**03. synchronizing shared resources**

public class counter {

private int count = 0;

// Synchronized method to ensure thread-safe access to the counter

public synchronized void increment() {

count++;}

public int getCount() {

return count;}

}

public class SynchronizedExample extends Thread {

private counter counter;

public SynchronizedExample(counter counter) {

this.counter = counter;}

@Override

public void run() {

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

counter.increment(); }}

}

counter counter = new counter();

// Create and start multiple threads

Thread thread1 = new SynchronizedExample(counter);

Thread thread2 = new SynchronizedExample(counter);

thread1.start();

thread2.start();

// Wait for threads to finish

try {

thread1.join();

} catch (InterruptedException ex) {

Logger.getLogger(Javathread.class.getName()).log(Level.SEVERE, null, ex);

}

thread2.join();

System.out.println("Final counter value: " + counter.getCount());

}

}

Output:

