

# Kapitel 2

## Prozesse und Java-Threads

Prof. Dr. Rolf Hennicker

27.04.2017

## 2.1 Prozessbegriff

### **Prozess:**

Programm in Ausführung

### **Prozesszustand** (zu einem Zeitpunkt):

Wird charakterisiert durch die Werte von

- ▶ expliziten Variablen (vom Programmierer deklariert)
- ▶ impliziten Variablen (Befehlszähler, organisatorische Daten)

### **Zustandsübergang** (eines Prozesses):

Wird von einer Aktion bewirkt. Aktionen sind elementar, d.h. nicht unterbrechbar.

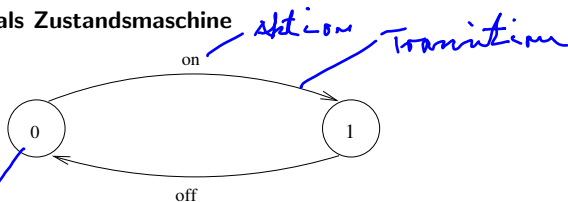
### **Bemerkung:**

Im Folgenden abstrahieren wir von den konkreten Zustandsdarstellungen, d.h. von den konkreten Werten der expliziten und impliziten Variablen.

Wir interessieren uns dafür, welche Aktionen in einem Prozesszustand als nächstes möglich sind.

## 2.2 Modellierung durch endliche Zustandsmaschinen

Beispiel: Lichtschalter als Zustandsmaschine



**Grafische Darstellung mit dem Tool LTSA** (Labelled Transition System Analyser):  
Zustände werden von 0 beginnend durchnummeriert.

0 ist der Anfangszustand.

**Ablauf:** on off on off on off ...

*unend. Ablauf*

Ein **Ablauf** ist eine Aktionsfolge, die ein Prozess ausführen kann, die entweder unendlich ist oder in einem Zustand endet, in dem der Prozess nicht fortgesetzt werden kann.

**Beachte:**

*überges.*  
↑

- ▶ Wir betrachten nur Prozesse mit endlich vielen Zuständen und endlicher Menge von Aktionen.
- ▶ Das Verhalten eines Prozesses kann aber unendlich sein (nicht terminierend).

Die hier betrachteten Zustandsmaschinen sind formal *endliche markierte Transitionssysteme* ("Labelled Transition Systems"), abgekürzt LTS.

**Definition:**

Sei  $\text{States}$  eine universelle, abzählbar unendliche Menge von Zuständen und  $\text{ACT}$  eine universelle, abzählbar unendliche Menge von (sichtbaren) Aktionen. Ein endliches LTS ist ein Quadrupel

$$(S, A, \Delta, q_0),$$

wobei

- ▶  $S \subseteq \text{States}$  eine endliche Menge von Zuständen ist,
- ▶  $A \subseteq \text{ACT}$  eine endliche Menge von Aktionen ist,
- ▶  $\Delta \subseteq S \times A \times S$  eine Übergangsrelation ist,
- ▶  $q_0 \in S$  ein Anfangszustand ist.

**Notation:** Für  $(q, a, p) \in \Delta$  schreiben wir auch  $q \xrightarrow{a}_{\Delta} p$

**Beispiel: Lichtschalter (formal)**

Das LTS  $(S, A, \Delta, q_0)$  besteht aus

$$S = \{0, 1\}$$

$$A = \{\text{on}, \text{off}\}$$

$$\Delta = \{(0, \text{on}, 1), (1, \text{off}, 0)\}$$

$$q_0 = 0$$

## 2.3 Prozessausdrücke

Prozesse werden kompakt beschrieben durch Ausdrücke der Sprache FSP (Finite State Processes) [Magee, Kramer].

FSP ist eine Variante einer "Prozessalgebra".

FSP orientiert sich

- ▶ syntaktisch an CSP [Hoare] (Communicating Sequential Processes)
- ▶ semantisch an CCS [Milner] (Calculus of Communicating Systems)

Die *Semantik* eines Prozessausdrucks  $E$  wird durch Übersetzung in ein LTS gegeben.

- ▶ Im Folgenden werden FSP-Prozessausdrücke induktiv (über deren strukturellen Aufbau) definiert.
- ▶ Dabei wird jedem Prozessausdruck  $E$  eine Menge von freien Variablen  $FV(E)$  zugeordnet. Die Variablen stellen Prozessidentifikatoren dar.

## 1. Konstante Prozessausdrücke und 2. Prozessidentifikatoren

Sei PID eine universelle, abzählbar unendliche Menge von Prozessidentifikatoren (Bezeichnern).

