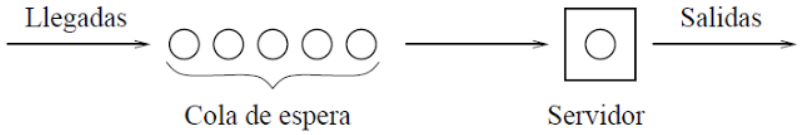
Universidad del Cauca
Departamento de Sistemas
Ingeniería de Software II

14 de enero de 2013 Profesor Julio Ariel Hurtado Alegría

Examen Final

1. Abstracción y Reutilización - 1.0 Punto

Considere un simulador orientado a eventos genérico, es decir un simulador esqueleto que simula un rango amplio de sistemas en los que hay solicitudes de servicios y servidores, cuando los servidores están ocupados las solicitudes típicamente se ponen en una cola y son atendidos en orden de llegada. El simulador contiene una cola de eventos y una variable que indica la 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 respecto 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());
     }
   }
 }
```

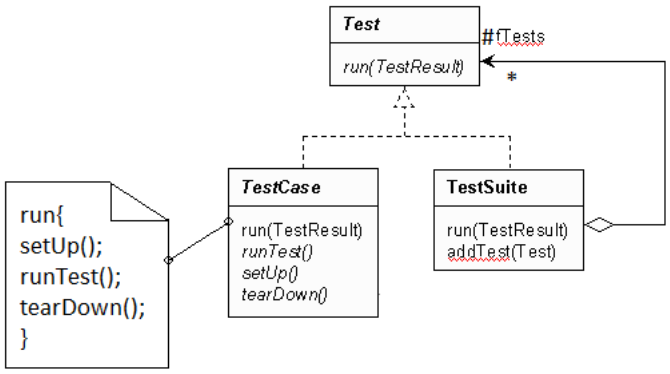
a) En la clase SimuladorGenerico descrita arriba, considere que Usted tiene que escribir un simulador para un sistema específico, por ejemplo la clase SimuladorBancoSimple (Un único Cajero, una única Fila (de clientes), eventos de llegada de un cliente y eventos de salida de un cliente de la atención del cajero). Dibuje las jerarquías de clases considerando el simulador específico y genérico, también considere en abstracto y en concreto los posibles eventos y la lógica específica del simulador del banco.

b) El diseño del simulador orientado a eventos genérico esta basado parcialmente sobre un patrón de diseño. ¿Cuál es el patrón? sustente su respuesta mediante un diagrama de clases y un diagrama de secuencia. Si está inseguro, sustente de la misma forma otras alternativas que Usted considere.

2. Reconocimiento de Patrones de Diseño Valor 1.0

JUnit es un framework para realizar y automatizar pruebas de aplicaciones Java. Es decir, JUnit soporta la disciplina de pruebas dentro del ciclo de desarrollo. La siguiente figura presenta la versión original creada por Erich Gamma (el mismo de los patrones GoF) y Kent Beck(El mismo de XP).

Una prueba es programada en una clase derivada de la clase *TestCase*, específicamente en el método *runTest()*. Antes de correr la prueba el framework está programado (método *setUP()*) para fijar unas condiciones iniciales de ejecución (inicializar algunos objetos y reservar recursos) y después de correr la prueba, está programado



(método *tearDown()*) para fijar unas condiciones finales de ejecución, útil por ejemplo para liberar recursos. Como la idea en procesos como XP, dónde existe la propiedad colectiva de código, es facilitar el cambiar contínuamente el código, pero garantizar la integridad, toda clase implementada requiere de un conjunto de casos de prueba, sumado a la cantidad de clases, se hace necesario manejar conjuntos de prueba y tratarlas en forma homogénea como si fuese una prueba más, en JUnit, se llama *TestSuite*. Una *TestSuite* permite la ejecución de varias pruebas.

a) Identifique dos patrones que está usando el framework JUnit. ¿Cuáles son los participantes?

b) Implemente el método *run()* de la clase *TestSuite*.

c) Explique el papel que cumple el principio de Hollywood y proponga la definición de un método Hook.

3. Caso de Aplicación de Patrones de Diseño - Valor 1.0

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 en los que las aplicaciones no requieren retener una conexión, se ha propuesto el desarrollo de una clase PoolConnections. ¿Qué patrón usaría para lograr esta solución? ¿Qué modificaciones se realizaron al patrón original para éste problema? Haga un diagrama de clases e implemente las

clases principales por completo. La lógica de conexión real a la base de datos déjela indicada mediante un comentario.

4. Notación UML - Valor 1.0

Revise todos los modelos de ésta prueba, cada uso inadecuado de UML les restará 1 décima hasta un máximo de 10 décimas. Uso inadecuado se refiere a la equivocada utilización de clases(abstractas y concretas), interfaces, relaciones como herencia, realización, composición, agregación, asociación y dependencia. También incluye el uso adecuado de la navegación, los roles, la multiplicidad. Igualmente los diagramas de secuencia o colaboración usados deben tener en cuenta condicionales, repetición de mensaje, paso de parámetros. En otras palabras, el UML usado en los puntos anteriores debe estar impecable para ganarse este punto, así que por favor revisite sus modelos y mejorelos lo que más pueda.

5. Arquitecturas de Software - Valor 1.0

El MVC es considerado un patrón arquitectónico, de acuerdo a la definición de arquitectura de software vista en clase determine cuales serían las estructuras, las componentes, los conectores, las propiedades externamente visibles que éste patrón expresa. Haga un gráfico a su gusto del patrón que concuerde con lo descrito y haga una descripción de la notación usada. Utilice este caso para mostrar la relación de la arquitectura de software con el diseño detallado, rescate el valor que tienen algunos principios de diseño orientado a objetos y los patrones de diseño.

Espero haberlos motivado lo suficiente
para seguir aprendiendo más de la ingeniería de software
Julio Ariel