3.3 Zustandsmaschinen-Sicht

3.3.1 Zustandsmaschine

3.3.2 Ereignisse

3.3.3 Zustände und Übergänge

3.3.1 Zustandsmaschine

- *Zustandsmaschine* (State Machine) = Beschreibung des Verhaltens eines einzelnen Objektes über den Zeitraum seiner Existenz
- Anwendung:
  - Klassen
  - Anwendungsfälle
  - ganze Systeme

Zustandsmaschine (Forts.)

- Visualisierung von Zustandsmaschinen:
  - *Aktivitätsdiagramm*
    Betonung auf Abläufen und Nebenläufigkeiten innerhalb eines Systems
  - *Zustandsübergangsdiagramm*
    Hervorhebung der möglichen Zustände und der Übergänge zwischen diesen

Zustandsmaschine (Forts.)

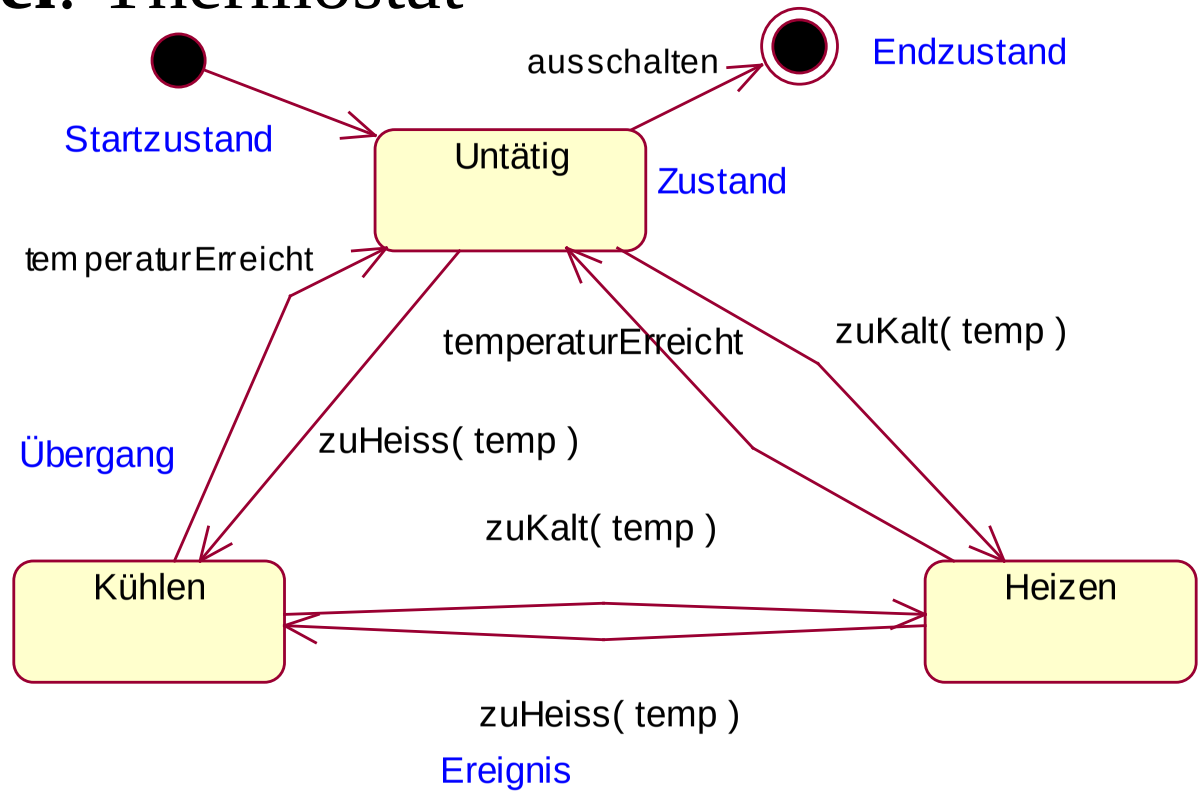
- Zustandsmaschinen-Sicht dient der
  - Betrachtung eines einzelnen Objektes als isolierte Einheit, die mit der Aussenwelt über Ereignisse kommuniziert
  - Fokussierung auf speziellen Bestandteil eines Systems, dessen Verhalten im Detail beschrieben wird

Zustandsmaschine (Forts.)

- Merkmale:
  - geeignet für Beschreibung und Verständnis komplexer reaktiver Objekte, z.B. Benutzer-schnittstellen und Kontrollobjekte
  - ungeeignet zur Veranschaulichung des Zusammen-spiels mehrerer Objekte oder von Teilen des Systems ($\Rightarrow$ Interaktions-Sicht)

Zustandsmaschine (Forts.)

Beispiel: Thermostat



3.3.2 Ereignisse

- *Ereignis* (Event) = Spezifizierung eines signifikanten Geschehens, welches durch Zeit und Ort beschreibbar ist
- Ereignisse haben keine Dauer (konzeptuell)
- Verwendung bei Zustandsmaschinen als Stimulus für Zustandsübergang

Ereignisse (Forts.)

- 4 Arten von Ereignissen:
  - Signal
  - Aufruf
  - zeitgesteuertes Ereignis
  - Änderung des Zustandes

Ereignisse - Signale

- *Signal* (Signal) = Konstrukt zur asynchronen Kommunikation zwischen Objekten
- Sprechweise:
  - Sender „wirft“ (throws) Signal
  - Empfänger „fangen“ (catch) Signal

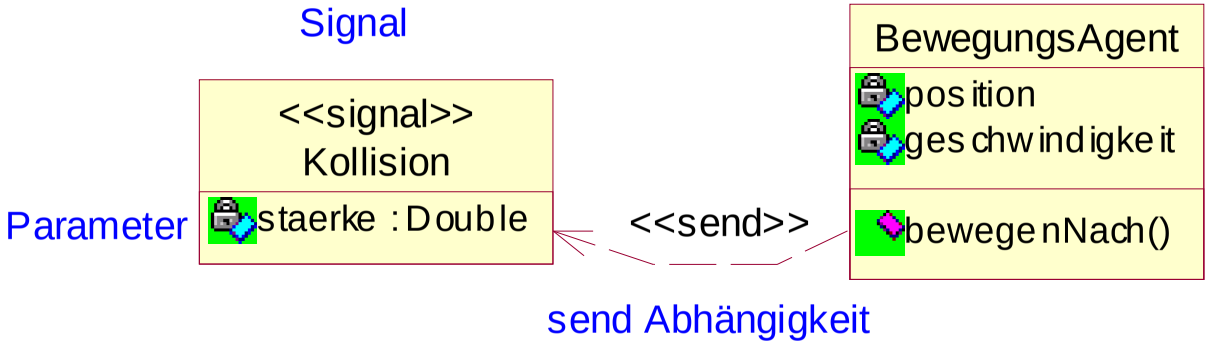
Ereignisse - Signale (Forts.)

- Verwendung/Einsatz:
  - Mitteilung einer Information (kleine Datenmenge), z.B. bei der Ausführung einer Operation
  - bewirkt u.U. einen Zustandsübergang in der Zustandsmaschine des Empfängers (wenn dieser das Signal behandeln kann)

Ereignisse - Signale (Forts.)

Graphische Darstellung (Sender):

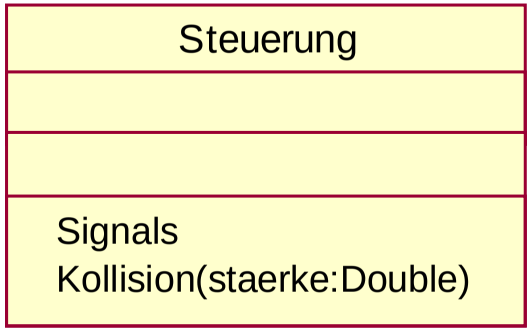
- Modellierung als stereotypisierte Klassen
- Mit `send` versehene Abhängigkeitsbeziehung für Zuordnung von Signalen zu Operationen



Ereignisse - Signale (Forts.)

