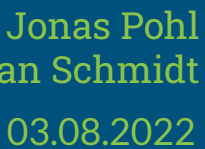




# Concurrency in Java



Jonas Pohl  
Florian Schmidt  
03.08.2022

# Gliederung

---

- Prozesse vs. Threads
- Was ist Concurrency?
- Arten von Concurrency
- Vorteile/Anwendungen
- Concurrency in Java
- Mögliche Probleme
- kleines Beispiel

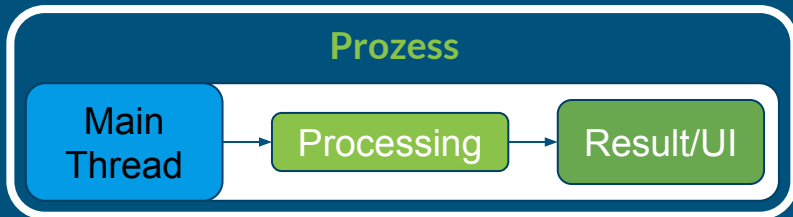
# Prozesse vs. Threads

---

- Prozesse = grundsätzlich isolierte Programme
  - Können jedoch Child-Prozesse spawnen;
  - und durch *Inter-Process Communication* (IPC) kommunizieren
  - Ein Prozess hat immer mindestens einen Thread
- Threads = sind Teil eines Prozesses
  - ein Prozess kann über mehrere Threads verfügen
  - Vorteil ggü. mehreren Prozessen:
    - geteilter Speicherbereich → einfacher Datenaustausch

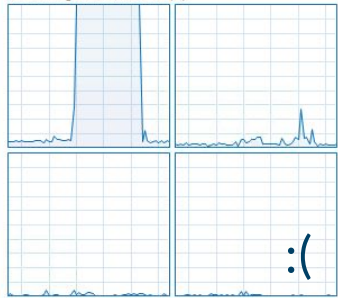
# Was ist Concurrency?

## Single-Threaded



### CPU

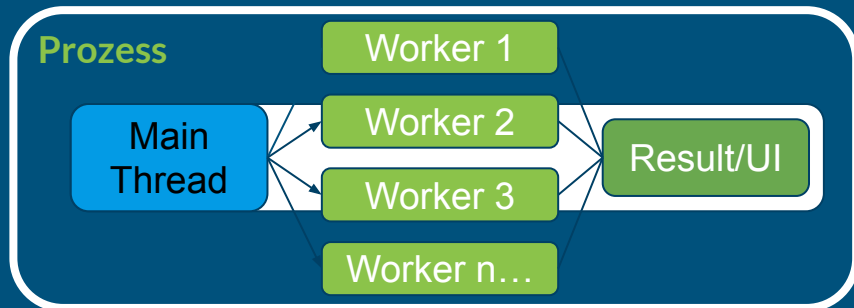
Auslastung in 60 Sekunden (%)



- langsamer
- Gefahr des *Blocking*: Hauptthread ist beschäftigt, aktualisiert UI nicht → "Programm friert ein"

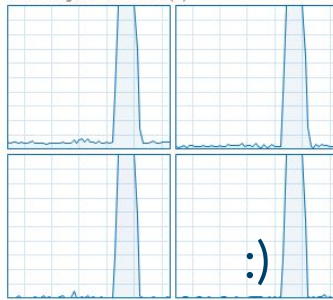
vs.

## Multi-Threaded



### CPU

Auslastung in 60 Sekunden (%)



- schneller (besonders auf Mehrkernprozessoren)
- *Blocking* des Hauptthreads wird vermieden
- (etwas geringerer Stromverbrauch)

# Vorteile von Concurrency

- **Parallelisierung** von Datenverarbeitung — z.B. Rendering, Videoencoding, ...
- **Trennung von GUI-Render-Schleife** und “backend”-Logik:
  - Allgemein: bessere UI-Responsiveness; z.B.:
    - GUI-Kontrolle während laufender Aufgabe — z.B. Fortschrittsanzeige, Abbrechen der Aufg.
    - flüssigeres Scrollen — Content kann asynchron nachgeladen werden
    - Animationen, während Programmfunktionen laden
- **Separation** → robusteres Programm
  - z.B. ein Browsertab hängt sich auf: (unendliche JS-Loop...)
    - Nur der Tab-Thread hängt sich auf, Main-Thread (also GUI) behält Kontrolle:
      - erkennt Loop, gibt Warnung, Nutzer kann Tab schließen/neustarten

