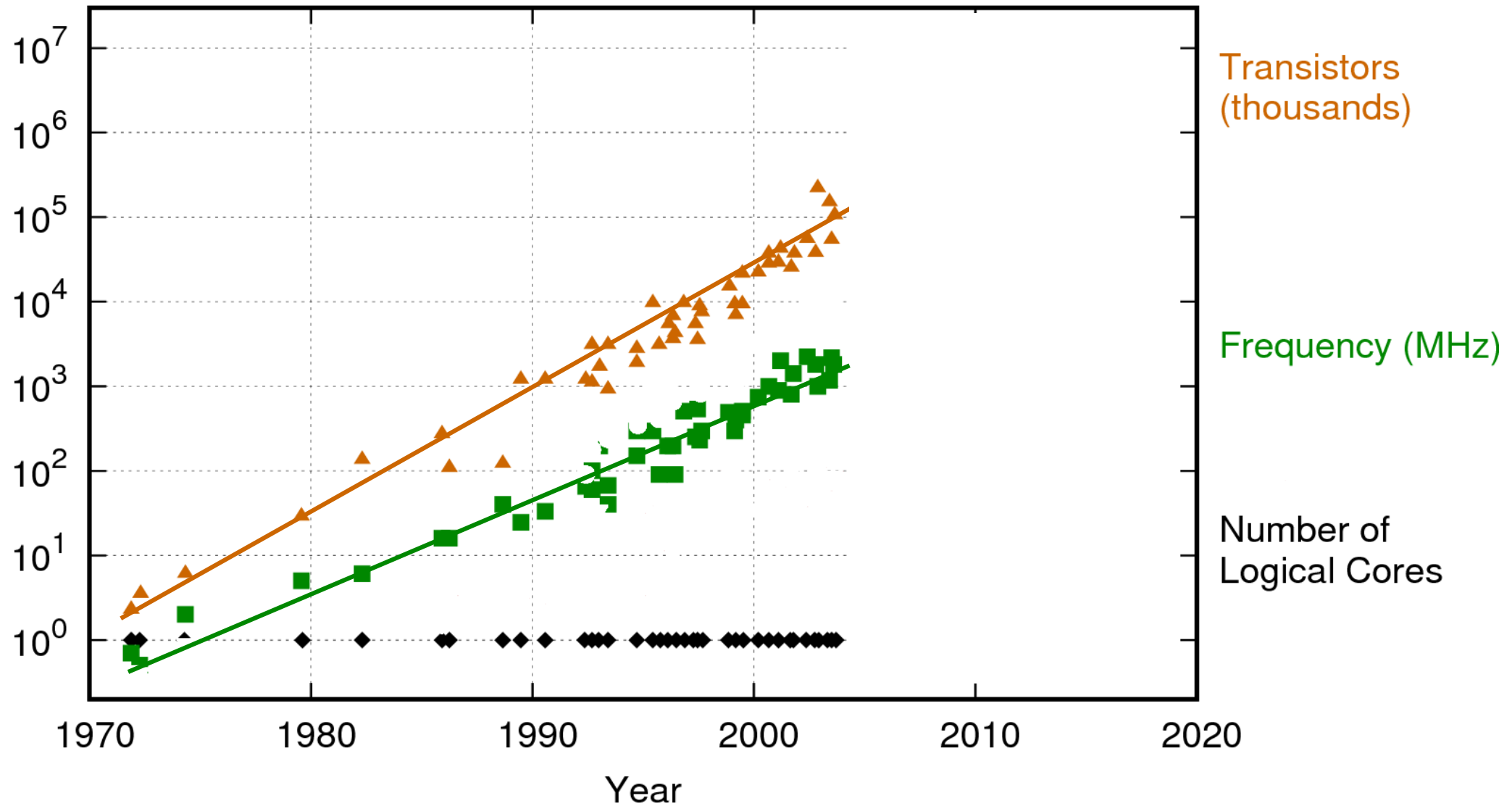


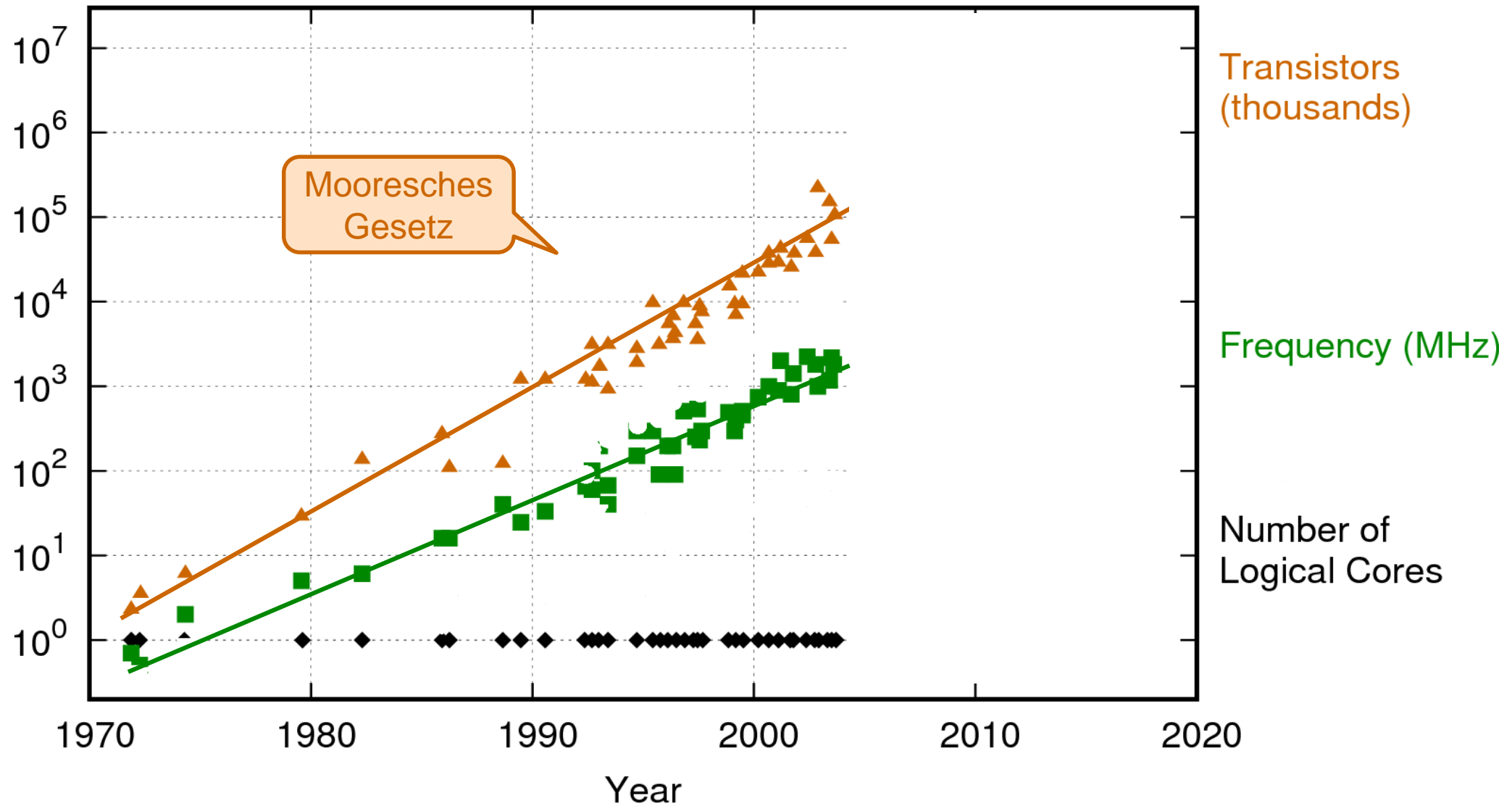
Multi-Thread- Programmierung

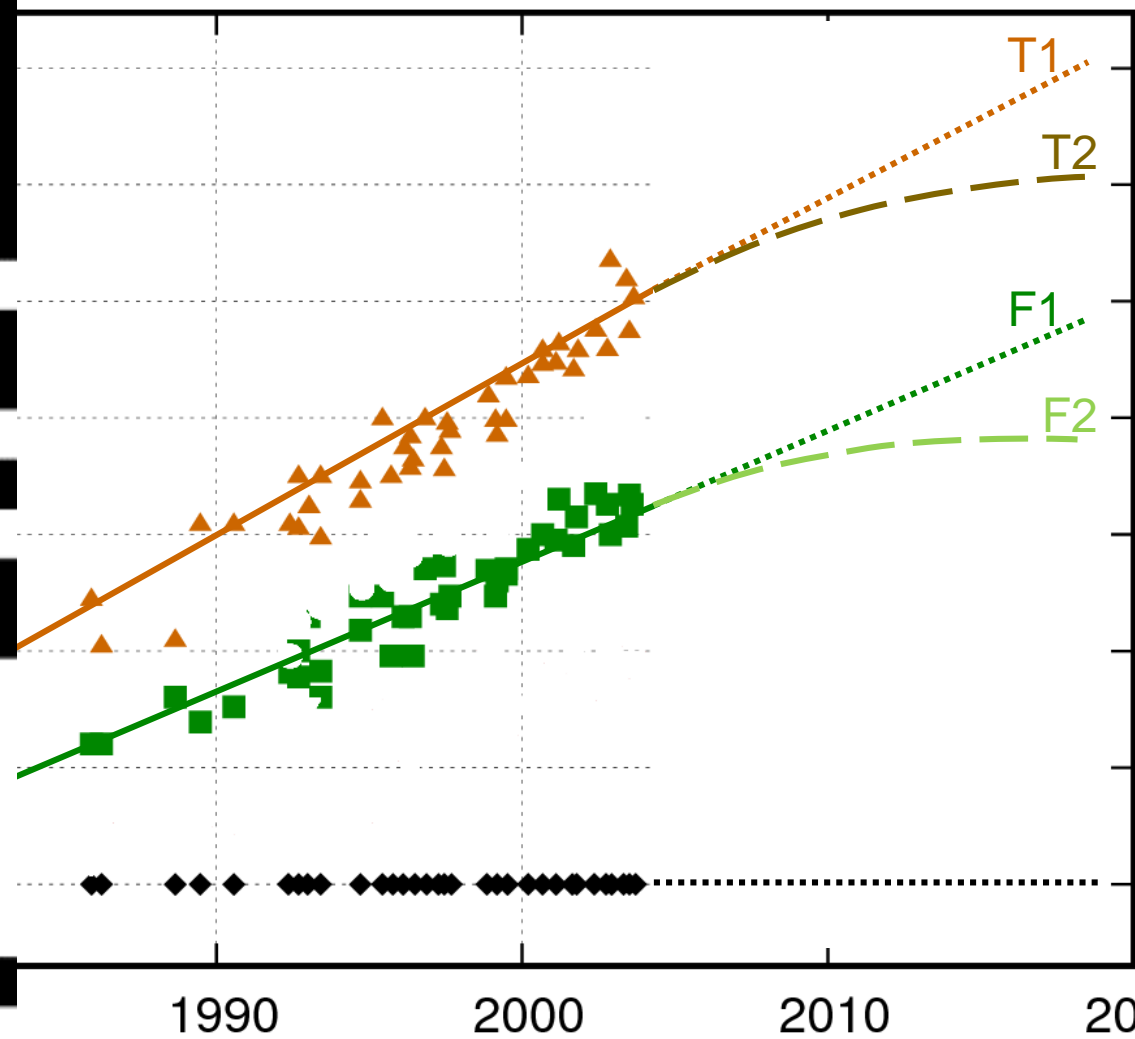
19.04.2023
Dr. Marcus Mews

42 Years of Microprocessor Trend Data [1]



42 Years of Microprocessor Trend Data [1]



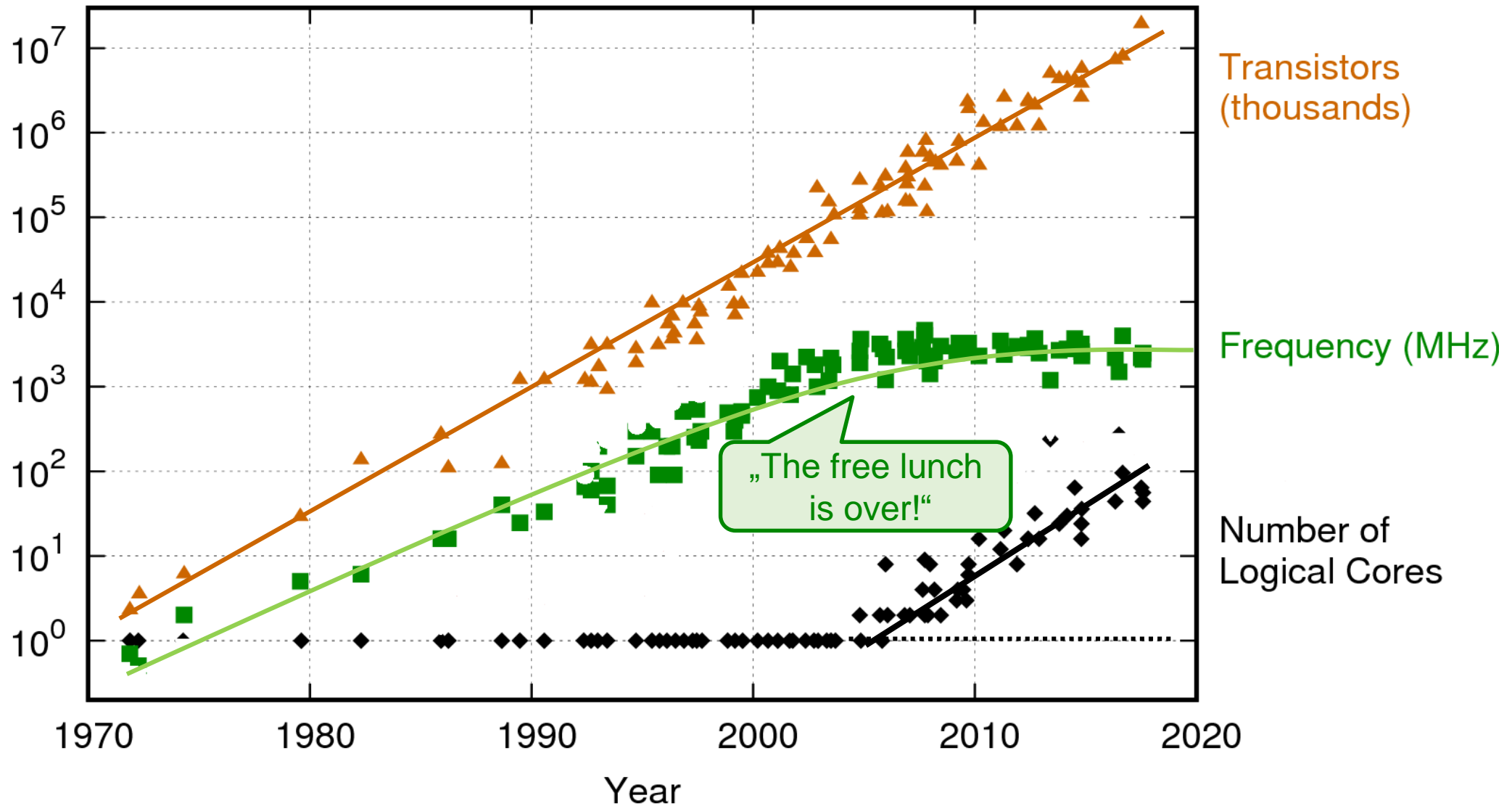


github.com/marcusmews/bht_plv

fast-poll.com/poll/11c73e9b

fast-poll.com/poll/results/11c73e9b

42 Years of Microprocessor Trend Data [1]



Ziele heute

Begriffe

- Thread, Prozess, Programm
- Konflikt, Synchronisation
- Scheduler, Race-Condition
- Warum überhaupt mehr als einen Thread nutzen?

Zusammenhänge

- Welche Zustände haben Threads?
- Wie vermeidet Synchronisation Konflikte?
- Wie erzeuge ich einen Thread?

Beispiele

- Wie synchronisiere ich mehrere Threads?
- Wie greifen Threads auf Ressourcen zu?

Begriffe

Programm

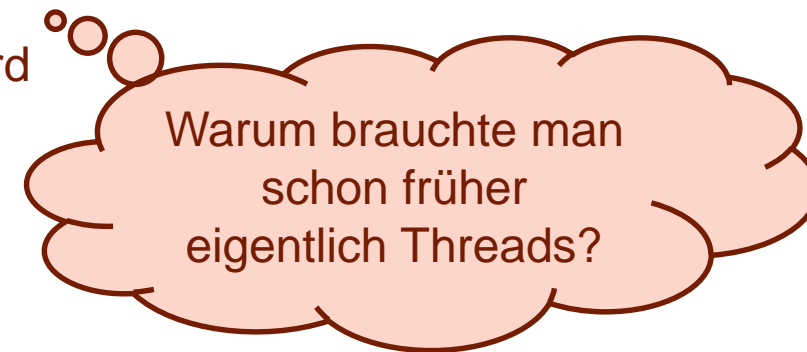
