[Literatur 1](#_Toc86918504)

[Grundlagen zu Threading 1](#_Toc86918505)

[Was ist ein Thread 1](#_Toc86918506)

[Was ist Nebenläufigkeit 2](#_Toc86918507)

[Was ist Parallelität 2](#_Toc86918508)

[Unterschied zwischen Nebenläufigkeit und Parallelität 2](#_Toc86918509)

[Folgen von Nebenläufigkeit 2](#_Toc86918510)

[Synchronisierung 2](#_Toc86918511)

[Blockende Synchronisierung 2](#_Toc86918512)

[Nicht Blockende Synch 2](#_Toc86918513)

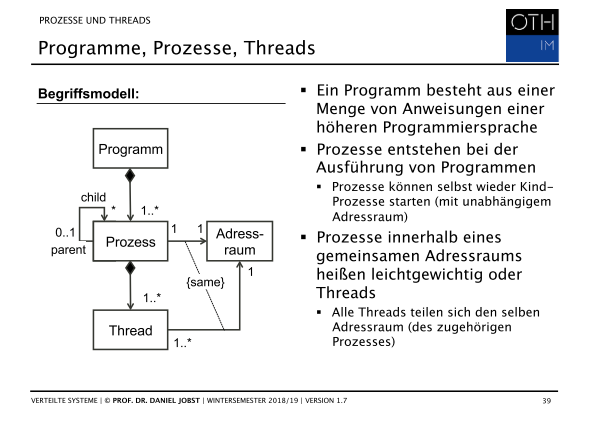
[Vergleich zwischen Block und nichtblock 2](#_Toc86918514)

# Literatur

* Concurrency is not Parallelism
* Parallelism /= Concurrency
* Parallelism Is Not Concurrency
* Nebenläufige Programmierung mit Java : Konzepte und Programmiermodelle für Multicore-Systeme
* Simple, fast, and practical non-blocking and blocking concurrent queue algorithms
* Verteilte Systeme

# Grundlagen zu Threading

## Was ist ein Thread

* Threads aus Java-, OS- und Hardwaresicht -> Bild (S13)
* 
* Context switch ist bei Threads einfacher als bei Prozessen, da weniger zwischengespeichert werden muss.

## Was ist Nebenläufigkeit

* „Zwei oder mehrere Aktivitäten (Tasks) heißen nebenläufig, wenn sie zeitgleich bearbeitet werden können.“
* Die Autoren bezeichnen Thread als Abstraktionskonzept der Nebenläufigkeit in Java.
* Die Autoren identifizieren die Zerlegung in unabhängige Teilaktivitäten und Synchronisierung bzw. Steuerung als Aufgaben der nebenläufigen Programmierung
* Programming as the composition of independently executing processes
* Concurrency is about dealing with lots of things at once.
* Concurrencys Goal is a good structure

## Was ist Parallelität

* Programming as the simultaneous execution of (possibly related) computations.
* Parallelism is about doing lots of things at once.

## Unterschied zwischen Nebenläufigkeit und Parallelität

* Concurrency is about structure, parallelism is about execution.
* Concurrency provides a way to structure a solution to solve a problem that may (but not necessarily) be parallelizable.

## Folgen von Nebenläufigkeit

* Race Conditions <-> Threadsicherheit
* Notwendigkeit der Koordinierung (Synchronisierung) von Zugriffen auf geteilte Ressourcen

## Synchronisierung

* „to algorithms for concurrent data structures, including FIFO queues, fall into two categories:blockingandnon-blocking.“

### Blockende Synchronisierung

* Blocking algorithms allow a slow or delayed process to prevent faster processes from completingoperations on the shared data structure indefinitely.

### Nicht Blockende Synch

* Zb. Synch durch Nutzung atomarer Operationen?
* „Non-blocking algorithms guarantee that if there are one or moreactive processes trying to perform operations on a shareddata structure, some operation will complete within afinitenumber of time steps“
* Compare\_and\_swap
* wait-free Algorithm

### Vergleich zwischen Block und nichtblock

* „On asynchronous (especially multi-programmed) multiprocessor systems, blocking algorithmssuffer significant performance degradation when a processis halted or delayed at an inopportune moment. Possible sources of delay include processor scheduling preemption,page faults, and cache misses. Non-blocking algorithmsare more robust in the face of these events.“