Grundlagen der Objektorientierung

Aktivitätsdiagramme

Was sind Aktivitätsdiagramme?

- Das Notationsmittel der Wahl, um beliebige Abläufe zu modellieren
- Modellierbar sind komplexe Verläufe, Nebenläufigkeiten, alternative Entscheidungswege etc.
- Aktivitätsdiagramme sind Graphen mit gerichteten Kanten.
- Es gibt im Wesentlichen folgende Elemente:
  - Eine oder mehrere (Unter-)**Aktivitäten**
  - **Aktionen**
  - **Objektknoten**
  - **Kontrollelemente** zur Ablaufsteuerung
  - Verbindende **Kanten**
- Sie erinnern an eine Mischung aus Petri-Netz und PAP

Was sind Aktivitätsdiagramme (2)?

- (Unter-)Aktivitäten und auch Aktionen können durch Pseudocode, natürliche Sprache oder Quellcode beschrieben werden.

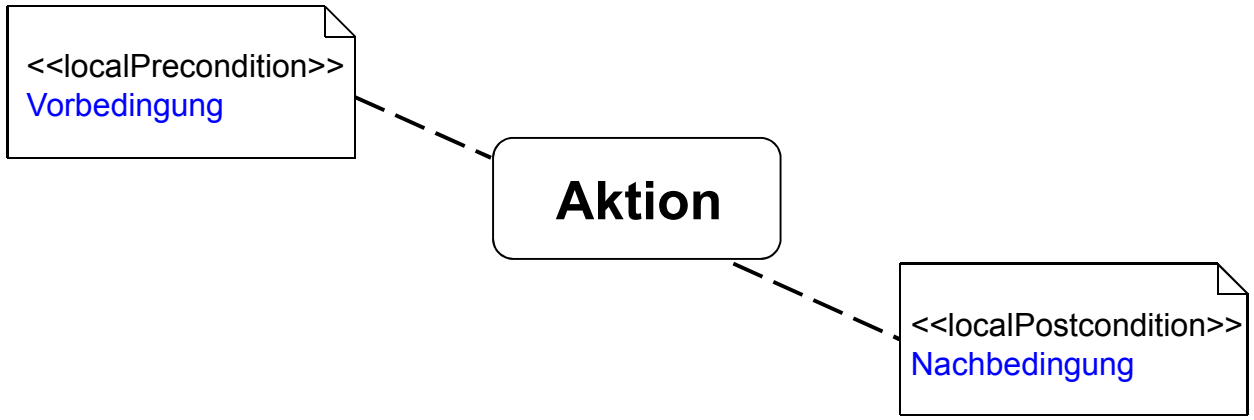
- Aktivitätsdiagramme dienen u. a. zur Darstellung von Abläufen. Der Fokus liegt dabei auf:
  - Parallelitäten / Nebenläufigkeiten
  - Sequenzen
  - Verzweigungen
  - Schleifen

- Aktivitäten, Aktionen und Aktivitätsdiagramme sind entweder
  - einer **Klasse**
  - einer **Operation** (besonders hilfreich für komplexe Operationen)
  - einem **Use-Case** zugeordnet.
    (besonders hilfreich bei Use-Cases mit Parallelität)

Komponenten eines Aktivitätsdiagramms

Aktion

- Zentrales Element eines Aktivitätsdiagramms
- Aufruf eines Verhaltens oder die Bearbeitung von Daten, die innerhalb einer Aktivität nicht weiter zerlegt wird.
- Die Summe aller Aktionen realisiert die Aktivität (incl. Ausführungsreihenfolge, erstellten und verwendeten Daten)
- Vor- und Nachbedingungen werden durch Notizen modelliert

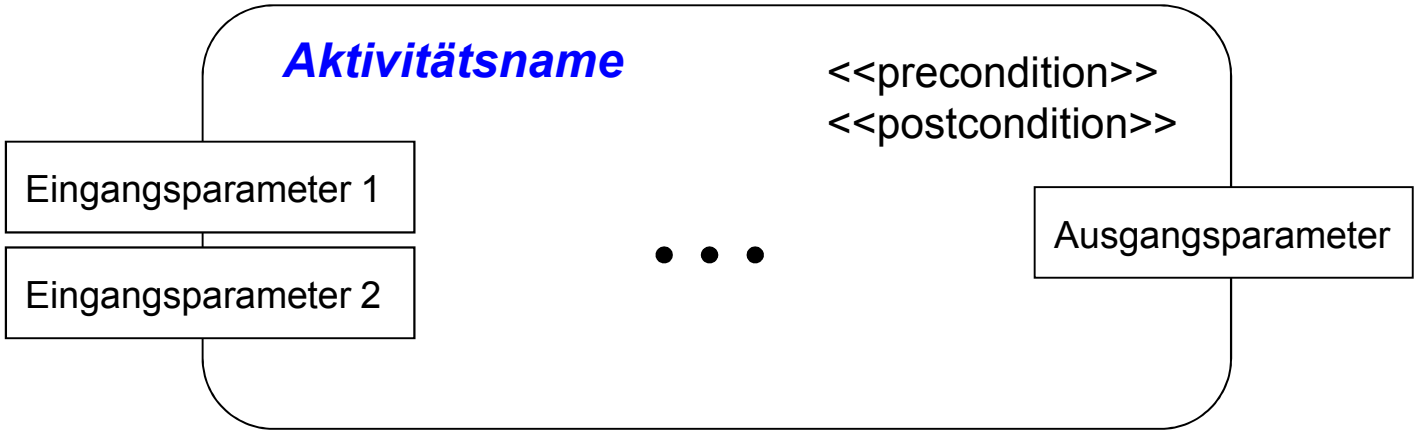


Aktivität

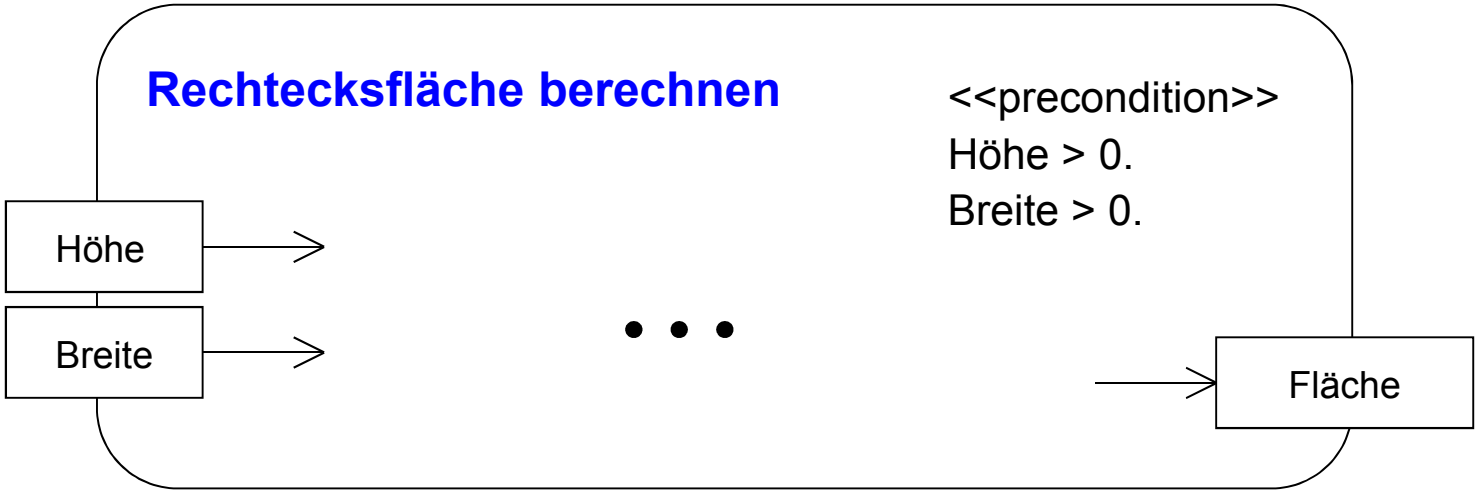
- Gesamte Einheit (von Aktionen etc.) die in einem Aktivitätsmodell modelliert wird.
- Aktivitäten können geschachtelt sein (eine Aktion kann durch eine separate Aktivität darstellbar sein)
- Parameter können in Form von Objekten übergeben werden (Objektknoten)
- Es lassen sich Vor- und Nachbedingungen für den Start und das Ende der Aktivität angeben (**<<*precondition*>>** und **<<*postcondition*>>**)

Aktivität (2)

- Darstellung durch Rechteck mit abgerundeten Ecken.
- Darstellung der Parameter durch Objektknoten auf der Grenze