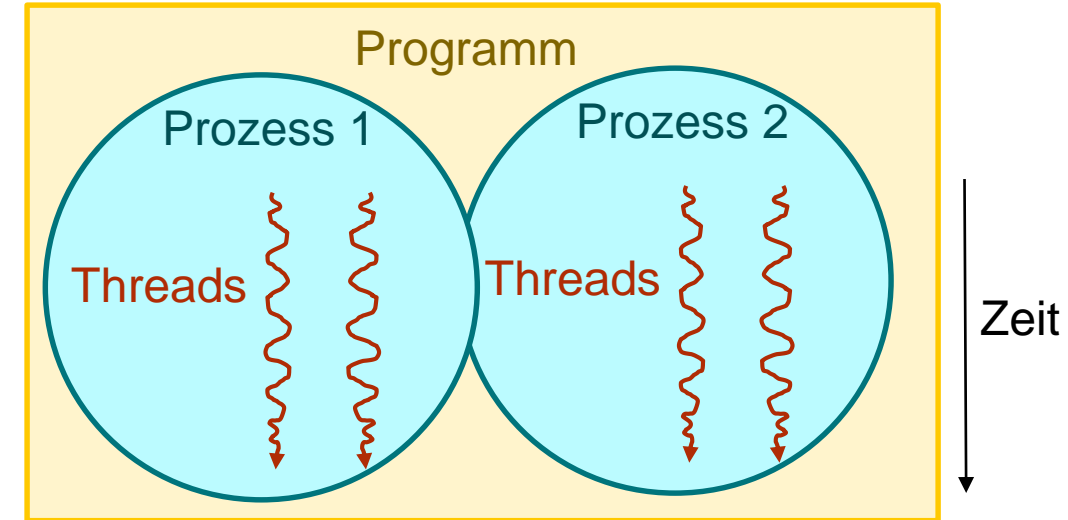
- Besteht aus einem oder mehreren Prozessen

Prozess

- Besteht aus mindestens einem Thread (Main-Thread)
- Kann auf Hardware-Ressourcen zugreifen mithilfe des Betriebssystems

Thread

- Kleinste Sequenz von Anweisungen, die von einem Scheduler verwaltet wird
- Kann auf Speicher / gemeinsame Ressourcen seines Prozess zugreifen
- Schnelles Umschalten zwischen Threads möglich



Früher...

MediaMarkt

8 MB ARBEITS-SPEICHER

1,6 GIGA-BYTE FESTPLATTE

PIONEER QUAD SPEED CD-ROM-LAUFWERK

Abb. ähnlich

2799.-

BARPREIS/OHNE MONITOR

150 MHz

inklusive Software

COREL DRAW! 4.0

Lotus SMARTSUITE 3.0

bestehend aus:

Lotus 1-2-3 5.0	Tabellenkalkulation
Lotus Ami Pro 3.1	Textverarbeitung
Lotus Freelance 2.1	Präsentationsgrafik
Lotus Organizer 1.1	Terminplaner
Lotus Approach 3.0	Datenbank

(In Deutsch, Englisch und Französisch)

