

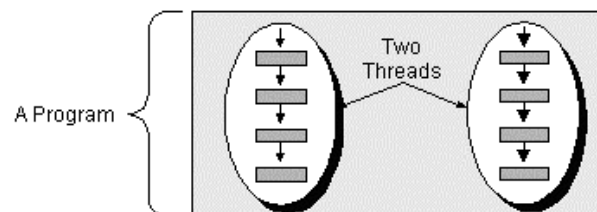
# Threads: Basics

Michael Hahsler

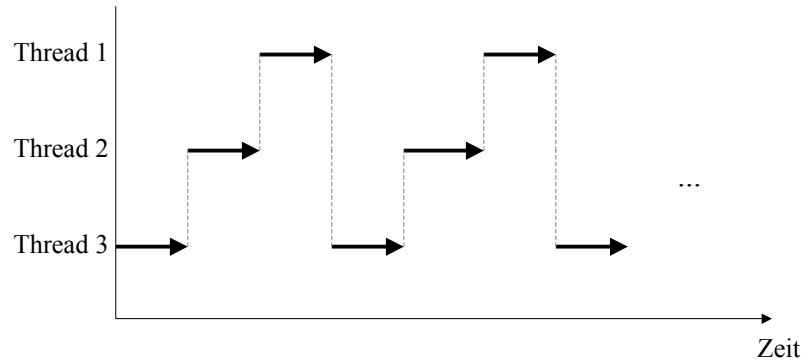
siehe: <http://java.sun.com/docs/books/tutorial/essential/threads>

## Was sind Threads

- Moderne Rechner können parallel mehrere Prozesse (Programme) ausführen (multi processing).
- Auch Java kann ein Programm gleichzeitig mehrere Kontrollflüsse ausführen (= Threads oder lightweight processes).



## Ablauf von Prozessen/Threads

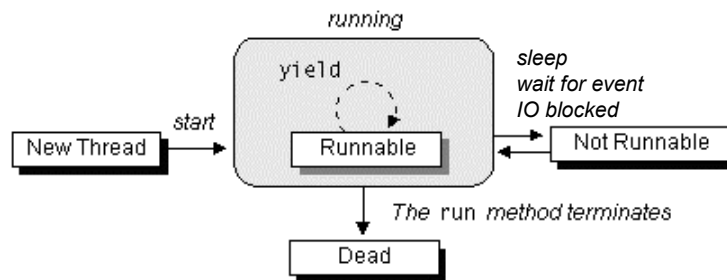


Zwischen den Threads wird umgeschaltet. Dadurch sieht es aus, als ob alle Threads/Prozesse gleichzeitig ablaufen!

## Implementierung in Java

- In Java muss dazu die Klasse Thread erweitert werden (`extends Thread`).
- Die `run`-Methode von Thread wird mit dem Code überschrieben:
  - `public void run()`
- Weitere wichtige Methoden (siehe `java.lang.Thread`):
  - `public void start()`
  - `public static void sleep(long millis) throws InterruptedException`
  - `public final void setPriority(int newPriority)`

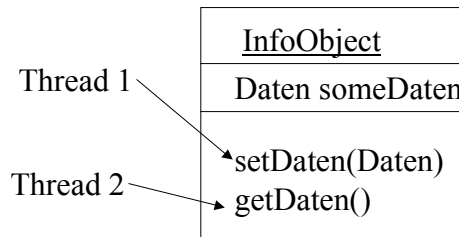
## Lebenszyklus eines Threads



## Priority Scheduling

- Java entscheidet anhand der Priorität (integer Zahl, größer ist besser) welcher Thread als nächstes ausgeführt wird.
- kann mit `setPriority(int)` verändert werden.

# Synchronizing



# Synchronizing

- Manchmal teilen sich Threads Informationen und es ist notwendig, dass ein Thread wartet, bis der Andere fertig ist (Transaktionen).
- In diesem Fall können Methoden in einer Klasse als `synchronized` definiert werden. Java sorgt dann dafür, dass nur ein Thread mit den Methoden arbeiten kann (setzt ein Lock auf die die Instanz). Die anderen Threads warten.

```
synchronized void setDaten (Daten sD) { ... }  
synchronized Daten setDaten () { ... }
```

## Wait und Notify

- kann ein Thread nicht weiterarbeiten, da z.B. für `getData()` stehen die Daten noch nicht vollständig zur Verfügung, kann er durch `wait()` schlafen geschickt werden. Dadurch wird das Lock aufgehoben und ein anderer Thread kann die Daten vervollständigen (mit `setData()`). Am Schluss weckt `setData()` den 1. Thread mit `notify()` wieder auf.
- `wait()` und `notify()` werden von `Object` geerbt.

## Beispiel Synchronizing

- siehe Beispiel: `SynchroDemo`