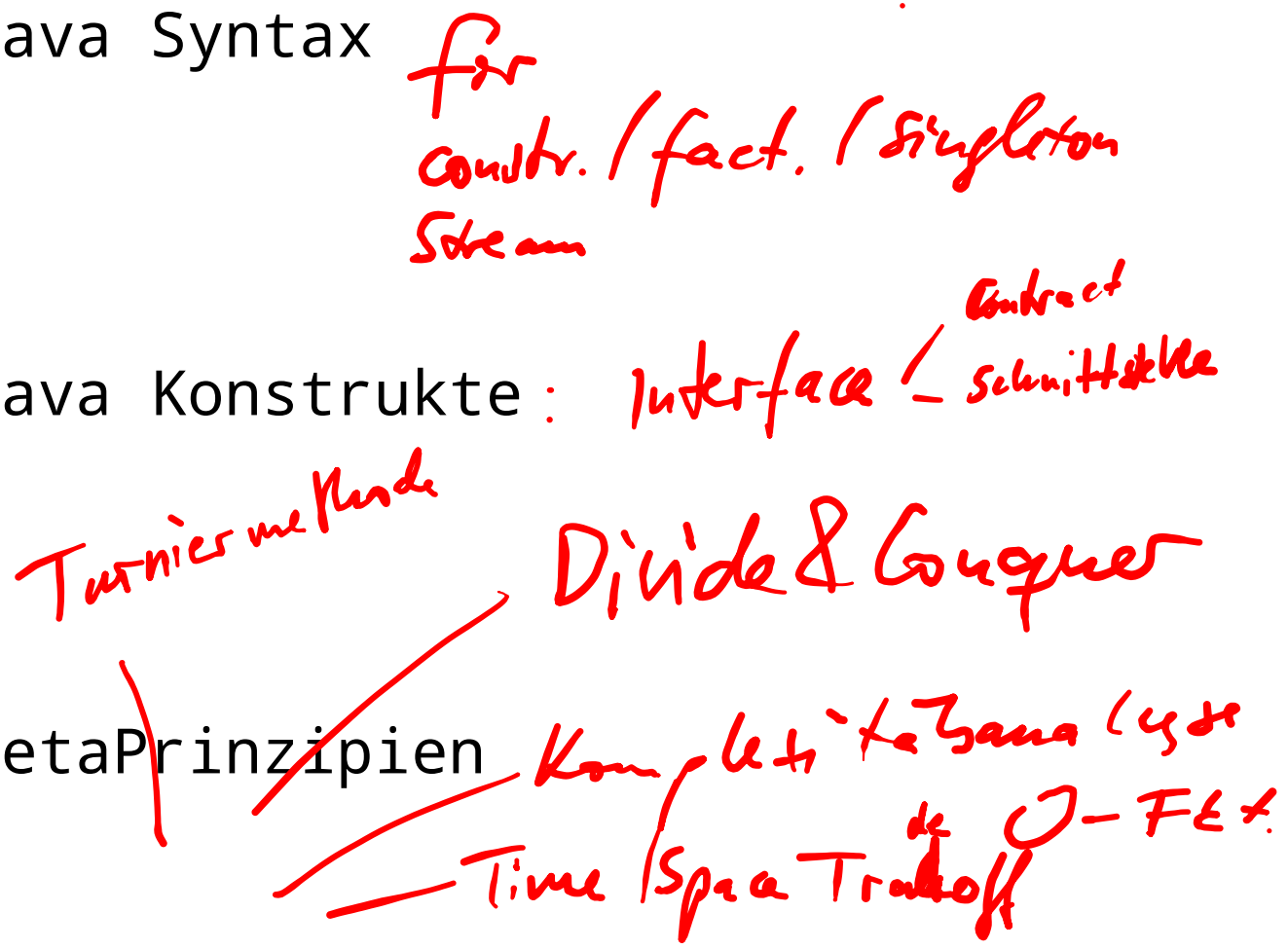
Mosbach – Java 2

VL 09

Wiederholung

- Java Syntax
- Java Konstrukte
- MetaPrinzipien



Wiederholung

- Algorithmen/Strukturen
  - Sortieren
  - Suchen
  - Selektieren
  - Stack/Queue
- Backup
  - git repo
- Testen
  - Extremfälle
  - Grenzwerte
  - innere

BubbleSort
InsertionSort
SelectionSort
QuickSort
MergeSort

in situ
stabil

Array
ArrayList

Lineare Suche
Binäre Suche
Hashing

3. größte - Sortier schau
- Quickselect.
1.+2. - Turniermethode

LI
FIFO
LIFO

Wiederholung: Schlagwörter

- schnell
- merge
- stabil
- compareTo()
- herrsche
- Dummy
- toString()
- korrekt
- pivot
- konsistent
- equals()
- Grenzwerte
- in-situ
- tue es immer

immer Köpfe

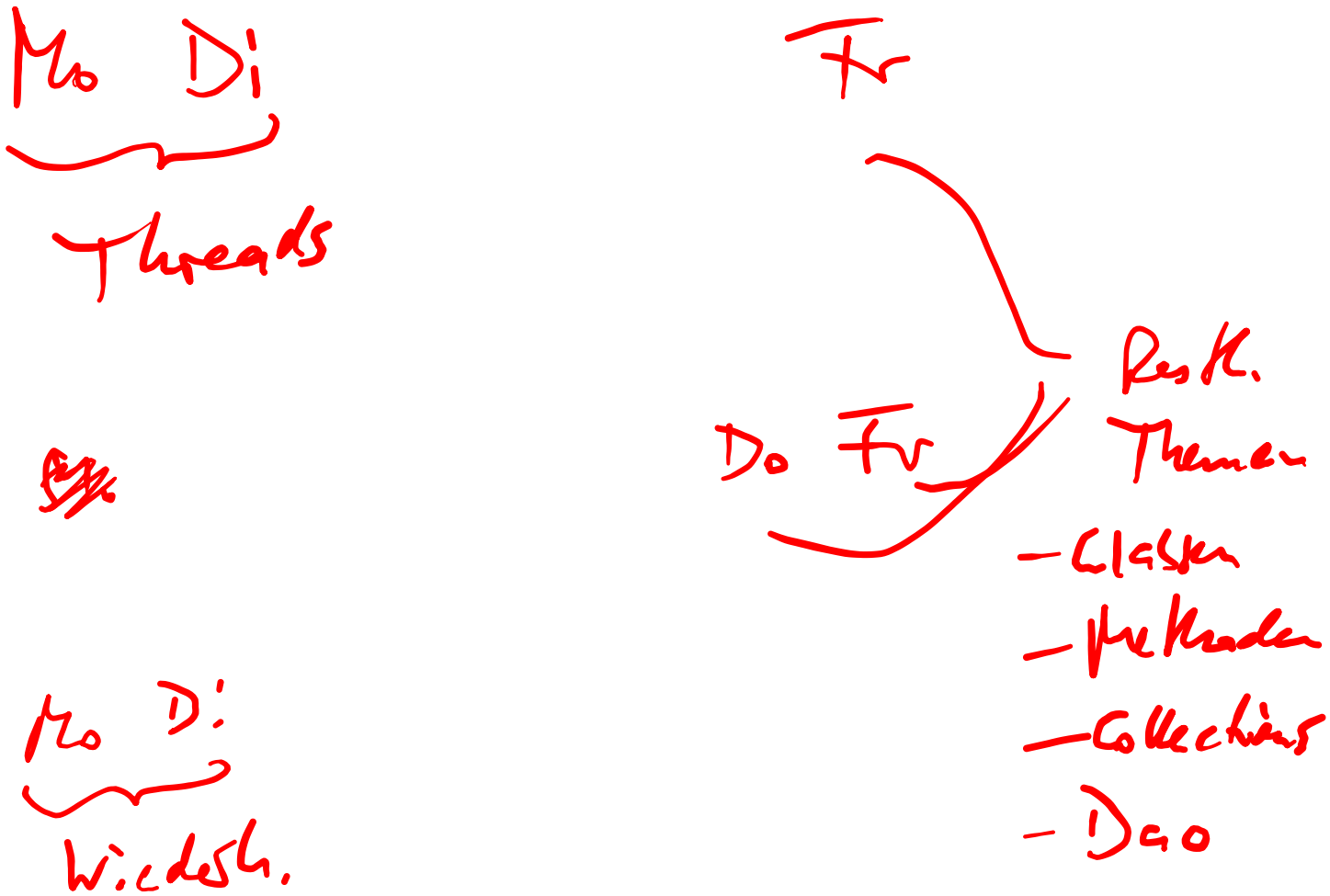
kommt noch als Thema

kommt noch als Thema

Getter/Setter, public/private

kommt noch als Thema

Planung

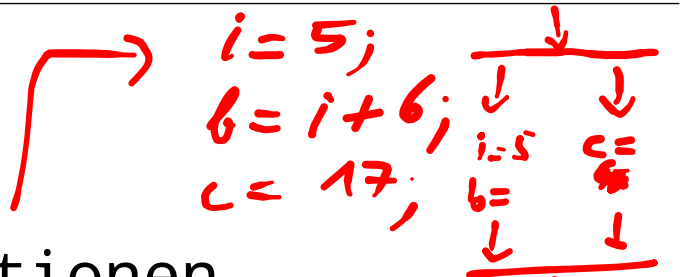

Mo Di
Threads
Mo Di:
Wiederh.
Fr
Do Fr
Restl.
Themen
- Classen
- Methoden
- Collections
- Dao

Threads

warum?

