

Advanced Aspects of Object-Oriented Programming (SS 2009)

Practice Sheet 10

Date of Issue: 30.06.09
Deadline: 06.07.09
(until 10 a.m. as PDF via E-Mail)

Exercise 1 Synchronization

a) As mentioned in the lecture, to simplify notation, one can write:

```
synchronized void mm() Block
```

instead of

```
void mm(){ synchronized( this ) Block }
```

How does this apply to synchronized *static* methods?

b) Explain *Reentrant Synchronization*.

c) Explain *Deadlock*, *Starvation* and *Livelock*.

d) Can the following Counter class be effectively shared among threads? If not, generate a test case to demonstrate the issue and fix the implementation.

```
class Counter {  
    private int c = 0;  
  
    public void increment() {  
        c++;  
    }  
  
    public void decrement() {  
        c--;  
    }  
  
    public int value() {  
        return c;  
    }  
}
```

Exercise 2 Programming with Threads

Realize the following program using threads. The program should create and keep alive 503 threads, explicitly or implicitly linked in a ring, and pass a token between one thread and the next thread at least N times.

The program should:

- create 503 linked threads (named 1 to 503)
- thread 503 should be linked to thread 1, forming an unbroken ring
- pass a token to thread 1
- pass the token from thread to thread N times
- print the name of the last thread (1 to 503) to take the token

Exercise 3 Sempahores

Semaphores are often used to restrict the number of threads that can access some (physical or logical) resource. Implement a semaphore according to the documentation at <http://java.sun.com/j2se/1.5.0/docs/api/java/util/concurrent/Semaphore.html>. You do not need to implement fairness. The only methods you need to implement are the following ones:

```
Semaphore(int permits) // constructor
void acquire() throws InterruptedException
boolean tryAcquire()
void release()
int availablePermits()
int drainPermits()
```

To implement the semaphore, you may however use other classes from the `java.util.concurrent` package. Test your implementation.