Programación 4

EXAMEN FEBRERO 2008

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)

Se desea construir un sistema que permita comunicar aplicaciones entre sí a través del envío y la recepción de mensajes. Las aplicaciones colocan los mensajes que desean enviar en canales. Posteriormente, otras aplicaciones podrán consumir los mensajes de estos canales completando así la comunicación.

Las aplicaciones poseen un identificador numérico único. Cada una puede consumir y enviar mensajes desde y hacia muchos canales, pero no puede realizar ambas tareas en un mismo canal. Previo al envío y/o consumo de mensajes, una aplicación debe registrarse con el canal a interactuar, siendo de interés almacenar la fecha en que se registró. Por último, para cada aplicación se debe conocer el conjunto de mensajes que fueron enviados o consumidos desde cada canal.

Los canales poseen un nombre que los identifica. Además deben mantener el conjunto de mensajes que fueron enviados a través de ellos. Los canales pueden ser de dos tipos: punto a punto (p2p) o publicación-subscripción (pub/sub).

Los canales p2p se caracterizan por permitir que cada mensaje enviado sólo pueda ser consumido por una única aplicación de las muchas consumidoras que pueda tener el canal.

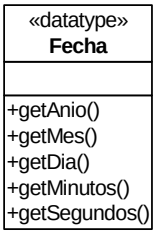
Los canales pub/sub se caracterizan por permitir la subscripción de aplicaciones para avisar la llegada de los mensajes a ser consumidos. Para que una aplicación pueda subscribirse a un canal pub/sub debe haberse registrado previamente como consumidora del mismo. Las subscripciones deben mantener su fecha de creación.

De cada mensaje se conoce un identificador numérico único, la aplicación que lo envió, los canales a los que fue enviado y su contenido (texto).

Se pide:

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

Nota: suponga dado el datatype Fecha con los operadores de comparación definidos:

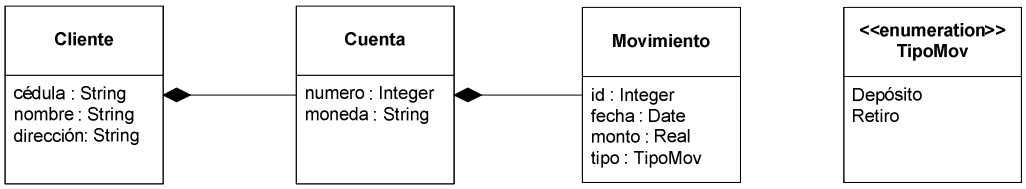


Problema 2 (25 puntos)

a) Defina brevemente los distintos tipos de visibilidad vistos en el curso.

b) Un banco esta desarrollando un nuevo programa para gestionar los clientes y sus cuentas. Ya se cuenta con una primera versión del modelo de dominio y de los casos de uso a desarrollar en la primera fase:

Modelo de dominio:



Notas:

- El atributo id identifica los movimientos y es auto-generado.
- El atributo número identifica las cuentas y es auto-generado.
- El atributo cédula identifica a los clientes.

Caso de uso 1: Iniciar una sesión
"Al iniciar la aplicación, el sistema solicita nombre de usuario y contraseña para iniciar la sesión. El usuario los provee, el sistema los valida y muestra la pantalla principal."

Caso de uso 2: Borrar una cuenta
"Una vez que el usuario ha iniciado la sesión y se encuentra en la pantalla principal, selecciona la opción de ver información de un cliente. El sistema solicita entonces la cédula del cliente y luego muestra sus datos (cédula, nombre y dirección) y una lista de sus cuentas (mostrando para cada cuenta el número y la moneda). El usuario indica al sistema la cuenta a borrar. El sistema borra la cuenta y actualiza la pantalla permitiendo al usuario seguir borrando cuentas. Para finalizar el usuario selecciona la opción de salir."

