Técnicas de Desarrollo de Programas
Ingeniería Informática
Curso 2008 / 2009

Ejercicios de Patrones de Diseño:

Iterator, Composite, Strategy, Observer, Decorator, Visitor

Ejercicio 1 (examen de junio año 2003/04)

El departamento de producción de una empresa informática tiene establecido un sistema que le asigna guardias a sus empleados sobre los servicios que ofrece. Existen distintos servicios (ej. bases de datos o servidores de aplicaciones) que tienen que monitorizarse, por lo que en el momento en que uno de ellos se activa, las personas encargadas de la guardia de ese servicio reciben una notificación.

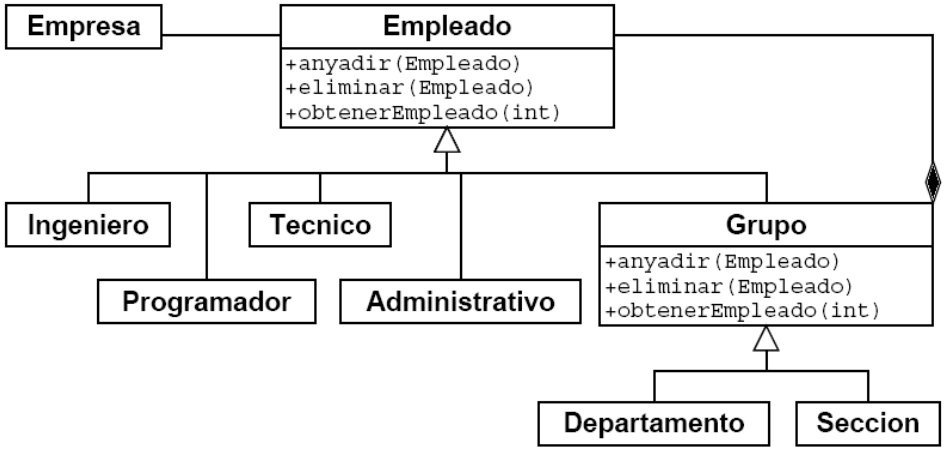
Se pide:

1. El diagrama de clases del patrón de diseño que facilita la notificación por parte de cada servicio al personal de guardia encargado de monitorizarlo. Suponer que el personal de guardia puede monitorizar más de un servicio de cualquier tipo. Comentar brevemente el sistema de actualización elegido, razonando la respuesta.
2. Especificar el diagrama de secuencia que refleja lo que ocurre desde el momento en que dos miembros del personal de guardia se suscriben a los eventos de un servicio y posteriormente se enteran de la activación de dicho servicio.
3. El departamento de producción decide modificar el sistema de guardia, por lo que de ahora en adelante cada miembro del personal sólo recibirá notificaciones sobre determinados tipos de eventos que se hayan producido en los servicios y por los que previamente haya mostrado su interés, como pueden ser la activación, parada o recuperación del servicio.

