Programación 4

EXAMEN FEBRERO 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 parcial junto al parcial*

Problema 1 (25 puntos)

a)

i. Describa brevemente las técnicas de identificación de conceptos vistas en el curso.
ii. Muestre, mediante un ejemplo sencillo, como se define utilizando UML un tipo no primitivo y un atributo cuyo tipo es el tipo no primitivo definido anteriormente.

b)

Una empresa consultora en gestión de proyectos desea desarrollar un sistema para la planificación y control de avance de sus proyectos. Se le ha encargado a usted que realice el análisis de requerimientos asociados a sus dos fases.

i. Fase 1: Tareas Planificadas.

De cada proyecto interesa registrar su nombre (que lo identifica) y una breve descripción. Un proyecto puede tener asociados planes para su concreción. De cada plan se conoce su nombre (que lo identifica) y una breve descripción de sus objetivos. Un plan consta de un conjunto no vacío de tareas planificadas para lograr cumplir sus objetivos. De cada tarea, el sistema debe registrar su nombre (que la identifica) y su descripción. Una tarea debe pertenecer al menos a un plan. Las tareas de un plan pueden estar relacionadas con otras tareas del mismo plan de las que depende para poder ser realizada. Esta dependencia varía de plan en plan, es decir, para una misma tarea planificada, sus dependencias son distintas en los distintos planes a los que pertenece. Finalmente, el sistema debe verificar que una tarea no dependa de sí misma. Sin embargo, no es necesario que el sistema detecte dependencias circulares entre las tareas.

Se pide: Realice el modelo de dominio correspondiente a la primera fase y escriba en OCL sus restricciones.

Para la segunda fase del proyecto se agregan los siguientes requerimientos:

ii. Fase 2: Tareas Ejecutadas.

De cada proyecto interesa registrar aquellas tareas (planificadas o no) que se hayan ejecutado para la concreción del mismo. En caso que la tarea ejecutada haya sido previamente planificada, el sistema deberá registrar la relación entre la tarea ejecutada y la tarea planificada. Una tarea planificada puede ser ejecutada a lo sumo una vez. Una tarea ejecutada no necesariamente tiene que haber sido planificada.

La primera acción que el sistema debe registrar sobre una tarea ejecutada es la iniciación de su ejecución. Luego de iniciada la ejecución de una tarea, la misma puede ser suspendida o finalizada. Además, una tarea suspendida sólo puede ser reanudada mientras que una tarea finalizada no puede sufrir acciones posteriores. El sistema debe asegurar la validez de la

secuencia de acciones de una tarea. De cada una de las acciones sobre las tareas, el sistema debe registrar la fecha y la hora en la que sucedió.

Se pide: Modifique y complete el modelo de dominio para cumplir con los requerimientos adicionales de la segunda fase y escriba en OCL sus restricciones.

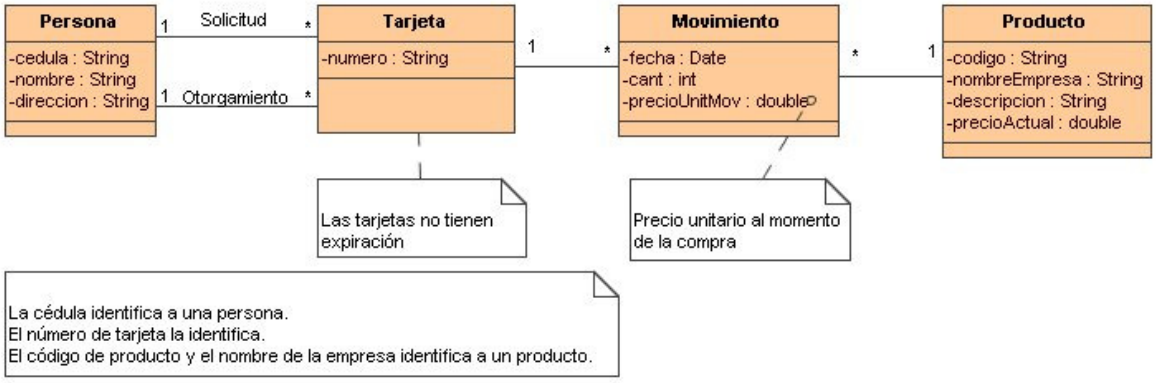
Notas:
- Asuma que dispone de un datatype FechaHora que permite representar fechas y horas y compararlas.
- Aquellas restricciones que sean necesarias en la segunda fase deben ser repetidas.

