

InGO

CHRISTIAN HUEMER

MARION SCHOLZ

Objektorientierte Modellierung mit UML

TEIL II - Sequenzdiagramm

ingo

Sequenzdiagramm Die Interaktionsdiagramme



ingo

Christian Huemer und Marion Scholz

Interaktionen und Nachrichten

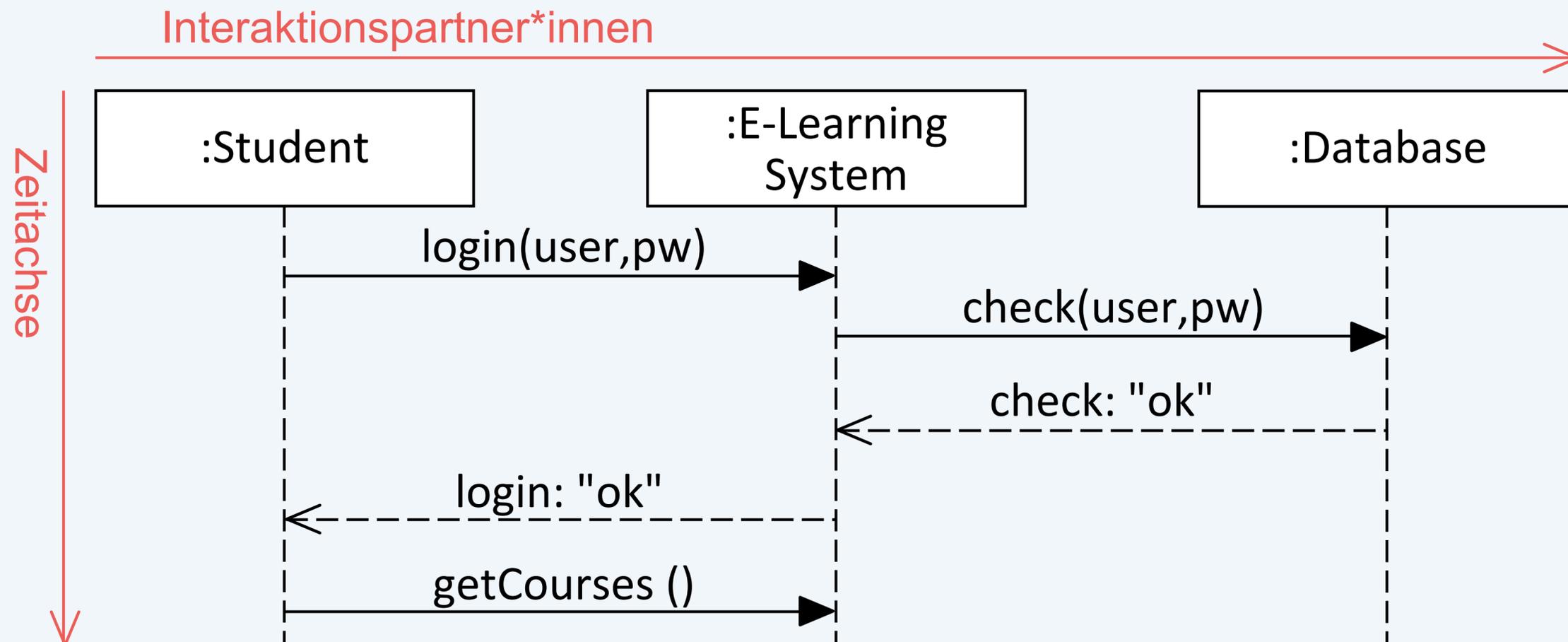
- Interaktion
 - Zusammenspiel mehrerer Kommunikationspartner*innen
 - Nachrichten- und Datenaustausch
- Interaktionen durch
 - Signale
 - Operationsaufrufe
 - Aufruf einer Operation einer Klasse
 - Antwort: Ergebnis der aufgerufenen Operation
- Steuerung der Interaktionen durch
 - Bedingungen
 - Zeitereignisse

Interaktionsdiagramme

- Zeigen, wie Nachrichten zwischen verschiedenen Kommunikationspartner*innen in einem bestimmten Kontext ausgetauscht werden
- Beschreibung von Kommunikationssituationen durch:
 - Kommunikationspartner*innen + deren Lebenslinien
 - Interaktionen
 - Nachrichten
 - Mittel zur Flusskontrolle
- Vier Typen von Interaktionsdiagrammen
 - Alle vier basieren auf ähnlichen Konzepten
 - Unterschiedliche Anforderungen und Betonung unterschiedlicher Aspekte
 - Für einfache Interaktionen semantisch äquivalent, aber unterschiedlicher Fokus

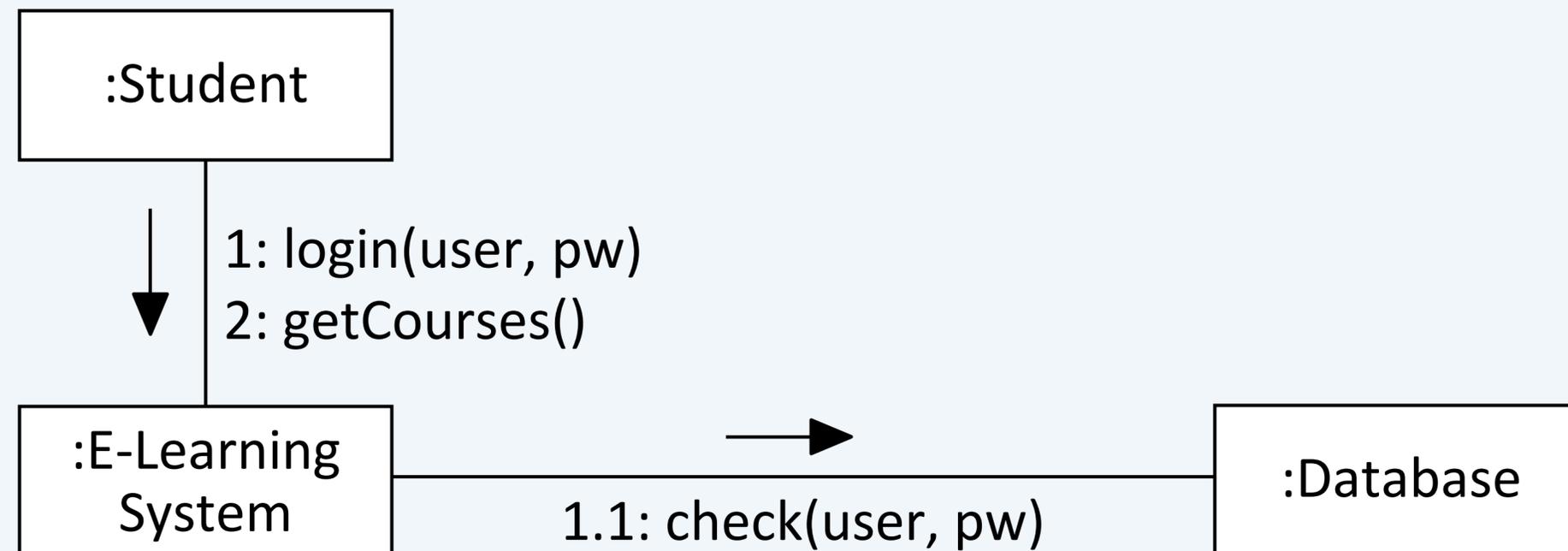
Interaktionsdiagramme – Arten (1/4)

- **Sequenzdiagramm** zeigt den zeitlichen und logischen Nachrichtenfluss
 - Zeit ist eigene Dimension



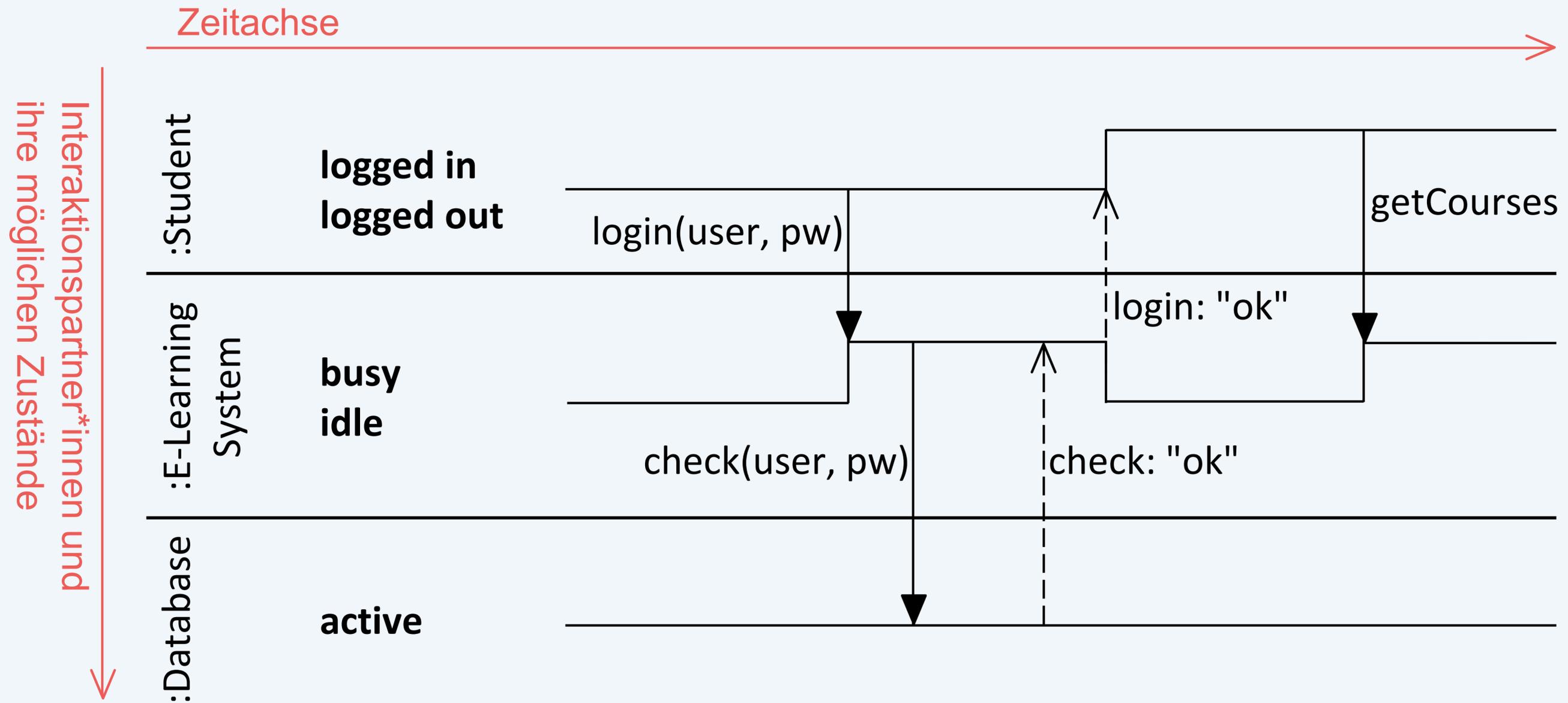
Interaktionsdiagramme – Arten (2/4)

- **Kommunikationsdiagramm** ist »strukturell« orientiert
 - Zeigt die Beziehungen zwischen Interaktionspartner*innen
 - Fokus: Wer kommuniziert mit wem
 - Zeit ist keine eigene Dimension
 - Reihenfolge von Nachrichten nur über Dezimalklassifikation ausgedrückt



Interaktionsdiagramme – Arten (3/4)

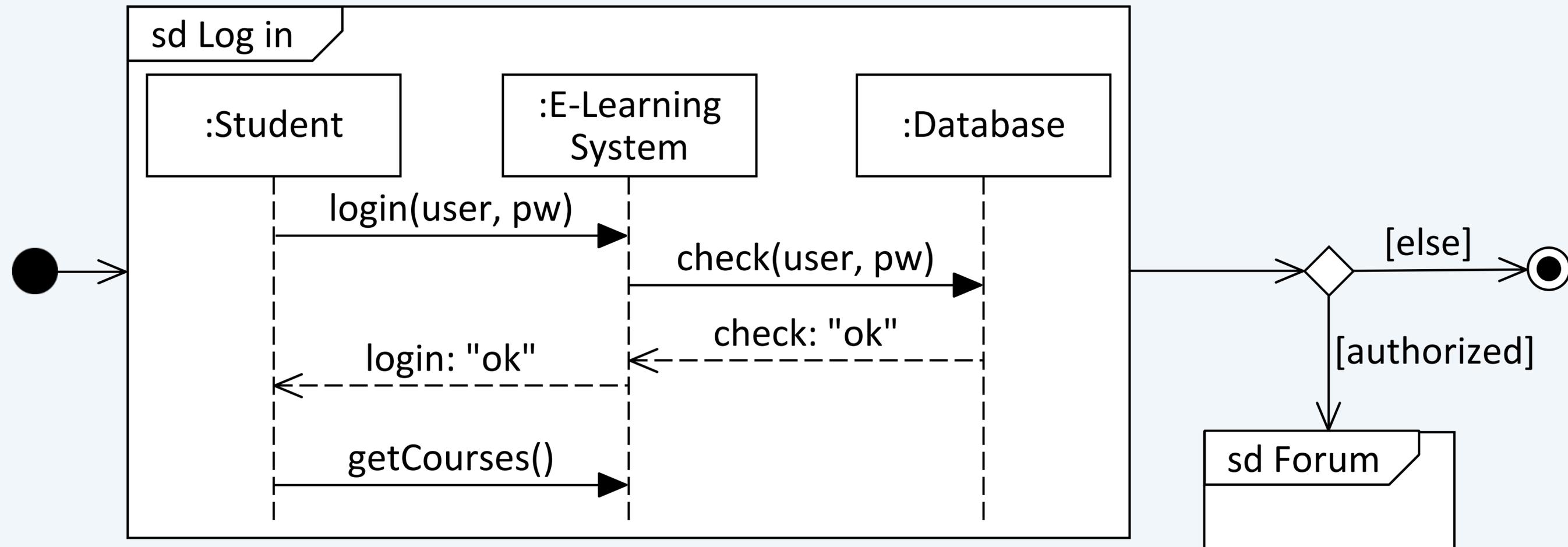
- **Zeitdiagramm** zeigt Zustandsänderungen der Interaktionspartner*innen aufgrund von Interaktionen oder Zeitereignissen



Interaktionsdiagramme – Arten (4/4)

■ Interaktionsübersichtsdiagramm

- zeigt das Zusammenspiel von verschiedenen Interaktionen
- Visualisiert, in welcher Reihenfolge und unter welchen Bedingungen Interaktionsabläufe stattfinden
- Notationselemente vom Aktivitätsdiagramm



Einsatzbereiche

Modellierung...

