Programación 4

PARCIAL FINAL EDICIÓN 2006

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 numero de parcial junto al parcial

Problema 1 (25 puntos)

a) Defina "tipo asociativo" haciendo especial énfasis en las multiplicidades de éste.
b) Se desea modelar un sistema que llevará control de los servicios brindados por un taller mecánico. Para ello, se ha elaborado el siguiente documento de Visión del problema y la descripción del caso de uso "Realización de un Servicio".

> El taller dispone de un conjunto de servicios (identificados por un nombre) destinados a realizarse sobre un vehículo cada cierto número de kilómetros (fijo), por ejemplo: afinado a los 10.000kms, revisión general a los 50.000kms, cambio de aceite a los 5.000kms, entre otros. Estos servicios serán definidos a posteriori. Cada servicio se realiza una única vez en cada vehículo cuando el kilometraje del vehículo supere el estipulado por el servicio.
>
> De los vehículos interesa saber su matrícula (que los identifica), la marca, el modelo, el kilometraje actual y si tienen motor a nafta o motor a diesel. Si es motor a nafta, interesa saber si funciona con la nafta eco-supra o no.
>
> Uno de los productos más utilizados en los diferentes servicios es el aceite, por lo que el taller define sus servicios especificando qué aceite particular se utiliza en dicho servicio (pudiendo un servicio no utilizar ninguno), por ejemplo, cambio de aceite en los 5.000kms utilizando aceite "Shell" para motores a nafta. Existen aceites para motores nafta y aceites para motores diesel (de los cuales interesa saber si son o no aceites turbo). De todos ellos interesa saber su nombre (que lo identifica) y el fabricante del aceite. El taller conoce el aceite específico que utiliza cada vehículo, debiendo respetarse que los vehículos a nafta utilicen aceites que estén diseñados para motores a nafta, lo mismo con los diesel. Además, se debe asegurar que los servicios que utilizan determinado aceite se apliquen a vehículos que utilicen ese mismo aceite.
>
> El costo de realizar un servicio es diferente para cada vehículo y para cada servicio. En particular los servicios que utilicen un aceite del fabricante "Shell" tendrán un costo superior a los 2500 pesos.

Caso de Uso:	Realización de un Servicio	Actor:	Mecánico
Descripción:	Este CU comienza cuando el cliente ingresa al taller con su vehículo. El mecánico lo recibe, le pregunta el servicio que desea realizar e ingresa la matrícula del vehículo y el nombre del servicio a realizar al sistema. Éste luego registrará el servicio a realizar, incluyendo la fecha actual, el nombre del mecánico que lo realizará y el costo total del servicio.		

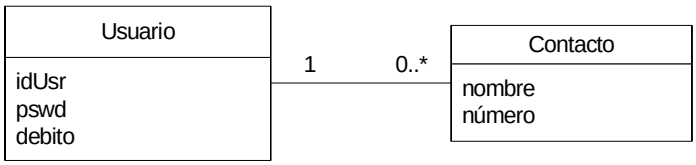
SE PIDE:

Construir el Modelo de Dominio de toda la realidad (incluyendo el caso de uso presentado) y presentarlo en un diagrama utilizando UML. Las restricciones deben ser expresadas en lenguaje natural y en OCL.

Problema 2 (25 puntos)

a) Conteste brevemente qué especifica y cómo está estructurado un Contrato.

b) Está en construcción un sistema de telefonía por Internet que permite a los usuarios registrados realizar llamadas a teléfonos fijos, cuyo cobro se realiza por medio de una tarjeta de crédito. Además, el sistema permite que los usuarios mantengan una lista de contactos personales a los efectos de facilitar la realización de llamadas. Un modelo de dominio preliminar es el que se muestra a continuación.



El principal caso de uso es el que permite realizar una llamada, especificado a continuación:

Nombre	Realizar una Llamada	Actores	Usuario
Descripción	El caso de uso comienza cuando el usuario se conecta al sistema indicando su nombre de usuario (que lo identifica) y contraseña, tras lo cual el sistema lo valida. Si el usuario es válido, puede realizar tantas llamadas como desee. Para realizar una llamada, el usuario ingresa de a uno los dígitos del teléfono al cual desea llamar y el sistema verifica que el número que se está ingresando sea valido, informando al cliente de ello. Si el ingreso es inválido, el cliente puede consultar opcionalmente los contactos (nombre y teléfono) que posee en su lista de direcciones antes de intentar realizar una nueva llamada. Si el ingreso es válido y el usuario terminó de ingresar el número de teléfono (se supone que el número existe), el usuario indica al sistema que llame a lo cual el sistema inicia la conversación. Cuando el cliente decide terminar la misma, avisa de esto al sistema y éste devuelve los datos de la llamada realizada (duración, número llamado y costo total de la llamada), pudiendo luego realizar una nueva llamada. Cuando el usuario no desea realizar más llamadas, se desconecta del sistema. El sistema devuelve el monto total de las llamadas realizadas e ingresa en la cuenta del usuario el monto a cobrar.		

i. Realice un único Diagrama de Secuencia de Sistema para el caso de uso anterior, incluyendo toda la información contenida en el mismo.

