Examen julio 2012

Presentar en la resolución del examen:

- Indique su nombre completo y número de cédula en cada hoja.
- Numere todas las hojas e indique la cantidad total de hojas que entrega en la primera.
- Escriba las hojas de un solo lado.
- Comience cada ejercicio en una hoja nueva.
- El examen es individual y sin material. APAGUE SU CELULAR.
- Escriba con lápiz y de forma prolija.
- Duración: 3:00 horas.

Problema 1 (30 puntos)

a) Mencione y explique brevemente tres *frames* (marcos) posibles de ser utilizados en los Diagramas de Secuencia.

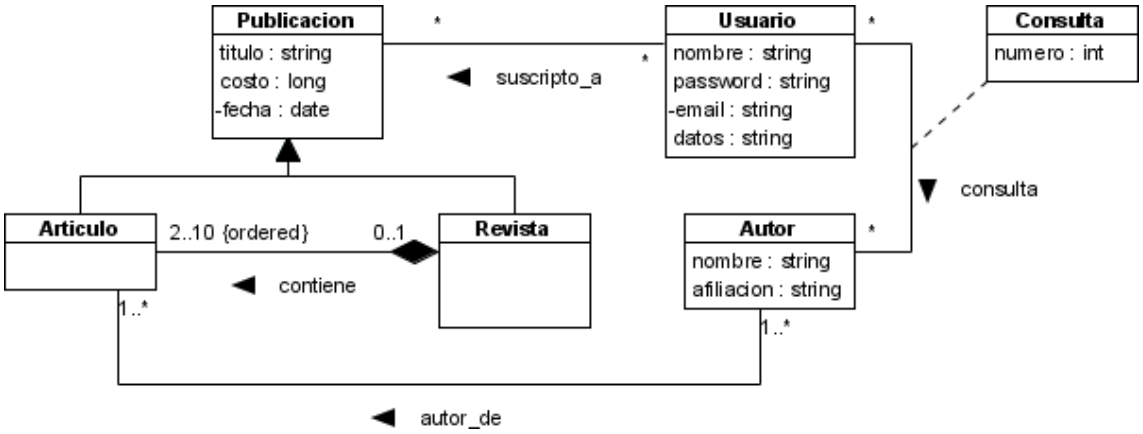
b) Un proyecto de software que siga el modelo de proceso Iterativo e Incremental como el visto en el curso constará de un conjunto de iteraciones (típicamente 2 o más pues de lo contrario no sería iterativo). Cada iteración involucra realizar las etapas de Análisis, Diseño, Implementación y Testeo (obviamente en ese orden). En cada una de estas etapas (para cada iteración) se realizan un conjunto de artefactos (como ser documentos, componentes de software, testeos, etc.) identificados por un nombre y cuyo responsable es un miembro del equipo de desarrollo. Asimismo, el proyecto contará con un jefe de proyecto, tendrá un título que lo diferencia de otros proyectos, un costo estimado y una fecha de entrega. Cada iteración contará con un número, una fecha de inicio y de finalización (por ejemplo, la primera iteración comienza el 1º de Junio de 2010 y finaliza el 30 de Julio del mismo año) y dentro de ella se realizan las 4 etapas ya mencionadas. También interesa conocer la duración de cada una de las etapas dentro de cada iteración (por ejemplo, la etapa de Análisis de la primera iteración llevará 2 semanas, la de Diseño 3 semanas, la de Implementación 2 semanas y la de Testeo 1 semana). No es necesario representar ni la etapa previa a las iteraciones (Relevamiento) ni las posteriores (Liberación y Mantenimiento).

Se pide:

Diagrama de dominio de la realidad planteada con restricciones en lenguaje natural.

Problema 2 (30 puntos)

Se está desarrollando un sitio web con un sistema de suscripción a publicaciones electrónicas. El modelo de dominio realizado se puede ver en la figura siguiente. El sitio tiene disponible un conjunto de publicaciones electrónicas que son revistas y artículos científicos, que pueden o no formar parte de una revista. Existen usuarios que se registran en el sitio para acceder a las publicaciones electrónicas. Además, se les permite realizar consultas sobre las publicaciones, llevándose registro de la cantidad de consultas que cada usuario realizó para un autor determinado.



Además, se definieron las siguientes operaciones de sistema.

Operación	consultarAutoresPub(nomUsr:String, titPub:String):Set(DataAutor)
Pre- y poscondiciones	
def: Definimos **colAutores** como la colección de todas las instancias de Autor asociadss a la Publicación con título = titPub. En el caso de una instancia de Revista, **colAutores** es la unión de los autores de todas las instancias de Artículo linkeadas con la Revista. pre: Existe una Publicación con título = titPub pre: Existe un Usuario con nombre = nomUsr con un link a la Publicacion anterior post: Para cada instancia de Autor en **colAutores**, si existía una instancia de Consulta entre la instancia de Autor y la de Usuario con nombre = nomUsr, entonces se suma 1 a la cantidad de consultas realizadas entre ambos (numero = numero + 1); sino, se crea una instancia de Consulta entre ambas instancias y se inicializa el atributo numero en 1. post: Se retorna una colección de datavalues DataAutor. Cada DataAutor se corresponde con una instancia de Autor existente en **colAutores** y contiene el nombre del Autor y el número de Consulta que ha realizado el Usuario con nombre = nomUsr para ese autor.	

