Universidad del Cauca
Departamento de Sistemas
Ingeniería de Software II

8 de enero de 2013 Profesor Julio Ariel Hurtado Alegría

Segundo Parcial

1. Conocimiento de Patrones de Diseño

a) Se le ha asignado a Usted construir un framework de persistencia, el cual va a ser usado por la capa de negocio de una aplicación empresarial. En esta capa de persistencia, se requiere implementar operaciones tales como la agregación, cambio o eliminación de objetos. Estas operaciones con encoladas para posteriormente hacer una ejecución de la transacción(es decir se ejecutan todas) y cuando sea necesario ejecutar la operación que desenrolle éstas operaciones (rollback, es decir ejecutar ninguna). ¿Qué patrón utilizaría? Justifique. Valor 0.5. Respuesta: el patrón a utilizar sería el patrón commando. Las operaciones pueden ser implementadas en objetos comando que se van agregando a la lista de comandos de la transacción. Cuando se haga commit, la transacción será finalizada. Si la transacción es desenrollada entonces se enviará el mensaje undo() a cada comando, para que deshaga la operación realizada en cada comando de la lista de comandos de la transacción.

b) Considere el problema de calcular el costo del uso de un servicio de telefonía celular. Suponga que tenemos una aplicación que conoce la tarifa del servicio usado, y la cantidad de tiempo de uso del servicio y entonces (a partir de su información de precios) puede determinar el valor a cobrar por el uso del servicio. Ahora tenemos que calcular el costo total del servicio. Esto no es un problema, si tenemos una empresa de telefonía celular donde se aplica a una única tarifa a todos sus usuarios. Sin embargo, es un problema si la tarifa aplicada depende del plan comprado por el cliente, del destinatario y de la hora del servicio, entre muchas otras variables. ¿Qué patrón utilizaría? Justifique. Valor 0.5. Se utilizaría el patrón estrategia. Al momento de tarificar un servicio, se utiliza un objeto estrategia asociado al servicio del cliente que encapsula el algoritmo de tarificación del servicio de acuerdo a la información de contexto del servicio.

c) Deseamos desarrollar un componente Java Swing (Como un JPanel o un JFrame) que pueda desplegar una animación Flash. Para lograr esto, deseamos reutilizar la clase WMediaPlayer(a través de su interface IWMediaPlayer), basada en Windows Media Player (aplicable solamente a sistemas operativos windows). Cada Componente Java Swing hereda directa o indirectamente de la clase JComponent.

Queremos implementar una clase JPanelFlash que reutilizará la siguiente clase de la librería de WMP (Windows Media Player):

```
class WMediaPlayer{
 /*indica si el video está en modo pantalla completa*/
 boolean          isFullScreen()
        {
        //code ..
        }
 /*carga un video usando la diracción resourcePath*/
 void    load(String resourcePath)
        {
          //code ..
        }
 /*fija el modo del video*/
 void    setFullScreen(boolean isFullScreen)
        {
        //code ..
        }

}
```

Proponer un diseño para la clase JPanelFlash que permita usar la clase WMediaPlayer y heredar de la clase JPanel. ¿Cuál patrón de diseño usó? Justifique. Haga una implementación de un método constructor de la clase JPanelFlash, el cual debe iniciar todos los sub-componentes gráficos, cargar el video y fijar la visualización en modo pantalla completa. Valor 1.0.

En la figura 1, el patrón utilizado es el patrón adapter. Este nos permite hacer parte de la jerarquía de los componentes gráficos del framework subyacente, mientras se utiliza en forma de caja negra la clase WMediaPlayer.

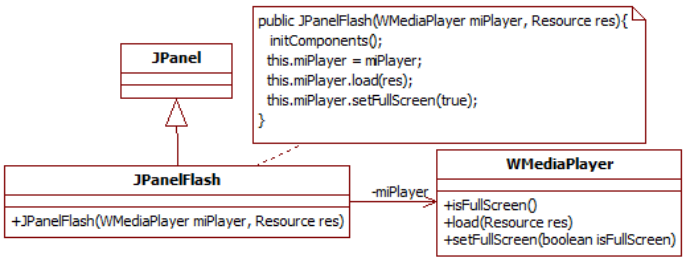

Figura 1: Solución patrón Adapter

2. Caso de Aplicación de Patrones de Diseño

Considere el desarrollo de una aplicación para monitorear el desempeño de una aplicación multi-hilo, es decir una aplicación en la que se ejecutan varios procesos livianos al mismo tiempo, los cuales tienen cada uno una prioridad. Suponga que la programación del hilo se ha hecho en la clase Hilo, la cual almacena el estado y la prioridad, y tiene los métodos arrancar(prepara el hilo y llama al método correr), correr(donde normalmente se programa el código deseado por el hilo por las subclases), bloquear(esperando un recurso), desbloquear(cuando el recurso se libera o cuando el hilo es forzado), dormir(se lo deja esperando por un tiempo sin hacer nada) y finalizar (llamado antes de morir para liberar recursos), los cuales producen cambio en el estado del hilo. El monitoreo se hace a través de un visualizador de los hilos(VisualizadorHilos), con el fin de saber visualmente el estado de los hilos. Es decir, si cada uno de éstos está en preparación(starting-amarillo), está activo(running-verde), está durmiendo(sleeping-azul) o está bloqueado(Waiting-rojo). Esto con el fin de evaluar el desempeño del sistema (consumo de tiempo) y poder realizar ciertas acciones como matar un hilo (kill), desbloquearlo o cambiarle la prioridad (por ejemplo bajarle de prioridad a un hilo voraz con alta prioridad). El visualizador también debe poder evaluar y ejecutar éstas mismas acciones sobre varios hilos, por lo cual los hilos son organizados en forma de grupos(GrupoHilo). Por defecto los hilos o grupos hilo, son creados en la aplicación principal(Principal) donde pertenece el hilo principal del programa (por defecto el sistema crea el grupoPrincipal), a menos que se le indique otro grupo al cuál pertenece.

