

**Aufgabe 1** (5 Punkte)

In 1.2.2 waren die Klammern `< >` eingeführt worden, damit die notwendige Unteilbarkeit in Fällen wie diesem

```
CO ... < seats = seats-1; > ...  
|| ... < seats = seats-2; > ...  
OC
```

garantiert werden kann.

Es wird vorgeschlagen, eine gemeinsame Variable `boolean busy` (initialisiert mit `false`) einzuführen und die Klammern durch die Anweisungen

```
while(busy); busy = true; // instead of <  
bzw. busy = false; // instead of >
```

zu ersetzen. Es zeigt sich leider, dass das keine gute Idee ist. Zeigen Sie mit Zeitschnitten, was schiefgehen kann!

**Aufgabe 2** (5 Punkte)

Wir setzen eine Java-Erweiterung voraus, die über die Gabelungsanweisung **FORK** aus 2.1.2 verfügt. Gegeben sei die Klassenmethode

```
static void nFork(int n, Procedure<Integer> p) {  
    PROCESS[] pr = new PROCESS[n+1];  
    for(int i=1; i<=n; i++) pr[i] = FORK p.call(i);  
    for(int i=1; i<=n; i++) JOIN pr[i];  
}
```

mit **interface** `Procedure<T> {void call(T x);}` .

Gesucht ist eine andere Version von `nFork`, die statt von **FORK/JOIN** von der Nebenläufigkeitsanweisung **CO/OC** Gebrauch macht (aber den gleichen Effekt hat!).

*Hinweis:* Rekursion bietet sich an.

### Aufgabe 3 (5 Punkte)

In 2.1.4 war ein hypothetisches Nebenläufigkeitskonzept für Java skizziert worden: in einer Klassenvereinbarung kann anstelle des Schlüsselworts **class** auch das Schlüsselwort **PROCESS** verwendet werden. Für die Semantik eines solchen **PROCESS** sind verschiedene Präzisierungen möglich. Entscheiden Sie sich für eine und verfassen Sie eine genaue Beschreibung der Semantik, so wie sie in einem „offiziellen“ Handbuch der Sprache beschrieben würde. (Qualitätsanspruch!)

(Achtung: Konstruktoren und Initialisierer!)

### Aufgabe 4 (12 Punkte)

a) (8 P.) Entwickeln Sie ein Java-Programm, das eine primitive Uhr darstellt! Es soll permanent die aktuelle Zeit anzeigen und zudem bei Drücken der Zeilenwechsel-Taste die Darstellung ändern. Ein einfaches Beispiel:

```
% java Clock
14:30                wird automatisch aktualisiert
                     RETURN bewirkt
2.30pm
                     RETURN bewirkt wieder
14:30
```

Diese Funktionalität ist ziemlich simpel; es bleibt Ihnen freigestellt, etwas leistungsfähigeres und ansehnlicheres zu entwickeln.

b) (4 P.) Verfassen Sie eine gut lesbare *Gebrauchsanweisung* für Ihre Uhr!

*Hinweise:*

- Die Verwendung von `sleep` empfiehlt sich.
- Für die Eingabe empfiehlt sich `System.in.read()`.
- Die aktuelle Uhrzeit erhält man z.B. durch Erzeugung eines Objekts der Klasse `java.util.GregorianCalendar`.