### Definition:

1. STOP ist ein (konstanter) Prozessausdruck mit  $FV(\text{STOP}) = \emptyset$ .
2. Jeder Prozessidentifikator  $P \in \text{PID}$  ist ein Prozessausdruck mit  $FV(P) = \{P\}$ .

### Wirkung:

1. STOP bezeichnet den Prozess, der keine Aktion ausführen kann.
2. Die Wirkung von  $P \in \text{PID}$  kann nur im Zusammenhang mit einer Prozessdeklaration "P = E." beschrieben werden (vgl. unten).

### 3. Aktionspräfix

**Definition:**

Ist  $a \in \text{ACT}$  eine Aktion und  $E$  ein Prozessausdruck, dann ist das *Aktionspräfix*  $(a \rightarrow E)$  ebenfalls ein Prozessausdruck mit  $\text{FV}((a \rightarrow E)) = \text{FV}(E)$ .

Statt von Prozessausdrücken sprechen wir häufig kurz von „Prozessen“.

**Wirkung:**

Der Prozess  $(a \rightarrow E)$  engagiert sich zunächst in die Aktion  $a$  und verhält sich dann wie  $E$ .

**Beispiele:**

$(\text{eat} \rightarrow \text{STOP})$

$(\text{eat} \rightarrow (\text{drink} \rightarrow \text{STOP}))$



## 4. Auswahl

### Definition:

Sind  $a_1, \dots, a_n$  Aktionen und  $E_1, \dots, E_n$  Prozessausdrücke mit  $n \geq 2$ , dann ist  $(a_1 \rightarrow E_1 \mid \dots \mid a_n \rightarrow E_n)$  ein Prozessausdruck mit  $FV((a_1 \rightarrow E_1 \mid \dots \mid a_n \rightarrow E_n)) = FV(E_1) \cup \dots \cup FV(E_n)$ .

### Wirkung:

Der Prozess engagiert sich entweder

- ▶ in  $a_1$  und verhält sich danach wie  $E_1$  oder
- ▶ in  $a_2$  und verhält sich danach wie  $E_2$  oder
- ⋮
- ▶ in  $a_n$  und verhält sich danach wie  $E_n$ .

### Beispiel:

$(\text{eat} \rightarrow \text{STOP} \mid \text{drink} \rightarrow \text{STOP})$

Kurznotation:  $(\{\text{eat}, \text{drink}\} \rightarrow \text{STOP})$

## 5. Prozessausdrücke mit Rekursion

Werden später bei der Definition der Semantik von Prozessidentifikatoren  $P$  im Kontext einer rekursiven Prozessdeklaration " $P = E$ ." verwendet.

### **Definition:**

Sei  $P$  ein Prozessidentifikator und  $E$  ein Prozessausdruck, so dass  $P \in FV(E)$ .  
Dann ist  $\text{rec}(P = E)$  ein Prozessausdruck mit  $FV(\text{rec}(P = E)) = FV(E) \setminus \{P\}$ .

### **Beispiel:**

$\text{rec}(H = (\text{eat} \rightarrow H))$

## (Rekursive) Prozessdeklarationen

### Definition:

Ist  $P$  ein Prozessidentifikator und  $E$  ein Prozessausdruck, dann ist

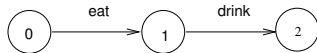
$$P = E.$$

eine *Prozessdeklaration*. Die Deklaration ist *rekursiv*, wenn  $P$  in dem Ausdruck  $E$  frei vorkommt, d.h.  $P \in FV(E)$ .

### Beispiele:

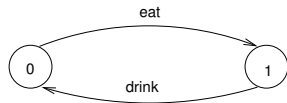
1.  $PERS = (\text{eat} \rightarrow \text{drink} \rightarrow \text{STOP})$ .

Das LTS von PERS ist gegeben durch das LTS des Prozessausdrucks  $(\text{eat} \rightarrow \text{drink} \rightarrow \text{STOP})$ .



2.  $PERSON = (\text{eat} \rightarrow \text{drink} \rightarrow PERSON)$ .

Das LTS von PERSON ist gegeben durch das LTS des Prozessausdrucks  $\text{rec}(PERSON = (\text{eat} \rightarrow \text{drink} \rightarrow PERSON))$ .



Äquivalente Prozessbeschreibung mit **lokalen** Prozessdeklarationen:

PERSON = EATING,

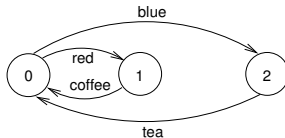
EATING = (eat  $\rightarrow$  DRINKING),

DRINKING = (drink  $\rightarrow$  PERSON).

