UML Sequenzdiagramm

Objektorientierte Programmierung

Agenda

- Übersicht
  - Beispiel
- Syntax
  - Klassen, Objekte, Lebenslinien und Ausführung
  - Nachrichten und Selbstaufruf
  - Verzweigung und Schleifen
- Übungsaufgabe

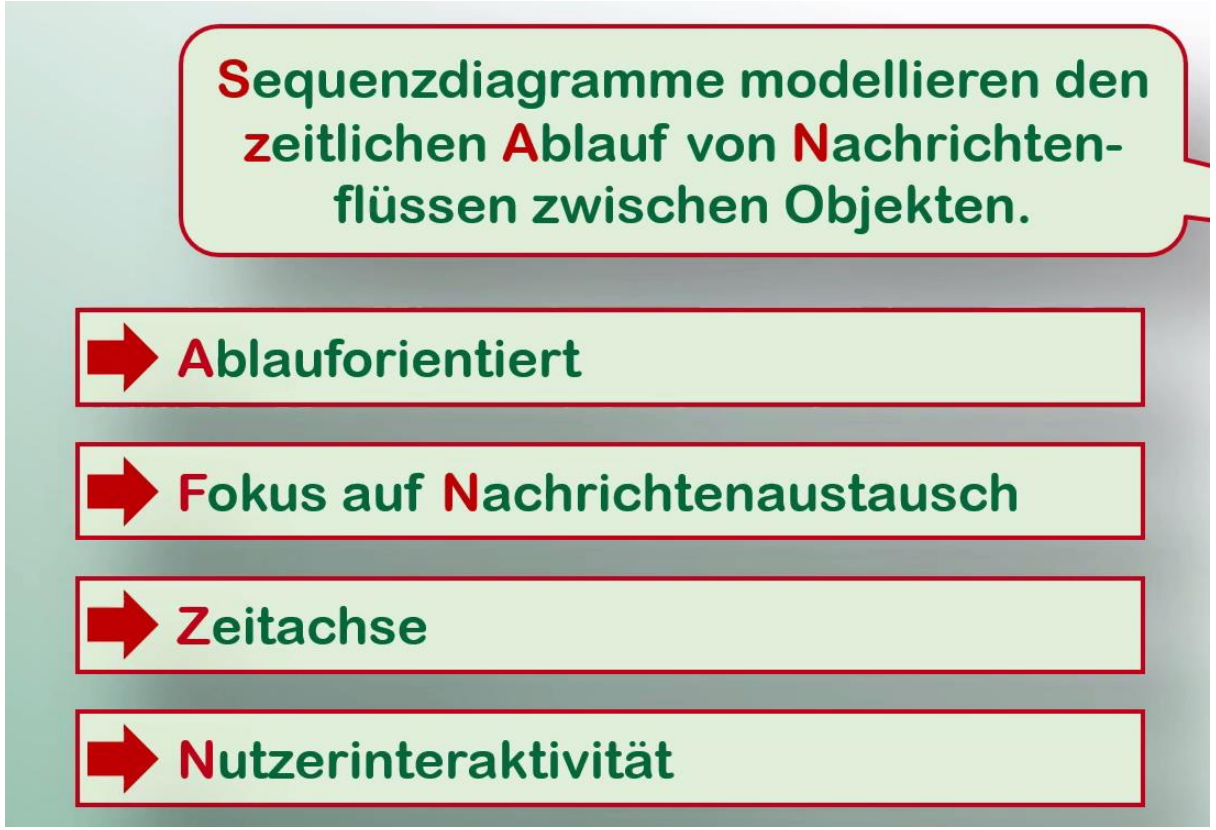
Übersicht

Übersicht

Ein Sequenzdiagramm (Sequence Diagram) wird verwendet, um den zeitlichen Ablauf von Methodenaufrufen (Nachrichtenflüssen) und Kommunikation zwischen Objekten zu modellieren.