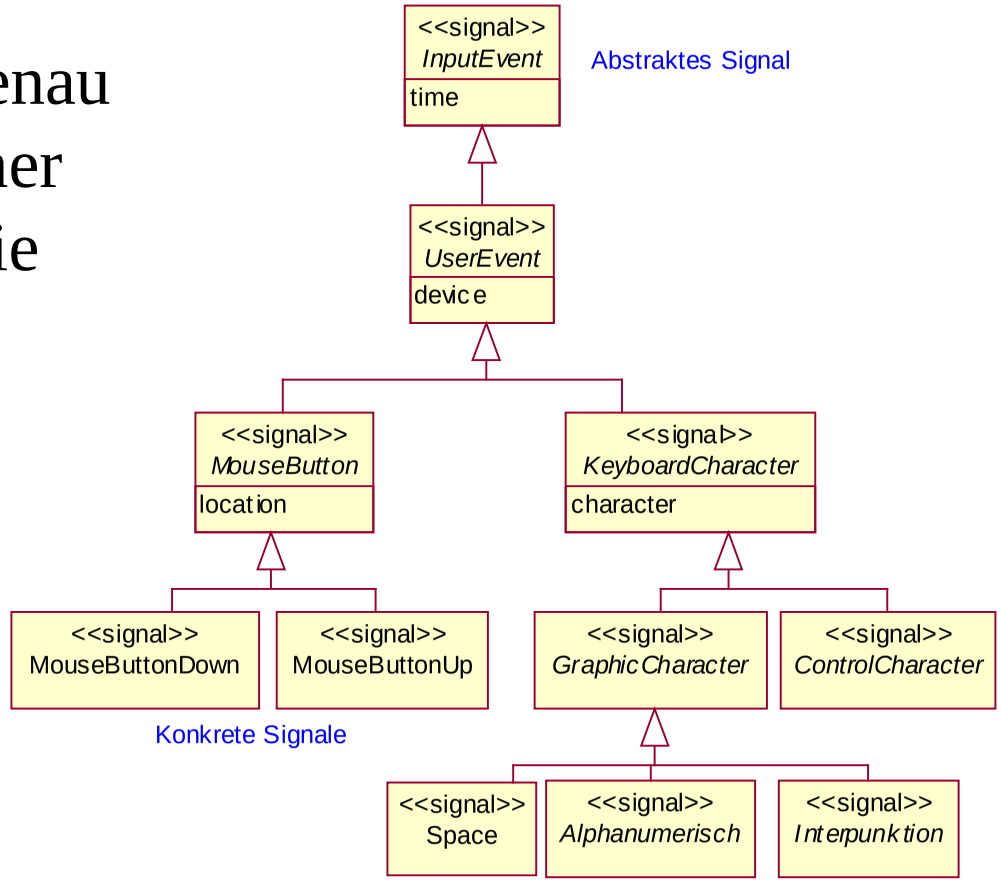
Graphische Darstellung (Empfänger):

- als Ereignis im Zustandsübergangsdiagramm des Empfängers
- explizit in der Klasse des Empfängers



Ereignisse - Signale (Forts.)

- Signale können - genau wie Klassen - in einer Ableitungshierarchie strukturiert werden



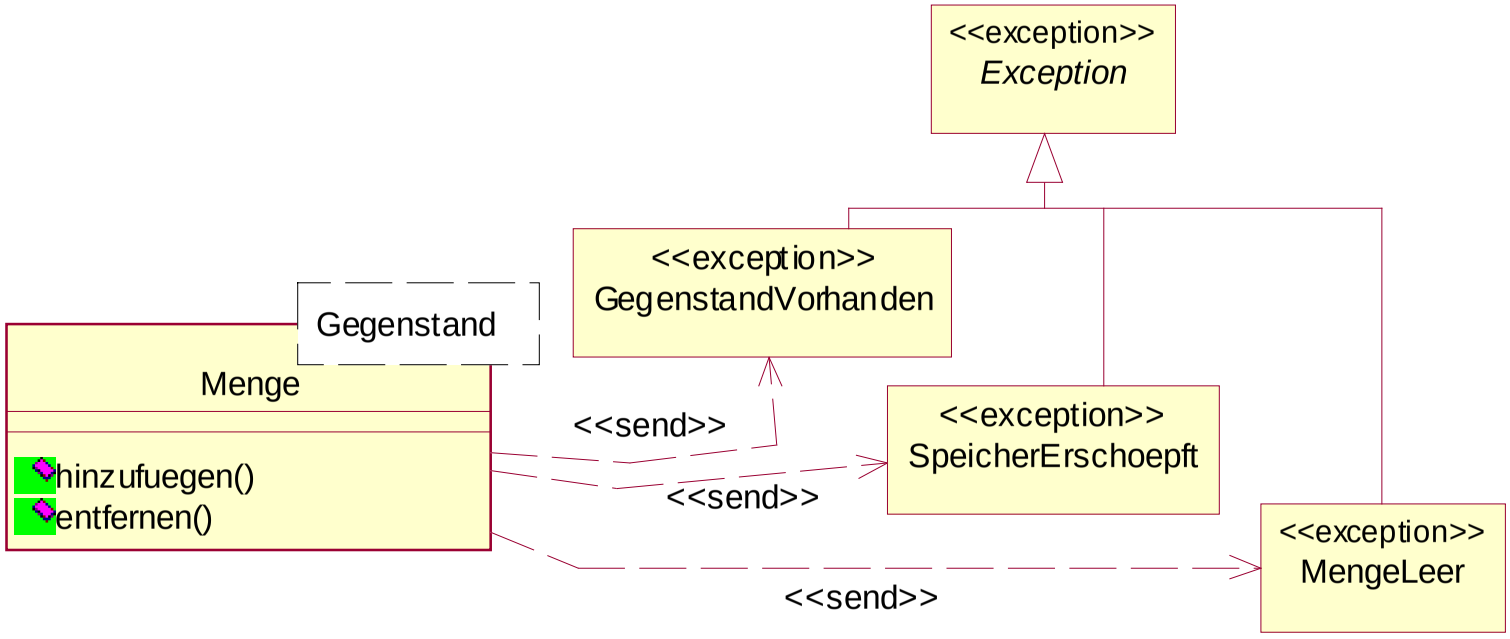
Ereignisse - Ausnahmen (Exceptions)

- *Ausnahmen* (Exceptions) = spezielle Art von Signalen, die i.a. von Klassenoperationen versendet (geworfen) werden
- OO Konzept für Fehlerbehandlung (C++, Java)
  - flexibleres und mächtigeres Konzept als Benutzung von Rückgabewerten
  - Programme besser lesbar

Ereignisse - Ausnahmen (Forts.)