**Beispiel** (Getränkeautomat):

DRINKS = (red  $\rightarrow$  coffee  $\rightarrow$  DRINKS | blue  $\rightarrow$  tea  $\rightarrow$  DRINKS).

Zugehöriges LTS:

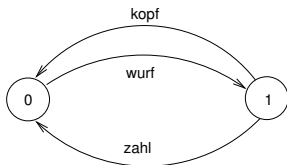


**Bemerkungen:**

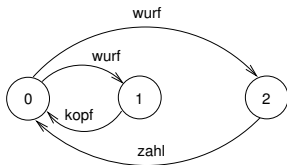
- ▶ blue, red  $\hat{=}$  Input-Aktionen (der Automat empfängt)
- ▶ coffee, tea  $\hat{=}$  Output-Aktionen (der Automat gibt aus)
- ▶ Häufig beginnen die Alternativen einer Auswahl mit Input-Aktionen.
- ▶ Im Beispiel DRINKS gibt es unendlich viele mögliche Abläufe (die alle unendlich lang sind):
  - ▶ red coffee red coffee ...
  - ▶ red coffee blue tea blue ...
  - ▶ ...
  - ▶ blue tea red coffee ...
  - ▶ ...

**Beispiel (Münzwurf):**

MÜNZE1 = (wurf → (kopf → MÜNZE1  
| zahl → MÜNZE1)).



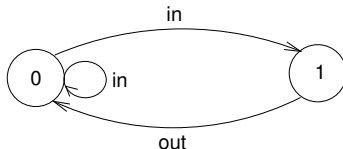
MÜNZE2 = (wurf → kopf → MÜNZE2  
| wurf → zahl → MÜNZE2).

**Beachte:**

Beide Prozesse haben dieselben Abläufe, jedoch verschiedene (nicht äquivalente) LTSe.

## Beispiel (Fehlerhafter Übertragungskanal):

$$F\_CHAN = (in \rightarrow out \rightarrow F\_CHAN \\ | in \rightarrow F\_CHAN).$$



Der Prozess ist nichtdeterministisch!

Im Folgenden betrachten wir **wichtige** abkürzende Schreibweisen für Prozessausdrücke, die sich alle auf die bisherigen 5 Konstrukte für Prozessausdrücke zurückführen lassen.

## Indizierte Aktionen und indizierte Prozesse

### Indizierte Aktionen:

Können zur Modellierung von Daten eines endlichen Datenbereichs (als Indizes von Aktionen) verwendet werden.

### Beispiel (Korrektener Übertragungskanal für Daten):

$$\text{CHAN} = (\text{in}[i:0..2] \rightarrow \text{out}[i] \rightarrow \text{CHAN}).$$

ist Kurzschreibweise für:

$$\begin{array}{l} \text{CHAN} = (\text{in}[0] \rightarrow \text{out}[0] \rightarrow \text{CHAN} \\ \quad | \text{in}[1] \rightarrow \text{out}[1] \rightarrow \text{CHAN} \\ \quad | \text{in}[2] \rightarrow \text{out}[2] \rightarrow \text{CHAN}). \end{array}$$

### Beachte:

Der Indexbereich muss beschränkt sein.



**Indizierte Prozesse:**

Dienen zur Vereinfachung von Prozessdeklarationen (mit lokalen Prozessen).

**Beispiel (Kanal):**

$$\text{CHAN} = (\text{in}[i:0..2] \rightarrow \text{TRANSMIT}[i] ),$$

$$\text{TRANSMIT}[i:0..2] = (\text{out}[i] \rightarrow \text{CHAN}).$$

steht für:

$$\text{CHAN} = \begin{array}{l} (\text{in}[0] \rightarrow \text{TRANSMIT}[0] \\ | \text{in}[1] \rightarrow \text{TRANSMIT}[1] \\ | \text{in}[2] \rightarrow \text{TRANSMIT}[2]), \end{array}$$

$$\text{TRANSMIT}[0] = (\text{out}[0] \rightarrow \text{CHAN}),$$

$$\text{TRANSMIT}[1] = (\text{out}[1] \rightarrow \text{CHAN}),$$

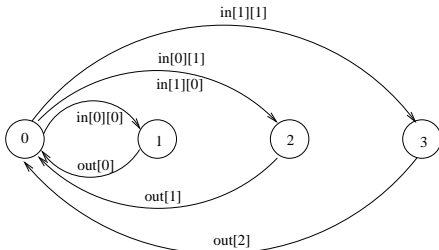
$$\text{TRANSMIT}[2] = (\text{out}[2] \rightarrow \text{CHAN}).$$

## Mehrfache Indizes, arithmetische Ausdrücke und Deklarationen von Konstanten und Bereichen:

### Beispiel (SUM):

const N = 1  
 range T = 0..N  
 range R = 0..2\*N

SUM = (in[a:T][b:T] → TOTAL[a+b]),  
 TOTAL[s:R] = (out[s] → SUM).



