Prof. Dr. A. Poetzsch-Heffter
Dipl.-Inform. Kathrin Geilmann

University of Kaiserslautern
Department of Computer Science
Software Technology Group

Advanced Aspects of Object-Oriented Programming (SS 2010)

Practice Sheet 10

Date of Issue: 17.06.10
Deadline: 23.06.10
(until 10 a.m. as PDF via E-Mail)

Exercise 1 The Java Collections Framework and Thread Safety

a) Explain the term “thread-safe”.

b) Inform yourself about the synchronization wrappers for the collection classes of `java.util`. How do they work and which guarantees are given?

c) Are there problems with the following code, if it is executed in a multi-threaded program context? If yes, explain and fix the issues.

```
Map<String,String> m = Collections.synchronizedMap(new HashMap<String,String>(...));
m.put("a", "b");
if !(m.containsKey("a")) {
      m.put("c", "d");
}
```

d) Java 5 introduced the package `java.util.concurrent`, that contains classes to support common tasks in multi-threaded programs. Consider the class `ConcurrentHashMap`. What is the difference between a `ConcurrentHashMap` and a synchronized Map as returned by the synchronization wrapper? Which are the (dis-)advantages of using a `ConcurrentHashMap` over a synchronized Map, and vice versa?

e) Are there problems with the following code, if it is executed in a multi-threaded program context? If yes, explain and fix the issues.

```
ConcurrentHashMap<String,String> m = new ConcurrentHashMap<String,String>(...);
m.put("a", "b");
if !(m.containsKey("a")) {
    m.put("c", "d");
}
```

Exercise 2 Synchronization and Shared Objects

Consider the following code. Which variables are shared between the threads? Is this code thread-safe? Remove as much synchronization actions as possible.

```
class C implements Runnable {
    void run() {
        Object[] a = {"a", "b"};
        List<Object> m = Collections.synchronizedList(new LinkedList<Object>(Arrays.asList(a)));
        while (true) {
            synchronized(m) {
                Object o = generateObject(m);
                if !(m.contains(o)) {
                    m.add(o);
                }
            }
        }
    }

    synchronized Object generateObject (List m) {
        return new Object();
    }
}

class D implements Runnable {
    private List m;

    void run() {
        Object[] a = {"a", "b"};
        List<Object> m = Collections.synchronizedList(new LinkedList<Object>(Arrays.asList(a)));
        while (true) {
            synchronized(m) {
                Object o = generateObject(m);
```

```
                if !(m.contains(o)) {
                    m.add(o);
                }
            }
        }

        synchronized Object generateObject (List m) {
            if (this.m == null) this.m = m;
            synchronized (this.m) {
                Object o = .... // calculate o, based on the parameter m
                if (!this.m.contains(o)) {
                    this.m.add(o);
                }
            }
            return o;
        }
    }

    public static void main(String ... a) {
        C c = new C();
        Thread t1 = new Thread(c);
        Thread t2 = new Thread(c);
        t1.start();
        t2.start();
        D d = new D();
        Thread t3 = new Thread(d);
        Thread t4 = new Thread(d);
        t3.start();
        t4.start();
    }
```