Programación Concurrente y de Tiempo Real
Grado en Ingeniería Informática
Asignación de Prácticas Número 7

En esta asignación aplicará control de exclusión mutua y sincronización, utilizando para ello el API estándar de Java, efecuando la implementación con monitores a soluciones de problemas clásicos de la concurrencia. `Documente todo su código con etiquetas (será sometido a análisis con javadoc)`.

tenemos dos necesidades con los monitores en java con el api estándar es de la condición de guarda, se despiertan hebras q luego son dormidas por lo que no es eficiente en rendimiento por el notifyall porque notify no sabemos

1. Enunciados

1. En el Tema 3 del curso teórico se ha analizado el concepto de monitor, y se ha resuelto el problema del productor-consumidor con él; a partir de aquí:
   - repase el funcionamiento del monitor productor-consumidor desde un punto de vista teórico, utilizando la descripción del texto de Ben-Ari.
   - navegue a `https://github.com/motib/concurrent-distributed/blob/main/Java/PCMonitor.java` y descargue la propuesta de implementación de Ben-Ari para un monitor (API estándar) que da solución en Java al productor-consumidor; analice el código; preste especial atención a las diferencias entre el modelo teórico del monitor productor-consumidor, que utiliza dos variables de condición para sincronizar, y su implementación en Java, que debe lograr la sincronización con un único *wait-set*; a continuación:
   - escriba un diseño de hebras productoras y consumidoras que utilicen este monitor, y guárdelo en `usaProdCon.java`.
   - reduzca el tamaño del buffer a uno y vea qué ocurre.
   - ejecute un productor y varios consumidores y vea qué ocurre.
   - ejecute varios productores y un consumidor y vea qué ocurre.
   - redacte un documento `analisis.pdf` con sus impresiones acerca de las diferentes pruebas de ejecución propuestas.
2. El problema de los lectores-escritores es otro problema clásico de la concurrencia. En él, hebras lectoras y escritoras tratan de acceder a un recurso común bajo los siguientes criterios de control:
   - Los lectores pueden acceder al recurso siempre que no haya un escritor escribiendo en él.

por lo tanto, 1 escritor con varios lectores

- Los escritores deben esperar a que no haya ni otro escritor, ni lectores, accediendo al recurso.
- Revise la descripción del problema en el texto de Palma et.al. Se le pide que programe una solución equivalente en Java utilizando un monitor soportado por el API estándar de control de la concurrencia. Guarde el monitor en `monitorLE.java`. Escriba también un diseño de hebras lectoras y escritoras en `usaMonitorLee.java` que emplee el monitor. Como recurso externo a controlar con el monitor, utilice un objeto de la clase `recursoParaLE.java` que se le proporciona.