## Prozesse mit bewachten Aktionen

(when B a  $\rightarrow$  E | b  $\rightarrow$  F).

Die Aktion a kann nur dann gewählt werden, wenn die Bedingung B erfüllt ist.

### Bemerkung:

- ▶ Bewachte Aktionen können bei der Deklaration indizierter, lokaler Prozesse verwendet werden:

$$P[i:T][j:R] = (\text{when } B \text{ a } \rightarrow E \mid \dots)$$

- ▶ Die Bedingung B darf an Variablen höchstens die Indizes der Prozessdeklaration und formale Parameter (von parametrisierten Prozessen) enthalten.
- ▶ Prozessdeklarationen mit bewachten Aktionen sind Kurzschreibweisen für Prozessdeklarationen ohne bewachte Aktionen.

**Beispiel (Countdown):**

COUNTDOWN = (start  $\rightarrow$  CD[2]),  
 CD[i:0..2] = (when (i > 0) tick  $\rightarrow$  CD[i-1]  
 | when (i == 0) beep  $\rightarrow$  STOP  
 | stop  $\rightarrow$  STOP).

ist Kurzschreibweise für:

COUNTDOWN = (start  $\rightarrow$  CD[2]),  
 CD[2] = (tick  $\rightarrow$  CD[1]  
 | stop  $\rightarrow$  STOP),  
 CD[1] = (tick  $\rightarrow$  CD[0]  
 | stop  $\rightarrow$  STOP),  
 CD[0] = (beep  $\rightarrow$  STOP  
 | stop  $\rightarrow$  STOP).

## Parametrisierte Prozesse

- ▶ Parametrisierte Prozesse erlauben eine generische Definition von Prozessen.
- ▶ Der Prozessparameter muss bei der Deklaration einen "Defaultwert" erhalten (sonst kein endliches LTS).
- ▶ Der Prozess kann jedoch für einen beliebigen aktuellen Parameter in einer anderen Prozessdeklaration aufgerufen werden.

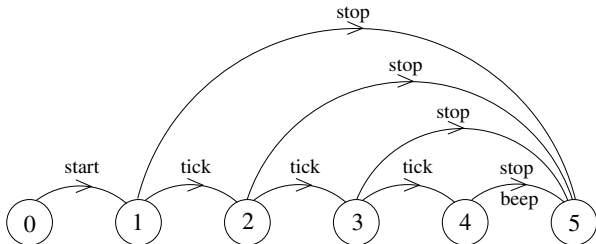
### Beispiel (COUNTDOWN(N)):

$$\begin{aligned} \text{COUNTDOWN}(N=2) &= (\text{start} \rightarrow \text{CD}[N]), \\ \text{CD}[i:0..N] &= (\text{when } (i > 0) \text{ tick} \rightarrow \text{CD}[i-1] \\ &\quad | \text{when } (i == 0) \text{ beep} \rightarrow \text{STOP} \\ &\quad | \text{stop} \rightarrow \text{STOP}). \end{aligned}$$

### Beachte:

Parameter werden nur in *globalen* Prozessdeklarationen verwendet,  
Indizes nur in *lokalen* Prozessdeklarationen.

**Anwendung:**  $\|MY\_COUNTDOWN = COUNTDOWN(3)$ .



## Zusammenfassung abkürzender Notationen: (bis jetzt)

- ▶ indizierte Aktionen,
- ▶ indizierte Prozesse in lokalen Prozessdeklarationen,
- ▶ bewachte Aktionen,
- ▶ parametrisierte Prozessdeklarationen und deren Aufrufe

## 2.4 Semantik von Prozessausdrücken und starke Äquivalenz

Es bezeichne  $\mathcal{E}$  die Menge aller Prozessausdrücke und  $\mathcal{T}$  die Menge aller (endlichen) LTSe über States und ACT.

Die **Semantik von Prozessausdrücken** ist gegeben durch eine Funktion

$$\text{Its}: \mathcal{E} \longrightarrow \mathcal{T},$$

die gemäß der Struktur von Prozessausdrücken folgendermaßen induktiv definiert ist:

1.  $\text{Its}(\text{STOP}) =_{\text{def}} (\{q_0\}, \emptyset, \emptyset, q_0)$ ,  $q_0 \in \text{States}$
2. Sei  $P \in \text{PID}$  ein Prozessidentifikator.  
Zur Definition von  $\text{Its}(P)$  muss eine Prozessdeklaration  $P = E$ . gegeben sein.  
Dann definieren wir:

$$\text{Its}(P) =_{\text{def}} \begin{cases} \text{Its}(E), & \text{falls } P \notin \text{FV}(E) \\ \text{Its}(\text{rec}(P=E)), & \text{falls } P \in \text{FV}(E) \end{cases}$$



### 3. Aktionspräfix:

Sei  $\text{Its}(E) = (S, A, \Delta, q_0)$ .