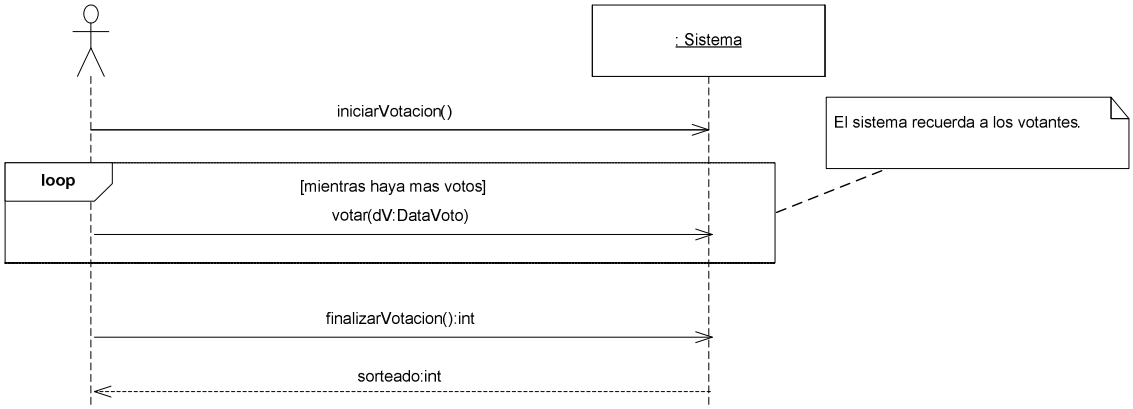
Caso de uso 3: Retiro
"Una vez que el usuario ha iniciado la sesión y se encuentra en la pantalla principal, el usuario selecciona la opción para hacer un retiro. El sistema solicita entonces la cédula del cliente y luego muestra sus datos (cédula, nombre y dirección) y una lista de sus cuentas (mostrando para cada cuenta el número y la moneda). El usuario indica al sistema la cuenta a utilizar. El sistema solicita el monto a retirar, el usuario lo ingresa y el sistema da de alta el movimiento. El sistema da la opción de imprimir un recibo, si el usuario acepta se imprime el mismo. Cualquier sea la opción anterior, el sistema vuelve a la pantalla principal."

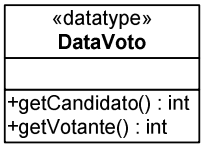
Se pide:

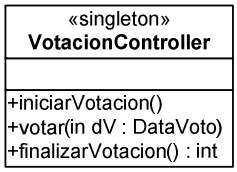
i. Los diagramas de secuencia de sistema (DSS) completos para los casos de uso 2 y 3.
ii. Las interfaces del sistema en UML tomando en cuenta las operaciones detectadas en la parte i.
iii. Los diagramas de comunicación incluyendo las visibilidades para las operaciones responsables de retornar la lista de cuentas de un cliente y cerrar una cuenta.

Problema 3 (25 puntos)

a) Se desea diseñar un sistema de votación genérico, el cual debe permitir comenzar una votación, registrar los votos, y finalizar la votación que entre otras cosas realiza un sorteo entre los votantes determinando un ganador. Durante la fase de análisis y diseño se obtuvieron los siguientes artefactos:



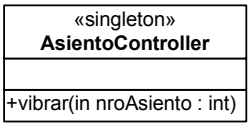




Se pide:

i. Modifique el diseño de la clase VotacionController teniendo en cuenta que deberá permita a otros (potencialmente múltiples) sistemas, realizar acciones una vez ejecutada la operación finalizarVotacion. Justifique brevemente y realice el diagrama de comunicación que muestre como resuelve dicho requerimiento.
ii. Realice el DCD correspondiente.
iii. Explique qué patrón(es) de diseño utilizó indicando (para cada patrón) las clases participantes y sus roles.

b) Dicho sistema pretende ser utilizado en el teatro de verano donde los espectadores votan a su artista favorito a lo largo de la jornada. Para ello se cuenta con un sistema electrónico en los asientos el cual tiene una consola con teclado, dicho sistema será utilizado para que los espectadores voten. Cada candidato a votar tiene asociado un entero que lo identifica, lo mismo sucede para los asientos. Para votar, el espectador simplemente elige un número de candidato preferido a través de la consola. Se asume que existe un mecanismo (que no se deberá diseñar) por el cual se pasa un dataValue *DataVoto* con la información relativa al candidato y del votante, (que en este caso es el numero de asiento) cada vez que se emite un voto. Tampoco se Las operaciones para el inicio y el registro de votos tampoco deberán ser diseñadas. Una vez finalizada la votación se realiza el sorteo entre todos los participantes y se le avisa al espectador ganador haciéndole vibrar su asiento. La siguiente clase, que no se puede modificar, es la encargada de hacer vibrar los asientos a través de la operación vibrar.



