Programación 4

EXAMEN - DICIEMBRE 2007

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 examen junto al examen

Problema 1 (25 puntos)

a)

i. Indicar y definir cuáles son las posibles relaciones entre dos conceptos de un Modelo de Dominio.

ii. Definir qué es una invariante del Modelo de Dominio e indicar tres tipos de invariantes habituales.

b) Construya un Diagrama de Modelo de Dominio UML para la siguiente realidad. Las restricciones deben ser expresadas en lenguaje natural y en OCL. Modele **exclusivamente** en base a la información presente en la descripción.

Una Empresa Productora de Televisión desea realizar un software que le permita gestionar varios concursos que la misma produce. Cada uno de estos concursos está formado por un conjunto de parejas que compite en una disciplina determinada (baile, canto, etc.), pudiendo existir varios concursos para una misma disciplina.

Un concurso se identifica por el nombre y tipo de disciplina, es de carácter eliminatorio y se realiza en distintas etapas que se llevan a cabo en una fecha determinada. Cabe notar que no puede haber dos etapas de un mismo concurso con la misma fecha. Como resultado de cada etapa, dos o más parejas quedarán sentenciadas, lo que habilitará a una votación del público. La votación se realiza por vía telefónica, por lo que a cada pareja se le asignará un número telefónico (que la identifica) al comenzar el concurso. La pareja sentenciada con menor cantidad de llamados será eliminada del concurso.

Cada pareja participante está formada por un soñador y un famoso identificados por su cédula. Otros datos de interés son el nombre y la edad de los participantes. Para los soñadores se necesita saber además su ciudad natal, si es profesional y una descripción del sueño a cumplir en caso de ganar el concurso. Las reglas establecen que a un soñador sólo se le permitirá participar en un concurso mientras que los famosos pueden participar en más de uno. No está permitido que un famoso pueda participar con más de un soñador de un mismo concurso.

Notas:

- Para la comparación de fechas en OCL se asume la existencia de un tipo Fecha que soporta los operadores habituales de comparación (=, <, >, <=, >= y <>)

Problema 2 (25 puntos)

a) Conteste brevemente qué es un contrato de software y cómo se relaciona con los DSS.

b) Actualmente la ciudad de Montevideo (y en particular la IMM) se encuentra muy cerca de implementar el nuevo Sistema de Transporte Metropolitano (STM). Este cambio está orientado a mejorar la movilidad de los ciudadanos (usuarios del nuevo sistema de transporte) en todo el departamento y tiene previstas ampliaciones para toda el área metropolitana.

Este nuevo sistema incorpora la utilización de nueva tecnología: la tarjeta inteligente. Esta tarjeta inteligente será utilizada por los usuarios del sistema para realizar viajes a través de la metrópolis, así como para recargar el saldo de la misma con dinero en efectivo.

Es sobre estos dos últimos puntos que se encuentra en vías de análisis el sistema informático que dará soporte a la utilización de dichas tarjetas. Este nuevo sistema permitirá, por un lado, la Recarga de una Tarjeta con dinero, y por otro lado, el Uso de la Tarjeta.

A continuación se presentan los Casos de Uso que el equipo de trabajo actual ha generado para cada uno de los puntos anteriores:

Nombre:	Recarga de Tarjeta	**Actor**:	Usuario del STM
Sinopsis:	Este caso de uso comienza cuando el usuario ingresa el PIN de su tarjeta inteligente. Si éste es correcto, el usuario podrá ingresar la cantidad de dinero que desee cuantas veces quiera. Cada vez que se ingrese un cierto monto de dinero el sistema irá acreditando ese monto a la tarjeta en cuestión. Finalmente el usuario terminará la recarga de la tarjeta y el sistema le devolverá el nuevo saldo de la tarjeta (en pesos).		

Nombre:	Uso de Tarjeta	**Actor**:	Usuario del STM
Sinopsis:	Este caso de uso comienza cuando el usuario ingresa el PIN de su tarjeta inteligente. Si éste es correcto, el usuario deberá pedir un listado con las modalidades de viajes disponibles en ese momento. El listado de opciones de modalidades está compuesto por un número no conocido a priori de opciones conteniendo: número de opción (por ejemplo: 1), descripción de la opción (por ejemplo: viaje metropolitano), el precio de esa modalidad para un viaje de 1 hora (por ejemplo: 10 pesos) y el precio de esa modalidad para un viaje de 2 horas (por ejemplo: 15 pesos). Una vez que el usuario ingresa el número de opción de la modalidad deseada de viaje así como la cantidad de boletos a comprar, deberá elegir por cuánto tiempo desea viajar, existiendo siempre dos tipos de duración: viaje por 1 hora y viaje por 2 horas. Sólo para el caso de viajar por 2 horas se debe indicar si se realizará trasbordo o no, ya que esto si bien no incide en el precio, debe constar en el boleto a imprimir. Luego de seleccionar el tiempo del viaje el Sistema controla que el saldo de la tarjeta sea suficiente según lo seleccionado, y en caso de serlo el usuario confirma la compra y de lo contrario cancela la compra. Ya sea que se acepte o cancele el viaje, el Sistema devolverá el saldo final de la tarjeta.		

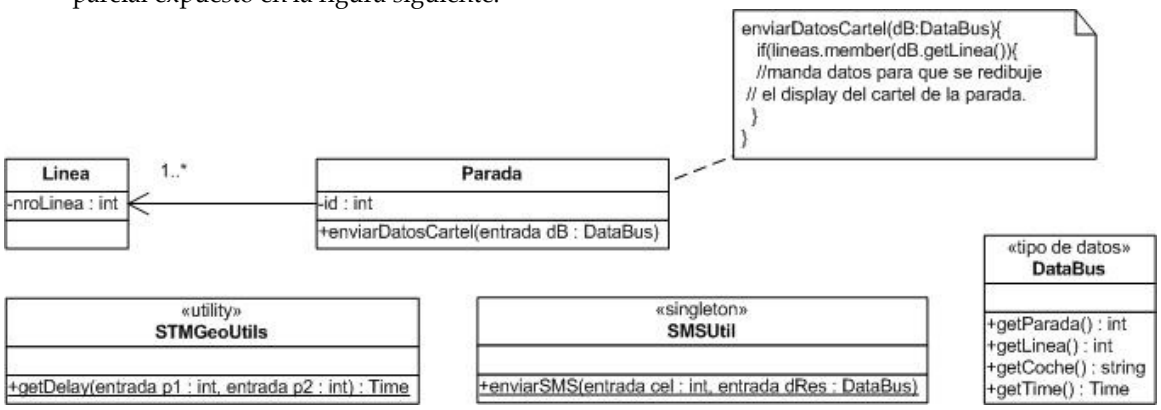
Se pide: Realice un Diagrama de Secuencia del Sistema por cada Caso de Uso. Incluya los tipos de los parámetros y valores de retorno de las operaciones.

Problema 3 (25 puntos)

a)
   i. ¿Qué son y para que sirven los patrones de Diseño? Justifique.
   ii. Defina bajo acoplamiento y alta cohesión. Cite dos ejemplos que aclaren dichos conceptos.

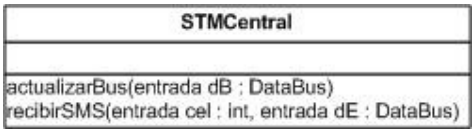
b) La IMM está implementado un nuevo STM (sistema de transporte metropolitano). Todos los ómnibus de Montevideo contarán con un Global Position System (GPS) a bordo, el cual se utilizará para poder ubicar geográficamente lo mas preciso posible a cualquier ómnibus en un momento dado. Dicho artefacto esta programado de manera tal que enviará información a la central cada vez que el ómnibus llegue a una parada. Aprovechando esta nueva capacidad, es que se desea ofrecer un nuevo servicio al ciudadano. El mismo consta de poder avisarle a su celular cuánto falta para que pase el ómnibus que él esta esperando. El sistema funciona de la siguiente manera.

El usuario envía un SMS a cierto número (el cual no es relevante), con los siguientes datos: la línea de ómnibus que le interesa y el identificador de la parada donde lo piensa tomar (`DataBus`). Segundos más tarde recibirá una respuesta del sistema con el número de coche, numero de línea y el tiempo estimado que tardará pasar por la parada consultada (`DataBus`). Para ser aún más interactivo el transporte es que en todas las paradas existirán carteles electrónicos que mostrarán una lista por orden de demora, de los ómnibus que están por pasar. Dichos carteles podrán ser habilitados y deshabilitados en cualquier momento. Por limitaciones en la infraestructura, las sucesivas posiciones de los ómnibus no pueden ser mantenidas en memoria. Es decir, el sistema no recuerda la posición actual de ningún ómnibus. Se cuenta con el diseño parcial expuesto en la figura siguiente.



Las clases `Parada` y `Línea` encapsulan la lógica de envío de información a los carteles, por lo que no pueden ser modificadas y deben ser utilizadas. Se cuenta con un único datatype `DataBus` utilizado para el intercambio de información. Además, se cuenta con los siguientes elementos auxiliares: la operación `getDelay::STMGeoUtils` permite conocer el tiempo aproximado de demora entre dos paradas y la operación `enviarSMS::SMSUTIL` permite enviar un SMS de respuesta al usuario con los datos solicitados del bus.