Problema 2 (25 puntos)

a) En forma breve y concisa:
   i. Explique la relación entre caso de uso, escenario y diagrama de secuencia del sistema.
   ii. Explique la relación entre caso de uso y colaboración.
   iii. Describa los 2 enfoques citados en el curso para diseñar una colaboración.

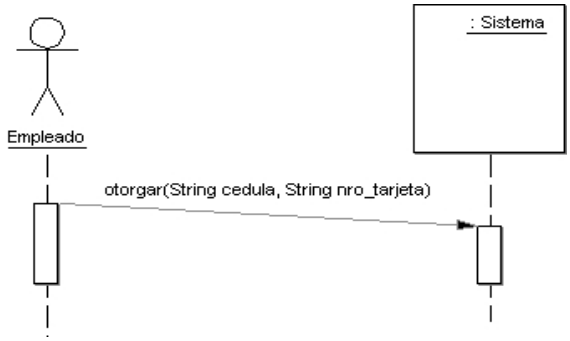
b) Se quiere terminar el análisis y realizar el diseño de una iteración de un sistema de gestión de solicitudes de tarjetas y registro de movimientos usando las mismas. A continuación se presentan el modelo conceptual y los casos de usos (junto con un diagrama de secuencia del sistema del curso típico de eventos) de esta iteración.

