**Tarea: Programación Multihilo en Java**

**Ejercicio 1: Sincronización de hilos**

* **Objetivo:** Evitar condiciones de carrera utilizando mecanismos de sincronización.
* **Instrucciones:** Crea un programa que utilice un recurso compartido (como una variable) y sincroniza el acceso a este recurso.
* **Código de ejemplo:**

public class SincronizacionHilos {

private static int contador = 0;

private static final Object lock = new Object();

public static void main(String[] args) throws InterruptedException {

Thread[] hilos = new Thread[10];

for (int i = 0; i < 10; i++) {

hilos[i] = new Thread(() -> {

for (int j = 0; j < 1000; j++) {

synchronized (lock) {

contador++;

}

}

});

hilos[i].start();

}

for (Thread hilo : hilos) {

hilo.join();

}

System.out.println("Valor final del contador: " + contador);

}

}

**Ejercicio 2: Prioridades de los hilos**

* **Objetivo:** Observar cómo se ejecutan los hilos con diferentes prioridades.
* **Instrucciones:** Crea un programa que asigne diferentes prioridades a los hilos.
* **Código de ejemplo:**

public class PrioridadesHilos {

public static void main(String[] args) {

Thread hiloPrioritario = new Thread(() -> {

for (int i = 0; i < 5; i++) {

System.out.println("Tarea prioritaria ejecutándose");

try {

Thread.sleep(500);

} catch (InterruptedException e) {

Thread.currentThread().interrupt();

}

}

});

Thread hiloNoPrioritario = new Thread(() -> {

for (int i = 0; i < 5; i++) {

System.out.println("Tarea no prioritaria ejecutándose");

try {

Thread.sleep(1000);

} catch (InterruptedException e) {

Thread.currentThread().interrupt();

}

}

});

hiloPrioritario.setPriority(Thread.MAX\_PRIORITY);

hiloNoPrioritario.setPriority(Thread.MIN\_PRIORITY);

hiloPrioritario.start();

hiloNoPrioritario.start();

}

}

**Ejercicio 3: Programación de aplicaciones multihilo**

* **Objetivo:** Realizar una tarea compleja utilizando múltiples hilos.
* **Instrucciones:** Implementa una aplicación que calcule la suma de una lista de números en paralelo.
* **Código de ejemplo:**

public class SumaMultihilo {

public static void main(String[] args) throws InterruptedException {

int[] numeros = new int[10000];

for (int i = 0; i < numeros.length; i++) {

numeros[i] = i + 1;

}

int[] resultado = new int[2];

int mitad = numeros.length / 2;

Thread hilo1 = new Thread(() -> {

resultado[0] = sumaParcial(numeros, 0, mitad);

});

Thread hilo2 = new Thread(() -> {

resultado[1] = sumaParcial(numeros, mitad, numeros.length);

});

hilo1.start();

hilo2.start();

hilo1.join();

hilo2.join();

int sumaTotal = resultado[0] + resultado[1];

System.out.println("Suma total: " + sumaTotal);

}

private static int sumaParcial(int[] numeros, int inicio, int fin) {

int suma = 0;

for (int i = inicio; i < fin; i++) {

suma += numeros[i];

}

return suma;

}

}

**Mejora:**

* **Objetivo:** Optimizar y mejorar los programas.
* **Instrucciones:** Implementa una mejora en cada ejercicio, como el uso de barreras para la sincronización, ajustar las prioridades de los hilos según el sistema operativo, o utilizar colas de tareas para distribuir el trabajo.

Espero que estos ejercicios sean útiles para el aprendizaje práctico. ¿Hay algo más en lo que pueda ayudarte?