Programación 4

EXAMEN DICIEMBRE 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) Responda de forma breve y concisa las siguientes preguntas:

i) ¿Qué es una agregación compuesta?
ii) ¿Qué es un tipo asociativo?
iii) ¿Cuándo es necesario utilizar un rol?

b) Una compañía de desarrollo de software desarrollará un sistema de gestión de inventario. El sistema será desarrollado siguiendo una metodología iterativa e incremental. En cada una de sus 3 fases se irán agregando nuevas funcionalidades al sistema. Como analista de requerimientos se le pide a usted que realice el modelado del dominio del sistema.

i) Fase 1: Cuentas de inventario.

"Una cuenta de inventario se identifica por un nombre y registra la cantidad de unidades existentes (balance). Además, cada cuenta registra su secuencia de movimientos. Cada movimiento de una cuenta se identifica por un número (relativo a la cuenta a la que pertenece), contiene una breve descripción del mismo y la cantidad de unidades que entra o sale. El balance de una cuenta debe corresponderse con sus movimientos."

Se pide: Realice el modelo de dominio para esta fase y escriba en OCL sus restricciones.

ii) Fase 2: Transacciones de inventario.

"Una transacción registra un conjunto no vacío de movimientos de distintas cuentas. Todos los movimientos son generados por una transacción. Éstas se identifican por un número único en todo el sistema y solamente incluyen movimientos del mismo tipo (todos de entrada o todos de salida)."

Por ejemplo, una transacción es: "entran 5 unidades a la cuenta X, 6 a la cuenta Y y 3 a la cuenta Z".

Se pide: Realice el modelo de dominio para esta fase y escriba en OCL sus restricciones.

iii) Fase 3: Cuentas agregadas.

"Para facilitar la administración, el sistema debe permitir la agrupación de 2 o más cuentas en un nuevo tipo de cuenta llamada cuenta agregada. Las cuentas agregadas tienen un nombre y su balance es la suma de los balances de las cuentas que la componen. Las cuentas agregadas no tienen movimientos. Una cuenta agregada no puede agregarse a si misma."

Se pide: Realice el modelo de dominio para esta fase y escriba en OCL sus restricciones.

Nota: Todos los modelos deben ser independientes y solo deben agregar o cambiar lo necesario.

Problema 2 (25 puntos)

a) Conteste brevemente acerca de contratos:
   i. Qué cosas se especifican en las precondiciones.
   ii. Qué cosas se especifican en las postcondiciones.

b) Se desea desarrollar un sistema que permita a clientes consultar y comprar artículos de un catálogo on-line mediante la utilización de un carrito de compras. Cada artículo posee un identificador, una descripción y un precio. Es imprescindible que para poder confirmar la compra on-line, el cliente esté autenticado (es decir loggeado) en el sistema, no así para poder consultar artículos ni para agregarlos al carrito de compras.

   El caso de uso principal es el siguiente:

Nombre	Compra on-line	Actores	Cliente
Descripción	El caso de uso comienza cuando el cliente inicia una nueva compra en el sistema. Éste mostrará una lista con todos los identificadores de artículos. El cliente podrá consultar un artículo por vez a partir de su identificador y el sistema le mostrará todos los detalles de ese artículo (identificador, descripción y precio). Luego de ver estos detalles, el cliente decide si agregar el artículo consultado al carrito de compras (junto con la cantidad de ese artículo a comprar) o no. Después de agregar al carrito todos los artículos de su interés, el cliente debe indicar la terminación de la compra. Si ya se encontraba loggeado en el sistema, se continuará con el proceso de compra on-line, de lo contrario el sistema le pedirá el ingreso de usuario y contraseña. Se le darán todas las oportunidades que sean necesarias para que se autentique, y no se permitirá continuar hasta que esté autenticado, pudiendo terminar inmediatamente el caso de uso si decide no loggearse. Una vez que se tiene autenticado al cliente (ya sea porque ya estaba loggeado o porque lo acaba de hacer) el cliente debe confirmar la compra, con lo cual el sistema controlará el stock de todos los artículos agregados al carrito. En caso de que no exista stock suficiente para algún producto, se le notifica al cliente y éste se des-loggea del sistema, no pudiendo modificar la compra. Si existe suficiente stock, el cliente ingresará los datos de su tarjeta de crédito (compañía, vencimiento y número) y se culminará el proceso de compra on-line.		