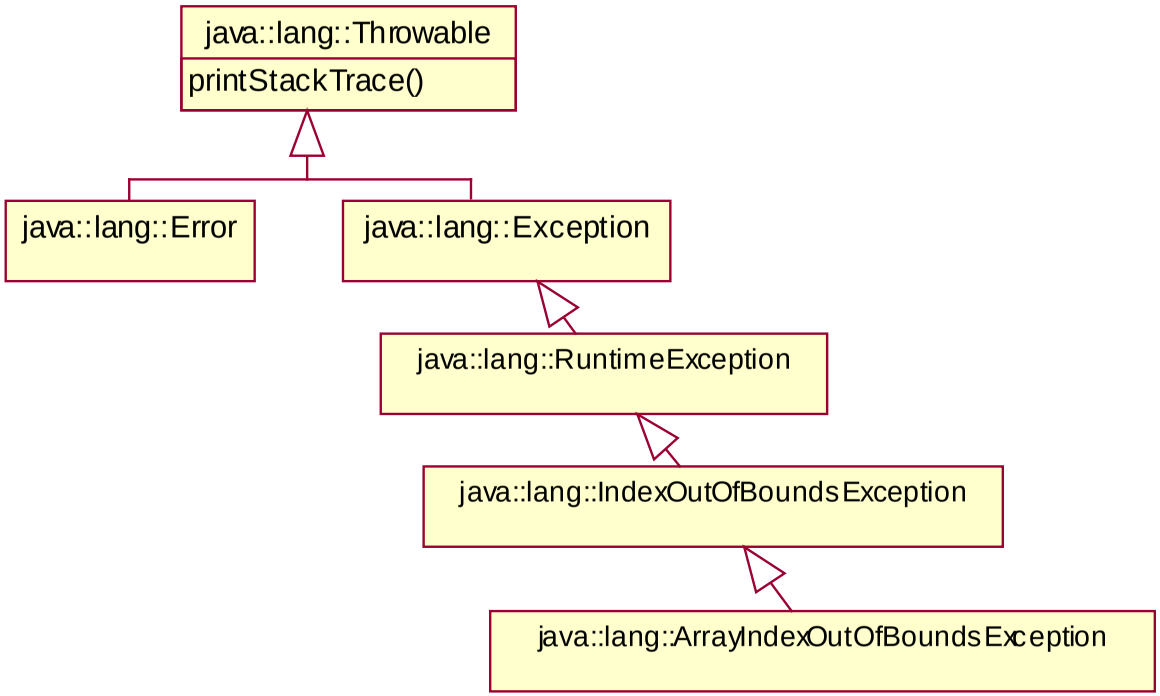
Graphische Darstellung:

– wie Signale nur mit Stereotyp `exception`



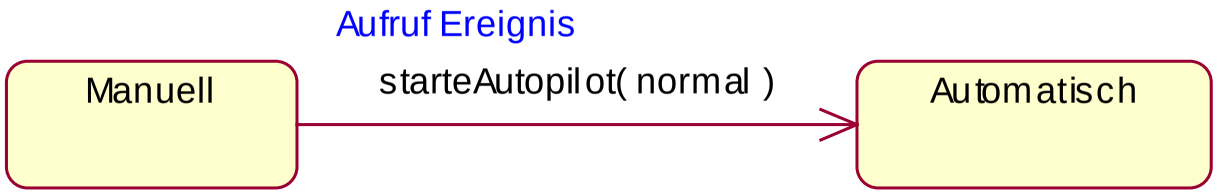
Ereignisse - Ausnahmen (Forts.)

Beispiel: Exceptions in Java



Ereignisse - Aufruf (Call)

- *Aufruf* (Call) = Ausführen einer Operation auf einem Objekt
- Aufruf ist i.a. synchron



Ereignisse - zeitgesteuert

- *Zeitgesteuertes Ereignis* (Time Event) = Ereignis findet nach Verstreichen einer Zeitspanne statt
- **Syntax**:
  `after(Zeitausdruck)`
- Startzeit für Auswertung des Zeitausdrucks = Zeitpunkt, an dem der betreffende Zustand angenommen wird

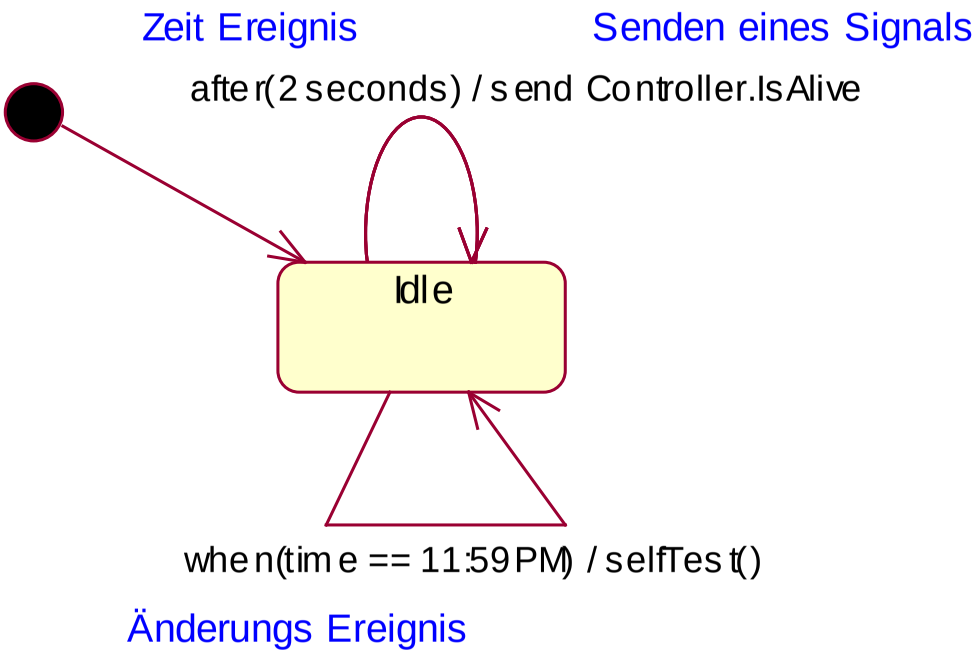
Ereignisse - Änderung

- *Änderungs Ereignis* (Change Event) findet statt, wenn eine überprüfbare Bedingung erfüllt wird
- **Syntax**:

```
when(Boolescher Ausdruck)
```

Ereignisse - Änderung

Beispiel:



Ereignisse - Änderung

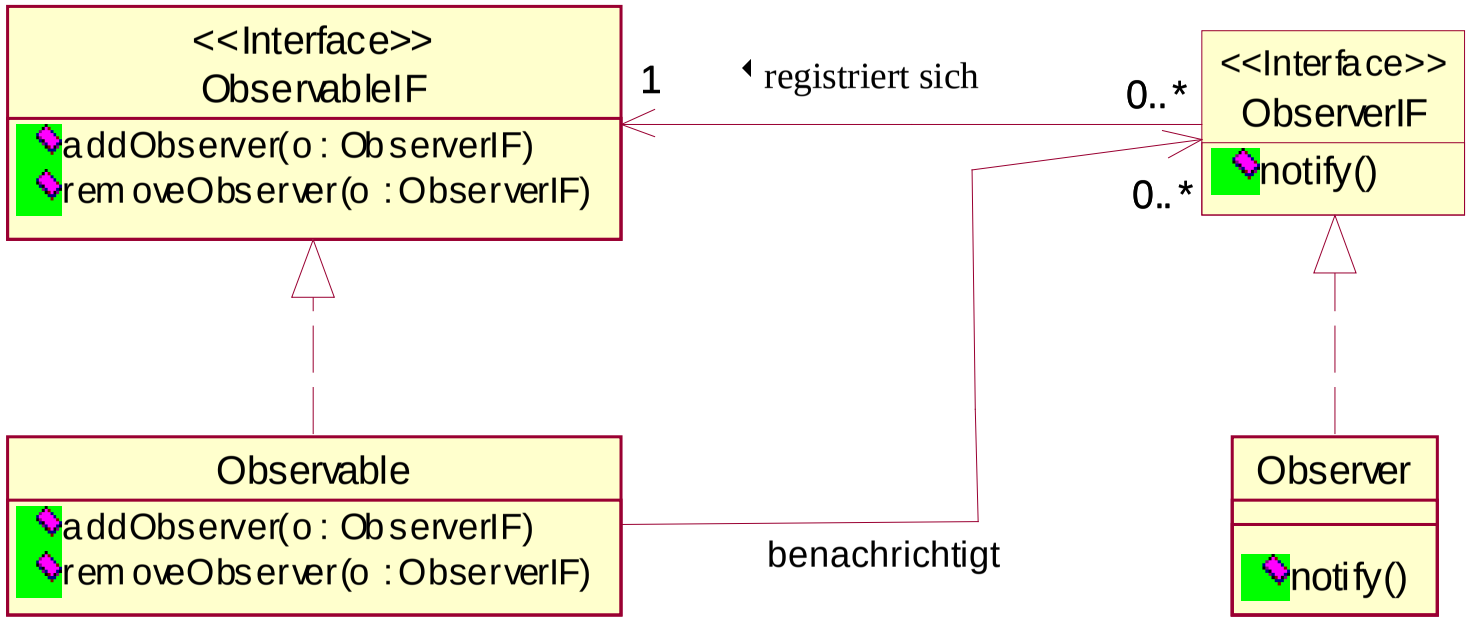
- Änderungs Ereignis impliziert kontinuierliche Überprüfung der Bedingung (ineffizient!)
- Elegante und effiziente Behandlung von Ereignissen durch Benutzung des Observer Entwurfsmusters

