Programación 4

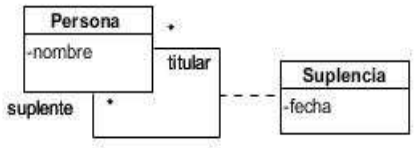
EXAMEN FEBRERO 2015

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 examen junto al examen
- Está prohibido el uso de computadoras, tabletas o teléfonos durante el parcial

Problema 1 (35 puntos)

Parte A:



Considere el diagrama de clases de la izquierda. La instancia dada por:

Persona	**Suplencia**
p1: "Joe Dalton", p2: "Ma Dalton", p3: "Averell Dalton" p4: "Jack Dalton"	l1: (p1, p2, 1/1/2014) l2: (p3, p2, 25/12/2014) l3: (p1, p2, 1/1/2015)

¿es una instancia válida de ese diagrama?. Explique clara y precisamente su respuesta.

Parte B:

Se desea construir un sistema que gestiona llamados a ocupar cargos para desempeñar tareas en una institución de enseñanza por personas que se postulan y son elegidas por concurso. Para este sistema no interesa mantener la información acerca de los cargos independientemente de los llamados.

Hay cargos docentes y no docentes. Los cargos se los conoce por los datos del escalafón y el grado. El escalafón es una letra y el grado es un número. Los valores de los escalafones para los cargos docentes y no docentes son disjuntos. Los escalafones de los cargos docentes solo pueden tener el valor 'G' y los grados van del 1 al 5.

Para cubrir los cargos docentes y no docentes, la institución realiza llamados docentes y no docentes que se publican en su sitio web. Del relevamiento realizado, los datos que interesan de los llamados son los siguientes:

- Datos básicos. Los llamados tienen un número de expediente que los identifica. Se conoce el cargo para el cual se hace el llamado, las horas por semana a desempeñar y la cantidad de plazas que se disponen. El período de inscripción al llamado se define por medio de una fecha de inicio y de fin. En el caso de llamados docentes se necesita indicar además el área temática de desempeño la cual no está establecida y por tanto debe permitirse el ingreso de texto libre.
- Modalidades. Un llamado puede tener de una a dos modalidades aprovechando el mismo llamado para el ingreso de personas nuevas a la institución (modalidad abierto) y/o promoviendo a los funcionarios ya ingresados (modalidad cerrado). Para cada modalidad del llamado se define la cantidad de plazas disponibles

donde la suma de esas plazas para un llamado debe corresponderse con la cantidad de plazas del llamado.

- Oficina del llamado. Se refiere a la oficina a la que los postulantes pueden recurrir en caso de preguntas acerca del llamado. La oficina se identifica por su dirección. Tiene un nombre, teléfono, y correo electrónico.
- Recepción de la documentación en papel. Un llamado requiere la entrega de certificados en papel por parte de los postulantes. Para ello, en el llamado se indican las oficinas (al menos una) a las que los postulantes deben asistir para presentarlos. Sin embargo, la oficina del llamado (definida anteriormente) no recibe las documentaciones en papel del llamado correspondiente.

Las personas se inscriben a los llamados, en las modalidades que les interesa y son asignadas a una oficina determinada para entregar su documentación papel dentro de las definidas para el llamado. El sistema registra la fecha de la inscripción y brinda un número de inscripción por modalidad de los llamados a los que la persona se postuló. Este número sirve para determinar los datos de la inscripción y por tanto pasible a utilizar cuando se realizan consultas y/o reclamos. Cuando el postulante asiste a la oficina de recepción asignada para presentar la documentación papel, se verifica que la persona se inscribió y se registra la fecha de esa recepción. Por último, se desea mantener la información de los postulantes que quedaron designados luego de realizados los concursos de cada uno de los llamados.