Modelo conceptual

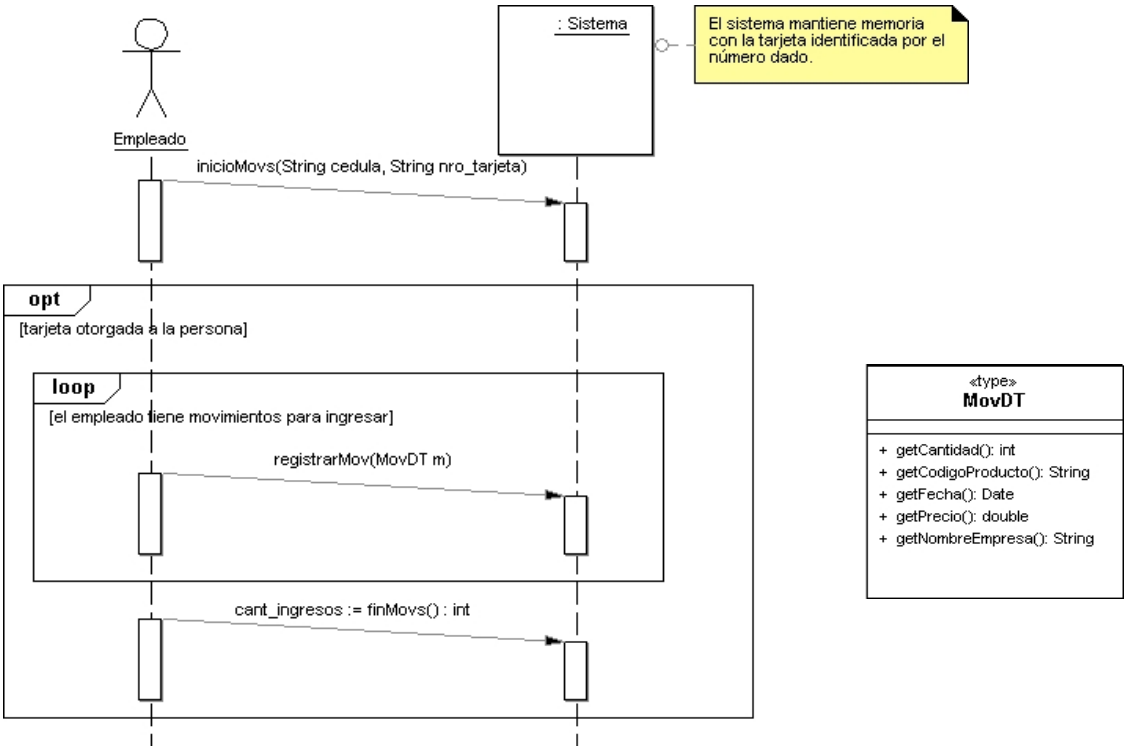


Casos de uso

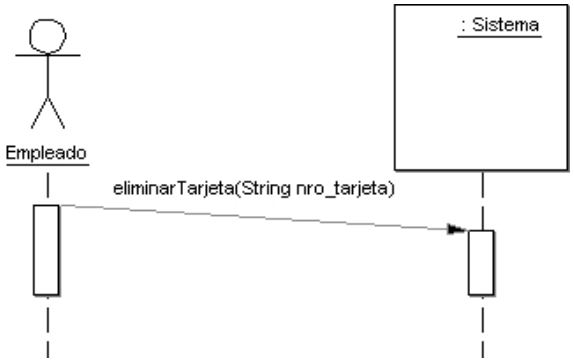
Nombre	Otorgamiento de tarjeta	**Actores**	Empleado
Sinopsis	El empleado indica al sistema el otorgamiento de una tarjeta conocida por su número a la persona identificada por su cédula de identidad. El sistema chequea la existencia de una solicitud de dicha tarjeta por la persona. En caso de que exista, se elimina la solicitud y se registra el otorgamiento. En caso contrario, la operación no debe tener efecto alguno sobre el sistema.		



Nombre	Ingreso de movimientos	**Actores**	Empleado
Sinopsis	El caso de uso comienza cuando el empleado quiere ingresar un conjunto de movimientos de compras indicando la tarjeta y la cédula de identidad de la persona que hizo la compra para que el sistema realice el chequeo de que la tarjeta ha sido otorgada a esa persona. Luego, el empleado agrega uno a uno los movimientos. Durante este proceso, si el producto al cual se refiere el movimiento no existe, se agrega al sistema. Finalmente, el empleado le indica al sistema el fin del ingreso de movimientos. El sistema le devuelve la cantidad de movimientos ingresados.		



Nombre	Baja de tarjeta	**Actores**	Empleado
Sinopsis	El empleado indica al sistema la baja de una tarjeta dando el número de la misma. El sistema elimina el registro de solicitud u otorgamiento, según corresponda. Por otro lado, todos los movimientos de la tarjeta son eliminados. La persona y los productos involucrados son mantenidos en el sistema.		



Se pide:

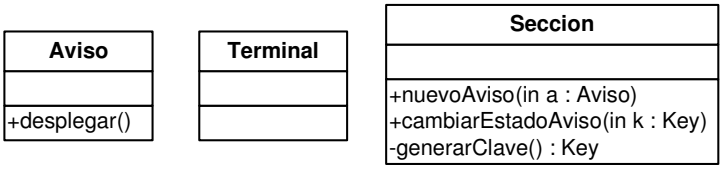
i. Especificar **claramente** las pre y post condiciones de las operaciones.
ii. Realizar los diagramas de comunicación de las operaciones teniendo en cuenta los criterios GRASP. Mencionar **explícitamente** los criterios GRASP usados.

Problema 3 (25 puntos)

a) Describir los cuatro tipos de visibilidad que pueden existir entre objetos, en el contexto de un diagrama de comunicación.

b) Un equipo de analistas trabaja en el diseño de un sistema que muestra información en pantalla a los usuarios en una estación de ómnibus interdepartamental. La información mostrada refiere a eventos relacionados con salidas de viajes programados. Existen terminales que despliegan información en pantalla. Las terminales se agrupan en distintas secciones; todas las terminales de una misma sección despliegan la misma información. Una misma terminal puede cambiar de sección frecuentemente, pero solo puede pertenecer a una sección a la vez. Cada sección mantiene de forma centralizada, un conjunto de avisos que deben desplegar sus terminales; un aviso pertenece a una sola sección. Cada vez que llega un nuevo aviso a una sección, se debe desplegar el nuevo conjunto de avisos en todas sus terminales. Existen dos tipos de terminales, tipo 1 y tipo 2, dependiendo de la tecnología de su pantalla. El procedimiento de despliegue de avisos en una terminal es el mismo para los diferentes tipos de terminales, se despliegan todos los avisos de la correspondiente sección; sin embargo existe un paso de inicialización que depende del tipo de terminal, que debe ejecutarse antes de desplegar el conjunto de avisos. Los avisos pasan por tres diferentes estados desde que se crean hasta que se destruyen, Iniciado, Confirmado, Embarcando, Partiendo y Finalizado. Al crearse un aviso, su estado es Iniciado. Existe un proceso (que no debe modelarse) que periódicamente recorre los avisos de todas las secciones, eliminando aquellos que se encuentran en estado Finalizado. El cambio de estado de un aviso se efectúa sobre la sección a la que pertenece, mediante la clave del aviso. Cada vez que un aviso de una sección cambia de estado, se deben desplegar todos los avisos de esa sección en todas sus terminales, de la misma forma que cuando llega un nuevo aviso.

