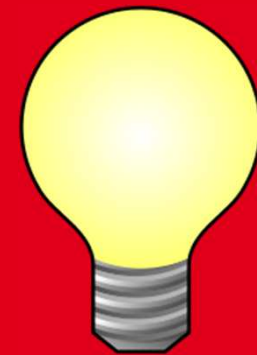


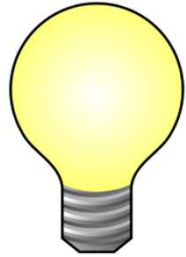


Hochschule **RheinMain**  
University of Applied Sciences  
Wiesbaden Rüsselsheim

20.11.2018  
Aktivitätsdiagramme

Aktivitätsdiagramme in UML





Hochschule **RheinMain**  
University of Applied Sciences  
Wiesbaden Rüsselsheim

# AGENDA

Einführung ins Thema

Loslegen

Beispiel

Modellelemente im Überblick

Analyse von Nebenläufigkeit

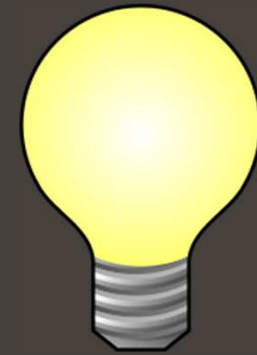
Fazit



Hochschule **RheinMain**  
University of Applied Sciences  
Wiesbaden Rüsselsheim

# 01 EINFÜHRUNG INS THEMA

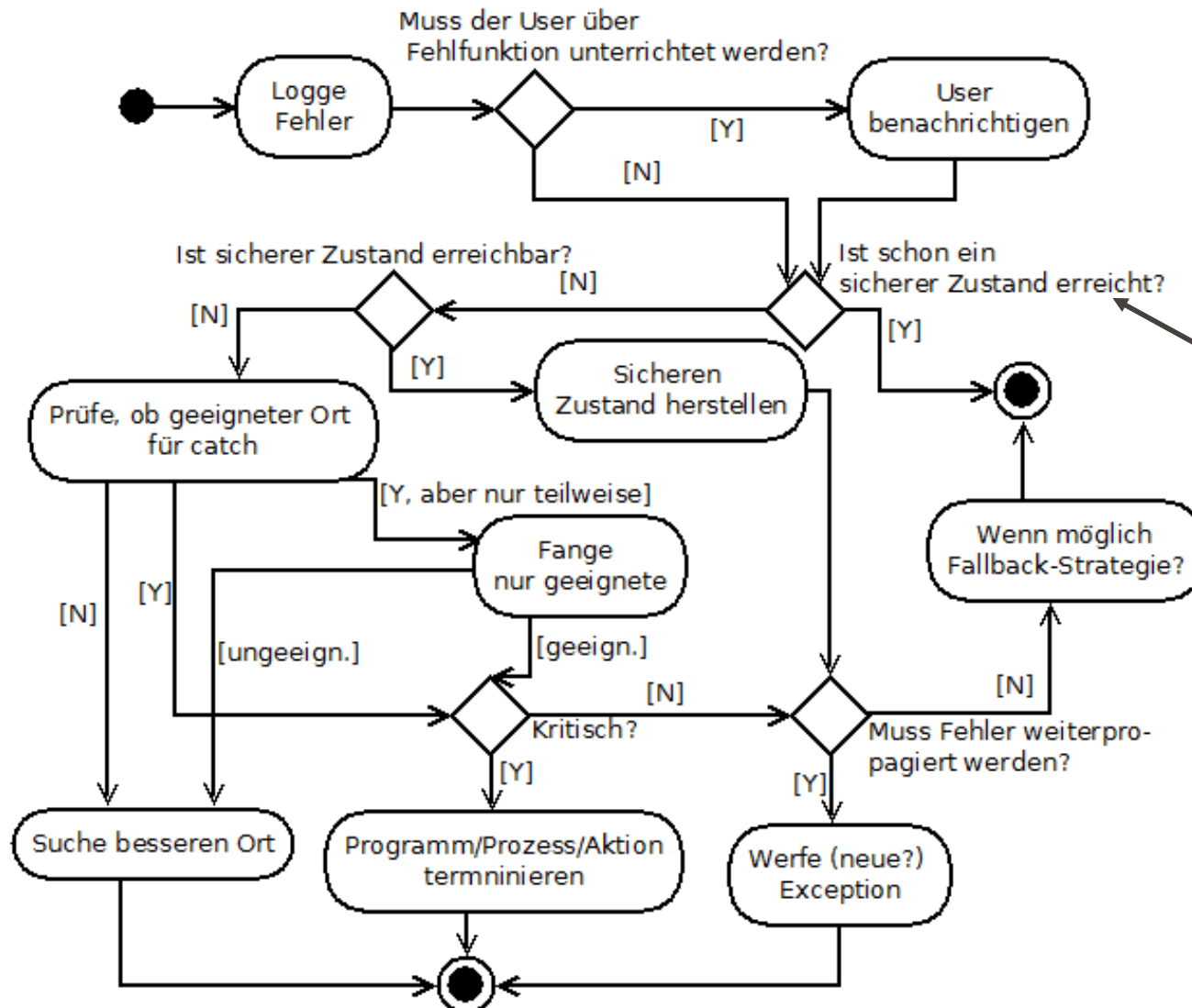
Ziel:  
Die Eckpunkte des Themas kennenlernen



# AUS DER PM-VORLESUNG: VORGEHEN BEHANDLUNGS- STRATEGIE IN REISSLEINE



Hochschule **RheinMain**  
University of Applied Sciences  
Wiesbaden Rüsselsheim



Bedingungsspezif.  
hier ist **KEIN**  
korrektes UML!  
→ Müsste so heißen:

```

<<decisionInput>>
Ist schon ein sicherer
Zustand erreicht?
  