Caso de Uso:	Consulta estado de inscripciones de una persona
Actor:	Administrativo
Descripción:	Comienza cuando el administrativo ingresa la cédula de identidad o el correo electrónico (sólo uno de los dos) de una persona. Si la persona nunca realizó una inscripción a un llamado, el caso de uso finaliza. En caso contrario, el sistema muestra el nombre y apellido de la persona e indica si tiene inscripciones. En caso de no tener inscripciones, el caso de uso finaliza. Si tiene, el administrativo continúa ingresando dos fechas (fecha inicio y fecha fin de un rango) y el sistema lista las inscripciones en ese rango de la persona. La información de una inscripción consiste en el número de inscripción, la fecha en que la realizó, el número de expediente y modalidad, el escalafón y grado del cargo, si entregó o no la documentación papel y, en caso de estar el resultado del concurso, si quedó o no designado.

Se pide:

a) Modelo de dominio con restricciones en lenguaje natural.

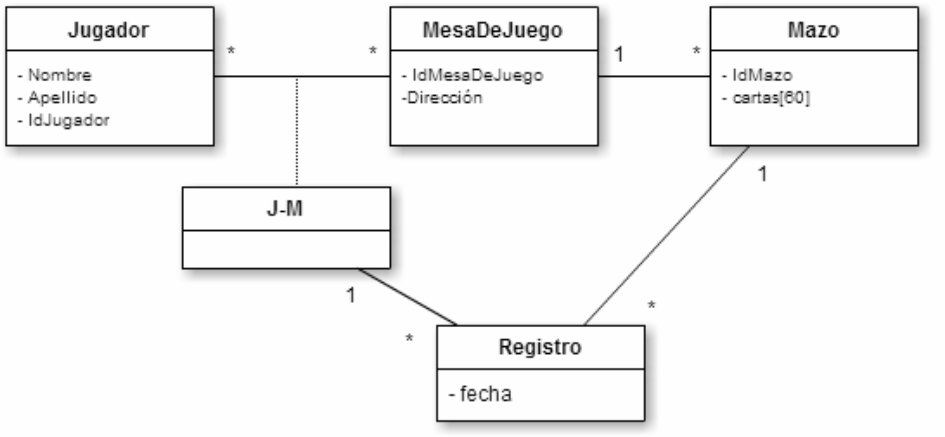
b) DSS para el Caso de Uso "Consulta estado de inscripciones de una persona"

Problema 2 (30 puntos)

Contexto:
Magic: el encuentro, originalmente en inglés **Magic: The Gathering** es un juego de cartas coleccionables. Cada partida de Magic representa una batalla entre dos jugadores, en donde cada jugador posee un mazo de cartas coleccionables que son usadas para la batalla. Un sistema organizado de torneos y una comunidad de jugadores profesionales se ha desarrollado alrededor del juego, por lo que a usted y a su equipo de desarrollo le han encomendado la tarea de diseñar un pequeño módulo que tiene como finalidad llevar registro de un torneo magic, llamado MagicUT (Magic uruguayan Tournament).

Realidad:
En MagicUT existe lo denominado "mesa de juego" que es una instancia en la que varios jugadores se reúnen en un lugar acordado y compiten entre sí. En el torneo, cada jugador puede haber participado en múltiples mesas de juego, así como cada una de estas puede haber incluido a varios jugadores. Interesa registrar las fechas en las que el jugador participó en cada mesa, así como el mazo que utilizó. Cada mesa de juego puede poseer varios mazos pre-armados (los cuales serán utilizados por los jugadores que participen en dicha mesa), así como cada mazo pertenece a una única mesa de juego.

El equipo de Análisis le ha otorgado el siguiente modelo:



Restricciones:
1) Un jugador se identifica por IdJugador
2) El concepto MesaDeJuego se identifica por IdMesaDeJuego
3) Cada mazo se identifica por IdMazo
4) Un jugador que genera un registro de un mazo en una mesa de juego, participó en esa mesa de juego.