- Performance, schneller
  - mehrere Prozessoren
  - im Programm gibt es oft "Wartezeiten", GUI
- Lebendigkeit von Programmen, "Apps"

Begrifflichkeiten



Probleme

- Zugriff auf gemeinsame Ressourcen
- Code wird „oft" schwieriger

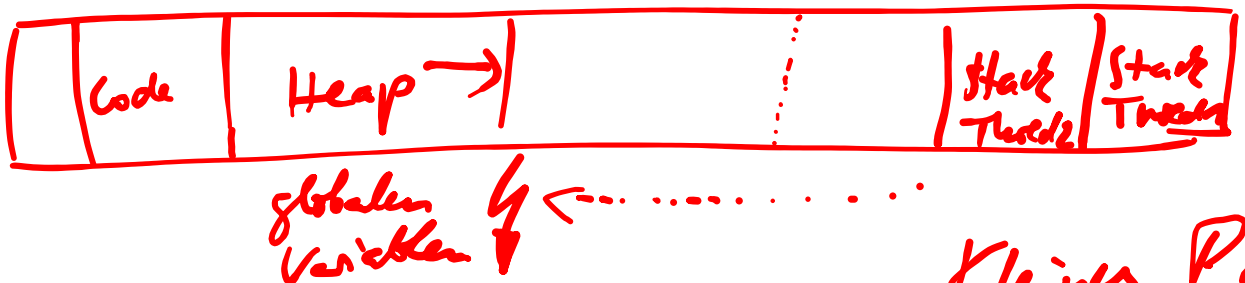
Definitionen

- nebenläufig: Ein logischer Zusammenhang
- parallel: Gleichzeitig
- verteilt: — " — und an phys. verschiedenen Orten

Scheduler
Zeitscheibe

Warum nicht neue Prozesse/Programme starten?

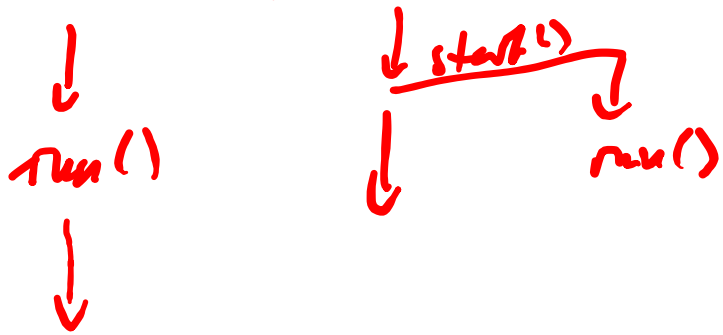
- Programm ist viel umfangreicher für den Prozessor
- Thread ist viel leichter anzulegen/zu stoppen



Kleines Problem: Was, wenn zu viele Threads?

Java Syntax

- Thread : geht, aber lieber vermeiden
- Runnable : besser
- run() : müssen wir überschreiben
- start() : a) baut im Speicher neuen Thread
  b) ruft dort run() auf



Aufgabe 1 zusammen

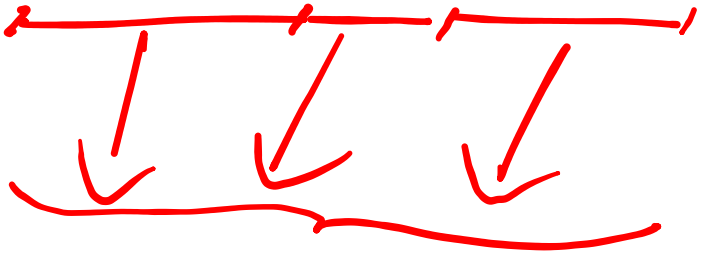
- Zähle die Anzahl von Zahlen
  - % 3 == 0
  - % 3 == 1
  - % 3 == 2

  in einem Array mit drei Threads

- Nutze
  - Thread
  - Runnable
  - priorities

Aufgabe 2 allein

- Gib Dir ein array mit Zufallszahlen
- Nutze x (3) Threads
- Jeder Thread addiert alle Zahlen in einem Intervall
- Dann Ausgabe



