**Ingeniería de Software II- Taller de Estudio Patrones**

9 de mayo de 2013 Julio Ariel Hurtado Alegría

1. Plantee una solución que permita la construcción de una gran variedad de relojes visuales con decorativos. Por ejemplo decorar las manecillas con un estilo gótico, agregarle adornos al tablero del reloj (tablero + numeración + estilo de la numeración).

2. En una aplicación de venta y configuración de computadores personales se considera un computador como el conjunto de varios componentes (una unidad central y varios elementos periféricos). El mínimo imprescindible para que se considere un computador es la unidad central, un dispositivo de entrada y otro de salida, pero pueden añadirse todos los que deseemos ofertar o que nos pidan los clientes. Para crearlo habrá que dar esos componentes mínimos y se puede modificar la configuración en cualquier momento añadiendo, quitando o cambiando exclusivamente periféricos. Por último, el precio de venta del computador es la suma de los precios de sus componentes. Todos componentes tienen información sobre el nombre del fabricante, el modelo y el precio de venta, que cambia con frecuencia. Los dispositivos de entrada que manejamos actualmente son el teclado, el ratón y la tableta gráfica; en todos los casos necesitamos saber el tipo de conector que utiliza (será un string) y los puertos válidos(varios valores de tipo entero). Los dispositivos de salida de que disponemos son las pantallas y las impresoras (de inyección y láser). También tenemos que saber los puertos válidos. Además para las impresoras necesitamos saber el tipo de cartucho o tóner utilizado y el número de páginas impresas desde el último cambio de tóner(sólo para impresoras láser). Por último tenemos un dispositivo especial, la pantalla táctil que sirve de dispositivo de entrada y de salida simultáneamente. Diseñe las clases y relaciones que representen una solución adecuada para este problema. Escriba en Java al menos las clases que representen el computador, los dos primeros niveles de la jerarquía de componentes y las impresoras láser.

3. Una empresa se organiza en una jerarquía de unidades de negocio. Para este caso, de la empresa nos interesa el nombre del presidente, el nit y la dirección, mientras que de cada unidad de negocio nos interesa el gerente de área, el número de empleados, los beneficios brutos del último trimestre, la inversión en edificios y el número medio de contratos realizados por semana. Además una unidad de negocio puede estar formada por varias unidades de negocio. En

este caso empleados, beneficios e inversiones se obtienen como la suma de los datos correspondientes a las unidades de negocio que la componen, mientras que el número medio de contratos es la media de los números medios de éstas unidades de negocio. Diseñar un conjunto de clases que resuelvan el problema, utilizando el patrón compuesto e implementando completamente en Java las clases necesarias.

4. Deseamos implementar un párrafo de texto. Un párrafo es una secuencia de líneas. Cada línea es representada por un String. La clase Parrafo debe proveer al menos uno de los siguientes métodos:

```
List<String> alinearTexto(); // retorna lineas formateadas
String getLinea(int i ); // retorna la línea i-ésima.
int getNumeroLineas(); // retorna el número de líneas.
void addLinea(String s); // agrega una línea.
```

El algorimo de formateado ( por ejemplo a la izquierda o al centro) puede ser seleccionado en tiempo de ejecución. Debe ser posible agregar nuevos algoritmos de formateado sin modificar la clase Parrafo. Desarrollar una solución para la clase Parrafo que satisfaga estos requisitos. ¿Cuál patrón de diseño usaría?

5. Deseamos desarrollar un componente Java AWT que pueda desplegar texto formateado en una forma flexible (AreaTextoConFormato). Para lograr esto, deseamos reutilizar la clase Párrafo del punto anterior. Cada Componente Java AWT hereda de la clase Component. Nuestro clase AreaTextoConFormato heredará de las siguientes clases de la librería:

```
public class TextComponent extends Component
                              implements Accessible { ... }
public class TextArea extends TextComponent { ... }
public void append(String str) { ... }
public String getText() { ... }
...
}
```

Proponer un diseño para la clase AreaTextoConFormato que permita usar la clase Párrafo y heredar de la clase TextArea. ¿Cuál patrón de diseño usaría?. Proveer la implementación de los métodos append y getText.

6. Se desea extender el diseño del punto anterior agregando un contador de caracteres. Este contador es un objeto separado que almacena el número de caracteres de un objeto párrafo. Cuando el párrafo cambia, el contador es actualizado automáticamente. Proponer un diseño para la clase Contador, es permitido cambiar la clase Parrafo y posiblemente la clase AreaTextoConFormato. ¿Cuál patrón de diseño usaría?