```



# AKTIVITÄTSDIAGRAMME



Hochschule **RheinMain**  
University of Applied Sciences  
Wiesbaden Rüsselsheim

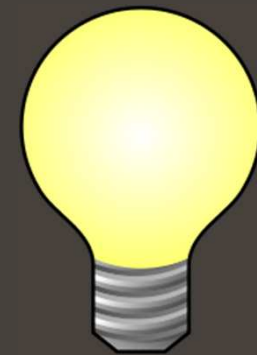
- Eignen sich zur Modellierung von:
  - Aktionen und deren Ablauf / Zusammenhang
    - Prozesse (Geschäftsprozesse und andere)
    - Algorithmen
- Darstellung komplizierter Abläufe mit Schleifen & Verzweigungen
- Gleichmäßiges Fließen (im Gegensatz zu stockender Abarbeitung bei Zustandsdiagrammen)



Hochschule **RheinMain**  
University of Applied Sciences  
Wiesbaden Rüsselsheim

## 02 Loslegen

Ziel:  
Erste Sachen kennenlernen





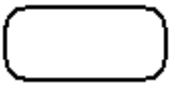
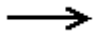
VORSICHT:



Hochschule **RheinMain**  
University of Applied Sciences  
Wiesbaden Rüsselsheim

Aktivitätsdiagramme haben:

- (Fast) gleiche Syntax (gleiches Aussehen) wie Zustandsdiagramme, aber komplementäre Semantik (Bedeutung)

Modellelement	Zustandsautomat	Aktivitätsdiagramm
	Zustand (oft: nichts passiert direkt)	Aktion
	Übergang (Aktion)	Verbindet Aktionen (im „→“ passiert nichts)

# BEVORZUGTE EINSATZGEBIETE



Hochschule **RheinMain**  
University of Applied Sciences  
Wiesbaden Rüsselsheim

- Ablauf / Prozesse / Algorithmen
  - Siehe Flussdiagramm
- Zusammenarbeit bei Nebenläufigkeit
  - Siehe Petri-Netze

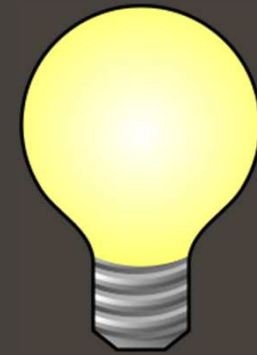




Hochschule **RheinMain**  
University of Applied Sciences  
Wiesbaden Rüsselsheim

# 03 Beispiel

Ziel:  
Ein Beispiel



# BEISPIEL



Hochschule **RheinMain**  
University of Applied Sciences  
Wiesbaden Rüsselsheim

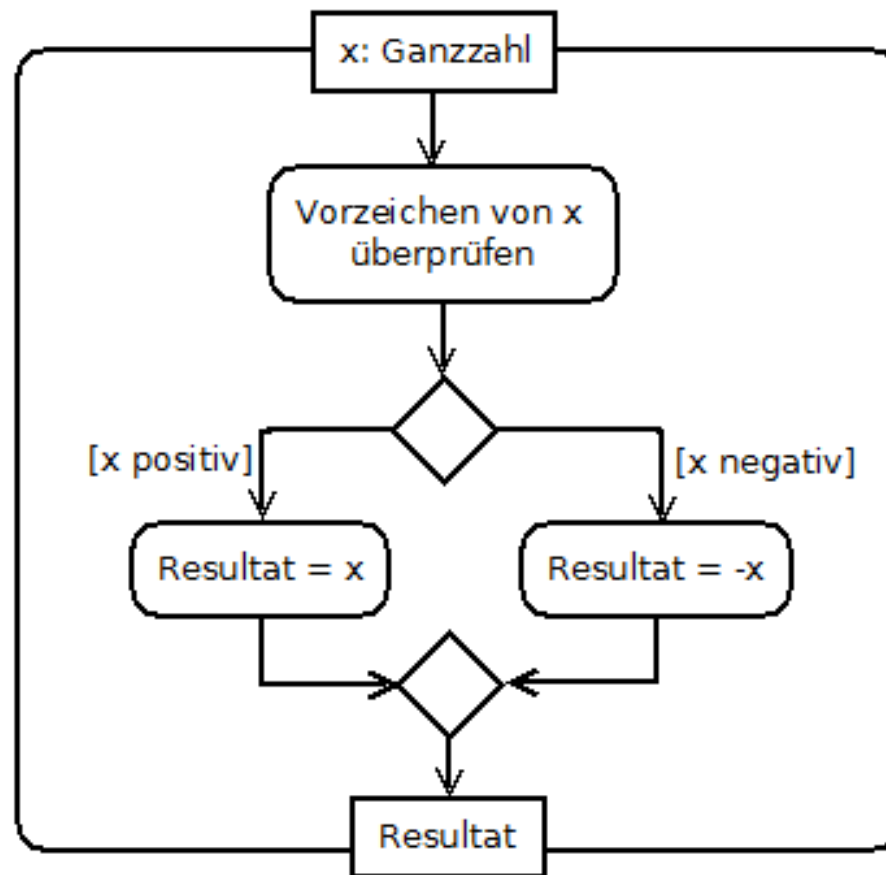
- Code einer Betragsfunktion

```
public static int betrag(int x) {  
    if (x >= 0) {  
        return x;  
    }  
    else {  
        return -x;  
    }  
}
```