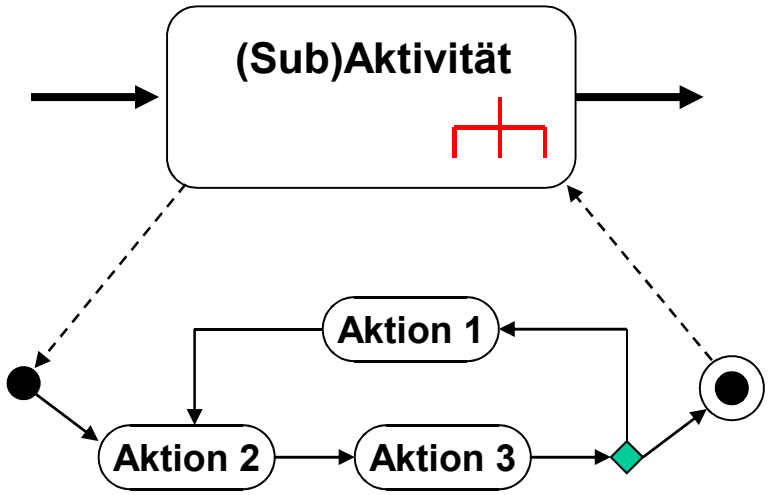


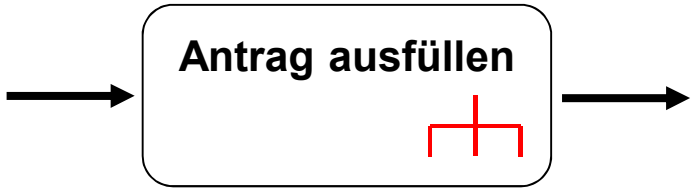
Aktivität (Beispiel)



Schachtelung von Aktivitäten

- Aktionen können weitere Aktivitäten aufrufen (Subaktivitäten) (eigentlich: verfeinerte Aktivitätsdiagramme).
- Wenn eine Subaktivität ausgelöst wird, wird der darin beschriebene (*nested*) Graph abgearbeitet.
- Sie werden erst verlassen, wenn der innere Graph beendet wurde.



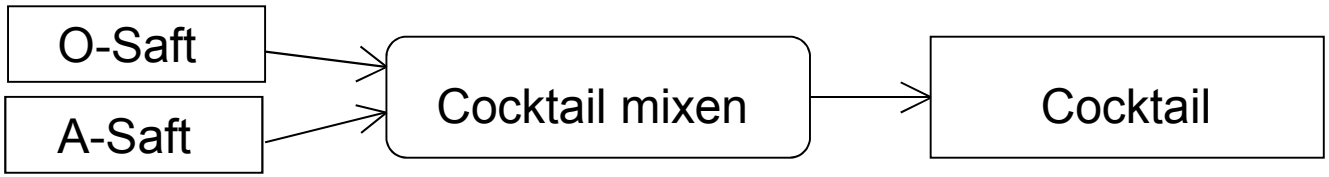


Objektknoten

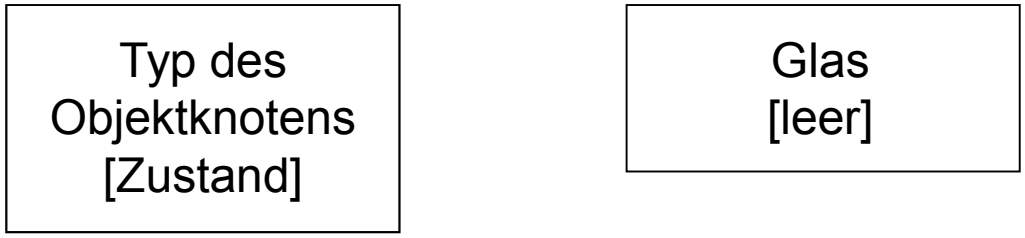
- Repräsentieren Ausprägungen eines bestimmten Typs innerhalb einer Aktivität
- Sind meist primitive Werte oder **Objekte** von Klassen (keine Klassen-instanzen im eigentlichen Sinne ➔ keine Methoden oder Attribute!))
- Bilden das logische Gerüst, um Daten und Werte innerhalb einer Aktivität zu transportieren.
- Können durch die Aktion/Aktivität geändert (Werte) oder erzeugt (Objekte) werden.
- Werden als Rechteck dargestellt

Objektknoten (2)

- Objektknoten als Eingangs- und Ausgangsparameter einer Aktion:

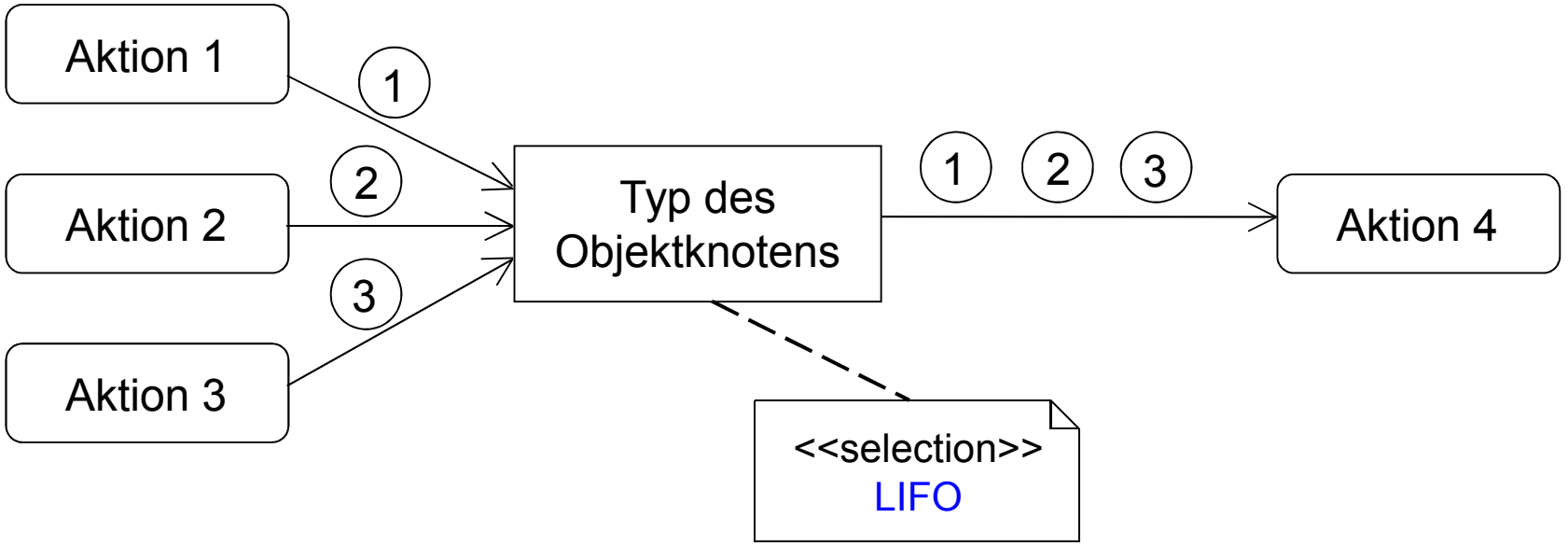


- Objektknoten mit Zustandsspezifikation: Er repräsentiert nur Instanzen, die vom Typ des Objektknotens sind **und** sich in dem Zustand befinden



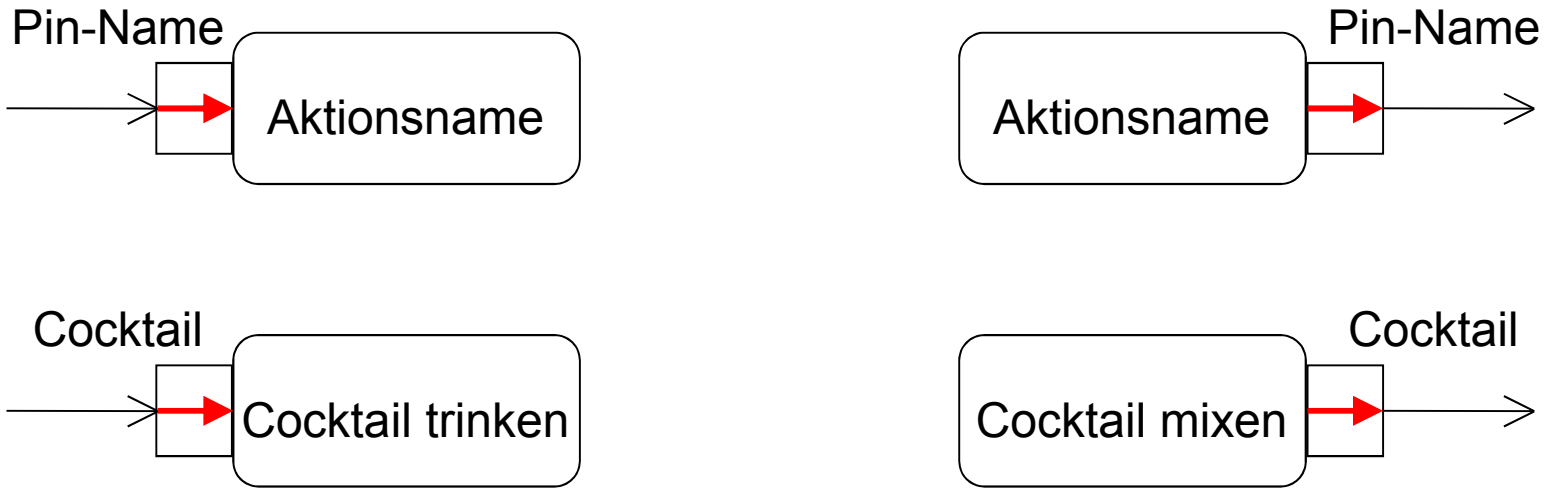
Objektknoten (3)

