Enunciados de Ejercicios de Patrones

Enunciado 1

Supongamos que estamos desarrollando un juego interactivo que permite al usuario interactuar con personajes que juegan ciertos roles. Se desea incorporar al juego una facilidad para crear nuevos personajes que se añaden al conjunto de personajes predefinidos.

En el juego, todos los personajes serán instancias de un pequeño conjunto de clases tales como Héroe, Villano, Príncipe o Monstruo. Cada clase tiene una serie de atributos como nombre, imagen, altura, peso, inteligencia, habilidades, etc. y según sus valores, una instancia de la clase representa a un personaje u otro. Por ejemplo podemos tener los personajes príncipe bobo o un príncipe listo, o monstruo bueno o monstruo malo.

Diseña una solución basada en patrones que permita al usuario crear nuevos personajes y seleccionar para cada sesión del juego personajes de una colección de personajes creados.

Enunciado 2

Tenemos un conjunto de clases que permiten la creación y envío de mensajes de correo electrónico y que entre otras incluye clases que representan el cuerpo del mensaje, los anexos, la cabecera, el mensaje, la firma digital y una clase encargada de enviar el mensaje.

El código cliente debe interactuar con instancias de todas estas clases para el manejo de los mensajes, por lo que debe conocer en qué orden se crean esas instancias, cómo colaboran esas instancias para obtener la funcionalidad deseada y cuáles son las relaciones entre las clases.

Idea una solución basada en algún patrón tal que se reduzcan las dependencias del código cliente con esas clases y se reduzca la complejidad de dicho código cliente para crear y enviar mensajes. Dibuja el diagrama de clases que refleje la solución e indica qué patrón has utilizado.

Enunciado 3

Supongamos que estamos construyendo una librería de clases para representar componentes GUI, se ha decidido que en vez de que un programador defina la posición de los componentes GUI (*Button*, *List*, *Dialog*,..) sobre una ventana, se incluyan manejadores de disposición de componentes (*layout manager*), cada uno de los cuales distribuye un conjunto dado de componentes gráficos de acuerdo a algún esquema de distribución: horizontalmente, verticalmente, en varias filas, en forma de una matriz, etcétera. Debe ser posible cambiar en tiempo de ejecución la distribución elegida inicialmente.

Supuesto que la clase JPanel es la que representa a un contenedor de componentes gráficos, diseña una solución basada en algún patrón de diseño para introducir en la librería los manejadores de disposición. Dibuja el diagrama de clases que refleje la solución e indica qué patrón has utilizado.

Enunciado 4

Se desea escribir una clase ***UtilTime***, que no es abstracta, que incluya un método estático **time** que mide el tiempo que tarda un método cualquiera en ejecutarse. Parte del código de dicho método sería:

```
public static long time ( "parámetros") {
    long t1 = System.currentTimeMillis();
     // falta aquí el código apropiado
    long t2 = System.currentTimeMillis();
    return t2 - t1
}
```

Diseña una solución basada en alguno de los patrones de diseño y completa el método: parámetros y código en la posición del comentario. Escribe un código cliente con un ejemplo de utilización del método ***time()***.

Enunciado 5

Sea una clase TextView que representa un componente GUI ventana de texto que es subclase de una clase Component raíz de la jerarquía de clases que representan componentes GUI. Queremos definir ventanas de texto con diferentes tipos de bordes (*Plain*, *3D*, *Fancy*) y barras de desplazamiento (horizontal, vertical). La clase TextView tiene un método dibujar entre otros

```
class TextView {
    public void mostrar() {
      // código para dibujar el objeto TextView }
}
```

Indicar qué patrón de diseño utilizarías, así como el diagrama de clases.

Enunciado 6

Supuesto que estamos desarrollando una aplicación financiera, se ha decidido emplear el patrón ***Composite***, puesto que existen diferentes valores elementales (acciones, bonos, futuros, fondos,..) y valores compuestos (carteras de valores, cuentas,..).

