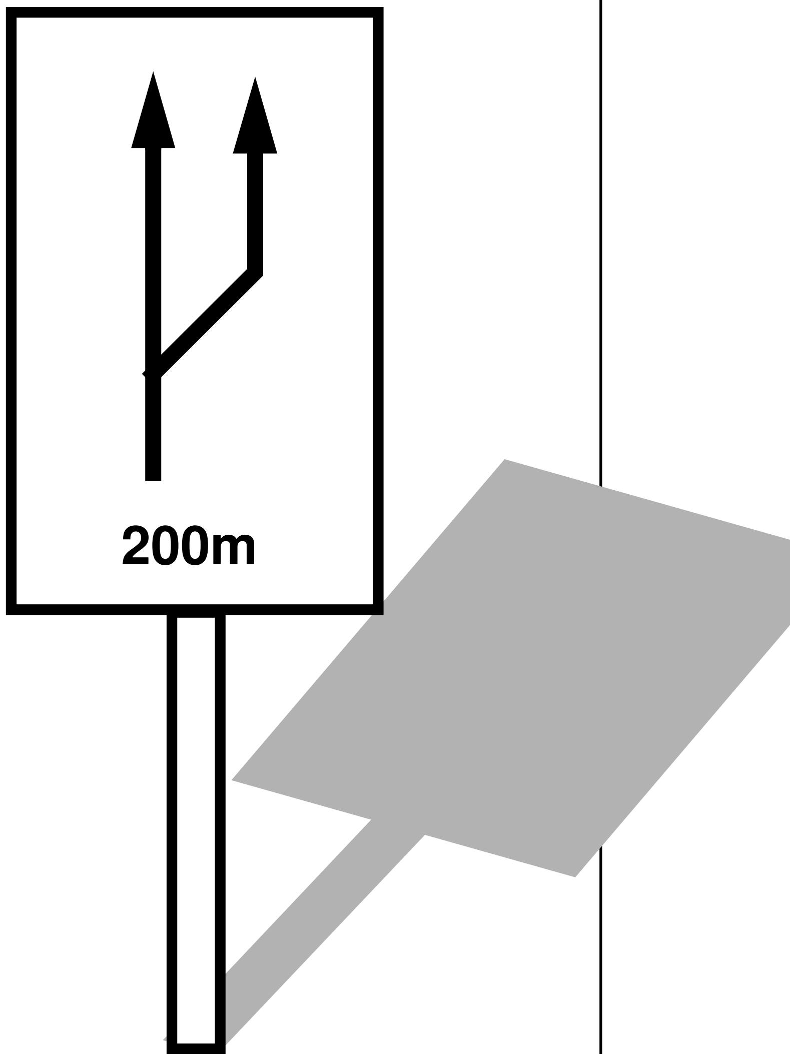
Java Threads

Grundlagen

Threads in Java

Thread-Synchronisation



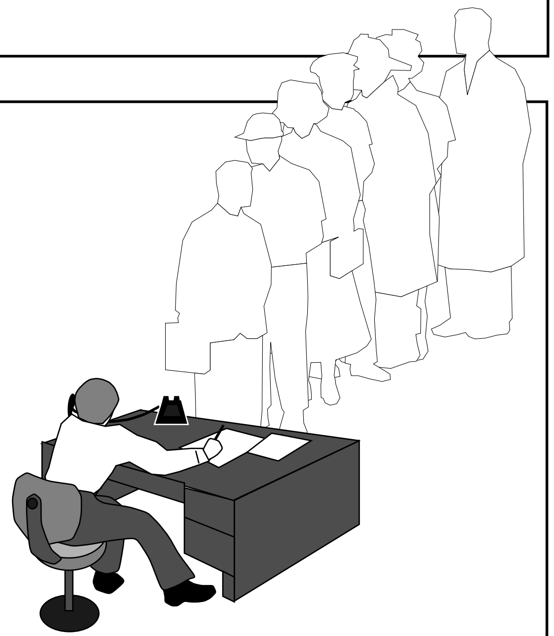
Grundlagen

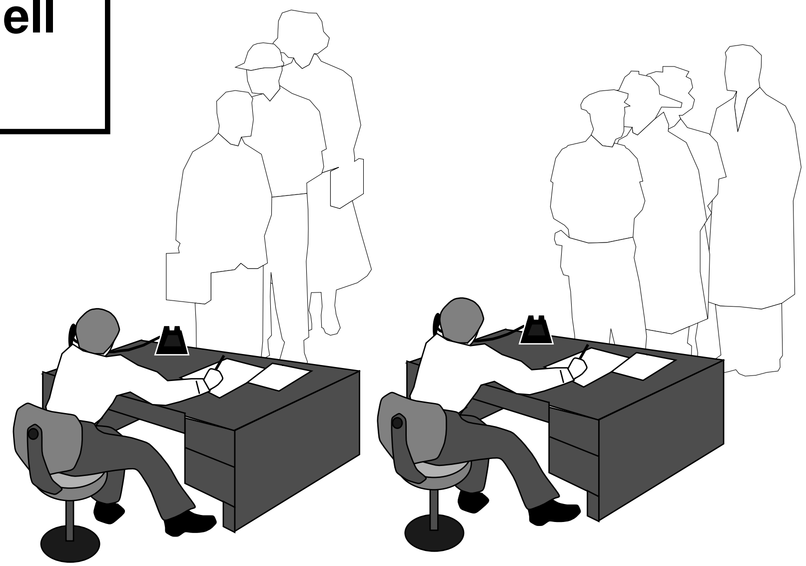
traditionell:

- **sequentielle Programme**