Entwurfsmuster - Observer

- *Observer* Entwurfsmuster definiert eine 1 zu *n* Abhängigkeit, so dass bei Zustandsänderung des einen Objektes alle abhängigen Objekte benachrichtigt und aktualisiert werden
- Anwendung:
  - MVC (Model-View-Controller) Konzept
  - für lose Kopplung zwischen abhängigen Objekten

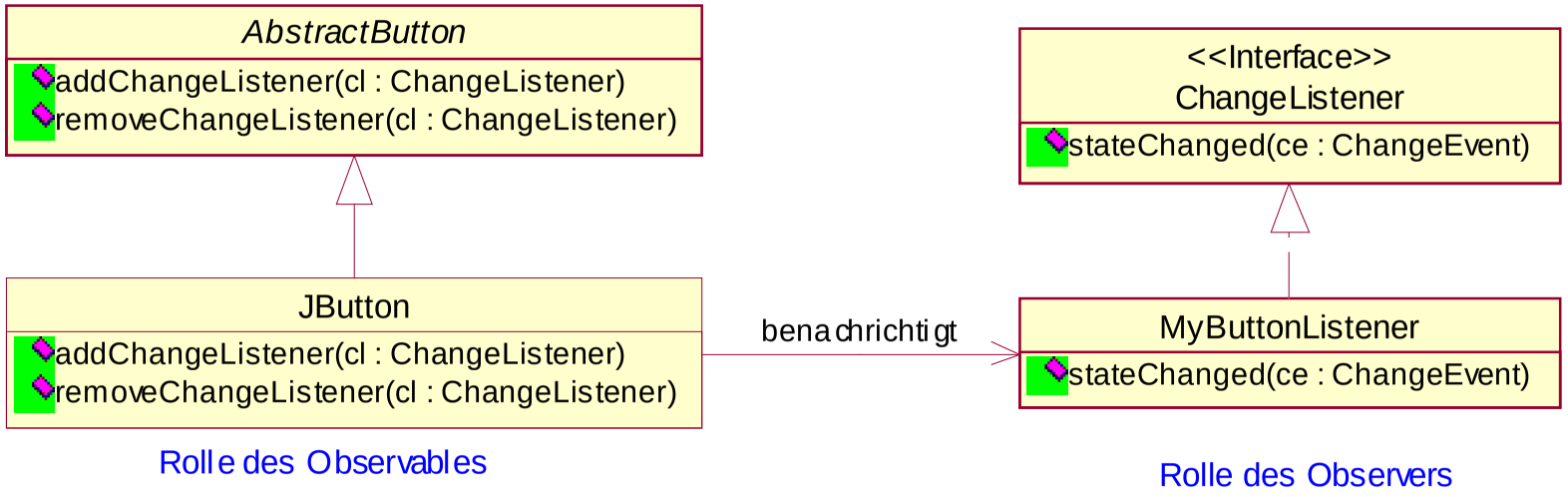
Entwurfsmuster - Observer (Forts.)

Abstrakte Modellierung:



Entwurfsmuster - Observer (Forts.)

- Konsequente Anwendung des Observer Entwurfsmusters in Java
- **Beispiel**: GUI-Komponente `JButton`



Entwurfsmuster - Observer (Forts.)

Implementationsskizze in Java:

```
public abstract class AbstractButton{
   private Vector m_changeListeners;
   ...
   public void addChangeListener(ChangeListener cl){
       m_changeListeners.add(cl);
   }
   public void removeChangeListener(ChangeListener cl){
       m_changeListeners.remove(cl);
   }
```

Entwurfsmuster - Observer (Forts.)

```
    private void notify(){
        Iterator it = m_changeListeners.iterator();
        while(it.hasNext()){
                ChangeListener cl =
                        (ChangeListener)(it.next());
                cl.stateChanged(new ChangeEvent(this));
        }
    }
    ...
} // class AbstractButton
```

3.3.3 Zustände und Übergänge

- *Zustand* (State) = Lage bzw. Situation eines Objektes, in der es
  - eine Bedingung erfüllt oder
  - eine Aktivität ausführt oder
  - auf ein Ereignis wartet
- Zusätzlich gibt es *Startzustand* und *Endzustände* (rein konzeptuell)

Übergang

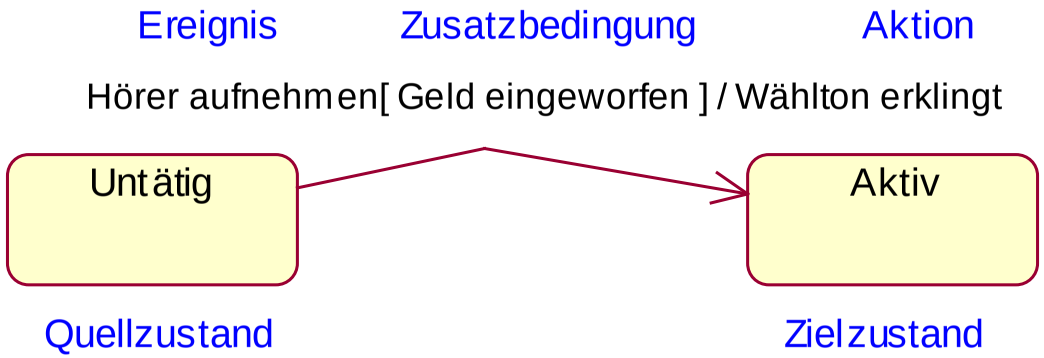
- *Zustandsübergang* (State Transition) = Beziehung zwischen zwei Zuständen (Quell- und Zielzustand)
- Übergang wird *vollzogen* (feuert), wenn ein spezielles Ereignis eintritt und ev. Zusatzbedingungen erfüllt sind

Übergang (Forts.)

- Spezifikation eines Übergangs hat folgende Bestandteile
  - Quellzustand (Source State)
  - Auslösendes Ereignis (Event Trigger)
  - Zusatzbedingung (Guard Condition)
  - Aktion (Action)
  - Zielzustand (Target State)

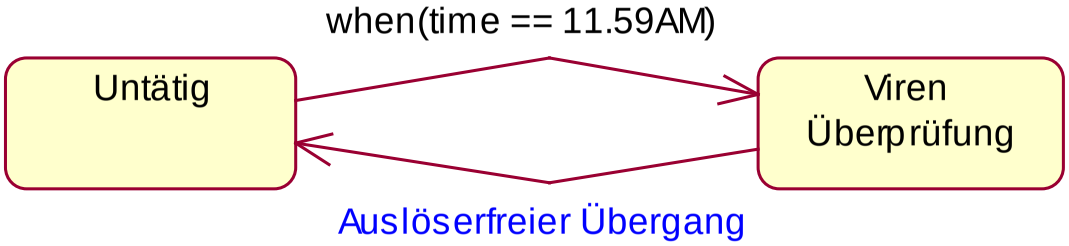
Übergang (Forts.)