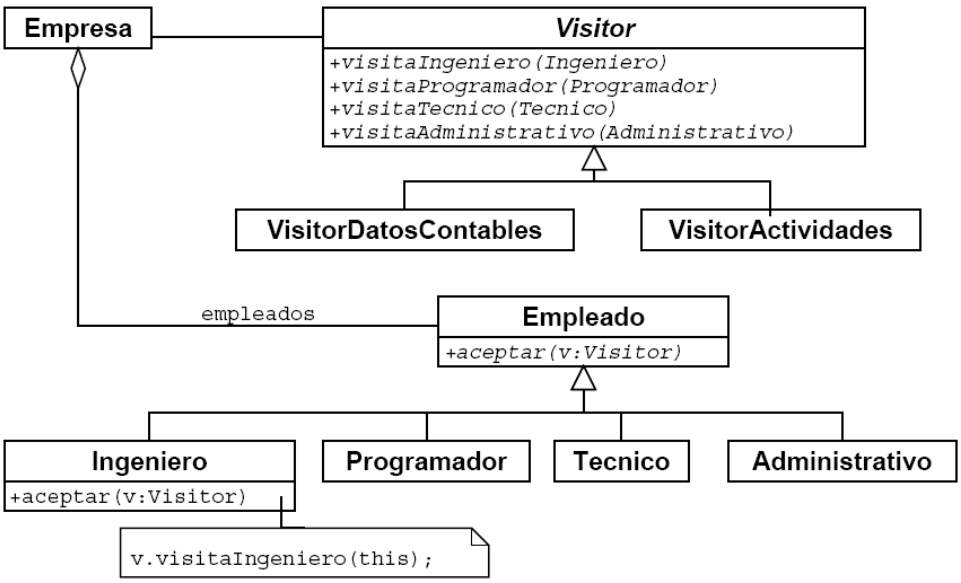
   Modificar el diagrama de clases propuesto en el apartado 1 para tener en cuenta que una persona de guardia puede expresar su interés no sólo por servicios concretos, sino por determinados cambios de estado en esos servicios.

Ejercicio 2 (examen de junio año 2003/04)

Los empleados de una empresa informática se pueden clasificar en ingenieros, programadores, técnicos y administrativos. La estructura en la que la empresa almacena a sus empleados es un *Composite*, como muestra la siguiente figura:



Teniendo en cuenta que las interfaces de los distintos tipos de empleados son heterogéneas, para poder incluir nuevas operaciones sin necesidad de modificar dichos empleados se ha aplicado un patrón *Visitor*. El diseño es el que muestra la siguiente figura:



Se pide:

1. Describir cada una de las soluciones posibles que permiten a la empresa recorrer a sus empleados para solicitarles una operación. Indicar claramente cuál es el cometido de la empresa en cada una de las soluciones propuestas.
2. Incluir el código de cada solución propuesta en el apartado anterior.

Ejercicio 3 (examen de septiembre año 2003/04)

La salida de una determinada aplicación se puede llevar a cabo a través de diferentes medios como fichero, pantalla, base de datos, etc. Por defecto la salida se produce siempre a un fichero de registro pero, dependiendo de lo que solicite el usuario, la salida puede ser además a través de la pantalla, en otros ficheros diversos, a una base de datos, etc.

Se pide:

1. El diagrama de clases que permite modelar la situación descrita, teniendo en cuenta que el cliente no debe verse afectado por el número de dispositivos finales de salida a la hora de solicitar escribir información de salida.
2. El código del método o métodos involucrados en uno de los tipos de escritura concreta, por ejemplo la escritura en pantalla, teniendo en cuenta que la información que se muestra en pantalla (líneas de salida) aparece precedida del nombre de la aplicación que la ha generado. Indicar claramente a qué clase pertenecen.

   Ejemplo:

   ```
   - nombreAplicacion – línea 1
   - nombreAplicacion – línea 2
   - nombreAplicacion – línea 3
   ```

3. Especifica el código del cliente que sirve para inicializar el objeto u objetos encargados de la escritura en el supuesto de que ésta se produzca sobre el fichero por defecto (`default.log`), por pantalla y a otro fichero (`otro.log`).

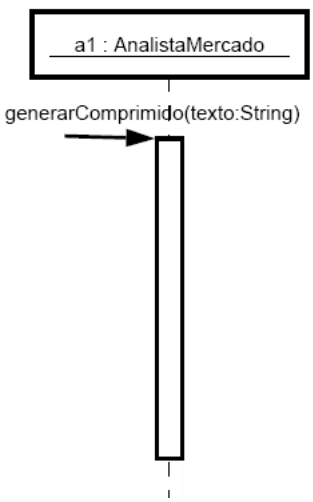
Ejercicio 4 (examen de septiembre año 2004/05)