- der **Interaktionen eines Systems mit seiner Umwelt** (Systemgrenzen festlegen, System als Black-Box)
- der **Realisierung eines Anwendungsfalls**
- des **Zusammenspiels der internen Struktur** einer Klasse, Komponente oder Kollaboration
- der **Operationen der Klassen**
- der Spezifikation von **Schnittstellen zwischen Systemteilen** (Zusammenspiel angebotene/benutzte Schnittstelle)

ingo

Sequenzdiagramm Die Lebenslinie



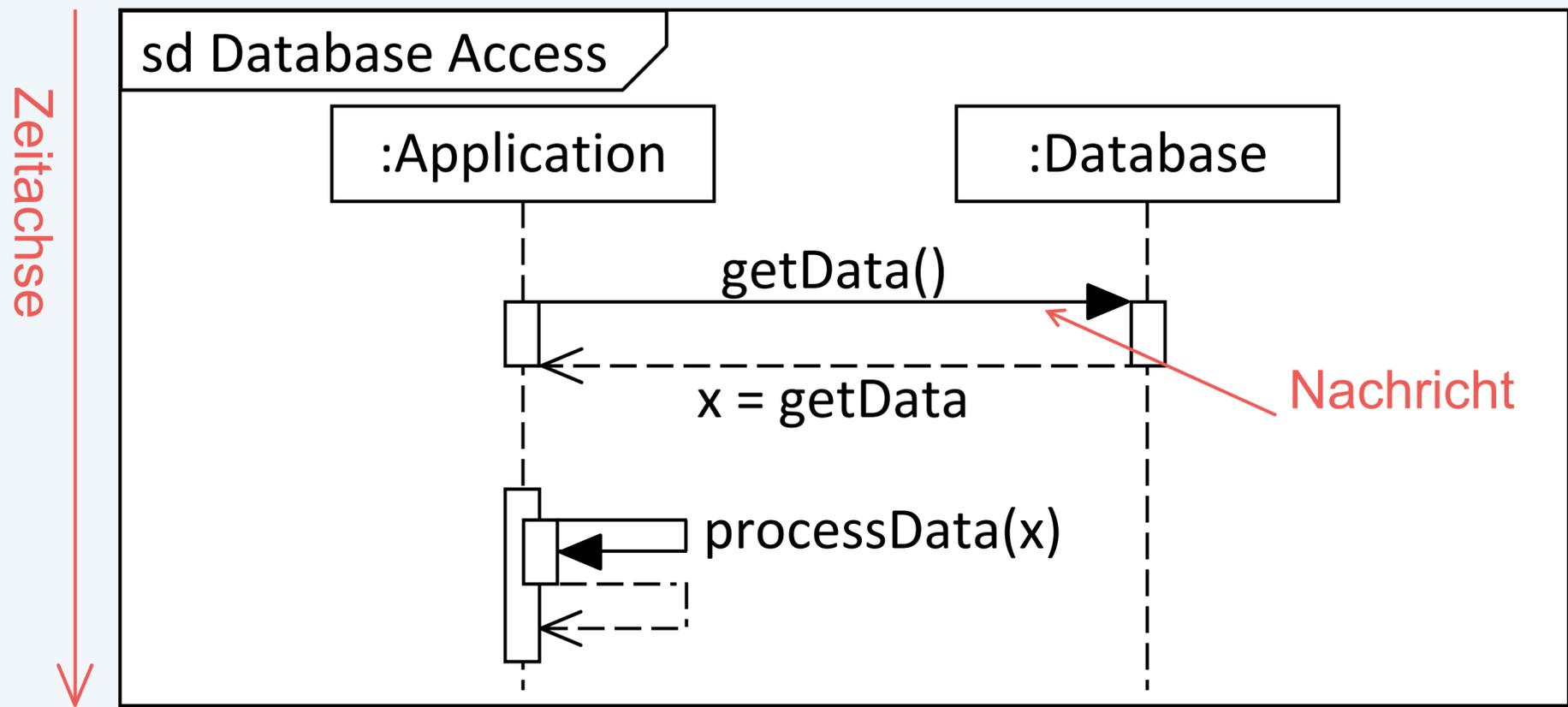
ingo

Christian Huemer und Marion Scholz

Sequenzdiagramm

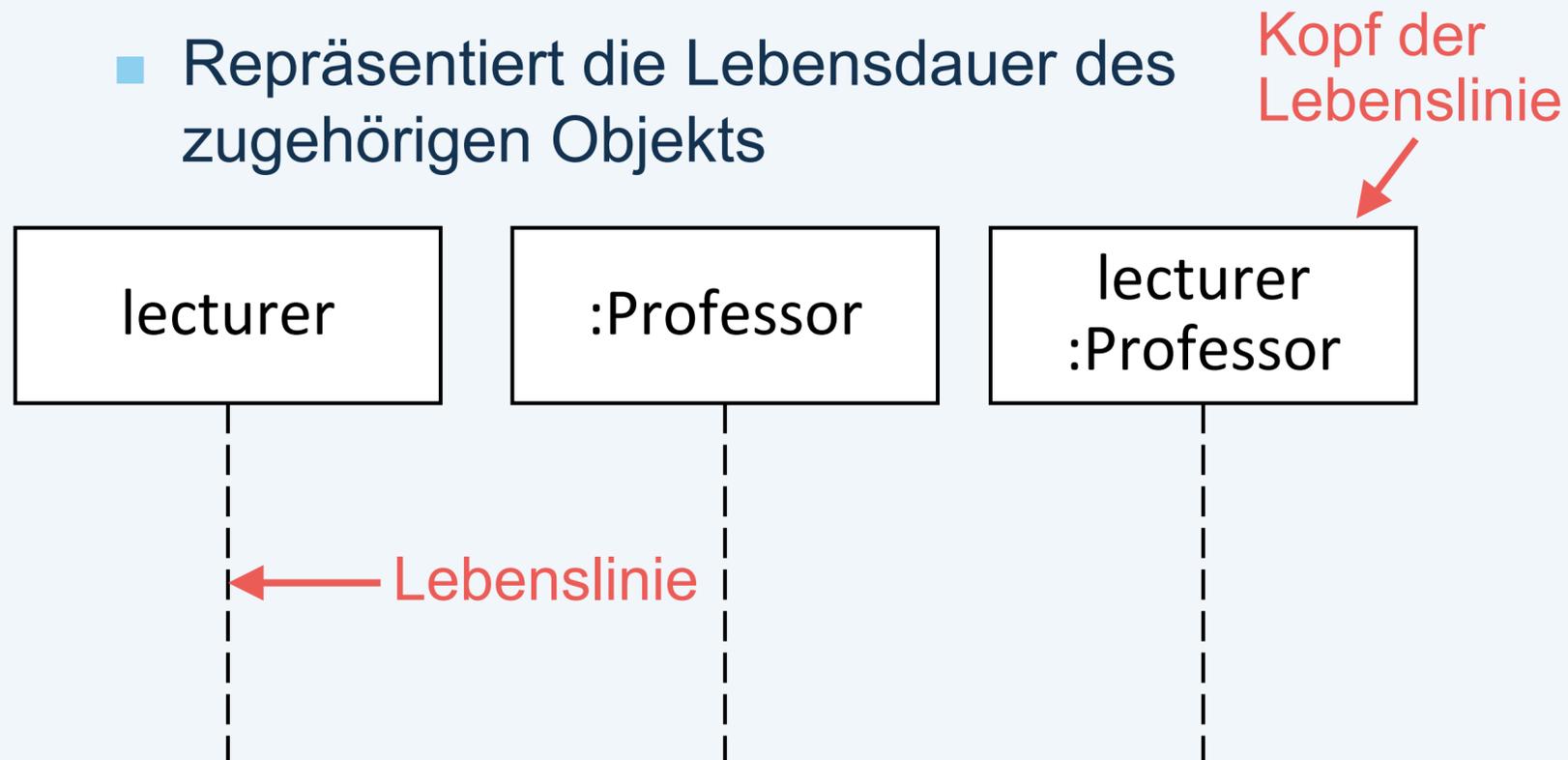
- Darstellung von Interaktionen in **2 Dimensionen**:
 - **horizontal**: Kommunikationspartner*innen in Form von Rollen; Reihenfolge ist unbedeutend
 - **vertikal**: Zeitachse
Darstellung des zeitlichen Ablaufs der Kommunikation

Kommunikationspartner*innen →



Lebenslinie

- Eine **Lebenslinie** beschreibt **genau einen/eine Kommunikationspartner*in**
- Kopf der Lebenslinie
 - Rechteck mit **Rollename : Klasse**
 - Objekt kann während seiner Lebensdauer unterschiedliche Rollen annehmen
- Körper der Lebenslinie
 - Vertikale, strichlierte Linie
 - Repräsentiert die Lebensdauer des zugehörigen Objekts



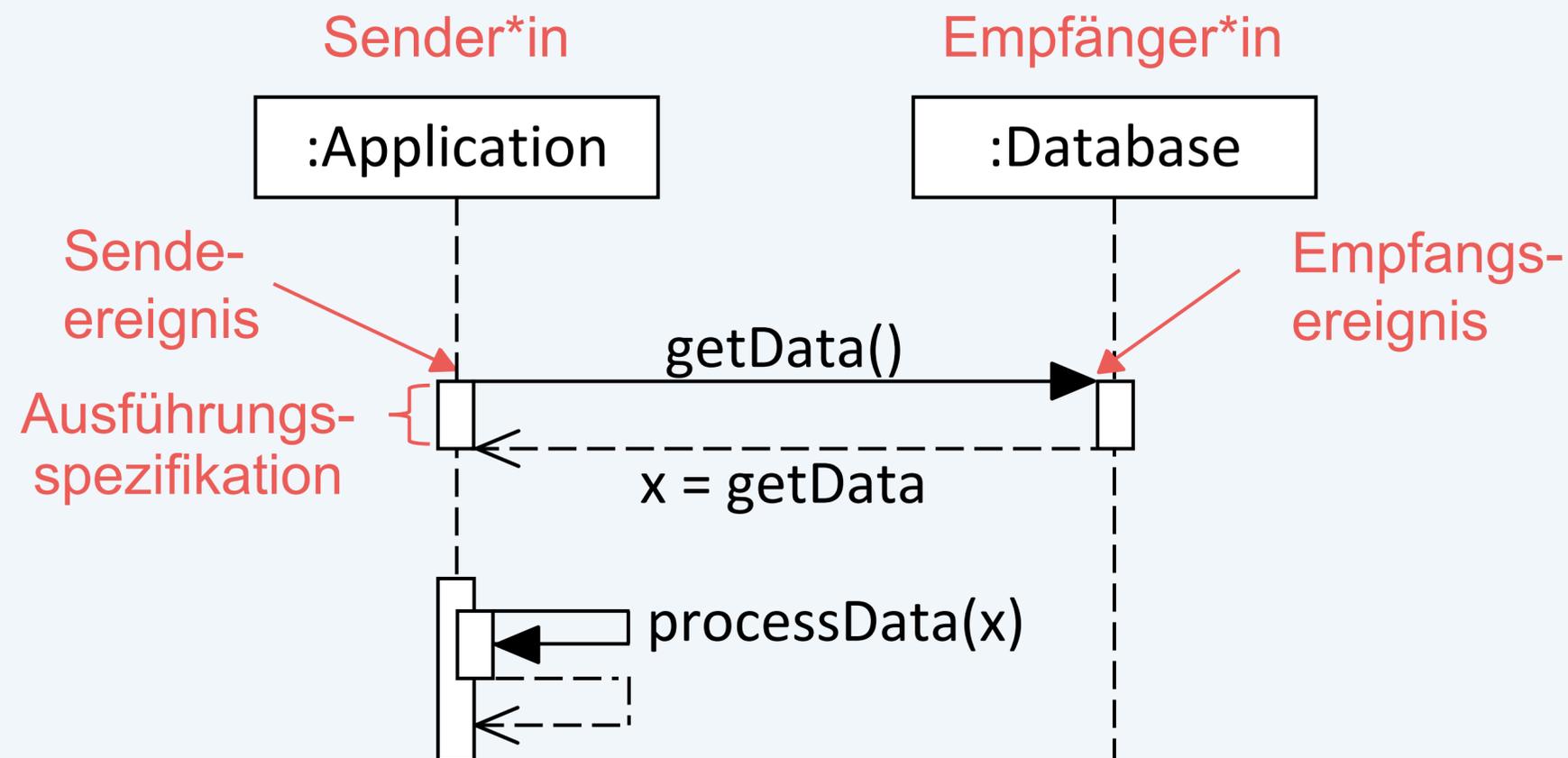
Typ- vs. Instanzebene

- Modellierung des Nachrichtenaustauschs immer auf **Instanzebene**
 - Rollen stellen repräsentative Objekte für den zu modellierenden Kontext dar
 - Tatsächliche Interaktion findet auf Instanzebene zwischen Objekten statt

- Trace
 - Abfolge von Nachrichten zwischen konkreten Objekten

Lebenslinie: Ausführungsspezifikation (1/2)

- Interaktionen sind die **Folge von Ereignisspezifikationen** auf den Lebenslinien
- Beispiel für Ereignisspezifikationen
 - **Senden** und **Empfangen** von Nachrichten auf verschiedenen Lebenslinien oder der gleichen Lebenslinie



Lebenslinie: Ausführungsspezifikation (2/2)

Reihenfolge von Ereignisspezifikationen

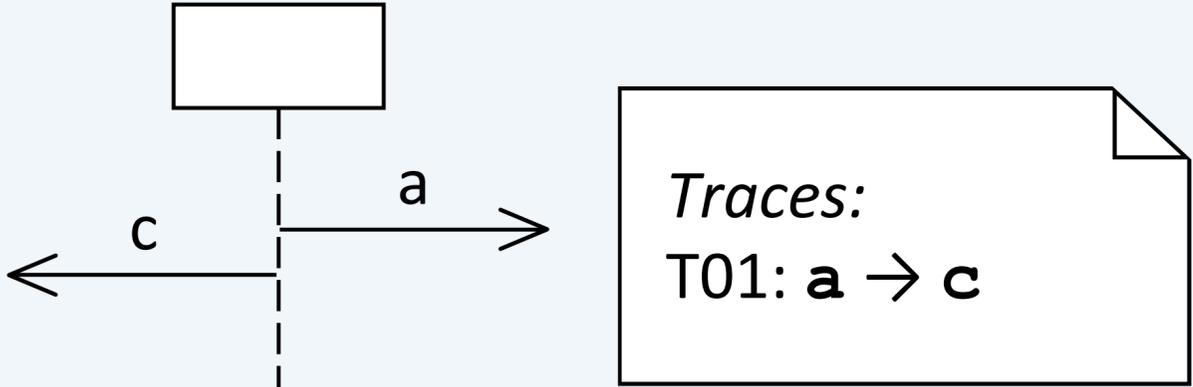
- Vertikale Zeitachse bestimmt die Ordnung der Ereignisseintritte pro Lebenslinie

Bestimmt nicht die Reihenfolge von Ereignisseintritten auf verschiedenen Lebenslinien

- **Nachrichten zwischen Lebenslinien erzwingen** eine Ordnung über Lebenslinien hinweg

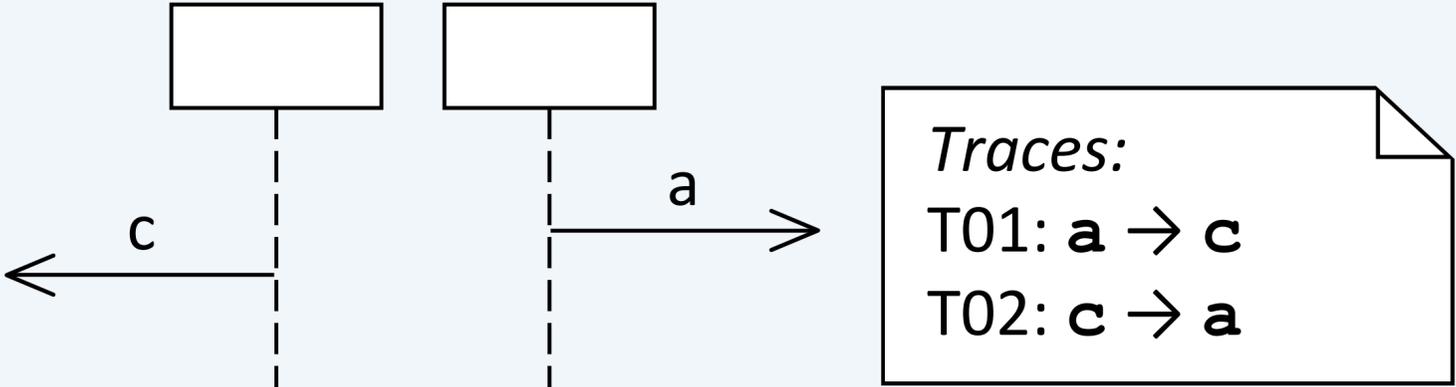
Lebenslinie: Reihenfolge von Ereignisseintritten