- Einzelne eingehende Kanten: entspricht dem Verhalten der Aktionen
- mehrere eingehende Kanten: mehrere Tokens werden gesammelt
  ➔ Schema für die Reihenfolge der verlassenden Tokens: „z.B. LIFO“



Objektknoten (4) in Pin-Notation

- Verdeutlichung des Zusammenhangs zwischen einer Aktion und einem Objektknoten als Ein- bzw. Ausgabeparameter:



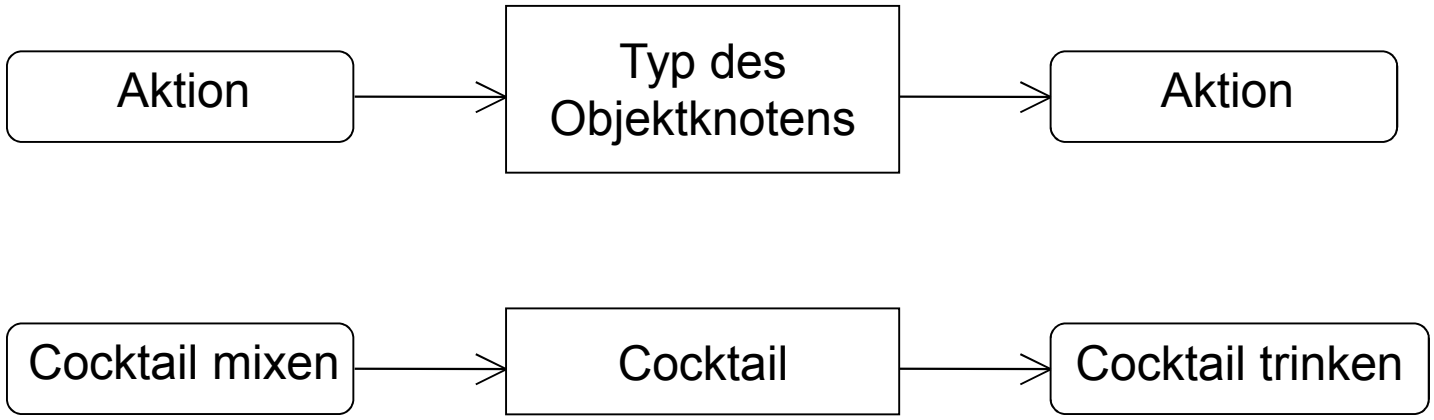
- Richtung der Kanten verdeutlicht, ob Eingangs- oder Ausgangsknoten
  Alternativ: Pfeile in den Pins

Token-Konzept

- Für den Objektfluss verwendbar
- **Logisches** Konzept für nebenläufige Abläufe
- Ursprung: Petri-Netze
- Token lösen einzelne Aktionen aus, d.h. eine Aktion startet dann, wenn ein Token auf der eingehenden Kante angeboten wird
- Werden grafisch **nicht** repräsentiert!

Objektfluss zwischen Aktionen

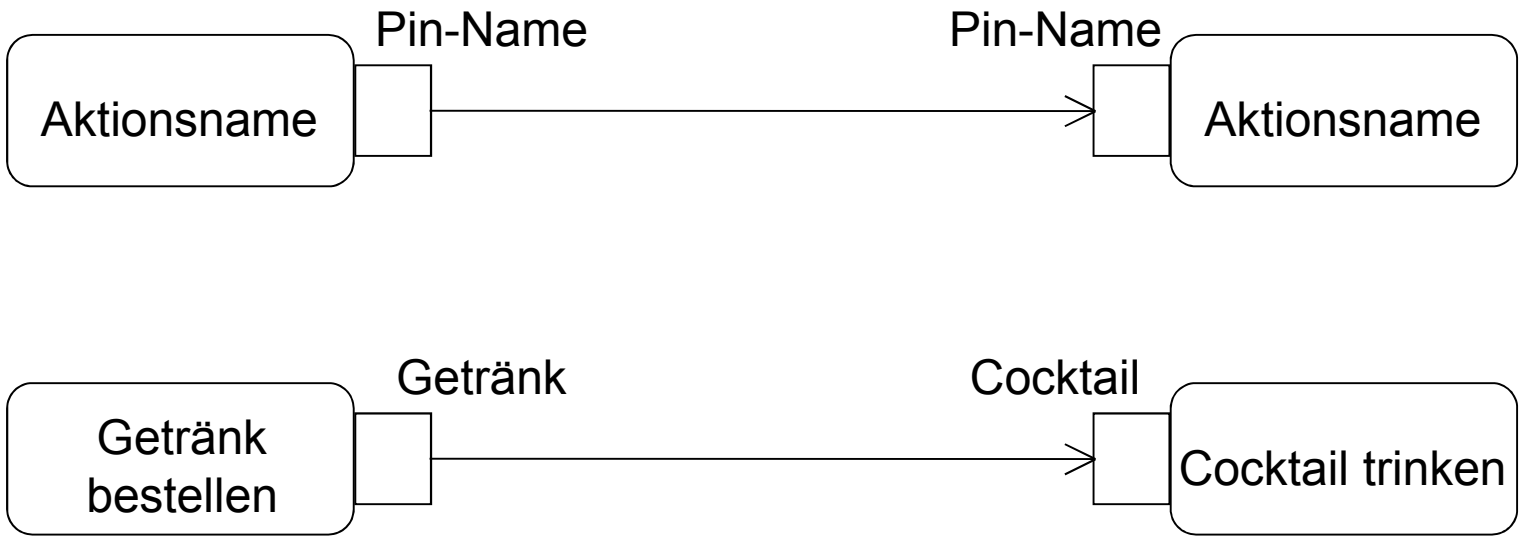
- Ein Objektknoten, der gleichzeitig Aus- und Eingabeparameter ist, kann ohne die Pin-Notation dargestellt werden:



Parameter müssen gleichen Namen und gleichen Typ besitzen!

Objektfluss zwischen Aktionen

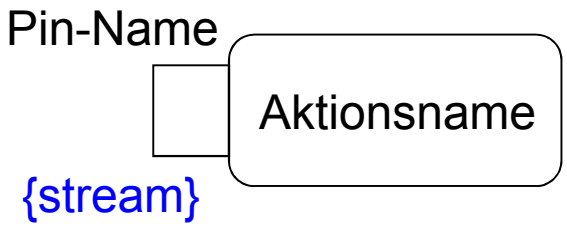
- Liegen unterschiedliche Typen und/oder Namen vor: **Pin-Notation**

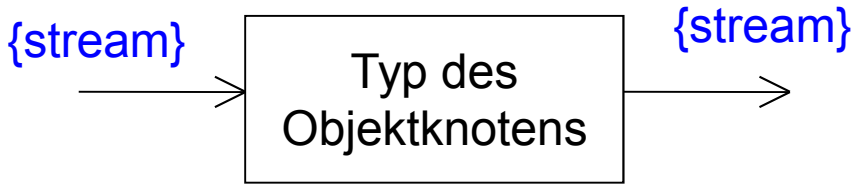


Typen der Objektknoten müssen kompatibel sein!