Se pide:

i. Justifique brevemente como resuelve los requerimientos planteados y realice el DCD completo.
ii. Explique qué patrón(es) de diseño utilizó indicando (para cada patrón) las clases participantes y sus roles.

Notas:

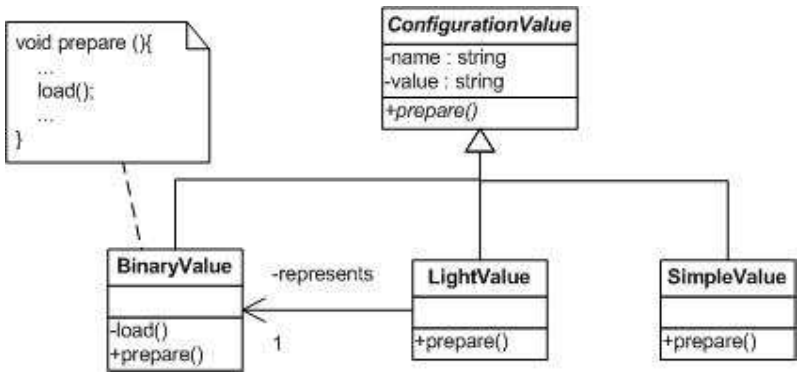
- Puede hacer todo tipo de modificación que considere necesaria a los componentes ya diseñados **salvo la clase AsientoController que no se puede modificar**.
- Tenga en cuenta que se busca minimizar la duplicación de código y tratar de definir mecanismos genéricos con la aplicación de patrones de diseño.

Problema 4 (25 puntos)

a) Enumere y describa bre vemente los diferentes tipos de relaciones que se pueden dar entre los elementos del diseño de un sistema orientado a objetos, brinde en cada caso un ejemplo de implementación en C++.

b) En un determinado proyecto se está desarrollando un programa que sea capaz de realizar la instalación de software en un determinado sistema operativo. Normalmente el funcionamiento de este tipo de programas se basa en una serie de pantallas que recopilan información sobre configuración, para finalmente realizar la instalación con las preferencias seleccionadas. Cada pantalla de configuración permite avanzar en el proceso de instalación así como también retroceder en el mismo, brindando la posibilidad de corregir posibles errores. En cada etapa, el usuario ingresa información que debe guardarse en una estructura interna como un conjunto de parejas nombre-valor con el fin de mantener la genericidad de la estructura y así poder reusarla en diferentes paquetes de instalación.

Durante el proceso de instalación, el programa principal que lo controla mantiene en memoria una instancia de la clase `InstallationManager`, la cual posee una referencia a la estructura interna que guarda temporalmente en memoria los valores de configuración del instalador ingresados por el usuario. Existen determinados valores de configuración que son costosos de cargar en memoria dado que pueden ser potencialmente muy grandes (un ejemplo puede ser un archivo binario conteniendo una actualización de firmware de un componente de hardware), por lo tanto dichos valores son cargados solo cuando la instalación se confirma y mientras esto no ocurre se guardan en la estructura otros objetos más livianos que los representan. Se muestra a continuación un diagrama con el diseño de estas clases.



Cuando se termina el proceso de configuración de la instalación y se confirma la misma, la instancia de InstallationManager ejecuta el un método propio con la firma: void makeInstall(), el cual utiliza una clase llamada MainInstaller (esta clase ya está desarrollada) que es quien finalmente realiza la instalación. Dicha clase posee un único método estático con la siguiente firma:

```
static void install(ICollection * config);
```

Nótese que el método de instalación de MainInstaller recibe una colección con los valores de configuración, pero antes de pasarle estos valores por parámetro todos estos deben estar cargados en memoria (incluso aquellos que inicialmente no se cargaron como se explico anteriormente).

Se pide:

i. Basado en lo visto en el curso para manejo de colecciones, implementar en C++ todas las interfaces necesarias para representar la estructura antes descrita así como su comportamiento. No se debe incluir en la solución las directivas del pre-procesador.

ii. Implementar completamente la clase `InstallationManager` utilizando todas las clases e interfaces definidas anteriormente, sin incluir su destructor.

iii. Identificar los patrones de diseño utilizados en la implementación de la solución.