PROGRAMACION CONCURRENTE (Curso 2020/2021)

PRÁCTICA 1
FECHA REALIZACIÓN: semana 22-26 de febrero

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

INTRODUCCION A LA CONCURRENCIA

Parte 1: Creación de procesos (threads). Escribe un programa concurrente en Java que arranque N procesos (threads) y termine cuando los N threads terminen. 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, e incluso se puede asignar un T distinto a cada proceso. Consulta la documentación sobre la clase *Thread* de Java.

Parte 2: Provocar una condición de carrera. Escribir un programa concurrente en el que múltiples threads compartan y modifiquen una variable de tipo int de forma que el resultado final de la variable una vez que los threads terminan no sea 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 almacena en un objeto Entero al que tienen acceso todos los threads. El programa concurrente pondrá en marcha M procesos de cada tipo y una vez que todos los threads han terminado imprimirá el valor de la variable compartida. El valor final de la variable debería ser 0 ya que se habrán producido M $\times$ N decrementos (n–) y M $\times$ N incrementos (n++), sin embargo, si dos operaciones (tanto de decremento como de incremento) se realizan a la vez el resultado puede no ser el esperado (por ejemplo, dos incrementos podrían terminar por no incrementar la variable en 2).

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 calcula una fila del producto.