

PRÁCTICA 2

Objetivos de la práctica:

1. Condición de carrera
2. Exclusión mutua
3. Implementación del concepto de *volatile* sobre arrays

Parte 1: Evitar condición de carrera con espera activa. La primera parte consiste en evitar la condición de carrera que se produjo en la práctica anterior. Para ello supondremos la existencia de **dos procesos**, que simultáneamente ejecutan sendos bucles de N pasos incrementando y decrementando, respectivamente, en cada paso una variable compartida (la operación de incremento y la de decremento sobre esa misma variable compartida constituyen las secciones críticas). El objetivo es evitar que mientras un proceso modifica la variable el otro haga lo mismo (propiedad que se denomina exclusión mutua: no puede haber dos procesos modificando concurrentemente esa variable) y el objetivo es hacerlo utilizando bien el algoritmo “Rompe-empate” o el algoritmo de “Bakery” **para dos procesos** explicado en clase.

Parte 2: Generalizar la parte 1 para que funcione con 2M procesos (M incrementadores y M decrementadores). Crea tres clases `LockRompeEmpate`, `LockTicket` y `LockBakery` que implementen los tres algoritmos vistos en clase para un número paramétrico de procesos (cada una de ellas con un método `takeLock` y un método `releaseLock`), y utilízalas para garantizar la exclusión mutua en el ejemplo de M incrementadores y M decrementadores.

Investiga el funcionamiento de *volatile* sobre arrays en Java.

No está permitido utilizar métodos `synchronized`, locks, semáforos, ni ningún otro mecanismo de concurrencia.