Programación 4

EXAMEN FEBRERO 2010

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 número de examen junto al examen.

Problema 1 (30 puntos)

a) Mencione 3 construcciones de los Diagramas de Estructura Estática de UML que NO son pertinentes en los Modelos de Dominio.

b) Un establecimiento agropecuario le ha encomendado la tarea de desarrollar un software para el manejo de sus rodeos.

Un rodeo es un conjunto disjunto de animales (para el caso de este establecimiento: vacas y toros). Todos los animales se encuentran identificados por su caravana (la cual escaneada con el dispositivo adecuado, emite el número identificatorio del animal). Dado que el establecimiento reporta leche a CONAPROLE, es de interés saber la producción promedio de leche de cada vaca, la cual varía de vaca en vaca. Si bien el establecimiento no cuenta con grandes reproductores, le interesa llevar el control de cuántos toros posee dentro de cada rodeo así como la ascendencia de cada uno de ellos por parte paterna (lo cual luego permite a los expertos identificar los mejores toros para cruzar con las vacas).

Al cruzamiento entre toro y vaca se lo conoce como entore, y el establecimiento desea conocer todos los entores ocurridos, y para cada uno, entre qué animales fue, cuándo ocurrió y qué resultado tuvo. Debido a que el establecimiento mantiene por largos períodos de tiempo a sus animales, puede ocurrir que una vaca sea entorada muchas veces por el mismo toro, siempre del mismo rodeo. El resultado de un entore puede ser nulo (es decir que la vaca no se preñó) o puede ser un nuevo animal (es decir que la vaca efectivamente se preñó y dió a luz un nuevo animal). En este último caso (entore efectivo) interesa también saber qué animal fue producto de qué entore.

Se pide:

i. Modelar la realidad planteada mediante un Diagrama de Modelo de Dominio UML.

ii. Expresar todas las restricciones del modelo en lenguaje natural.

iii. A partir del modelo definido en la primera parte, exprese la siguiente restricción en OCL: “no hay dos entores de la misma vaca y toro en la misma fecha”.

iv. ¿Qué modificaciones haría al Modelo de Dominio si el resultado de un entore efectivo fueran muchos animales y no uno solo?

v. ¿Qué modificaciones haría al Modelo de Dominio de la parte i) si se desea saber cuáles fueron todos los toros con los cuales fue entorada una vaca, independientemente si el entore fue efectivo o no?

Problema 2 (35 puntos)

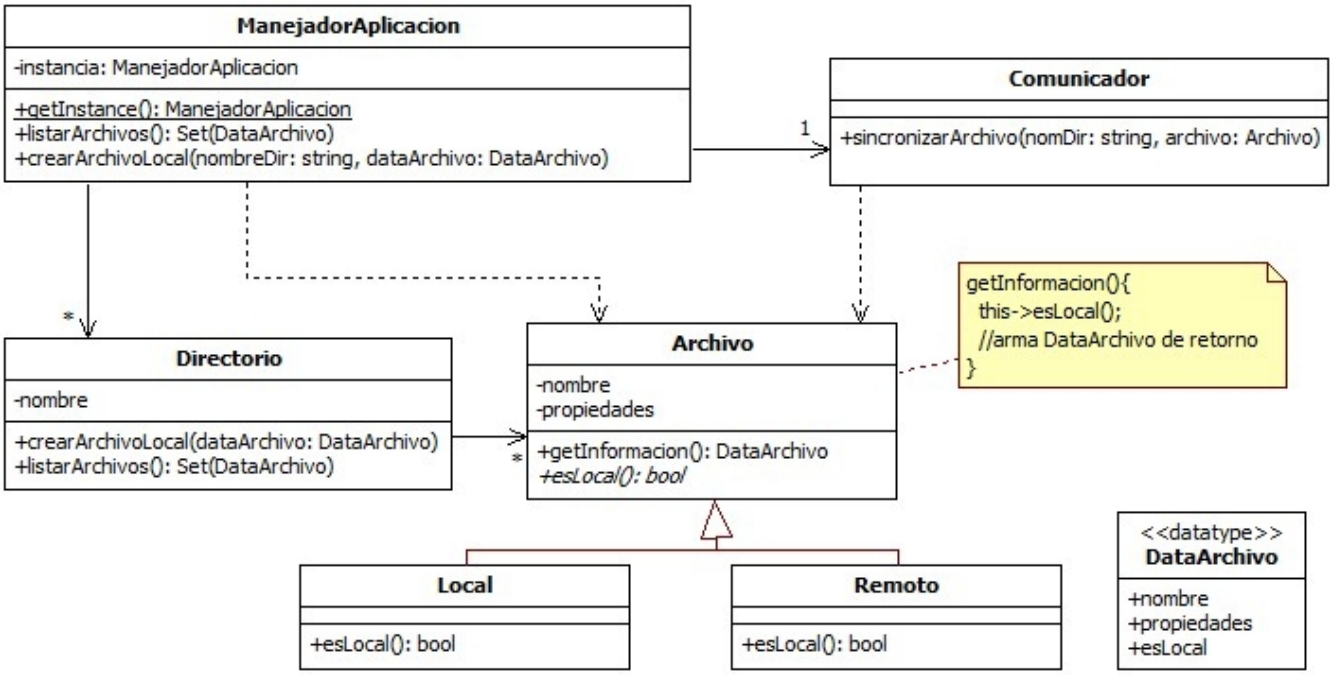
Un equipo de desarrollo está realizando el diseño de una aplicación para la sincronización de documentos a través de una red. La aplicación permite a un conjunto de usuarios crear directorios compartidos en un sistema de archivos y sincronizar su contenido entre todos los equipos de la red que tengan instalado el producto en cuestión.

