//EXERCICIO 1:

```
package LP3A5;

public class THREADS {
	private static int num = 0;

	public static void main(String[] args) {
	Runnable incrementTask = () -> {
	for (int i = 0; i < 800; i++) {
		synchronized (THREADS.class) {
		num++;
		}
	}
	};

	Thread thread1 = new Thread(incrementTask);
	Thread thread2 = new Thread(incrementTask);

	thread1.start();
	thread2.start();

	try {
	thread1.join();
	thread2.join();
	} catch (InterruptedException e) {
	System.out.println(e);
	}

	System.out.println("Final sharedVariable value: " + num);
	}
}
```

//EXERCICIO 2:

```
package LP3A5;

import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;

public class DeadlockSimulation {
	private final Lock lock1 = new ReentrantLock();
	private final Lock lock2 = new ReentrantLock();
	private final Lock lock3 = new ReentrantLock();
```

```
public void method1() {
boolean lock1Acquired = false;
boolean lock2Acquired = false;
while (true) {
try {
        lock1Acquired = lock1.tryLock();
        if (lock1Acquired) {
        System.out.println("Method 1 acquired lock 1");
        lock2Acquired = lock2.tryLock();
        if (lock2Acquired) {
        System.out.println("Method 1 acquired lock 2");
        break;
        }
        }
} finally {
        if (!lock2Acquired && lock1Acquired) {
        lock1.unlock();
        System.out.println("Method 1 released lock 1");
        }
}
try {
        Thread.sleep(100);
} catch (InterruptedException e) {
        e.printStackTrace();
}
}
lock2.unlock();
System.out.println("Method 1 released lock 2");
lock1.unlock();
System.out.println("Method 1 released lock 1");
}

public void method2() {
boolean lock1Acquired = false;
boolean lock3Acquired = false;
while (true) {
try {
        lock1Acquired = lock1.tryLock();
        if (lock1Acquired) {
        System.out.println("Method 2 acquired lock 1");
        lock3Acquired = lock3.tryLock();
        if (lock3Acquired) {
        System.out.println("Method 2 acquired lock 3");
        break;
        }
        }
} finally {
```

```
            if (!lock3Acquired && lock1Acquired) {
            lock1.unlock();
            System.out.println("Method 2 released lock 1");
            }
}
try {
            Thread.sleep(100);
} catch (InterruptedException e) {
            e.printStackTrace();
}
}
lock3.unlock();
System.out.println("Method 2 released lock 3");
lock1.unlock();
System.out.println("Method 2 released lock 1");
}

public void method3() {
boolean lock2Acquired = false;
boolean lock3Acquired = false;
while (true) {
try {
            lock2Acquired = lock2.tryLock();
            if (lock2Acquired) {
            System.out.println("Method 3 acquired lock 2");
            lock3Acquired = lock3.tryLock();
            if (lock3Acquired) {
            System.out.println("Method 3 acquired lock 3");
            break;
            }
            }
} finally {
            if (!lock3Acquired && lock2Acquired) {
            lock2.unlock();
            System.out.println("Method 3 released lock 2");
            }
}
try {
            Thread.sleep(100);
} catch (InterruptedException e) {
            e.printStackTrace();
}
}
lock3.unlock();
System.out.println("Method 3 released lock 3");
lock2.unlock();
System.out.println("Method 3 released lock 2");
}
```

```
        public static void main(String[] args) {
         DeadlockSimulation simulation = new DeadlockSimulation();
        Thread thread1 = new Thread(simulation::method1);
        Thread thread2 = new Thread(simulation::method2);
        Thread thread3 = new Thread(simulation::method3);

        thread1.start();
        thread2.start();
        thread3.start();
        }
}
```