...auf einer Lebenslinie



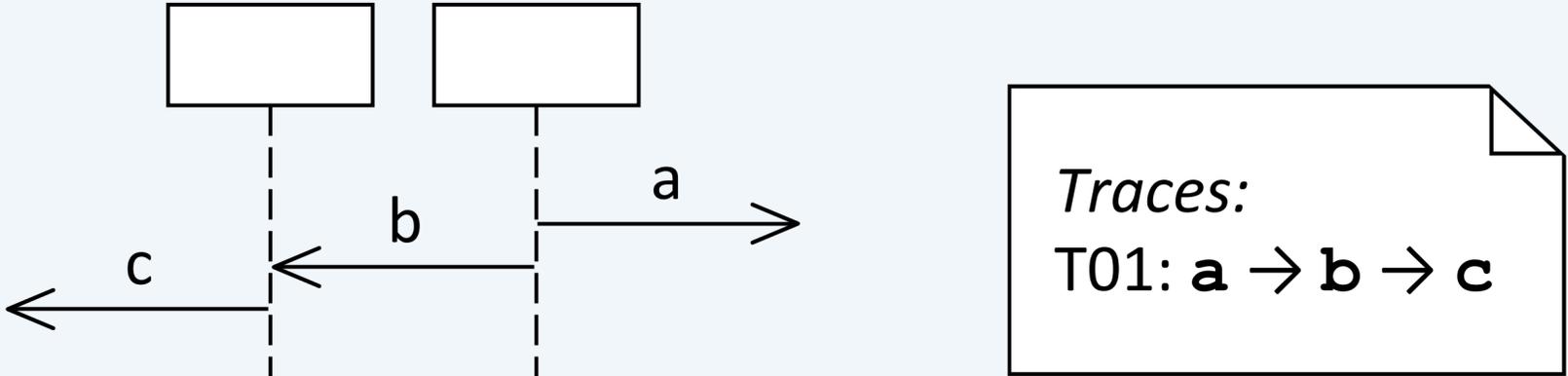
Traces:
T01: **a** → **c**

... auf verschiedenen Lebenslinien



Traces:
T01: **a** → **c**
T02: **c** → **a**

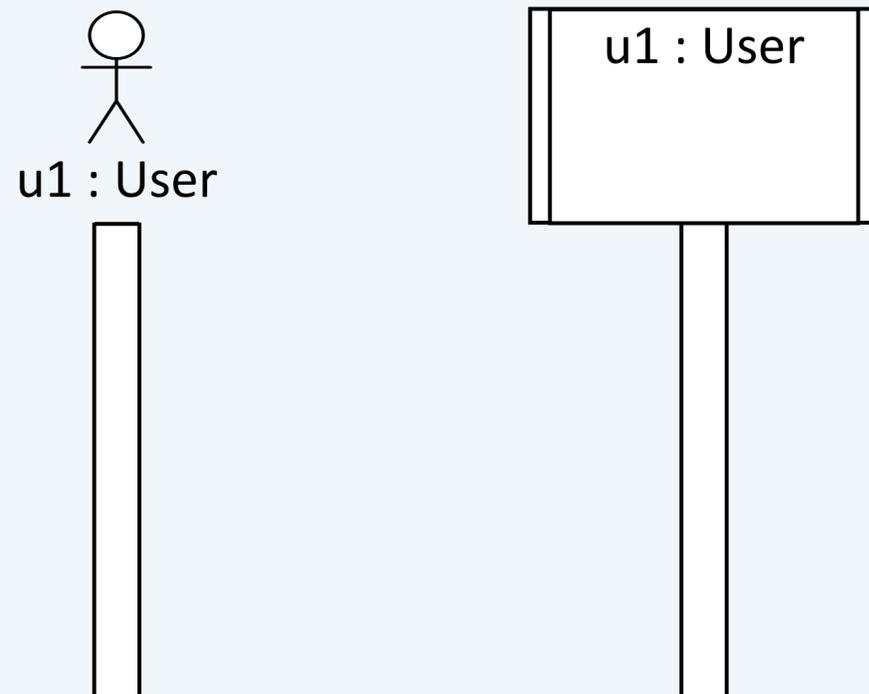
... auf verschiedenen Lebenslinien,
verbunden durch Nachrichtenaustausch



Traces:
T01: **a** → **b** → **c**

Lebenslinie: Aktives Objekt

- **Aktive Objekte** verfügen über **eigenen Kontrollfluss** (Prozess oder Thread)
- Können **unabhängig** von anderen Objekten operieren
- Notation
 - Kopf der Lebenslinie wird links und rechts mit doppeltem Rand versehen
 - durchgehender Balken über gesamte Lebenslinie



ingo

Sequenzdiagramm Die Nachricht



ingo

Christian Huemer und Marion Scholz

Arten von Nachrichten

■ Synchrone Kommunikation

- Sender*in wartet bis zur Beendigung der Interaktion, die durch die Nachricht ausgelöst wurde



■ Asynchrone Kommunikation

- Nachricht ist ein Signal
- Sender*in wartet nicht auf das Ende der Interaktion



■ Antwortnachricht (optional)

- Syntax:
`att=msg (par1 ,par2) :val`

att: Name eines Attributs, dem der Rückgabewert zugewiesen werden soll

msg: Name der Nachricht, auf die geantwortet wird

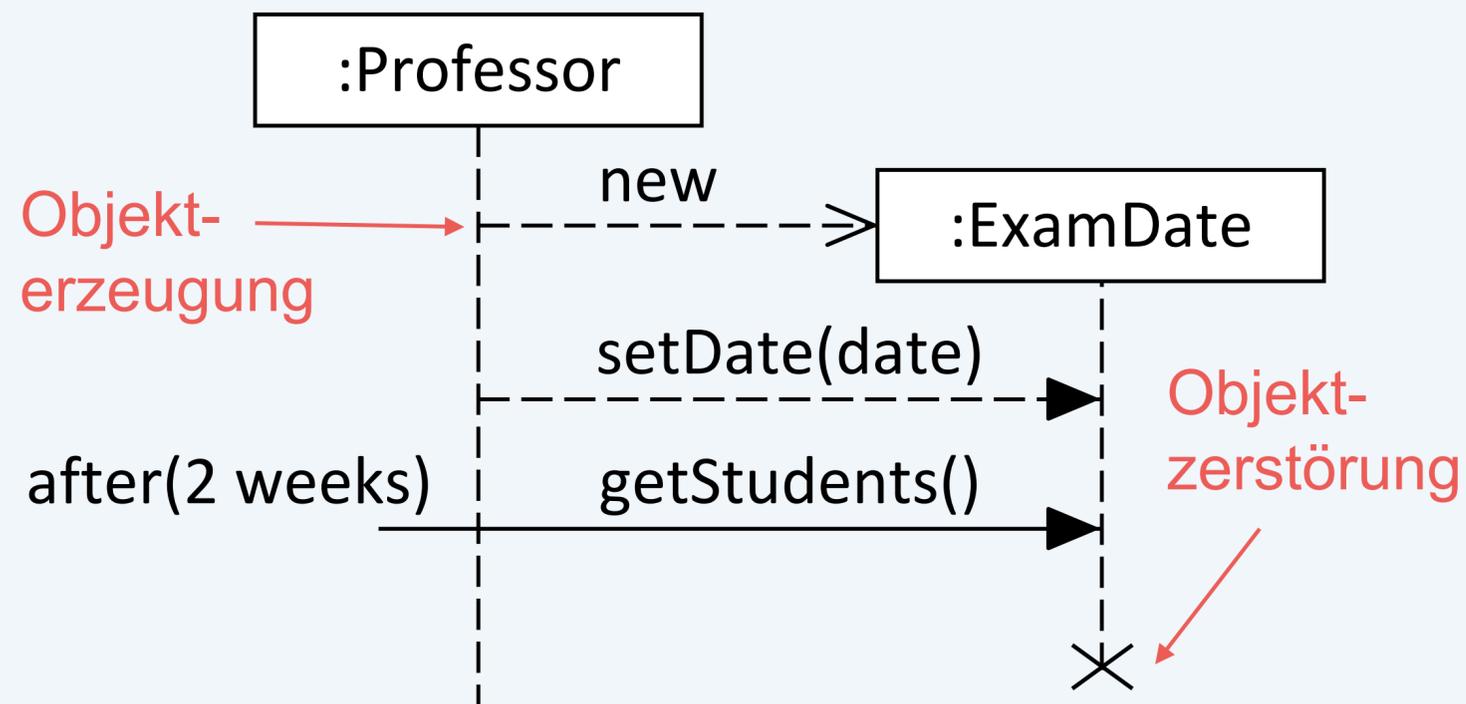
par: Parameterliste

val: Rückgabewert



Nachricht: Spezielle Nachrichtenarten (1/2)

- Objekterzeugung
 - Kommunikationspartner*in wird im Laufe der Interaktionsfolge erzeugt
 - Schlüsselwort **new**
- Objektzerstörung
 - Objekt wird gelöscht
 - Großes „X“ am Ende der Lebenslinie



Nachricht: Spezielle Nachrichtenarten (2/2)

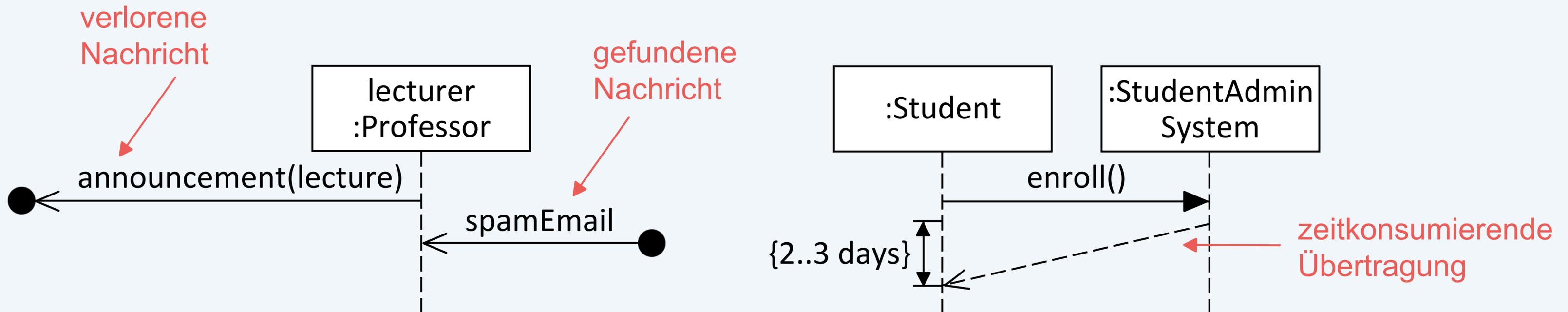
- Verlorene Nachricht

Senden einer Nachricht an unbekanntem/unbekannte oder irrelevanten/irrelevante Kommunikationspartner*in

- Gefundene Nachricht

Empfang einer Nachricht von unbekanntem/unbekannter oder irrelevantem/irrelevanter Kommunikationspartner*in

- Zeitkonsumierende Übertragung

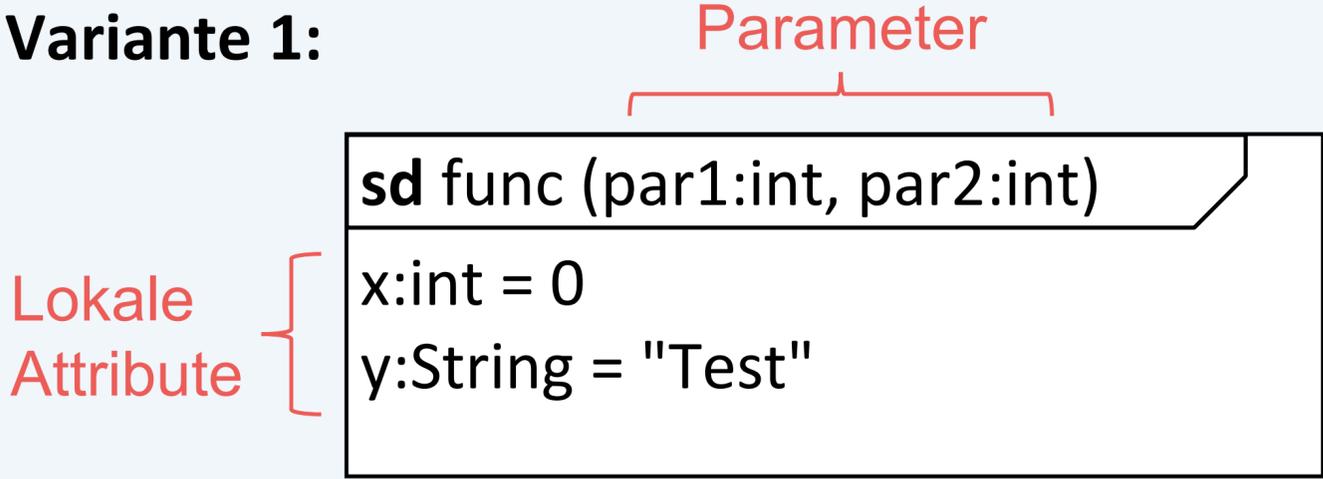


Basiskonzepte – Parameter, lokale Attribute

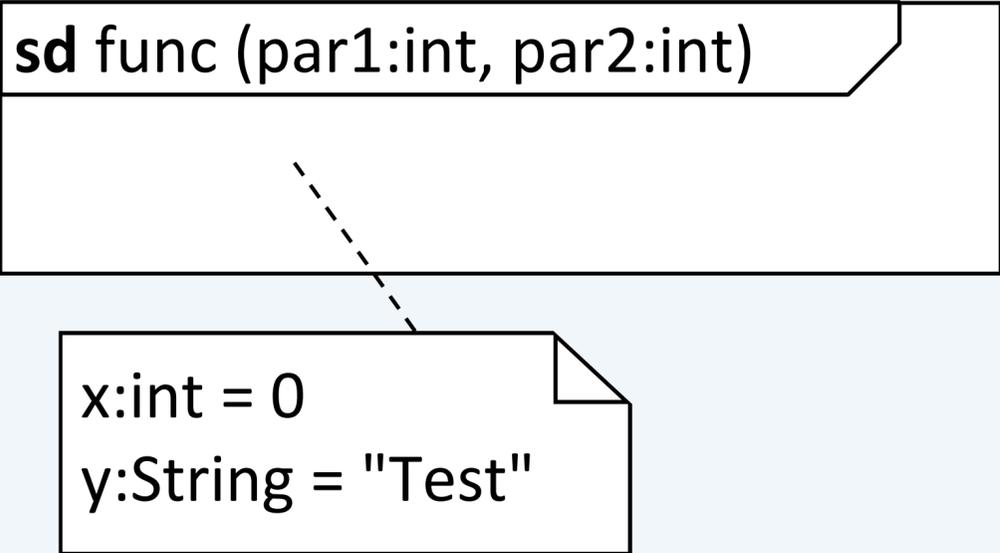
- Darstellung von Parametern und lokalen Attributen
- Beispiel: Modellierung der Operation **func**

```
void func (int par1, int par2) {
    int x = 0;
    String y = "Test";
    ...
}
```

Variante 1:



Variante 2:



INGO

Sequenzdiagramm

Die Zeiteinschränkung und die Zustandsinvariante

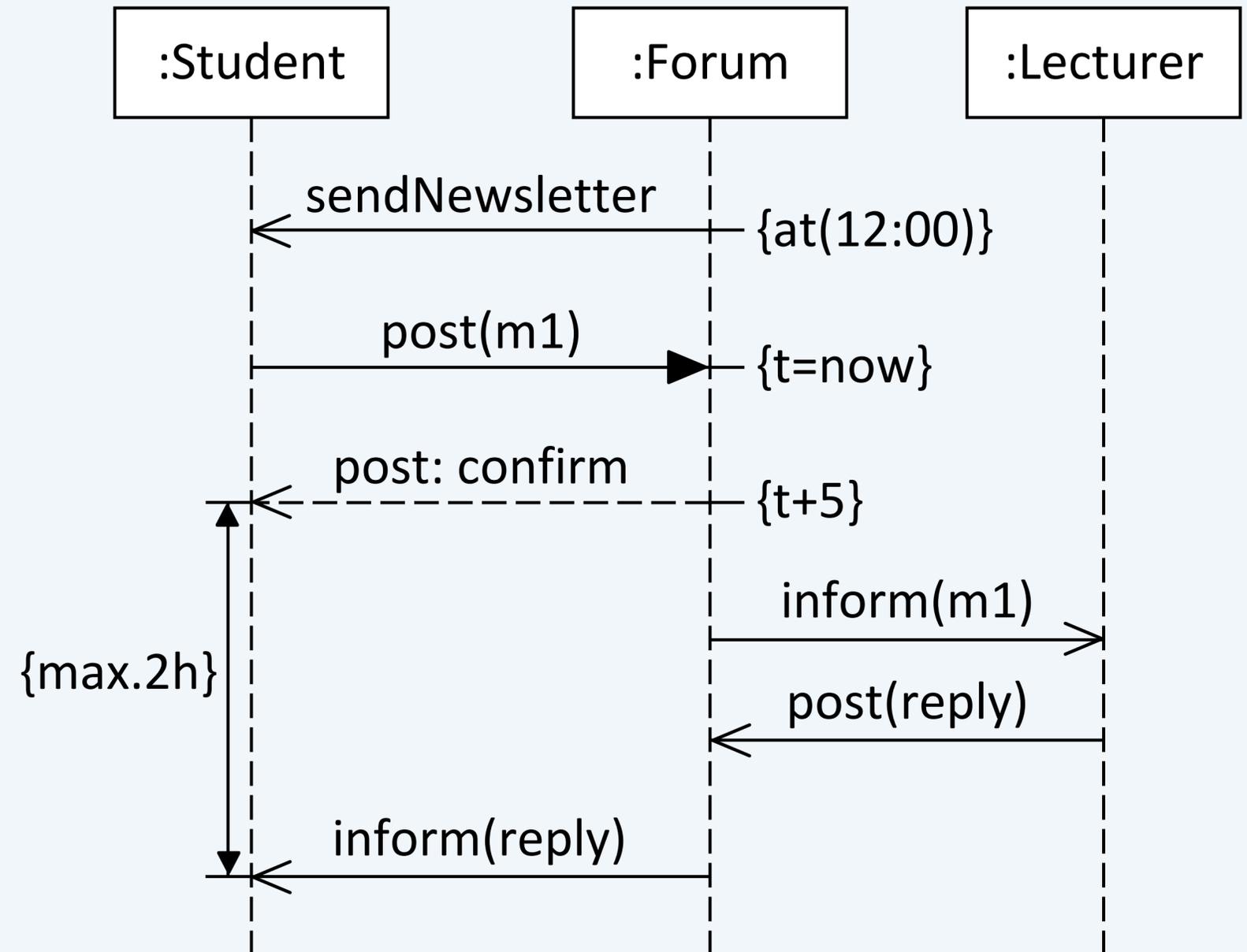


INGO

Christian Huemer und Marion Scholz

Zeiteinschränkungen

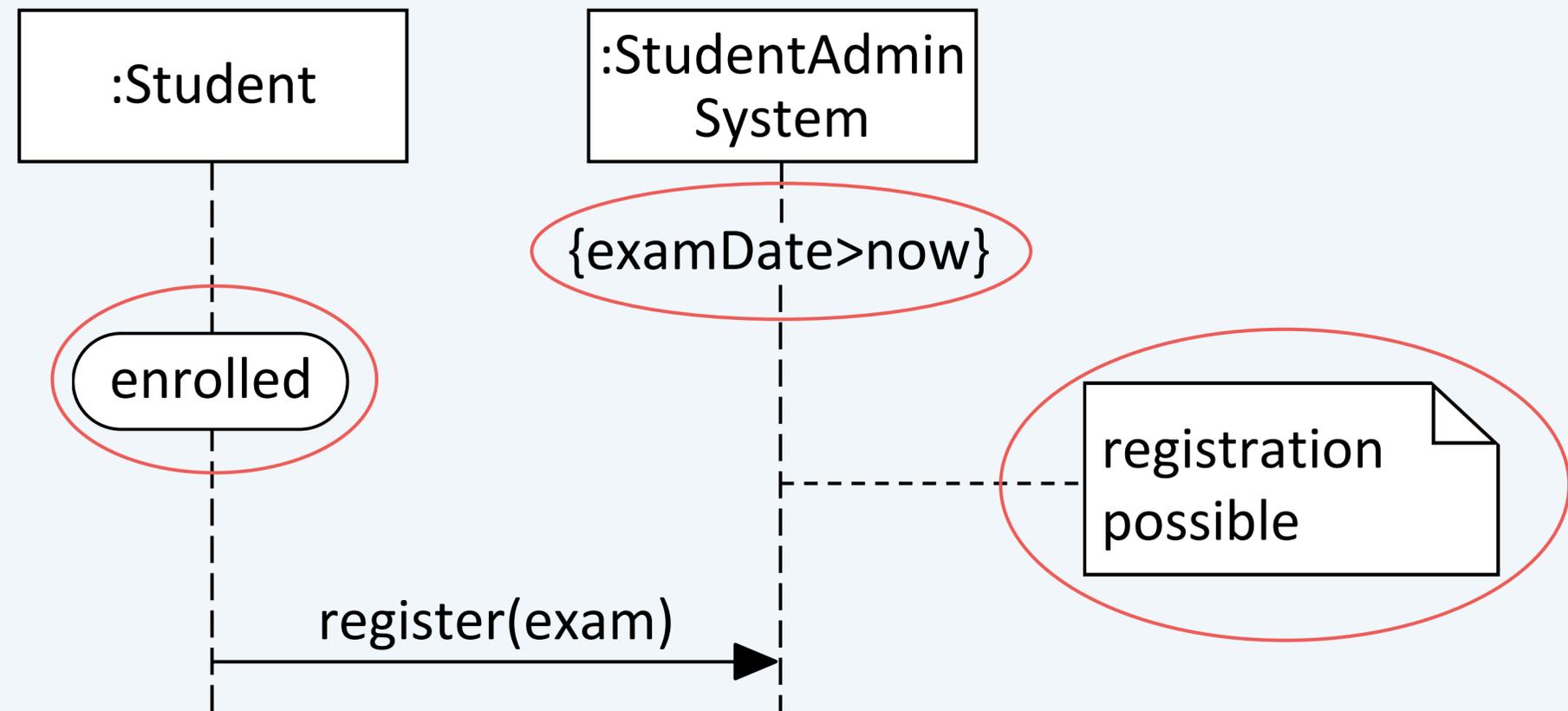
- Arten
 - **Zeitpunkt** einer Ereignisspezifikation
 - absolut: z.B. `at(12.00)`
 - relativ: z.B. `after(5sec)`
 - **Zeitintervall** zwischen zwei Ereigniseintritten
 - z.B. `{12.00 .. 13.00}`
- Vordefinierte Aktionen zur Zeitberechnung
 - `now`: aktuelle Zeit
 - `duration`: Berechnung einer Zeitdauer der Nachrichtenübertragung
 - Erhaltene Werte müssen Variablen zugewiesen werden
 - Variablen können in Zeitausdrücken verwendet werden



Zustandsinvariante

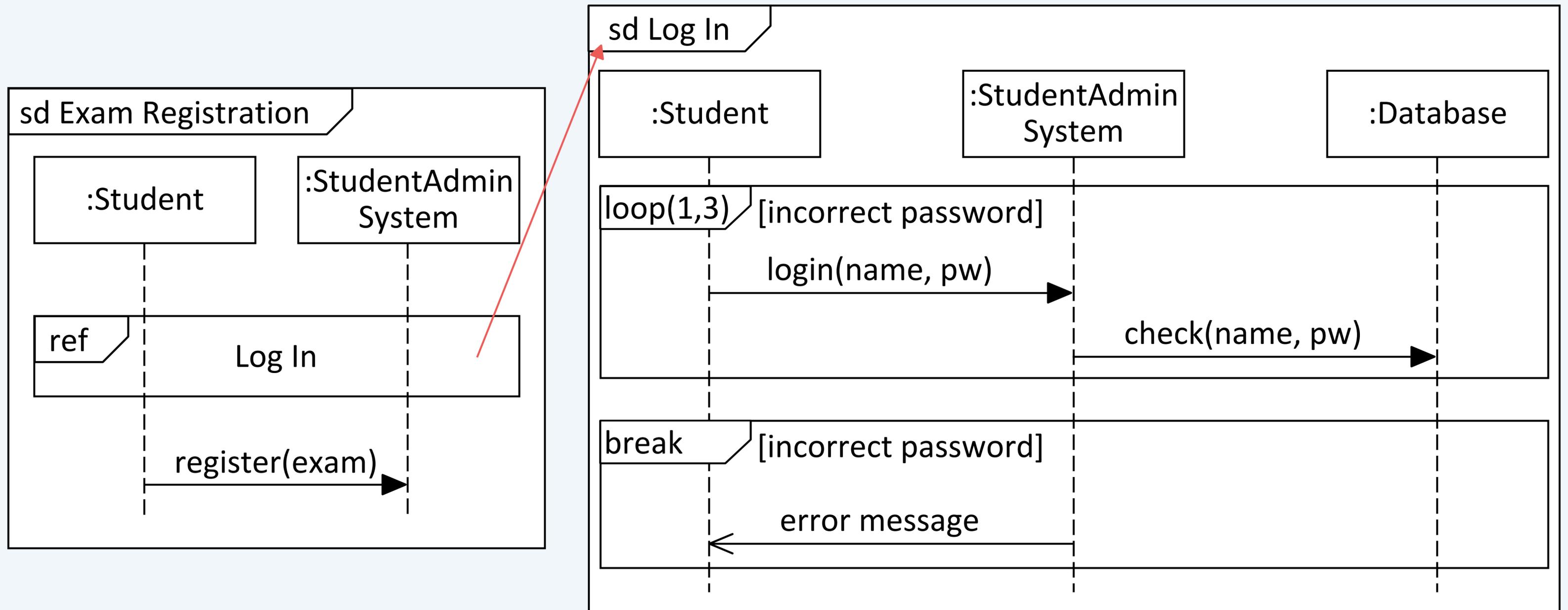
- **Zusicherung**, dass eine bestimmte **Bedingung** zu einem bestimmten Zeitpunkt **erfüllt** ist
- Bezieht sich immer auf eine bestimmte Lebenslinie
- Wird vor Eintritt des darauf folgenden Ereignisses ausgewertet
- Falls Zustandsinvariante nicht erfüllt ist \Rightarrow Fehler

- Drei Notationsvarianten:



Interaction Reference

- Das Sequenzdiagramm **Log In** wird über eine Interaktionsreferenz in das Sequenzdiagramm **Exam Registration** eingebunden



INGO

Sequenzdiagramm Die Kombinierten Fragmente, Teil 1



INGO

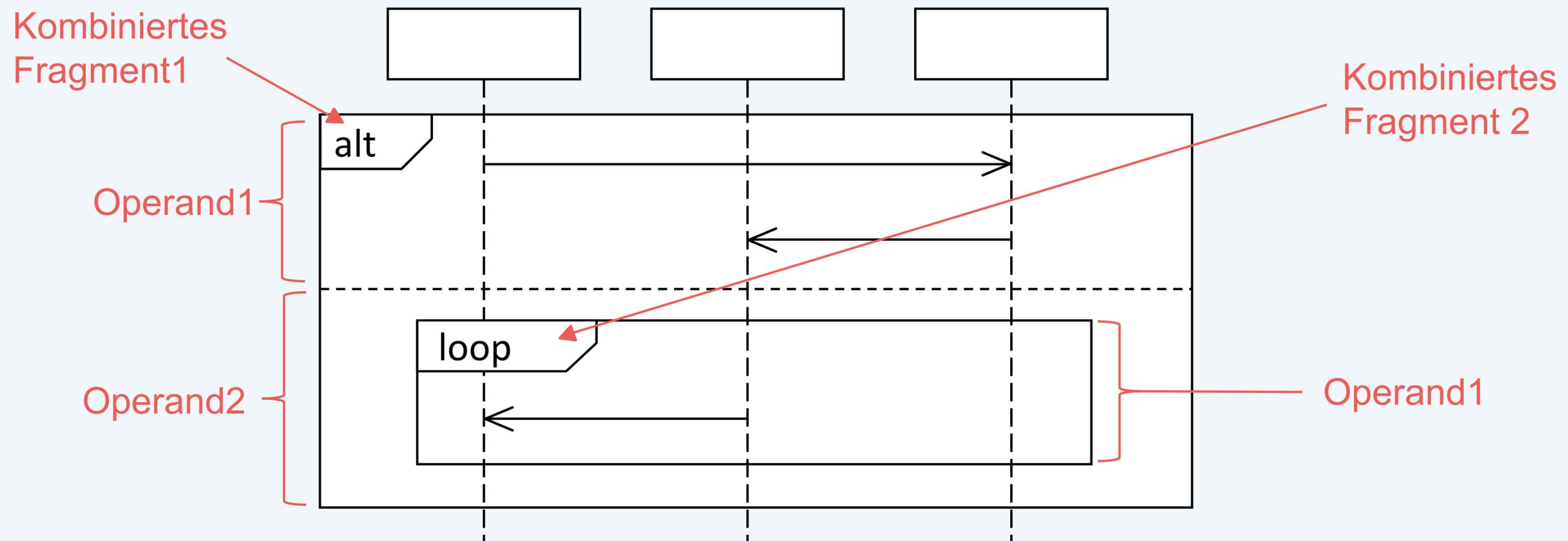
Christian Huemer und Marion Scholz

Kombinierte Fragmente

- Modellierung von **Kontrollstrukturen**
- Bestandteile: Operator und Operanden
- **Operator**
 - Definiert Art des kombinierten Fragments
 - 12 vordefinierte Operatoren
- **Operand**
 - Ein Operator enthält 1 oder mehrere Operanden, je nach Operatorart
 - Umfasst Interaktionen, kombinierte Fragmente und Referenzen auf Sequenzdiagramme

Kombinierte Fragmente – Notation

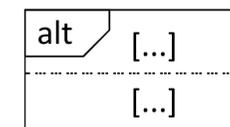
- Kombiniertes Fragment wird mit Rahmen dargestellt
- Art des Fragments wird durch Operator im Pentagon festgelegt
 - default: `seq`
- Operanden werden durch gestrichelte Linien voneinander getrennt
- 12 vordefinierte Arten von Operatoren



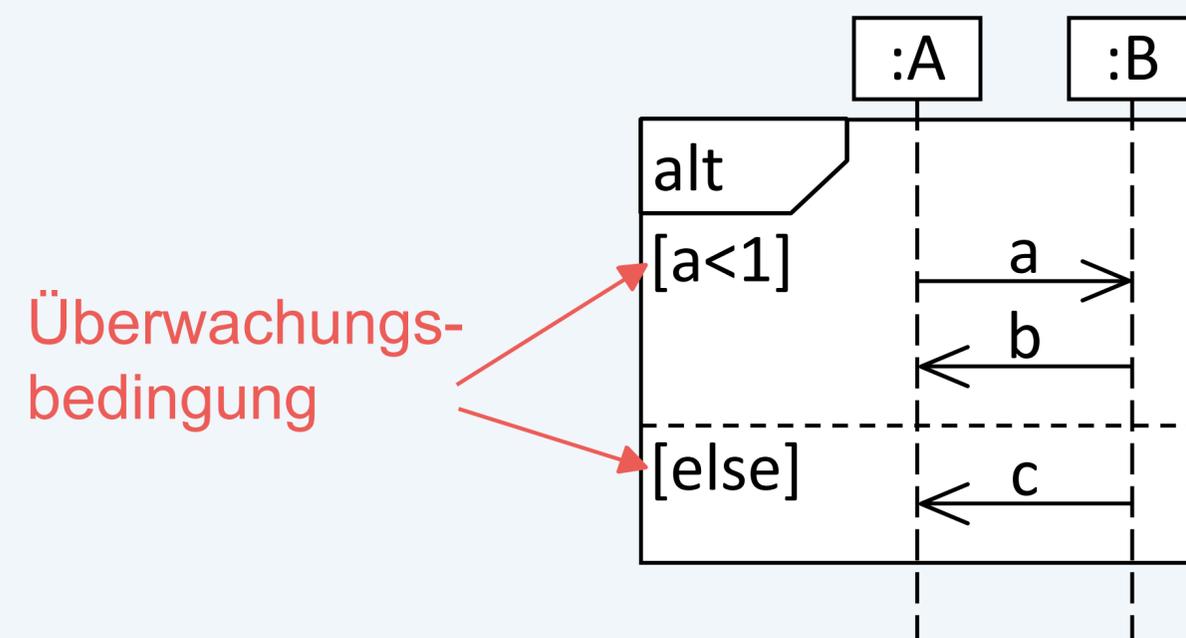
Kombinierte Fragmente – Operatorarten

	Operator	Zweck
Verzweigungen und Schleifen	alt	Alternative Interaktionen
	opt	Optionale Interaktionen
	break	Ausnahme Interaktionen
	loop	Iterative Interaktionen
Nebenläufigkeit und Ordnung	seq	Sequentielle Interaktionen mit schwacher Ordnung (Default)
	strict	Sequentielle Interaktionen mit strenger Ordnung
	par	Nebenläufige Interaktionen
	critical	Atomare Interaktionen
Filterungen und Zusicherungen	ignore	Irrelevante Interaktionen
	consider	Relevante Interaktionen
	assert	Zugesicherte Interaktionen
	neg	Ungültige Interaktionen

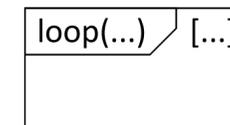
Verzweigungen u. Schleifen: **alt**-Operator



- Darstellung von zwei oder mehreren **alternativen Interaktionsabläufen**
- Zur Laufzeit wird maximal ein Operand ausgeführt
- Auswahl eines Operanden anhand von Überwachungsbedingungen
- Überwachungsbedingung
 - Boolescher Ausdruck in eckigen Klammern
 - Default-Wert: **[true]**
 - Vordefiniert: **[else]**: Operand wird ausgeführt, falls die Bedingungen aller anderen Operanden nicht erfüllt sind

