Ü 2.1 JAVA THREAD SAFETY

Recherchieren Sie was Thread Safety im Zusammenhang mit Java Klassen bedeutet

Thread Safety in Java ist der Prozess, ein Programm sicher in Multithread-Umgebungen zu verwenden. Es gibt verschiedene Möglichkeiten, wie man ein Programm threadsafe machen kann:

- Synchronisation (mittels keyword synchronized) ist die einfachste und am weitesten verbreitete Methode für Thread Safety in Java.
- Verwenden von der Atomic Wrapper Klasse (zum Beispiel AtomicInteger)

  Der eigentliche Vorteil der Atomic * -Klassen besteht darin, dass sie eine atomare Compare-and-Swap-Methode bereitstellen, die für die Implementierung von Lock-Free-Algorithmen sehr nützlich sein kann.

  https://docs.oracle.com/javase/7/docs/api/java/util/concurrent/atomic/package-summary.html

- Verwenden von Locks aus dem java.util.concurrent.locks Package
- Verwenden von threadsicheren „Collection classes“
  - BlockingQueue definiert eine first-in-first-out Daten Struktur welche blockiert oder pausiert wenn man versucht etwas zu einer vollen Queue hinzuzufügen oder aus einer Leeren etwas abzurufen.
  - ConcurrentMap ist ein Subinterface von java.util.Map und definiert atomare Operationen. Diese entfernen oder ersetzen ein key-value pair nur wenn der key verfügbar ist oder fügen ein key-value par hinzu wenn der key nicht verfügbar ist. Der Standardfall für die Implementierung der ConcurrentMap ist die ConcurrentHashMap
  - ConcurrentNavigableMap ist ein Subinterface von ConcurrentMap die ungefähre Übereinstimmungen unterstützt. Der Standardfall für die Implementierung der ConcurrentNavigableMap ist die ConcurrentSkipListMap
- Verwenden des keywords „volatile“ für Variablen, damit jeder Thread Daten vom Speicher liest und nicht vom Thread-Cache.