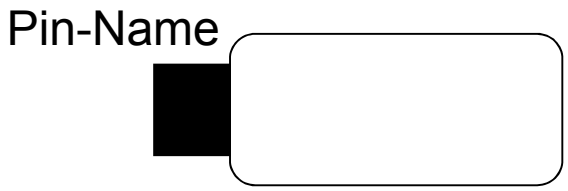
Objektfluss zwischen Aktionen - Streaming

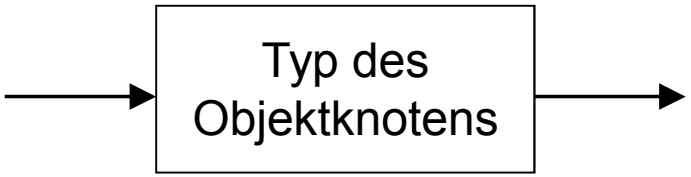
- Besondere Art: **Streaming Modus (kontinuierlicher Tokenfluss)**
- Daten-Token fließen auch dann (fortwährend) in oder aus einer Aktion, wenn diese gerade aktiv ist





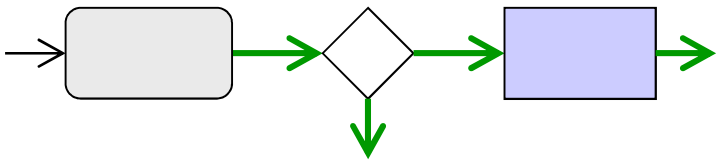
Alternativ:



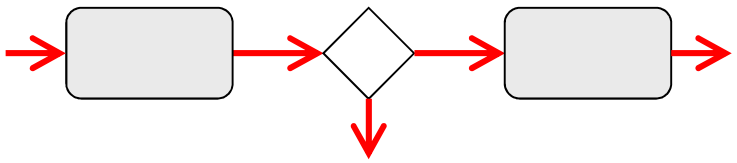


Kanten

- Übergang zwischen zwei Knoten (Aktionen, Objektknoten)
- Kanten sind immer gerichtet und können mit Namen versehen werden
- **Kontrollflüsse:**
  Kante zwischen zwei Aktionen oder zwischen einer Aktion und einem Kontrollelement. Tokens tragen keine Werte oder Daten.
- **Objektflüsse:**
  Hier ist immer mindestens ein Objektknoten beteiligt.
  Tokens enthalten Werte oder Daten



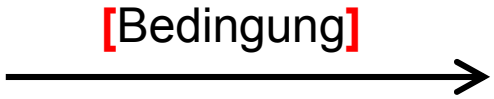
Objektflüsse



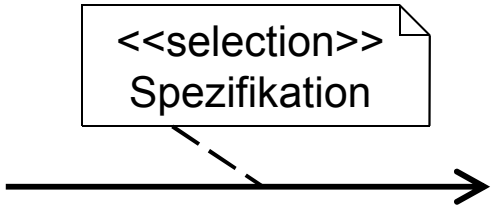
Kontrollflüsse

Kanten

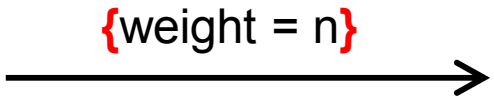
- **Kanten mit Bedingungen:**



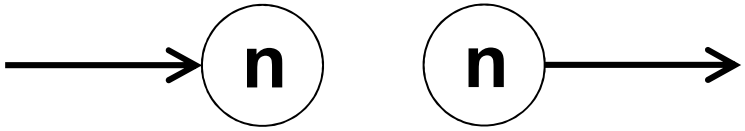
- **Sortierungskriterien:**



- **Gewichtete Kanten:**

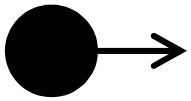


- **Sprungmarken**

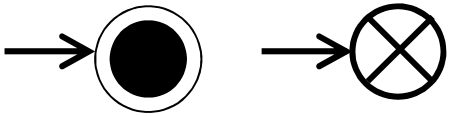


Kontrollelemente

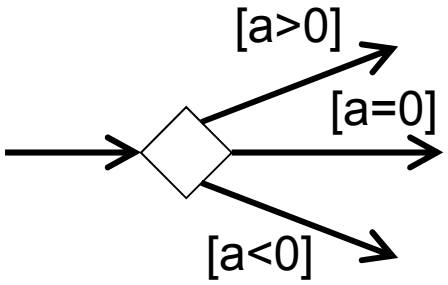
- **Startknoten**



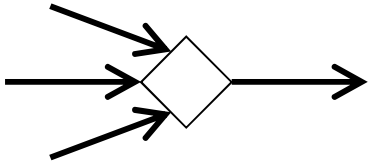
- **Endknoten:**



- **Verzweigungsknoten:**
  (optional mit Bedingungen)

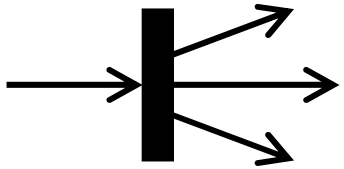


- **Verbindungsknoten:**

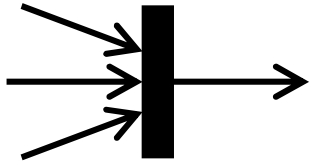


Kontrollelemente (2)

- **Parallelisierungsknoten:**



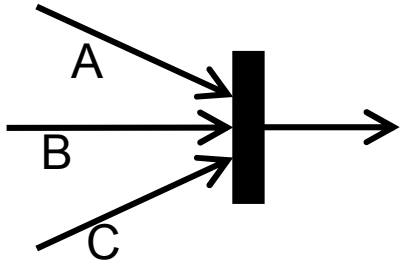
- **Synchronisationsknoten: (implizites AND)**



- **Synchronisationsknoten mit Synchronisations- Spezifikation:**

anderes Bsp.:
{joinspec = (A and B) xor (A and C)**}**

{joinspec = A or B or C}



(erweitertes) Beispiel aus der UML-Spezifikation

Use-Case „Getränk vorbereiten“

Eine Person möchte je nach Situation entweder Kaffee oder Cola trinken.
Dabei muss der Kaffee erst in einer einfachen Kaffeemaschine zubereitet bzw. die Flasche geöffnet werden. Einige Aktionen können „parallel“ ausgeführt werden

Dieser Use-Case kann mit folgenden Aktionen modelliert werden:

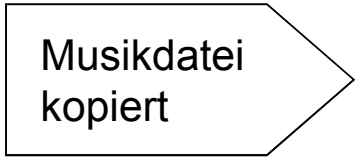
- Getränk aussuchen
- Kaffee in den Filter füllen
- Filter in Maschine setzen
- Wasser in den Wasserbehälter gießen
- Tasse holen
- Glas holen
- Colaflasche holen
- Maschine einschalten
- Kaffee aufbrühen
- Kaffee einschenken
- Evtl. Milch und/oder Zucker dem Kaffee zugeben
- Cola einschenken
- Maschine ausschalten
- Trinken

Versuchen Sie, die einzelnen Zustände ebenfalls zu modellieren

Besondere Komponenten

Signale und Ereignisse

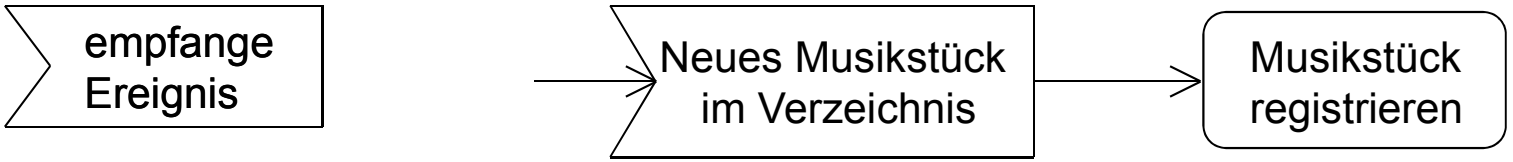
- Sonderformen einer Aktion (***SendSignalAction***, ***AcceptEventAction***)
- Der Signalsender erstellt aus seinen Eingabedaten ein Signal, das an einen Ereignisempfänger gesendet wird.
- Das verschickte Signal darf Daten „transportieren“



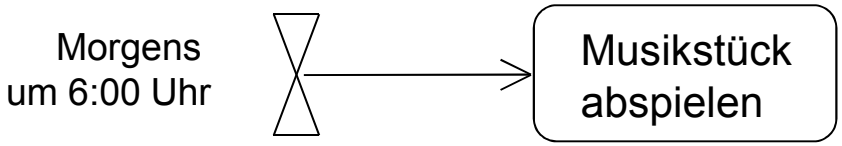
Signale und Ereignisse (2)

- Ereignisempfänger kann zur Synchronisation verwendet werden:

  erreicht ein Ablauf einen EE, dann verharrt er (bzw. das Token) in der Aktion, bis das Ereignis eintrifft. Anschließend wird die Abarbeitung fortgesetzt

