



Vorbereitung

- Die Dateien zum heutigen Praktikumstermin laden Sie von der Vorlesungsseite herunter:
`wget http://hm.hgesser.de/bs-ss2009/prakt06.tgz`
- Entpacken Sie das Archiv und wechseln Sie in das neue Unterverzeichnis `prakt06`.

12. Monitore in Java (Producer-Consumer-Problem)

Aus dem Quellcode-Archiv entnehmen Sie die Datei `PC.java`, die das Producer-Consumer-Problem in Java löst. Sie können es mit `javac PC.java` kompilieren und anschließend mit `java PC` starten. Das Programm deklariert die Methoden `Get` und `Put` als `synchronized`, aber es arbeitet nicht korrekt.

Eine sehr ausführliche Beschreibung der Thread-Programmierung in Java (die zum Lösen der folgenden Aufgaben nicht zwingend nötig ist) finden Sie im E-Learning-System (Moodle) unter dem Link [Zusatzmaterial: Java Thread Synchronization](#); unter dem Link [Zusatzmaterial: Monitore](#) finden Sie eine allgemeinere Darstellung des Monitor-Konzepts. Wenn Sie noch keinen Account für das Moodle-System haben, ist jetzt die perfekte Gelegenheit, einen anzulegen :)

- a) Das Programm arbeitet fehlerhaft. Beschreiben Sie das Fehlverhalten und nennen Sie die Ursache.
- b) Der Producer wartet *innerhalb* des Monitors, wenn der Puffer voll ist. Ergänzen Sie an geeigneten Stellen die Aufrufe `wait()` und `notify()`, um die Zustandsvariable des Monitors zu verwenden – der Producer soll sich also schlafen legen, wenn der Puffer voll ist (und der Consumer, wenn der Puffer leer ist); beide sollen einander wecken, wenn sich der jeweils relevante Zustand ändert. (Denken Sie dabei an die Lösung des Problems mit Semaphoren.) Beachten Sie bei Ihrer Lösung, dass Sie nicht einfach

```
while(count == ...) { wait(); }
```

schreiben können, sondern

```
while(count == ...) {  
    try { wait(); }  
    catch (InterruptedException e) { }  
    finally { }  
}
```

schreiben müssen und dass `notify()` der letzte Befehl (ggf. vor einem `return`) in der Methode sein muss. Testen Sie das derart veränderte Programm.

- c) Wie in der letzten Übung sollen jetzt mehrere Producer-Threads erzeugt werden. Passen Sie die Klassendefinition von `Producer` so an, dass der Generator `Producer(Buffer b)` ein zweites Argument erhält, das den Startbuchstaben (bisher immer „A“) enthält – dann können Sie die Threads z. B. mit den Buchstaben „A“, „K“ und „U“ beginnen lassen. Korrigieren Sie auch die Klasse `Consumer`, so dass sie genau so viele Zeichen aus dem Buffer liest, wie in der Summe von Ihren Producer-Threads erzeugt werden. Vergessen Sie nicht, eine geeignete Anzahl Producer-Threads zu erzeugen *und* auch zu starten. Überprüfen Sie das Verhalten des neuen Programms. Tipp: Sie erhalten eine übersichtlichere Ausgabe, wenn Sie in den Producer-Threads die Ausgabe deaktivieren und sie im Consumer-Thread auf die Ausgabe des konsumierten Buchstabens (ohne Zeilenumbruch) beschränken. Ergänzen Sie dann am Ende der Methode noch ein `println`-Kommando, damit die Ausgabe nach Programmende sichtbar bleibt (sie wird sonst von der Shell mit dem Prompt überschrieben).



- d) Ersetzen Sie probeweise im Programm aus Aufgabe c) die Aufrufe von `notify()` durch `notifyAll()`. Ändert sich dadurch etwas am Programmverhalten?
- e) Erzeugen Sie eine neue Version des Ausgangsprogramms aus Aufgabe a), die ganz ohne Synchronisation arbeitet. Damit hier Fehler auftreten, müssen Sie Consumer und Producer an geeigneten Stellen mit `sleep()` künstlich aufhalten. Definieren Sie dazu innerhalb der Hauptklasse `PC` die Funktion `randomsleep()` wie folgt:

```
public static void randomsleep () {  
    try {  
        Thread.sleep( (int)Math.floor((Math.random() * 40) + 20));  
    }  
    catch(InterruptedException ie) { ; };  
}
```

Die Funktion `randomsleep()` rufen Sie aus dem Consumer/Producer über `PC.randomsleep()` auf. Die hier angegebenen Werte ($40 * \text{random} + 20$) müssen Sie evtl. anpassen, damit das Fehlverhalten auftritt – das liegt am unterschiedlichen Thread-Scheduling auf verschiedenen PCs. (Sie finden den Funktionsquelltext auch in der Datei `randomsleep.txt`.)

Beschreiben Sie, inwiefern sich das Programm jetzt fehlerhaft verhält.

[ Abgabe via Moodle] Die Lösungen (Programmdateien und Beobachtungen zum jeweiligen Programmverlauf) laden Sie bitte als Zip- oder Tar.gz-Archiv im Moodle unter „Praktikum – Blatt 6“ hoch. Sie erhalten im Moodle-System dann auch die Bewertungen Ihrer Lösungen.

Moodle: <http://hm.hgesser.de/moodle/>