Describe cómo utilizar el patrón *Visitor* para realizar diferentes operaciones sobre un valor compuesto, tales como calcular su precio u obtener información fiscal anual ¿Qué beneficios se obtienen al aplicar el patrón *Visitor*?

Enunciado 7

Se requiere construir un editor de expresiones matemáticas. Una expresión válida estará formada o bien por un número, o bien por la suma/resta/división/multiplicación de dos expresiones. Ejemplos de expresiones válidas:

- 4
- 3+8
- 14 * (3+5)
- …

Indicar el patrón de diseño que utilizarías y especificar el diagrama de clases que permita representar expresiones válidas.

Enunciado 8

Queremos hacer un agente de base de datos (o Broker) en el que se centralice el acceso a la BBDD. Queremos asegurarnos de que existe una única instancia de ese agente, para que todos los objetos que la usen estén tratando con la misma instancia, accedan a ella de la misma forma. Usar una variable global no garantizaría que sólo se instancie una vez.

Indicar qué patrón de diseño se debería utilizar de manera razonada y qué características y métodos debería tener.

Enunciado 9

Supongamos que una entidad bancaria dispone de un sistema con las siguientes clases:

- La clase “Tarjeta” tiene un método
  - transferir (float importe, String cuentaDestino)
- La clase “Cuenta” tiene un método
  - transferir (float importe, String cuentaDestino)

El banco ha adquirido un modelo novedoso de cajeros automáticos que permiten hacer transferencias, pero la interfaz esperada para esta operación es:

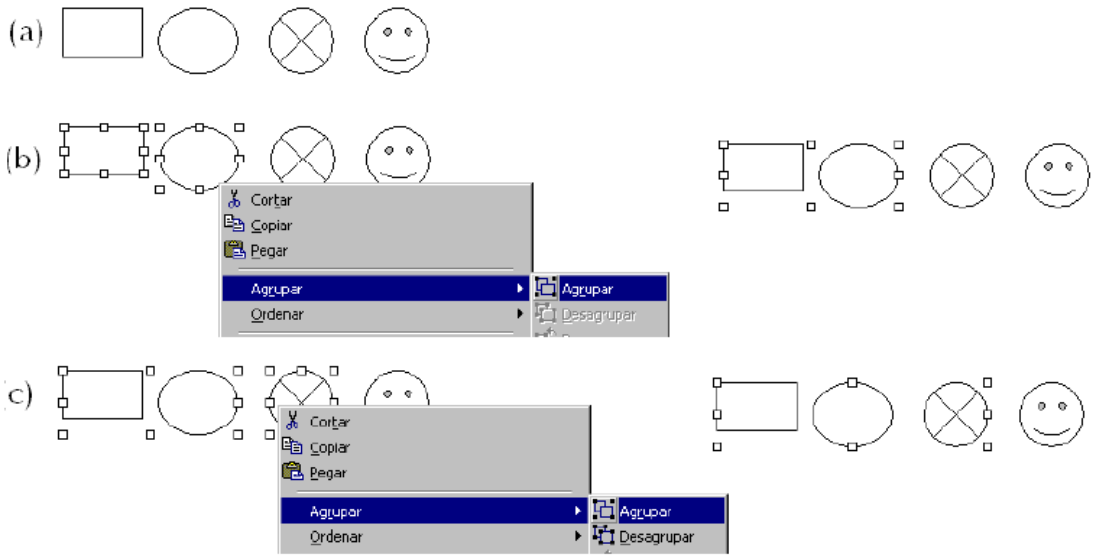
- order (String target, float amount)

¿Qué podríamos hacer para que nuestras instancias de Tarjeta y Cuenta respondan adecuadamente a los nuevos cajeros? Indica el patrón de diseño que utilizarías y diseña la solución (diagrama de clases).