- **Graphische Darstellung**:
  - Pfeil von Quell- zum Zielzustand
  - Label mit folgender Syntax:
    `Ereignis(Parameter)[Zusatzbedingung]/Aktionsliste`
- **Beispiel**: Telefonzelle



Übergang - Auslösendes Ereignis

- Ereignisse wie besprochen
  - Signal, Aufruf, zeitabhängig, Zustandsänderung
- *Auslöserfreier Übergang* (Triggerles Transition)
  - wird ausgelöst, wenn Quellzustand seine Aktivität beendet hat



Übergang - Zusatzbedingung

- *Zusatzbedingung* ist Boolescher Ausdruck
- Angabe *disjunkter* und *vollständiger* Zusatzbedingungen ermöglicht für dasselbe auslösende Ereignis verschiedene Übergänge

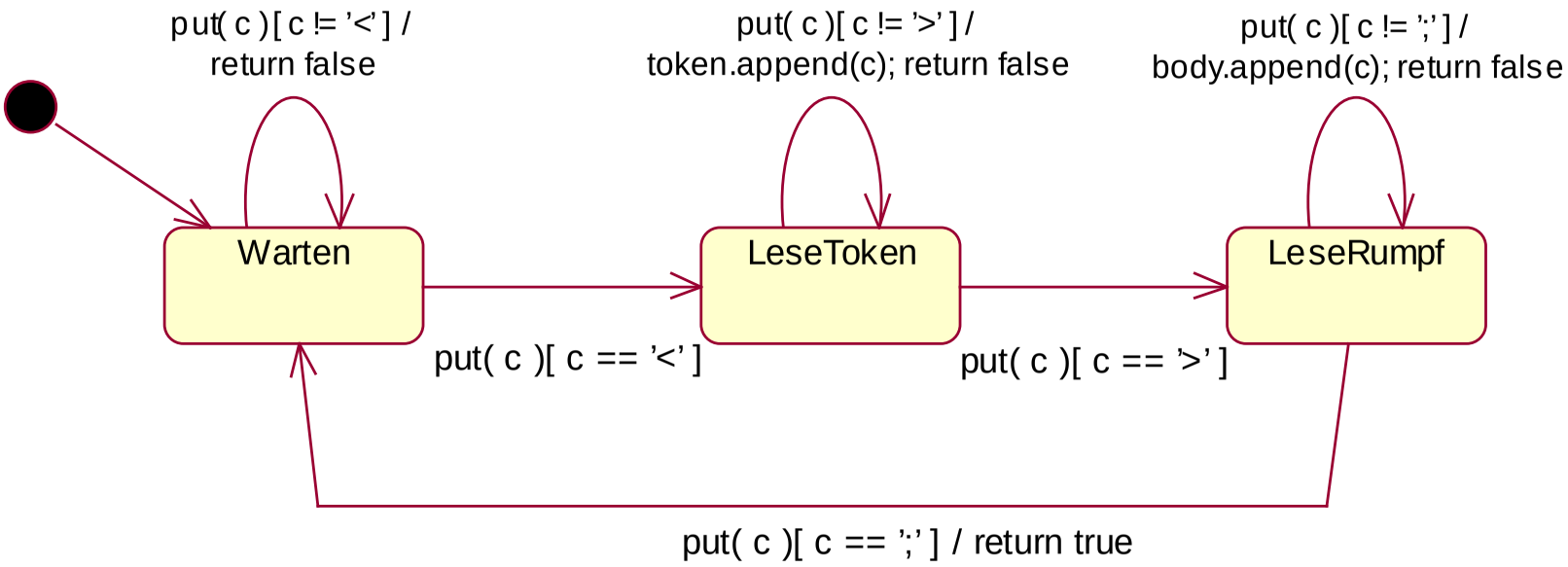
Übergang - Aktion

- Verschiedene Arten von Aktionen:
  - Zuweisung `target:=expression`
  - Aufruf einer Operation `opname(param)`
  - Kreieren eines Objektes `new obj(param)`
  - Zerstören eines Objektes `obj.destroy()`
  - Rückgabe eines Wertes `return value`
  - Verschicken eines Signals `sname(param)`
  - Terminierung `terminate`
  - Uninterpretiert (sprachspezifische Aktion)

Übergang (Forts.)

Beispiel: Einfacher Parser

– akzeptiert Eingaben `<string>string;`

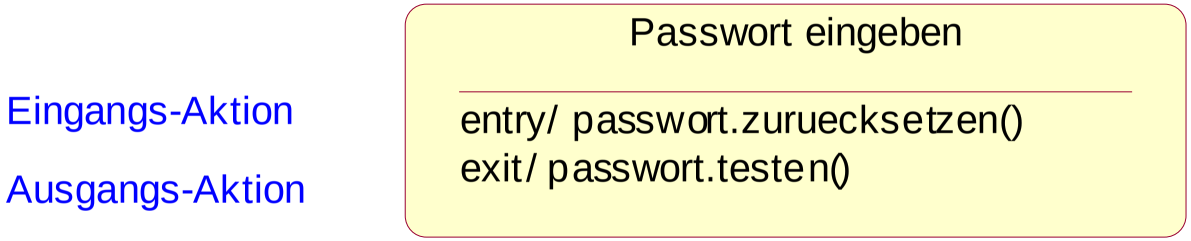


Zustand

- Spezifikation eines Zustands kann folgende Bestandteile haben:
  - Eingangs-/Ausgangsaktion (Entry-/Exit-Action)
  - Interne Übergänge (Internal Transitions)
  - Interne Aktivität (Internal Activity)
  - Aufgeschobene Ereignisse (Deferred Events)
  - Unterzustände (Substates)
- Graphische Darstellung innerhalb des Zustandssymbols

Zustand - Eingangs-/Ausgangsaktion

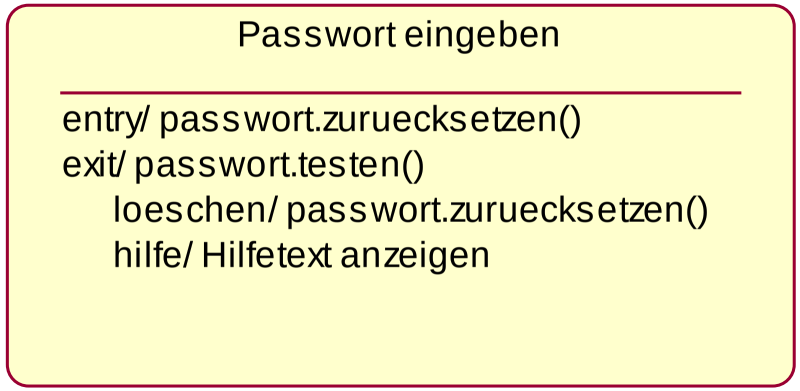
- Ermöglicht Ausführung einer Aktion immer dann, wenn Zustand betreten bzw. verlassen wird
- **Syntax**:
  ```
  entry/Aktionsliste
  exit/Aktionsliste
  ```



