Programación Concurrente y de Tiempo Real

PRÁCTICA 4

Mª Teresa Supervielle Sánchez

1. Ejercicio 1: tryThree.java y tryFour.java

Una vez leída la documentación indicada e implementado y ejecutado los dos algoritmos, podemos observar que, en el primer algoritmo, el tryThree.java, entra aposta en *Livelock*, eso significa que el algoritmo se sigue ejecutando en un bucle infinito, sin llegar a hacer nada útil. Por otro lado, el segundo algoritmo, tryFour.java si se ejecuta correctamente dando como resultado 0, es decir, lo esperado.

2. Ejercicio 2: algDekker.java

El algoritmo de Dekker permite a dos procesos compartir un recurso **(n)** sin conflictos, para la implementación de la misma se hace uso de un variable **(turn)** la cuál elige cuál de los dos procesos es el que accederá a la Sección Crítica.
En este caso, el algoritmo nunca termina su ejecución (debido a las restricciones de implementación), se encuentra en un bucle infinito, pero el intercambio de hilos se hace correctamente, en ningún momento se bloquea.

3. Ejercicio 3: algPeterson.java

El algoritmo de Peterson es una simplificación del algoritmo de Dekker, al seguir el mismo razonamiento de este, también se ejecuta correctamente, por lo que la exclusión mutua de la Sección Crítica esta asegurada.
En el caso de la implementación propuesta, tampoco termina su ejecución pero no se produce ningún tipo de bloqueo, la alternancia entre hilos se da de manera correcta.