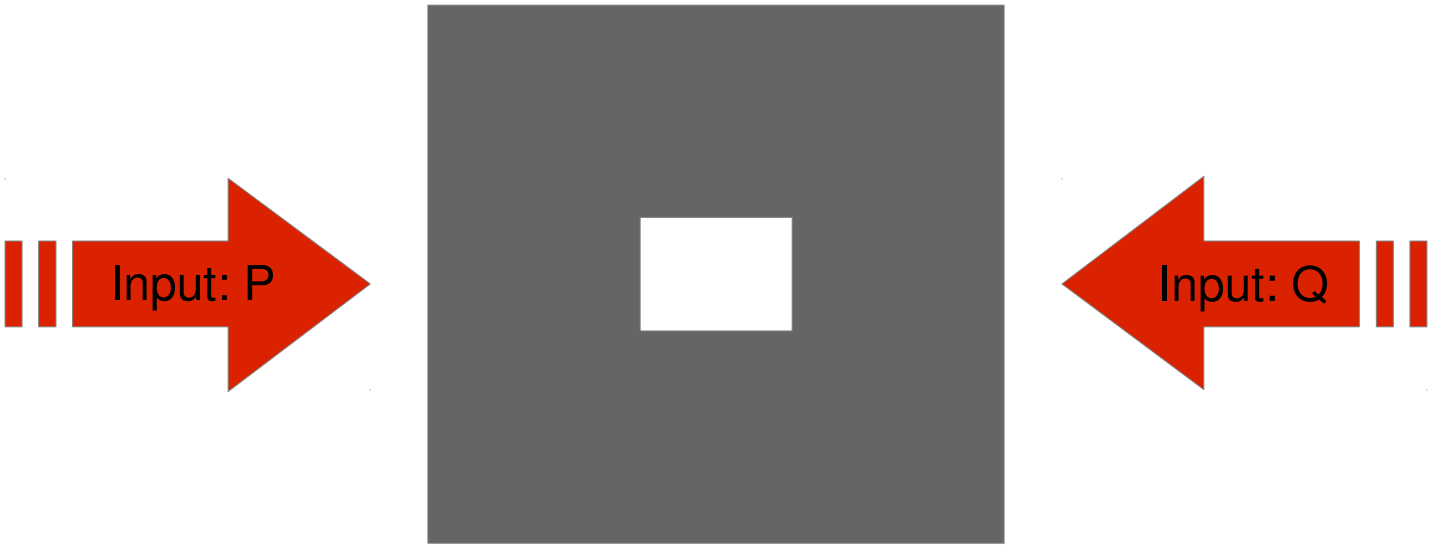
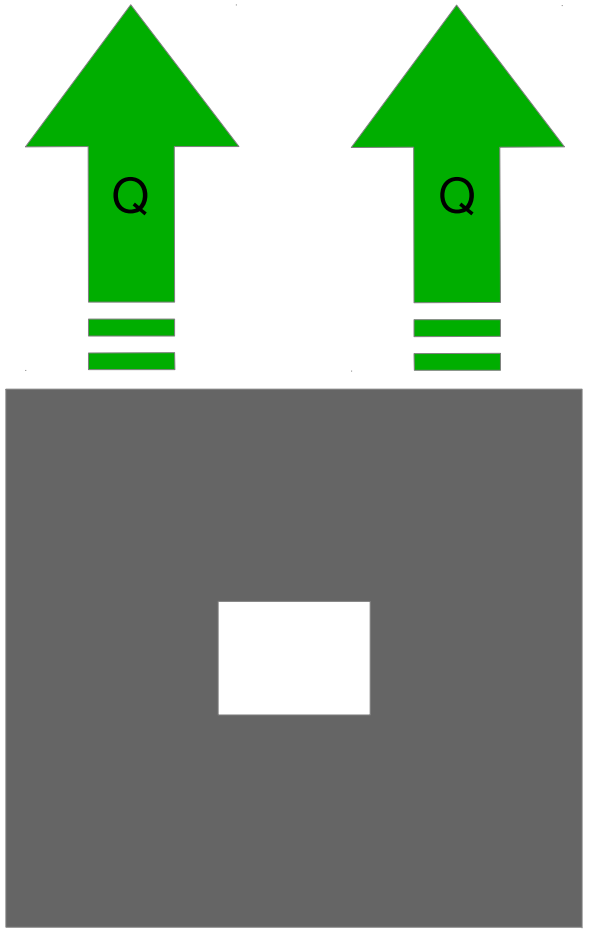
Wartefreie Synchronisation

Ralph Mück
ralph.mueck@studium.uni-erlangen.de

//Motivation



//Motivation



//Gliederung

- Definition
- Beispiele
- Universalobjekte
- Zusammenfassung & Ausblick

//**Definition:** Wartefreie Synchronisation

- Jede Operation auf ein nebenläufiges Datenobjekt wird in endlich vielen Schritten fertiggestellt