Dann definieren wir:

$$\text{Its}((a \rightarrow E)) =_{\text{def}} (S \cup \{p_0\}, A \cup \{a\}, \Delta \cup \{(p_0, a, q_0)\}, p_0),$$

wobei  $p_0 \in \text{States} \setminus S$ .

### 4. Auswahl:

Für  $i = 1, \dots, n$  sei  $\text{Its}(E_i) = (S_i, A_i, \Delta_i, q_i)$  mit paarweise disjunkten  $S_i$ .

Dann definieren wir:

$$\begin{aligned} \text{Its}((a_1 \rightarrow E_1 \mid \dots \mid a_n \rightarrow E_n)) =_{\text{def}} \\ (S_1 \cup \dots \cup S_n \cup \{p_0\}, \\ A_1 \cup \dots \cup A_n \cup \{a_1, \dots, a_n\}, \\ \Delta_1 \cup \dots \cup \Delta_n \cup \{(p_0, a_1, q_1), \dots, (p_0, a_n, q_n)\}, \\ p_0), \end{aligned}$$

wobei  $p_0 \in \text{States} \setminus (S_1 \cup \dots \cup S_n)$ .

## 5. Rekursive Prozessausdrücke:

Sei  $P$  ein Prozessidentifikator und  $E$  ein Prozessausdruck, so dass  $P \in FV(E)$ .

Sei  $\text{Its}(E) = (S, A, \Delta, q_0)$  wobei wir annehmen  $\text{Its}(P) = (\{q_P\}, \emptyset, \emptyset, q_P)$ .

Dann definieren wir:

$$\text{Its}(\text{rec}(P = E)) =_{\text{def}} (S \setminus \{q_P\}, A, \Delta_{\text{rec}}, q_0),$$

wobei

$$\Delta_{\text{rec}} = \{(q, a, q') \mid (q, a, q') \in \Delta \text{ und } q, q' \neq q_P\} \cup \{(q, a, q_0) \mid (q, a, q_P) \in \Delta\}.$$

**Definition (Starke Bisimulation):**

Seien  $T, T' \in \mathcal{T}$ ,  $T = (S, A, \Delta, q_0)$ ,  $T' = (S', A', \Delta', q_0')$  mit  $A = A'$ .

Eine **starke Bisimulation** zwischen  $T$  und  $T'$  ist eine Relation  $R \subseteq S \times S'$ , so dass für alle  $(q, q') \in R$  und für alle  $a \in A$  gilt:

$$(1) \quad q \xrightarrow{a}_{\Delta} p \implies \exists p' \in S' \text{ mit } q' \xrightarrow{a}_{\Delta'} p' \text{ und } (p, p') \in R.$$

$$(2) \quad q' \xrightarrow{a}_{\Delta'} p' \implies \exists p \in S \text{ mit } q \xrightarrow{a}_{\Delta} p \text{ und } (p, p') \in R.$$

**Bemerkung:**

Sei  $T = (S, A, \Delta, q_0) \in \mathcal{T}$ .

Die Identität  $= \subseteq S \times S$  ist eine starke Bisimulation zwischen  $T$  und  $T$ .

**Definition (Starke Äquivalenz von LTSen):**

Seien  $T, T' \in \mathcal{T}$ ,  $T = (S, A, \Delta, q_0)$ ,  $T' = (S', A', \Delta', q_0')$ .

$T$  und  $T'$  sind **stark äquivalent**, geschrieben  $T \sim T'$ , wenn gilt:

- (a)  $T$  und  $T'$  haben dieselben Aktionen, d.h.  $A = A'$ .
- (b) Es gibt eine starke Bisimulation  $R \subseteq S \times S'$  zwischen  $T$  und  $T'$ , so dass  $(q_0, q_0') \in R$ .

**Lemma:**

$\sim$  ist eine Äquivalenzrelation auf  $\mathcal{T}$ .

**Bemerkung:**

Stark äquivalente LTSen haben dieselben Abläufe.

Die Umkehrung gilt jedoch nicht (vgl. Beispiel Münzwurf von oben).