7. Considere un simulador orientado a eventos genérico, es decir un simulador esqueleto que simula un rango amplio de sistemas. El simulador contiene una cola de eventos y una variable que indica lal hora de simulación(contador). Cada evento tiene grabado el tiempo que indica el momento en que éste debe ocurrir. La cola contiene eventos, los cuales son almacenados en orden ascendente resepcto al tiempo en que el evento ocurre. El simulador orientado a eventos genérico ejecuta el siguiente ciclo infinito en el método simular de la clase SimuladorGenerico:

```
abstract class SimuladorGenerico
 {
  protected Estado nuevoEstado, viejoEstado;
  protected double tiempoSimulado;
  protected ColaEventos colaEventos;
  protected Evento eventoActual;
  public abstract Estado cambiar(Estado viejoEstado, Evento evento);
  public abstract void actuar(Estado nuevoEstado, Estado viejoEstado);
  public void simular()
   {
    while (true)
    {
    tiempoSimulado = colaEventos.top().getTiempo();
     do
     {
       eventoActual = colaEventos.pop();
       nuevoEstado = cambiar(viejoEstado, eventoActual);
       actuar(nuevoEstado, viejoEstado);
       viejoEstado = nuevoEstado;
           } while(tiempoSimulado <= colaEventos.top().getTiempo());
    }
   }
 }
```

(a) El diseño del simulador orientado a eventos genérico esta basado parcialmente sobre un patrón de diseño. ¿Cuál es el patrón? motive su respuesta. Si está inseguro, motive otras alternativas que Usted considere.

(b) En la clase SimuladorGenerico descrita arriba, ¿Por qué algunos métodos son abstractos?. Considere que Usted tiene que escribir un simulador para un sistema específico, por ejemplo la clase SimuladorBancoSimple (Un único Cajero, una única Fila (de clientes), eventos de llegada de un cliente y eventos de salida de un cliente de la atención del cajero). Dibuje una jerarquía de clases para el simulador específico, considere los posibles eventos y la lógica específica del simulador del banco.

(c) ¿Podría usarse los patrones Fábrica Abstracta o el Método de Fábrica con el fin de generar eventos específicos al sistema simulado? Motive su respuesta.

(d) ¿Cómo podría utilizar diferentes simuladores específicos usando el patrón de diseño Estrategia? Considere cambiar el código dado. Dibuje el diagrama de clases del sistema mejorado.

8. Hemos adquirido un sistema de gestión de bases de datos con una licencia para soportar 5 conexiones simultáneas. Con el fin de permitir muchas más conexiones simultáneas, aprovechando los tiempos muertos, se ha propuesto el desarrollo de una clase PoolConnections. ¿Qué patrón usaría para lograr esta solución? ¿Qué modificaciones se realizaron? Motive sus respuestas.

9. Use el patrón Interprete para implementar un evaluador de expresiones en el dominio de los enteros. Por ejemplo al evaluar la expresión 4*2+5 el resultado debe ser 13.

10. Plantee como el patrón flyweight podría mejorrer el rendimiento de la simulación de la proliferación celular cuando un tejido alcance más de cien mil célular en la simulación (Más de 100.000 objetos que lo único que los diferencia de muchos otros es su posición), aprovechando que cada célula pertenece a un arquetipo especifico. (Un arquetipo podría ser el número de lados).

11. Investigue los usos y aplicación de los patrones Fachada y Visitante.

12. Describa un escenario donde el objeto cliente del objeto observado en el patrón observador, sea el responsable siempre de llamar el método notificar. Muestre que problemas podrían surgir de ésta decisión.

13. Describa la diferencia entre delegación y el llamado a un objeto asociado. ¿Cuándo se dice que un elemento es el delegado de otro?.

14. El patrón adapter cumple con el principio favorezca la composición a la herencia. Muestre que aquí el principio es más que opcional, particularmente en los lenguajes que no soportan herencia múltiple.

15. Explique que son operaciones hooks en el patrón Plantilla. Describa cuál es su utilidad.

16. Un problema de las fábricas abstractas es como agregar un nuevo producto de la familia. Explique el impacto de agregar un nuevo producto en un sistema con 25 fábricas concretas. Describa una posible solución al problema.

17. Explique como podría usarse el patrón Prototipo para desarrollar una herramienta de composición musical. Esto es, sobre un pentagrama, desde una caja de herramientas, se puede arrastar una nota musical de cierta duración y colocarla en el pentagrama, facilitando así una composición musical.

18. Explique como podría aplicarse el patrón comando para mejorar el juego Picas y Fijas. Es decir, poder almacenar algunas órdenes y hacer deshacer cuando el jugador lo requiera. Por ejemplo al configurar la palabra de juego, o al intentar adivinar una palabra.