Lehrstuhl für Programmiersprachen und Übersetzerkonstruktion
CAU Kiel, Informatik und Prakt. Mathematik
Frank Huch

24118 Kiel
Christian-Albrechts-Platz 4
Tel.: 0431 / 880-7277

4. Übung „Nebenläufige und verteilte Programmierung“

Abgabe in der Vorlesung oder bis zum 22. November in Raum 704

Aufgabe 10 **10 Punkte**

In Aufgabe 8 der letzten Übung haben Sie die Klasse `MVar` definiert. Die Verwendung von MVars zum Message Passing zwischen Threads hat zwei Nachteile:

- Schreiber blockieren bei einer gefüllten MVar
- Schreiboperationen werden nicht unbedingt in der gewünschten Reihenfolge ausgeführt (es ist sogar möglich, daß ein Thread immer wieder von anderen Threads überholt wird und nie schreibt)

In der letzten Übung hatten wir diese Probleme durch zusätzliche Threads und einen Zeitstempel gelöst. Ein alternativer Ansatz sind unbeschränkte Puffer, im folgenden Channel genannt.
Channel können mit Hilfe von MVars definiert werden. Ein Channel ist ein Verkettung von MVars, wobei die beiden ausgezeichneten Enden (`MVar read, write`) des Channels gespeichert werden. Ein Eintrag des Channels besteht aus einem Wert (vom Typ `Object`) und einer MVar, welche als Zeiger auf das nächste Element des Channels verwendet wird. Man benötigt keine separate Markierung des Channelendes, da die entsprechende MVar in der Kette hier einfach leer ist und die MVar `write` auf dieses Ende zweigt.

a) Implementieren Sie eine Klasse `Channel`, welche folgende Methoden zur Verfügung stellt:

   - `write` schreibt einen Wert in den Channel
   - `read` liest einen Wert aus dem Channel; suspendiert, falls der Channel leer ist
   - `isEmpty` liefert `true`, falls der Channel leer ist
   - `unGetChan` fügt einen Wert vorne in den Channel ein

   Überlegen Sie bei Ihrer Implementierung genau, welche Methoden synchronisiert ablaufen müssen. Zuviel Synchronisation führt schnell zu einem Deadlock.

b) Modifizieren Sie die Implementierung des Erzeuger-Verbraucher-Problems aus Aufgabe 8b, so daß die Kommunikation über einen Channel erfolgt.

c) Modifizieren Sie die Implementierung der Klasse `Channel`, so daß keine der Methoden mehr als synchronized deklariert werden muß. Die Idee ist die Verwendung weiterer MVars zur Synchronisation.

   Vergleichen Sie die beiden Implementierungen des Channels bzgl. Ihres Synchronisationsverhaltens. Welchen Methoden bzw. Anweisung laufen in welcher Version synchronisiert ab.

d) Vergleichen Sie die definierten Channel mit den in der Vorlesung vorgestellten Pipes. Diskutieren Sie Vor- und Nachteile. Entwickeln Sie ein Testprogramm, mit welchem Sie den Durchsatz vergleichen können und überlegen Sie sich sinnvolle Meßwerte. Ist der Vergleich des Durchsatzes in Ihrem Ansatz fair?

Aufgabe 11 **6 Punkte**

a) Erweitern Sie den Zähler aus Aufgabe 9 um einen Copy-Button. Wenn dieser Button gedrückt wird, wird eine Kopie des Zählers angelegt, welche sich zunächst identisch wie das Original verhält. Alle weiteren Aktionen des kopierten bzw. des Originalzählers haben allerdings keine Auswirkungen auf den jeweils anderen Zähler.

b) Erweitern Sie den Zähler um einen Clone-Button. Nach drücken des Clone-Buttons geschieht das gleiche, wie beim Copy-Button. Allerdings verändern alle Aktionen des Klons bzw. des Originals auch das Original bzw. den Klon. Ausnahmen sollen der Close-Button und der Copy-Button sein: Nur der Klon bzw. das Original wird beendet bzw. kopiert.

   Beachten Sie, daß die Zähler auch mehrfach (und auch rekursiv) geklont werden können. Alle Klone sollen sich identisch verhalten.

Aufgabe 12 **4 Punkte**

Entwickeln Sie ein Programm, mit dem Sie testen können, ob die in der Vorlesung vorgestellten Möglichkeiten zum Debuggen von Threads, das Scheduling verändern. Überprüfen Sie für jede Debug-Methode, ob sie das Scheduling verändert.