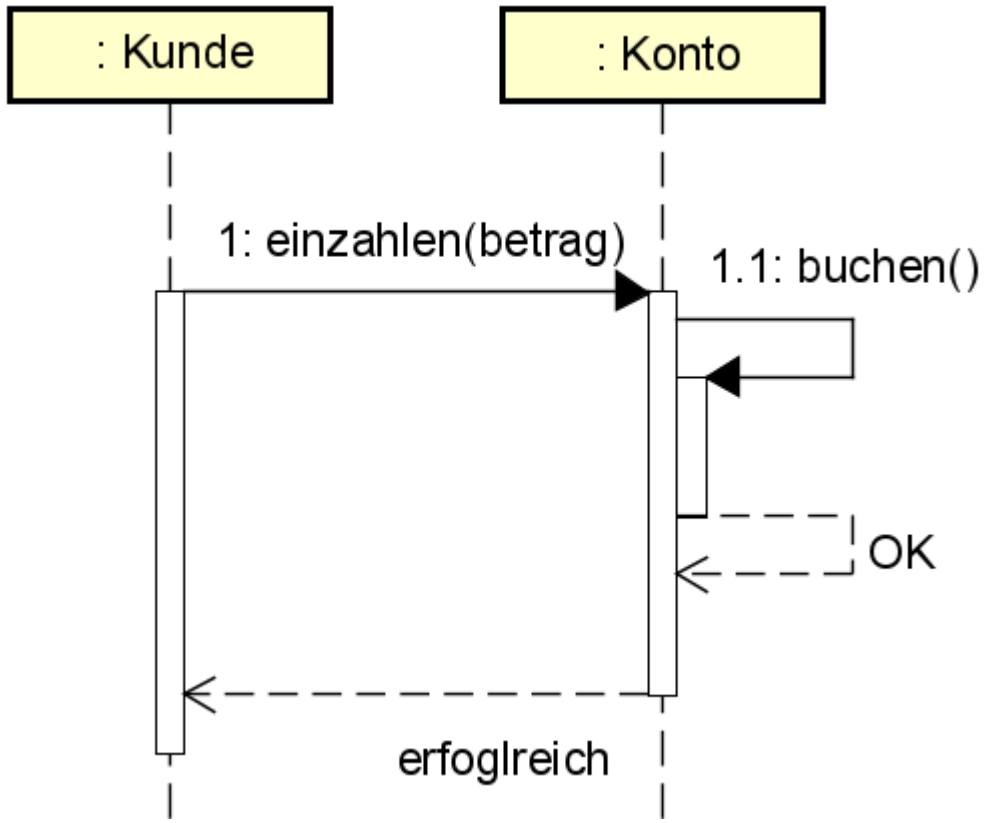
Objekt der Klasse Kunde ruft die Methode getKontostand() auf einem Objekt der Klasse Konto auf und bekommt den Kontostand zurück.

Ein Objekt ist nicht immer eine Instanz einer Klasse.



