Übungszettel 5

Allgemein:
Abgabe via OLAT Upload bis Montag vor der nächsten Übung um 08:00! Für dieses Übungsblatt Abgabe bis 1.2.2016!
Upload muss aus einem File mit Beschreibung der Lösung (pdf oder txt) **und** einem weiteren Archiv das Source-Code enthält (zip oder tar.gz, Eclipse Project exportieren!) bestehen. Punktabzug bei falschem Dateiformat oder mehr oder weniger als 2 Files (kein docx, tex, rar, 7z, Ohne Endung, …) bestehen! Zwei gültige Files z.B. name-u1.txt und name-u1.zip

Bitte nicht Vergessen, das 60% der Aufgaben gelöst werden muessen um fuer die Klausur am 8.2.2016 zugelassen zu werden!

Nützlicher Link:
http://java.sun.com/javase/6/docs/api/java/util/concurrent/ThreadPoolExecutor.html

1) GUI Tasks: (50%)

Implementieren Sie mit Hilfe der in der Vorlesung Vorgestellten Möglichkeiten eine GUI welche es erlaubt eine Anwendung (z.b. das Berechnen von Pi) zu Starten, man Feedback über den Fortschritt der Anwendung erhält (Progressbar) und es die Möglichkeit gibt diese Anwendung abzubrechen (Chancel button).

a) Das GUI sollte ein Mindestmaß an Ästhetik aufweisen.
b) Die Anwendung kann eine Beliebige sein, die Längere Zeit in Anspruch nimmt (nicht sleep()!) und die Möglichkeit bietet den Fortschritt abzuschätzen und einen Abbruch zu erzwingen.
c) Das Ergebnis der Anwendung soll bei ihrer Beendigung Angezeigt werden.
d) Stellen sie sicher, das die in der Vorlesung gelehrten Thematiken berücksichtig werden mit dem Resultat einer guten GUI die den Ansprüchen eines Kunden entsprechen würde.

1)ThreadPoolExecutor: (50%)

Implementieren Sie mit Hilfe des Files MCS.java ein Java Programm, das mit einem ThreadPoolExecutor die Montecarlo-Simulation mit mehreren Threads berechnet:

a) Passen Sie das gegebene Beispiel File an, das die Ausführung mittels Runnable (oder Callable) Objekten abgearbeitet wird.
b) Jedes Runnable Object soll 8000 Iterationen der j-Schleife berechnen. Insgesamt soll bis 16000000 Iterationen simuliert werden (Bei langsamen oder schnelleren Systemen sind beide Werte entsprechend anzupassen um eine Laufzeit von einigen Sekunden zu erreichen!).
c) Das Programm soll die 2000 Runnable Objekte sequentiell ausführen und die Ausführungszeit dabei gemessen werden.

d) Ein ThreadPoolExecutor soll verwendet werden um die 2000 Runnables abzuarbeiten. Legen Sie diesen mit entsprechenden Queue- und Pool-Größen an, das die Ausführung ohne Probleme funktioniert.
e) Wie Synchronisieren Sie das ende der Ausführungen? Beschreiben Sie kurz Ihre Lösung und andere Möglichkeiten die sich anbieten würden um 2000 Runnables zu Synchronisieren.
f) Stellen Sie sicher, das die Konfiguration des Executors nicht verändert werden kann!
g) Unter welchen Umständen kann der Executor Jobs “rejecten”? Erläutern Sie dieses mögliche Szenario.
h) Erweitern Sie das Programm, das der Fall von g) eintritt (Limits für Queuegrösse und Anzahl Threads). Die abgelehnten Jobs müssen aufgefangen und später (sobald möglich) erneut an den Executor übergeben werden. Welche der 4 möglichen Policy ist hierbei am Zielführensten? Beschreiben Sie auch die 3 nicht verwendeten und begründen Sie Ihre Auswahl.
i) Es gibt 3 Strategien was das Queuen betrifft (siehe Java API). Beschreiben Sie die 3 Strategien und nennen Sie Beispiele, wann diese Vor- und Nachteile haben und deren unterschiedlichen Anwendungsgebieten.