Enunciado 10

Objetos de dibujo en un procesador de texto. Los objetos se pueden agrupar en un "grupo" que se maneja como un único objeto.

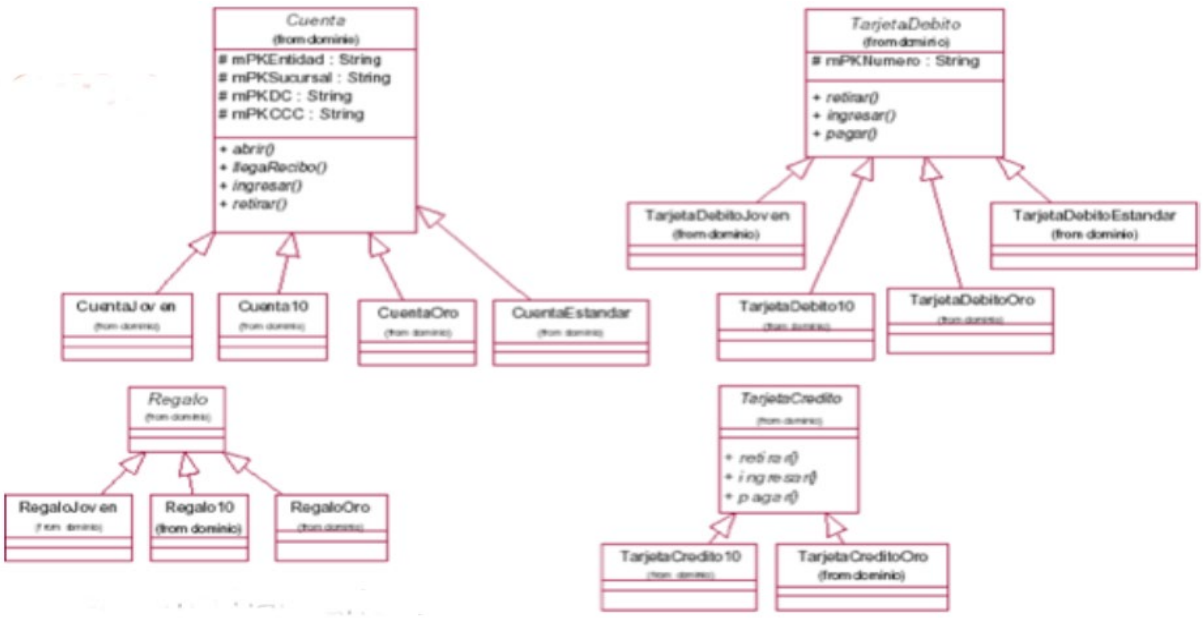
Ese "grupo" a su vez se puede agrupar con otros objetos, de forma que cada "grupo" puede contener objetos simples, grupos, mezclados,...



Indicar el patrón de diseño que debería utilizarse, así como el diagrama de clase propuesto.

Enunciado 11

En un sistema bancario se tienen diferentes mecanismos para realizar operaciones comunes como transferir dinero, sacar dinero o consultar saldo. Estas operaciones están disponibles para diferentes productos (cuentas corrientes, tarjetas de débito, tarjetas de crédito,… .



Supongamos que el banco desea dotar a los desarrolladores de la capa de presentación de sus aplicaciones de un **mecanismo unificado de acceso** a la capa de dominio. ¿Cómo se podría modelar esto? Indicar el patrón de diseño y estructura de clases.

Enunciado 12

En el sistema bancario anterior, nos centraos en la clase **Cuenta**. Sabemos que una parte importante del comportamiento de este objeto para la operación "retirar(float importe)" depende del estado de la cuenta, que puede ser:

- Cuenta ConDinero
- Cuenta SinDinero
- Cuenta BloqueadaJudicialmente

Queremos delegar el comportamiento de la operación "retirar", ¿cómo lo haríamos? Indica el patrón de diseño a utilizar y el diagrama de clases.