32 BIT

INKLUSIVE

Microsoft® Windows® 95

Günstige Finanzierung für den Computer

durch unsere Partnerbank

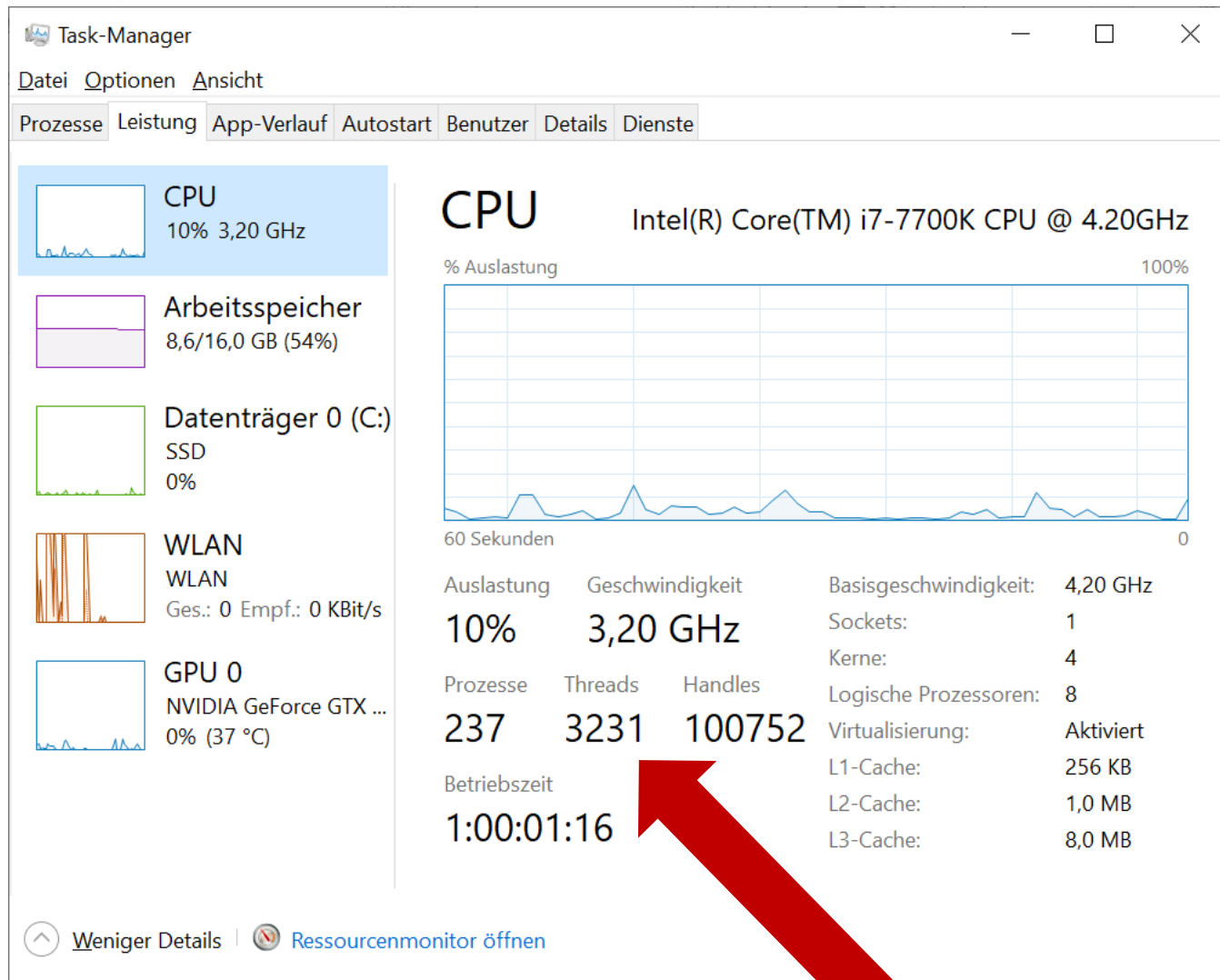
72 Monatsraten zu je 52,74 DM

effektiver Jahreszins 11,2%

n@twork

P-150-CD2D-S

Intel Pentium® Prozessor, 150 MHz,
2 MB RAM Grafikkarte



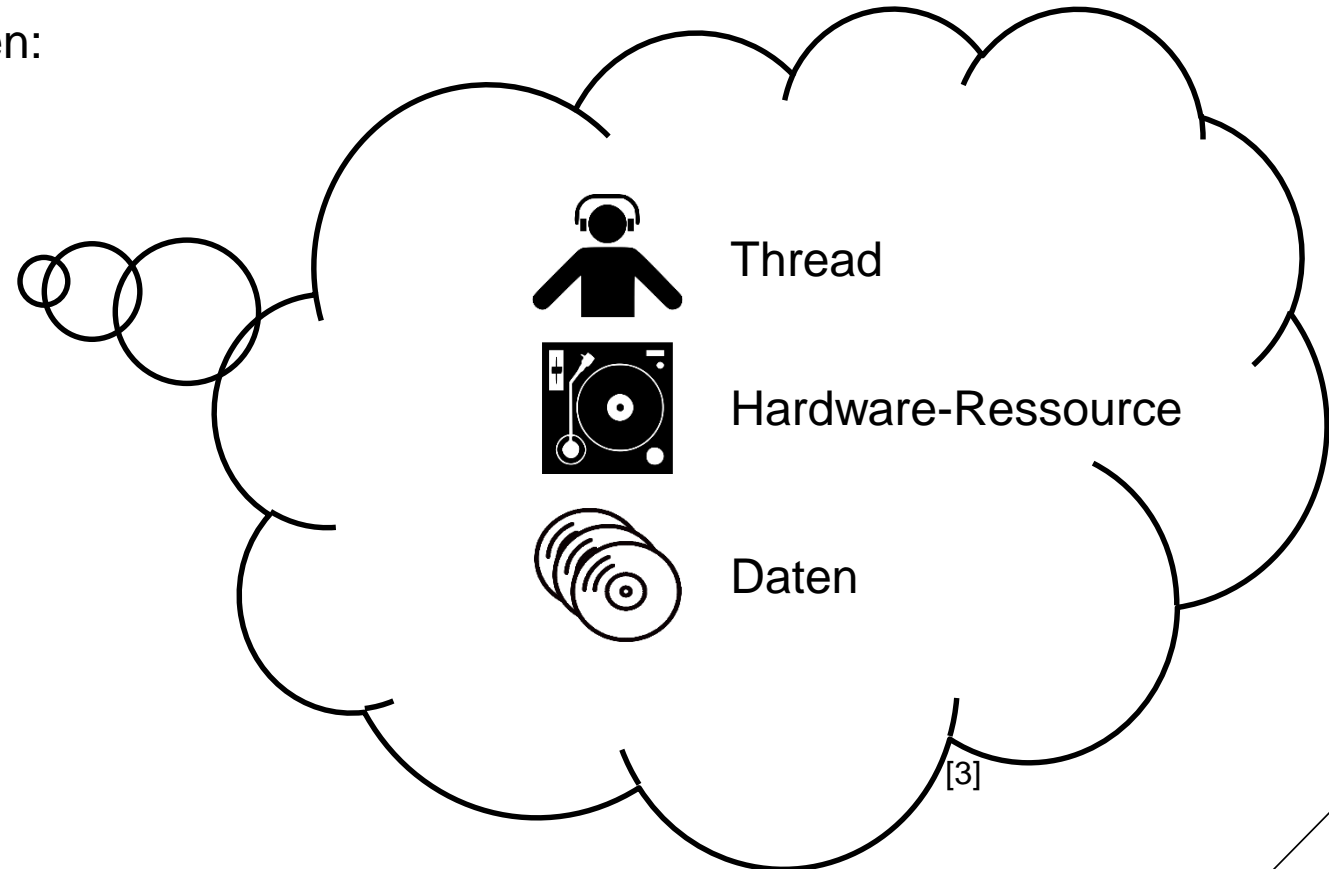
13x mehr Threads als Prozesse – Warum?

Warum Threads bei Single-Cores?

- Aufteilung der Programmaufgaben nach Thema
- Verschiedene Prioritäten
- Kanalisierung nach Hardware-Ressourcen:
 - UI/3D
 - Konsole
 - Datei-Ein-/Ausgabe
 - Audiowiedergabe
 - ...

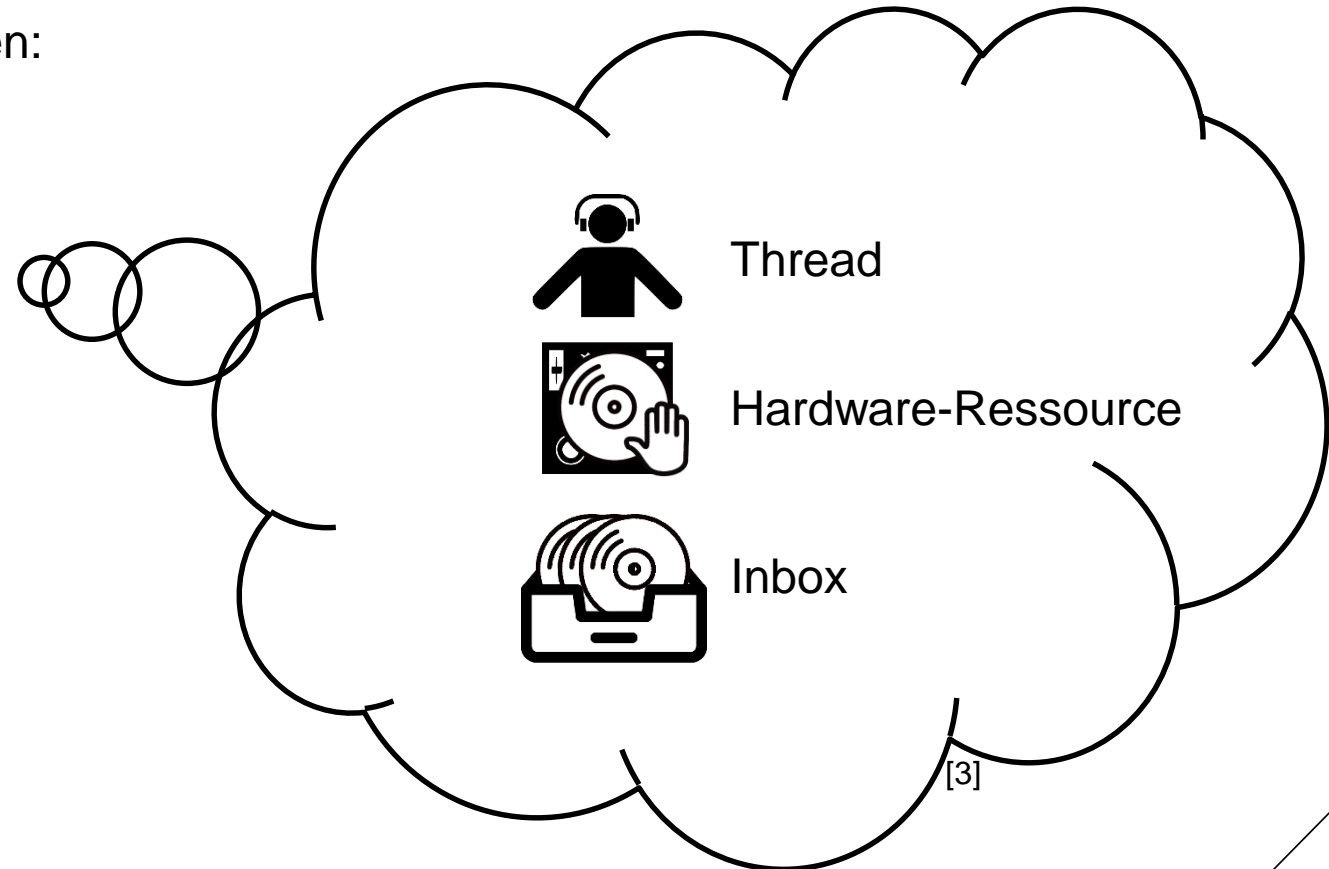
Warum Threads bei Single-Cores?