Operación	bajaUsuario(nomUsr:String)
Pre- y poscondiciones	
pre: Existe un Usuario con nombre = nomUsr post: Se eliminó la instancia de Usuario con nombre = nomUsr y los links de la instancia de Usuario a las instancias de Publicacion post: Se eliminaron las instancias de Consulta linkeadas con la instancia de usuario	

Se pide:

a) Realice los Diagramas de Comunicación para las operaciones especificadas en los contratos. No es necesario indicar las visibilidades.
b) Realice el Diagrama de Clases de Diseño **completo** de la solución.

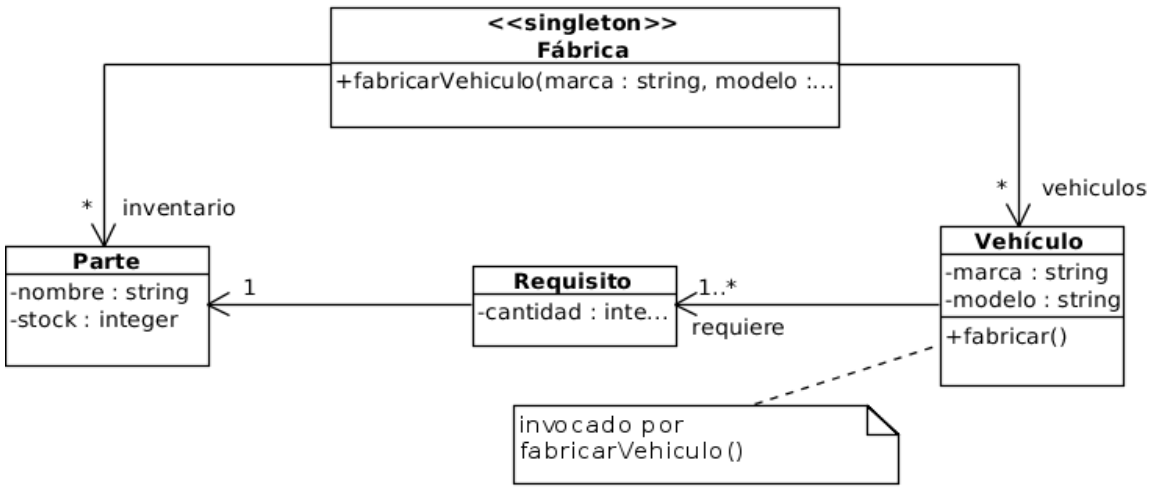
Problema 3 (30 puntos)

Se le ha pedido implementar un sistema para una fábrica de automóviles. Los vehículos se identifican conjuntamente por su marca y su modelo (por ejemplo Fiat Uno). Para construir cada automóvil se necesita cierta cantidad de partes que se identifican por su nombre. De cada parte se conoce además su stock disponible. Para fabricar un vehículo tiene que haber stock suficiente de cada una de las partes requeridas, y cada vez que se construye un vehículo, el stock de las partes usadas decrece.

Como parte del análisis se generó el siguiente contrato de la clase Fábrica.

Operación	**`fabricarVehiculo(marca: String, modelo: String)`**
Descripción	`Fabrica un vehículo decrementando el stock de partes usadas para fabricarlo, según los requisitos de partes para el vehículo.`
Pre y poscondiciones	
`pre:` `Existe una instancia de vehículo` **`v`** `cuyo atributo marca es igual a` **`marca`** `y cuyo atributo modelo es igual a` **`modelo.`** **`pre:`** `Para cada instancia de Requisito` **`r`** `asociado a` **`v`** `se cumple que el atributo cantidad de` **`r`** `no es mayor al valor del atributo stock de la instancia Parte que tiene asociada` **`post:`** `Para cada instancia de Parte asociada a las instancia de Requisito` **`r`** `anteriores, se decrementa el valor del atributo stock en N unidades, siendo N el valor del atributo cantidad de` **`r`**	

A partir de dicho contrato se generó el siguiente diseño que debe ser respetado en la implementación.



Se pide implementar en C++

a) El .h y .cpp de la clase `Vehículo`. No es necesario incluir en la implementación constructores, destructores ni operaciones get y set.

b) El .h y el .cpp de la clase `Fabrica`.

Incluya explícitamente la verificación de las precondiciones en la implementación de la operación `fabricarVehiculo`(), lanzando excepciones en caso de que no se cumplan.

Observaciones:

- Asuma que existen implementaciones estándar de las interfaces `ICollection`, `IIterator`, `IDictionary` e `IKey`.
- Asuma la existencia de la interface `ICollectible`.
- Asuma la existencia de una clase `Exception` para manejar las excepciones de las operaciones.
- No es necesario incluir directivas de preprocesador.