PROGRAMACIÓN AVANZADA
Práctica de Laboratorio
Convocatoria Ordinaria – Abril 2015

Publicado el 24 de abril de 2015

Simulación de la escalada al pico K8

Se pretende simular el comportamiento de unos **escaladores** (threads) que intentan coronar el K8. La subida comienza con un **Funicular** que asciende desde el **Llano** hasta la cota de los 4000m donde se encuentra el **CampoBase**. Allí los escaladores pasan un tiempo para aclimatarse a la altura, tras lo que intentan subir hasta el **CampoUno** que tiene una capacidad limitada y en el que los escaladores esperan otro tiempo de aclimatación y después intentan coronar la **Cumbre** del pico K8. Tras coronar y contemplar el paisaje un rato, los escaladores bajan directamente hasta el **CampoBase** sin pasar ya por el **CampoUno** y tras un nuevo periodo de adaptación, toman el **Funicular** para regresar abajo y terminar su aventura.

Cuando las condiciones climatológicas son adversas, las autoridades prohíben la subida a la **Cumbre**. En ese caso, los escaladores que ya hayan agotado el tiempo de aclimatación en el CampoUno deben desistir de coronar y terminan su aventura de la misma forma que los escaladores que sí lo consiguieron. Cuando el tiempo cambia y las autoridades levantan la prohibición de subir a la Cumbre, todo vuelve a funcionar como al principio.

Un **Vigilante,** que estará pendiente de las condiciones climatológicas en la cumbre, vigilando a distancia, podrá prohibir o permitir la subida a la **Cumbre**.

En un **Monitor** situado a distancia se podrá observar en todo momento y en tiempo real, el número total de alpinistas que han utilizado el **Funicular** en cualquiera de los sentidos, subida o bajada.

Queremos crear una aplicación en Java Concurrente que simule el comportamiento del sistema (colas de espera, tiempos, mecanismos de sincronización etc.) que permita también comunicarse con él desde otras aplicaciones Java Distribuidas, ejecutándose en el mismo equipo o en equipos diferentes.

Para ello hay que construir un modelo en el que estén representados:

1. Los *escaladores* (threads de la clase **Escalador**), de los que se crearán 100 numerados del 1 al 100. En todo momento deberá poder conocerse en qué punto se encuentra cada escalador, mostrándose su número en alguno de los campos de texto de la interfaz gráfica.
2. El *llano* (recurso compartido de la clase **Llano**) es el lugar donde estarán los escaladores antes de comenzar su ascensión y donde se quedarán apuntados cuando hayan terminado.
3. El *funicular* (recurso compartido de la clase **Funicular**) es un objeto con movimiento (*thread*), que tiene una única cabina con capacidad para 4 escaladores, que espera abajo un rato

mientras se montan escaladores, sube (lo que le lleva un tiempo) y espera arriba otro rato para que se monten hasta 4 escaladores que quieran bajar y baja (a la misma velocidad que subió, completando así el ciclo y volviendo a empezar.

4. El *campoBase* (recurso compartido de la clase **CampoBase**) en la que se mezclan los escaladores que están subiendo con los que están bajando, y donde permanecen un tiempo antes de abandonarlo. La capacidad del campoBase es ilimitada.
5. El *campoUno* (recurso compartido de la clase **CampoUno**) en el que solamente caben 10 escaladores. Cuando está completo, los escaladores del campoBase deben esperar por riguroso turno a que haya un hueco para ascender. Tras un periodo de aclimatación en el campoUno, los escaladores intentarán ascender a la cumbre de uno en uno. Si el mal tiempo lo impide y ya han consumido el tiempo de aclimatación, se deben volver al campoBase.
6. La *cumbre* (recurso compartido de la clase **Cumbre**) donde solo puede haber un escalador en cada momento. El objeto *cumbre* deberá poseer los métodos para *abrirCumbre()* y *cerrarCumbre()*, que serán invocados remotamente, mediante RMI, por el *vigilante*.
7. El *vigilante* (cliente RMI de la clase **Vigilante**), será un JFrame con botones para abrir y cerrar el ascenso a la cumbre.
8. El *monitor* (servidor de sockets de la clase **Monitor**) esperará conexiones por socket del cliente **Funicular**, que le irá enviando el total de escaladores que se han montado. El monitor será un JFrame con un campo de texto y deberá poder pararse y arrancarse en cualquier momento.

Funcionamiento general:

1. El programa principal será el encargado de crear los objetos que representan los elementos del pico K8 y de los 100 escaladores. Será un JFrame con 5 campos de texto, uno por cada elemento, en los que se mostrarán los números de los escaladores que se encuentren en ese sitio en cada momento. Además, el funicular tendrá asociada una JLabel que indique si está subiendo o bajando y la cumbre, tendrá otra que mostrará si el ascenso está permitido o prohibido. Mediante un botón se deberá poder "detener" o "reanudar" la marcha de los escaladores, de forma que se pueda comprobar mejor el funcionamiento del sistema.
2. Los escaladores deben esperar en cada sitio un tiempo. Los tiempos pueden ajustarse para que el sistema pueda ser observado convenientemente. Como sugerencia, se proponen los siguientes tiempos:
   - Estancia en el llano para subir: aleatorio entre 0,1 y 0,4 seg.
   - Aclimatación en el campo base: aleatorio entre 0,3 y 0,6 seg.
   - Aclimatación en el campo uno: fijo de 1 seg.
   - Contemplación en la cumbre: aleatorio entre 1 y 3 seg.
3. Asimismo, el funicular realizará un ciclo con los tiempos siguientes:
   - Apertura de puertas para que monten alpinistas: 1seg.
   - Tiempo de marcha (subida o bajada): 1 seg.
4. Cuando el sistema esté **detenido** por haberse pulsado el botón correspondiente, los números de cada uno de los alpinistas se visualizarán en la campo de texto correspondiente a la situación en que se encuentren en ese momento.

Módulo vigilante:

5. Tendrá a su disposición dos métodos remotos ofrecidos, mediante RMI de Java, por algún módulo del programa principal. El método **cerrarCumbre()** impedirá que asciendan escaladores a la cumbre y el método abrirCumbre() volverá a permitir el ascenso a la cumbre. La interfaz con dichos métodos deberá llamarse **InterfaceCumbre**.

Módulo Monitor:

6. Será un servidor que creará sockets en el puerto 5000 por el recibirá del funicular el total de viajeros que lo han usado. El funicular, cada vez que llegue abajo, abrirá un nuevo socket y enviará al monitor dicho total. El monitor podrá pararse y arrancarse sin que deje de funcionar correctamente.

Condiciones de entrega

1. *La práctica es individual y deberá ser entregada a través de la tarea correspondiente del Aula Virtual, mediante la subida de dos archivos: la memoria de la práctica en formato PDF o DOC y el proyecto Netbeans completo, comprimido como ZIP o RAR. El plazo de entrega finaliza **el día 14 de mayo por la noche**.*
2. *Es condición necesaria para aprobar, que todos los programas funcionen correctamente y de acuerdo a las especificaciones indicadas en los enunciados.*
3. *Es condición necesaria para aprobar, que todos los nombres de las clases comiencen por una letra mayúscula y todos los nombres de atributos y métodos comiencen por una letra minúscula.*
4. *Para detener y reanudar los hilos es obligatorio utilizar la clase **Paso** que se desarrolló en la sesión 3 del laboratorio.*
5. *El id de cada escalador deberá estar en la lista asociada a cada uno de los campos de texto que representan los elementos del sistema, tanto si se encuentran haciendo tiempo (sleep), como si están esperando para acceder al siguiente lugar. El acceso a un lugar nuevo, supone sacarlo de la lista del lugar anterior y meterlo en la lista del nuevo, sin que se pueda detener el hilo entre ambas operaciones.*
6. *Para representar las cinco listas de escaladores necesarias (formadas cada una, por un ArrayList y un JTextField), se utilizará obligatoriamente la clase **ListaThreads** utilizada en la sesión 6 del laboratorio.*
7. *Todos los escaladores deberán estar visibles en alguno de los cinco JTextField de la ventana principal. Dicha información deberá actualizarse en tiempo real y deberá poderse verificar que es correcta cuando el sistema esté detenido.*
8. *En la portada de la memoria deberán figurar los datos siguientes:*
    a. ***Grado en Ingeniería _____** (Informática o de Computadores)*
    b. ***Curso 2014/2015 – Convocatoria Ordinaria***
    c. ***DNI – Apellidos, Nombre***
9. *La memoria explicativa de la práctica realizada deberá incluir, en el orden siguiente: 1) un análisis de alto nivel; 2) diseño general del sistema y de las herramientas de sincronización utilizados en cada lugar del K8; 3) soluciones cliente-servidor utilizadas para implementar los*

módulos Vigilante y Monitor; 4) las clases principales que intervienen con su descripción (atributos y métodos) 5) un diagrama de clases que muestren cómo están relacionadas

10. *Dicha documentación, exceptuando el código, no deberá extenderse más de 20 páginas. La calidad de la documentación – presentación, estructura, contenido, redacción – será un elemento básico en la evaluación de la práctica.*
11. *Para la defensa de la práctica, deberá presentarse una copia en papel de la memoria, impresa por las dos caras y grapada. Este documento podrá ser utilizado por el estudiante como base para responder a las cuestiones que se le planteen en el ejercicio escrito sobre la realización de la aplicación.*
12. *Para mostrar el funcionamiento de los programas, es conveniente que cada estudiante utilice su propio ordenador portátil, en previsión de posibles problemas al instalarlos en alguno de los ordenadores del laboratorio.*