Programación 4

EXAMEN JULIO 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 (30 puntos)

Parte a)

En la Orientación a Objetos, ¿es posible obtener polimorfismo sin herencia? Justifique.

Parte b)

Las empresas de software típicamente se dividen en dos tipos: empresas de producto y empresas de servicio. Las primeras cuentan con un producto de software previamente desarrollado que se busca vender tal cual está a la mayor cantidad de clientes posibles (ej: una "app") mientras que las segundas se dedican al desarrollo de software a medida para clientes medianos y grandes (ej: una software factory).

Es muy común que las empresas de servicios les vendan a otras empresas de servicios (ej: una tercerización) así como a empresas de producto (ej: ayudándolas a desarrollar su producto). Estas ventas se representan como contratos, los cuales establecen la fecha del contrato, la empresa proveedora (siempre de servicios) y la empresa cliente, así como fechas de inicio y fin del contrato y un código identificatorio del contrato y un nombre descriptivo.

Los desarrolladores (identificados por cédula y con un nombre) por su parte trabajan en proyectos que se enmarcan en dichos contratos, debiendo saberse la fecha en la que inició su trabajo en ese proyecto y la fecha en la que lo finalizó, de forma de poder luego ser asignado a otro proyecto.

Considere el siguiente Caso de Uso:

Caso de Uso:	Alta Proyecto
Actores:	Usuario
Descripción:	El CU comienza cuando el usuario solicita un listado de todos los contratos activos del Sistema (aquellos cuya fecha de finalización son posteriores a la fecha actual). Luego el usuario selecciona uno de esos contratos al cual le quiere asignar el nuevo proyecto. Luego el usuario solicita una lista de todos los desarrolladores y, para cada uno que desee agregar al proyecto, se consulta su disponibilidad (es decir si el desarrollador se encuentra sin asignación en las fechas del contrato). Solo se permitirá agregar desarrolladores que estén disponibles. El CU finaliza cuando el usuario confirma el alta del nuevo proyecto, con los desarrolladores (disponibles) elegidos, y eligiendo un nombre para dicho proyecto.

i) Se pide: realice un Modelo de Dominio de esta realidad con restricciones en lenguaje natural.

ii) Se pide: realice el Diagrama de Secuencia del Sistema (DSS) correspondiente incluyendo el uso de datatypes y memoria del Sistema en caso de ser necesario.

Problema 2 (35 puntos)

Parte a)

ι. ¿Qué son los criterios GRASP? Defina en una frase.

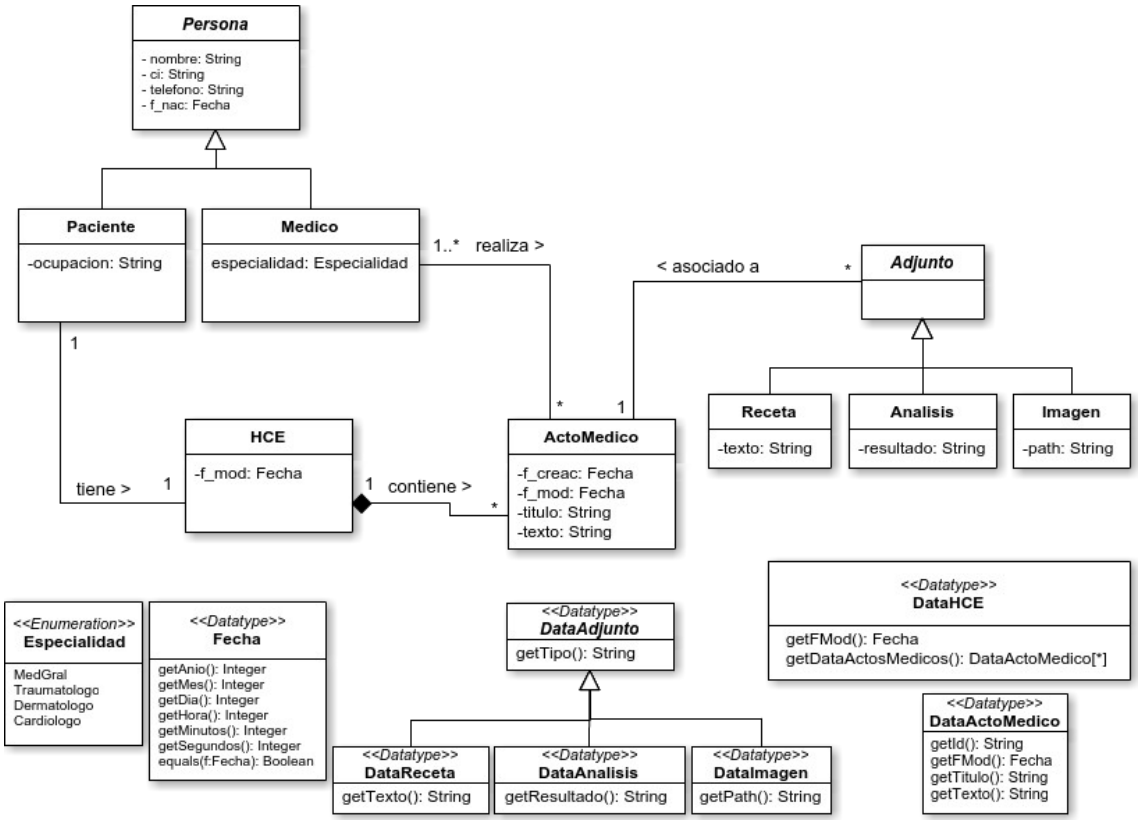
ιι. Nombre (sin describir) tres criterios GRASP.