La aplicación se basa en la clase `ManejadorAplicacion` que se encarga de ofrecer el comportamiento principal del sistema. En particular, se han definido los siguientes contratos parciales para la primera etapa de construcción de la aplicación.

Operación	`crearArchivoLocal(nombreDir:string, data:DataArchivo)`
Descripción	Se crea un archivo local dentro de una de los directorios registrados en el sistema. Se asume que el directorio existe, no así el archivo. Luego, se comunica a las otras instalaciones del sistema existentes en la red de la existencia de este nuevo archivo. Para ello se hace uso de la clase `Comunicador` que simplifica el envío y recepción de mensajes a través de la red. Se utiliza en este caso la operación `sincronizarArchivo(…)`.

Operación	`listarArchivos():Set(DataArchivo)`
Descripción	El sistema retorna la información de todos los archivos del sistema indicando en cada caso el nombre del archivo, sus propiedades y si el archivo es local o remoto.

En la siguiente figura se muestra el Diagrama de Clases de Diseño de la solución diseñada para el funcionamiento de la aplicación.



Se pide:

i. Realizar los Diagramas de Comunicación de las operaciones definidas anteriormente.

ii. Si utilizó algún patrón de diseño, especifique su nombre, clases participantes y roles.

La sincronización de la información de archivos remotos es similar a la creación de archivos locales salvo que una vez creado el archivo remoto se comienza automáticamente la transferencia de información desde el archivo original. Dado que no hay límite en el tamaño del archivo, la transferencia puede durar mucho tiempo por lo que eventualmente la operación podrá finalizar antes de que la sincronización se complete. Por tal motivo, se desea modificar la operación `listarArchivos():Set(DataArchivo)` para retornar, además de la información original, un booleano que indique si el contenido del archivo se está transfiriendo o si la transferencia fue completada.

Se pide:

iii. Modificar el Diagrama de Clases de Diseño original de manera tal de soportar este nuevo requerimiento con la restricción de que no se puede alterar las clases `Local` ni `Remoto`.

iv. Indique qué patrón de diseño utilizó, especifique su nombre, clases participantes y roles.

Para finalizar el desarrollo de la aplicación se incluirá la posibilidad de definir permisos sobre los archivos, tanto locales como remotos. La información que se brinda en el listado de archivos podrá variar dependiendo del tipo de permiso que tenga un archivo. Los permisos pueden variar con el tiempo y pueden agregarse nuevos tipos de permisos, por lo que el diseño debe ser lo más genérico posible como para soportar estos requerimientos y no modificar la estructura ya diseñada.