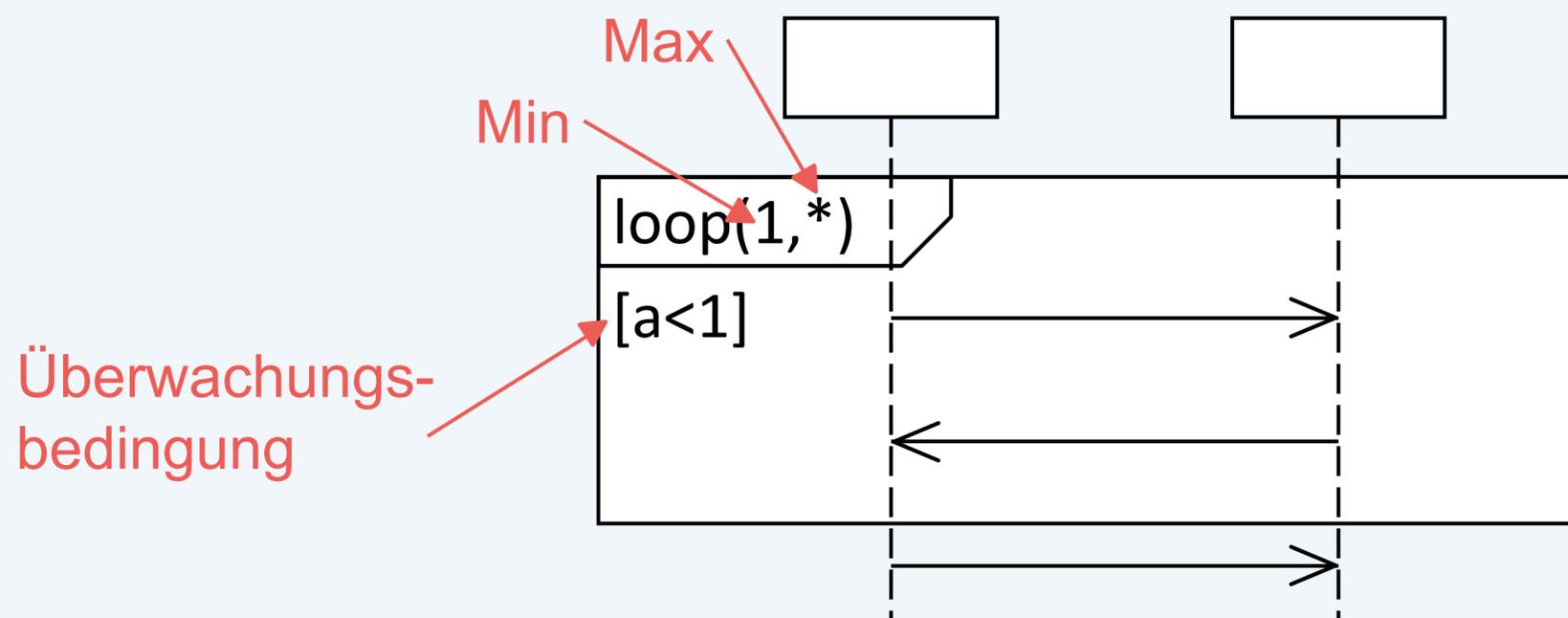


Verzweigungen u. Schleifen: loop-Operator



■ Darstellung einer **Schleife**

- Fragment enthält nur einen Operanden
- Ausführungshäufigkeit wird durch Zähler mit Unter- und Obergrenze dargestellt
- Überwachungsbedingung
 - optional
 - wird bei jedem Durchlauf überprüft, sobald die minimale Anzahl an Durchläufen stattgefunden hat



Notationsvarianten:

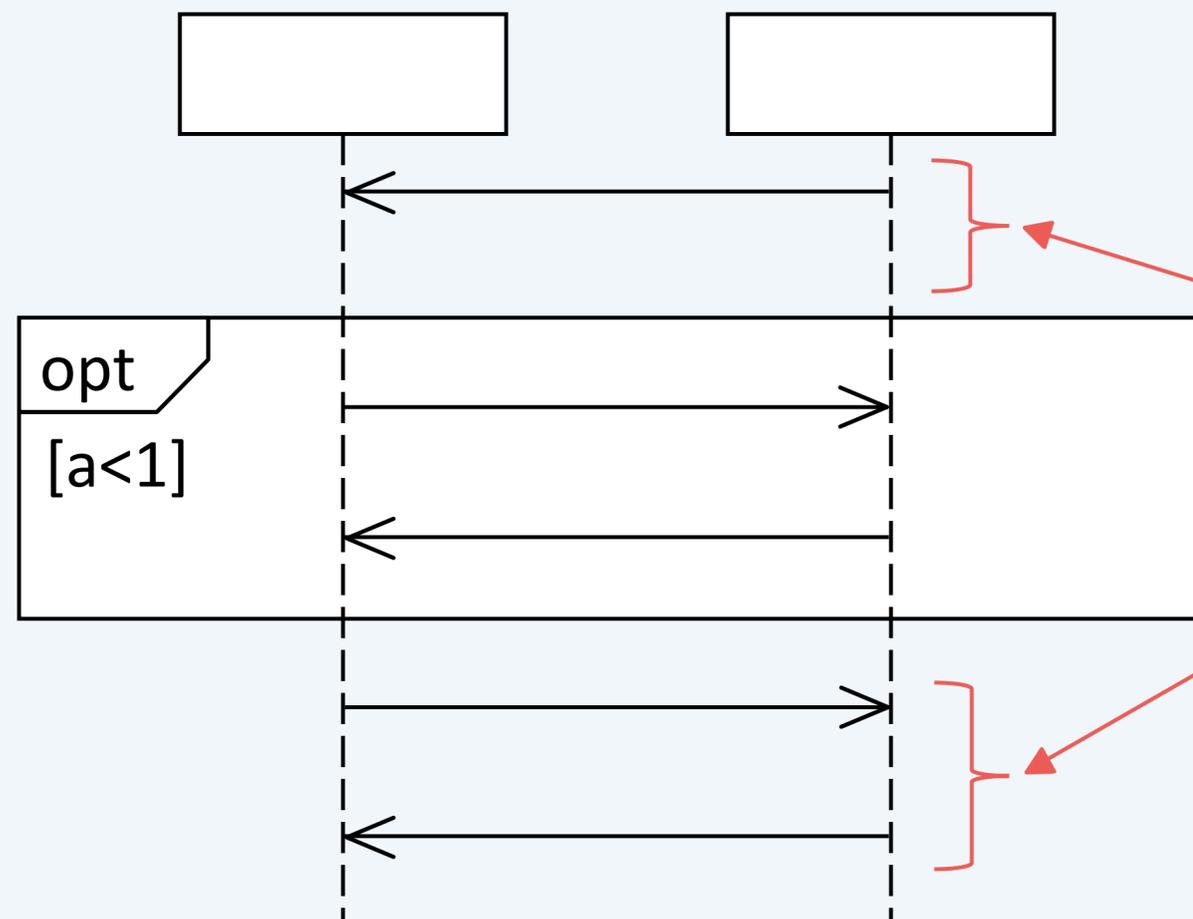
`loop(3,8) = loop(3..8)`

`loop(8,8) = loop(8)`

`loop = loop(*) = loop(0,*)`

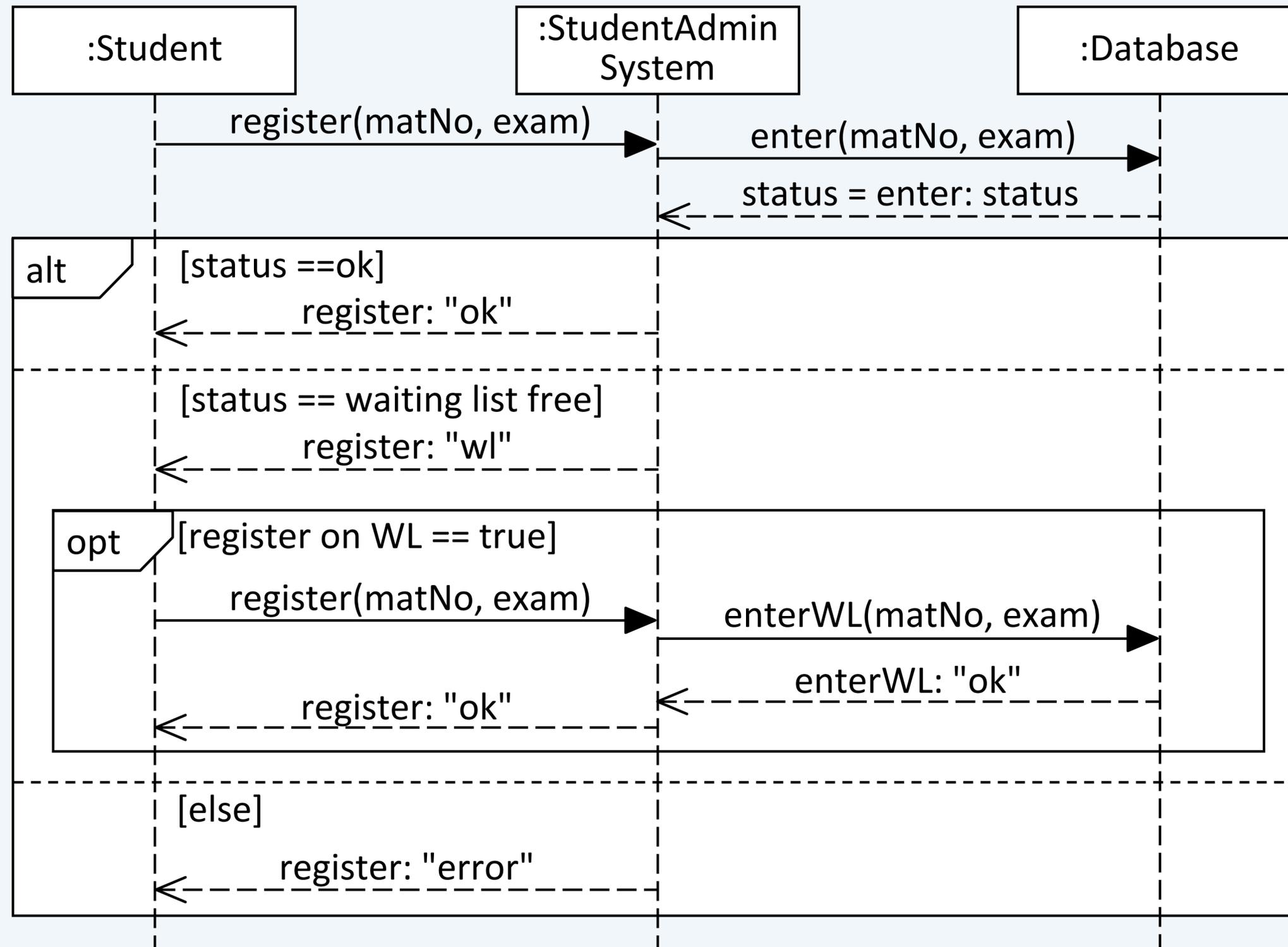
Verzweigungen u. Schleifen: **opt**-Operator

- **Optionale** Interaktionen
- Durch Überwachungsbedingung gesteuert
- Fragment wird nur aktiv, wenn Bedingung erfüllt ist
 - Modellierung von "wenn ..., dann ..."



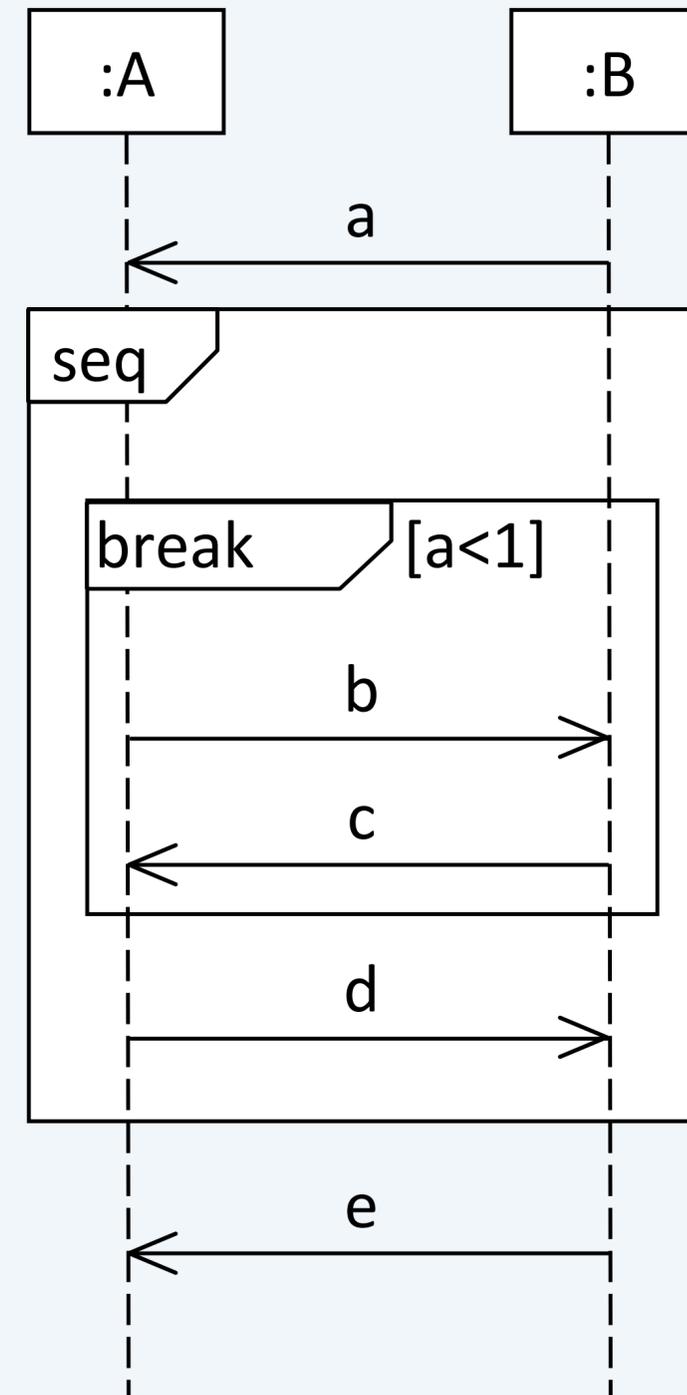
Interaktionen, die unabhängig von der Bedingung des **opt**-Fragments ausgeführt werden

opt- und alt-Operator – Beispiel



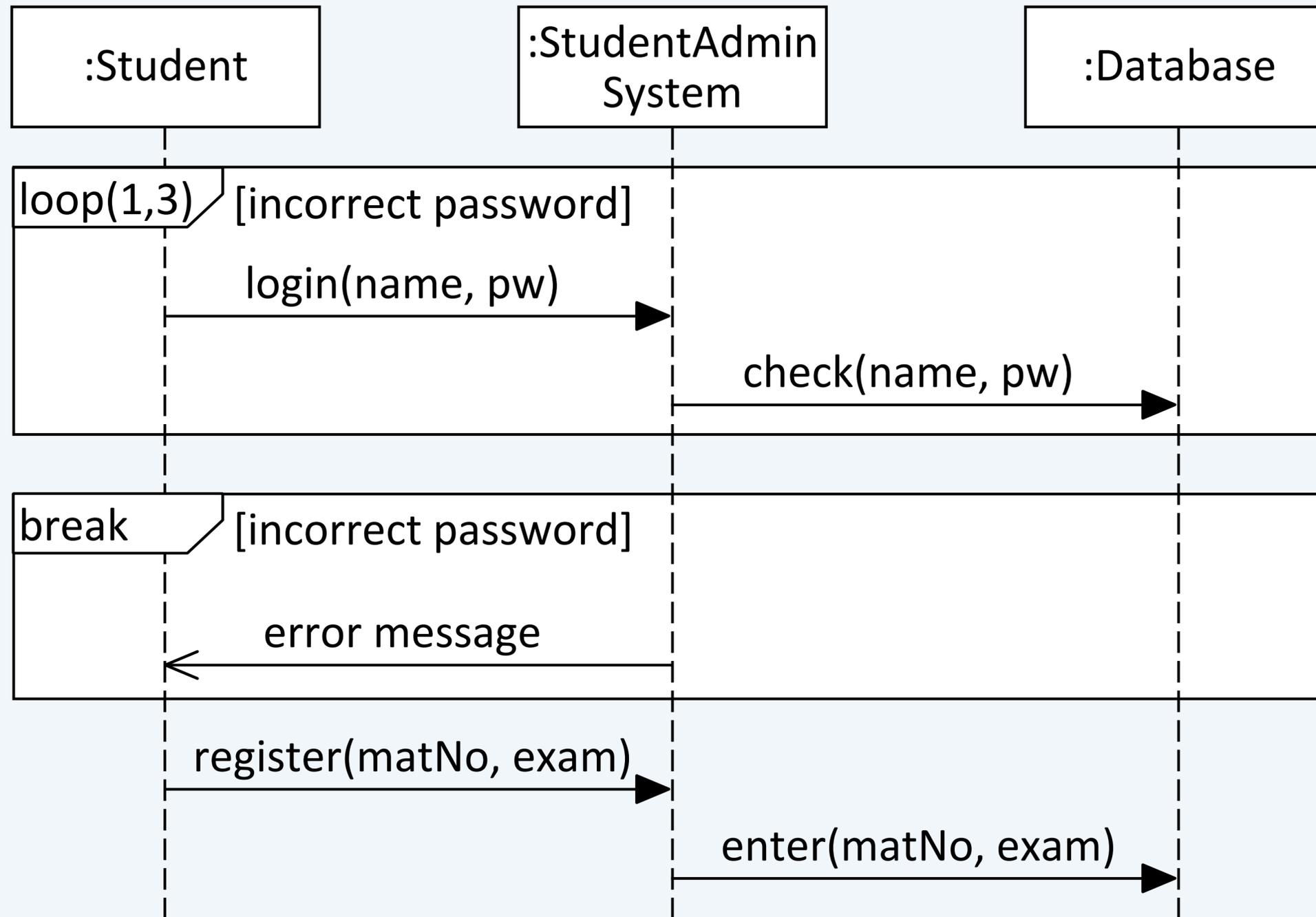
Verzweigungen u. Schleifen: **break**-Operator

- **Ausnahme-Interaktionen**
 - Durch Überwachungsbedingung gesteuert
 - Behandlung von Sonderfällen und Ausnahmen
 - Wenn die Bedingung wahr ist:
 - Interaktionen innerhalb des break werden ausgeführt
 - Interaktionen des umgebenden Fragments werden verworfen
 - Interaktionen des übergeordneten Fragments werden fortgesetzt
- ⇒ ≠ opt!