- Aufteilung der Programmaufgaben nach Thema
- Verschiedene Prioritäten
- Kanalisierung nach Hardware-Ressourcen:
 - UI/3D
 - Konsole
 - Datei-Ein-/Ausgabe
 - Audiowiedergabe
 - ...

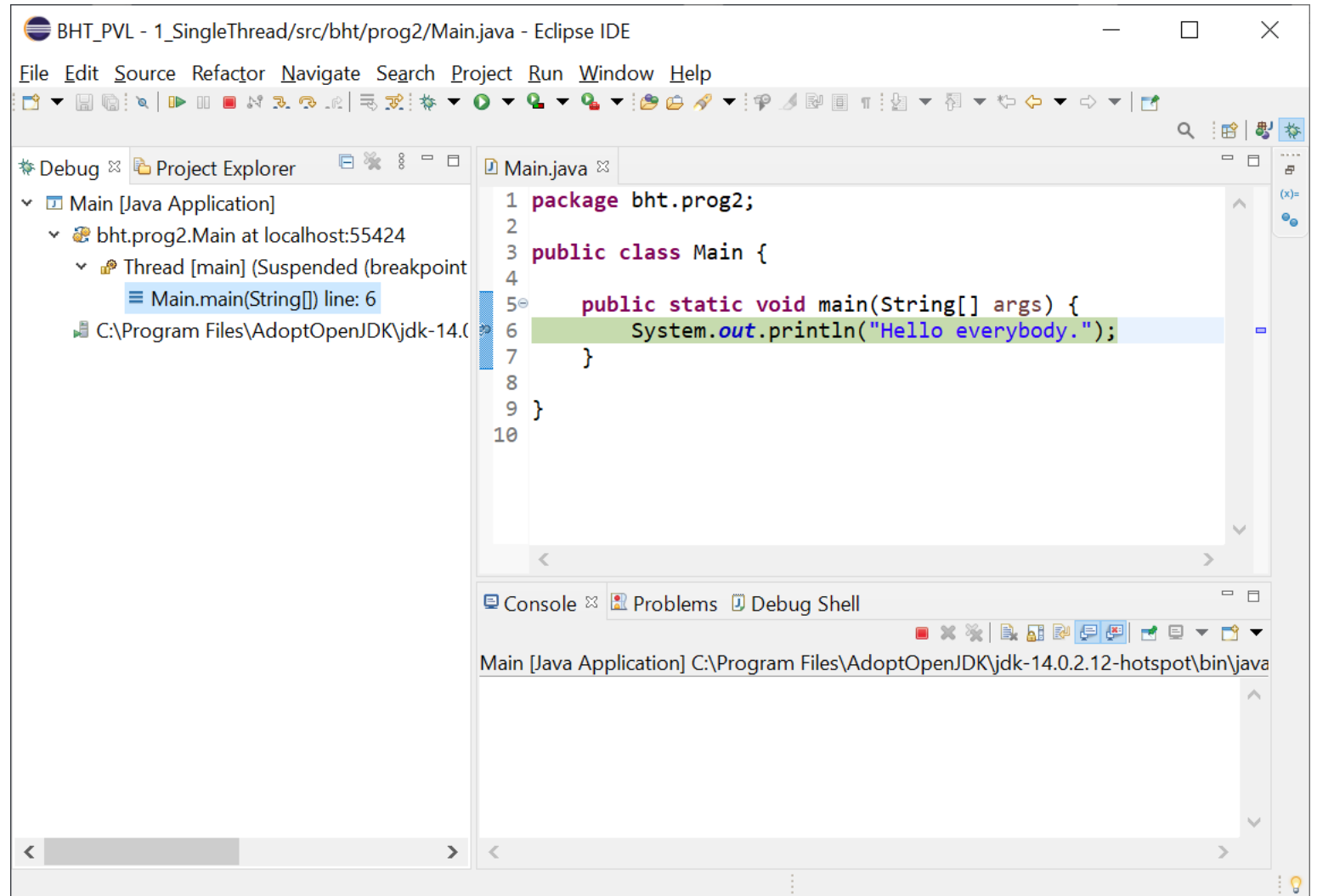


Warum Threads bei Single-Cores?

- Aufteilung der Programmaufgaben nach Thema
- Verschiedene Prioritäten
- Kanalisierung nach Hardware-Ressourcen:
 - UI/3D
 - Konsole
 - Datei-Ein-/Ausgabe
 - Audiowiedergabe
 - ...

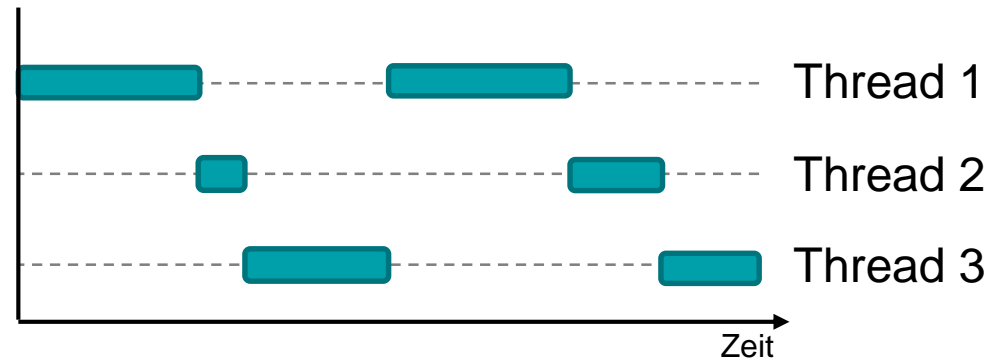


- Was ist der Main-Thread?
- Was ist der Debugger?
- Was ist der Stack?



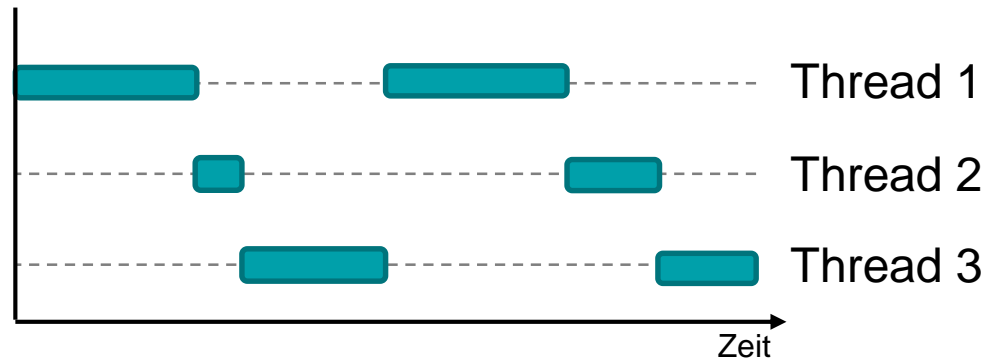
Scheduler

- Teil des Betriebssystems
- Kann Threads unterbrechen
- Verteilt Threads auf vorhandene CPU-Kerne
- Abarbeitung erfolgt oft verschränkt auf einem Kern

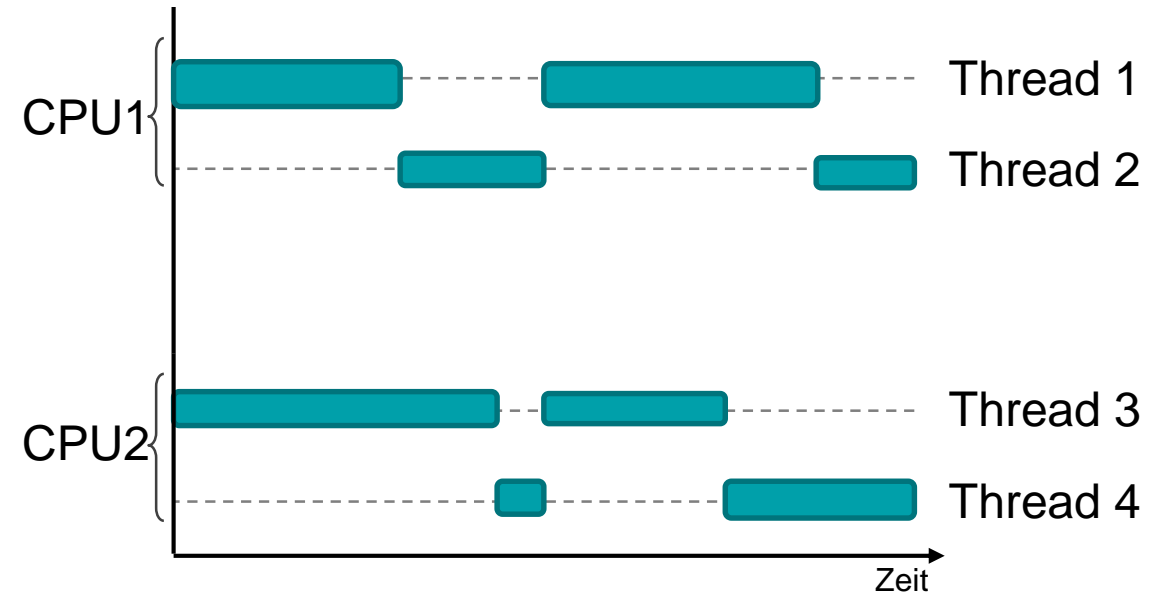


Scheduler

- Teil des Betriebssystems
- Kann Threads unterbrechen
- Verteilt Threads auf vorhandene CPU-Kerne
- Abarbeitung erfolgt oft verschränkt auf einem Kern



Nebenläufigkeit
Threads laufen abwechselnd
auf selben Kern



Parallelität
Threads laufen gleichzeitig
auf verschiedenen Kernen

Thread als Java-Klasse

Thread

```
void run()  
void start()  
void stop()  
void interrupt()  
void isInterrupted()  
void join()
```

Durch Überschreiben implementiert der Thread seine Aufgabe.

Startet den Thread im Betriebssystem und ruft `run()` auf.

Unsicher. Freundliches Beenden durch `interrupt()` soll verwendet werden.

Setzt das Bit `interrupted` auf wahr.

Liefert Wert von `interrupted`. Setzt `interrupted` zurück auf falsch.

Wartet bis der Thread seine `run()`-Methode beendet hat.

- Wie implementiere ich einen Thread?
- Wie starte ich einen Thread?
- Was ändert sich im Debugger?

