HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Threads und nebenläufige Programmierung

Programmierpraktikum 2012

Jiaxiang Zhang

Threads und Prozesse

Definition 1: Parallelität

-gleichartige Beschaffenheit

-quasi-parallelität

-echter Parallelismus

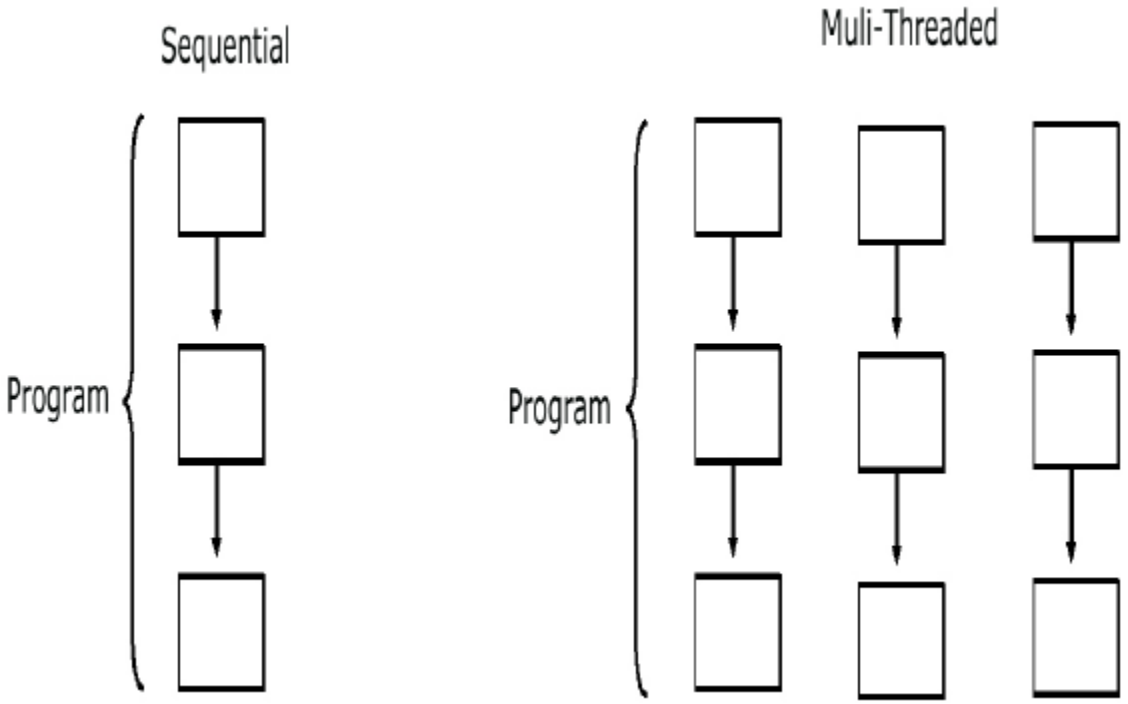
Definition 2: Nebenläufigkeit (Concurrency)

Parallele Ausführung von Anweisungen auf einem oder mehreren Prozessoren.

Definition 3: Thread

Ein Thread (engl. für Faden) stellt eine nebenläufige Ausführungseinheit innerhalb genau eines Prozesses dar.

Threads und Prozesse



Warum Threads in Java?

- **Warum unterstützt die Java-API und die virtuelle Maschine Threads?**

-Threads nutzen die vorhandenen Ressourcen moderner Betriebssysteme und Hardware effizienter aus.-transparent auf mehrere Prozessoren verteilt.

-Threads werden durch ein geeignetes Betriebssystem für den Programmierer und Applikationsanwender transparent auf mehrere Prozessoren verteilt.

-Threads erhöhen den Interaktionskomfort für den Applikationsanwender.

Threads erzeugen

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

- Thread über die Schnittstelle Runnable implementieren

- interface java.lang.Runnable
- void run()

  Diese Methode enthält den parallel auszuführenden Programmcode

- Thread mit Runnable starten
- class java.lang.Thread
- Thread(Runnable target)

Erzeugt einen neuen Thread mit einem Runnable,das den parallel auszuführenden Programmcode vorgibt.

- void start()

Ein neuer Thread -neben dem die Methode aufrufenden Thread - wird gestartet. Der neue Thread führt die run()-Methode nebenläufig aus.Jeder Thread kann nur einmal gestartet werden.

Synchronisation über Kritische Abschnitte

- Gemeinsam genutzte Daten

```
class T extends Thread
{
   static int result;
      public void run(){...}
}
```

-verschiedenen Exemplare der Klasse T,die jeweils einen Thread bilden , Daten austauschen,indem sie die Informationen in result ablegen oder daraus entnehmen
-Thread können Informationen entnehmen oder Zugriff auf gemeinsame Objekte über eine Referenz
-potenziell parallel ablaufende Aktivitäten

Synchronisieren mit synchronized

kritische Abschnitte:

Zusammenhängende Programmblöcke,die nicht unterbrochen werden dürfen und besonders geschützte werden müssen.

Nötigkeit:

Wir können das etwas lockerer sehen,wenn wir wissen,dass innerhalb der Programmblöcke nur von den Daten gelesen wird.Sobald aber nur ein Thread Änderungen vornehmen möchte,ist ein schutz nötig.

Monitor:
Soll die Laufzeitumgebung nur einen Thread in einen Block lassen,nutzt Java einen Monitor.

Synchronisieren mit synchronized

Lock:

Ein Monitor wird mit Hilfe eines Lockrealisiert,welches ein Thread öffnen oder schließen kann.Tritt ein Thread in den kritischen Abschnitt ein,können wir uns vorstellen,dass die virtuelle Maschine den Programmcode wie eine Tür abschließt.

Unlock:

Erst wenn der Abschnitt durchlaufen wurde,schließt JVM die Tür wieder auf und ein anderer Thread kann den Abschnitt bereten.

Synchronisieren mit synchronized

- kritsche Abschnitte mit synchronized geschützt

```
public synchronized void fun()
```

- Mit synchronized synchronisierte Blöcke

```
sysnchronized(objektMitDemMonitor)
{
...
}
```

Threads mit Rückgabe über Callable

- schnittstelle callable die dem Aufrufer eine Rückgabe übermittelt
- Interface java.util.concurrent.Callable<v>
- V call()

Diese Methode enthält den parallel auszuführenden Programmcode und liefert eine Rückgabe vom Typ V.

```
class WordLengthCallable implements Callable{
        private String word;
        WordLengthCallable(String word) {
          this.word = word;
        }
        public Integer call() {
                return Integer.valueOf(word.length());
        }
}
```

Deadlocks

eadlocks

In dieser Situation können beide nicht vor und zurück und befinden sich in einem dauernden Wartezustand.

Vielen Dank für Ihre Aufmerksamkeit!

Quelle:Java ist auch eine Insel: Programmieren mit der Java Standard Edition Version 5(Galileo Computing)

http://www.jeckle.de/vorlesung/javaThreads/script.html#Einfuehrung

https://blogs.oracle.com/CoreJavaTechTips/entry/get_netbeans_6