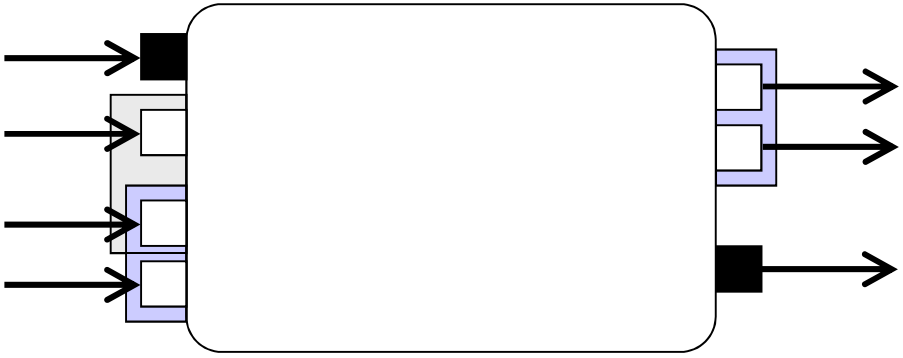


- Ereignisse können auch zeitlich initiiert werden („alle 5 Min“, „um 20 Uhr“)



Parametersätze

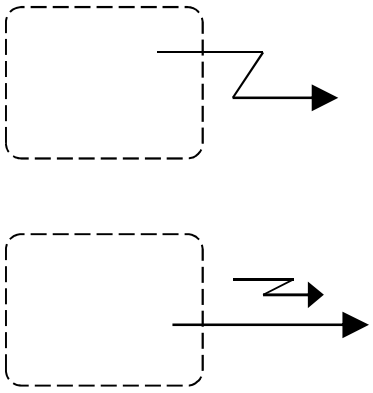
- Zur Gruppierung von Ein- und Ausgabeparametern.
- Nur „sortenreine“ Ein- und Ausgabeparameter gruppierbar
- Ein Pin darf zu mehreren Gruppen gehören
- Gemeinsame Überführung der Daten-Tokens, wenn an **ALLEN** Pins eines Parametersatzes Daten-Tokens vorliegen (AND-Beziehung)
- Eine Aktion wird pro Aufruf von genau einem Parametersatz bedient (XOR)
  ➔ nicht an sämtlichen Pins müssen Daten-Tokens vorliegen
- Kennzeichnung aller Parameter **ohne** Gruppenzugehörigkeit als **Streamings**



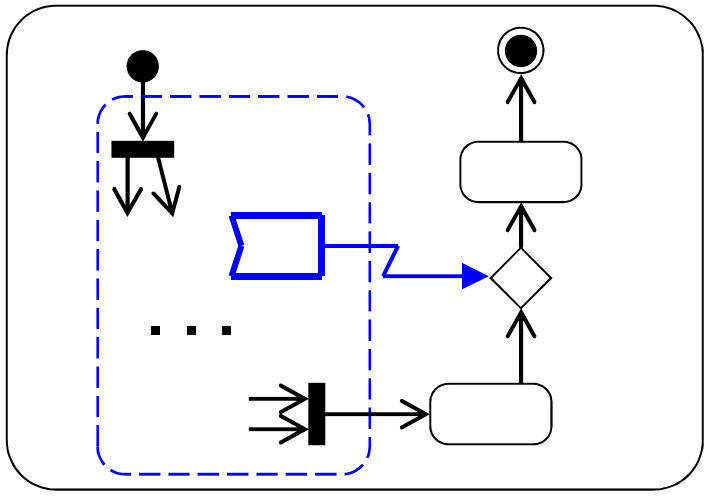
Unterbrechungsbereich

- Darstellung bzw. Abgrenzung eines Bereichs, der durch *Exceptions* verlassen werden kann
- Zur sofortigen Beendigung mehrerer Aktionen (durch Abbruch)
- Alle in diesem Bereich vorhandenen Tokens werden verworfen

- **Notation:**

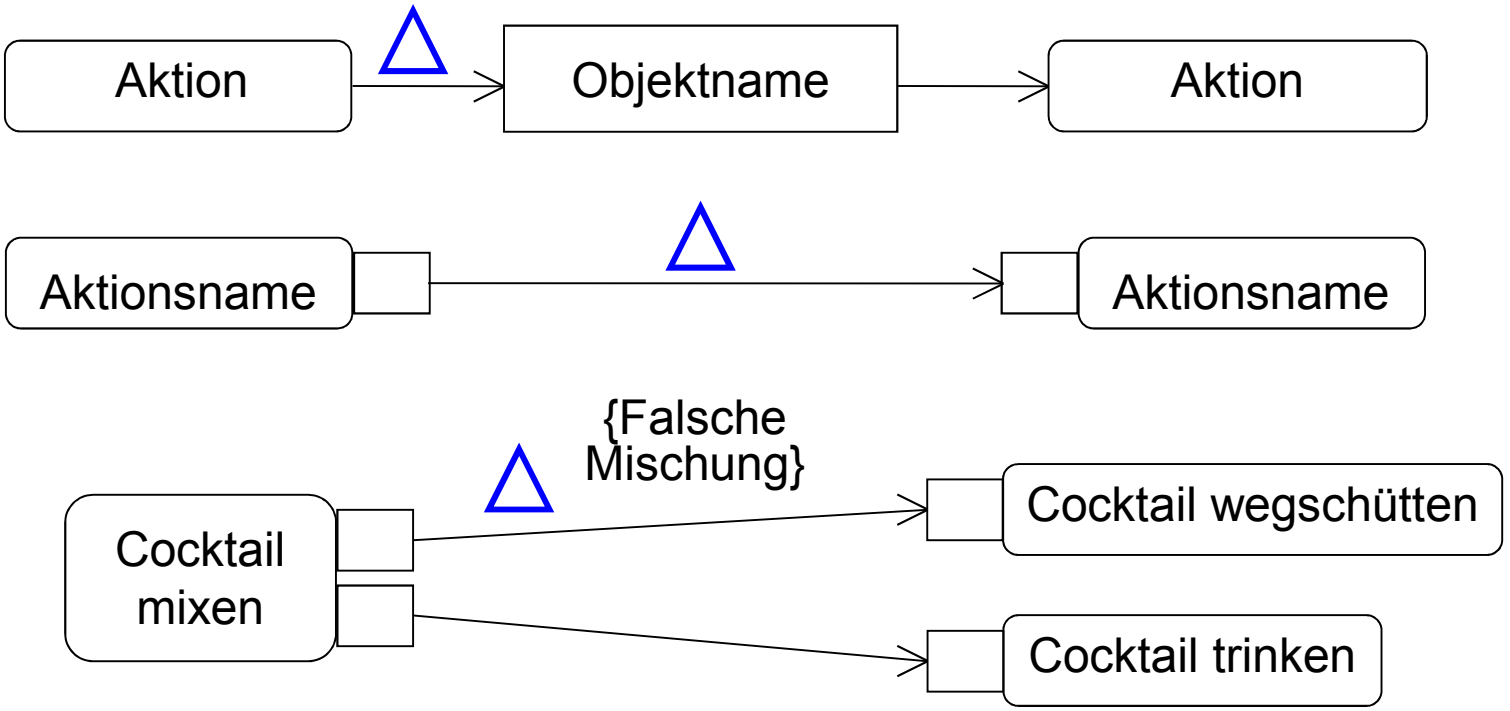


Beispiel:



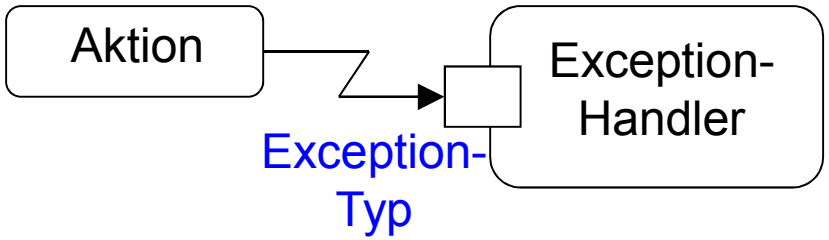
Exception-Objekte

- Variante von Ausgabeparametern
- Die folgende Aktion wird nur ausgeführt, wenn die angegebene Exception geworfen wurde.