# BEISPIEL

- Code einer Betragsfunktion

```
public static int betrag(int x) {  
    if (x >= 0) {  
        return x;  
    }  
    else {  
        return -x;  
    }  
}
```





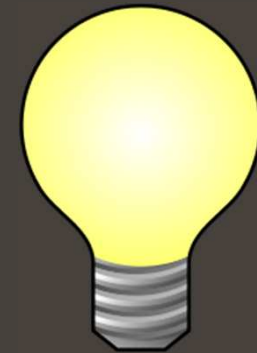
Hochschule **RheinMain**  
University of Applied Sciences  
Wiesbaden Rüsselsheim

04

# Aktivitätsdiagramme - Modellelemente im Überblick

Ziel:

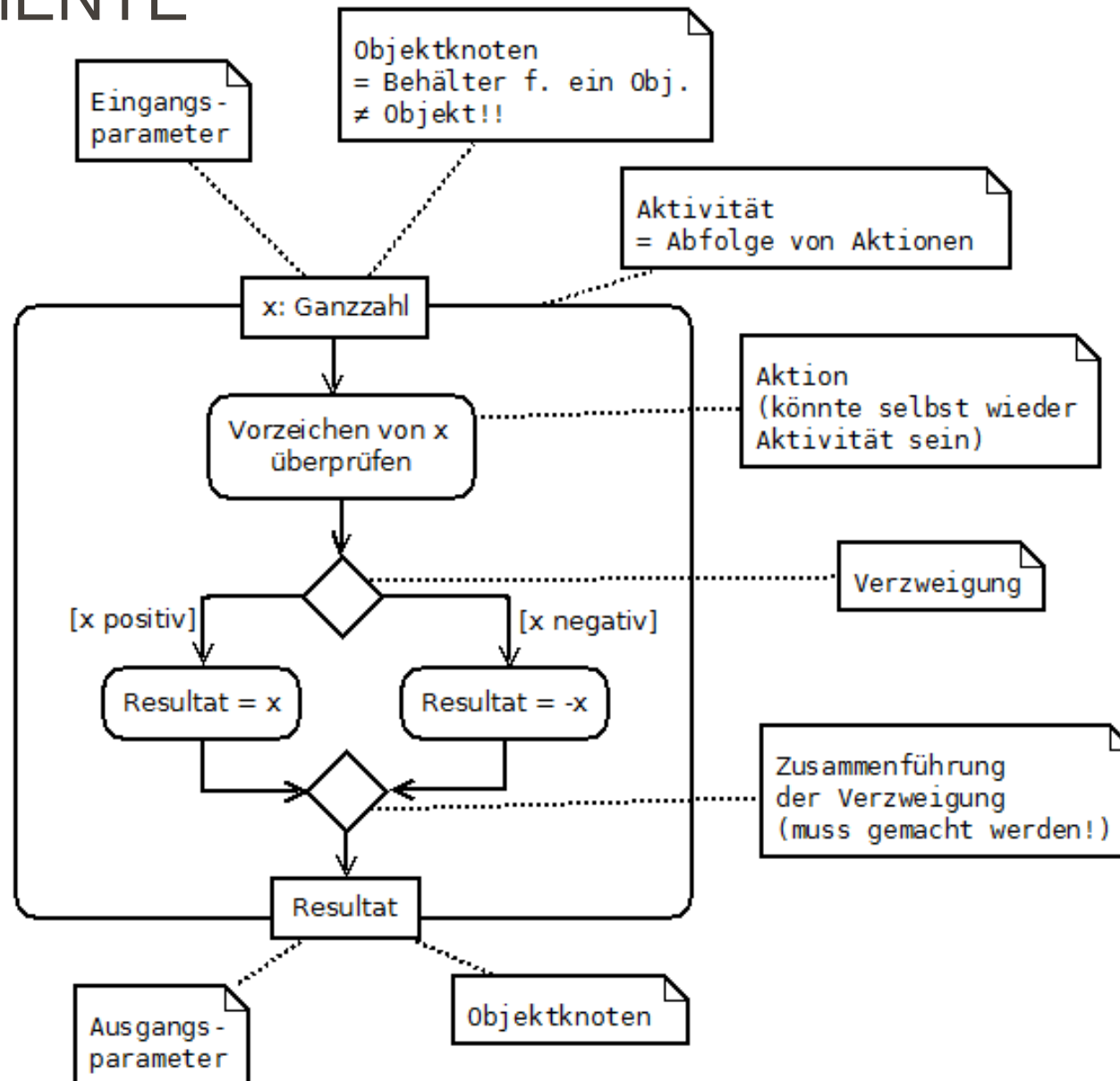
Die Elemente im Überblick erfassen



# DIE WESENTLICHEN ELEMENTE



Hochschule RheinMain  
University of Applied Sciences  
Baden Rüsselsheim



# START- & ENDE- SYMBOLE



Hochschule **RheinMain**  
University of Applied Sciences  
Wiesbaden Rüsselsheim

