Programación Concurrente y de Tiempo Real
Grado en Ingeniería Informática
Asignación de Prácticas Número 2

Se le plantean a continuación un conjunto de ejercicios sencillos de programación sobre el uso del multihebrado en Java, utilizando las distintas técnicas que este lenguaje ofrece para ello; también se realizará una introducción elemental al paralelismo de datos. Resuelva estos ejercicios como complemento a la segunda sesión práctica. Para cada uno, debe desarrollar un programa independiente que lo resuelva. **Documente todo su código con etiquetas, de forma que pueda generarse la documentación asociada con** `javadoc`.

1. Ejercicios

1. Utilizando herencia de la clase `Thread`, cree una condición de concurso sobre una variable común n (valor inicial 0) entre dos hilos que respectivamente incrementen y decrementen el mismo número de veces a n. Lance ambos hilos concurrentemente utilizando `start()-join()` y compruebe que, aunque el valor teórico final debe ser cero, en la práctica no tiene por qué ser así. Guarde su código en `hebra.java` y `Usa_hebra.java`. Escriba (utilizando LaTeX vía *OverLeaf*) una corta tabla de prueba en `tabla.pdf` donde recogerá el número de iteraciones que realizaron los hilos y el valor final obtenido para n junto con su análisis acerca de todo ello.

2. Utilizando implementación de la interfaz `Runnable`, cree una condición de concurso sobre un objeto común que albergará una variable entera n con valor inicial 0. La clase que modela al objeto tendrá dos modificadores que respectivamente incrementen y decrementen a la variable n, y un observador para conocer su estado. Ahora, cree dos hebras que compartan el acceso a un objeto de la clase ya construida (una que incremente y otra que decremente) concurrentemente y compruebe que, aunque el valor teórico final de n debe ser cero, en la práctica no tiene por qué ser así. Guarde su código en `tareaRunnable.java` y `Usa_tareaRunnable.java`. Escriba (utilizando LaTeX vía *OverLeaf*) una corta tabla de prueba en `tabla2.pdf` donde recogerá el número de iteraciones que realizaron los hilos y el valor final obtenido para n junto con su análisis acerca de todo ello.

3. Se desea realizar el escalado de un vector de números enteros de 10^6 componentes. Escriba un programa secuencial `escalaVector.java` que haga el trabajo. Ahora, escriba una nueva versión paralela multihebrada y llámela `escalaVPar.java` que utilice 4 hebras paralelas con división manual del dominio de datos (vector) entre ellas. Escriba un documento que incluya una tabla de análisis `tablaCPU.pdf` que deberá recoger de forma aproximada los picos de uso máximo de la CPU como una función del tamaño del vector ($10^6, 2{\times}10^6, 3{\times}10^6$...) y del tipo de procesamiento empleado. No tiene por qué haber una mejora radical del aprovechamiento del procesador.

4. Escriba en `cuentaCorriente.java` una clase que modele una cuenta corriente. Incorpore al menos los atributos número de cuenta y saldo, y los métodos depósito y reintegro. Simule ahora, utilizando hilos mediante tareas por implementación de la interfaz `Runnable` a una red de cajeros automáticos, donde cada cajero (guardar en fichero `cajero.java`) realiza una operación de ingreso o de reintegro sobre un cuenta corriente. Provoque ahora una condición de concurso de los hilos contra una instancia de la clase anterior, de forma que la suma neta de las operaciones de todos ellos sea igual a 0. En esta situación, el saldo inicial de la cuenta debería haber permanecido constante. Compruebe que no tiene por qué ser así. Guarde la clase que crea y simula a los cajeros en `redCajeros.java`.

5. Escriba ahora una condición de concurso de dos tareas sobre una variable compartida, utilizando para modelar a la tareas expresiones λ. Guarde el código en `concursoLambda.java`