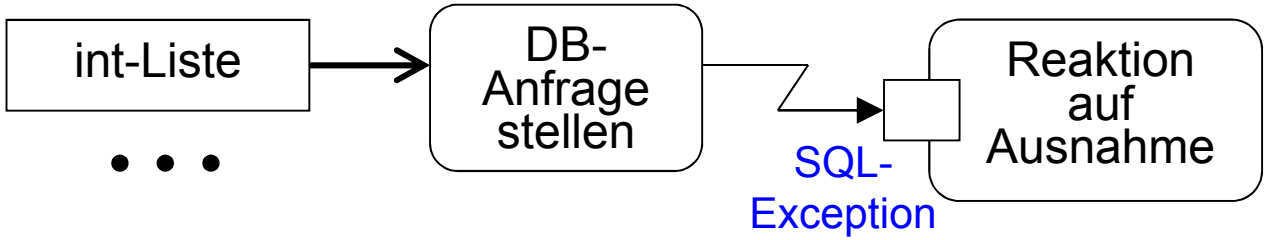
Exception-Handler

- Zur Bearbeitung vordefinierter Ausnahmen (Exceptions), die während der Ausführung einer Aktion auftreten (analog zum Exception-Handling bei Java)
- Eine Aktion kann mehrere Ausnahmen werfen
- **Notation:**

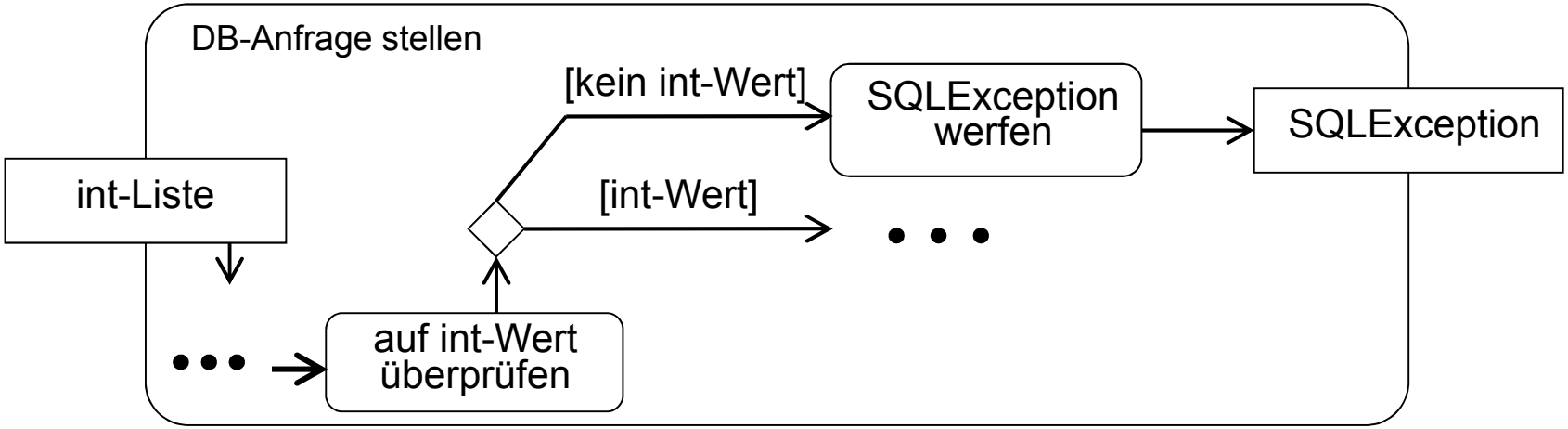


Exception-Handler (2)

Beispiel: SQL-Befehl aus einer Integer-Liste generieren und Datenbank abfragen

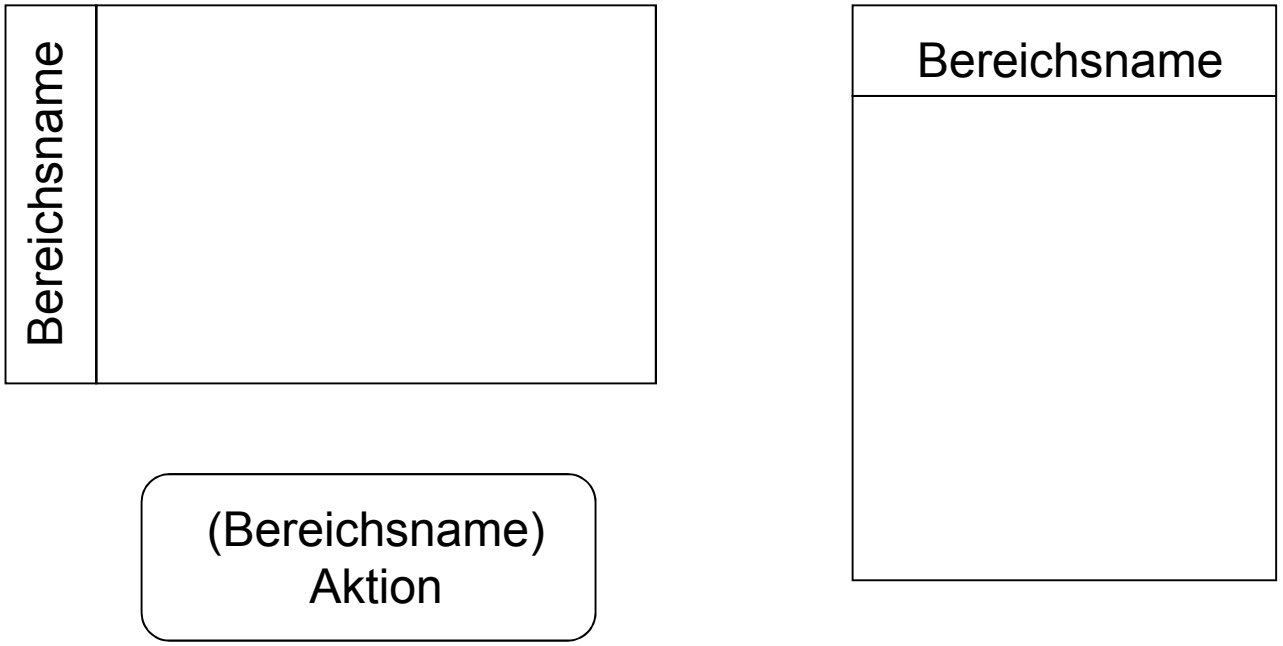


Modellierung innerhalb der Aktion:



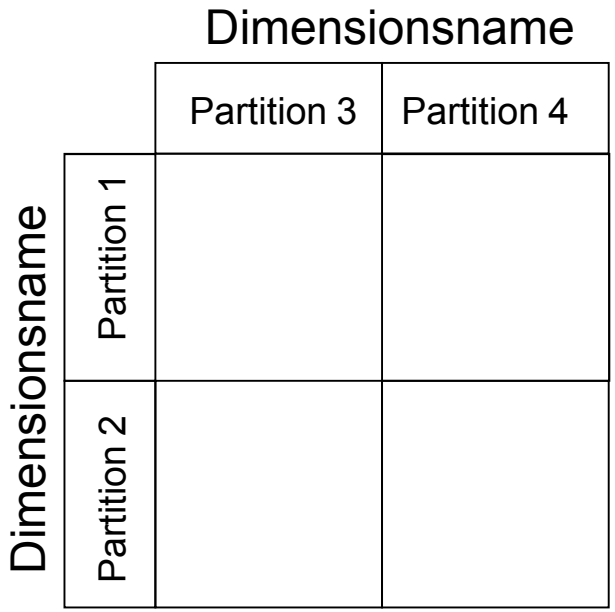
Aktivitätsbereiche

- Unterteilung von Aktivitäten in Bereiche mit gemeinsamen Eigenschaften (z. B.: Standort, Abteilung, Rolle, Verantwortlichkeit, Subsystem, ...)
- Keine semantische Veränderung, nur optische Verbesserung
- **Notation:**

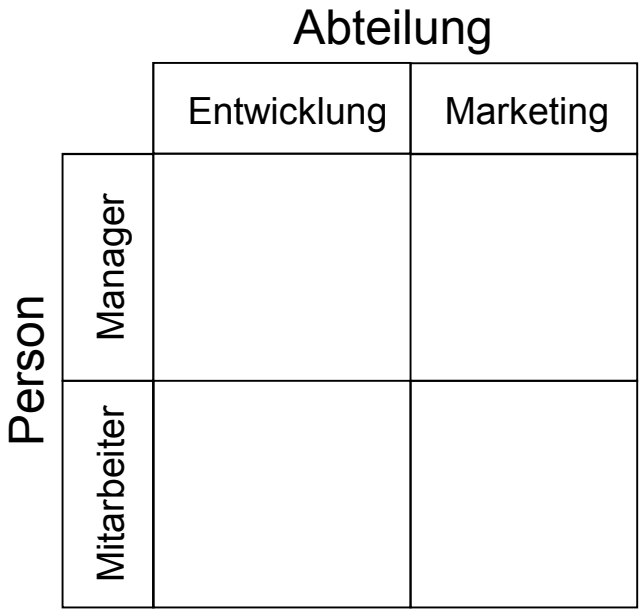


Aktivitätsbereiche (2)

- **Mehrdimensionale Aktivitätsbereiche**



(Partition1, Partition 3)
Aktion



(Manager, Marketing)
Aktion

Aktivitätsbereiche (2)

