TECHNISCHE UNIVERSITÄT MÜNCHEN
FAKULTÄT FÜR INFORMATIK

Lehrstuhl für Sprachen und Beschreibungsstrukturen
Praktikum Grundlagen der Programmierung
Prof. Dr. Helmut Seidl

SS 2008
Aufgabenblatt 10
23. Juni 2008

Abgabe: 01.-03. Juli 2008 beim jeweiligen Tutor

Praktikum Grundlagen der Programmierung

Themen: Threads, Nebenläufige Programmierung

Wichtiger Hinweis: Es wird dringend angeraten, die oben genannten Kapitel im Skript *vor* dem Praktium *selbständig* zu erarbeiten, um Unklarheiten, insbesondere im Bezug auf die Hausaufgaben, rechtzeitig mit dem Tutor abklären zu können.

Aufgabe 46 (Ü) Einfacher Thread

Schreiben Sie ein Programm, das fünf Threads erzeugt. Die Threads sollen mit einem zufälligen Zeitabstand von ein bis drei Sekunden erzeugt und fortlaufend nummeriert (Thread-Nummer) werden. Alle Threads besitzen einen gemeinsamen Zähler. Jeder der Threads soll nun nach einem zufälligen Zeitintervall zwischen einer und drei Sekunden den gemeinsamen Zähler inkrementieren und zusammen mit seiner Thread-Nummer ausgeben. Jeder Thread soll diesen Vorgang solange wiederholen bis der gemeinsame Zähler 42 erreicht und dann terminieren.

Aufgabe 47 (Ü) Verklemmung

a) Implementieren Sie die Klasse `Siedler` als Kindklasse der Klasse `Thread`. Statten Sie Ihre Klasse mit einem Konstruktor aus, der die Übergabe von zwei Monitorobjekten als erwünschte Rohstoffe erlaubt.

b) Erweitern Sie ihre Klasse um eine `run`-Methode, in der Sie zuerst ein Lock auf den ersten Rohstoff beantragen, dann 2 Sekunden warten, und dann ein Lock auf den zweiten Rohstoff beantragen und wieder 2 Sekunden warten, bevor Sie beide Locks wieder aufgeben. Versehen Sie Ihre `run()`-Methode mit ausreichend vielen erläuternden Statusmeldungen.

c) Testen Sie Ihre Klasse `Siedler` mit den zwei Siedlern *Hugo* und *Egon*, sowie den Rohstoffen *Lehm* und *Holz* in einem `main()`-Programm und untersuchen Sie, ob und unter welchen Umständen es zu einer Verklemmung kommen kann.

Aufgabe 48 (H) Parallele Matrixaddition (6 Punkte)

Nutzen Sie das Java Thread-Konzept zur parallelen Addition von Matrizen,indem Sie jedes Feld der Ergebnismatrix in einem separaten Thread berechnen lassen. Die Addition zweier Matrizen ist folgendermaßen definiert:

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix} + \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ b_{m1} & b_{m2} & \dots & b_{mn} \end{pmatrix} = \begin{pmatrix} a_{11}+b_{11} & a_{12}+b_{12} & \dots & a_{1n}+b_{1n} \\ a_{21}+b_{21} & a_{22}+b_{22} & \dots & a_{2n}+b_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1}+b_{m1} & a_{m2}+b_{m2} & \dots & a_{mn}+b_{mn} \end{pmatrix}$$

Aufgabe 49 (H) Speisende Philosophen **(8 Punkte)**

Ein berühmtes Problem zur Illustration von Deadlocks ist das der speisenden Philosophen. In einer vereinfachten Form sitzen dabei n Philosophen um einen runden Tisch, auf dem ein Teller mit Spaghetti steht. Ein Philosoph braucht zwei Gabeln, um damit ordentlich Spaghetti zu essen; leider gibt es auch nur n Gabeln. Ein naiver Algorithmus könnte so aussehen, dass jeder Philosoph zuerst versucht, die Gabel zu seiner linken Hand aufzunahmen und dann die zu seiner rechten. Wenn die Philosophen eine Gabel erst dann wieder ablegen, wenn sie gegessen haben, führt dieser Algorithmus in ein Deadlock.

Schreiben Sie ein Java-Programm, das diese Verklemmung illustriert. Dabei soll jeder Philosoph als ein eigener Thread implementiert sein und die Gabeln als Monitore von Objekten. Achtung: Es kann sein, dass der erste Philosoph sich beide Gabeln greift, bevor der zweite überhaupt die erste nimmt. In diesem Fall bleibt das Deadlock aus. Sie können den Deadlock zuverlässig provozieren, indem Sie eine Verzögerung von 10ms zwischen dem Ergreifen der beiden Gabeln einbauen.