En base a esta descripción se han diseñado parcialmente algunas clases, que se muestran a continuación:



Se pide:

i. Realice diagramas de comunicación que muestren todas las interacciones que ocurren al ejecutarse las operaciones nuevoAviso() y cambiarEstadoAviso() de la clase Seccion.

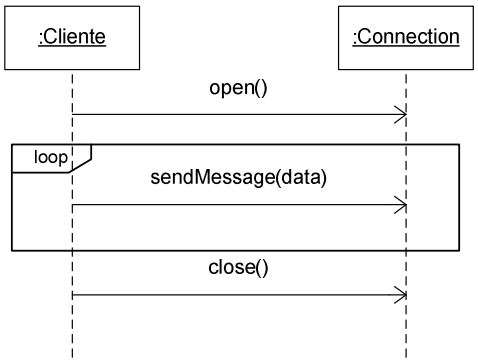
ii. Realice el DCD completo, incluyendo las clases Seccion, Aviso, Terminal y otras clases que usted considere necesarias.
iii. Explique qué patrón(es) de diseño utilizó indicando (para cada patrón) las clases participantes y sus roles.

Observaciones:
- Puede agregar 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.

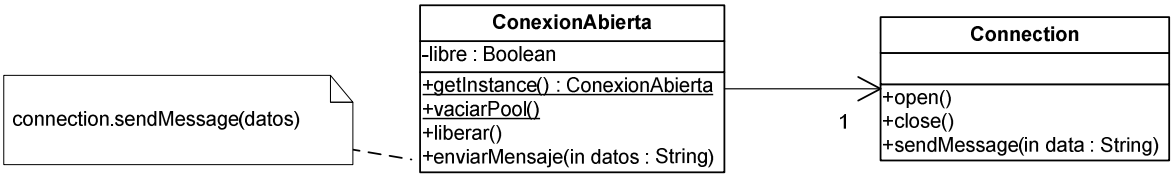
Problema 4 (25 puntos)

a) Defina brevemente los conceptos de segment y full descriptor.

b) Se necesita construir un mecanismo que permita a la aplicación que se está desarrollando comunicarse con un servidor remoto. Para esto se utilizará una clase provista por terceros llamada `Connection` cuyo uso sigue el siguiente protocolo:



Se busca minimizar la cantidad de veces que se abren conexiones (`open`) porque éste es un proceso pesado. Por ello se decidió construir un *pool de conexiones*, esto es, un grupo de conexiones que se mantienen abiertas y prontas para enviar mensajes. Cabe destacar que de todas formas es importante cerrar las conexiones (`close`) en algún momento para no dejar recursos ocupados en el servidor remoto.
Para la construcción del pool se decidió utilizar una variante del patrón Singleton que maneje N instancias en lugar de una única, siendo N una constante definida en tiempo de compilación. Se realizó el siguiente diseño parcial:

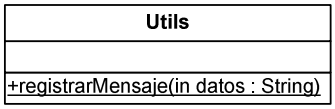


Cuando una conexión es devuelta por la operación `getInstance` se marca como "ocupada", y es responsabilidad del programador que la utiliza marcarla como "liberada" (mediante la operación `liberar`) al terminar de usarla. Por lo tanto, la operación `getInstance` debe:
- Devolver sólo conexiones libres
- Crear conexiones a demanda si no tiene ninguna conexión libre y no llegó a N conexiones abiertas
- Devolver NULL si las N conexiones están ocupadas

La operación `vaciarPool` libera totalmente la memoria reservada por el pool de conexiones.

i. **Se pide**:
   - Implementar en C++ completamente la clase `ConexionAbierta`, de forma que resuelva todos los puntos mencionados por sí misma. Incluir constructor(es) y destructor.

Luego de tener el mecanismo de comunicación funcionando surgió la necesidad de registrar todos los mensajes enviados. Para esto se cuenta con la siguiente clase utilitaria:



ii. **Se pide**:
   - Implementar en C++ el cambio teniendo en cuenta las siguientes restricciones
     - NO se puede modificar el cuerpo de la operación `enviarMensaje` de la clase `ConexionAbierta`
     - La modificación debe ser totalmente transparente para el resto de la aplicación
     - Evite que la solución contenga código duplicado

OBSERVACIONES:
- No es necesario incluir directivas al preprocesador en el código.