- **Hierarchische Aktivitätsbereiche**

Dimensionsname

Partitionsname	
Unterpartition	Unterpartition

Person

Manager	Mitarbeiter	
	Entwicklung	Vertrieb

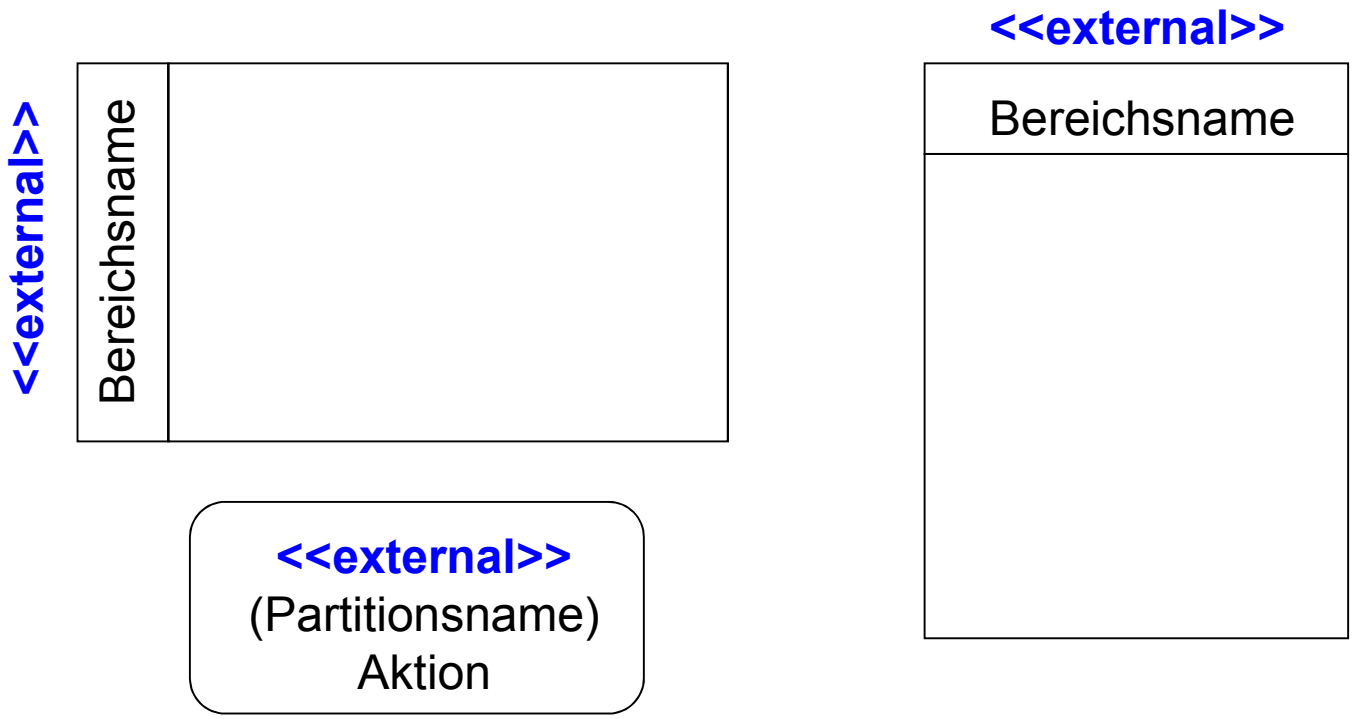
(Partition::Unterpartition)
Aktion

(Mitarbeiter::Entwicklung)
Aktion

Aktivitätsbereiche (3)

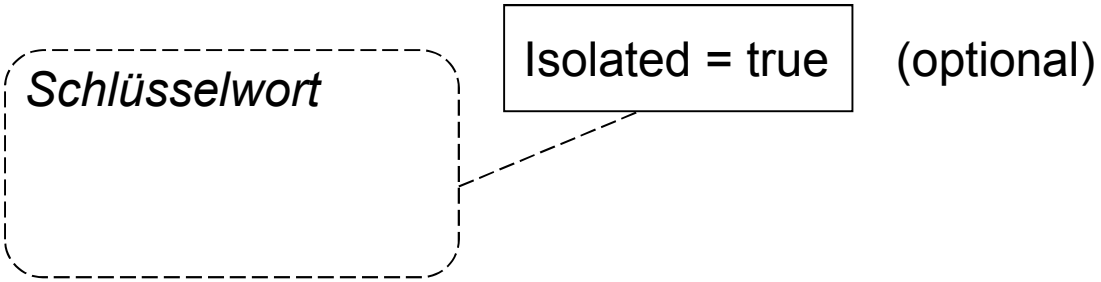
- **Externe Aktivitätsbereiche**
  Integration von Bereichen, die eigentlich nicht zum Diagramm dazugehören bzw. nicht dort modelliert werden

- **Notation:**



Strukturierte Knoten

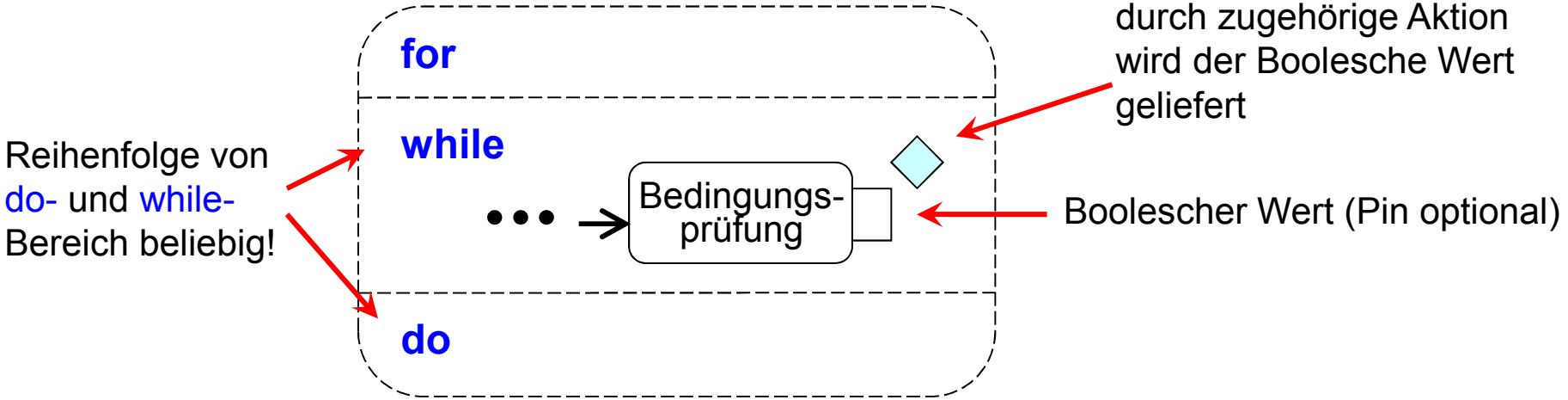
- **Zwei Funktionen:**
  1. Gruppierung von Elementen zur Strukturierung
  2. Verwendung als ausführbare Knoten (eingebettete Aktionen)
     ➔ sie dürfen auch Objektknoten besitzen (Pins)
- Schachtelung ist möglich, Elemente können jedoch nur einem Knoten direkt zugeordnet werden
- Zugriffsschutz für gemeinsam genutzte Elemente durch Isolation (opt.)
- Notation:



Schleifenknoten

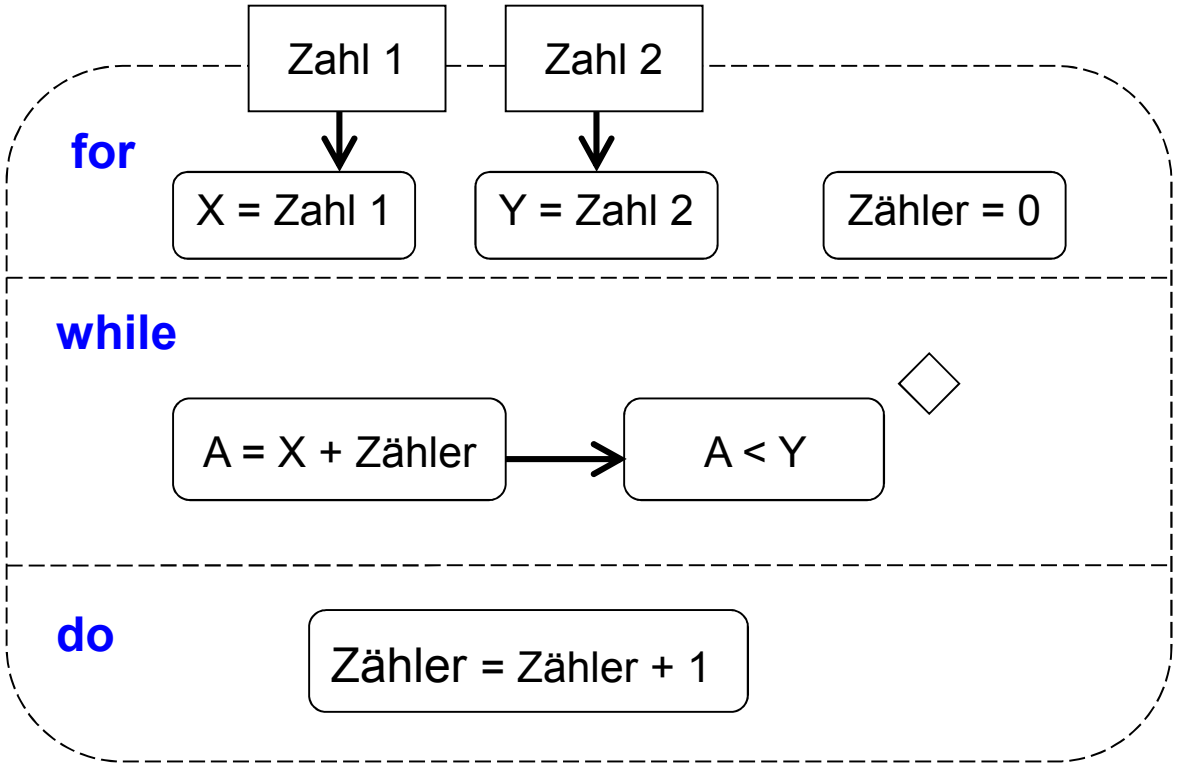
- Spezieller strukturierter Knoten mit Aufteilung in drei Bereiche:
  1. **for**-Bereich (optional), Beschreibung der initialen Aufgaben. Der for-Bereich wird genau einmal durchlaufen.
  2. **while**-Bereich (optional), enthält die Elemente, die überprüfen, ob der Schleifenrumpf (nochmals) durchlaufen wird
     (Ergebnis: boolesche Variable; Pin an Aktion, die Bed. prüft sowie Raute)
  3. **do**-Bereich, enthält eigentlichen Schleifenrumpf

