

Programación Concurrente y de Tiempo Real

Grado en Ingeniería Informática

Asignación de Prácticas Número 6

Se le plantean a continuación un conjunto de ejercicios sencillos de programación de control de la exclusión mutua y uso de ejecutores, que debe resolver de forma individual como complemento a la sexta sesión práctica. Para cada uno, debe desarrollar un programa independiente que lo resuelva. **Documente todo su código con etiquetas (será sometido a análisis con javadoc).**

1. Ejercicios

1. En la carpeta de la práctica se le proporciona el código necesario para implantar una solución cliente-servidor simple multihebrada. El inconveniente de esta solución es la necesidad de crear un *thread* para dar servicio a cada petición de un cliente recibida por el servidor. Una solución mucho más elegante es modelar la tarea de servicio mediante objetos `Runnable`, y delegar su ejecución a un *pool* de *threads*. Se pide:

- Reescriba el código del servidor de forma que cada petición de servicio sea atendida por una hebra que procesará un objeto de clase `ThreadPoolExecutor`, y guárdelo en `ServidorHiloconPool.java`.
- Escriba ahora una versión del cliente que genere un número fijo de peticiones al servidor, y guárdela en `clienteMultiple.java`.

2. Deseamos que varias hebras soportadas mediante herencia de la clase `Thread` utilicen un TAD lista de enteros, implementado con un array de forma segura. Provea una solución llamada `tADListaSegura.java` que cumpla con la especificación descrita, utilizando cerrojos `synchronized`. Ahora, escriba un programa en `usaTADListaSegura.java` que instancie una lista

segura y múltiples hebras soportadas mediante la interfaz `Runnable` que la utilicen.

3. Escriba una clase en `heterogenea.java` que tendrá dos atributos enteros n, m a incrementar mediante **métodos diferentes**; incluya también un par de método observadores. Proteja los métodos que gestionan a n mediante `synchronized`, mientras que los métodos que gestionan a m no tendrán control alguno. A continuación, escriba en `usaheterogenea.java` un programa donde múltiples hebras accedan a un objeto de clase `heterogenea` concurrentemente, y compruebe, estudiando los valores finales de n y m , que en el mismo objeto de Java pueden existir regiones de código bajo exclusión mutua y sin ella.

4. Escriba, utilizando `synchronized`, un código donde tres hebras diferentes (soportadas por herencia de `Thread`) entren en *deadlock*. Guarde el código en `deadlock.java`

5. Escriba una versión de la integración paralela de *Monte-Carlo* para aproximar la integral definida de la función $f(x) = \cos(x)$ en $[0, 1]$ utilizando tareas `Callable` y la interfaz `Future`; genere los números aleatorios necesarios utilizando `ThreadLocalRandom`. Guarde el código resultante en `integCallable.java`