Parte b)

Una mutualista pequeña le plantea realizar un sistema de gestión de historias clínicas electrónicas (HCE). Como cuenta con poco presupuesto, su equipo y la mutualista acuerdan realizar un sistema básico.

El sistema registrará los Pacientes y Médicos de la mutualista. Cada Paciente tiene una HCE, que contiene los Actos Médicos realizados. Cada Acto Médico puede tener asociados Adjuntos, cada uno de los cuales puede ser una Receta, el resultado de un Análisis clínico o una Imagen médica (como por ejemplo una radiografía). Interesa particularmente saber qué Médico realizó cada Acto Médico, y que cada Médico sepa qué Actos Médicos realizó.

Su equipo de Analistas realizó el siguiente modelo de dominio:



Obs: Persona, Adjunto y DataAdjunto son abstractos.

Restricciones:

- No existen dos Actos Médicos con mismo valor de *f_creac* realizados por el mismo Médico.

Además se generaron los siguientes contratos, que corresponden a operaciones del sistema necesarias para realizar algunos de los casos de uso relevados:

generarActoMedico (ciMedico, ciPaciente, titulo, texto: String)	
Descripción	Registra en el Sistema un Acto médico realizado por el médico con cédula *ciMedico,* en la HCE del Paciente con cédula *ciPaciente.*
Parámetros	• ciMedico: Cédula del Médico que realiza el Acto médico. • ciPaciente: Cédula del Paciente. • titulo: Título del Acto médico. • texto: Descripción detallada del acto médico.
Pre-condiciones	- Existe en el Sistema un Paciente con cédula *ciPaciente.* - Existe en el Sistema un Médico con cédula *ciMedico.* - *titulo* es una cadena de texto no vacía. - *texto* es una cadena de texto no vacía.
Post-condiciones	• El Sistema genera una nueva instancia de ActoMedico. Los valores de *f_creac* y *f_mod* son iguales a la actual del Sistema. • El Sistema vincula el ActoMedico del punto anterior a la HCE del Paciente con ci igual a *ciPaciente,* y al Medico de ci *ciMedico.* • El valor del campo *f_mod* de la HCE del punto anterior pasa a ser la fecha actual del Sistema.

agregarAdjuntosActoMedico (ciMedico: String, f_creac: Fecha, adjuntos: set(DataAdjunto))	
Descripción	Agrega una lista de Adjuntos al Acto médico con fecha de creación *f_creac,* realizado por el médico con cédula *ciMedico.*
Parámetros	• ciMedico: Cédula del Médico que realiza el Acto médico. • f_creac: Fecha en que se realizó el Acto médico • adjuntos: Lista de Adjuntos a agregar en el Acto médico.
Pre-condiciones	- Existe en el Sistema un Médico con cédula *ciMedico.* - El Médico con cédula *ciMedico* tiene un link a una instancia de ActoMedico con fecha de creación *f_creac.*
Post-condiciones	• Se crean nuevas instancias de Receta, Analisis o Imagen, según corresponda, a partir de los DataAdjunto recibidos en el parámetro *adjuntos* • Cada una de las instancias creadas se linkea al ActoMedico con fecha de creación *f_creac,* realizado por el médico con cédula *ciMedico.* • Se actualiza el valor del campo *f_mod* de la instancia de ActoMedico del punto anterior con la fecha actual del Sistema.

