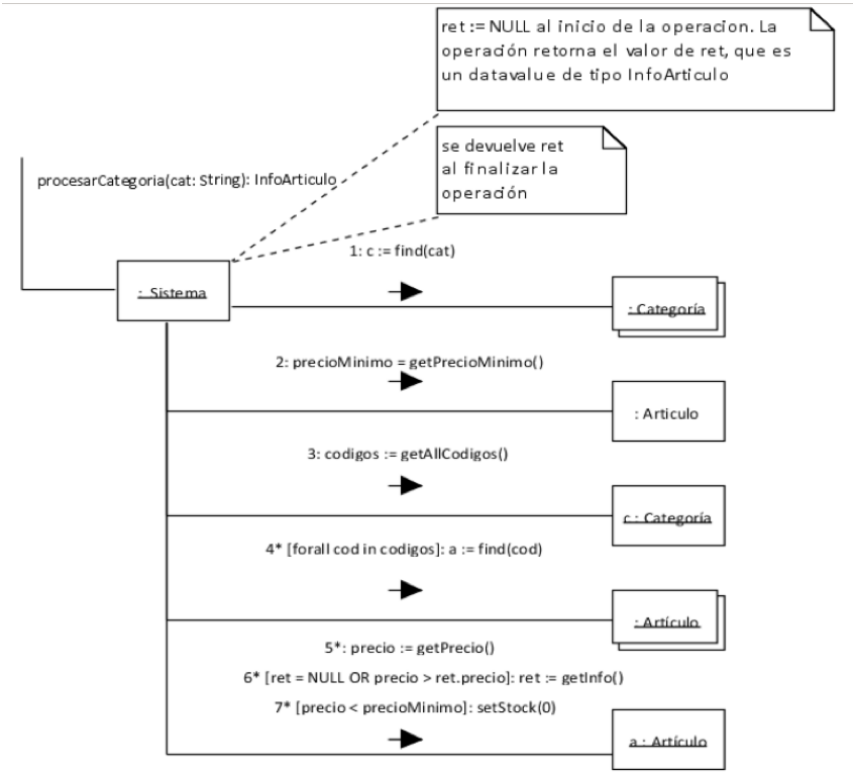
1. Realizar el Diagrama de Comunicación de getLinks(), indicar visibilidades.
2. Realizar el Diagrama de Clases de Diseño de la solución

Ejercicio 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.

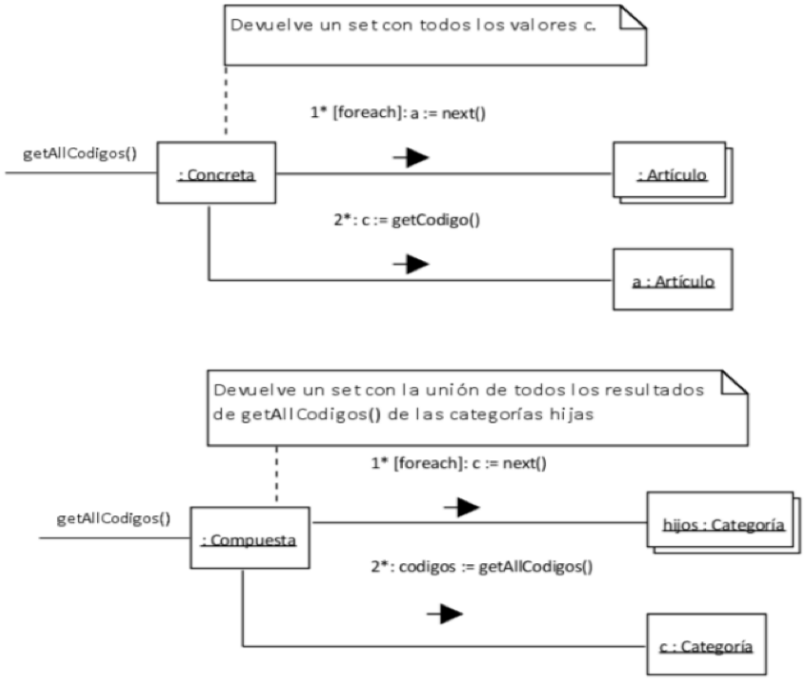


La operación *procesarCategoría(cat: String)* reduce a 0 el stock de todos los artículos de la categoría *cat* cuyo precio es menor a *Articulo::precioMinimo* y devuelve la información del artículo de precio mayor que esté asociado a dicha categoría. Para esto, utiliza la operación *Categoria::getAllCodigos()*. Se tiene el siguiente diagrama de comunicación para la operación *procesarCategoria()*.



La operación *procesarCategoria()* 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, IDictionary y de List.
- No incluya constructores ni destructores de las clases.