2001 Paper 4 Question 3

Further Java

(*a*) Describe how mutual-exclusion locks provided by the `synchronized` keyword can be used to control access to shared data structures. In particular you should be clear about the behaviour of concurrent invocations of different synchronized methods on the same object, or of the same synchronized method on different objects. [6 marks]

(*b*) Consider the following class definition:

```
class Example implements Runnable {
  public static Object o = new Object();
  int count = 0;

  public void run() {
    while (true) {
      synchronized (o) {
        count ++;
      }
    }
  }
}
```

Show how to start two threads, each executing this `run` method. [2 marks]

(*c*) When this program is executed, only one of the `count` fields is found to increment, even though threads are scheduled preemptively. Why might this be? [2 marks]

(*d*) Define a new class `FairLock`. Each instance should support two methods, `lock` and `unlock`, which acquire and release a mutual exclusion lock such that calls to `unlock` never block the caller, but will allow the longest-waiting blocked thread to acquire the lock. The lock should be *recursive*, meaning that the thread holding the lock may make multiple calls to `lock` without blocking. The lock is released only when a matched number of `unlock` operations have been made.

You may wish to make use of the fact the `Thread.currentThread()` returns the instance of `Thread` that is currently executing. [10 marks]