Übungszettel 3

Allgemein:
Abgabe via OLAT Upload bis Montag vor der nächsten Übung um 08:00! Für dieses Übungsblatt Abgabe bis 7.12.2014!
Upload muss aus einem File mit Beschreibung der Lösung (pdf oder txt) **und** einem weiteren Archiv das Source-Code enthält (zip oder tar.gz, 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

Nützlicher Link: http://jcip.net.s3-website-us-east-1.amazonaws.com/listings.html

1) Sharing Objects: (45%)

Schreiben sie die Klassen EventGeneratorThread, MySafeListener und ListeningObjectsThread die folgende Eigenschaften erfüllt und ein endsprechendes Testprogramm mit main():

a) Die Klasse MySafeListener soll sich als Listener Klasse bei einer anderen Klasse anmelden können entsprechend dem Codebeispiel SafeListener. Im falle eines Events soll der laufende Thread und das Event auf der Konsole ausgegeben werden. Wieso ist eine Listener Registrierung direkt im Konstruktor zu vermeiden? Begründen Sie die Antwort ausführlich!
b) Die Klasse EventGeneratorThread soll über eine ThreadLocal ArrayList verfügen die jeweils nur einem Thread zungaenglich ist. In der ArrayListe werden die Objekte gespeichert, die als Listener darauf angemeldet sind. Über die Funktion infoListeningObjects() soll auf der Konsole der Inhalt der Liste ausgegeben werden. Hinweis: Durch ThreadLocal wird die Ausgabe dieser Liste nur vom EventGeneratorThread aus moeglich!
c) Ein Testprogramm das zwei EventGeneratorThreads startet (alle 3 Sekunden einen Event mit der Zeit und der Herkunft des Events als Message.). Jeweils ein MySafeListener Objekt soll als Listener für jeden Thread registriert werden.
d) Ein weiterer ListeningObjectsThread soll alle 4 Sekunden von alle Laufenden EventGeneratorThreads die Funktion infoListeningObjects() aufrufen. (Implementierung soll mit beliebiger Anzahl von Threads funktionieren!)
e) Das Testprogramm soll um 4 weitere Threads erweitert werden die ebenfalls Events generieren. Die 2 bestehenden MySafeListener sollen sich jeweils bei 3 dieser 4 Threads anmelden.
f) Stellen Sie sicher, das der EventGeneratorThread Immutable Event Objecte verschickt (Sie müssen selbst eine endsprechende Eventklasse Implementieren). Welchen Vorteil haben Immutable Klassen im Allgemeinen?

2) Composing Objects: (30%)

a) Nehmen sie das Beispiel NumberRange und Programmieren Sie mehrere Threads die jeweils Zufällige Upper und Lower Werte setzen.

http://jcip.net.s3-website-us-east-1.amazonaws.com/listings/NumberRange.java

Ein weiterer Thread soll Zufallszahlen mit isInRange(int i) Überprüfen und auf der Konsole deren Wert und das Ergebnis Ausgeben.

b) Wie müssen Sie das Timing Ihrer Threads und der Klasse anpassen das es zu möglichen Problemen / Race conditions kommen kann?

c) Implementieren Sie diese Änderungen und schildern Sie wieso und was für Probleme auftreten können.

d) Verändern Sie die Klasse NumberRange so, das Sie Thread save ist.

3)HashMap Benchmark: (25%)

In dieser Aufgabe geht es um den Vergleich der verschiedenen HashMap-Zugriffe: ohne Synchronisierung, Synchronized und Concurrent

a) Ihr Benchmark soll von 1, 2, 4, 8 und 16 Threads aus auf eine gemeinsame HashMap zugreifen und dort zufällige Integer-Objekte einfügen (50% der zugriffe), lesen (contains, 25%) und entfernen (25%).

b) Messen Sie den möglichen Objektdurchsatz (HashMap-Zugriffe pro Sekunde) für die unterschiedlichen Threadanzahlen und HashMaps:
    a) Ohne Synchronisierung
    b) Synchronized (muss selbst implementiert werden)
    c) Concurrent

c) Wiederholen Sie die Messungen mit 80% Einfügen, 15% Lesen und 5% Entfernen (80, 15, 5) aus der Map und (5, 90, 5). Gibt es unterschiede in den Ergebnissen? Erklären Sie diese. Vergessen Sie nicht anzugeben auf was für einem System die Tests durchgeführt wurden.