Programación Concurrente y de Tiempo Real
Grado en Ingeniería Informática
Asignación de Prácticas Número 8

La presente asignación tiene por objetivo continuar trabajando en el desarrollo de soluciones de control de la concurrencia basadas en monitores teóricos, y en su implementación posterior utilizando el lenguaje Java y las primitivas de control que este provee para ello en el API estándar para control de concurrencia. Documente todo su código con etiquetas (será sometido a análisis con javadoc). Si lo desea, puede también agrupar su código en un paquete de clases, aunque no es obligatorio

1. Enunciados

1. Se dispone de n procesos concurrentes que comparten tres dispositivos de impresión. Antes de poder utilizarlos, un proceso debe pedir que le asigne una de las impresoras que esté libre, si la hay. En caso contrario, deberá esperar. En caso de que la haya, el sistema asignará una impresora disponible al proceso peticionario. Cuando la tarea de impresión concluye, el proceso debe liberarla explícitamente. Se pide:

- Desarrolle un monitor teórico que modele el sistema, utilizando variables de condición y disciplina de señalización `SC` (señalar y continuar). Guarde su solución en un documento `monitorImpresion.pdf`. Deberá desarrollar el documento utilizando LaTeX con *OverLeaf*.
- A partir del monitor anterior, efectúe su implementación en lenguaje Java, aplicando el protocolo de diseño de monitores con el API estándar, encapsulando los recursos necesarios, escribiendo el conjunto de métodos que proporcionen la gestión desarrollada en el monitor teórico, y sincronizando cuando sea necesario. Recuerde que debe utilizar condiciones de guarda en todo momento. Guarde el código del monitor en `monitorImpresion.java`, y escriba ahora un programa que lo utilice donde haya algunos hilos concurrentes. Guarde este programa en `UsamonitorImpresion.java`.

2. Un problema clásico de la programación concurrente es el conocido como problema de los filósofos, que ya le ha sido presentado en clase de problemas desde un punto de vista teórico. En esta ocasión, deseamos implementar una solución a este problema utilizando un monitor construido con el API estándar de Java. Para ello debe:

- Leer el documento `filosofos.pdf` disponible en la carpeta de la práctica, extraido del libro de Ben-Ari, que analiza el problema y propone un modelo de monitor teórico. Sáltese el análisis del Teorema 7.5, donde se prueba que el monitor propuesto está libre de interbloqueos.
- Aplicar el protocolo ya conocido para a partir de este modelo teórico, desarrollar un equivalente funcional en Java con el API estándar que guardará en `forkMonitor.java`. Recuerde que debe utilizar condiciones de guarda en todo momento. Finamente escriba un programa que guardará en `usaforkMonitor.java`, que active a cinco filósofos (hebras) que se sincronizan a través del monitor ya implementado.

Nota: el array `fork` del monitor teórico propuesto en la solucion de Ben-Ari tiene estructura circular. Téngalo en cuenta a la hora de computar los índices `i+1` e `i-1` que indexan dicho array en las operaciones de asignación ($\leftarrow$) situadas en los métodos `takeForks` y `releaseForks` de esa solucion.

2. Procedimiento y Plazo de Entrega

PRODUCTOS A ENTREGAR

- Ejercicio 1: `monitorImpresion.pdf`, `monitorImpresion.java`, `UsamonitorImpresion.java`,
- Ejercicio 2: `forkMonitor.java` y `usaforkMonitor.java`.

MÉTODO DE ENTREGA: Tarea de Moodle.