

Quiz 5, Spring 2019

```
class R1 implements Runnable {
    public static Object obj = new Object( );

    public synchronized void f1( ) {
        obj = new Object( );
    }

    public synchronized void run( ) {
        f1( );
    }
}
```

```
class Main {
```

```
    public static void main(String args[]) throws Exception {
```

```
        Thread t1 = new Thread(new R1( ));
        Thread t2 = new Thread(new R1( ));
        t1.run( ); // S1
        t2.run( ); // S2
        t1.start( ); // S3
        t2.start( ); // S4
        Thread t3 = new Thread(new R2( ));
        Thread t4 = new Thread(new R2( ));
        t3.run( ); // S5 note corrected thread reference name from t1 to t3
        t4.run( ); // S6 note corrected thread reference name from t2 to t4
        t3.start( ); // S7 note corrected thread reference name from t1 to t3
        t4.start( ); // S8 note corrected thread reference name from t2 to t4
    }
}
```

```
class R2 implements Runnable {
    public static Object obj = new Object( );

    public void f2( ) {
        synchronized(obj) {
            obj = new Object( );
        }
    }

    public synchronized void run( ) {
        f2( );
    }
}
```

Q1: is there a race on *obj* on the calls in S1 and S2? **No.** Calls to *run* do not spawn a new thread and execute on after the other. So regardless of what the actual *run* method does, there is not a race.

Q2: is there a race on *obj* on the calls in S3 and S4? **Yes.** The call to *start* spawn new threads, and thus the calls can execute at the same time. The synchronized method *f1* synchronizes on the object referenced by *t1* in the first call to *run*, and the object referenced by *t2* in the second call to *run*. Thus both *f1*s can execute at the same time and access the shared static variable *obj*, leading to a race.

Q3: is there a race on *obj* on the calls in S5 and S6? **Same answer and reason as Q1.**

Q4: is there a race on *obj* on the calls in S7 and S8? **No.** As in Q2, the call to *start* spawn new threads, and thus the calls can execute at the same time. Unlike in Q2, the calls to *f2* both synchronize on the object referenced by *obj*, and so only one update of *obj* can happen at a time, and no race exists.