BHT_PVL - 3_MaxPerformanceAsOneThread/src/bht/prog2/MainThread.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Debug Project Explorer

MainThread (2) [Java Application]

- ▼ bht.prog2.MainThread at localhost:55456
 - ▼ Thread [main] (Suspended (breakpoint at MainThread.main(String[])) line: 8
 - ▼ Thread [CounterThread] (Suspended (breakpoint at CounterThread.run()) line: 19

C:\Program Files\AdoptOpenJDK\jdk-14.0.2\bin\java.exe

```
3 public class MainThread {
4     public static void main(String[] args) {
5         System.out.println("Main started");
6         Thread counterThread = new CounterThread();
7         counterThread.start();
8         System.out.println("Main end");
9     }
10 }
11
12 class CounterThread extends Thread {
13     public CounterThread() {
14         super("CounterThread");
15     }
16
17     @Override
18     public void run() {
19         System.out.println("CounterThread start");
20         for (int i = 0; i < Integer.MAX_VALUE; i++) {
21             System.out.println("i=" + i);
22         }
23         System.out.println("CounterThread end");
24     }
25 }
```

Console Problems Debug Shell

MainThread (2) [Java Application]

Main started

Writable Smart Insert 8:1

Schritt für Schritt

Main-Thread

```
public static void main(String[] args) {  
    System.out.println("Main started");  
  
    Thread counterThread = new CounterThread();  
    counterThread.start();  
  
    System.out.println("Main end");  
}
```

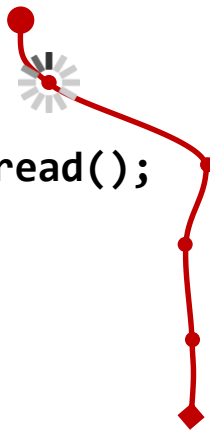
Counter-Thread

```
public void run() {  
    System.out.println("CounterThread start");  
  
    for (int i = 0; i < Integer.MAX_VALUE; i++) {  
        System.out.println("i=" + i);  
    }  
  
    System.out.println("CounterThread end");  
}
```

Schritt für Schritt

Main-Thread

```
public static void main(String[] args) {  
    System.out.println("Main started");  
  
    Thread counterThread = new CounterThread();  
    counterThread.start();  
  
    System.out.println("Main end");  
}
```



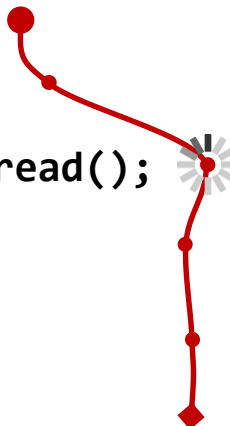
Counter-Thread

```
public void run() {  
    System.out.println("CounterThread start");  
  
    for (int i = 0; i < Integer.MAX_VALUE; i++) {  
        System.out.println("i=" + i);  
    }  
  
    System.out.println("CounterThread end");  
}
```

Schritt für Schritt

Main-Thread

```
public static void main(String[] args) {  
    System.out.println("Main started");  
  
    Thread counterThread = new CounterThread();  
    counterThread.start();  
  
    System.out.println("Main end");  
}
```



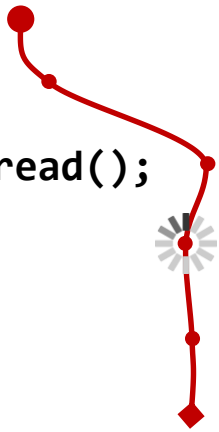
Counter-Thread

```
public void run() {  
    System.out.println("CounterThread start");  
  
    for (int i = 0; i < Integer.MAX_VALUE; i++) {  
        System.out.println("i=" + i);  
    }  
  
    System.out.println("CounterThread end");  
}
```

Schritt für Schritt

Main-Thread

```
public static void main(String[] args) {  
    System.out.println("Main started");  
  
    Thread counterThread = new CounterThread();  
    counterThread.start();  
  
    System.out.println("Main end");  
}
```



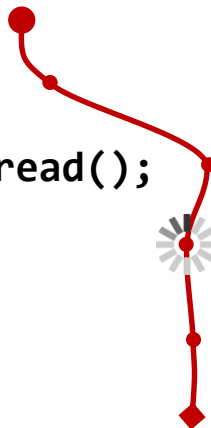
Counter-Thread

```
public void run() {  
    System.out.println("CounterThread start");  
  
    for (int i = 0; i < Integer.MAX_VALUE; i++) {  
        System.out.println("i=" + i);  
    }  
  
    System.out.println("CounterThread end");  
}
```

Schritt für Schritt

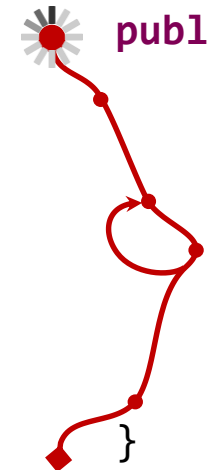
Main-Thread

```
public static void main(String[] args) {  
    System.out.println("Main started");  
  
    Thread counterThread = new CounterThread();  
    counterThread.start();  
  
    System.out.println("Main end");  
}
```



Counter-Thread

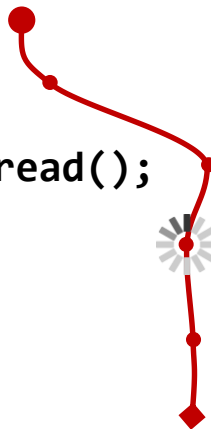
```
public void run() {  
    System.out.println("CounterThread start");  
  
    for (int i = 0; i < Integer.MAX_VALUE; i++) {  
        System.out.println("i=" + i);  
    }  
  
    System.out.println("CounterThread end");  
}
```



Schritt für Schritt


Main-Thread

```
public static void main(String[] args) {  
    System.out.println("Main started");  
  
    Thread counterThread = new CounterThread();  
    counterThread.start();  
  
    System.out.println("Main end");  
}
```



Counter-Thread

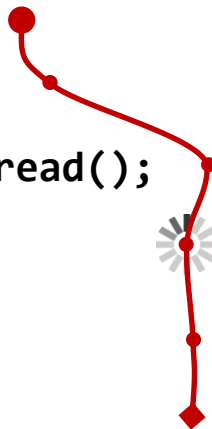
```
public void run() {  
    System.out.println("CounterThread start");  
  
    for (int i = 0; i < Integer.MAX_VALUE; i++) {  
        System.out.println("i=" + i);  
    }  
  
    System.out.println("CounterThread end");  
}
```



Schritt für Schritt


Main-Thread

```
public static void main(String[] args) {  
    System.out.println("Main started");  
  
    Thread counterThread = new CounterThread();  
    counterThread.start();  
  
    System.out.println("Main end");  
}
```



Counter-Thread

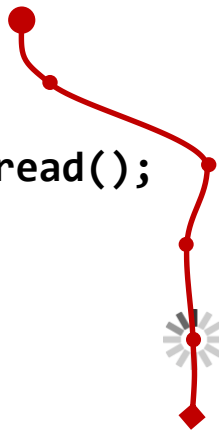
```
public void run() {  
    System.out.println("CounterThread start");  
  
    for (int i = 0; i < Integer.MAX_VALUE; i++) {  
        System.out.println("i=" + i);  
    }  
  
    System.out.println("CounterThread end");  
}
```



Schritt für Schritt


Main-Thread

```
public static void main(String[] args) {  
    System.out.println("Main started");  
  
    Thread counterThread = new CounterThread();  
    counterThread.start();  
  
    System.out.println("Main end");  
}
```



Counter-Thread

```
public void run() {  
    System.out.println("CounterThread start");  
  
    for (int i = 0; i < Integer.MAX_VALUE; i++) {  
        System.out.println("i=" + i);  
    }  
  
    System.out.println("CounterThread end");  
}
```



Schritt für Schritt

Main-Thread

```
public static void main(String[] args) {  
    System.out.println("Main started");  
  
    Thread counterThread = new CounterThread();  
    counterThread.start();  
  
    System.out.println("Main end");  
}
```

Counter-Thread

```
public void run() {  
    System.out.println("CounterThread start");  
  
    for (int i = 0; i < Integer.MAX_VALUE; i++) {  
        System.out.println("i=" + i);  
    }  
  
    System.out.println("CounterThread end");  
}
```

Wer gewinnt das Rennen?



→ Hier egal, da es keinen Einfluss auf das Ergebnis hat!

Schritt für Schritt

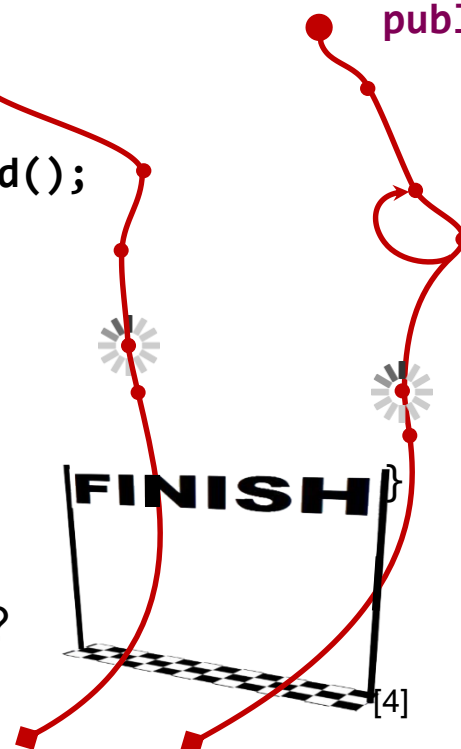
Main-Thread

```
public static void main(String[] args) {  
    System.out.println("Main started");  
  
    Thread counterThread = new CounterThread();  
    counterThread.start();  
  
    System.out.println("Main end");  
    ERGEBNIS = 1;  
}
```

Counter-Thread

```
public void run() {  
    System.out.println("CounterThread start");  
  
    for (int i = 0; i < Integer.MAX_VALUE; i++) {  
        System.out.println("i=" + i);  
    }  
  
    System.out.println("CounterThread end");  
    ERGEBNIS = 2;  
}
```

Wer gewinnt das Rennen?



Race Condition

Situation, bei der mehrere Threads auf gemeinsame Daten lesend oder schreibend zugreifen, sodass deren Konsistenz von der Thread-Reihenfolge abhängt

- Wie sieht eine Race-Condition aus?
- Wie kann ein Thread warten mit `sleep()`?
- Wie synchronisiere ich Threads mit `join()`?