Se consideran los casos de uso "Agregar nuevo registro a jugador en mesa de juego" y "Obtener información de un jugador", cada uno de los cuales es modelado con una única operación del Sistema, cuyos contratos se especifican a continuación:

registrarJugadorEnMDJ(IdJ, IdMDJ, IdMazo**: Integer,** fActual**: DateTime)**	
Descripción	Se agrega un nuevo registro de un jugador en una mesa de juego en la que ya ha participado previamente
Parámetros	- IdJ: Id del jugador. - IdMDJ: Id de la mesa de juego. - IdMazo: Id del mazo. - fActual: fecha actual del Sistema.
Precondiciones	- Existe en el Sistema una instancia de Jugador con IdJugador igual a IdJ. - Existe en el Sistema una instancia de MesaDeJuego con IdMesaDeJuego igual a IdMDJ. - Existe en el Sistema una instancia de Mazo con IdMazo igual a IdMazo. - Existe una instancia de tipo asociativo J-M entre un Jugador cuyo IdJugador es igual a IdJ y una MesaDeJuego cuyo IdMesaDeJuego es igual a IdMDJ - Existe una asociación entre una instancia de Mazo cuyo IdMazo es igual a IdMazo y una instancia de MesaDeJuego cuya IdMesaDeJuego es igual a IdMDJ
Postcondiciones	- Existe en el Sistema una nueva instancia de Registro con fecha actual. - Se crea una asociación entre el nuevo Registro y el tipo asociativo J-M de Jugador cuyo IdJugador es igual a IdJ y MesaDeJuego cuyo IdMesaDeJuego es igual a IdMDJ. - Se crea una asociación entre el nuevo Registro y el Mazo cuyo IdMazo es igual a IdMazo.

obtenerInfoJugador(IdJ **: Integer) : DataJugador**	
Descripción	Devuelve información de un jugador
Parámetros	- IdJ: Id del jugador
Precondiciones	- Existe en el sistema una instancia de Jugador identificada por IdJ.
Postcondiciones	Se retorna un datavalue con la siguiente información: - Nombre del Jugador cuyo IdJugador es igual a IdJ. - El IdMesaDeJuego y dirección de todas las mesas de juego en las que participó, incluyendo para cada una, todas las fechas de sus registros y los IdMazos de cada registro.

Se pide:

a) Realizar el Diagrama de Comunicación para cada una de las operaciones previamente declaradas.

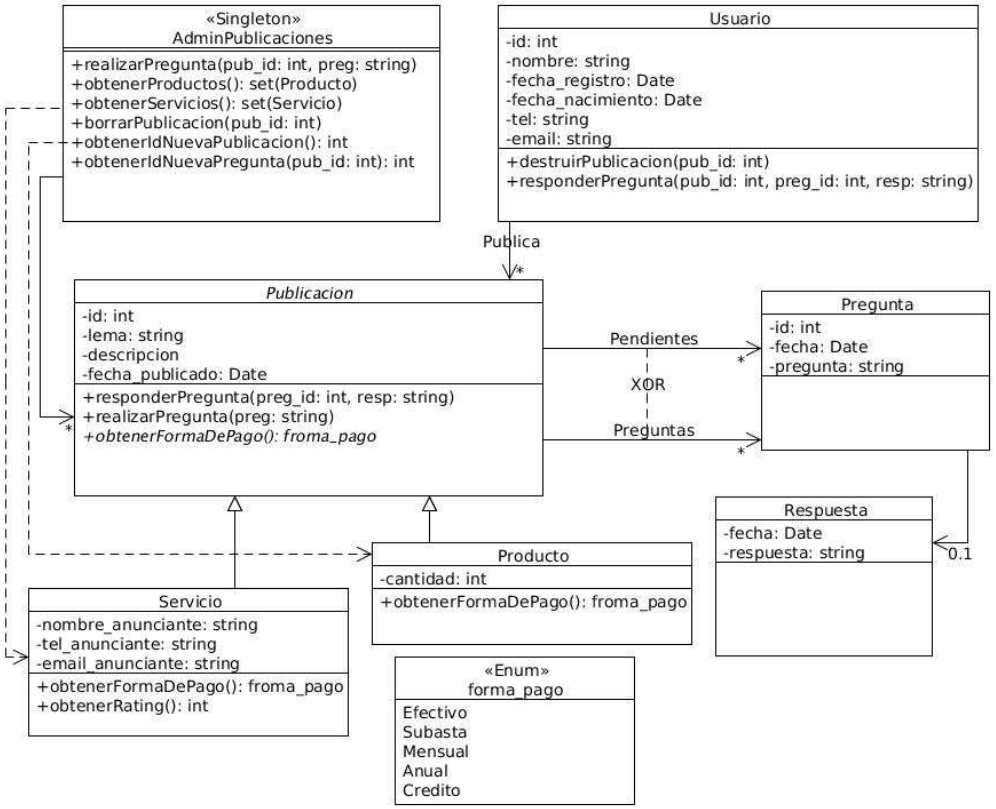
b) Realizar el Diagrama de Clases de Diseño correspondiente.