Para el almacenamiento de los contactos de un usuario se definió el siguiente caso de uso, el cual, puede ser llevado a cabo por una única operación ***altaContacto()*.**

Nombre	Alta de un Contacto	Actores	Usuario
Descripción	El caso de uso comienza cuando el usuario ingresa su nombre de usuario y pide al sistema agregar en su lista de contactos uno nuevo. Para ello especifica un nombre para el contacto (que lo identifica) y el número de teléfono del mismo. Si el contacto no se encuentra en la lista de contactos de ese usuario, el sistema da de alta al contacto en la lista. El sistema devuelve ***true*** si el contacto se da de alta, ***false*** en caso contrario.		

ii. Especifique las pre- y post-condiciones de la operación ***altaContacto()***.

Problema 3 (25 puntos)

a) Nombrar tres criterios GRASP y explicar brevemente en qué consiste cada uno.

b) Una oficina pública que atiende trámites se apresta a implementar un sistema informático que permita a los usuarios realizar dichos trámites a través de Internet. Existen dos tipos de trámites, A y B, los cuales pasan sucesivamente por diferentes ventanillas (cada una representada por una instancia de Ventanilla). En cada ventanilla el trámite se procesa mediante la invocación a la siguiente operación (que no puede ser modificada),

```
void Tramite::proc() {
      f();
      g();
      h();
}
```

donde `f()` ejecuta código común a todos los tipos de trámites y a todas las ventanillas, `g()` ejecuta código que depende exclusivamente del tipo de trámite y `h()` ejecuta código dependiente de la ventanilla. Además se debe tener en cuenta que son los usuarios los encargados de secuenciar el pasaje de los trámites a través de las ventanillas y de invocar la operación `proc()` (comportamiento que no se debe diseñar).

SE PIDE:

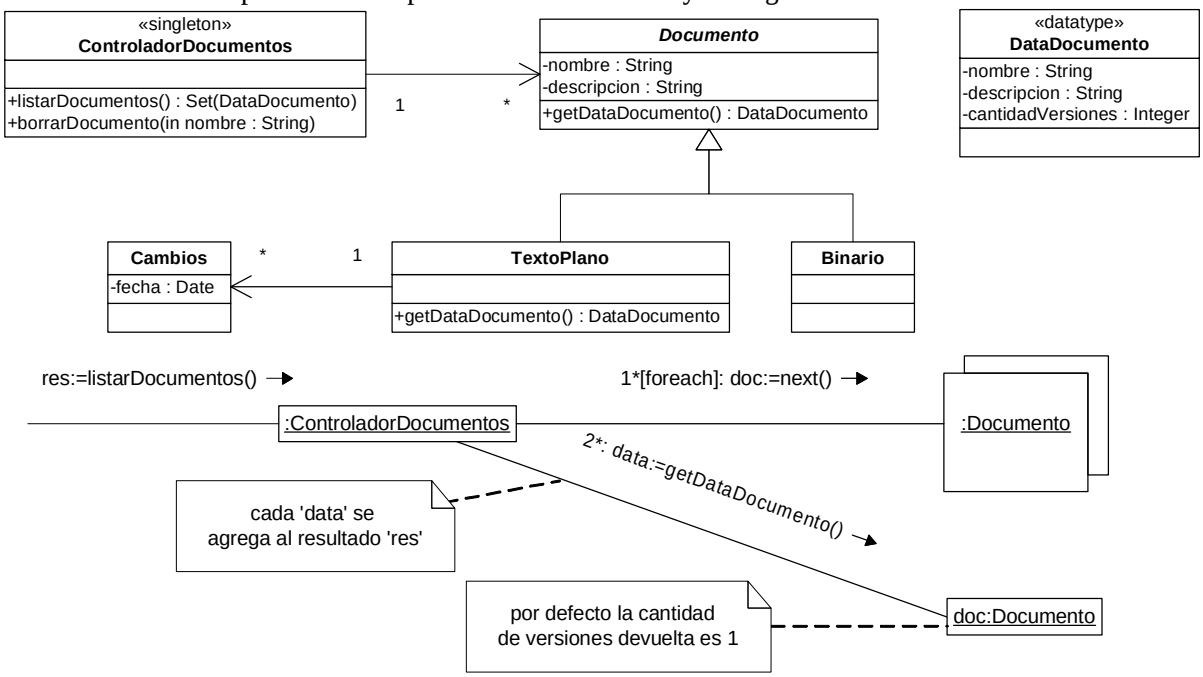
i. Diseñar la operación `proc()` (Diagrama de Clases de Diseño y Diagrama de Comunicación) teniendo en cuenta que se busca minimizar la duplicación de código, respetando las consideraciones de diseño dadas.