- Notation:



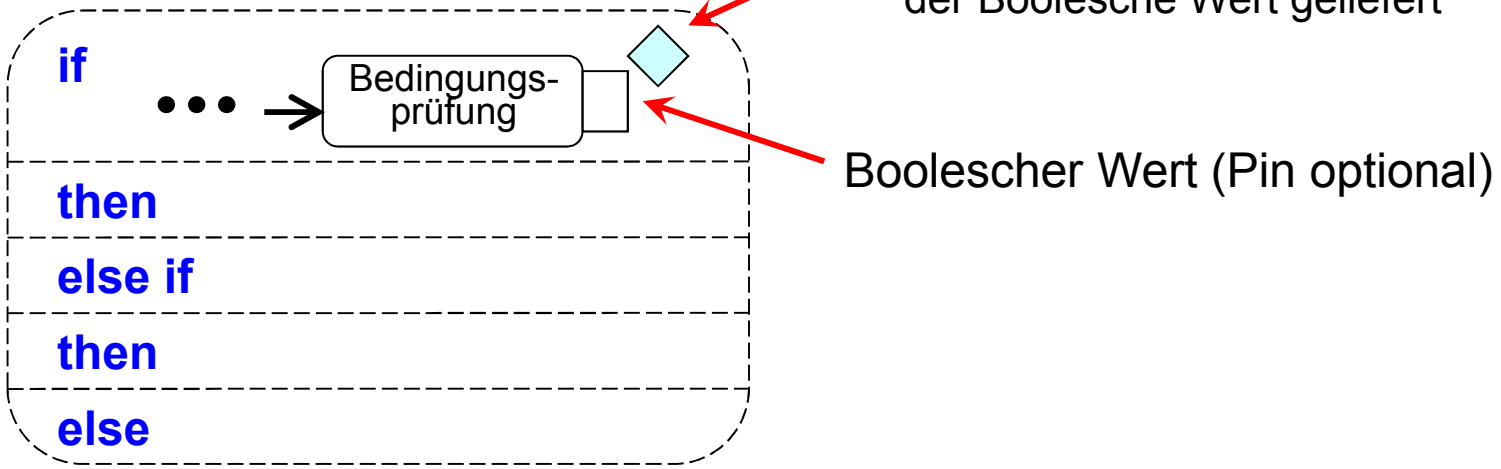
Schleifenknoten (2)

- Beispiel:



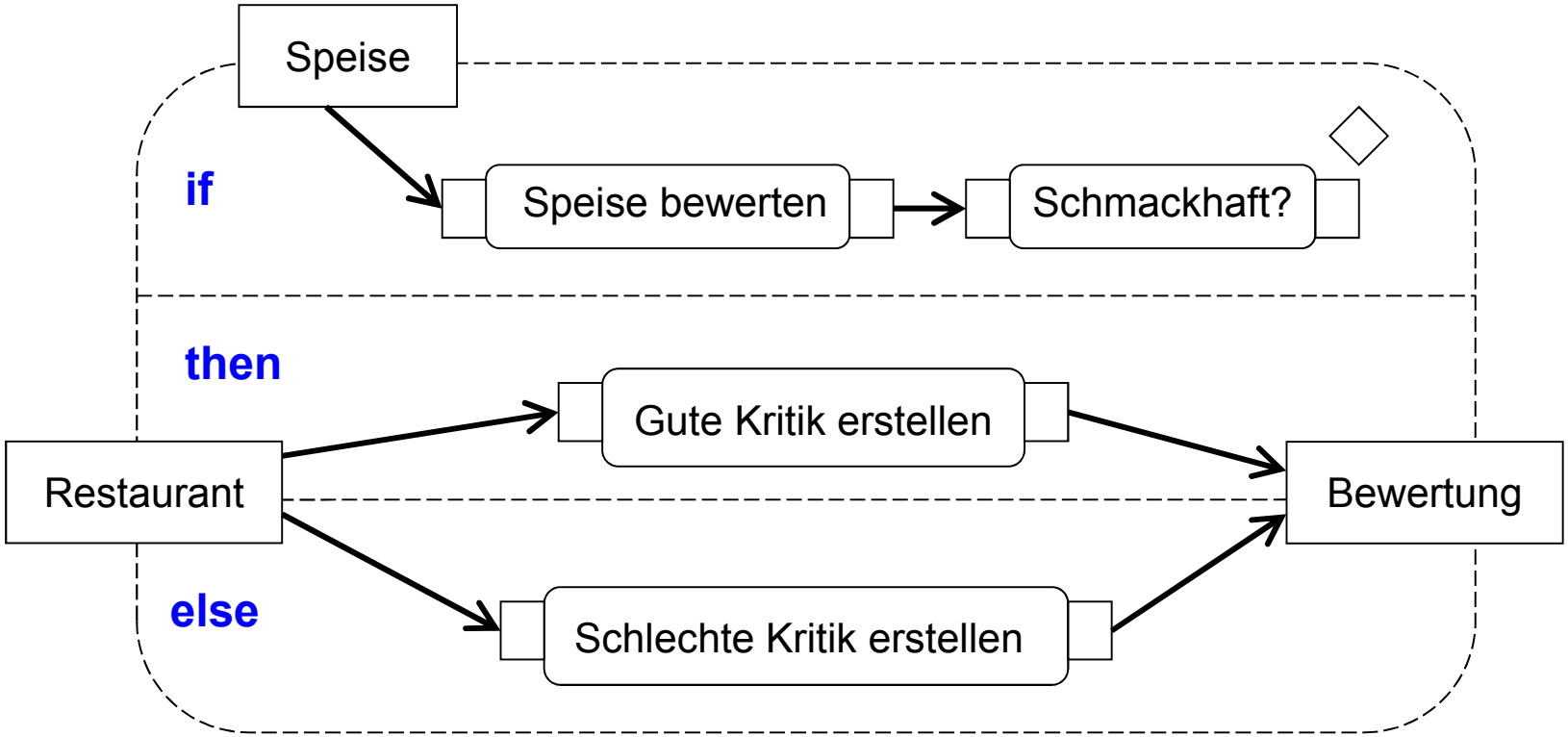
Entscheidungsknoten

- Spezieller strukturierter Knoten mit Aufteilung in mehrere Bereiche:
  1. **if**-Bereich (auch mehrfach), überprüft eine Bedingung (Ergebnis: boolesche Variable; Pin an Aktion, die Bed. prüft sowie Raute)
  2. **then**-Bereich (optional), enthält Elemente, die nach Erfüllung der Bedingung ausgeführt werden
  3. **else-if**-Bereich (optional) und der
  4. **else**-Bereich (optional)
- Notation:



Entscheidungsknoten (2)

- Beispiel (Bewertung einer Speise):



Literatur

- **UML 2 glasklar**
  Mario Jeckle, Chris Rupp, Jürgen Hahn, Barbara Zengler, Stefan Queins
  Hanser Verlag München Wien, 2004

- **UML 2.0 in a Nutshell**
  Dan Pilone, Neil Pitman
  O'Reilly Verlag, 2006