



Universidad Autónoma de Nuevo León  
**Facultad de Ciencias Fisico Matematicas**



**Laboratorio de Programacion Orientada a Objetos**  
**Practica 10:**  
**Documentacion de estrategias de sincronizacion**

Luis Enrique Matta Gonzalez  
Grupo 031

2114428

**Objetivo.**

Asegurar consistencia de datos y progreso bajo concurrencia. Se evita condicion de carrera, sobre todo en operaciones bancarias y en el patron productor–consumidor.

**CuentaLEMG (seccion critica).**

Se usan metodos synchronized en depositar, retirar y getSaldo. La invariante es saldo consistente en todo momento. Concurrencia de cajeros y clientes no rompe el saldo.

**Productor–Consumidor (Buffer4428).**

Monitor con wait() y notifyAll(). El productor espera cuando el buffer esta lleno; el consumidor espera cuando esta vacio. La invariante es tamano entre 0 y BUFFER\_SIZE.

**Por que notifyAll().**

Hay multiples productores y consumidores. notifyAll() evita perdidas de señal y hambruna al despertar a todos los hilos que compiten por la misma condicion.

**Pool de hilos.**

ThreadPoolMatta usa cola protegida y workers que esperan con wait(). Cierre ordenado al vaciar cola. Se compara contra ExecutorService para mostrar costo/beneficio.

**Prevencion de deadlocks.**

Sin locks anidados ni ordenes cruzados. Secciones criticas cortas. No se hace I/O dentro de regiones sincronizadas. Esto elimina ciclos de espera y mejora latencia.

**Manejo de condiciones de carrera.**

Acceso a saldo siempre bajo lock. El buffer solo se toca con los metodos sincronizados del monitor. Las pruebas confirman buffer en 0 al final y saldo estable.

**Medicion y trazas.**

Log de exec\_2114428.log. Nivel INFO para hitos (inicio, fin, tiempos). Nivel FINE para detalles (depositos, retiros, esperas del buffer).

El archivo se genera en la ejecucion en el directorio test/practica10/exec\_2114428.log.

**Resultados.**

Las demos muestran ausencia de carreras, procesamiento completo de trabajos y tiempos comparables entre ExecutorService y el pool custom. Para produccion, preferir BlockingQueue y ExecutorService con politicas y manejo de errores.