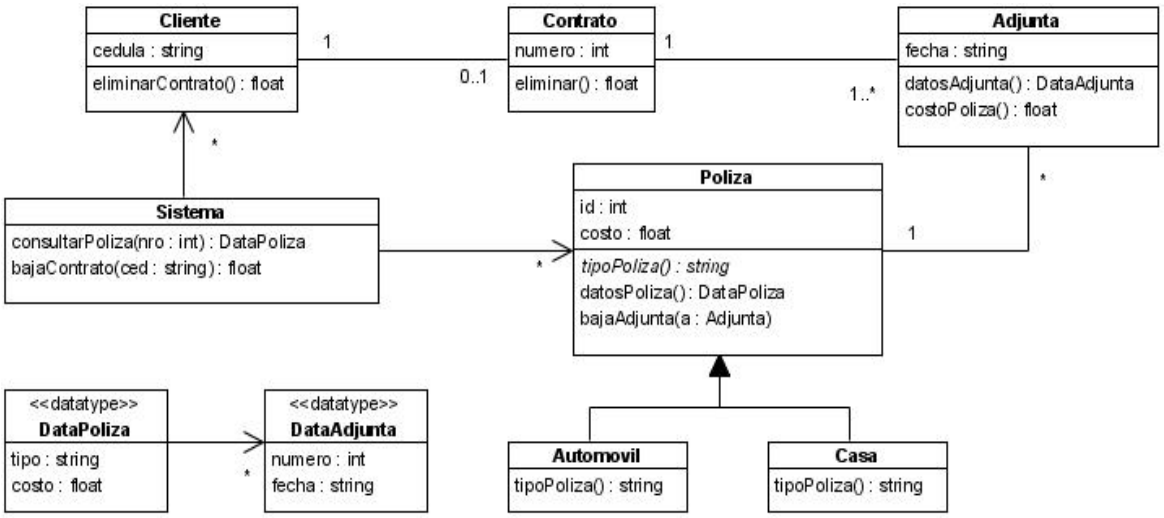
Se pide:

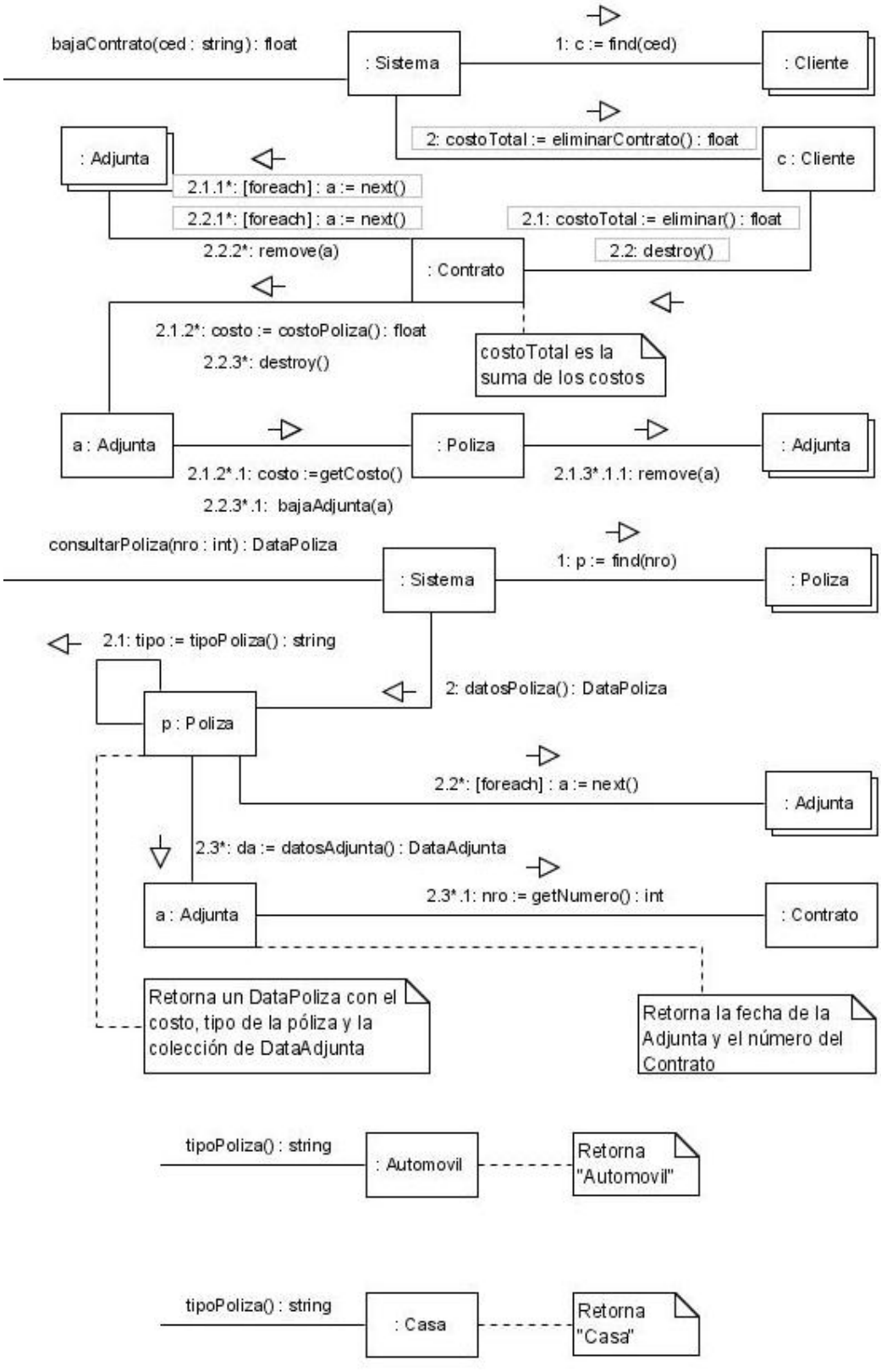
v. Modificar el Diagrama de Clases de Diseño original de manera tal de soportar este nuevo requerimiento.

