# Übungsblatt 10

## 4 Aufgaben, 20 Punkte

Objektorientierte Modellierung und Programmierung (inf031) Sommersemester 2020
Carl von Ossietzky Universität Oldenburg, Fakultät II, Department für Informatik

Dr. C. Schönberg

---

**Ausgabe:** 2020-06-23 14:00 **Abgabe:** 2020-06-30 12:00

**Aufgabe 1:** *Threads: Suchen im Array* **(4 Punkte)**

Schreiben Sie ein Programm, das ein gegebenes **`int`**-Array daraufhin untersucht, ob im Array ein bestimmter **`int`**-Wert enthalten ist. Die Arbeit sollen sich dabei zwei Threads teilen: Ein Thread durchsucht den linken Teil und ein zweiter den rechten Teil. Beide Threads sollen die Ergebnisse dem Benutzer kundtun.

**Beispiel:**

- Array { 2, 7, 3, 9, 23 }
- Eingabe: 7
- `Thread 1: found`
- `Thread 2: not found`

Erweitern Sie anschließend Ihr Programm so, dass es die Ergebnisse der beiden Threads zusammenfasst und ausgibt:
`Found: true`

**Hinweis:** Die Bezeichnungen Thread 1 bzw. Thread 2 können Sie über die Methode `Thread.getName()` erfahren.

**Aufgabe 2:** *Threads: Sortierung* **(5 Punkte)**

Gegeben ist eine Klasse `QuickSort` mit einer statischen Methode `sort`, die ein als Parameter übergebenes Array in aufsteigender Reihenfolge gemäß dem Quicksort-Algorithmus sortiert. In der Hilfsmethode `quickSort` werden nacheinander der Bereich links vom Pivot-Element und der Bereich rechts vom Pivot-Element durch rekursiven Aufruf der Methode sortiert. Prinzipiell sind diese beiden Aktionen aber unabhängig voneinander und können auch parallel erfolgen. Parallelisieren Sie entsprechend den Quicksort-Agorithmus durch Verwendung adäquater Thread-Mechanismen in der Klasse `QuickSortThreaded`, die von `QuickSort` erbt.

**Aufgabe 3:** *Bildverarbeitung mit Threads* **(6 Punkte)**

Gegeben sei eine Java-Klasse `ImageFilter` zur Anwendung von grafischen Filtern auf Bilder. Diese Klasse nutzt die Exception-Klasse `InvalidFilterException`. Die Klasse `ImageFilterThreaded` erbt von `ImageFilter`, stellt bisher aber noch keine eigene Funktionalität zur Verfügung. Die Klasse `ImageFilterTest` wendet den Bild-Filter auf ein Beispielbild an und vergleicht die Laufzeit der zwei Varianten des `ImageFilter`s.

Überschreiben Sie in der Java-Klasse `ImageFilterThreaded` die Methode `filterMatrix`, so dass sie die gleiche Funktionalität hat wie in der Klasse `ImageFilter`, aber die Anwendung des Filters parallelisiert und auf vier Arbeits-Threads aufteilt. Jeder dieser Threads soll eine der vier Ecken des Bildes bearbeiten.

Überlegen Sie sich, wie Sie den Threads die nötigen Informationen zur Verfügung stellen können, und wie sie das Ergebnis zusammensetzen können. Die Laufzeit der parallelisierten Version sollte kürzer sein als die der sequentiellen Version.

**Aufgabe 4:** *Reihenfolge von Threads* **(5 Punkte)**

Java besitzt einen präemptiven Scheduler, d.h. u.a. sind die Aktivierungsreihenfolgen von Threads nicht definiert. Erarbeiten Sie ein Konzept, mit dem sie erreichen, dass $n$ Threads immer in einer definierten Reihenfolge arbeiten.

Machen Sie konkret folgendes: Leiten Sie eine Klasse `NameOutput` von der Klasse `Thread` ab. Erzeugen Sie in der `main`-Methode $n$ (hier $n = 3$) Thread-Objekte der Klasse `NameOutput` und starten Sie über die `start`-Methode den dazu gehörenden Thread der drei Objekte. In der `run`-Methode sollen die `NameOutput`-Threads jeweils ihren Namen ausgeben (Methode `Thread.getName`), sie sollen das jedoch immer in derselben Reihenfolge tun, müssen sich also entsprechend untereinander synchronisieren:

```
Thread-0
Thread-1
Thread-2
Thread-0
Thread-1
Thread-2
Thread-0
Thread-1
...
```