Name: _____

1. Explain how you modified `SynchronizedThreads` in task 2.

> **Solution:** Add synchronized to Counter.increment method.

2. Briefly describe (1-2 sentences) an alternate implementation for task 2.

> **Solution:** Put synchronized block on counter object inside Incrementer.

3. Explain the deadlock that happens in `Deadlock` in terms of locks and threads waiting on locks. The two threads in this case are alphonse and gaston. Your explanations can be short (2-3 sentences) but must be precise!

   (a) What objects are being locked?

   Solution: Friend objects Alphonse and Gaston.

   (b) Who has what locks?

   Solution: Alphonse has lock on himself. Gaston has lock on himself.

   (c) How does deadlock occur?

   Solution: Alphonse requests lock on Gaston and vice versa.

   (d) Return to `SynchronizedThreads`, can deadlock occur here? Explain why or why not.

   Solution: No, there is only one shared object.

4. Explain why `acquireLock` uses a synchronized statement inside the body of the method. In other words, why not just make the `acquireLock` method synchronized, just like `releaseLock`? Will this work? Why or why not?

Solution: This will create deadlock. Consider the following: thread 1 acquires the lock, set inUse to true, and go do some work. Meanwhile, thread 2 will call acquire-Lock on LockManager object and busy wait until thread 1 is finished. The problem is that thread 1 can never "finish" because when it calls releaseLock, it will wait on thread 2 until thread 2 releases the lock on the LockManager object.