Se desea diseñar la clase STMCentral cuyas operaciones permiten recibir un SMS de consulta de un cliente (`recibirSMS`) y recibir el número de parada a la que llegó un bus determinado (`actualizarBus`).



Se pide:

i. Realice diagramas de comunicación que muestren todas las interacciones que ocurren al ejecutarse la operaciones `recibirSMS()` y `actualizarBus()` de la clase `STMCentral`.
ii. Realice el DCD completo, incluyendo las clases que usted considere necesarias.
iii. Explique qué patrón(es) de diseño utilizó indicando (para cada patrón) las clases participantes y sus roles.

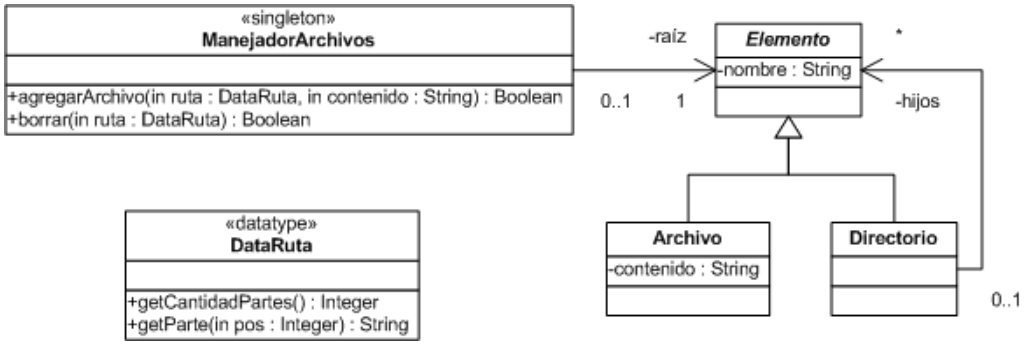
Notas:

- Las clases del diseño parcial no podrán ser modificadas.
- Se asume la existencia de un tipo Time que soporta los operadores habituales de comparación (=, <, >, <=, >= y <>)
- Puede agregar y/o quitar elementos que considere necesarios a los componentes ya diseñados.
- Tenga en cuenta que se busca minimizar la duplicación de código y tratar de definir mecanismos genéricos con la aplicación de patrones de diseño.

Problema 4 (25 puntos)

a) Defina en UML las interfaces `IDictionary`, `IIterator`, `IKey` e `ICollectible`.

b) Considere el siguiente Diagrama de Clases de Diseño parcial relativo a un sistema de manejo de archivos dentro de una estructura arborescente de directorios:



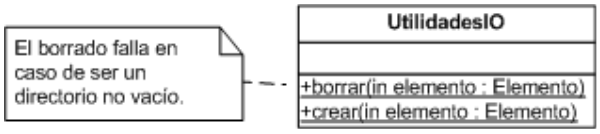
Para facilitar la tarea las rutas vienen dadas en el datatype `DataRuta` cuyas operaciones permiten acceder a cualquier trozo (o parte) de la misma. Las partes se numeran en el intervalo: `1..getCantidadPartes()`. A modo de ejemplo, si la ruta fuera `/dir1/dir2/archivo`, las siguientes expresiones serían verdaderas:

- `getCantidadPartes() == 3`
- `getParte(1) == "dir1"`
- `getParte(3) == "archivo”`

Comportamiento esperado de las operaciones de la clase `ManejadorArchivo`:

- **`agregarArchivo`**: en caso de ser necesario se deben crear los directorios que falten. Dentro de un directorio no puede existir un archivo y un directorio con el mismo nombre. Devuelve `true` si logra crear el archivo, y `false` en caso contrario.
- **`borrar`**: borra tanto un directorio (recursivamente) como un archivo. No se puede borrar la raíz. Devuelve `true` si logra borrar el objetivo y `false` en caso contrario.

La estructura de objetos en memoria debe reflejar exactamente la estructura almacenada en disco. Para impactar los cambios hechos desde el manejador se cuenta con el singleton:



Tener en cuenta que los elementos pasados como parámetro deben estar correctamente ubicados en el árbol de objetos al momento de invocar tanto `crear` como `borrar`.

Se pide:

Implemente en C++ completamente las clases del diseño anterior.

- No incluir el datatype `DataRuta` ni el singleton `UtilidadesIO`.
- Asuma que cuenta con una implementación de las interfaces definidas en la parte a) de este ejercicio.

Notas:

- No es necesario incluir directivas al preprocesador en el código.
- Asuma que cuenta con una implementación del datatype `String`.