} Interaktionen, die im Falle des **break** nicht ausgeführt werden

loop- und break-Operator – Beispiel



INGO

Sequenzdiagramm Die Kombinierten Fragmente, Teil 2

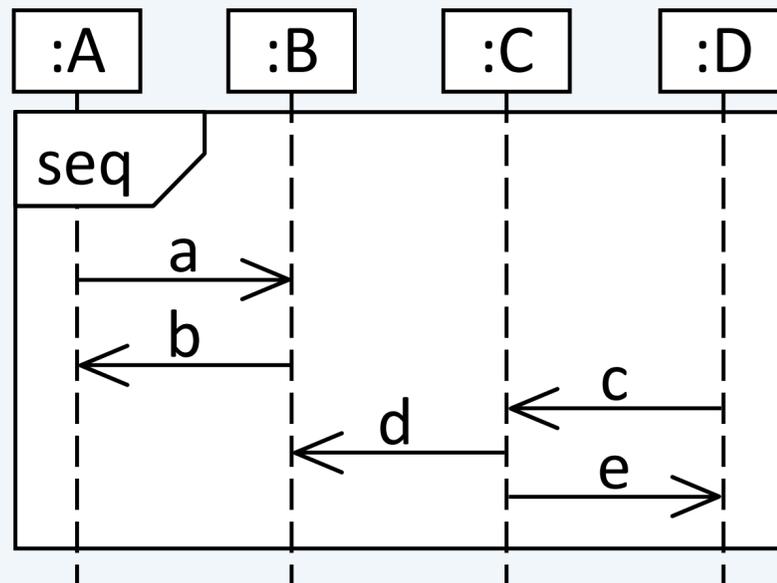


INGO

Christian Huemer und Marion Scholz

Nebenläufigkeit u. Ordnung: `seq`-Operator

- default
- Sequentielle Interaktion mit schwacher Ordnung
- mind. 1 Operand
- Reihenfolge der Ereignisseintritte:
 - Reihenfolge der Ereignisse pro Lebenslinie fix
 - Reihenfolge auf unterschiedlichen Lebenslinien nur signifikant, wenn ein Nachrichtenaustausch stattfindet
 - Operanden können zur „optischen Gruppierung“ eingesetzt werden, aber nicht um eine Ordnung zu erzwingen (\neq `strict`!)



Traces:

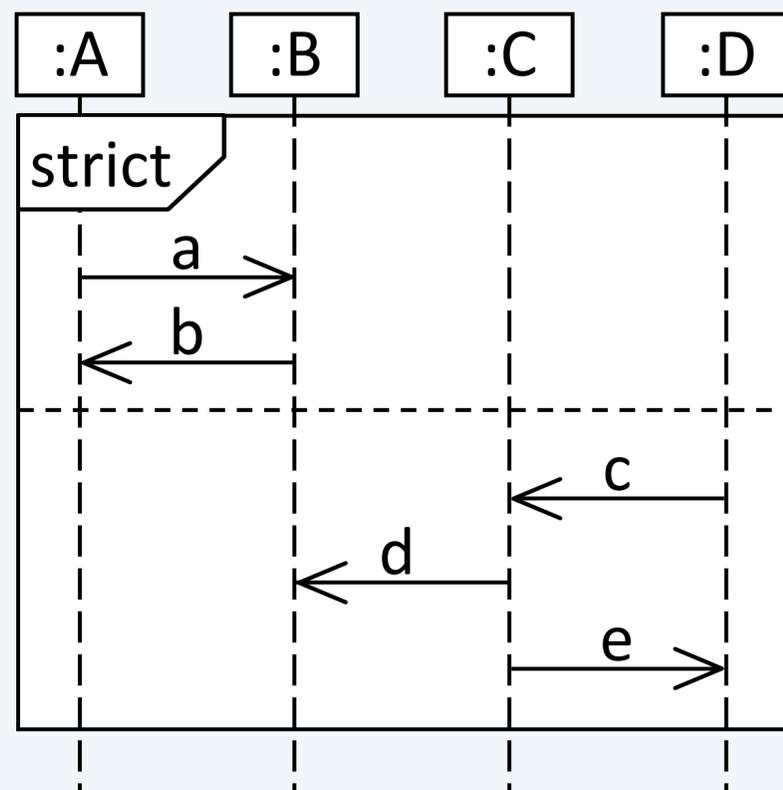
T01: **a** → **b** → **c** → **d** → **e**

T02: **a** → **c** → **b** → **d** → **e**

T03: **c** → **a** → **b** → **d** → **e**

Nebenläufigkeit und Ordnung: `strict`-Operator

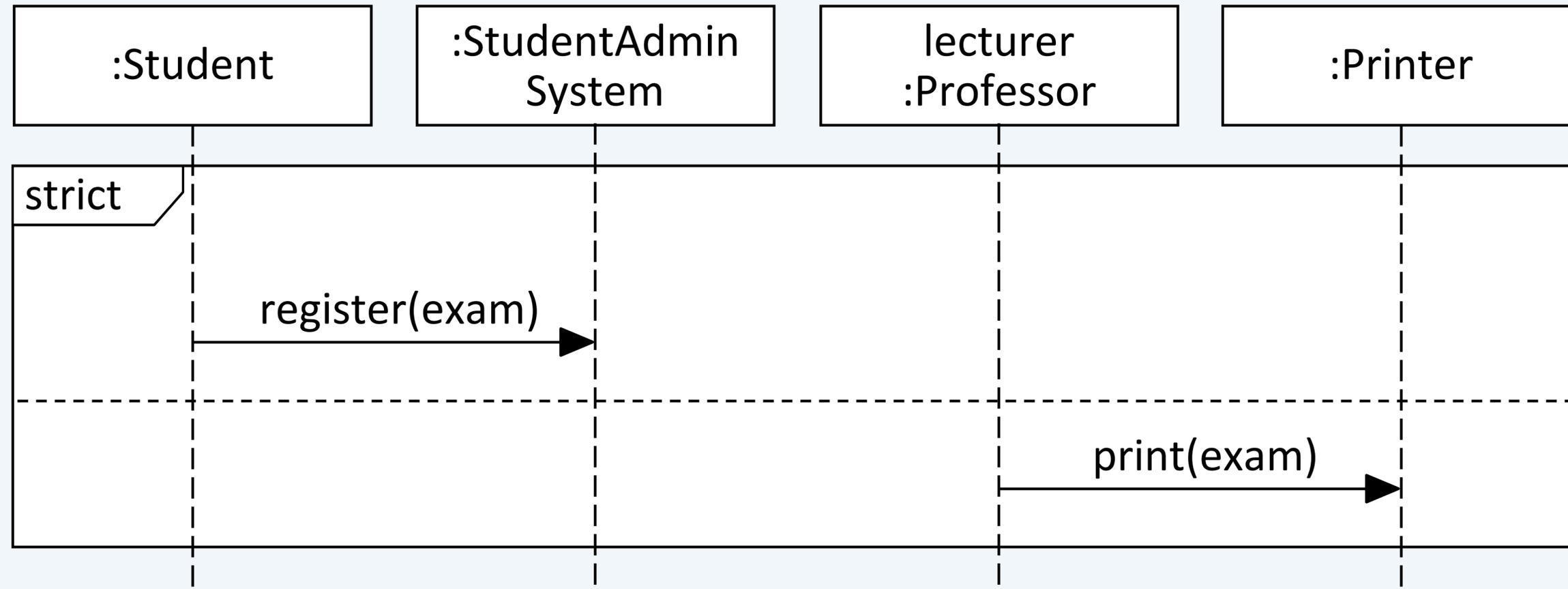
- Sequentielle Interaktion mit **strenger** Ordnung
- Reihenfolge auf unterschiedlichen Lebenslinien von unterschiedlichen Operanden ist signifikant



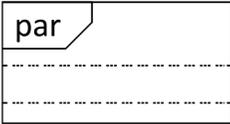
Traces:

T01: **a** → **b** → **c** → **d** → **e**

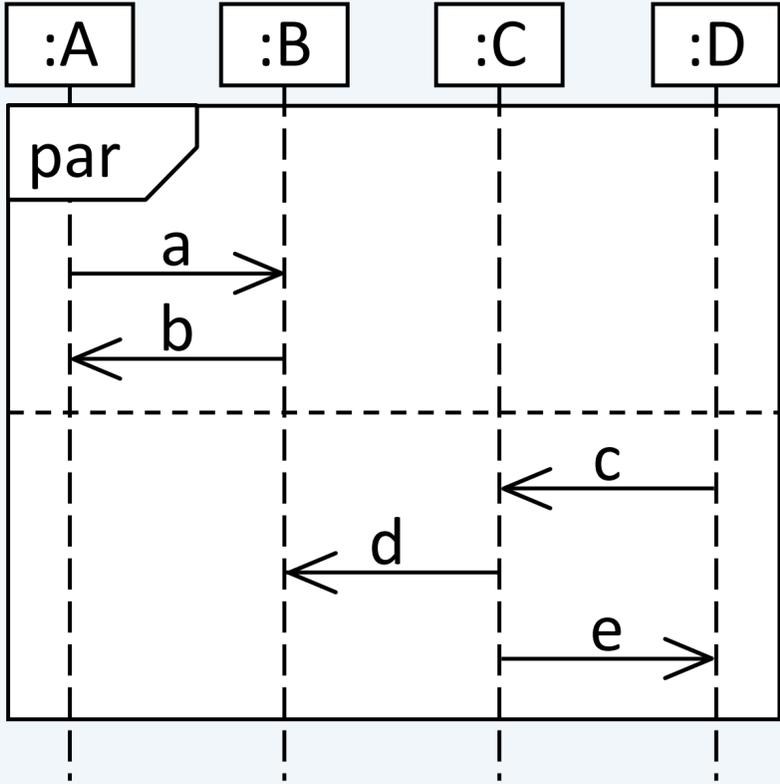
strict-Operator – Beispiel



Nebenläufigkeit und Ordnung: `par`-Operator



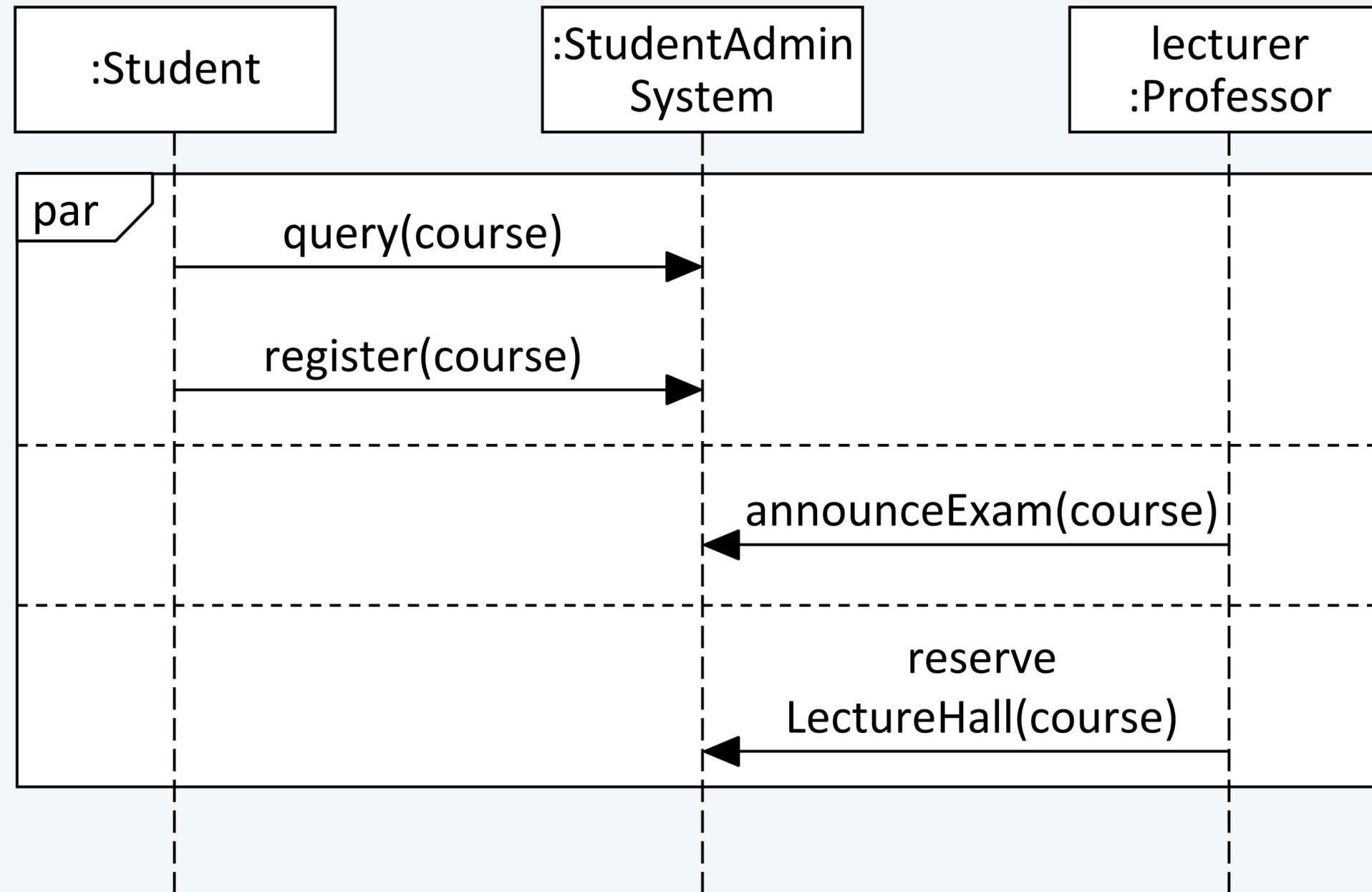
- Nebenläufige Interaktionen
 - Lokale Reihenfolge pro Operand muss erhalten bleiben
 - Reihenfolge der Operanden im Diagramm ist irrelevant!
 - mind. 2 Operanden



Traces:

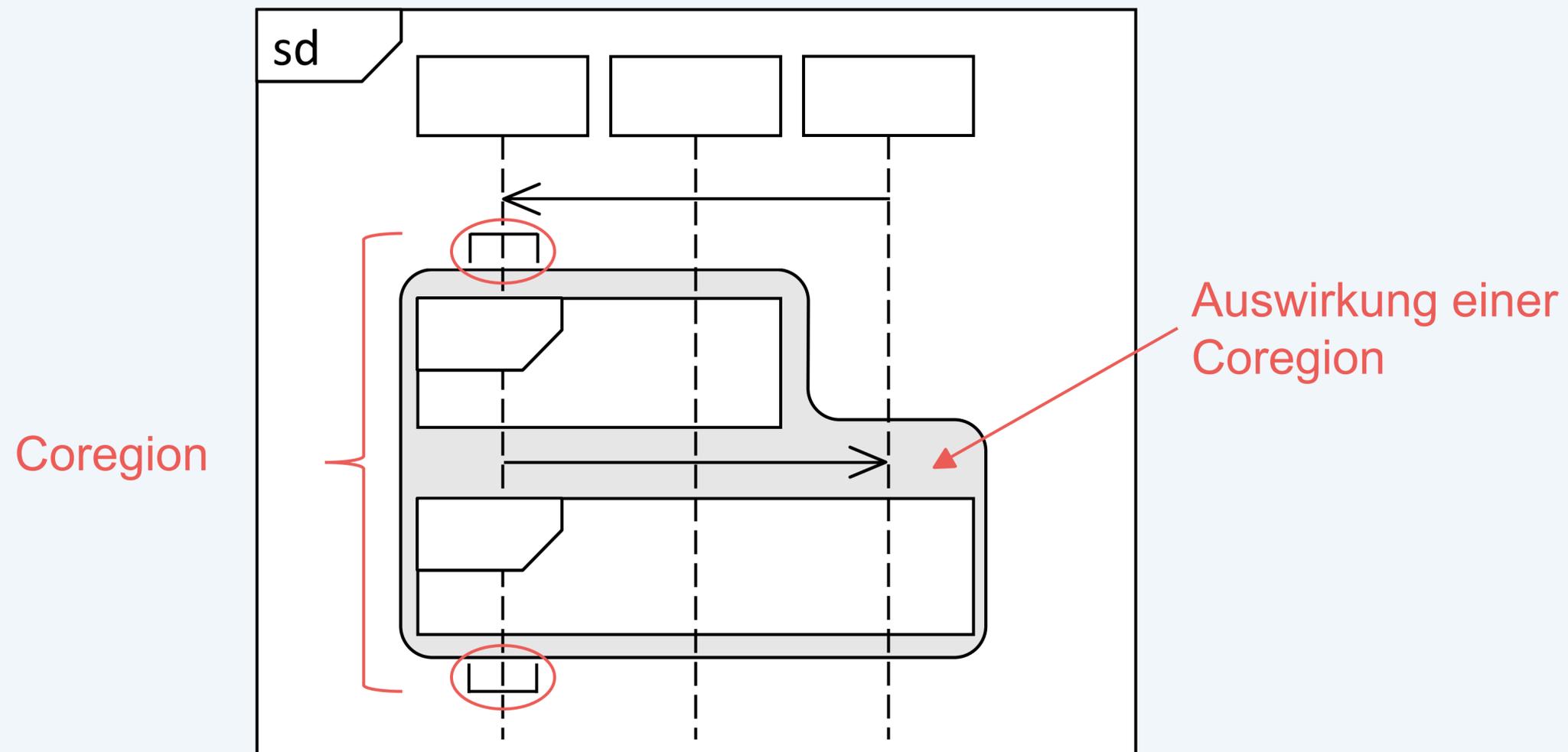
T01: **a** → **b** → **c** → **d** → **e**
T02: **a** → **c** → **b** → **d** → **e**
T03: **a** → **c** → **d** → **b** → **e**
T04: **a** → **c** → **d** → **e** → **b**
T05: **c** → **a** → **b** → **d** → **e**
T06: **c** → **a** → **d** → **b** → **e**
T07: **c** → **a** → **d** → **e** → **b**
T08: **c** → **d** → **a** → **b** → **e**
T09: **c** → **d** → **a** → **e** → **b**
T10: **c** → **d** → **e** → **a** → **b**

par-Operator – Beispiel

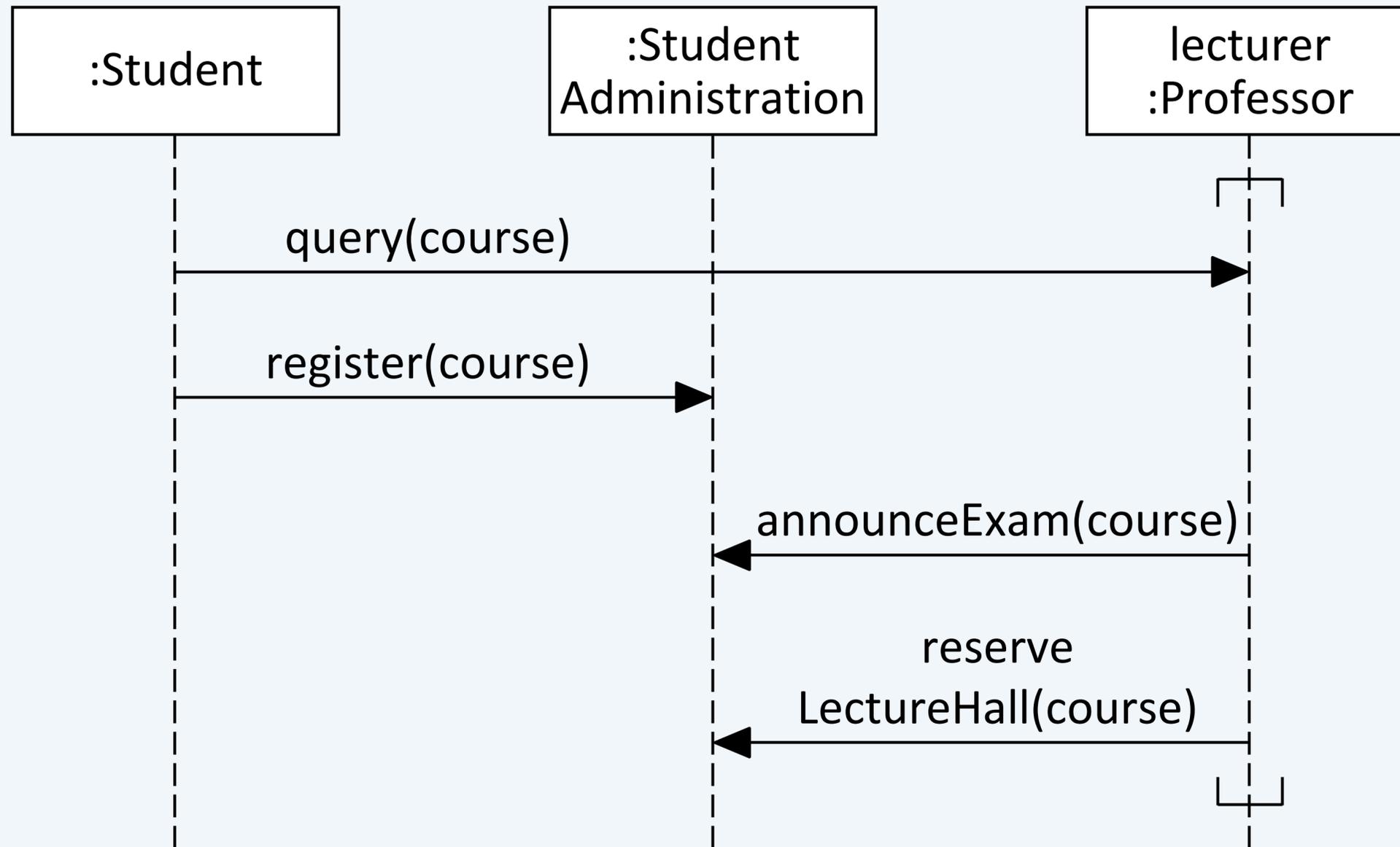


Nebenläufigkeit und Ordnung: Coregion

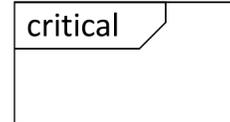
- Coregion: Darstellung von nebenläufigen Abläufen auf EINER Lebenslinie
- Reihenfolge der Ereigniseintritte innerhalb von Coregions ist auf keine Weise beschränkt ("Aufhebung der Zeitdimension")
- Coregion kann weitere kombinierte Fragmente beinhalten – kombinierte Fragmente können als Ganzes in beliebiger Reihenfolge ausgeführt werden



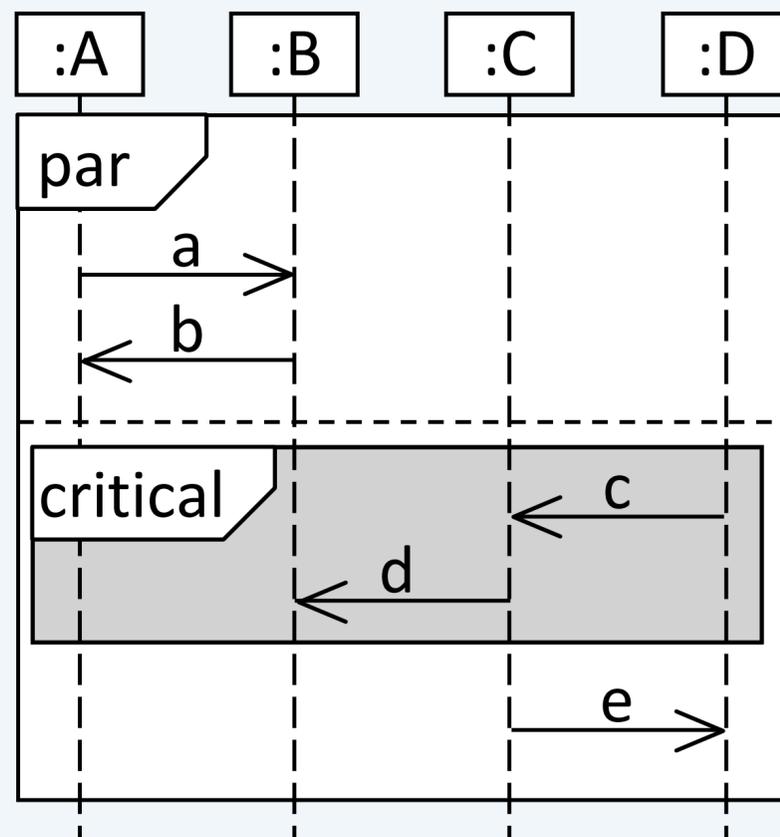
Coregion – Beispiel



Nebenläufigkeit und Ordnung: `critical`-Operator



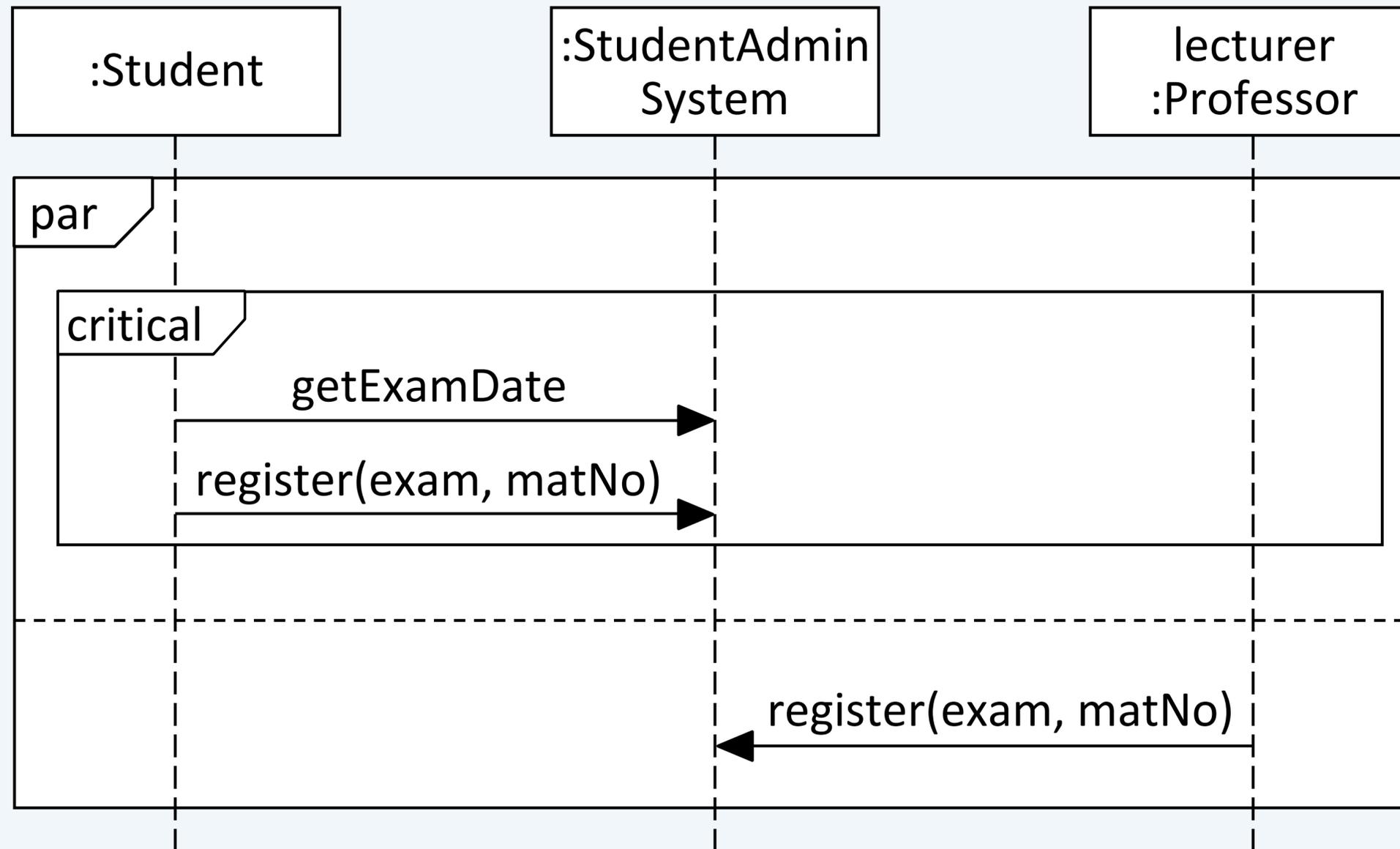
- Kritischer Bereich: atomarer (nicht unterbrechbarer) Interaktionsablauf
- Keine Beschränkung auf Interaktionen außerhalb des kritischen Bereichs



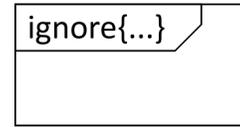
Traces:

T01: **a** → **b** → **c** → **d** → **e**
T02: **a** → **c** → **d** → **b** → **e**
T03: **a** → **c** → **d** → **e** → **b**
T04: **c** → **d** → **a** → **b** → **e**
T05: **c** → **d** → **a** → **e** → **b**
T06: **c** → **d** → **e** → **a** → **b**