# Multithreading in Java

---

„extends Thread“ (Class)	„implements Runnable“ (Interface)
<ul style="list-style-type: none"><li>• eine Instanz → ein Thread<ul style="list-style-type: none"><li>◦ Threads haben eigenen Speicherbereich / Variablen</li></ul></li><li>• Unterklasse kann keine weitere Klasse erweitern</li></ul>	<ul style="list-style-type: none"><li>• eine Instanz der Klasse kann mehrere Threads erzeugen<ul style="list-style-type: none"><li>◦ Threads einer Instanz greifen auf gleiche Variablen zu</li></ul></li><li>• implementierende Klassen können eine andere Klasse erweitern und/oder weitere Interfaces implementieren</li></ul>

# Java: Lifecycle / Thread States

---

- Newborn → *Thread-Klasse wurde instanziiert*
  - Thread wurde erzeugt, aber start() noch nicht aufgerufen → keine Codeausführung
- Runnable → *start() wurde aufgerufen ...*
  - Thread wartet auf Ausführung (Java Thread Scheduler)
- Running → *Thread wird derzeit ausgeführt*
- Dead → *Thread ist fertig*
  - run() - Methode ist abgeschlossen, oder stop() wurde aufgerufen
- Blocked → *Thread wurde pausiert*
  - sleep() // wait() // suspend() [deprecated]

---

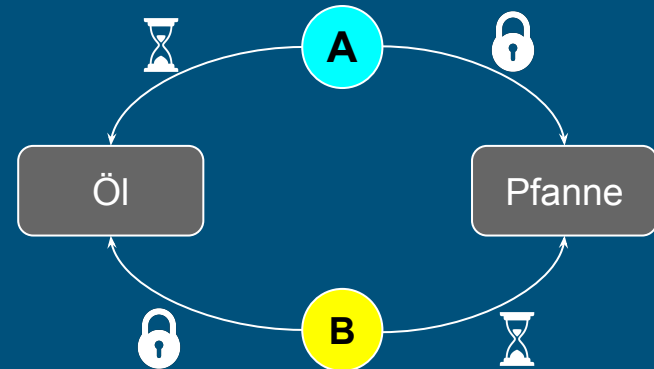
- Das Risiko von `suspend()` vs. `sleep()`

- `sleep()` ist `statisch`, kann also nur IM Thread aufgerufen werden.
  - Es ist unmöglich, dass der Thread sich selbst `sleep()` aufruft, während Ressourcen gesperrt sind.
  - Ressourcen bleiben nie gesperrt
- `suspend()` ist eine Instanzmethode – wird also „von außen“ aufgerufen
  - kann (mit Pech) aufgerufen werden, während Ressourcen genutzt werden
  - Deadlock-Risiko



# Mögliche Probleme

- Deadlock - ganz allgemein:
  - Alice möchte ein Spiegelei braten.
  - Bob möchte gleichzeitig Zwiebeln anbraten.
- 1. Alice nimmt sich die (einzige) Pfanne.
- 2. Bob sieht: die Pfanne ist (noch) besetzt. Er nimmt sich das Öl.
- 3. Alice sieht: das Öl ist (noch) besetzt.
- I. Alice wartet bis Bob fertig ist.
- II. Bob wartet bis Alice fertig ist.



# Mögliche Probleme

---

- Race Conditions (Wettlaufsituation)
    - Situation, in der das Ergebnis von einem Prozess abhängig von der zeitlichen Abfolge der Threads ist
    - Beispiel:
      - Bei Programmstart werden Daten von einer API abgefragt. (asynchron)
      - Ein Button startet eine Datenverarbeitungs-Methode.
- Der Entwickler hat gutes Internet, deshalb vergisst er:  
Die API-Response kann länger dauern als bis zum Button-Click.

# kleine Anwendung

---

# Quellen

---

<https://www.javatpoint.com/thread-states-in-java> (Abgerufen: 20.02.2022)

<https://www.geeksforgeeks.org/multithreading-in-java/> (Abgerufen: 07.03.2022)

<https://mkyong.com/java/java-get-number-of-available-processors/> (Abgerufen: 07.03.2022)

<https://sites.google.com/site/projectcodebank/fyi/java-core/concurrency/difference-between-sleep-suspend-and-wait> (Abgerufen: 07.03.2022)

<https://www.javatpoint.com/runnable-interface-in-java> (Abgerufen: 07.03.2022)

## Bildquellen:

<https://icon-library.com/images/lock-icon-transparent-background/lock-icon-transparent-background-15.jpg>

<https://www.iconsdb.com/icons/preview/white/sandglass-xxl.png>