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

   ii. Se desea ahora estudiar la forma en que este caso de uso será utilizado por los clientes. Se sabe que existirán muchos clientes accediendo a la vez al sistema (multi-usuario) y que éste debe manejar el estado para cada uno de ellos (en particular se debe guardar para cada cliente, si se encuentra autenticado o no y los artículos del carrito con sus cantidades). Además, como es usual, se desea quebrar la dependencia de las clases de la presentación hacia las clases de la lógica. Se pide entonces realizar un Diagrama de Clases de Diseño que muestre las clases e interfaces necesarias para resolverlo.

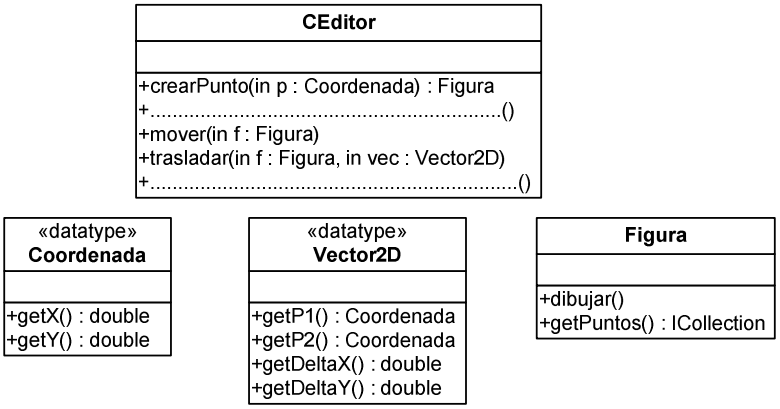
   Nota: para esta última parte, no se pide realizar ninguna colaboración, solamente el DCD.

Problema 3 (25 puntos)

a) Explique brevemente el problema tipo que resuelven los patrones Strategy y State.

b) Se desea implementar un editor gráfico, que maneje figuras geométricas, por el momento: puntos, rectas y polígonos. El editor debe ser capaz de realizar transformaciones geométricas para dichas figuras, como ser: rotaciones, traslaciones, homotecias, etc. Se debe permitir seleccionar una figura y una transformación, y realizar tantas veces como el usuario lo desee la transformación sobre esa figura, pudiendo modificar la transformación a aplicarse en cualquier momento

A continuación se muestran algunos componentes ya desarrollados en un DCD incompleto.



Aclaraciones `CEditor` es el controlador del editor, a modo de ejemplo se muestran 3 operaciones (`crearPunto()`, `trasladar()` y `mover()`, que crean un punto, trasladan una figura y aplican la transformación asociada a la figura, respectivamente), los puntos suspensivos indican que hay más operaciones.

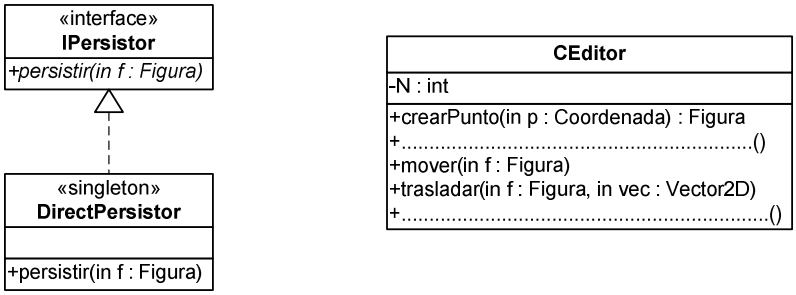
Se pide:

i. Realice el DCD completo correspondiente a las figuras y transformaciones.
ii. Realice un diagrama de comunicación que muestre todas las interacciones que ocurren cuando se ejecuta la operación `trasladar()` en la clase `CEditor`.
iii. Explique qué patrón(es) de diseño utilizó indicando (para cada patrón) las clases participantes y sus roles.

Notas:

- Puede agregar toda modificación que sea necesaria a los componentes ya desarrollados.
- Tenga en cuenta que se busca minimizar la duplicación de código y tratar de definir mecanismos genéricos con la aplicación de patrones de diseño.

c) Como todo editor gráfico, grabar a disco las figuras generadas es un requerimiento fundamental. Para ello se cuenta con una interfaz `IPersistor` que declara la operación `persistir(Figura f)` y una clase `DirectPersistor` que la implementa (escribiendo a disco la figura pasada por parámetro). Cada vez que se crea una figura debe recibir en su constructor un `IPersistor`, para poder persistirse luego de haberse efectuado algún cambio en ella.



Por problemas de desempeño del sistema, no es deseable que se persista cada vez que ocurra una transformación en una figura, por lo que existe una constante N (definida en `CEditor`), que indica la cantidad de transformaciones que se deben realizar antes de que efectivamente se persista a disco.