BHT_PVL - 4_RaceCondition/src/bht/prog2/Main.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Debug Project Explorer

Main (2) [Java Application]

- bht.prog2.Main at localhost:55492
 - Thread [main] (Suspended (breakpoint at line 14) Main.main(String[])) line: 14
 - Thread [A] (Suspended (breakpoint at line 25) Main\$PlusOneThread.run()) line: 25
 - Thread [B] (Suspended (breakpoint at line 25) Main\$PlusOneThread.run()) line: 25

C:\Program Files\AdoptOpenJDK\jdk-14.0.2.12-

```
3 public class Main {
4     static int ERGEBNIS = 0;
5
6     public static void main(String[] args) throws Exception {
7         System.out.println("start");
8
9         PlusOneThread threadA = new PlusOneThread("A");
10        PlusOneThread threadB = new PlusOneThread("B");
11        threadA.start();
12        threadB.start();
13
14        System.out.println("end");
15    }
16
17    static class PlusOneThread extends Thread {
18        PlusOneThread(String name) {
19            super(name);
20        }
21
22        @Override
23        public void run() {
24            ERGEBNIS++;
25            System.out.println(getName() + ": " + ERGEBNIS);
26        }
27    }
28 }
```

Console Problems Debug Shell

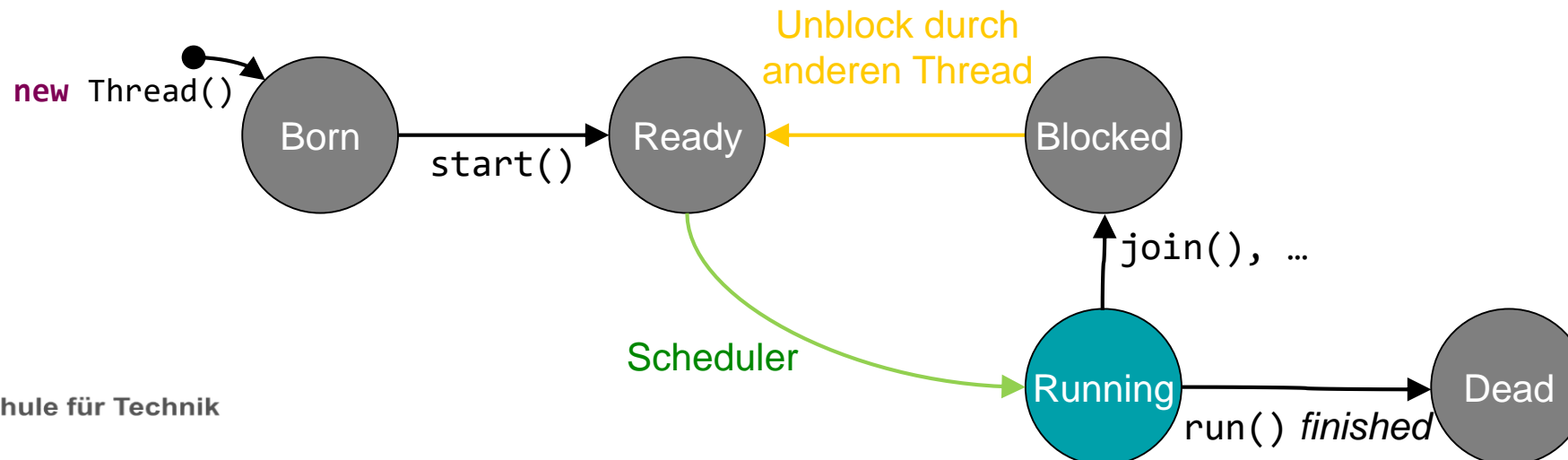
Main (2) [Java Application] C:\Program Files\AdoptOpenJDK\jdk-14.0.2.12-hotspot\bin\javaw.exe

start

Writable Smart Insert 14 : 1 : 310

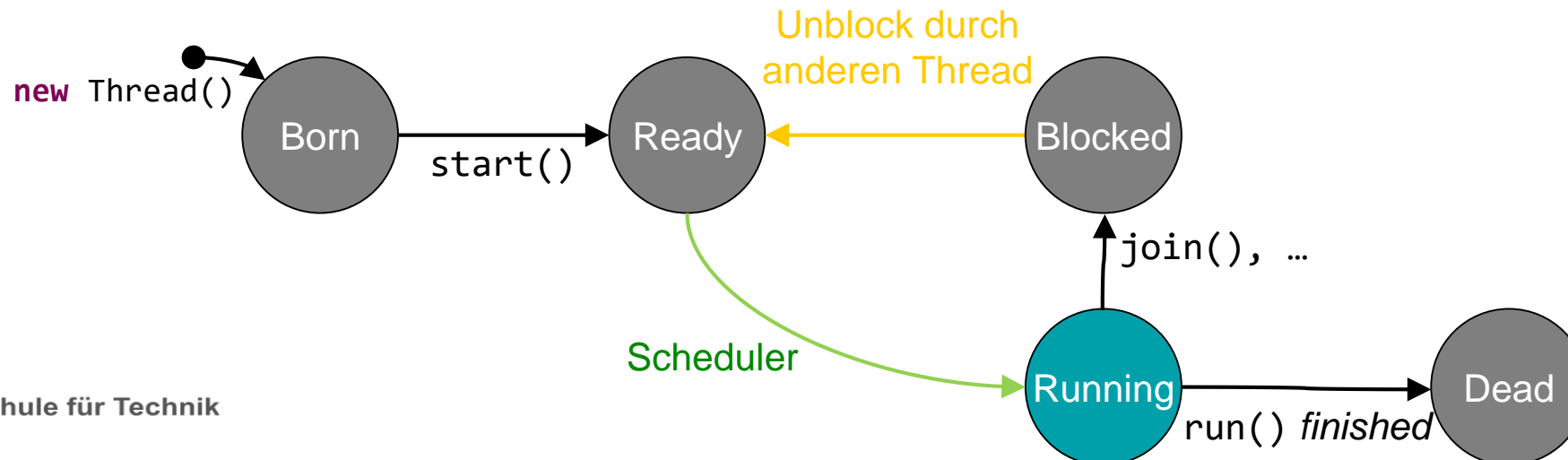
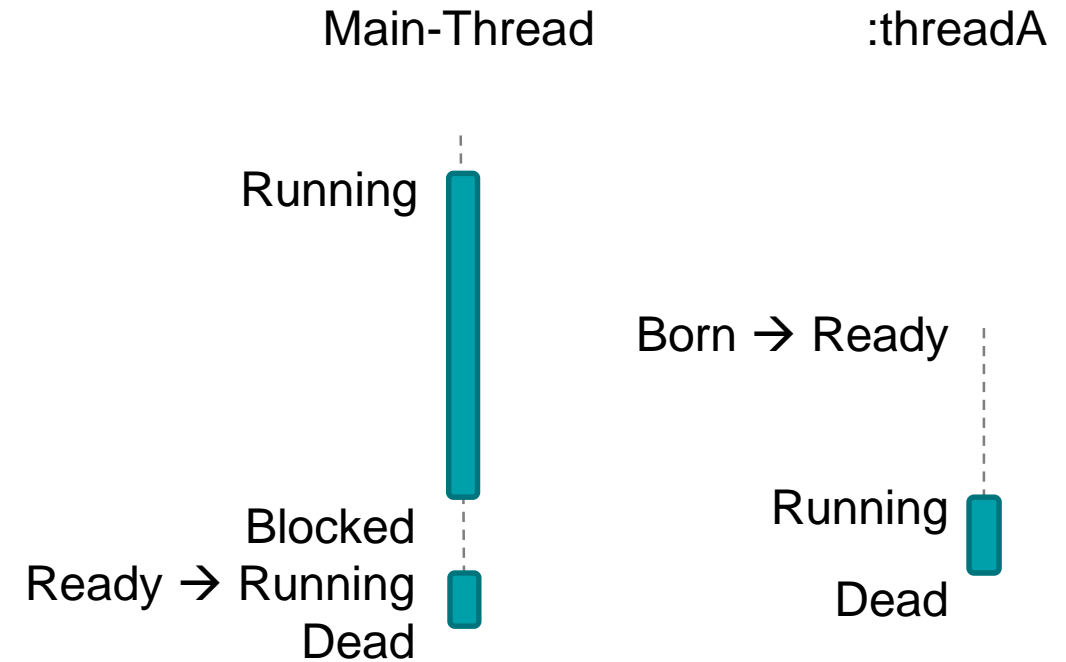
Thread-Status

```
public static void main(String[] args) throws Exception {  
    System.out.println("start");  
  
    PlusOneThread threadA = new PlusOneThread("A");  
  
    threadA.start();  
  
    threadA.join();  
  
    System.out.println("end");  
}
```



