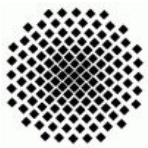

Universität Stuttgart — Wintersemester 2007/2008
Fakultät 5, Institut FMI — Claus, Riexinger
Einführung in die Informatik III — 09.12.2007

Aufgabenblatt 8 — **Abgabe: 16.12.2007, 24.00 Uhr**

---

1. Kontoführung (mittel) (12 Punkte)

Schreiben Sie ein Programm, das Synchronized-Methoden verwendet.

- Implementieren Sie dazu zwei Klassen **KontoBenutzer** und **Konto**, wobei **KontoBenutzer** von Thread abgeleitet wird. Ein Konto soll die übliche Funktionalität bieten (abheben, einzahlen, Kontostandsabfrage . . .). Dabei darf auf einem Konto niemals ein negativer Betrag entstehen, auch wenn mehrere Konto-Benutzer gleichzeitig abheben wollen (der aktuelle Betrag wird bei jeder Aktion auf der Konsole ausgegeben)! (7 Punkte)
- Implementieren Sie nun zusätzlich die Methode Überweisung. (5 Punkte)

Geben Sie ein Programm ab, das drei Konten und drei Konto-Benutzer verwendet und mindestens 10 Aktionen ausführt.

2. Semaphore (mittel) (10 Punkte)
Eine andere Möglichkeit einen wechselseitigen Ausschluss zu realisieren, bieten Semaphore (siehe Skript Info II, Def. 13.2.8 - auf der Übungshomepage von Info III).
In Java gibt es bereits die Klasse Semaphore (benötigt "import java.util.concurrent.*;").

Ein Beispiel:

```
class MeinThread extends Thread {
    private Semaphore sem; /* Semaphor-Objekt */
    private String name;

    MeinThread(String name, Semaphore sem) {
        this.name = name;
        this.sem = sem;
    }

    public void run() {
        try {
            sem.acquire();
            System.out.println(name+" -> betritt kritischen Bereich");
            sleep(1800); /* Kritischer Bereich */
            System.out.println(name+" <- verl¨asst kritischen Bereich");
            sem.release();
        } catch (Exception e) { }
    }
}
```

```
class Beispiel {
    public static void main(String[] args) {
        Semaphore s = new Semaphore(1);
        MeinThread t1 = new MeinThread("Thread_1",s);
        MeinThread t2 = new MeinThread("Thread_2",s);
        t1.start();
        t2.start();
    }
}
```

Schreiben Sie ein Java-Programm, das eine Auto-Fabrik simuliert. Implementieren Sie dazu eine abstrakte Klasse **KFZModellAuftrag** (wieder von Thread abgeleitet) und leiten Sie davon wiederum verschiedene Modelle ab. Leiten Sie außerdem von der Klasse **Semaphore** mehrere Klassen ab, um die benötigten Maschinen (Arbeitsschritte für die Produktion von **KFZModell**) zu simulieren. Die Idee ist nun, dass in der Parameterliste des Konstruktors des KFZ-Modells Instanzen der benötigten Maschinen übergeben werden. Auf diese Weise kann auf einer Maschine (abgeleitet von Semaphore) ein Produktionsschritt eines Modells abgearbeitet werden (sleep(...) einfügen). Geben Sie ein Programm ab, das fünf Aufträge mit mindestens drei KFZ-Modellen auf mindestens vier Maschinen produziert und dabei ausführliche Kommentare auf der Konsole ausgibt.