Programación 4

PARCIAL FINAL EDICIÓN 2012

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 (35 puntos)

Se está desarrollando un software para gestionar el envío de mensajes (mail) de un usuario y en este proceso se ha relevado lo siguiente:

El sistema apunta a soportar las funcionalidades clásicas de envío de mensajes. En este sentido, cada mensaje tendrá un título y un texto, así como un número único asignado por el sistema. Además, cada mensaje será enviado a un conjunto de destinatarios (al menos uno) de los que se conocerá su dirección electrónica. A esta dirección serán enviados los mensajes y al mismo tiempo servirá para identificar a los destinatarios. No todos los destinatarios recibirán el mensaje de la misma forma. En particular, existen tres tipos de recepción de un mensaje: normal (TO), con copia (CC), con copia oculta (BCC). Por ende, para cada mensaje se desea conocer de qué forma recibe el mensaje cada destinatario (un destinatario podrá recibir cada mensaje sólo de una forma).

Por otro lado, el sistema apunta a incluir algunas funcionalidades que faciliten el almacenamiento y búsqueda de los mensajes. Para ello se desea poder crear carpetas en las cuales se puedan almacenar los diferentes mensajes que se envían. Cada carpeta será identificada por un nombre y particularmente existirá una carpeta (de nombre ENVIADOS) en la que se guardarán por defecto todos los mensajes enviados. El sistema tendrá la particularidad de que un mensaje podrá ser almacenado en más de una carpeta. Adicionalmente, se podrán definir carpetas cuyo contenido sea propiedad de un destinatario, es decir, que contengan mensajes enviados (de cualquier manera) a un destinatario particular. Estas carpetas las definirá el usuario en la medida de lo necesario. Una última funcionalidad permitirá determinar la prioridad de un mensaje y controlar si se recibió alguna respuesta al mismo. Para ello es necesario determinar cuándo un mensaje es prioritario, y en este caso, asociar al mismo la fecha de envío.

a) **Se pide**: modelar la realidad planteada mediante un Diagrama de Modelo de Dominio UML y expresar todas las restricciones del modelo en lenguaje natural.

Como parte del relevamiento se especificó el siguiente caso de uso:

Nombre	Envío de Mensaje
Actores	Usuario
Sinopsis	El caso de uso comienza cuando el usuario desea enviar un mensaje. Para ello el usuario crea un nuevo mensaje indicando el título del mismo y el texto que contiene, así como indica si el mensaje es prioritario o no. Ante esto, el sistema registra el mensaje asignándole un número que lo identifica. Luego, el usuario ingresa el conjunto de destinatarios que recibirán el mensaje, indicando para cada uno el tipo de recepción del mensaje. El sistema registra la información y controla que haya al menos un destinatario para el envío del mensaje. Finalmente, el usuario envía el mensaje y el sistema almacena el mismo en la carpeta por defecto, así como en las propias de los destinatarios, en caso de que existan.

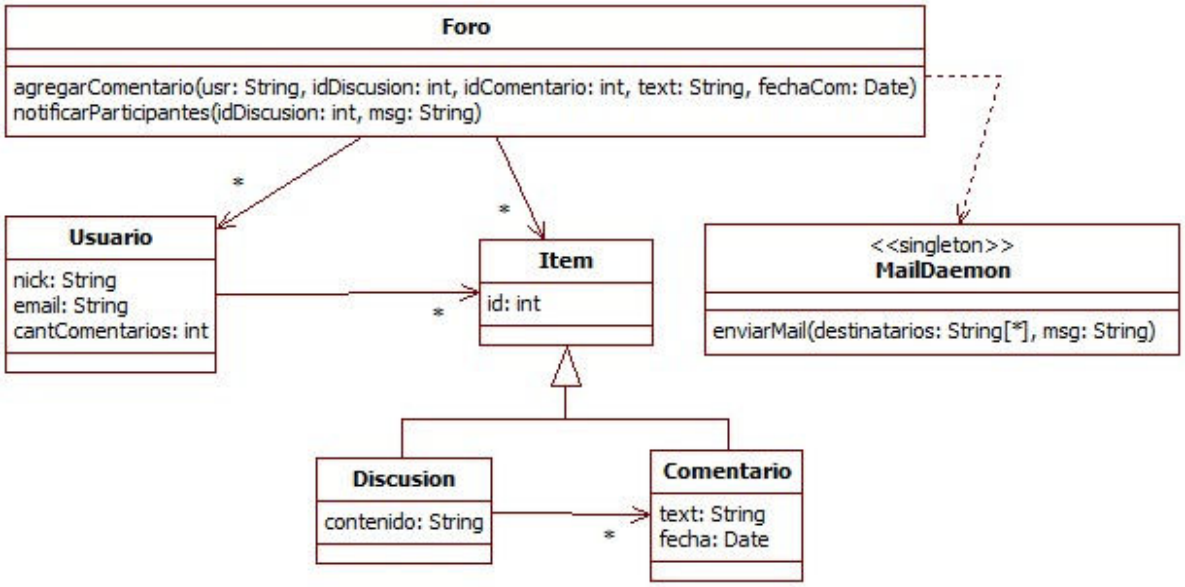
b) **Se pide**:

i. Realizar el Diagrama de Secuencia del Sistema (DSS) correspondiente al caso de uso.

ii. Expresar las pre- y post-condiciones, en lenguaje natural, de los contratos correspondientes a las operaciones del DSS anterior, de acuerdo al modelo de dominio realizado en la parte a).

Problema 2 (30 puntos)

Se cuenta con el siguiente Diagrama de Clases de Diseño (DCD) parcial para la implementación de un foro de discusión donde los usuarios pueden crear discusiones y/o comentar sobre ellas.