- Unabhängig von Ausführungsgeschwindigkeit der anderen Prozesse

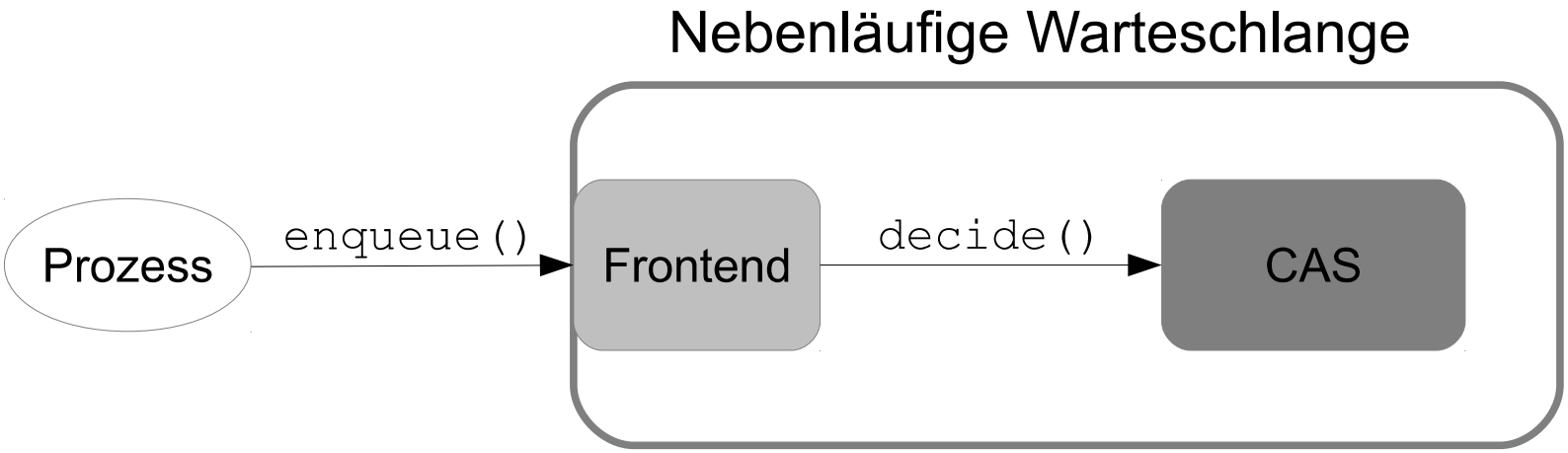
Einigungsprotokoll *consensus protocol*

- Schnittstelle:

```
value decide(value proposal){...}
```

- Protokolleigenschaften:
  - ✔ Konsistenz
  - ✔ Wartefreiheit
  - ✔ Gültigkeit

Einigungsprotokoll *consensus protocol*



Einigungsnummer *consensus number*

- Nummer n ist höchste Anzahl an Prozessen für die das Einigungsproblem gelöst wird

- Kein Objekt kann durch Objekt niedrigerer Ebene implementiert werden

Einigungsnummer *consensus number*

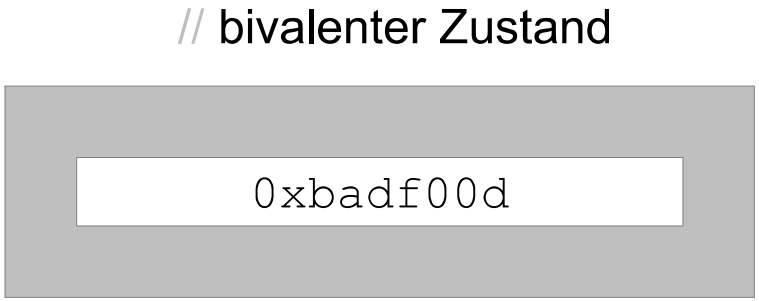
Einigungsnummer	**Objekt**
∞	Compare & Swap, erweiterte Warteschlange
...	...
2	Test & Set, Fetch & Add, Warteschlange
1	Lese- / Schreiberegister

//**Beispiel 1:** Atomare Lese-/Schreibregister

- Einigungsnummer 1

- Trotz starkem Fokus in Forschung wenig bis keine Relevanz in wartefreier Synchronisation

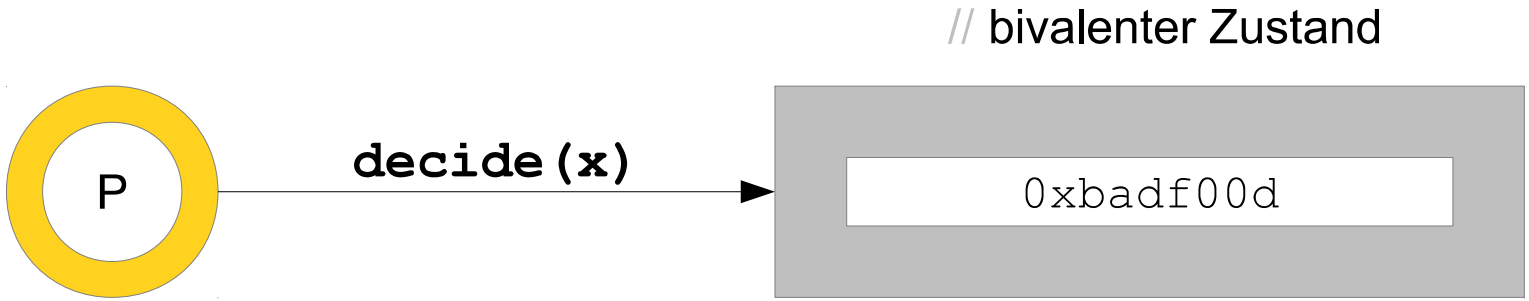
Veranschaulichung:



Ausgangszustand:

Register mit beliebigem Inhalt

Veranschaulichung:

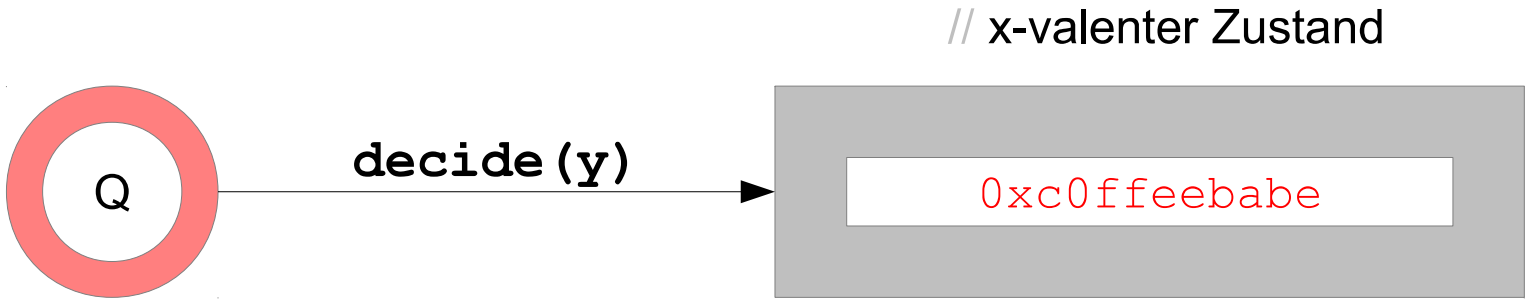


Prozess P liest das Register...

Veranschaulichung:

… und überführt es damit in einen x-valenten Zustand.

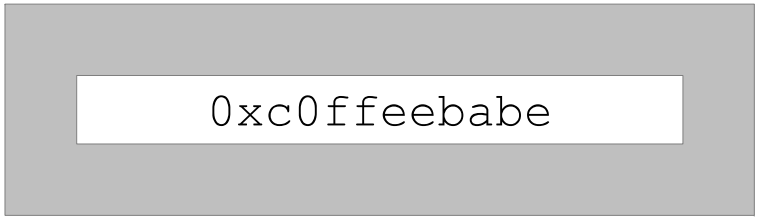
Veranschaulichung:



Dann:
Prozess Q beschreibt das Register...

Veranschaulichung:

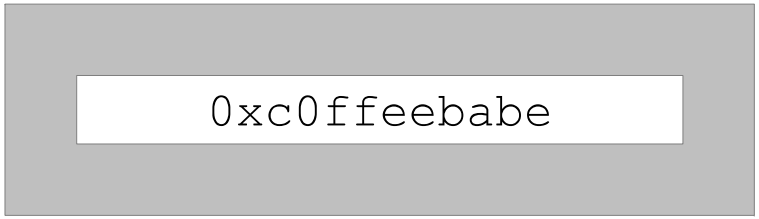
// x-valenter Zustand



...weil Prozess P schneller war und sich das Protokoll bereits in einem x-valenten Zustand befindet, muss sich Q dem Entscheidungswert beugen

Veranschaulichung:

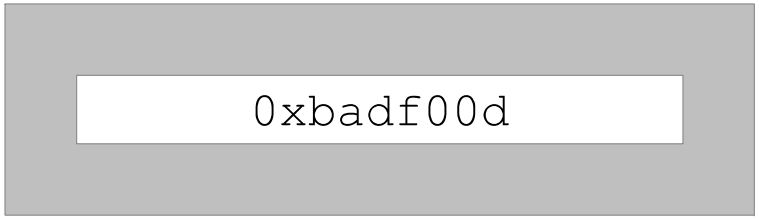
// x-valenter Zustand



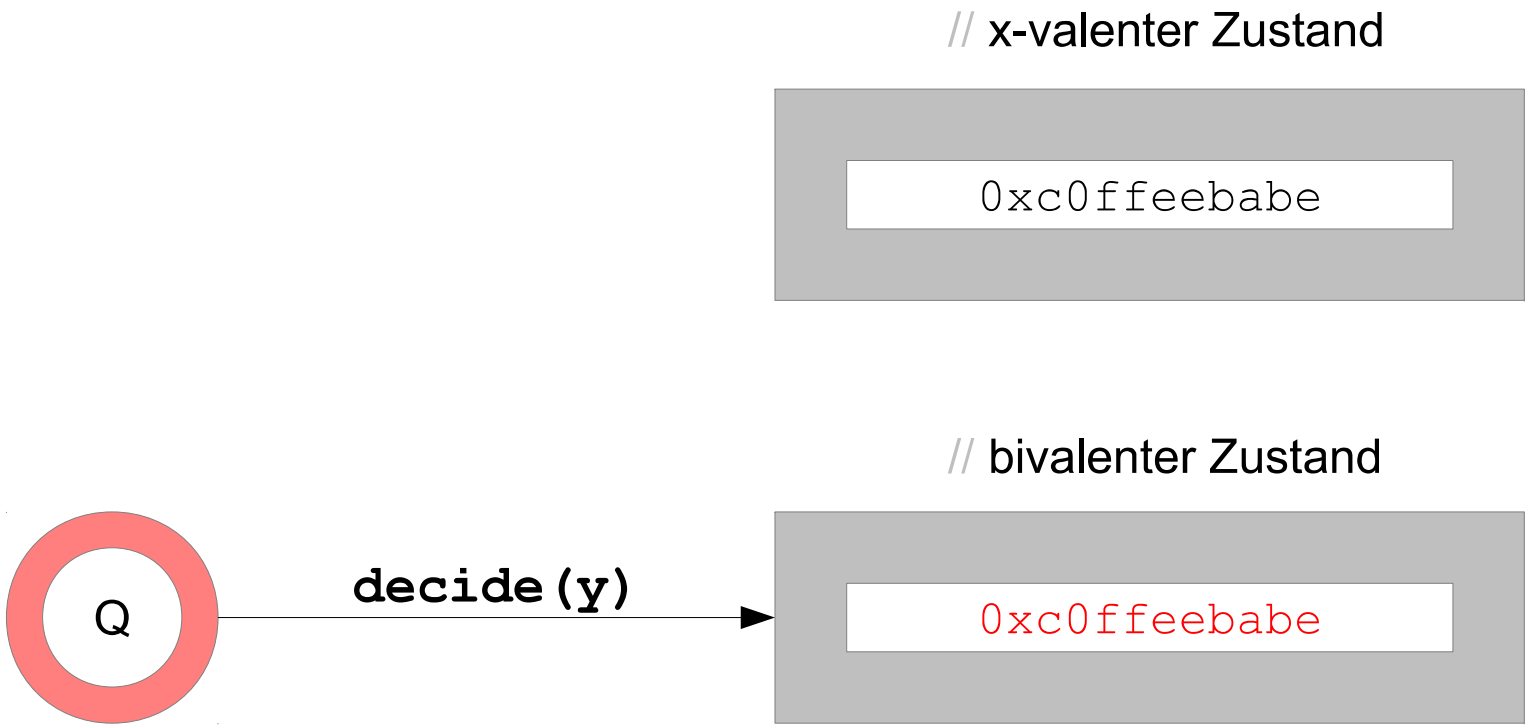
Version 2:

Ebenfalls beschriebenes Register
im Ausgangszustand

// bivalenter Zustand

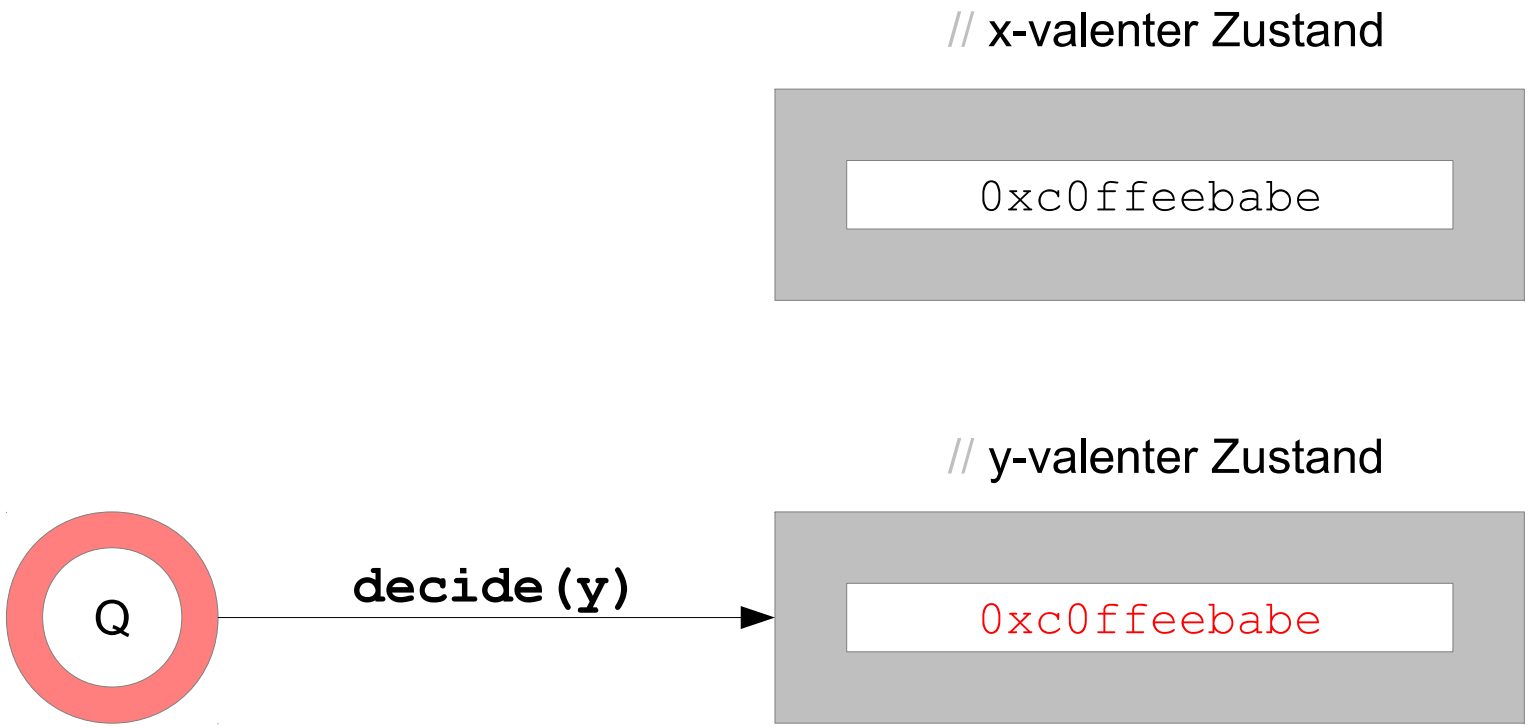


Veranschaulichung:



Diesmal ist Q schneller, beschreibt das Register...

Veranschaulichung:



...und überführt das Protokoll in einen y-valenten Zustand.

Veranschaulichung:

// x-valenter Zustand

```
0xc0ffeebabe
```

Widerspruch:

P wird den Registerinhalt nicht mehr modifizieren – in beiden Fällen steht also der gleiche Wert im Register, obwohl wir zwei unterschiedliche Zustände haben!