vi. Indique qué patrón de diseño utilizó, especifique su nombre, clases participantes y roles.

Problema 3 (35 puntos)

Durante la construcción de un sistema de gestión de pólizas de seguros se logró el diseño parcial que se presenta a continuación, que incluye los Diagramas de Comunicación y el Diagrama de Clases de Diseño. El sistema registra los contratos firmados por clientes donde se adjuntan diferentes pólizas de seguros.




bajaContrato(ced : string) : float
: Sistema
1: c := find(ced)
: Cliente
2: costoTotal := eliminarContrato() : float
c : Cliente
: Adjunta
2.1.1*: [foreach] : a := next()
2.2.1*: [foreach] : a := next()
2.1: costoTotal := eliminar() : float
2.2.2*: remove(a)
2.2: destroy()
: Contrato
2.1.2*: costo := costoPoliza() : float
2.2.3*: destroy()
costoTotal es la suma de los costos
a : Adjunta
: Poliza
: Adjunta
2.1.2*.1: costo :=getCosto()
2.1.3*.1.1: remove(a)
2.2.3*.1: bajaAdjunta(a)
consultarPoliza(nro : int) : DataPoliza
: Sistema
1: p := find(nro)
: Poliza
2.1: tipo := tipoPoliza() : string
2: datosPoliza() : DataPoliza
p : Poliza
2.2*: [foreach] : a := next()
: Adjunta
2.3*: da := datosAdjunta() : DataAdjunta
2.3*.1: nro := getNumero() : int
a : Adjunta
: Contrato
Retorna un DataPoliza con el costo, tipo de la póliza y la colección de DataAdjunta
Retorna la fecha de la Adjunta y el número del Contrato
tipoPoliza() : string
: Automovil
Retorna "Automovil"
tipoPoliza() : string
: Casa
Retorna "Casa"

Se pide:

i. Implemente en C++ el `.h` de la clase `Sistema`

ii. Implemente en C++ el `.cc` de la clase `Sistema`, salvo la operación `consultarPoliza()`. Incluya código de manejo de excepciones en la operación `bajaContrato()`para el caso en que no exista el cliente cuya cédula se utiliza en la eliminación.

iii. Implemente en C++ los `.h` de la jerarquía de clases compuesta por `Poliza`, `Automovil` y `Casa`. No incluya constructores, destructores ni operaciones `set` y `get` de atributos.

iv. Implemente en C++ la operación de la clase `Poliza::DataPoliza datosPoliza()`.

v. Implemente en C++ el destructor de la clase `Contrato`.

Observaciones:

- Asuma que existe una implementación estándar de las interfaces `ICollectible`, `ICollection`, `IIterator`, `IDictionary` e `IKey` y que existe una clase `Lista` que realiza las interfaces `ICollection` e `IDictionary` y una clase `KeyString` que realiza la interfaz `IKey`. No defina colecciones concretas.
- Asuma la existencia de una clase `String`
- No incluya directivas de preprocesador.
- No incluya la eliminación de la memoria temporal utilizada en el código C++.