Universidad del Valle
Facultad de Ingeniería
Escuela de Ingeniería de Sistemas y Computación
SISTEMAS OPERATIVOS (750008M)

Proyecto
"Uso de semáforos y monitores v2 (2011-09-07)"

INTRODUCCIÓN
El objetivo de este proyecto es interiorizar los conceptos de semáforos y monitores.

Puede hacerse en grupos de 2 estudiantes como máximo, pero la evaluación será individual. Solamente se recibirán trabajos entregados a través del Campus Virtual. Si se detecta copia (fuerte similitud entre dos o más trabajos de estudiantes, o con trabajos de semestres pasados o encontrados en internet, etc.), todos los estudiantes involucrados tendrán 0.0.

PROBLEMA A RESOLVER

Tenemos varias tiendas web, que venden libros al público. Para simplificar, todos los libros tienen el mismo precio P (pesos). Cada vez que ocurre una venta, la tienda reparte el 90% para el editor y el 10% para el autor (suponemos que hay un solo autor, para simplificar). El dinero se guarda en un banco y el editor y el autor son simplemente cuentas bancarias (son 2 números enteros). Hay un auditor que verifica continuamente esto: el auditor sabe cuantos libros se han vendido (N), le pregunta al banco cuanto dinero tienen el editor y el autor (E y A, respectivamente), y verifica que E sea (P*N*90)/100 y que A sea (P*N*10)/100. En caso de que no sea así, imprime en pantalla un mensaje de error (y permite que continúe la ejecución).

El objetivo es implementar las clases Banco, Tienda y Auditor. El Auditor debe ser un hilo (para que audite cuando lo desee). Todos los valores numéricos se implementarán como enteros. Para evitar problemas de redondeo, el precio P debe ser un múltiplo de 100 (por ejemplo, 10.000 pesos). Simule que hay 10 tiendas y cada tienda vende 1.000 libros.

a) Elija un lenguaje de programación orientado a objetos. Asegúrese de que dispone de semáforos, monitores e hilos.
   - Por ejemplo, en Java, los semáforos se consiguen con la clase **Semaphore** y los monitores anteponiendo la palabra **synchronize** a la clase que quiero proteger. Los hilos se consiguen con la clase **Thread**.
   - Por ejemplo, en C++ los semáforos, monitores e hilos se consiguen con **QSemaphore, QMutexLocker** y **QThread** respectivamente, siendo las tres clases de la biblioteca Qt.
   - Por ejemplo, en Ruby, los monitores se consiguen con la clase **Monitor**, los hilos con la clase **Thread**. Y no hay semáforos propiamente dichos, pero sí hay semáforos binarios: la clase **Mutex** con sus funciones **lock** y **unlock**.
   - Lo más recomendable es usar uno de estos tres lenguajes, pero si desea usar otro, asegúrese de que sea OO y que tenga acceso a estas tres construcciones.
b) Lea la documentación respectiva de su lenguaje elegido, respecto a estas tres clases.
c) Escriba una clase **Banco**, que va a ser el recurso compartido. Debe contener dos variables enteras (**dineroAutor** y **dineroEditor**) y disponer de cuatro funciones **cuantoDineroAutor(), cuantoDineroEditor(), guardarDineroAutor(), guardarDineroEditor()** que lean y escriban (añadiendo un valor) respectivamente en esas variables.
d) Escriba una clase **Tienda**, que sea un hilo y que cada vez que ejecutemos su funcion **libroVendido()** guarde el correspondiente dinero en el **Banco**, en las respectivas cuentas**.** Esta función también debe llamar a la función **libroVendido()** de la clase **Auditor()**, para informarle que se vendió un libro.

e) Escriba una clase **Auditor**, que sea un hilo y ejecute repetidamente y en cualquier momento su función **auditar()**. Esta función debe verificar las fórmulas mencionadas arriba e imprimir en pantalla un mensaje de error cuando detecte inconsistencias financieras. También tiene la función **libroVendido()** que simplemente incrementa una variable entera interna (el **numeroDeLibrosVendidos**, o sea, la N de la que hablábamos antes). Y al finalizar debe indicar cuantos libros se vendieron y cuantas inconsistencias hubo.

f) El programa principal debe ser (esto es solo un boceto que debe adaptar a su lenguaje, añadiendo el manejo de hilos, etc.):

```
main()
{
    // Creacion de los objetos:
    banco = Banco.new;
    auditor = Auditor.new(banco);
    tienda1 = Tienda.new(banco, auditor);
    tienda2 = Tienda.new(banco, auditor);

    // Ejecucion de hilos concurrentemente:
    auditor.start();
    tienda1.start();
    tienda2.start();

    // Esperar que los hilos terminen:
    tienda1.join();
    tienda2.join();
    auditor.informeFinal();
}
```

g) Compile y ejecute el programa. Explique lo que ve.

h) Corrija el problema usando un semáforo para proteger las secciones críticas de los hilos. NOTA: es muy sencillo hacerlo, pues se requiere un único semáforo.

i) Corrija el problema pero ahora convirtiendo la clase **Banco** a un monitor. NOTA: es algo complicado hacerlo, pues un monitor protege un recurso, y aquí la información a proteger está distribuida en dos recursos: el banco (con sus cuentas) y el auditor (con el número de libros vendidos). Una solución a ello sería crear un nuevo objeto que centralice los recursos. Pero hay más soluciones.