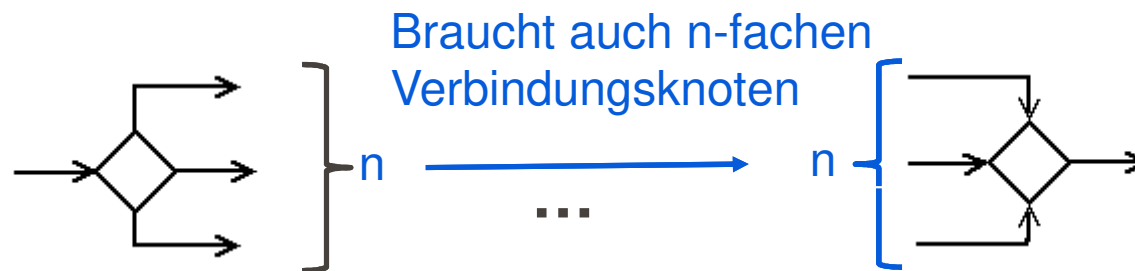
- Start:
  - Start des Aktivitätsdiagramms (Initial Node)
- Ende:
  - ⊙ Komplettes Ende des Aktivitätsdiagr.  
(Activity Final Node)
  - ⊗ Nur Ende des Pfads (Flow Final Node)

# VERZWEIGUNGEN



Hochschule **RheinMain**  
University of Applied Sciences  
Wiesbaden Rüsselsheim

Einfache Verzweigungen (alternative Wege, keine Parallelität):



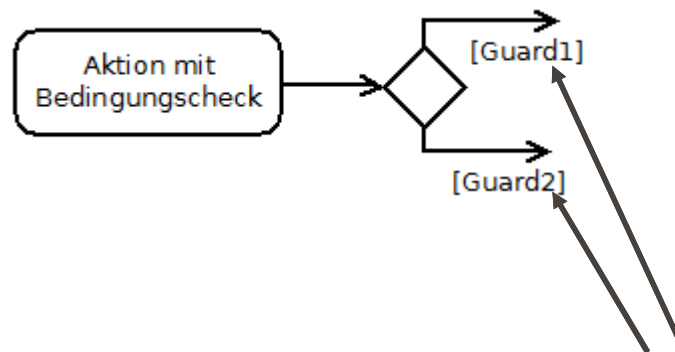
- Für jede öffnende Raute muss auch eine schließende Raute gemacht werden
  - Ansonsten Probleme mit checks bei Nebenläufigkeit
  - Token-Konzept (siehe folgendes Kap.) funktioniert dann nicht

# VERZWEIGUNGEN – WIE BEDING. FORMULIEREN?

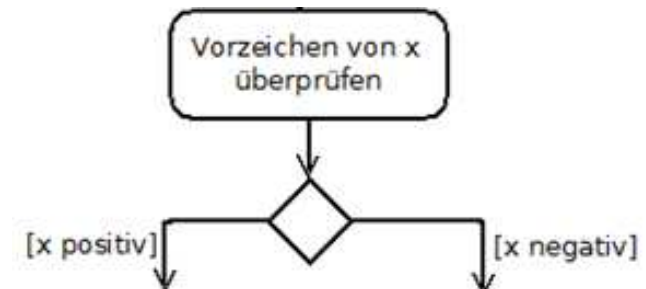


Hochschule **RheinMain**  
University of Applied Sciences  
Wiesbaden Rüsselsheim

## 1. Möglichkeit: Die Aktion vorher liefert die Bedingung:

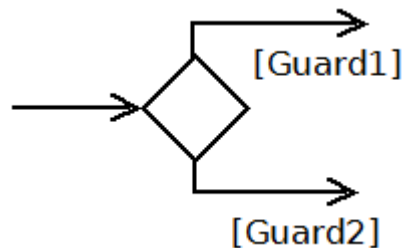


BSP (siehe vorher):

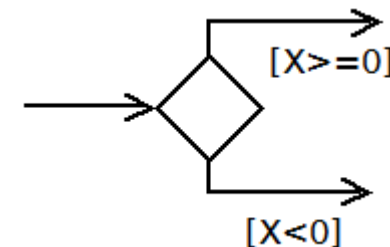


Guards nicht vergessen!!

## 2. Möglichkeit: Spezifikation am Entscheidungsknoten über Guards (Wächter)



BSP:



→ Eignet sich eher für einfache Bedingungen

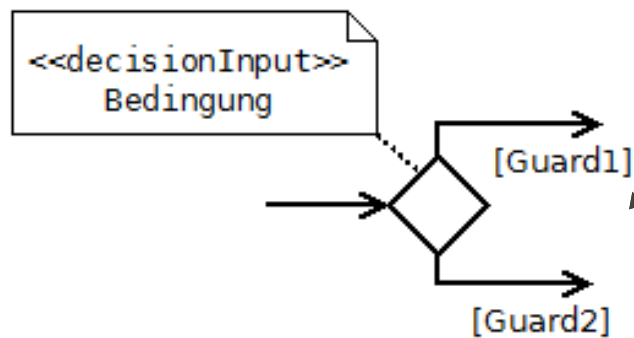


# VERZWEIGUNGEN – WIE BEDING. FORMULIEREN?

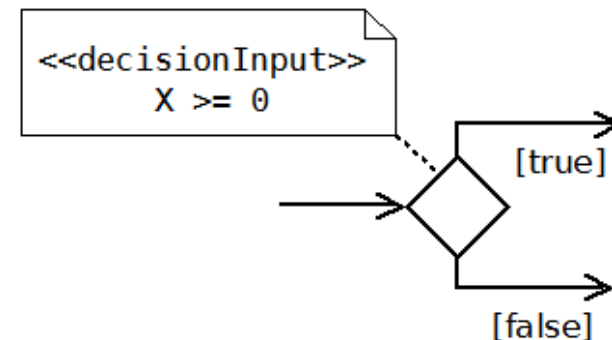


Hochschule **RheinMain**  
University of Applied Sciences  
Wiesbaden Rüsselsheim

## 3. Möglichkeit: Spezifikation am Entscheidungsknoten über Kommentar mit <<decisionInput>>-Stereotypen



BSP:



Guards nicht vergessen!!