Se pide:

i. Realizar el diagrama de comunicación de la siguiente operación.

Operación	`agregarComentario(usr : String, idDiscusion : int, idComentario : int, text : String, fechaCom : Date)`
Descripción	Agrega un nuevo comentario realizado por un usuario a una discusión existente (no necesariamente iniciada por dicho usuario).
Pre y Post Condiciones	Pre: Existe una instancia de `Usuario` con `nick = usr`. Pre: Existe una instancia de `Discusion` con `id = idDiscusion`. Post: Se crea una instancia de `Comentario` con `id = idComentario`, `text = texto` y `fecha = fechaCom`. Post: Se genera un link entre la instancia de `Comentario` y la de la `Discusion` identificada en la precondición. Post: Se genera un link entre la instancia de `Comentario` y la del `Usuario` identificada en la precondición. Post: Se aumenta en 1 el atributo `cantComentarios` de la instancia de `Usuario` identificada en la precondición.

ii. Realizar el diagrama de comunicación de la operación `notificarParticipantes(idDiscusion:int, msg:String)`, ejecutada cada vez que se agrega un comentario en una discusión. Esta operación envía un email con el mensaje `msg` a todos los usuarios que hayan comentado en dicha discusión, haciendo uso de la operación `enviarMail(…)` de la clase `MailDaemon`.

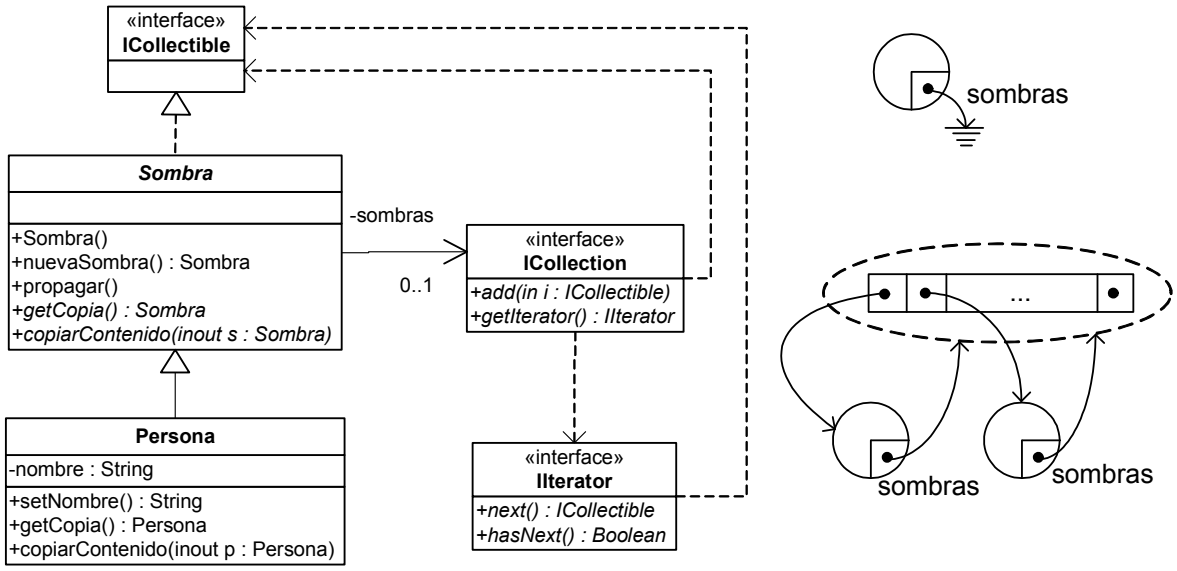
Nota: Para las partes i e ii, no puede modificarse el DCD a excepción de agregar DataTypes en caso de que lo requiera.

Problema 3 (35 puntos)

Se desea implementar un tipo de objetos que represente el concepto de *sombra*. Una sombra de un objeto es otro objeto con el mismo contenido pero replicado, por lo tanto ocupa un lugar diferente en la memoria. Todos los cambios que se aplican a un objeto de este tipo, deben reflejarse en todas sus sombras. Para cumplir con este requerimiento se ha diseñado la clase abstracta `Sombra`, según el diseño de la figura. Hay dos operaciones clave:

- La operación `nuevaSombra` devuelve una referencia a una copia de la sombra implícita, además de anexarla a la colección `sombras` de todas las sombras relacionadas (que incluye a la implícita); para obtener la copia de la instancia concreta utiliza la operación polimórfica `getCopia`.
- La operación `propagar`, efectivamente propaga a todas las sombras relacionadas, los cambios realizados mediante cualquier operación de tipo *set*; para copiar el contenido de la instancia concreta utiliza la operación polimórfica `copiarContenido`, que copia el contenido del objeto implícito en el parámetro.

Como criterio de implementación se ha decidido que al momento de crearse un objeto de una subclase concreta de `Sombra`, su colección `sombras` no se instancia; la primera vez que se invoca la operación `nuevaSombra` tiene sentido la existencia de tal colección en el objeto `Sombra`, por lo tanto en ese momento se instancia (ver figura).



Se pide: implementar en C++

i. La clase `Sombra`.
ii. La clase `Persona`.

***Aclaraciones*:**

- No se sugiere el uso de ningún patrón de diseño.
- No incluir directivas al pre-compilador.

- Implementar únicamente las operaciones incluidas en el Diagrama de Clases de Diseño (DCD) de la figura.
- Utilizar solamente las operaciones de `ICollection` e `IIterator` incluidas en la figura.
- Asuma que cuenta con una realización de la interfaz `ICollection` denominada `Lista`.