Problema 3 (35 puntos)

Se desea implementar en C++ una aplicación online que permita realizar ventas de productos y/o servicios. El comportamiento del sistema es el siguiente:

- Para que un usuario pueda comprar y/o vender debe estar registrado en el sistema.
- Cualquier usuario puede realizar preguntas sobre un **Producto** o **Servicio**.
- La clase **AdminPublicaciones** es la encargada de administrar los **id** de las instancias de **Publicacion** y **Pregunta**, por lo que posee métodos que permiten obtener un nuevo id para cada instancia a través de
  int obtenerIdNuevaPublicacion(),
  int obtenerIdNuevaPregunta(pub_id: int),
  los cuales retornan un nuevo id válido para el nuevo objeto *(Asuma que cuenta con la implementación de dichos métodos)*.

A continuación se muestra una parte del modelo de diseño de dicha aplicación.



Se pide:

a) Implementar los **.h** de **AdminPublicaciones, Publicacion y Servicio.**

b) Implementar completamente las siguientes operaciones, esto significa que se debe implementar todas las operaciones necesarias para completar la funcionalidad aunque estas no se pidan explícitamente. Tener en cuenta que la única clase que es utilizada por el resto del programa es **AdminPublicaciones**, por lo que las únicas referencias a las clases del diagrama son las definidas en el mismo.

Métodos de la clase **AdminPublicaciones** a implementar:

- *realizarPregunta(pub_id: int, preg: string)*
  Crea una nueva pregunta para la publicación con id *pub_id* con el texto *preg*. Asuma que existe la instancia de **Publicacion** a la que se le quiere realizar una pregunta.

- *obtenerProductos(): Set(Producto)*
  Retorna un set con todos los **Productos** publicados en el sistema.

- *borrarPublicacion(pub_id: int)*
  Desreferenca la instancia de Publicación con id **pub_id.**

Métodos de la clase **Usuario** a implementar:

- *destruirPublicacion(pub_id: int)*
  Elimina la instancia de **Publicacion** con id **pub_id**. Notar que las preguntas y respuestas dependen solamente de la publicación a la que pertenecen.

Implementar en **Publicacion:**

- *responderPregunta(preg_id: int, resp: string)*
  Crea una instancia de **Respuesta** y la asocia a la **Pregunta** con id **preg_id,** la que deja de estar en el rol de *pendientes* para pasar a estar en *preguntas*. Asuma existencia de la instancia de **Pregunta** con id **preg_id**.

Nota: Las preguntas y respuestas deben tener registro de la fecha en que fueron realizadas. Para obtener la fecha del sistema basta con crear una instancia de la clase **Time**, la cual queda con la fecha del sistema si se usa el constructor por defecto.

Observaciones:
- No incluir directivas al precompilador.
- Puede suponer la existencia de implementaciones de IDictionary, ICollection,
- IIterator y KeyInt según sea necesario.
- Es posible utilizar las clases set<T> y map<K,V> de la STL.