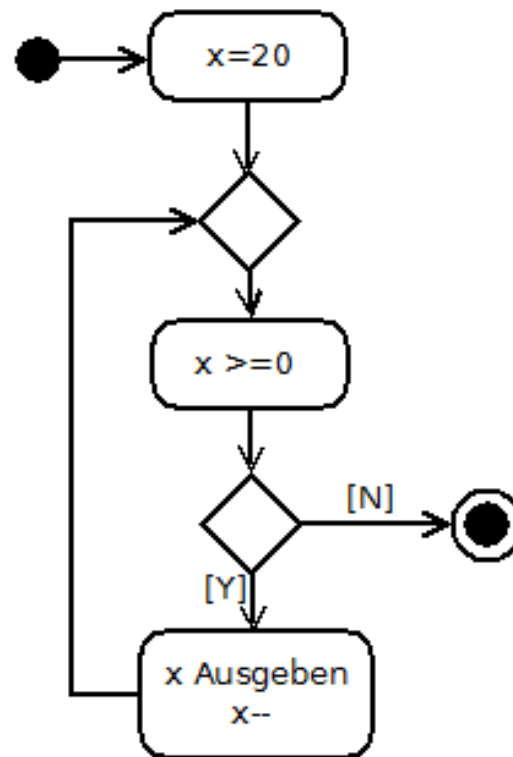
→ Eignet sich eher für komplexere Bedingungen

# WIE SCHLEIFEN MODELLIEREN?



Hochschule **RheinMain**  
University of Applied Sciences  
Wiesbaden Rüsselsheim

- Schleifen können über Verzweigungskomponente modelliert werden:



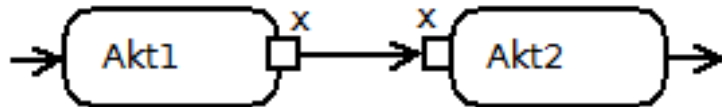
# DIE PIN-NOTATION



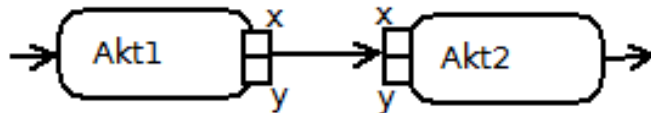
Hochschule **RheinMain**  
University of Applied Sciences  
Wiesbaden Rüsselsheim

Pins zeigen Objektflüsse an:

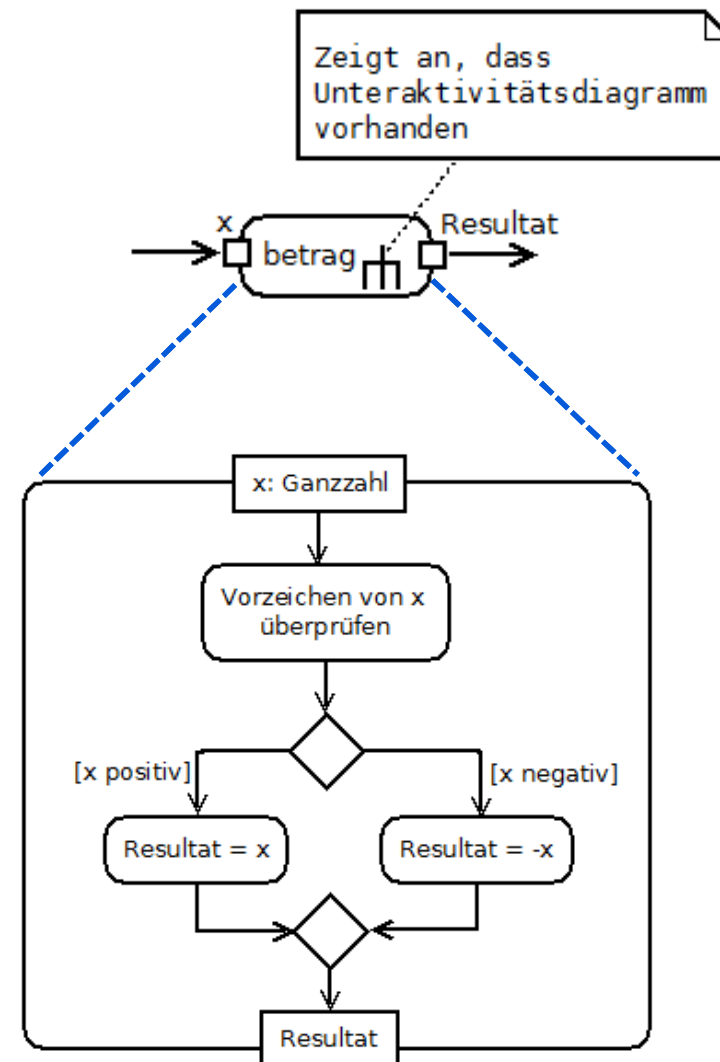
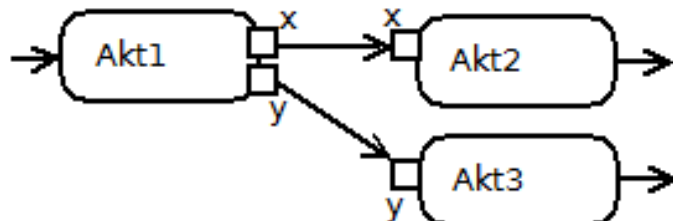
- Einzelne Objekte:



- Objektmengen:



- Objektfl. (UND-Semantik):

