Shared Memory

Shared memory allows threads to communicate by reading and writing to shared variables.

Synchronization is crucial to avoid race conditions where multiple threads modify shared data concurrently.

*public class* SharedMemory {  
 *private int* counter = 0;  
  
 *public synchronized void* increment() {  
 counter++;  
 }  
  
 *public synchronized int* getCounter() {  
 *return* counter;  
 }  
  
 *public static void* main(String[] args) *throws* InterruptedException {  
 SharedMemory sharedMemory = *new* SharedMemory();  
 Thread[] threads = *new* Thread[10];  
  
 *for* (*int* i = 0; i < 10; i++) {  
 threads[i] = *new* Thread(() -> {  
 *for* (*int* j = 0; j < 100; j++) {  
 sharedMemory.increment();  
 System.*out*.println( Thread.*currentThread*().getName() + " Counter value: " + sharedMemory.getCounter() );  
 }  
 });  
 threads[i].start();  
 }  
  
 *for* (Thread thread : threads) {  
 thread.join();  
 }  
  
 System.*out*.println("Final counter value: " + sharedMemory.getCounter()); *// Should print 1000* }  
}

Thread-1 Counter value: 2

Thread-1 Counter value: 11

Thread-4 Counter value: 5

Thread-8 Counter value: 9