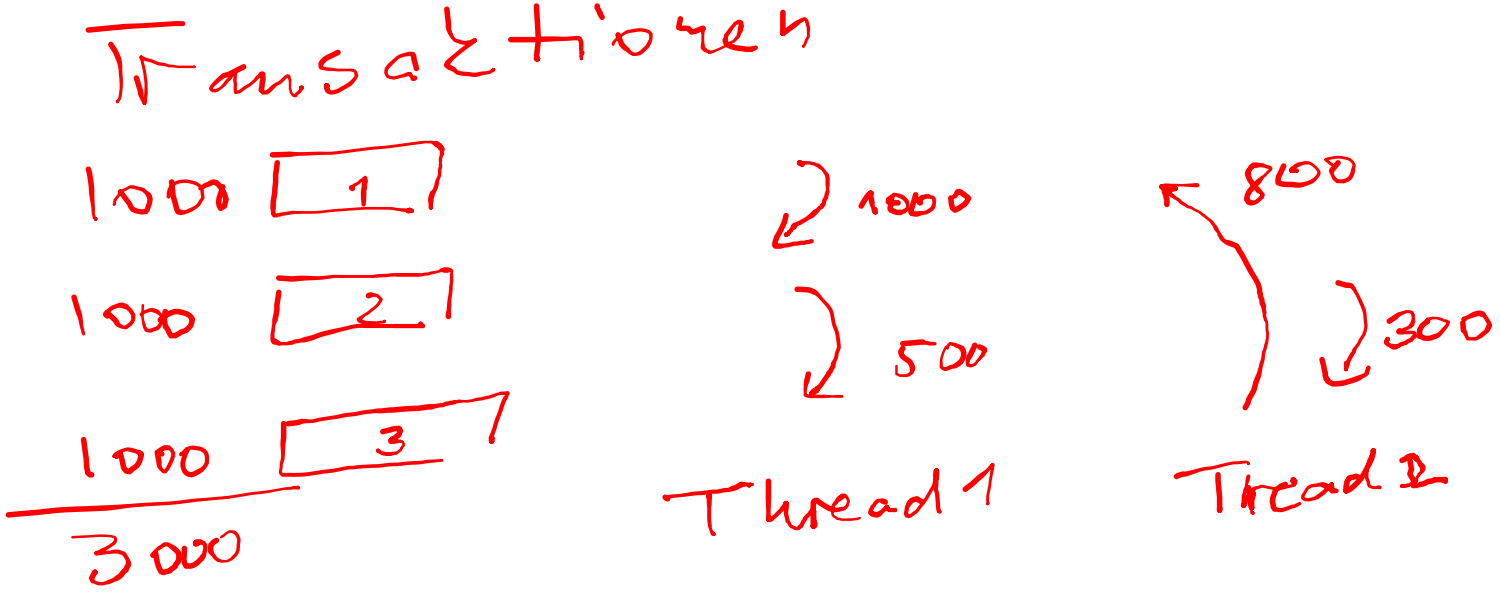
- Mache einen Wettkampf gegen ein Programm mit nur einem Thread

join()

Aufgabe 3

- Bank mit 3 Konten
- Kunden mit Zugriffen

- Zeige: Kontostand wird chaotisch



Lösung des Zugriffsproblemes

$i = i + 5$ ist nicht eine Einzelaktion sondern mehrere, Probleme, wenn unterbrochen

Lösung: Schutz

```
{
synchronized(      )
}
```

↑ Monitor

Neues Problem

Synchronized / Lock

aber

Deadlock-Gefahr

Aufgabe 3

- Bank mit 3 Konten
- Kunden mit Zugriffen
- synchronized

- thread-safe!

Aufgabe 3 Atomic

- Bank mit 3 Konten
- Kunden mit Zugriffen
- AtomicInteger

- thread-safe!

Die Besonderheit von Tests bei Threads

- Korrektheit
  - (single thread) Teste den Code wie üblich als gäbe es nur einen Thread
  - (multi threads) Zeige, dass der Code thread-safe ist
- Performanz
  - Zeige, dass Multithread geforderte Performanzparameter einhält

Wiederholung

- Gründe für Threads
- Shared resources
- synchronization
- deadlocks

- Thread
- Runnable
- run, start
- synchronized(mon)

Optimierungen, Portfolio-Aufgabe

- Anzahl Threads
- ThreadPools
- synchr. klein + Lock
- Atomic Strukturen
- volatile !!
- sleep/wait
- BlockingQueue

Future
Streams
virt. Threads (Java 15)