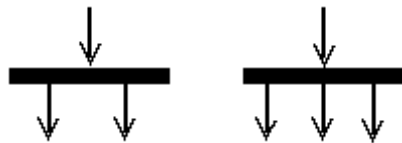


# NOTATION VON NEBENLÄUFIGKEIT

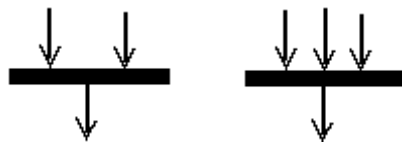


Hochschule **RheinMain**  
University of Applied Sciences  
Wiesbaden Rüsselsheim

- Prozesse teilen (fork):



- Prozesse zusammenführen (join):



- Prozesse müssen immer auch wieder zusammengeführt werden, oder  $\rightarrow \otimes$  muss verwendet werden

# WAS IST NEBENLÄUFIGKEIT?



Hochschule **RheinMain**  
University of Applied Sciences  
Wiesbaden Rüsselsheim

- Auch Parallelität oder Concurrency genannt  
→ Es können Sachen parallel abgearbeitet werden
- Einfachstes Beispiel: Threads (siehe letztes Semester PM)

```
private void myAlgorithm() {  
    doTask1();  
  
    Thread th=new Thread(()-> {  
        doTask2();  
    });  
    th.start();  
  
    doTask3();  
    th.join();  
}
```

- ABER: Nicht nur, sondern auch bei Geschäftsprozessen, Client-Server-Abläufe, Verteilte Systeme (SOA, RPCs), ...

# WAS IST NEBENLÄUFIGKEIT?



Hochschule **RheinMain**  
University of Applied Sciences  
Wiesbaden Rüsselsheim

- Aufgabe: Das Bsp. modellieren

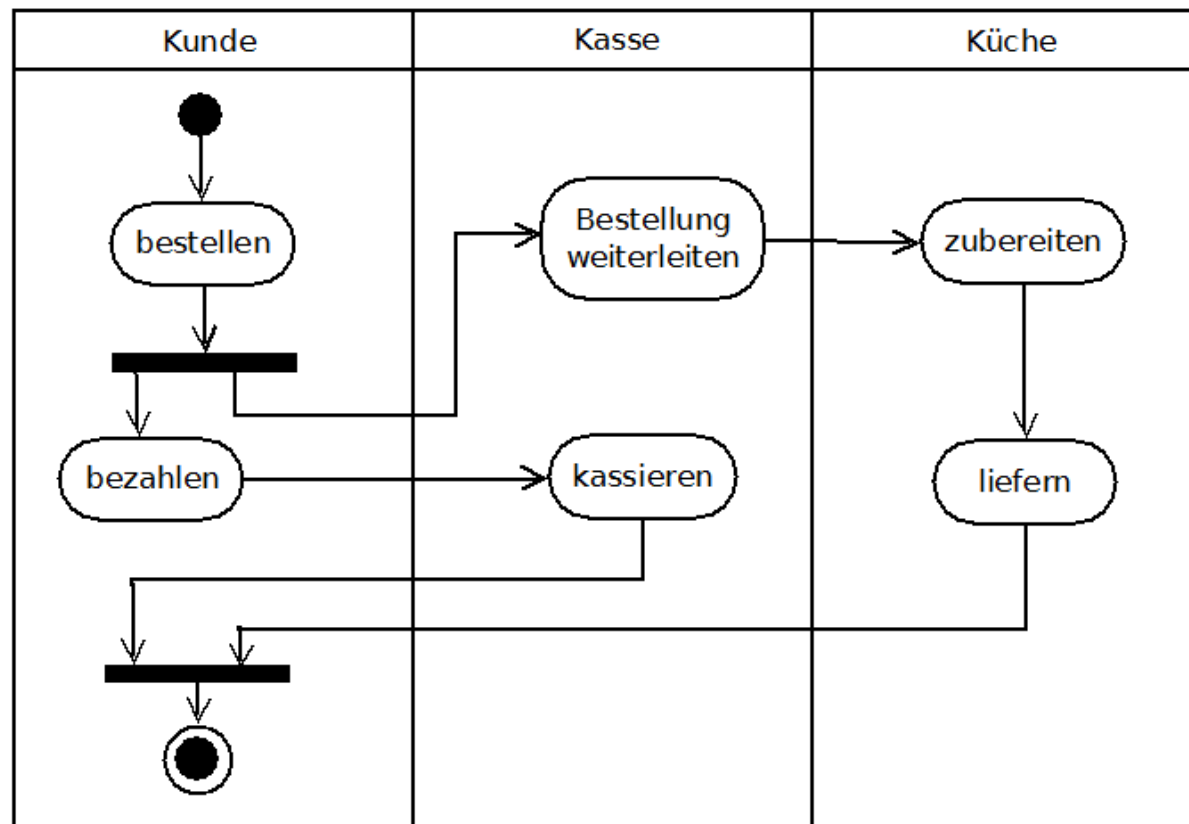
```
private void myAlgorithm() {  
    doTask1();  
  
    Thread th=new Thread(() -> {  
        doTask2();  
    })  
    ;  
    th.start();  
  
    doTask3();  
    th.join();  
}
```

# AKTIVITÄTSBEREICHE – („ACTIVITY PARTITIONS“)



Hochschule **RheinMain**  
University of Applied Sciences  
Wiesbaden Rüsselsheim

- Inoffiziell: Schwimmbahnen (Swimlanes)  
→ Erlauben Mapping von Aktionen auf Systeme / Stakeholder:



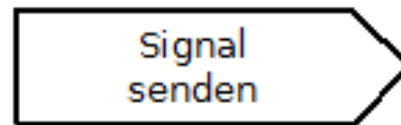
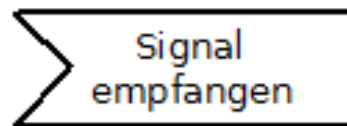
→ Beispiel eines Geschäftsprozesses

# WEITERE DARSTELLUNGS- MITTEL



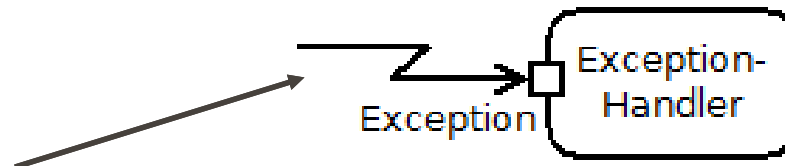
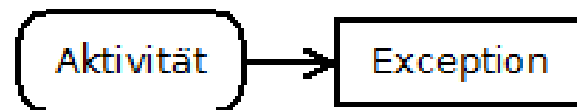
Hochschule **RheinMain**  
University of Applied Sciences  
Wiesbaden Rüsselsheim

Signale:



Zeitereignis  
empfangen

Werfen und Fangen von Exceptions:



Unterbrechungskante





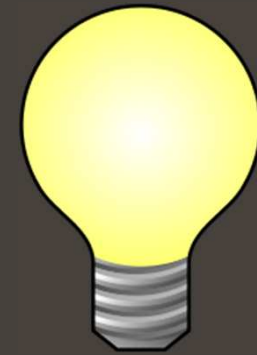
Hochschule **RheinMain**  
University of Applied Sciences  
Wiesbaden Rüsselsheim

05

## Analyse von Nebenläufigkeit

Ziel:

Das formale Modell des Aktivitätsdiagramms  
erlaubt die korrekte Analyse der Nebenläufigkeit



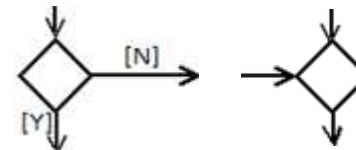
# DEF: KONTROLLKNOTEN



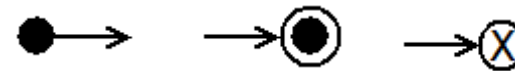
Hochschule **RheinMain**  
University of Applied Sciences  
Wiesbaden Rüsselsheim

- Als Kontrollknoten werden alle Knoten genannt, die keine Aktionen oder Aktivitäten sind:

- Verzweigung und Merge:



- Start- und Endknoten:



- Fork and Join:

(Nebenläufigkeit, z.B. Threads)



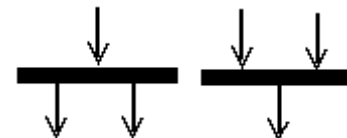


# DAS TOKEN-KONZEPT

- Das Token-Konzept bildet die Grundlage der Semantik von Aktivitätsdiagrammen
  - Das Token-Konzept eignet sich sog. Verklemmungen (Deadlocks) aufzudecken
  - Vgl. Petri-Netze

## IDEE:

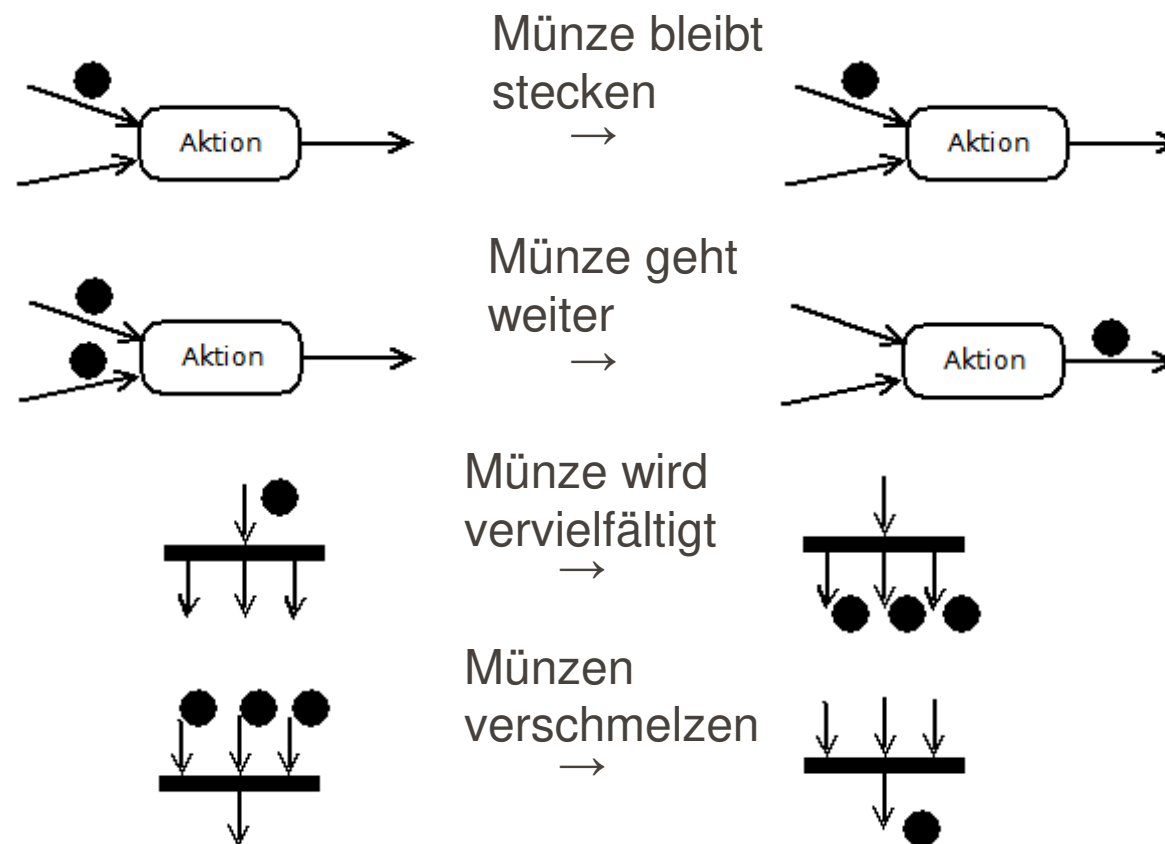
- Ein oder mehrere Token (Münzchips) zeigt / zeigen an in welchem Zustand sich die Verarbeitung gerade befindet.
- Token laufen durch das Diagramm → Kontroll-/Datenfluss
- Token wird erst weitergeleitet, wenn alle Kanten und Zielknoten bereit sind.
- Token darf an Kontrollknoten nicht warten
  - Ausnahme bei Fork & Joinknoten:  
→ Darf gewartet werden