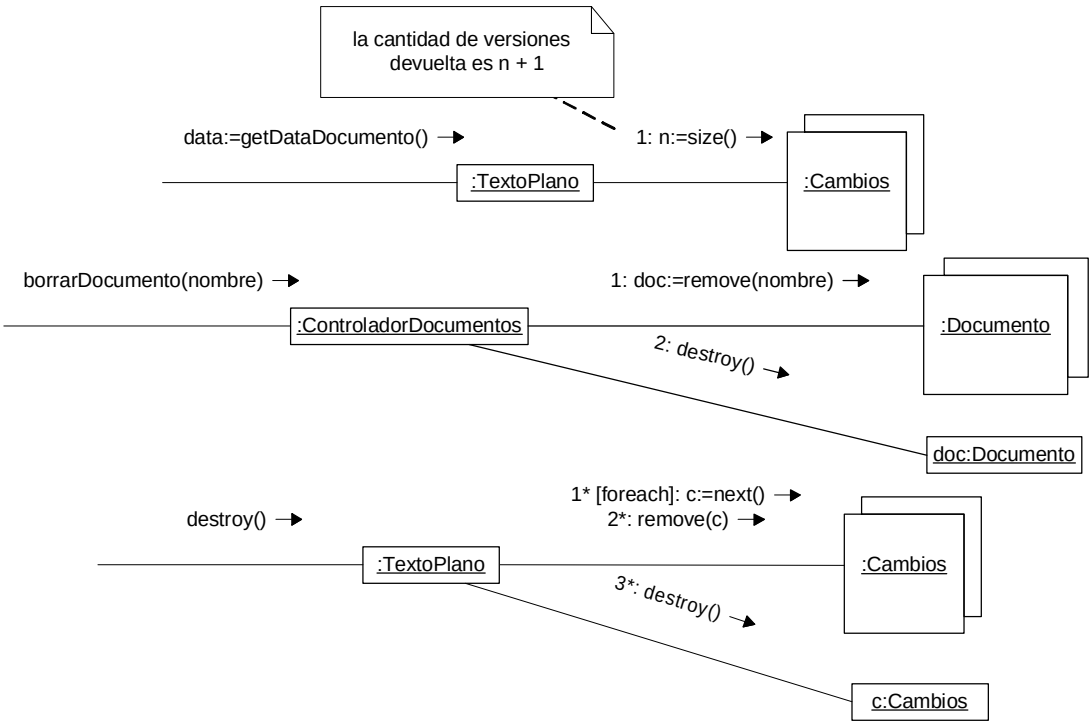
ii. Explicar qué patrón(es) de diseño utilizó indicando (para cada patrón) las clases participantes y sus roles.

Problema 4 (25 puntos)

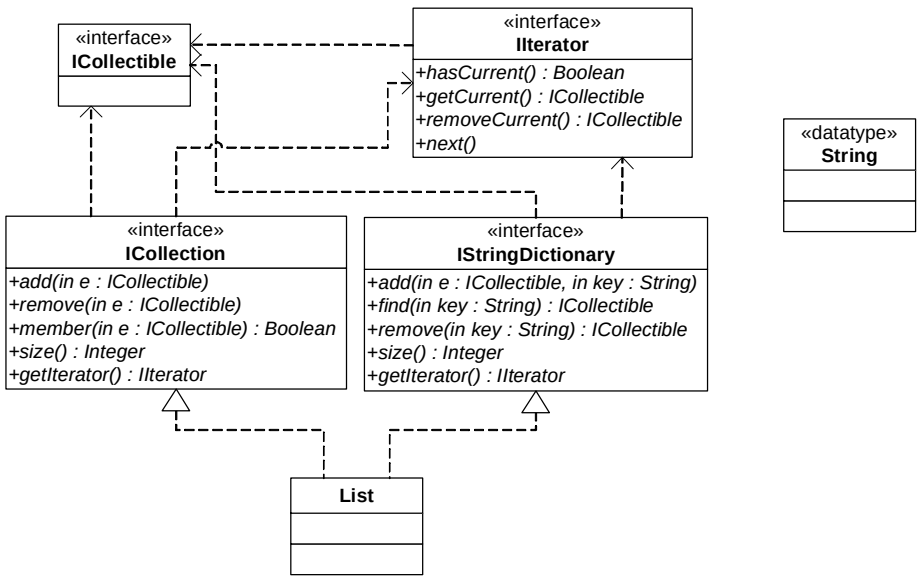
a) Defina brevemente cómo se mapean a C++ las relaciones de generalización, realización y dependencia.

b) Se desea construir un repositorio de documentos. El mismo debe ser capaz de manejar tanto documentos en formato binario (por ejemplo PDF's) como documentos de texto plano. De los últimos es necesario mantener registrados todos los cambios que hayan sufrido desde su creación. Como parte de la etapa de diseño se construyó la siguiente colaboración:





Se cuenta con las siguientes clases ya implementadas como parte de la infraestructura:



SE PIDE:

Implementar en C++ completamente la colaboración salvo la clase `Cambios`. Incluir el código relativo al patrón Singleton en la clase `ControladorDocumentos`. Del datatype `DataDocumento` es necesario incluir únicamente su módulo de definición, es decir su `.h`.

OBSERVACIONES:

i. No es necesario definir colecciones concretas.
ii. No es necesario incluir directivas al preprocesador en el código.
iii. Le operación `removeCurrent()` de la interfaz `IIterator` remueve el elemento actual de la colección y lo devuelve. A la vez posiciona el iterador en el elemento siguiente al removido.