- **nur ein Kontrollfluß (Thread)**

wünschenswert:

- **Programmteile parallel abarbeiten**
- **mehrere Threads**
- **Abarbeitung in einem Thread sequentiell**

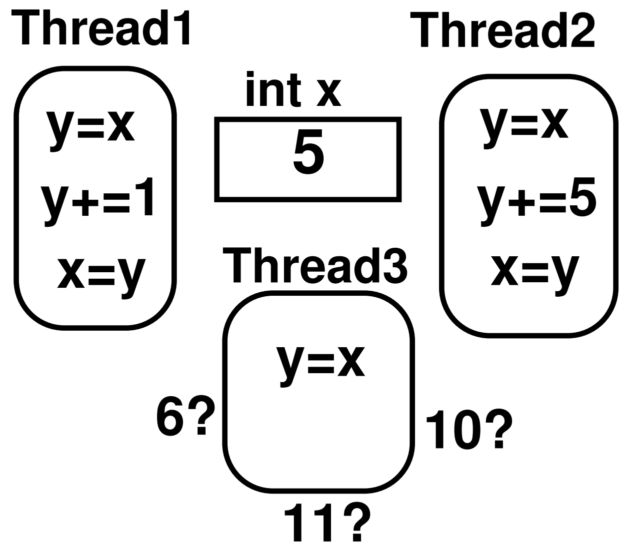




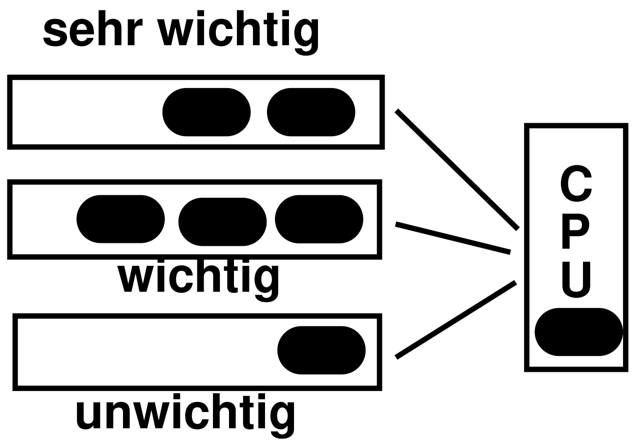
Grundlagen

Threads:

- **laufen im selben Adreßraum ab (leichtgewichtige Prozesse)**
  - **müssen sich synchronisieren**

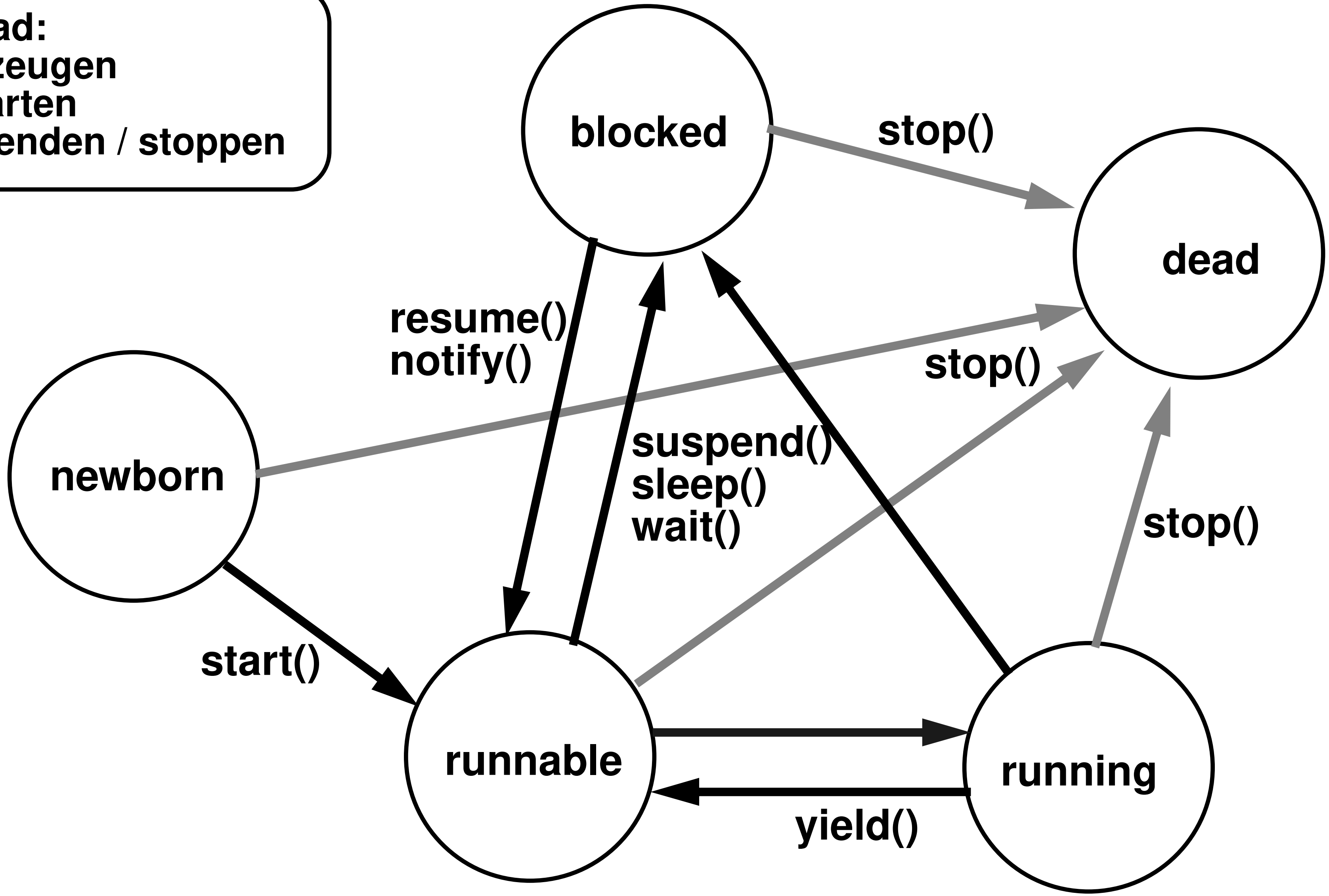


- **müssen sich eine CPU Teilen**
  - **besitzen Prioritäten**
  - **besitzen verschiedene Zustände**