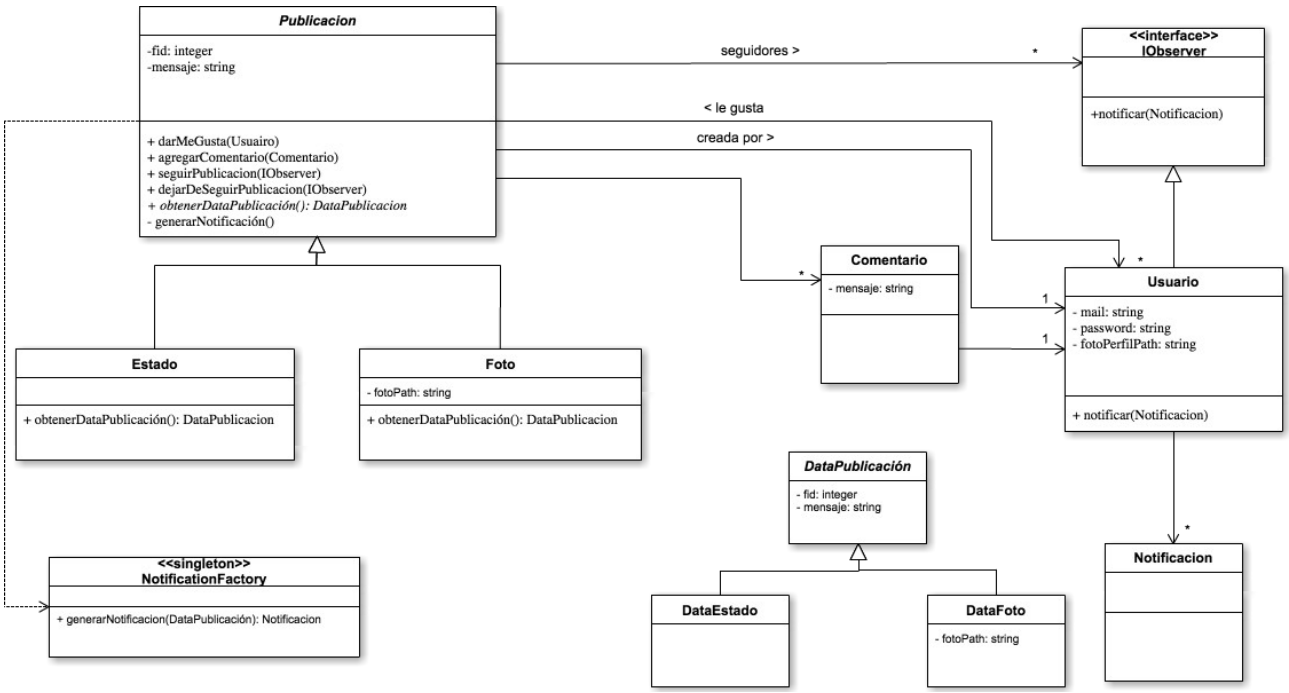
Se pide:

i. Realizar el Diagrama de Comunicación (incluyendo visibilidades) para cada una de las operaciones previamente declaradas.

ii. Realizar el Diagrama de Clases de Diseño correspondiente.

Problema 3 (35 puntos)

Se desea realizar una red social similar a la red social Facebook y para esto el equipo de diseño llegó al siguiente diagrama de clases:



Considerar:

- Puede suponer la existencia de la interface ICollectible e implementaciones de ICollection (clase List) e IIterator según sea necesario.
- Es posible utilizar las clases set<T> o vector<T> de la STL.
- Las implementaciones **deben** incluir constructores, destructores.
- Asumir existencia y no implementar los get y set de los atributos.
- **No** incluir directivas al precompilador.
- **No** implementar los datatypes.

Se pide:

- Implementar los .h de las clases Publicación, Foto, IObserver, Usuario y NotificationFactory
- Implementar los .cpp de las clases
  - Publicación, teniendo en cuenta:
    - Tiene un constructor que toma un Usuario siendo éste el creador.
    - El Usuario creador queda suscrito a la Publicación.
    - Cuando un Usuario da me gusta o agrega un comentario a una Publicación, queda suscrito a la Publicación.
    - Cuando un Usuario da me gusta o agrega un comentario a una Publicación, son notificados todos los Usuarios suscritos incluyendo al Usuario que da me gusta o comenta.
    - *seguirPublicacion* agrega el objeto pasado como parámetro de tipo IObserver a la colección de seguidores.
    - *dejarDeSeguirPublicación* elimina el objeto pasado como parámetro de tipo IObserver de la colección de seguidores.
    - Para generar las notificaciones se deberá utilizar la función privada *generarNotificacion()*, que utiliza *NotificationFactory* para obtener la notificación.
    - La operación *obtenerDataPublicacion* es una función virtual pura.
    - El destructor debe destruir los comentarios asociados.
  - Foto, teniendo en cuenta:

- La operación *obtenerDataPublicacion* debe retornar un *datavalue* de tipo *DataFoto* con los datos de la instancia de *Foto* correspondiente.

- NotificationFactory, teniendo en cuenta:
  - La función *generarNotificacion(DataPublicacion)* se encuentra implementada.