**Beispiele:** (Werden in der Vorlesung eingetragen.)

**Definition (Reach(T)):**

Sei  $T = (S, A, \Delta, q_0)$  ein LTS.

Das **reachable Sub-LTS** von  $T$  ist gegeben durch  $\text{Reach}(T) = (S_r, A, \Delta_r, q_0)$ , wobei

- ▶  $S_r \subseteq S$  die kleinste Teilmenge von  $S$  ist, so dass gilt:

$$(0) \quad q_0 \in S_r,$$

$$(1) \quad q \in S_r \text{ und } q \xrightarrow{a}_{\Delta} p \implies p \in S_r,$$

- ▶  $\Delta_r = \{(q, a, p) \in \Delta \mid q, p \in S_r\}$ .

$T$  heißt **reachable**, wenn  $T = \text{Reach}(T)$ .

**Bemerkung:**

Für alle FSP-Ausdrücke  $E$  ist  $\text{Its}(E)$  reachable.

**Lemma:**

Seien  $T, T' \in \mathcal{T}$ ,  $T = (S, A, \Delta, q_0)$ ,  $T' = (S', A', \Delta', q_0')$ . Es gilt:

1.  $T \sim \text{Reach}(T)$ ,
2.  $T \sim T' \iff \text{Reach}(T) \sim \text{Reach}(T')$ .

### Definition (Starke Äquivalenz von Prozessen):

Zwei Prozesse  $E, F \in \mathcal{E}$  sind **stark äquivalent** (stark bisimilar), geschrieben  $E \sim F$ , wenn gilt:  $\text{Its}(E) \sim \text{Its}(F)$ .

### Beispiele (Algebraische Gesetze):

Seien  $a, b \in \text{ACT}$  und  $E, F$  Prozessausdrücke.

- ▶  $(a \rightarrow E \mid b \rightarrow F) \sim (b \rightarrow F \mid a \rightarrow E)$  (*Kommutativgesetz für Auswahl*)
- ▶  $(a \rightarrow E \mid a \rightarrow E) \sim (a \rightarrow E)$  (*Idempotenzgesetz für Auswahl*)
- ▶  $E \sim F \implies (a \rightarrow E) \sim (a \rightarrow F)$  (*Kongruenzregel bzgl. Aktionspräfix*)
- ▶  $E \sim E'$  und  $F \sim F' \implies (a \rightarrow E \mid b \rightarrow F) \sim (a \rightarrow E' \mid b \rightarrow F')$   
(*Kongruenzregel bzgl. Auswahl*)

## 2.5 Implementierung von Prozessen

### Betriebssystem-Prozesse und Threads

Ein *BS-Prozess* besitzt einen eigenen Adressraum und wird repräsentiert durch

- ▶ Daten (globale und lokale Variable);  
die lokalen Variablen sind in einem Keller organisiert,  
die globalen Variablen in einem Heap
- ▶ Code (Befehle)
- ▶ Deskriptor (organisatorische Daten und Werte der Maschinenregister)

Ein *BS-Prozess* ist ein "schwergewichtiger Prozess" (z.B. Ausführung eines Anwendungsprogramms).

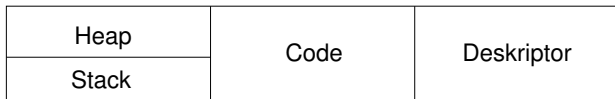
Ein *Thread* ist ein "leichtgewichtiger Prozess", der innerhalb eines *BS-Prozesses* (evt. parallel zu anderen *Threads*) abläuft.

- ▶ Jeder *Thread* besitzt einen eigenen Stack für seine lokalen Variablen und einen eigenen Deskriptor.
- ▶ Der *Thread-Code* ist im Code-Segment des *BS-Prozesses* enthalten.
- ▶ Jeder *Thread* hat Zugriff auf die globalen Variablen des *BS-Prozesses*.

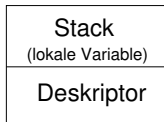


# Betriebssystem-Prozess

BS-Prozess

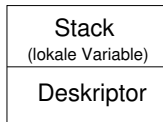


Thread 1



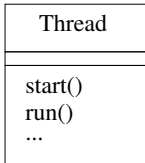
....