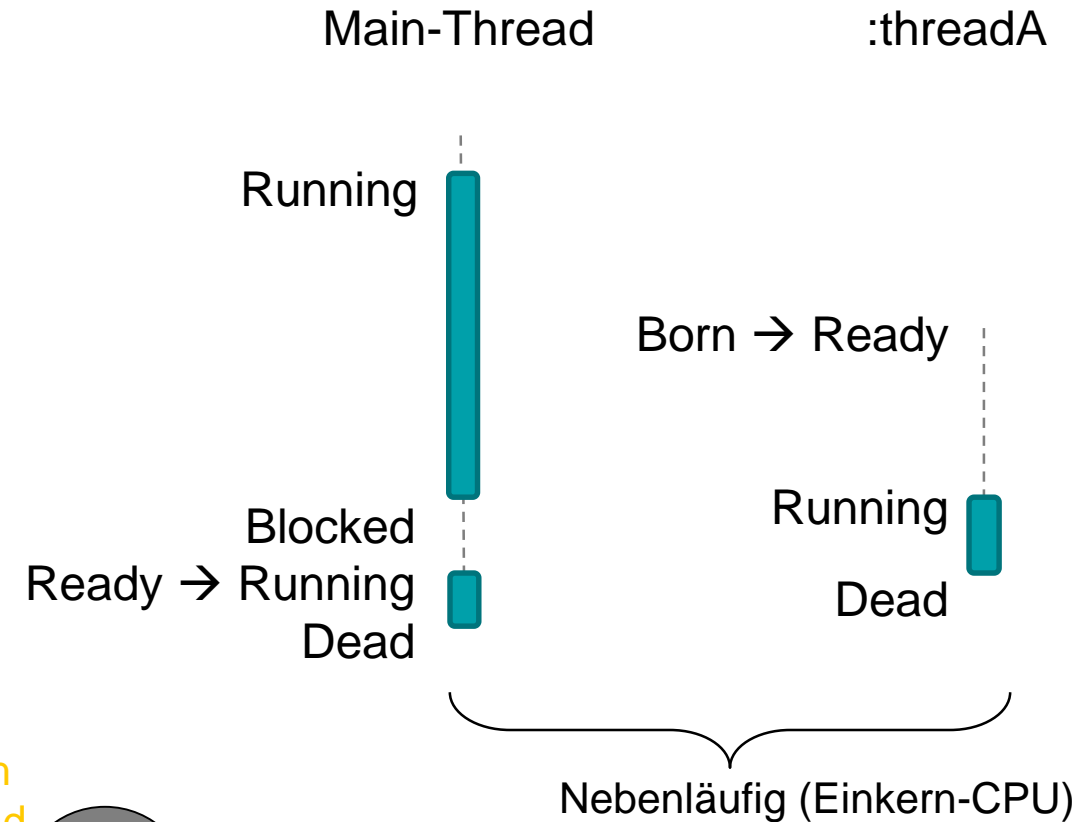
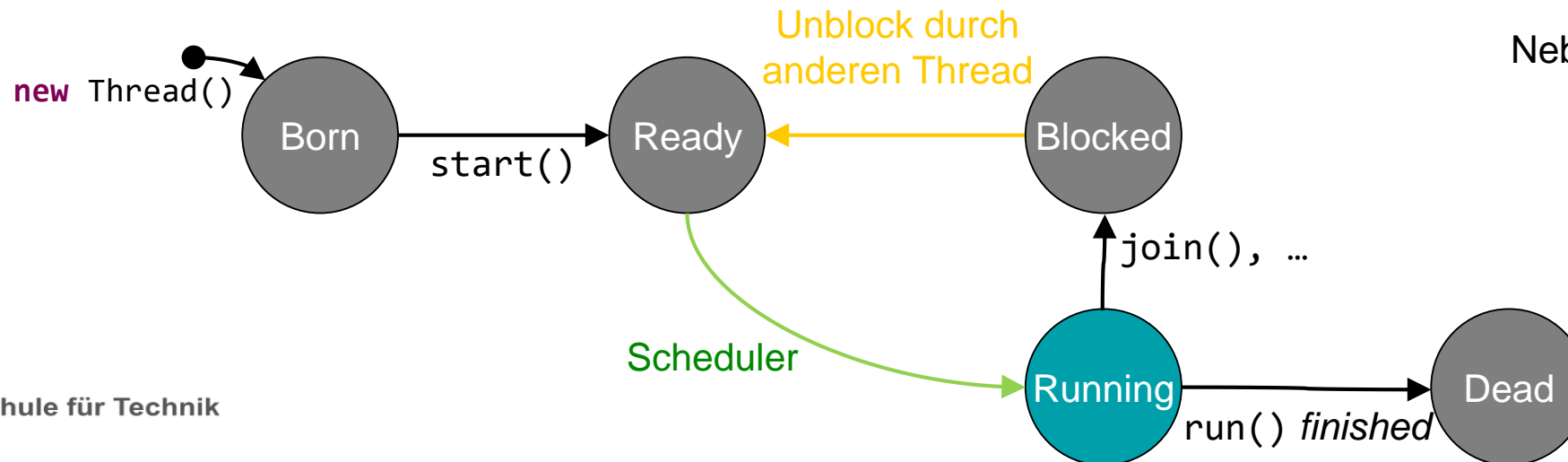
Thread-Status

```
public static void main(String[] args) throws Exception {  
    System.out.println("start");  
  
    PlusOneThread threadA = new PlusOneThread("A");  
  
    threadA.start();  
  
    threadA.join();  
  
    System.out.println("end");  
}
```



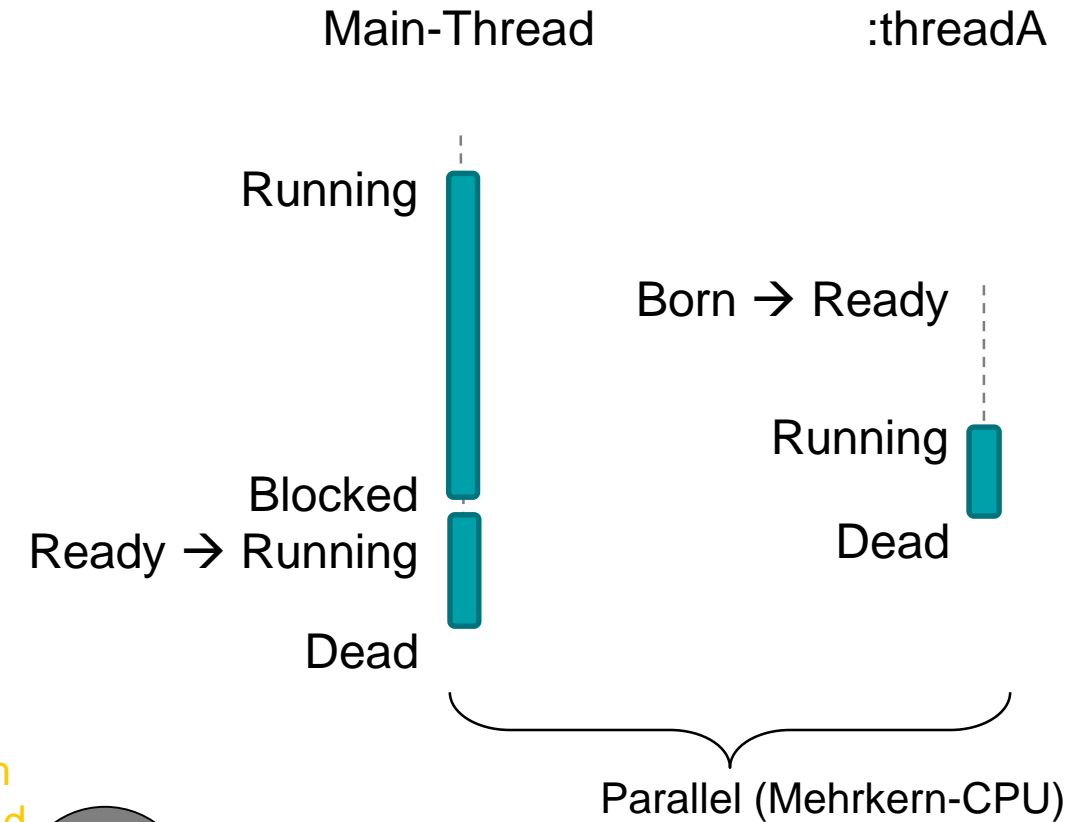
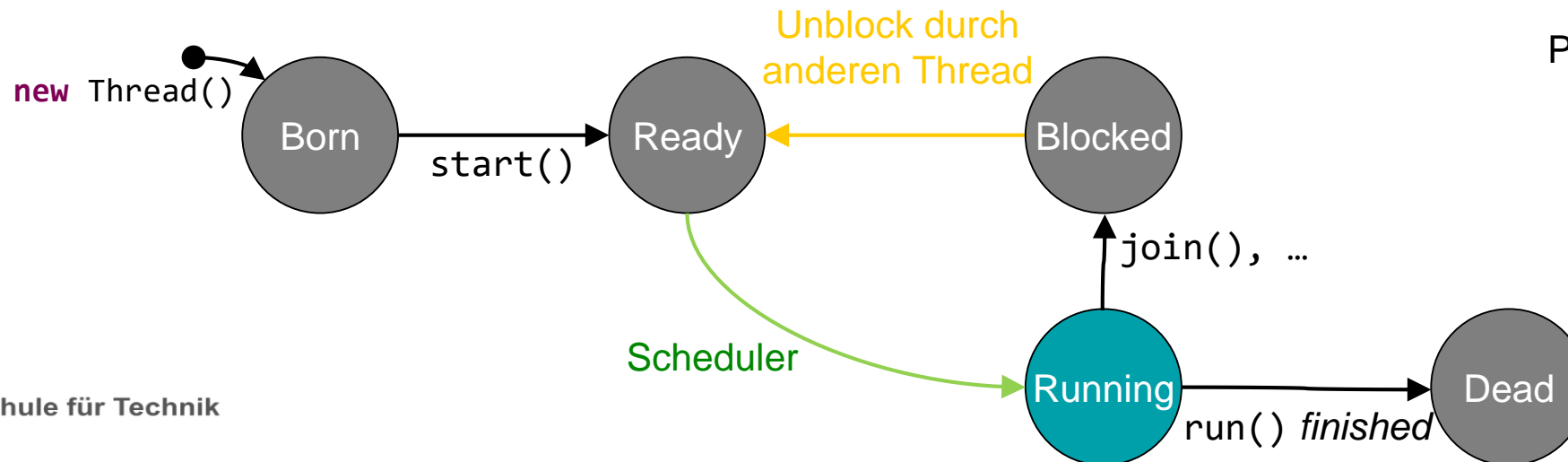
Thread-Status

```
public static void main(String[] args) throws Exception {  
    System.out.println("start");  
  
    PlusOneThread threadA = new PlusOneThread("A");  
  
    threadA.start();  
  
    threadA.join();  
  
    System.out.println("end");  
}
```



Thread-Status

```
public static void main(String[] args) throws Exception {  
    System.out.println("start");  
  
    PlusOneThread threadA = new PlusOneThread("A");  
  
    threadA.start();  
  
    threadA.join();  
  
    System.out.println("end");  
}
```



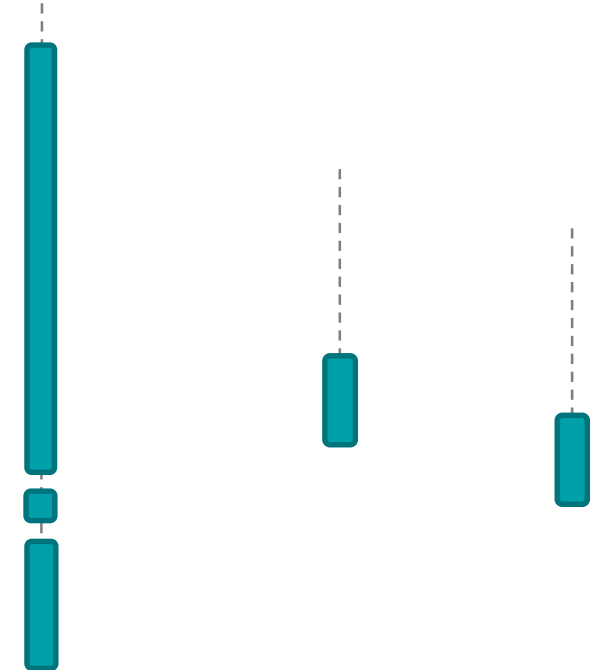
Thread-Synchronisation

```
public static void main(String[] args) throws Exception {  
    System.out.println("start");  
  
    PlusOneThread threadA = new PlusOneThread("A");  
    PlusOneThread threadB = new PlusOneThread("B");  
  
    threadA.start();  
    threadB.start();  
  
    threadA.join();  
    threadB.join();  
  
    System.out.println("end");  
}
```

} main_1()

} main_2()