Grundlagen

Thread:
1. erzeugen
2. starten
3. beenden / stoppen



Klasse java.lang.Thread

generischer Name
Thread-<nummer>

Identifikation

Konstruktoren:

- public **Thread**()
- public **Thread**(String **name**)
- public **Thread**(Runnable target)
- public **Thread**(Runnable target, String name)

Methoden:

- void **run**()
- void **start**()
- **static** void **sleep**(long millis) **static** void **yield**()
- void **join**()
- void **setName**() String **getName**()
- void **setPriority**(int prio) int **getPriority**()
- boolean **isAlive**()

von Object ererbt:

- void **wait**() void **notify**()

Threads Synchronisieren

Zweck:

- gegenseitiger Ausschluß
- vermeiden gleichzeitigen Objektzugriffs (aus mehreren Threads)

Lösung: Objekt-Monitore

Monitor:

- verwaltet Zugriffe auf ein Objekt
- immer nur ein Zugriff möglich
- zwei Zustände: frei --- vergeben

Monitor anlegen:

- Methoden synchronized deklarieren
- synchronized(Object) { ....}
  verwenden

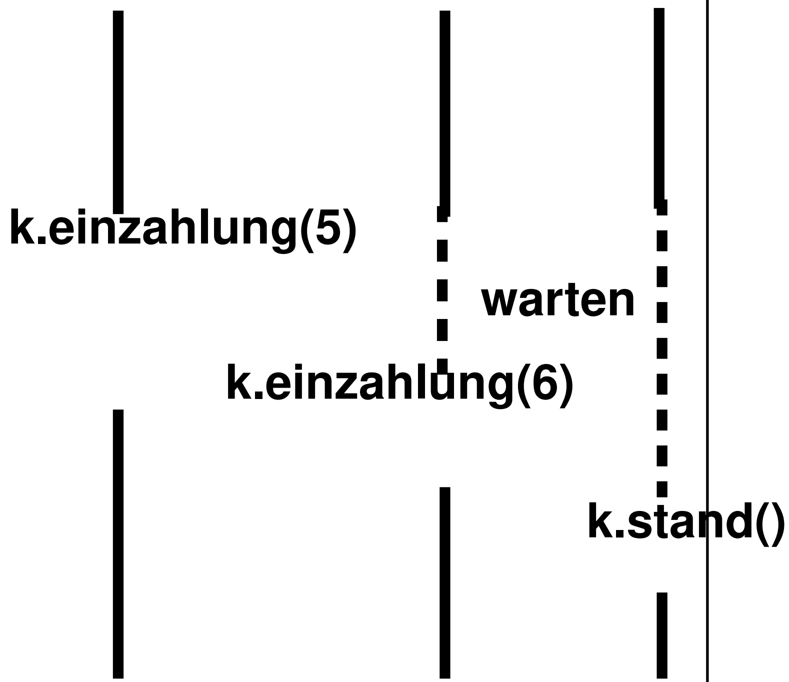
Keine andere synchronized Methode ausführbar

critical regions

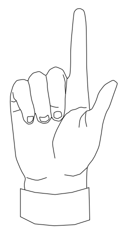
```
synchronized(x){
    y=x;
    y += 1;
    x=y;
}
```

Beispiel

```
class Konto{
    privat double kontostand;

    public Konto (double x){
        kontostand = x;
    }

    public synchronized double stand(){
        return kontostand;
    }

    public synchronized void einzahlung(double x){
        kontostand += x;
    }
```



nächster Monitorbesitzer
ist zufällig



Zusammenfassung

- Threads ermöglichen paralleles Arbeiten
- Ablauf ist nichtdeterministisch
- bei Zugriff auf gemeinsame Daten: Synchronisation
- Synchronisation durch Objektmonitore
- synchronized Objektmethoden
- synchronized-Anweisungsblöcke
- Synchronisationsfehler sehr schwer zu finden