Se pide:

i. Realice el DCD completo correspondiente al requerimiento de persistencia, incluyendo sólo lo necesario del DCD de la parte anterior.
ii. Explique qué patrón(es) de diseño utilizó indicando (para cada patrón) las clases participantes y sus roles.

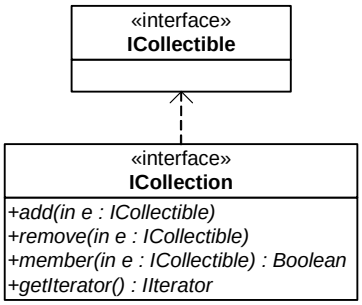
Notas:

- Puede agregar toda modificación que sea necesaria a los componentes ya desarrollados.
- 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)

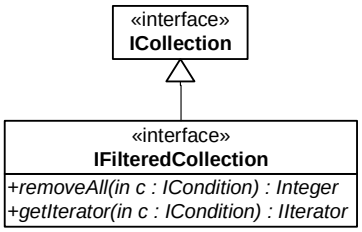
a) Explique brevemente que es una colección genérica y una colección concreta (o tipada). Indique cómo se relacionan ambos conceptos a nivel de diseño.

b) Considere la interfaz `ICollection` definida de la forma usual:



i. Realice el DCD de una realización, basada en una lista encadenada, de las interfaces `ICollection` e `IIterator`. Detallar el comportamiento de cada una de las operaciones de la interfaz `IIterator`.

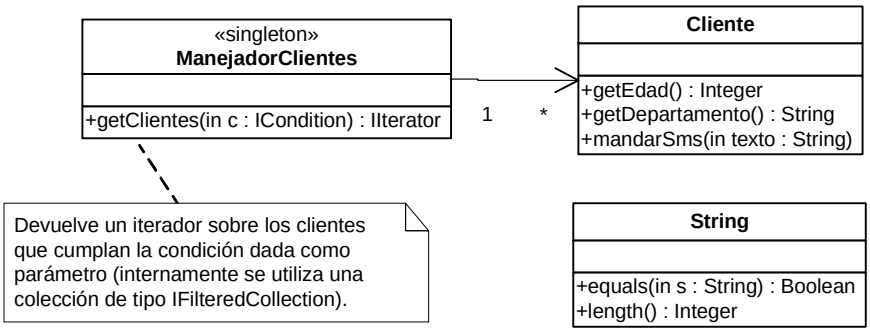
Se desea extender las funcionalidades de la colección definida anteriormente de forma de contar con las siguientes operaciones:



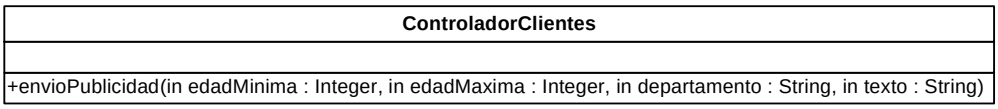
La operación `removeAll` quita de la colección todos los elementos que cumplan la condición dada como parámetro y devuelve la cantidad de elementos removidos. La nueva operación `getIterator` permite iterar únicamente sobre los elementos que cumplen la condición dada como parámetro.

ii. Diseñe (sólo DCD) e implemente en C++ esta extensión (incluyendo `ICondition`) sin modificar ninguna de las clases definidas en la parte anterior. No incluya el código de la interfaz `IFilteredCollection` ni de las clases o interfaces definidas en la parte anterior.

Una conocida empresa de telefonía celular del medio, como parte de su sistema informático, desea implantar un módulo de envío de publicidad vía mensajes de texto (sms). El envío debe ser selectivo, ya que el público objetivo depende de las características de la promoción. Uno de los CU muestra la necesidad de definir un criterio por una franja etaria y, opcionalmente, por el departamento de residencia del cliente. Se adjunta un DCD parcial para este CU:



La operación del sistema es entonces:



Como ya se mencionó, esta operación debe enviar un mensaje de texto con el texto indicado a todos los clientes que tengan una edad comprendida entre `edadMinima` y `edadMaxima`, y que vivan en el departamento indicado. Si el parámetro `departamento` es un `String` vacío se toman en cuenta todos los departamentos.

iii. Implementar en C++ la operación `envioPublicidad` incluyendo el código de la o las clases auxiliares utilizadas. No incluya las clases `ManejadorClientes`, `Cliente`, `String` ni ninguna de las clases o interfaces definidas en las partes anteriores.

OBSERVACIONES:

i. No es necesario incluir directivas al preprocesador en el código.