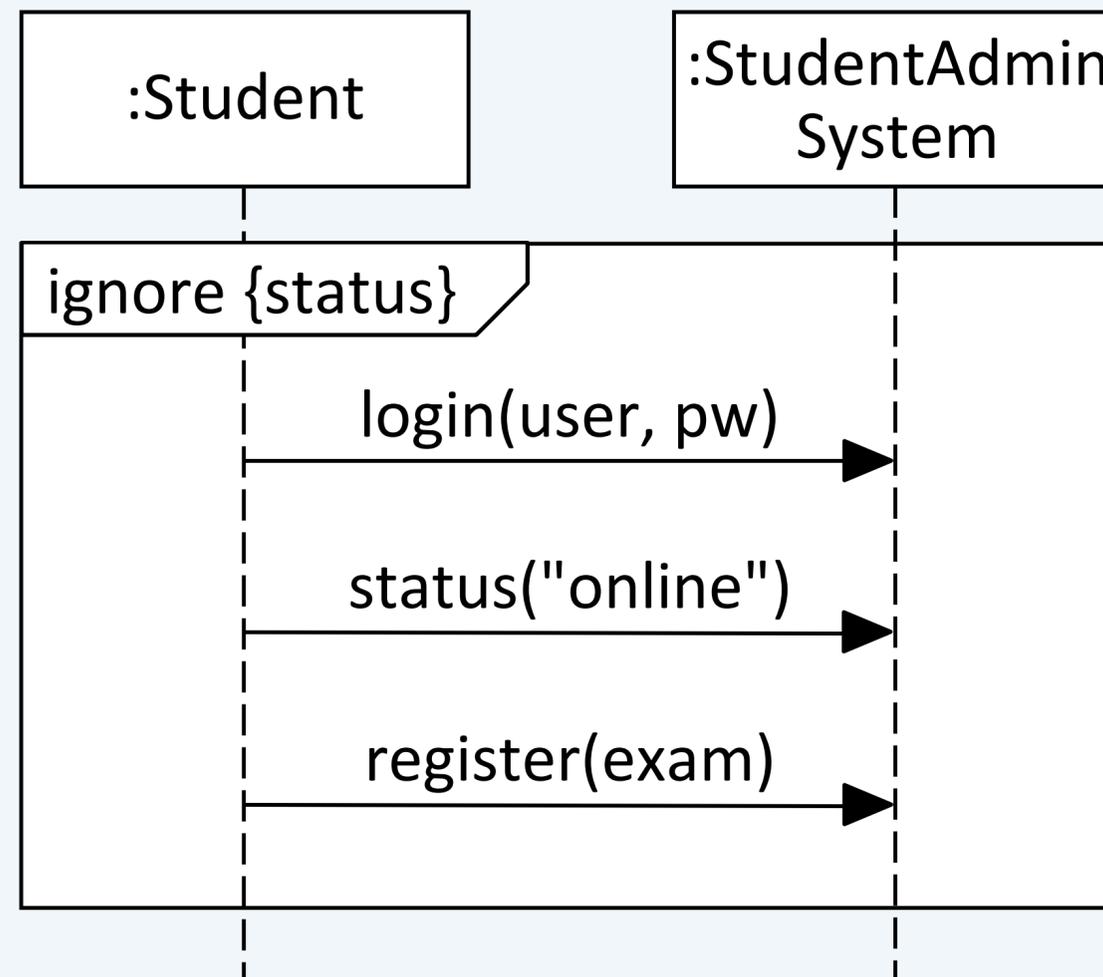
critical-Operator – Beispiel



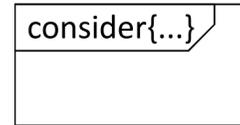
Filterungen u. Zusicherungen: `ignore`-Operator



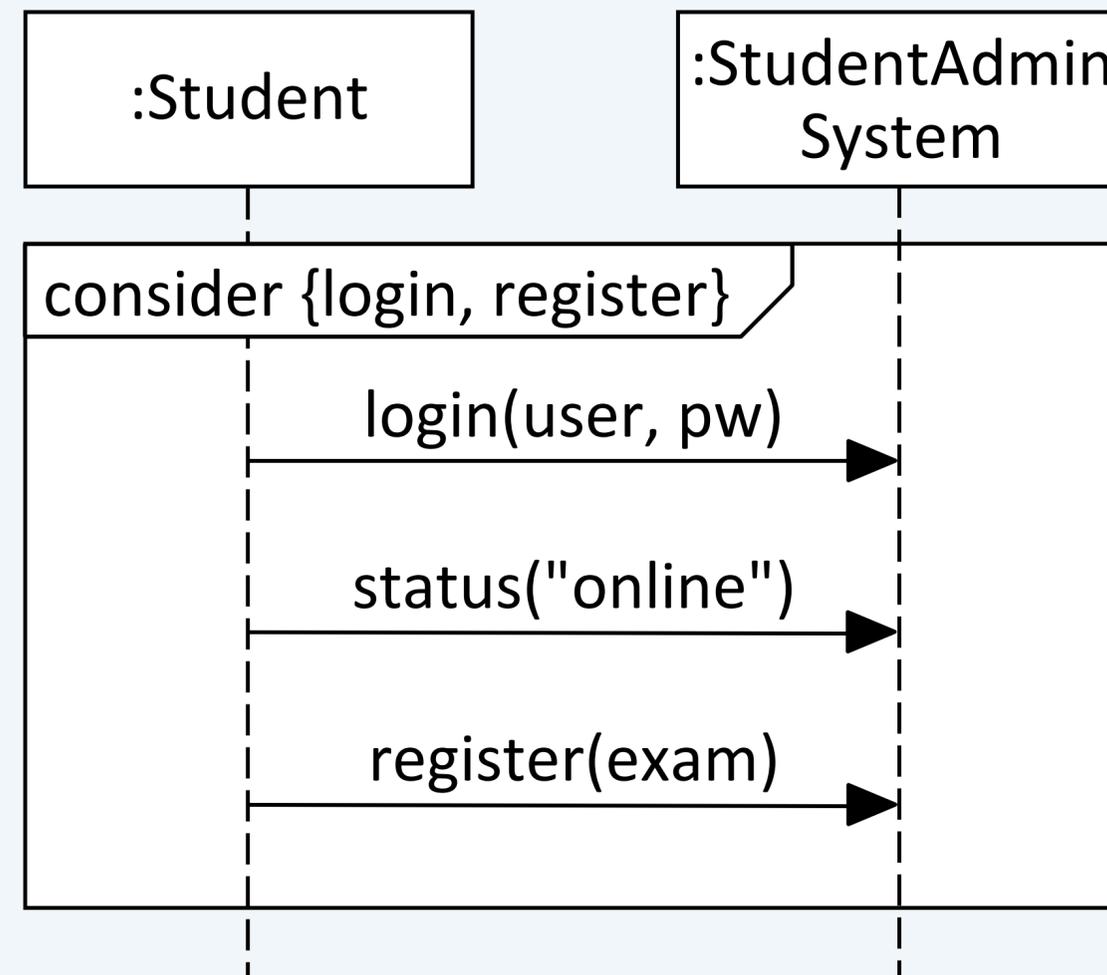
- Darstellung von irrelevanten Nachrichten
 - Modellierung aus technischen Gründen
 - Modellierung wegen syntaktischer Vollständigkeit
 - Nachrichten, die zur Laufzeit auftreten können (z.B. keep-alive Signale)



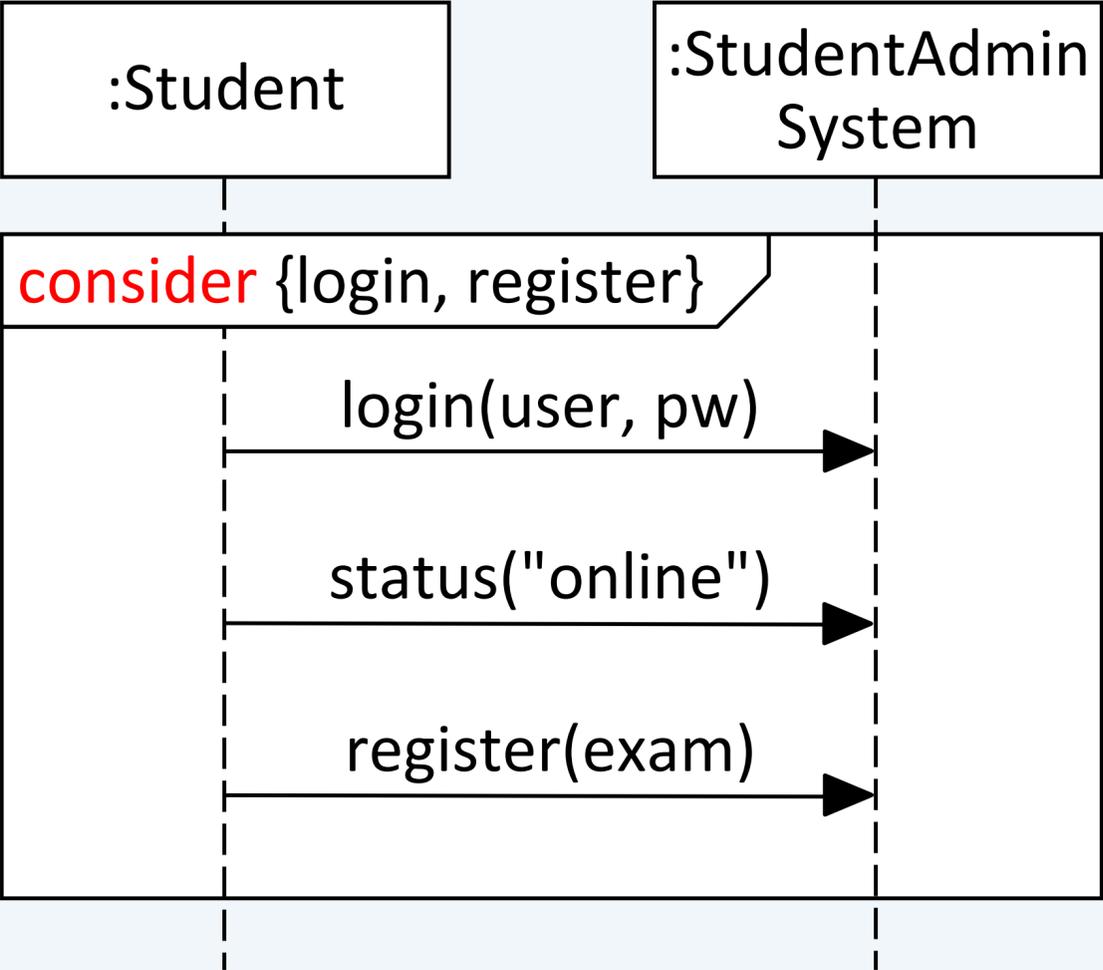
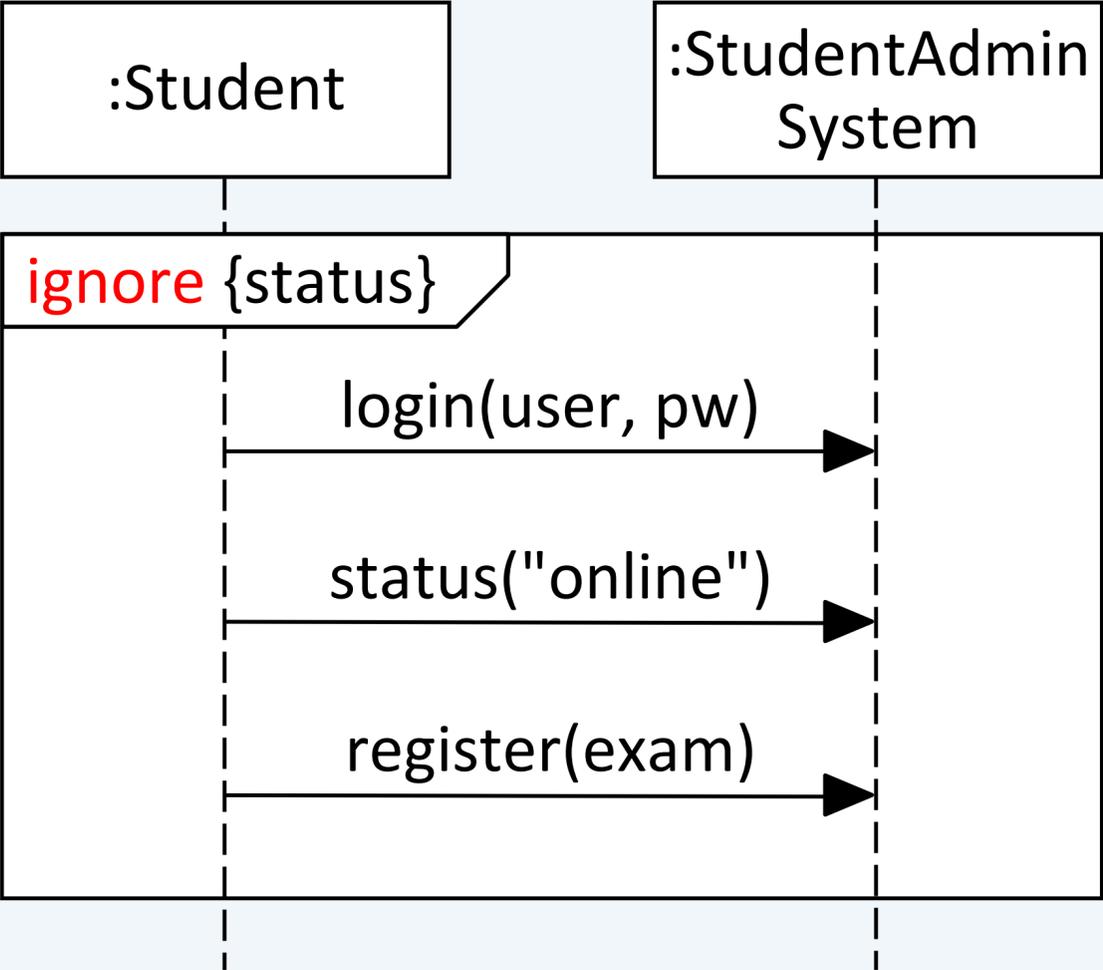
Filterungen u. Zusicherungen: **consider**-Operator



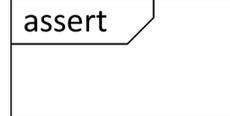
- Gegenstück von **ignore**
- Spezifikation von besonders relevanten Nachrichten
- andere Nachrichten im Operanden werden automatisch als nicht relevant eingestuft



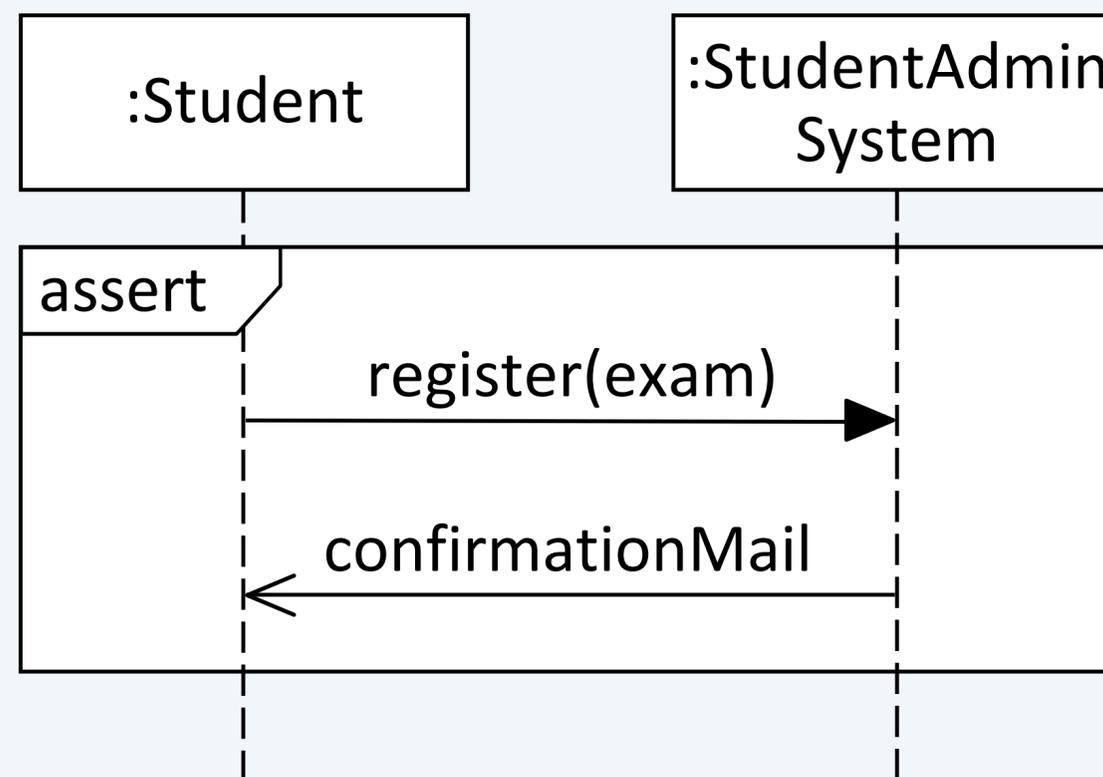
ignore VS. consider



Filterungen u. Zusicherungen: **assert**-Operator



- Kennzeichnung der Interaktion als verpflichtend
- Abweichungen, die im Diagramm nicht berücksichtigt sind, aber in der Realität auftreten, sind nicht zulässig
 - ⇒ Forderung von getreuer Abbildung in der Implementierung
- 1 Operand



Filterungen und Zusicherungen: neg-Operator

- Abbildung eines ungültigen Interaktionsablaufs
- Situationen, die in dieser Form nicht eintreten dürfen
- Genau 1 Operand
- Zweck
 - Explizit auf häufig auftretende Fehler hinweisen
 - Relevante, aber falsche Abläufe abbilden