Zustand - Interne Übergänge

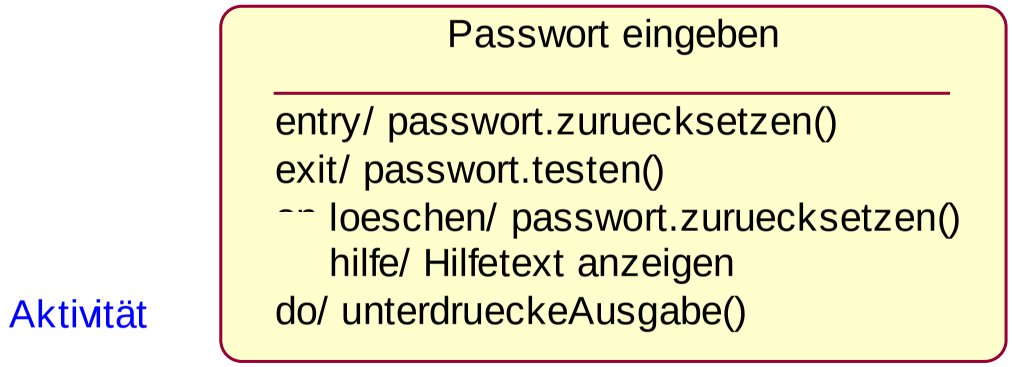
- Verwendung bei Ereignissen, die nicht zu einer Zustandsänderung führen sollen
- **Syntax**:
  `Ereignis/Aktionsliste`

Interne Übergänge



Zustand - interne Aktivität

- *Aktivität* (Activity) = fortlaufende Tätigkeit in einem Zustand
- **Syntax**:
  `do/Aktion`



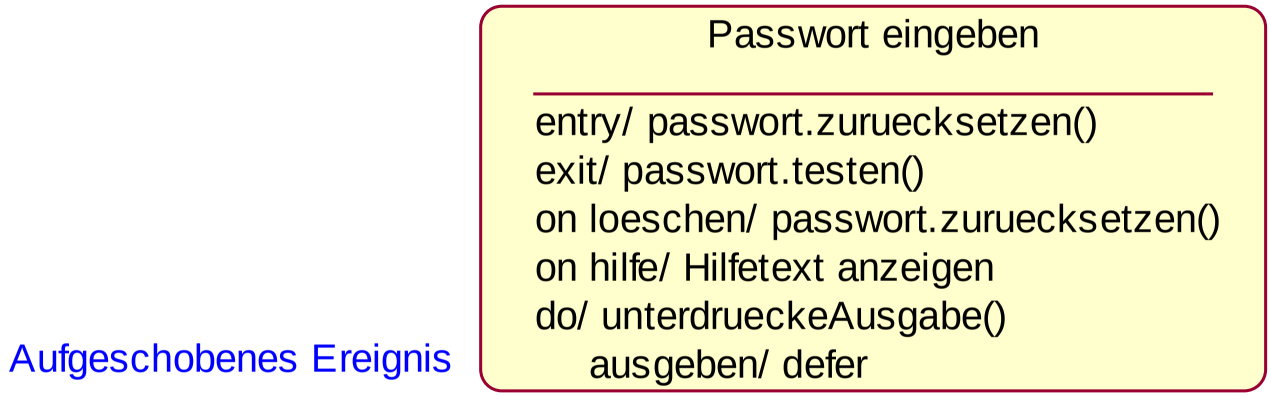
Zustand - aufgeschobenes Ereignis

- Problem:
  - In manchen Situationen kann auf gewisse Ereignisse nicht reagiert werden; wünschenswert ist spätere Behandlung der Ereignisse
- Modellierung:
  - Kennzeichnung dieser Ereignisse mit Aktion `defer`

    $\Rightarrow$ Ereignis wird aufgeschoben, bis Zustand erreicht ist, in dem das Ereignis nicht als defer gekennzeichnet ist

Zustand - aufgeschobenes Ereignis (Forts.)

- **Syntax**:
  `Ereignis/defer`



- Implementierung durch interne Warteschlange

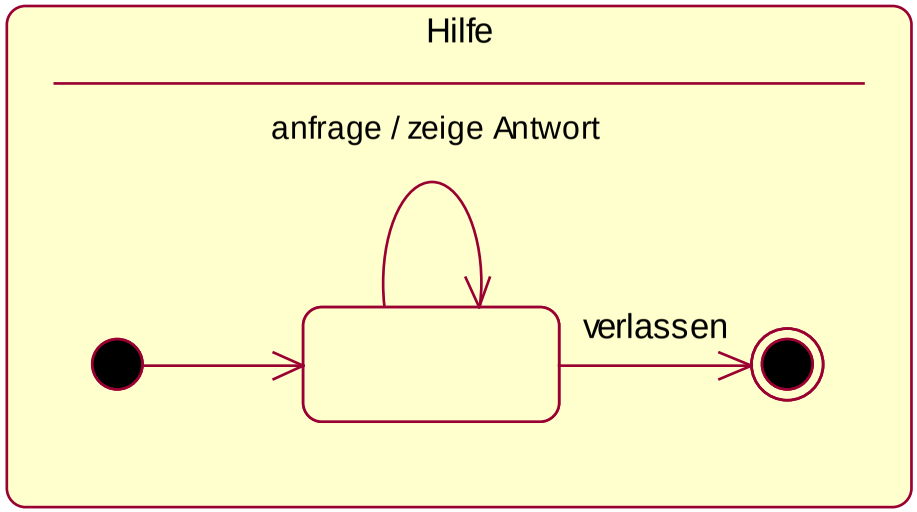
Zustand - Unterzustände

- Zur Modellierung von Zustandsmaschinen auf verschiedenen Abstraktionsebenen werden *Unterzustände* (Substates) verwendet
- *Kompositionszustand* (Composition State) = Zustand, der Unterzustände enthält
- Beliebige Verschachtelungstiefe möglich
- Zustände können *sequentiell* oder *nebenläufig* verlaufen

Zustand - Unterzustände (Forts.)

Graphische Darstellung:

– Kompositionszustand enthält in seinem Inneren die Zustandsmaschine seiner Unterzustände



Zustand - Unterzustände - Übergänge

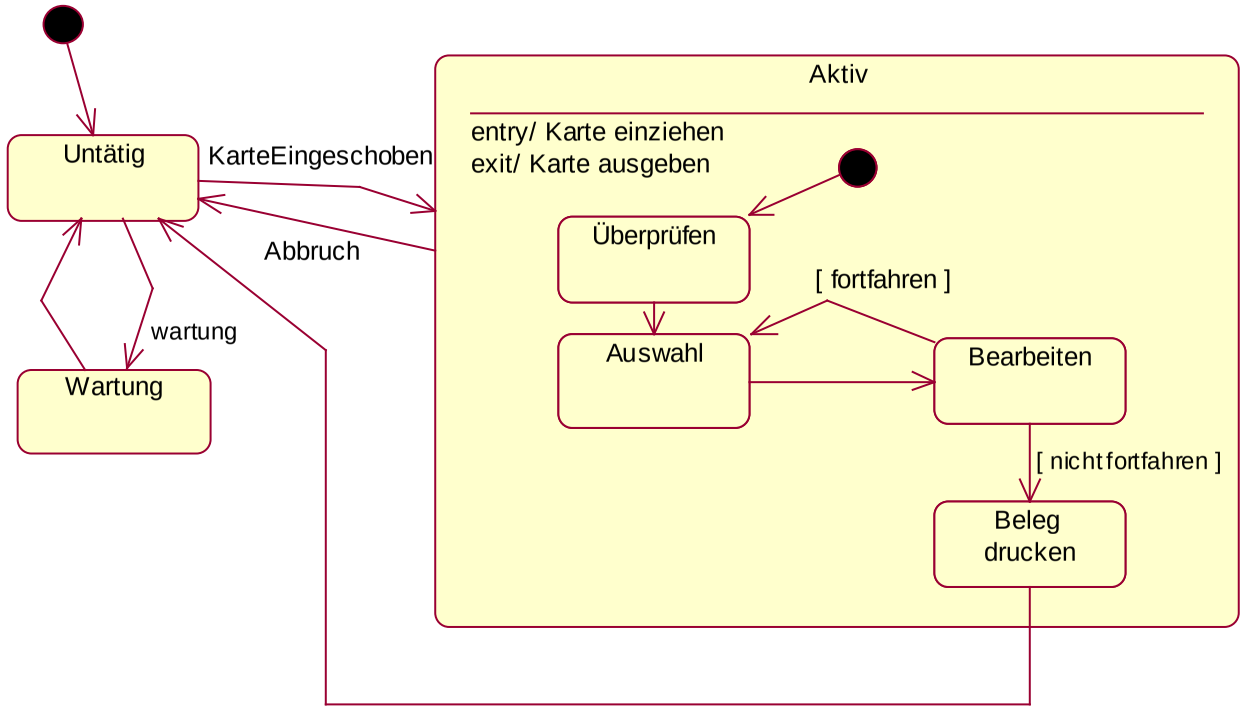
- Übergänge für Kompositionszustände:
  - von ausserhalb zum Kompositionszustand
    - Eingangsaktion des Kompositionszustandes wird ausgeführt; Übergang in Startzustand der eingebetteten Zustandsmaschine
  - von ausserhalb zu einem Unterzustand
    - Eingangsaktion des Kompositionszustandes wird ausgeführt; anschliessend die des Unterzustandes