// y-valenter Zustand

```
0xc0ffeebabe
```

//**Beispiel 2:** Warteschlange

- Einigungsnummer 2
- Operationen: `enqueue()`, `dequeue()`
- Array `prefer[]`

//**Beispiel 2:** Warteschlange

```
//initialize queue q
q.enqueue(0);
q.enqueue(1);

//FIFO consensus
value decide(value proposal)
{
   prefer[P] = proposal;
   if (q.dequeue() == 0)
   {
      return prefer[P];
   }
   else
   {
      return prefer[Q];
   }
}
```

//**Beispiel 3:** Compare & Swap

- Einigungsnummer ∞
- Operation `cas()`
- Register `reg`, initialisiert auf `false`

//**Beispiel 3:** Compare & Swap

```
//initialize Register reg
reg = false;

//CAS consensus
value decide(value proposal)
{
   value first = cas(reg, false, proposal);
   if (first == false)
   {
      return proposal;
   }
   else
   {
      return first;
   }
}
```

//Universalobjekte

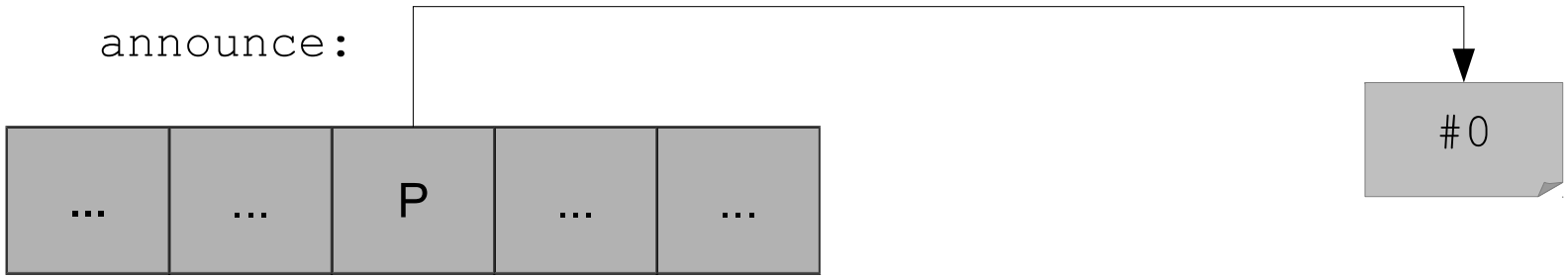
- Zweck: Transformation in wartefreien Code

- In System mit n Prozessen → Entscheidungsnummer ≥ n

Algorithmus:

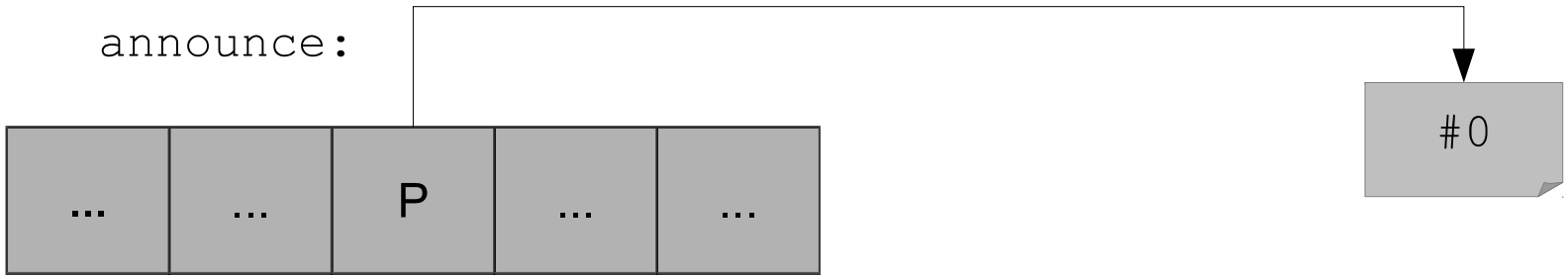
Schritt 1: Initialisierung der Zelle

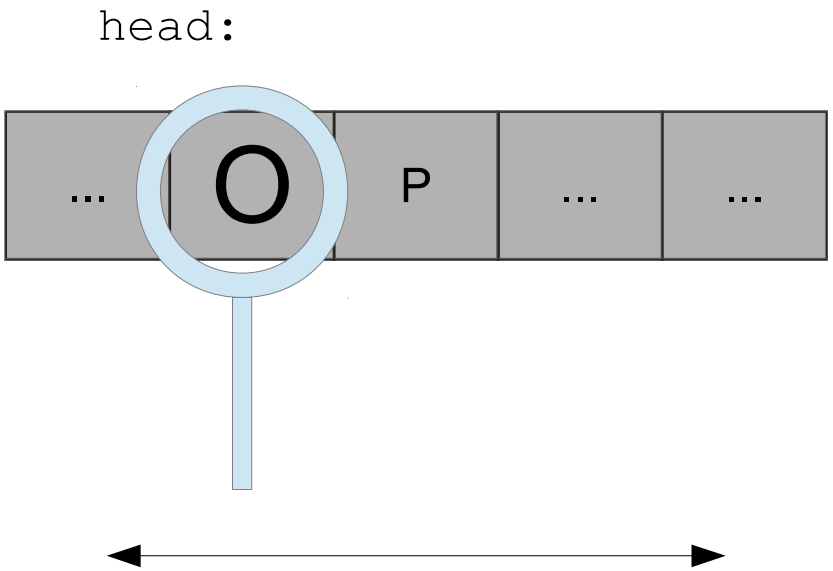
Algorithmus:



Schritt 2: Verzeigerung der Zelle

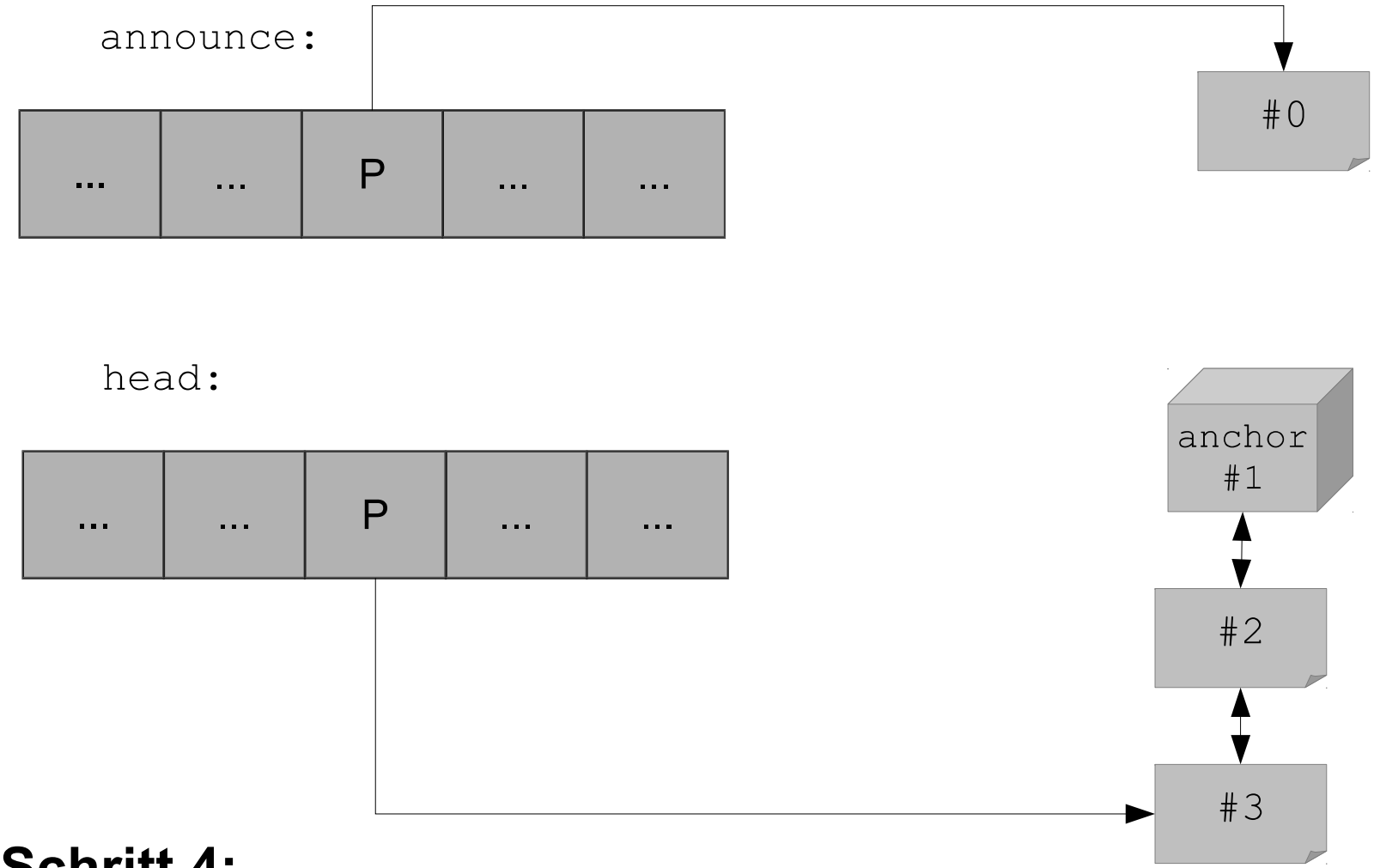
Algorithmus:





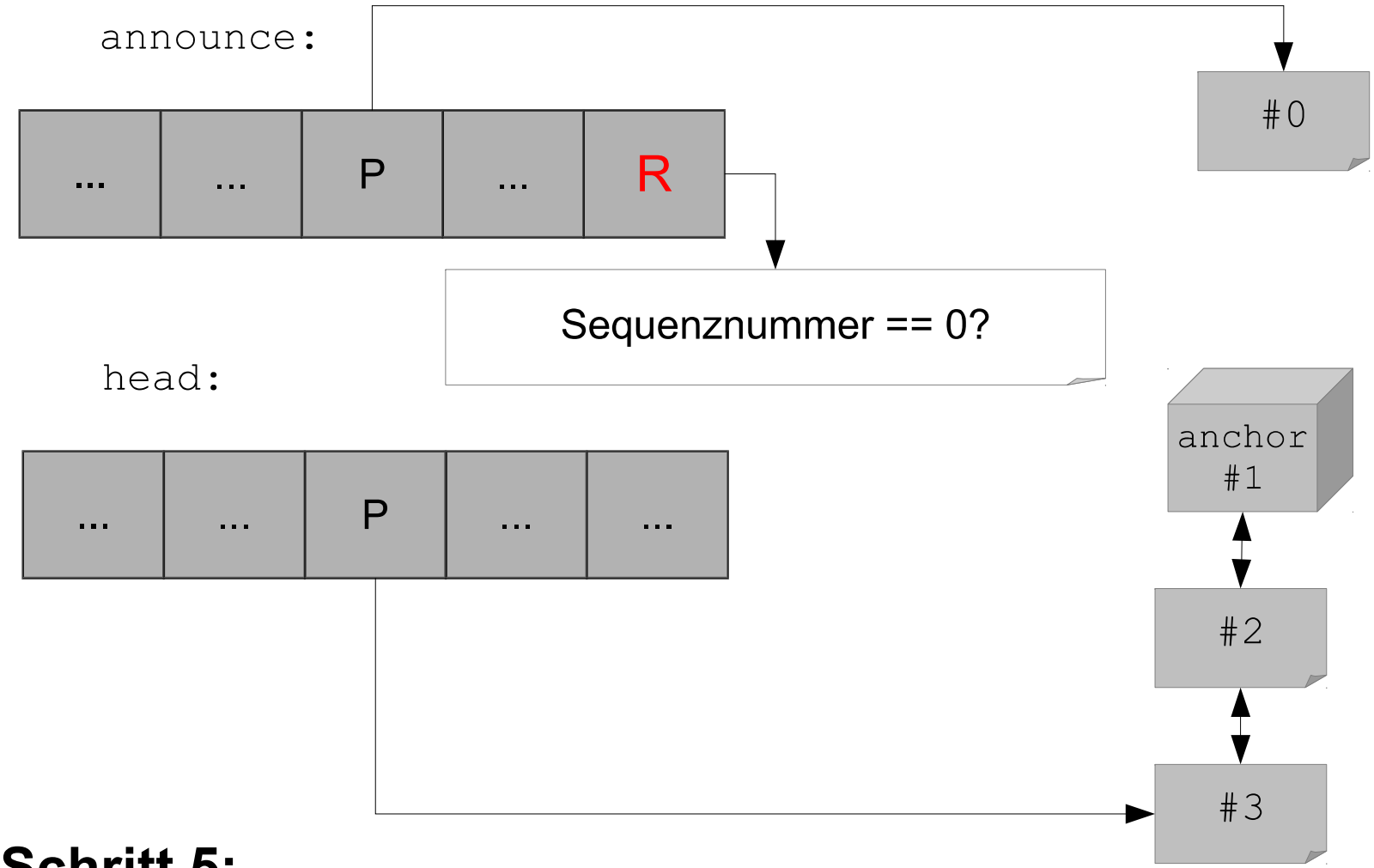
Schritt 3:
Zelle mit der höchsten Sequenz-nummer suchen

Algorithmus:



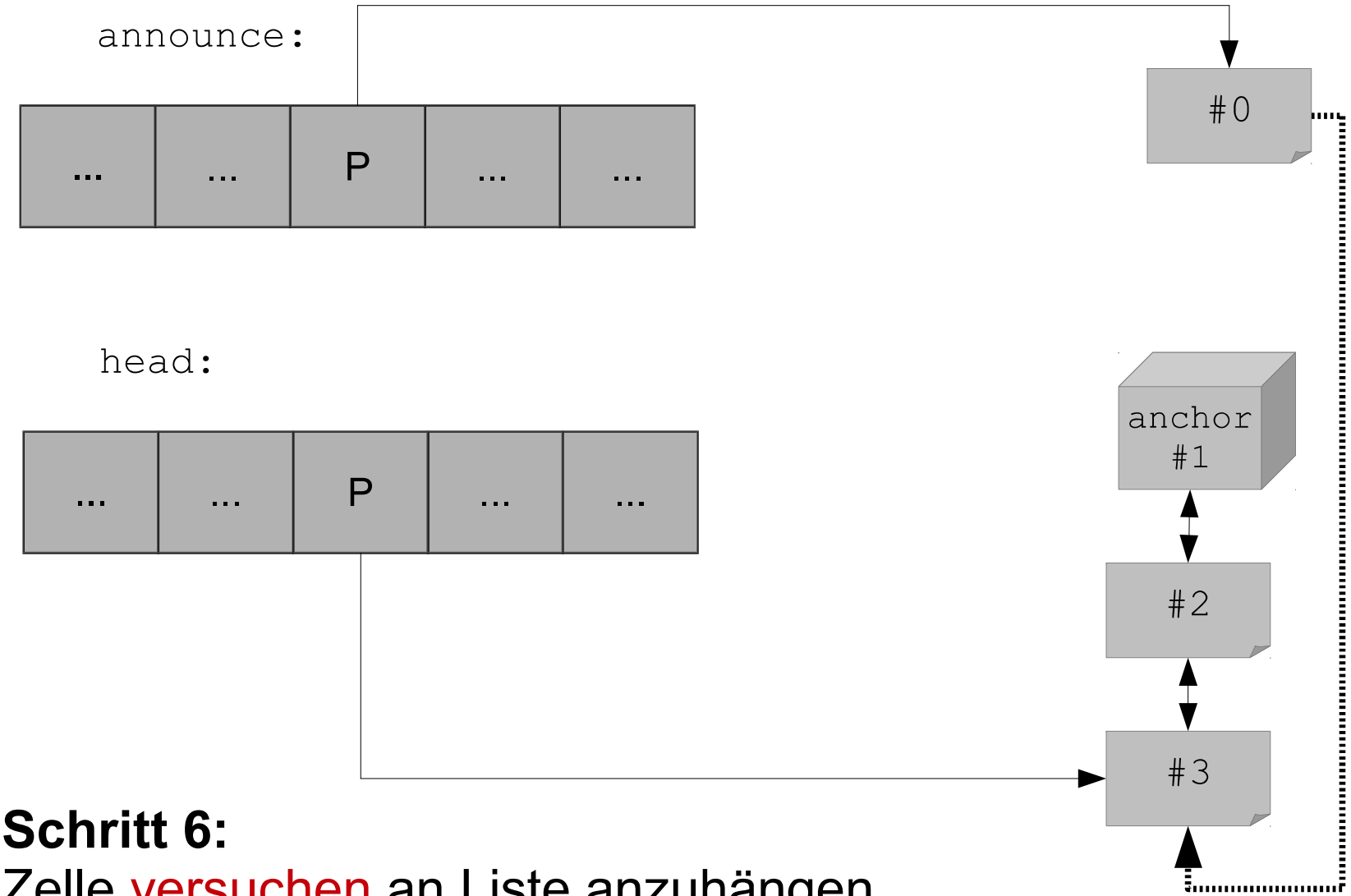
Schritt 4:
Zelle mit der höchsten Sequenznummer verzeigern

