// Implementation of classical problems ( reader writer using threads & semaphore. (reader writer, producer consumer, dining philosopher )

package lab7;

import java.util.concurrent.Semaphore;

class ReaderWriter {

private int readCount = 0; // Count of active readers

private final Semaphore mutex; // Mutex to protect readCount

private final Semaphore writeLock; // WriteLock to ensure only one writer

public ReaderWriter() {

mutex = new Semaphore(1); // Binary semaphore for reader count update

writeLock = new Semaphore(1); // Binary semaphore for writer synchronization

}

// Reader class

class Reader implements Runnable {

private final String readerName;

Reader(String name) {

this.readerName = name;

}

@Override

public void run() {

try {

// Entry Section for Reader

mutex.acquire(); // Protecting readCount

readCount++;

if (readCount == 1) {

writeLock.acquire(); // First reader locks the writeLock

}

mutex.release(); // Release mutex after updating readCount

// Critical Section for Reader (Reading)

System.out.println(readerName + " is READING");

// Simulate reading with sleep

Thread.sleep(1000);

// Exit Section for Reader

mutex.acquire();

readCount--;

if (readCount == 0) {

writeLock.release(); // Last reader releases the writeLock

}

mutex.release();

System.out.println(readerName + " has FINISHED READING");

} catch (InterruptedException e) {

System.out.println(e.getMessage());

}

}

}

// Writer class

class Writer implements Runnable {

private final String writerName;

Writer(String name) {

this.writerName = name;

}

@Override

public void run() {

try {

// Entry Section for Writer

writeLock.acquire(); // Writer acquires writeLock

// Critical Section for Writer (Writing)

System.out.println(writerName + " is WRITING");

// Simulate writing with sleep

Thread.sleep(1500);

// Exit Section for Writer

writeLock.release();

System.out.println(writerName + " has FINISHED WRITING");

} catch (InterruptedException e) {

System.out.println(e.getMessage());

}

}

}

public static void main(String[] args) {

ReaderWriter rw = new ReaderWriter();

// Creating reader and writer threads

Thread reader1 = new Thread(rw.new Reader("Reader 1"));

Thread reader2 = new Thread(rw.new Reader("Reader 2"));

Thread writer1 = new Thread(rw.new Writer("Writer 1"));

Thread writer2 = new Thread(rw.new Writer("Writer 2"));

// Start threads

reader1.start();

writer1.start();

reader2.start();

writer2.start();

}

}