Zustand - Unterzustände - Übergänge (Forts.)

- von Kompositionszustand nach ausserhalb
  - Übergang ist von jedem Unterzustand aus möglich, d.h. Unterzustände werden unterbrochen; Ausführung der Ausgangsaktion des Kompositions-zustandes
- von Unterzustand nach ausserhalb
  - Ausgangsaktion des Unterzustandes und anschliessend die des Kompositionszustandes werden ausgeführt

Zustand - Unterzustände - Übergänge (Forts.)

Beispiel: Geldautomat



Zustand - Unterzustände - Gedächtniszustände

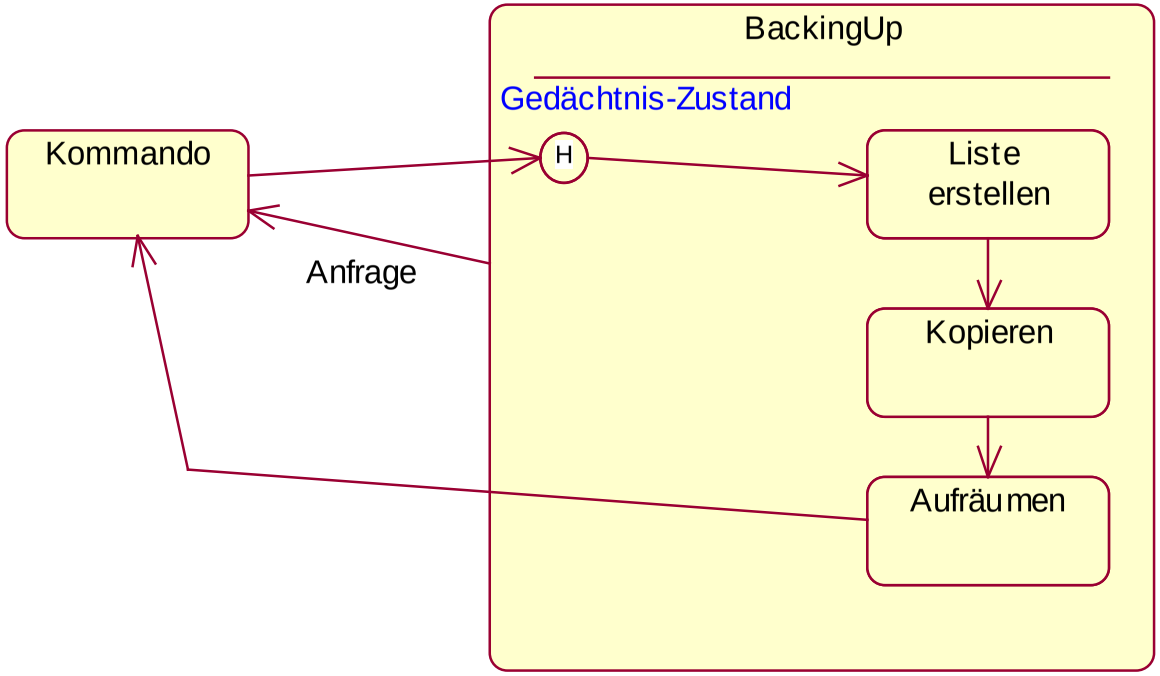
- Problem:
  - Verlassen eines Kompositionszustandes und späteres Wiederbetreten führt zum Startzustand der eingebetteten Zustandsmaschine
  - Wünschenswert ist Aufsetzen in demjenigen Unterzustand, der zuletzt unterbrochen wurde
- Modellierung:
  - Konstrukt des *Gedächtniszustandes* (History State)

Zustand - Unterzustände - Gedächtniszustände (Forts.)

- Eigenschaften:
  - merkt sich bei Übergang von Kompositionszustand aus den zuletzt aktiven Unterzustand; kein Vermerk bei Übergang von Unterzustand aus
  - verzweigt zum gespeicherten Unterzustand, wenn über ihn der Kompositionszustand betreten wird
  - Standard Übergang muss festgelegt werden, für den Fall, dass Gedächtniszustand zum ersten mal betreten wird oder kein Unterzustand gespeichert ist

Zustand - Unterzustände - Gedächtniszustände (Forts.)

Beispiel: Backup Service



Zustand - Unterzustände - Gedächtniszustände (Forts.)

- Gedächtniszustände können auf Zustände in beliebiger Schachtelungstiefe zugreifen
  - Tiefe 1 ⇒ *flacher* (shallow) Gedächtniszustand

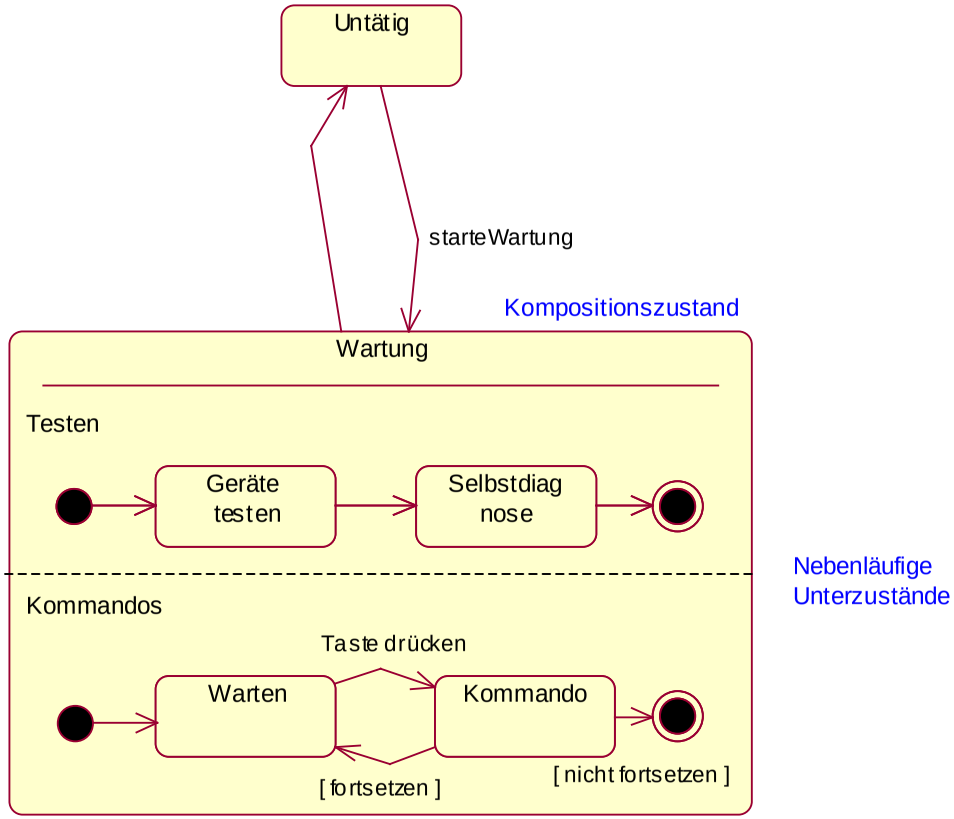
  - Tiefe > 1 ⇒ *tiefer* (deep) Gedächtniszustand