Thread n



## Realisierung von Threads in Java

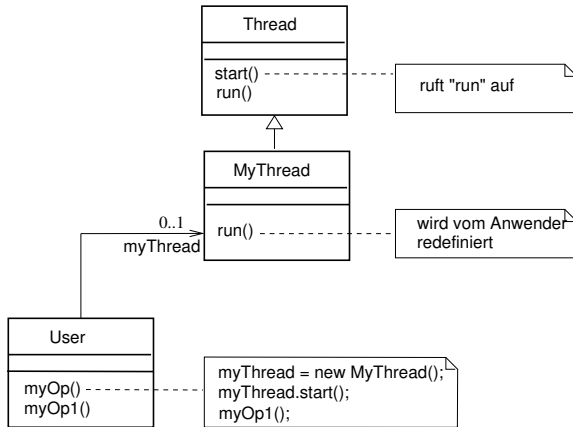
Threads werden in Java durch Objekte der Klasse "Thread" (im Paket java.lang) realisiert.



Es bezeichne  $t$  ein Objekt der Klasse Thread oder einer Subklasse von Thread.

- ▶ Der Methodenaufwurf  $t.start()$ ; bewirkt, dass das Thread-Objekt  $t$  aktiviert wird und seine `run`-Methode aufgerufen wird.
- ▶ Der aufrufende Thread setzt dann seine Tätigkeit parallel zur Ausführung der `run`-Methode des Threads  $t$  fort.

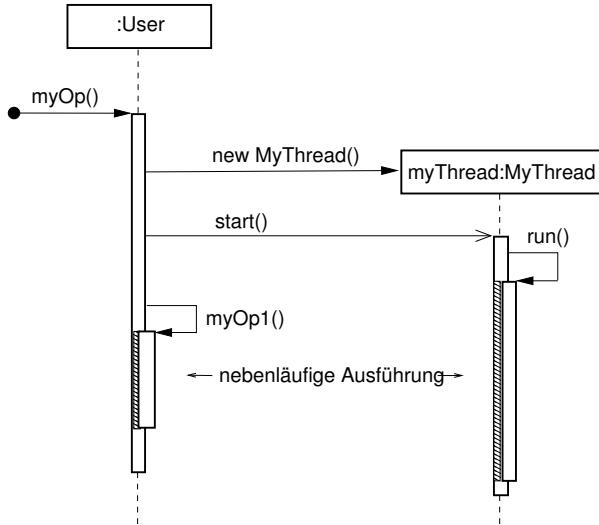
# 1. Realisierung von Threads mittels Vererbung



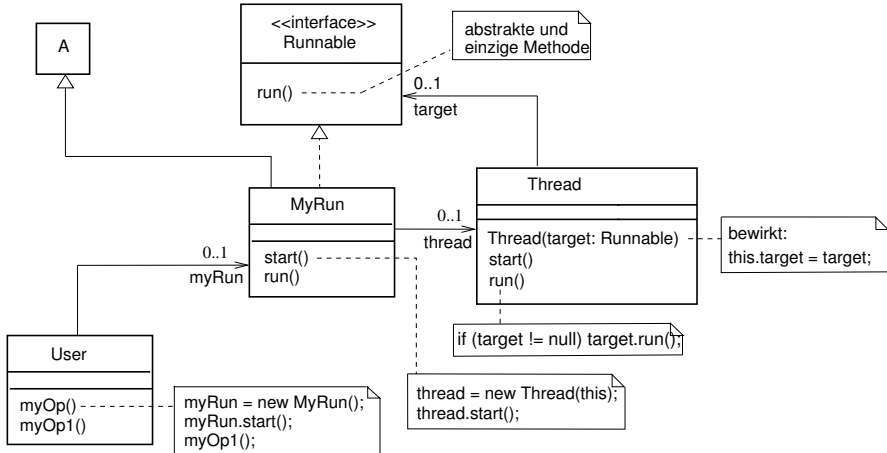
## Beachte:

“MyThread“ kann nicht Erbe einer weiteren Klasse sein, da in Java Mehrfachvererbung nicht möglich ist!

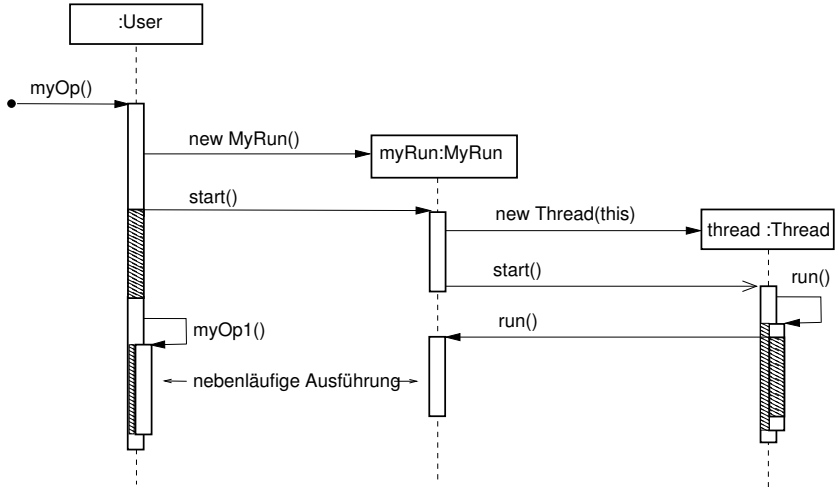
## Sequenzdiagramm mit Objekt der Klasse MyThread