Main-Thread :threadA :threadB



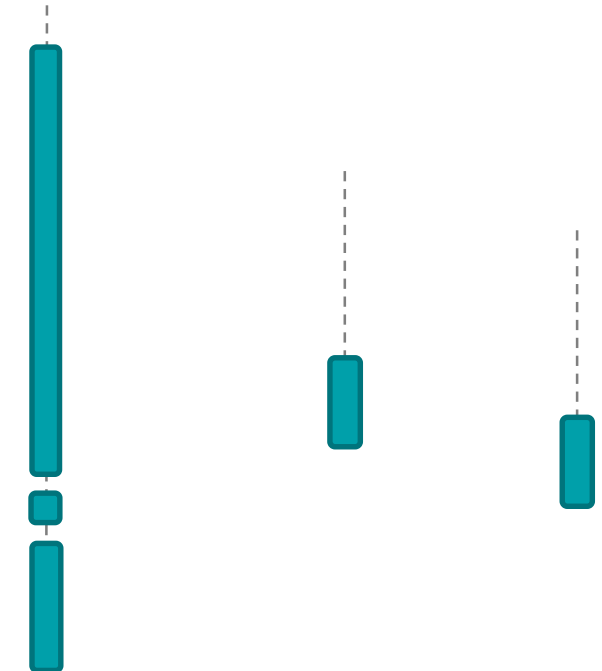
Thread-Synchronisation

```
public static void main(String[] args) throws Exception {  
    System.out.println("start");  
  
    PlusOneThread threadA = new PlusOneThread("A");  
    PlusOneThread threadB = new PlusOneThread("B");  
  
    threadA.start();  
    threadB.start();  
  
    threadA.join();  
    threadB.join();  
  
    System.out.println("end");  
}
```

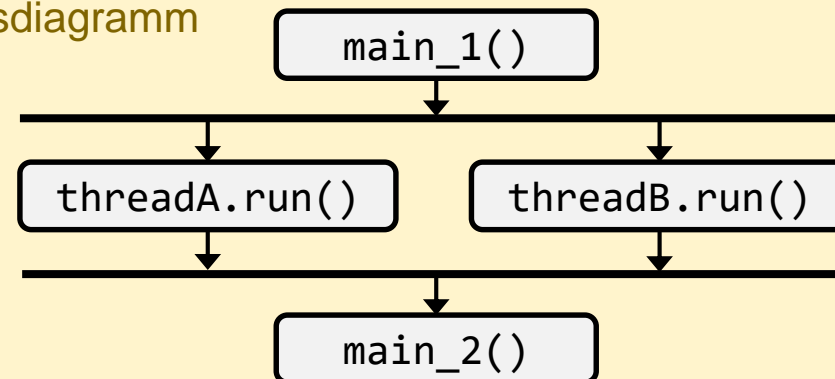
main_1()

main_2()

Main-Thread :threadA :threadB



UML-Aktivitätsdiagramm



Multi-Threading

Vorteile

- Höhere Verarbeitungsgeschwindigkeit durch Aufteilung und Parallelisierung
- Themen-/Hardware-spezifische Programmierung durch Threads
- Priorisierung von Berechnungen

Nachteile

- Zusammenspiel verschiedener Threads oft zufällig (nicht-deterministisch)
- Neue Arten von Fehlern: Race-Conditions, ...
- Zugriffe auf Speicher und Hardware muss synchronisiert werden
- Synchronisierung von Threads kann sehr komplex sein

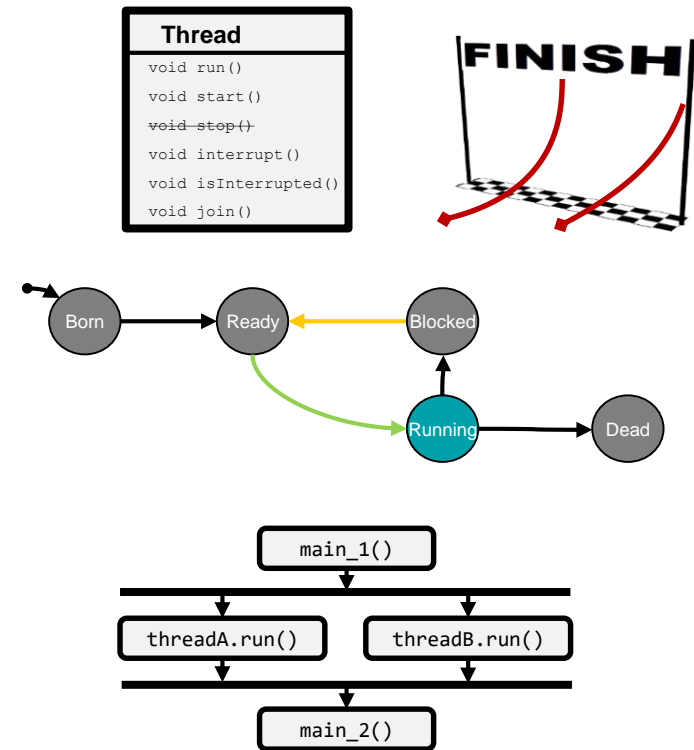
Rekapitulation

Jetzt kennen wir Thread-

- Befehle: `start()`, `sleep()`
- Konflikte: Race-Conditions
- Synchronisation mithilfe von `join()`
- Zustände: Born, Ready, Running, Blocked, Dead
- Darstellung im UML-Aktivitätsdiagramm

Weiter geht es mit

- Wie können wir Threads von außen beenden? → `interrupted()`
- Wie wartet ein Thread auf Ereignisse? → `wait()`, `notify()`
- Wie können mehrere Threads einzeln nacheinander auf Ressourcen zugreifen? → `synchronized`



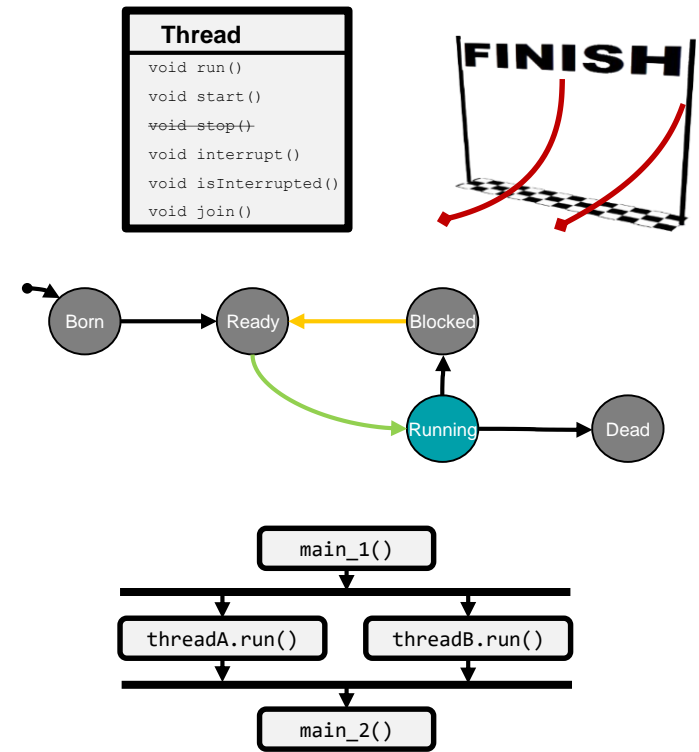
Rekapitulation

Jetzt kennen wir Thread-

- Befehle: `start()`, `sleep()`
- Konflikte: Race-Conditions
- Synchronisation mithilfe von `join()`
- Zustände: Born, Ready, Running, Blocked, Dead
- Darstellung im UML-Aktivitätsdiagramm

Weiter geht es mit

- Wie können wir Threads von außen beenden? → `interrupted()`
- Wie wartet ein Thread auf Ereignisse? → `wait()`, `notify()`
- Wie können mehrere Threads einzeln nacheinander auf Ressourcen zugreifen? → `synchronized`



- Warum muss ein Thread beendet werden?
- Wie beende ich einen Thread mit `interrupt()`?
- Warum soll ein Thread nicht mit `stop()` beendet werden?

The screenshot shows the Eclipse IDE interface. The top toolbar includes icons for File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The Project Explorer on the left shows a project named 'bht_plv' with a sub-project '5_TerminateThread' containing a package 'src/bht/prog2' with a class 'Main.java'. The Main [Java Application] view shows the 'bht.prog2.Main' at localhost:49728, with a breakpoint at 'Main.main(String[]) line: 9' and another at 'CounterThread.run() line: 20'. The main editor displays the source code of 'Main.java' and 'CounterThread.java'. The 'Main' class has a 'main' method that creates a 'CounterThread' and calls 'interrupt()'. The 'CounterThread' class overrides 'run()' and prints 'CounterThread interrupted' when interrupted. The Console view at the bottom shows the output: 'Main started', 'CounterThread start', and 'i=0'.

```
3 public class Main {
4     public static void main(String[] args) throws Exception {
5         System.out.println("Main started");
6         Thread counterThread = new CounterThread();
7         counterThread.start();
8         counterThread.interrupt();
9         System.out.println("Main end");
10    }
11 }
12
13 class CounterThread extends Thread {
14     @Override
15     public void run() {
16         System.out.println("CounterThread start");
17         for (int i = 0; i < Integer.MAX_VALUE; i++) {
18             System.out.println("i=" + i);
19             if (interrupted()) {
20                 System.out.println("CounterThread interrupted");
21                 return;
22             }
23         }
24         System.out.println("CounterThread end");
25     }
26 }
```

Console Output:

```
Main [Java Application] C:\Program Files\AdoptOpenJDK\jdk-14.0.2.12-hotspot\bin\javaw.exe (18.04)
Main started
CounterThread start
i=0
```

Literaturempfehlungen

- Goll, Joachim; Heinisch, Cornelia: *Java als erste Programmiersprache. Grundkurs für Hochschulen*. 8. Auflage. Springer Vieweg, 2016. ISBN 978–3–658–12118–1
- Hettel, Jörg; Tran, Manh T.: *Nebenläufige Programmierung mit Java*. dpunkt, 2016. ISBN 978–3–96088–012–7
- Louis, Dirk; Müller, Peter: *Java. Eine Einführung in die Programmierung*. 2. Auflage. Hanser, 2018. ISBN 978–3–446–45362–3
- Ullenboom, Christian: *Java ist auch eine Insel. Einführung, Ausbildung, Praxis*. 12. Auflage. Rheinwerk Verlag, openbook.rheinwerk-verlag.de/javainsel/

Abbildungen

- [1] enccs.github.io/OpenACC/0.01_gpu-introduction/ (abgerufen 12.04.2023)
- [2] tutonaut.de/alte-werbung-was-computer-frueher-gekostet-haben (abgerufen 10.04.2023)
- [3] icon-library.com/icon/icon-dj-1.html.html (abgerufen 15.04.2023)
- [4] pngkey.com/maxpic/u2w7e6t4t4u2a9t4/ (abgerufen 10.04.2023)