PROGRAMACION CONCURRENTE (Curso 2024/2025)

PRÁCTICA 1
FECHA REALIZACIÓN: semana 27-31 enero

Objetivos de la práctica:

1. Iniciación a la concurrencia en Java
2. Creación de threads en Java
3. Condiciones de carrera
4. Clase *Thread* de Java
5. Modificador *volatile*

INTRODUCCION A LA CONCURRENCIA

Parte 1: Creación de procesos (threads). Escribe un programa concurrente en Java que cree N procesos (threads) y termine cuando los N threads terminen (es decir, creación de procesos con "join"). A cada thread se le asignará un identificador único. Todos los threads deben realizar el mismo trabajo: imprimir su identificador, dormir durante T milisegundos y terminar imprimiendo su identificador. El thread principal además de poner en marcha los procesos, debe imprimir una línea avisando de que todos los threads han terminado una vez lo hayan hecho. Debes observar como la ejecución lleva a resultados diferentes, para ello puedes jugar con los valores N y T, asignando un T distinto a cada proceso. Consulta la documentación sobre la clase *Thread* de Java.

Parte 2: Provocar salida indeterminista. Escribir un programa concurrente en el que múltiples threads compartan y modifiquen una dato de tipo "Entero" sin utilizar ningún mecanismo de sincronización de forma que el resultado final, una vez que los threads terminan, pueda no ser el valor esperado. Tendremos dos tipos de procesos, decrementadores e incrementadores que realizan N decrementos e incrementos, respectivamente, sobre una misma variable (n) de tipo int inicializada a 0 y almacenada **en un objeto de la clase Entero** al que tienen acceso todos los threads. Investiga el uso de *volatile*. El programa concurrente pondrá en marcha M procesos de cada tipo y una vez que todos los threads hayan terminado imprimirá el valor de la variable compartida. En un programa concurrente correcto, el valor final de la variable debería ser 0 ya que se habrán producido M × N decrementos (n–) y M × N incrementos (n++), sin embargo, sin utilizar herramientas de sincronización, el resultado puede no ser el esperado.

Parte 3: Multiplicación de matrices por N threads. Implementa la multiplicación de dos matrices de tamaño NxN utilizando N threads de manera que cada thread calcule una fila del producto.