**CICLO JAVA - CONCURRENCIA**

**EJERCICIOS**

**Ejercicio 1 - Suma paralela de elementos de un arreglo**

Escribe un programa que divida un arreglo grande (por ejemplo, de 1 a 1,000) entre dos hilos. Cada hilo debe calcular la suma de la mitad del arreglo. Cuando ambos hilos terminen, muestra la suma total combinando los resultados parciales.

**Puntos clave a enseñar**:

* Implementar Runnable.
* Coordinar tareas paralelas.
* Compartir resultados entre hilos.

**Requerimientos:**

* Implementa dos clases diferentes para los hilos.
* Utiliza start() para iniciar ambos hilos.
* Asegúrate de que los números pares e impares se impriman de manera concurrente.

**Ejercicio 2 - Simulación de banco con sincronización**

Simula un sistema bancario con dos cuentas y un saldo inicial de $1,000 cada una. Crea varios hilos que simulen transferencias aleatorias entre estas cuentas. Asegúrate de que el saldo total de las cuentas permanezca constante durante todas las transferencias. Usa sincronización para evitar condiciones de carrera.

Puntos clave a enseñar:

* Bloquear métodos con synchronized.
* Uso correcto de objetos compartidos.
* Conciencia de condiciones de carrera y sincronización.

**Ejercicio 3 - Gestión de Inventarios con Múltiples Hilos**

Implementa un sistema que simule la gestión de un inventario en un almacén.

1. Crea un inventario con un número inicial de 100 unidades de un producto.
2. Crea tres tipos de hilos:
   * **Proveedor**: Agrega unidades al inventario.
   * **Cliente**: Retira unidades del inventario.
   * **Auditor**: Consulta el estado actual del inventario.
3. Asegúrate de que el inventario no permita cantidades negativas ni exceda un máximo de 200 unidades.
4. Los hilos deben ejecutarse simultáneamente durante un tiempo definido (por ejemplo, 10 segundos).

**Requisitos**:

* Usa una clase Inventario como recurso compartido con métodos sincronizados para agregar, retirar y consultar el estado del inventario.
* Asegura que las operaciones no excedan los límites definidos (mínimo 0 y máximo 200).
* El programa debe finalizar después de un tiempo determinado, mostrando el estado final del inventario.