a) Muestre mediante un diagrama de clases como el patrón composité puede ayudarle a organizar los hilos para lograr un manejo homegéneo de los mismos. Valor 0.5

   La parte sin color de la figura 2 muestra la solución a ésta pregunta. En este caso todos los hilos simples y compuestos son manipulados mediante la interfa-

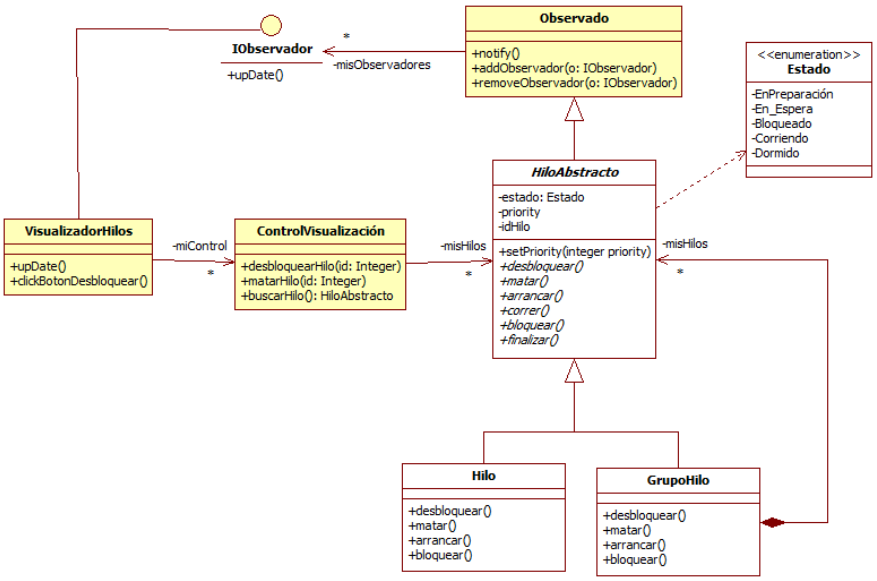

Figura 2: Solución a la Parte 2

ce HiloAbstracto, los grupos de hilos hacen de compustos, los cuales a su vez manipulan internamente otros hilos simples y grupos de hilos usando la misma interface de HiloAbstracto.

b) Complete el diagrama de clases anterior considerando la clase VisualizadorHilos haciendo uso del patrón MVC simplificado(Observador e Intermediario), usando un controlador intermedio entre el visualizador y los objetos hilo. Valor 1.0. La parte sin color de la figura 2 muestra la solución a éste problema. El intermediario ha sido implementado como una clase simple, mientras el modelo y la vista han usado el patrón observador para manejar la actualización permanente de la vista respecto al modelo.

c) Haga un diagrama de secuencia que ilustre el comportamiento del sistema de monitoreo. Valor 0.5. La figura 3 presenta la secuencia en la que un objeto cualquiera bloquea un hilo, muestra como el cambio de estado es notificado al visualizador. El administrador del sistema en cualquier momento puede desbloquearlo(al notarlo bloqueado) y así hace click en el botón desbloquear sobre el hilo(visual), así inicia la secuancia en la que se le pide al controlador que lo desbloquee, éste lo busca y finalmente le envía el mensaje desbloquear.

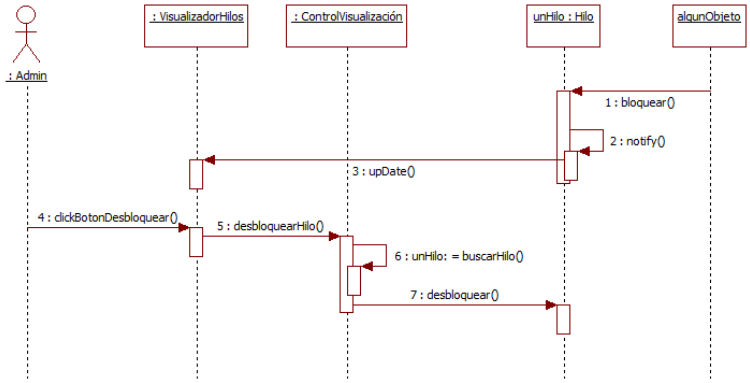

Figura 3: Colaboración Desbloqueo de un Hilo

d) Programe en su lenguaje de programación orientado a objetos favorito las clases Hilo, GrupoHilo, Visualizador y Controlador. Programe únicamente la lógica básica suficiente para demostrar la implementabilidad del sistema que Usted propone. Valor 1.0

Puro dibujo es una abstracción.
El dibujo y el color no son contrarios, todo en la naturaleza es color.
Paul Cezanne