# DAS TOKEN-KONZEPT

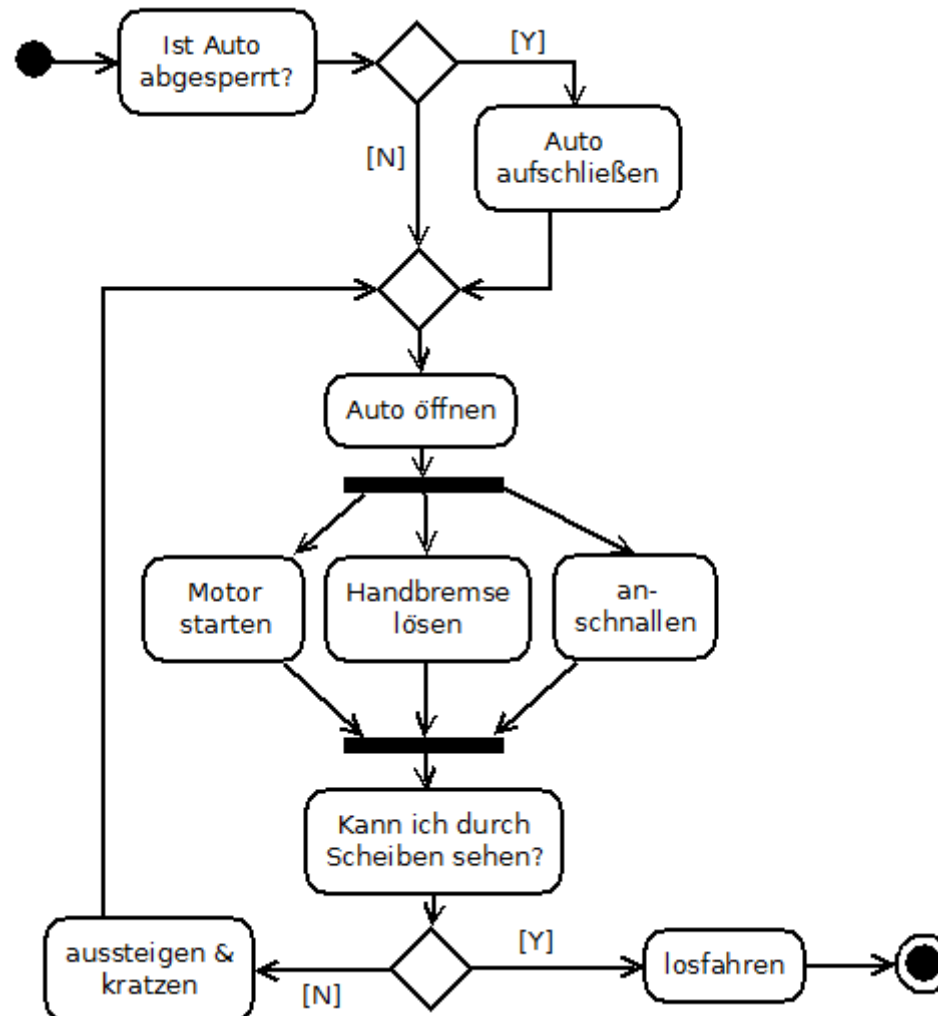
- Münzen zirkulieren (reines Gedankenkonstrukt):
  - Eine Aktion schaltet die Münze(n) nur weiter, wenn an allen Eingängen Münzen anliegen:





# DAS TOKEN-KONZEPT

- Beispiel:

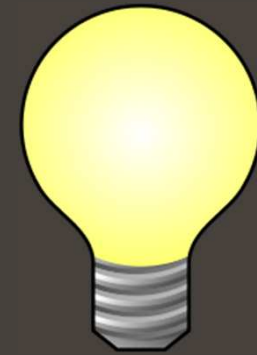




Hochschule **RheinMain**  
University of Applied Sciences  
Wiesbaden Rüsselsheim

## 06 Fazit

Ziel:  
Was haben wir damit gewonnen?





# WAS HABEN WIR GELERNT?

- Das Aktivitätsdiagramm
  - Hilft Prozesse oder Algorithmen zu modellieren
  - Aktivität, Aktion, Pins, Verzweigungen und Fork-Join-Balken
- Token-Konzept hilft bei der Analyse von Nebenläufigkeiten
  - Deadlocks aufdecken



Hochschule **RheinMain**  
University of Applied Sciences  
Wiesbaden Rüsselsheim

AUF GEHT'S!!

SELBER MACHEN UND LERNEN!!

