Prácticas de Programación Concurrente y Distribuida

3º Curso de Grado en Ingeniería Informática

Curso 2014-15

EXAMEN

Enero de 2015

CONSIDERACIONES PREVIAS:

- **No se permite el uso de ningún tipo de documentación.**
- **El acceso a Internet está desactivado conscientemente.**
- **Apague el teléfono móvil.**

ANTES DE COMENZAR EL EXAMEN:

- Cree una carpeta con su nombre y primer apellido en el **Escritorio** separados por un guión bajo (ejemplo: **`Pedro_Abad`**).
- En dicha carpeta aparecerá un proyecto por cada una de las preguntas del examen. Dichos proyectos se denominarán **`Proyecto1`**, **`Proyecto2`**,…, **`Proyecto5`**.

ENUNCIADO:

Un ambulatorio dispone de una unidad de curas. A dicho ambulatorio llegan dos tipos de pacientes, los que tienen cita y los que no. La unidad de cura, atenderá a los pacientes sin cita únicamente cuando no haya pacientes con cita esperando.

PROYECTO1.

Tiempo estimado: 15 minutos.
Puntos: 1,5

Será el proyecto base para solucionar el enunciado. Contendrá las siguientes clases:

- **`Concita`**. Representará cada uno de los pacientes que tienen cita mediante un hilo. El hilo se creará heredando de la clase `Thread`. El hilo pondrá un mensaje de inicio indicando su identificador, esperará una cantidad aleatoria de segundos entre 2 y 5 (tiempo de cura) y pondrá un mensaje final con su identificador indicando que ha finalizado la cura.

- **`Sincita`**. Representará cada uno de los pacientes sin cita mediante un hilo. El hilo se creará implementando el *interface* `Runnable`. El hilo pondrá un mensaje de inicio indicando su identificador, esperará una cantidad aleatoria de segundos entre 2 y 5 (tiempo de cura) y pondrá un mensaje final con su identificador indicando que ha finalizado la cura.

- **`Generador`**. Contendrá el método `main` y será quién comience la ejecución. Debe lanzar, de forma aleatoria, pacientes de cada tipo a intervalos de tiempo de entre 1 a 3 segundos. Deberá esperar a que finalicen todos los hilos para finalizar.

Dado que es el proyecto base, no se controlará el acceso a la sala de curas.

PROYECTO2.

Tiempo estimado: 25 minutos.
Puntos: 2,5

Se añadirá al *Proyecto1* una nueva clase llamada **`Sala`**. La clase `Sala` controlará el acceso a la sala de curas mediante la implementación de los siguientes métodos:

- **`ConcitaIN`**. Que deberá ser invocado por los pacientes con cita cuando quieren ser atendidos.
- **`ConcitaOUT`**. Que deberá ser invocado por los pacientes con cita al terminar de ser atendidos.
- **`SincitaIN`**. Que deberá ser invocado por los pacientes sin cita cuando quieren ser atendidos.
- **`SincitaOUT`**. Que deberá ser invocado por los pacientes sin cita al terminar de ser atendidos.

Se modificarán también el comportamiento de pacientes para controlar el acceso a la sala de curas mediante la invocación de los métodos de la clase `Sala`.

El control de la concurrencia y la sincronización se realizará en la clase `Sala`, mediante las primitivas de Java `wait()`, `notify()` y/o `notifyAll()`.

PROYECTO3.

Tiempo estimado: 30 minutos.
Puntos: 3

Se modificará el *Proyecto2* para que la clase `Sala` controle la concurrencia mediante `ReentrantLocks` y `Conditions`.

Se modificará también la forma de acceder de los pacientes, de forma que se atienda a un paciente sin cita si se han atendido dos pacientes con cita consecutivos mientras había pacientes sin cita esperando.

PROYECTO4.

Tiempo estimado: 15 minutos.
Puntos: 2

Tomará como base el *Proyecto2* y se modificará en la forma necesaria, de manera que **generador** permita el acceso de pacientes de forma remota, haciendo uso de R.M.I. Es decir, generador hará de servidor del objeto de la clase `Sala`.

PROYECTO5.

Tiempo estimado: Depende de la implementación que se pretenda
Puntos: 1

Se creará un *Applet* que visualice de forma gráfica, mediante un *Canvas*, la situación de las colas de espera y la ocupación de la sala para el *Proyecto2*.

El nivel de representación gráfica es libre, pero al menos, se deberán identificar la ocupación de la sala y las colas de pacientes.