Zustand - Nebenläufige Unterzustände

- In einem Kompositionszustand laufen zwei oder mehrere Zustandsmaschinen parallel ab
  - Beim Betreten des Kompositionszustandes wird in alle Startzustände verzweigt
  - Kompositionszustand wird verlassen, wenn
    - expliziter Übergang nach ausserhalb stattfindet
    - alle parallel ablaufenden Zustandsmaschinen im Endzustand angelangt sind

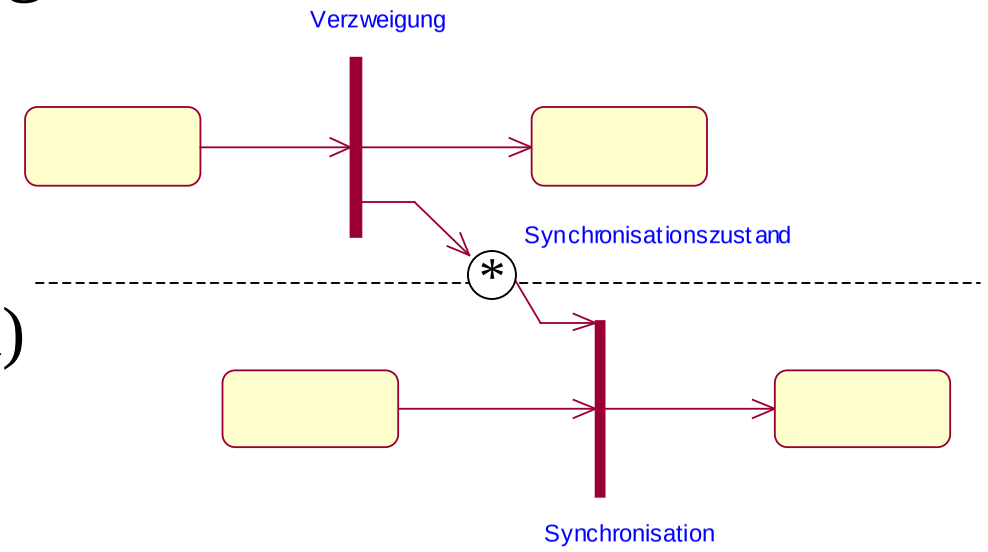
Zustand - Nebenläufige Unterzustände (Forts.)

Beispiel:



Zustand - Synchronisationszustand (Synch state)

- *Snchronisationszustand* = Konstrukt zur Synchronisation nebenläufiger Ausführungs-folgen in Zustandsdiagrammen
  - Quelle ist Verzweigung (Fork)
  - Ziel ist Synchronisation (Join)

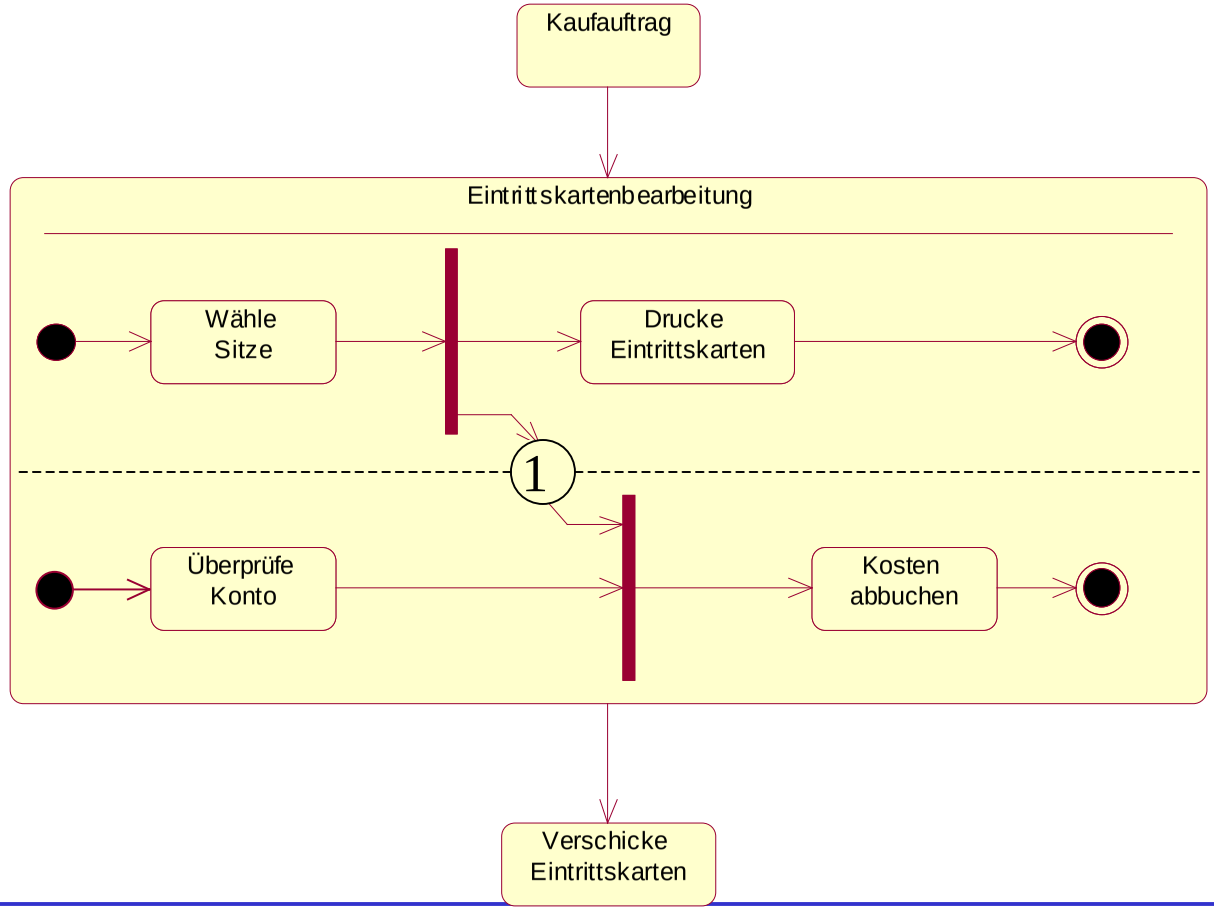


Zustand - Synchronisationszustand (Forts.)

- Synchronisationszustand hat Multiplizität
  - *n* ⇒ es können *n* Synchronisationstoken abgelegt werden (meistens *n* = 1)
  - * ⇒ es können beliebig viele Synchronisations-token abgelegt werden
- Semantik:
  - An dem Synchronisationsbalken kann nur fortgefahren werden, wenn mindestens ein Token im Synchronisationszustand abgelegt ist.

Zustand - Synchronisationszustand (Forts.)

Beispiel:



Zustandsmaschinen-Sicht - Zusammenfassung

- Modellierung von
  - Systemen, deren Verhalten von internem Zustand abhängt
  - detailliertem Verhalten eines Objektes

⇒ Modell des endlichen Automaten erweitert um:

- hybride Aktionen (bei Zustand und Übergang)
- Bedingte Übergänge
- Hierarchie
- Gedächtnis und Nebenläufigkeit