

Vorbereitung

Die Dateien zur Übung laden Sie von der Webseite (<http://fom.hgesser.de/bs-ss2012/prakt04.zip>) herunter. Unter Linux entpacken Sie zip-Archive mit `unzip datei.zip`.

9. 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 in der Shell 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.

Sie können auch Windows oder Mac OS für diese Aufgabe verwenden.

Eine sehr ausführliche Beschreibung der Thread-Programmierung in Java (die zum Lösen der folgenden Aufgaben nicht zwingend nötig ist) finden Sie unter

<http://www.artima.com/insidejvm/ed2/threadsynch.html>; unter

<http://plg.uwaterloo.ca/~usystem/pub/uSystem/MonitorClassification.pdf> gibt es eine allgemeinere Darstellung des Monitor-Konzepts.

- Das Programm arbeitet fehlerhaft. Beschreiben Sie das Fehlverhalten und suchen Sie die Ursache.
- 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.

- Jetzt sollen 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).