# Exercise 5

## 2.

(1) Interference occurs when two operations, running in different threads, but acting on the same data, interleave.

(2) Yes. Thread `t1` and `t2` share the same variable `counter`, which can cause interference.

(3) Making methods synchronized has two effects:

- It is not possible for two invocations of synchronized methods on the same object to interleave. When one thread is executing a synchronized method for an object, all other threads that invoke synchronized methods for the same object block (suspend execution) until the first thread is done with the object.
- When a synchronized method exits, it automatically establishes a happens-before relationship with any subsequent invocation of a synchronized method for the same object. This guarantees that changes to the state of the object are visible to all threads.

(4)

Simply add `synchronized` keyword to increment and decrement method:

```
public synchronized void increment()
public synchronized void decrement()
```

(5)

The most common difference is:

When you extend Thread class, you can't extend any other class which you require. (As you know, Java does not allow inheriting more than one class). When you implement Runnable, you can save a space for your class to extend any other class in future or now.

However, the significant difference is:

When you extends Thread class, each of your thread creates unique object and associate with it. When you implements Runnable, it shares the same object to multiple threads.

*(via http://manikandanmv.wordpress.com/tag/extends-thread-vs-implements-runnable/)*

## 3.

(1) ACBACBACBACBACBACBACBACBACBACB

(2)
Problem:

We cannot ensure that the thread starting order is A->B->C.

Solution:

```
new Thread(pa).start();
Thread.sleep(10);
new Thread(pb).start();
Thread.sleep(10);
new Thread(pc).start();
```