Übungszettel 1

Allgemein:
Abgabe via OLAT Upload bis Montag vor der nächsten Übung um 08:00! Fuer dieses Uebungsblatt Abgabe bis 9.11.2014!
Upload muss aus einem File mit Beschreibung der Lösung (pdf oder txt) **und** einem weiteren Archiv das Source-Code enthält (zip oder tar.gz, Eclips project exportieren!) bestehen. Punktabzug bei falschem Dateiformat oder mehr oder weniger als 2 Files (kein docx, tex, rar, 7z, Ohne Endung, …)! Zwei gültige Files z.b. name-u1.txt und name-u1.zip
Hilfreicher Link: http://openbook.galileodesign.de/javainsel5/javainsel09_000.htm

1) Java Threads: (23%)

Implementieren Sie ein Thread-Beispiel und erzeugen Sie in der main()-Methode 8 Threads. Die run()-Methode soll dabei lediglich die laufende Thread-Nummer anzeigen.

Wie viele Threads lassen sich erzeugen, bis das System "steht"? Beobachte den Speicherverbrauch (top oder ps). Lässt sich abschätzen, was ein Thread Speicher "kostet"? Mit welchen Parametern der Java Virtual Machine können Sie die Anzahl der startbaren Threads erhöhen? Schildern Sie ihre Konfiguration (Rechner, Betriebssystem, Java Version) und maximal erreichte Anzahl laufender Threads (+ mögliche mit getunten VM Settings).

2) “Thread Pool”: (23%)

Schreiben Sie ein Programm das mit Hilfe eines einfachen selbstgeschriebenen Thread-Pools die Konstante Pi (π) berechnet. Die Anzahl der Worker-Threads soll konfigurierbar sein. Verwenden Sie die Klasse java.math.BigDecimal für die Berechnung und folgende Summenformel:

$$\sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} = \frac{1}{1} - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \ldots = \frac{\pi}{4}$$

Jeder Arbeiter soll hierbei einen Teilbereich der Summe berechnen (z.b.: 2 Threads mit 4 Werten T1: 1-4, T2: 5-8, T1: 9-12, ...) und der Master-Thread entscheidet wann eine vom Benutzer definierte Genauigkeit erreicht ist. Beachten Sie das Ihre Berechnung ein korrektes Ergebnis liefern muss unabhängig der Threadanzahl!
Die Zeit zur Berechnung von Pi soll für 1 - 20 Threads gemessen werden. Wählen Sie eine entsprechende Genauigkeit (Laufzeit von wenigen Sekunden) für diese Versuche und vermerken Sie auch die verwendete Hard und Software.

3) Die Zustände eines Threads und deren Anzeige: (23%)

Schreiben Sie ein Programm das mehrere Threads erstellt. Das Programm soll versuchen je einen Thread in den von der VM möglichen Zuständen zu bringen. Ein weiterer (Monitoring)-Thread soll alle 3 Sekunden Details zu allen in der VM laufenden Threads (Thread.getAllStackTraces().keySet()) ausgeben inklusive deren Zustände. Beschreiben Sie wie die einzelnen Zustände eines Threads erreicht werden können und wozu diese dienen. Fertigen Sie ein State Diagram fuer Java Threads an.

4) Producer / Consumer: (30%)

Implementieren Sie mit Hilfe der Klasse java.lang.Thread einen Produzenten und einen Konsumenten von Zufallszahlen. Dabei sollen Produzent und Konsument parallel in zwei verschiedenen Threads laufen.

- Implementieren Sie vier Java-Klassen: Producer, Consumer, Buffer und MainTest.
- Der Producer soll als Thread solange Ganzzahlige Zufallszahlen von 0 bis 100 generieren, bis eine 0 auftritt. Der Consumer soll als Thread solange vom Producer Zufallszahlen abfragen, bis 0 geliefert wird.
- Der Producer speichert die produzierten und noch nicht abgeholten Zufallszahlen in einem Buffer. Es kann nur konsumiert werden, wenn sich Zahlen in dem Buffer befinden.
- Der Producer soll sich nach Produktion einer Zufallszahl für eine zufällige Anzahl von Sekunden aus dem Intervall [0, 3] schlafen legen. Der Consumer soll, wenn der Buffer leer ist, sich für genau zwei Sekunden schlafen legen und erst dann wieder beim Producer nachfragen.
- Die Klasse MainTest legt zuerst ein Objekt der Klasse Producer und dann ein Objekt der Klasse Consumer an. Dann startet die Klasse MainTest die beiden Threads.
- Das Programm soll auf der Konsole ausgeben was welcher Thread macht. Gestalten sie entsprechende Ausgabefunktionen.