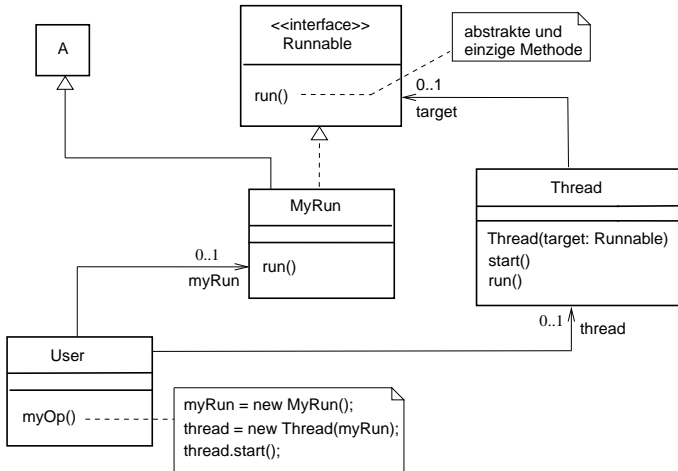
## 2. Realisierung von Threads durch Verwendung des Interfaces "Runnable"



## Sequenzdiagramm mit Objekt der Klasse MyRun



## Klassendiagramm mit Interface Runnable (Variante)



## Beispiel (Implementierung des Countdown-Prozesses):

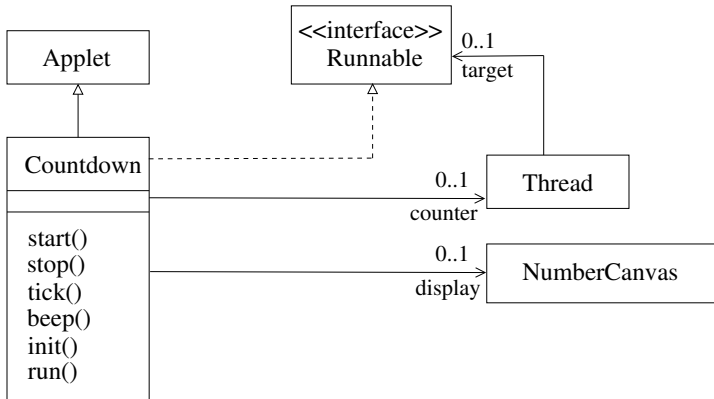
$$\begin{aligned} \text{COUNTDOWN}(N=10) &= (\text{start} \rightarrow \text{CD}[N]), \\ \text{CD}[i:0..N] &= (\text{when } (i > 0) \text{ tick} \rightarrow \text{CD}[i-1] \\ &\quad | \text{when } (i == 0) \text{ beep} \rightarrow \text{STOP} \\ &\quad | \text{stop} \rightarrow \text{STOP}). \end{aligned}$$

### Aktionen:

- ▶ externe: start, stop
- ▶ interne: tick, beep



## Klassendiagramm der Implementierung



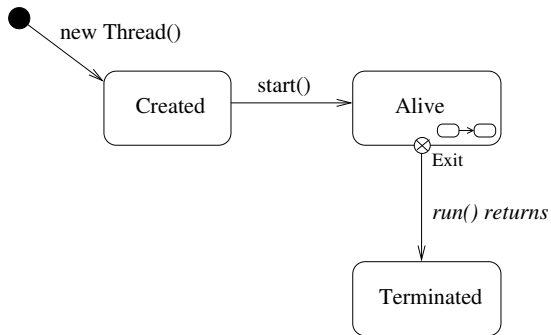
## Java-Implementierung

```
public class Countdown extends Applet implements Runnable {
    final static int N = 10;
    int i;
    Thread counter;
    AudioClip beepsound, ticksound;
    NumberCanvas display;

    public void start() {
        i = N;
        counter = new Thread(this);
        counter.start();
    }
    public void stop() {
        counter = null;
    }
    private void tick() {...}
    private void beep() {...}
    public void init() {...}

    public void run() {
        while (true) {
            if (counter == null) return;
            if (i == 0) {beep(); return;}
            if (i > 0) {tick(); i = i-1;}
        }
    }
}
```

## Lebenszyklus eines Java-Threads



## Unterezustände des Alive-Zustands