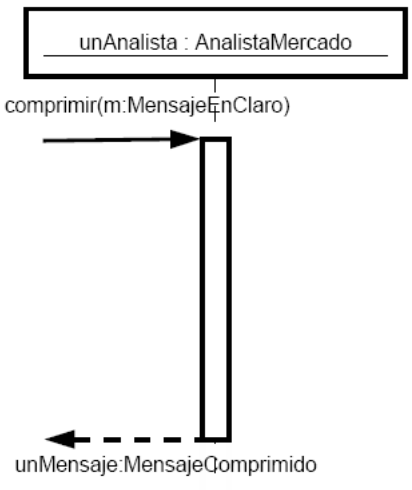
La empresa DEISOFT, especializada en el desarrollo de aplicaciones de gestión para el mercado bursátil, desea desarrollar un componente para homogeneizar el envío de mensajes de los analistas de mercados a sus clientes más relevantes. En una primera versión, el componente deberá contemplar los siguientes requisitos:

- Por defecto, el mensaje se enviará como texto en claro.
- A petición del usuario, el contenido del mensaje podrá enviarse comprimido, cifrado o firmado digitalmente.
- Se facilitarán dos algoritmos de compresión: Huffman y Lempel-Ziv. Existe la intención de incorporar nuevos algoritmos de compresión específicos en sucesivas versiones.

Se pide:

1. El patrón de diseño que permite el procesamiento del contenido del mensaje teniendo en cuenta que a petición del emisor puede tratarse como texto en claro, comprimido, cifrado, firmado digitalmente o cualquier combinación de las mismas. Deberán responderse las siguientes cuestiones:
    a. La explicación del porqué del patrón seleccionado.
    b. La especificación en UML del diagrama de clases correspondiente.
    c. Completar el diagrama de secuencia izquierdo para que muestre el proceso de creación y tratamiento de un mensaje comprimido y cifrado.
2. El patrón de diseño que permite la compresión del mensaje utilizando diferentes algoritmos. Deberán responderse las siguientes cuestiones:
    a. La explicación del porqué del patrón seleccionado.
    b. La especificación en UML del diagrama de clases correspondiente.
    c. Completar el diagrama de secuencia derecho para que muestre el proceso de compresión de un mensaje en claro utilizando el algoritmo de compresión Lempel-Ziv.





Ejercicio 5 (examen de septiembre año 2007/08)

Se desea realizar un sitio web de subastas por internet. Al acceder a la aplicación, el usuario debe registrarse indicando un nombre de usuario, password, datos de contacto y dirección de correo.

Para pujar por un artículo, los usuarios deberán hacer una oferta mayor que la última realizada. Si un usuario puja por un artículo y posteriormente se realiza otra puja por él, recibirá un correo electrónico indicando la fecha y cuantía de la última puja realizada. Por su parte, el vendedor también recibirá un correo electrónico de cada puja realizada sobre los artículos que subasta.

En cualquier momento el vendedor podrá concluir la subasta contactando con el último usuario en pujar. En ese momento, todos los que pujaron por el artículo recibirán un correo indicando el final de la subasta.

Se pide:

1. Utilizando patrones de diseño, especifica el diagrama de clases de la aplicación (únicamente de los requisitos que contempla el enunciado). Indica qué patrones has usado y qué beneficios obtienes de su uso. Documenta cualquier suposición sobre la aplicación que no esté en el enunciado.
2. Especifica el diagrama de secuencia correspondiente a la puja de un artículo por un usuario del sistema.
3. Se quiere premiar a los usuarios con más de un año de antigüedad haciéndoles clientes VIP. Para ello supondremos que los usuarios tienen un atributo `esVip` de tipo *boolean* que modela este hecho, y guardan la fecha en que se registraron por primera vez en el sistema. Describe tres formas de implementar iteradores para recorrer los usuarios del sistema y clasificarlos como VIP si cumplen los requisitos de antigüedad. Indica las ventajas o inconvenientes de cada iterador propuesto.

   Nota: no es necesario implementar los iteradores, sólo describirlos.