Algorithmus:



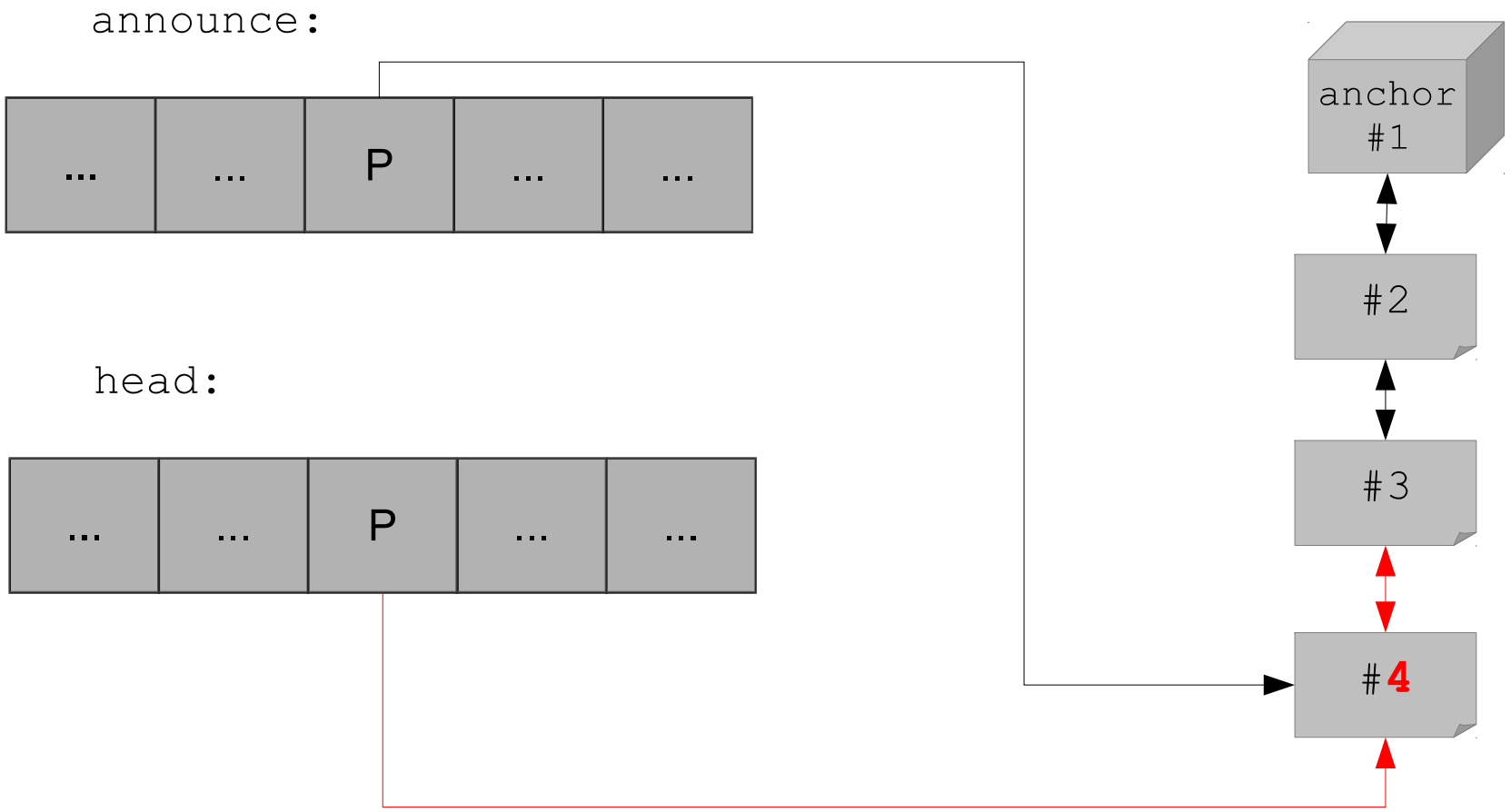
Schritt 5:
Hilfszelle suchen und ggf. versuchen diese anzuhängen

Algorithmus:



Schritt 6:
Zelle versuchen an Liste anzuhängen

Algorithmus:



Schritt 7:
Sequenznummer aktualisieren, Zeiger „umbiegen“

Universalobjekte:

- Vorteile:
  - Jedes Objekt wartefrei implementierbar
  - Vorteile von Wartefreiheit treffen auf Objekt zu

- Nachteile:
  - u.U. lange Laufzeit
  - u.U. hoher Speicherbedarf

//Zusammenfassung

- Wartefreie Synchronisation ist die stärkste Form von nichtblockierender Synchronisation
- Einigungsnummer als Einordnung in wartefreie Hierarchie
- Mit Universalobjekten (z.B. CAS) können alle Objekte wartefrei implementiert werden