Übersicht

Beispiel



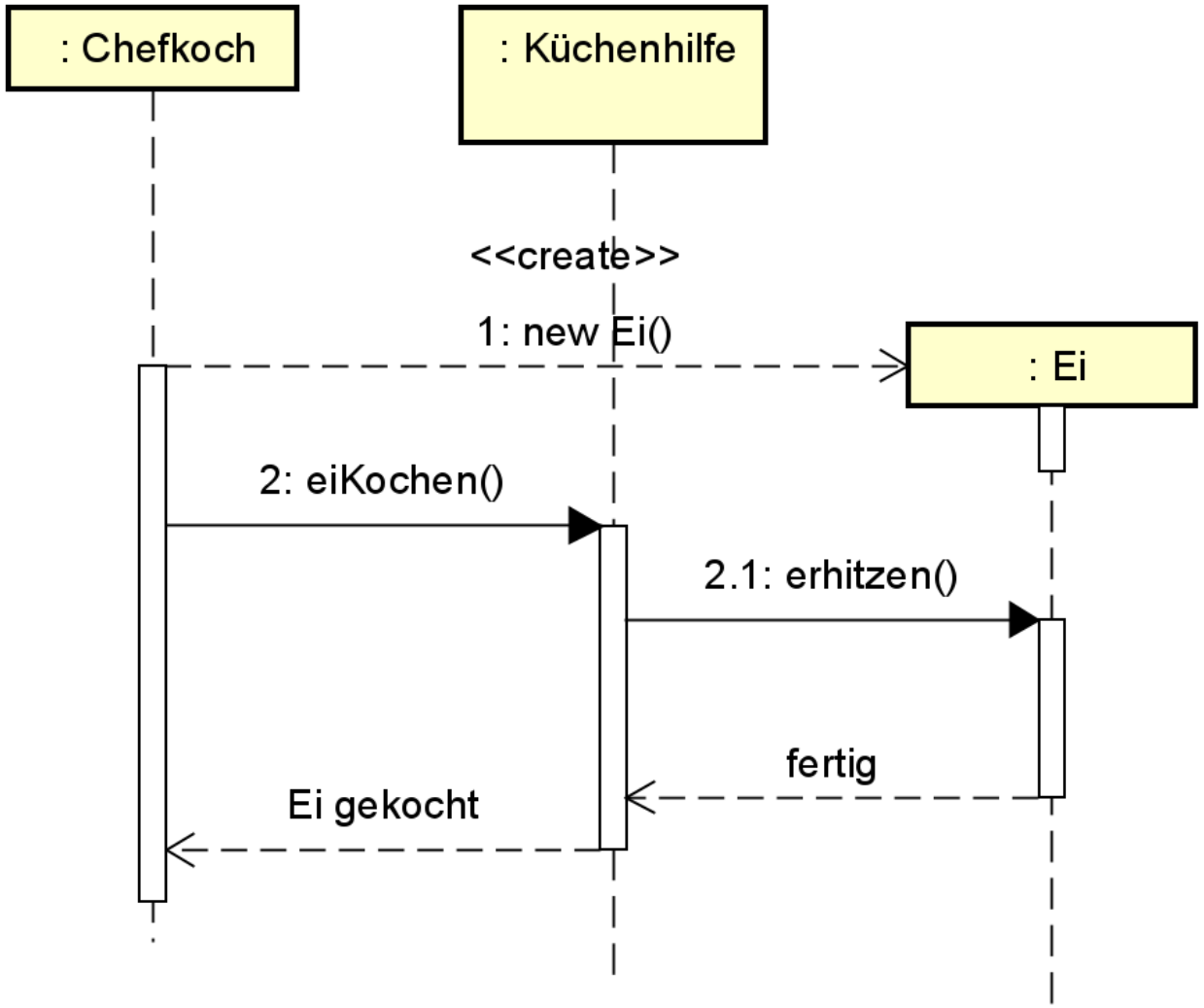
Ein Objekt der Klasse Kunde ruft die Methode Einzahlen mit einem Betrag auf einem Objekt der Klasse Konto auf. Das Konto bucht den Betrag auf sich selbst mit der Buchen()-Methode und gibt „OK“ zurück. Die Einzahlen()-Methode gibt „Erfolgreich“ an den Kunden zurück.

Übersicht

Beispiel

Der Chefkoch erzeugt ein Objekt von Ei und weist die Küchenhilfe an, das Ei zu kochen. Die Küchenhilfe erhitzt das Ei und gibt es an den Chefkoch zurück, sobald es erhitzt ist.

Während das Ei kocht, können Chefkoch und Küchenhilfe andere Aufgaben erledigen und müssen erst wieder aktiv werden, wenn das Ei fertig gekocht ist.



Syntax

Syntax

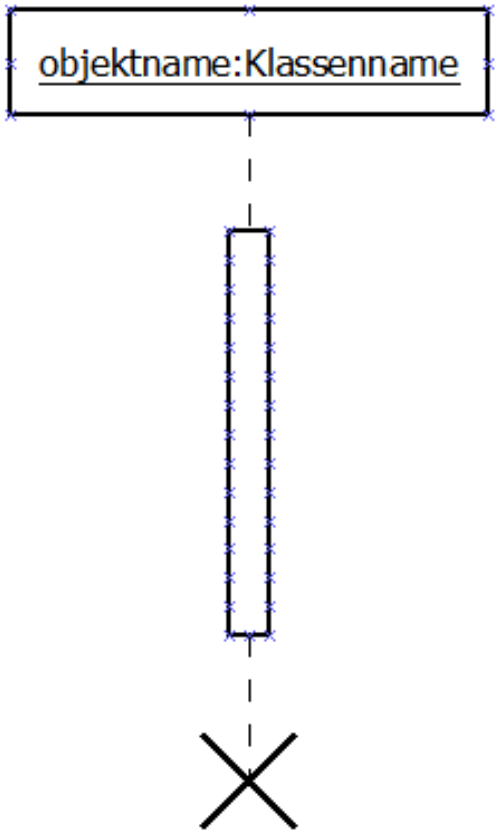
Klassen, Objekte, Lebenslinien und Ausführung

Objekte werden in Rechtecken dargestellt und haben unter sich eine Lebenslinie.

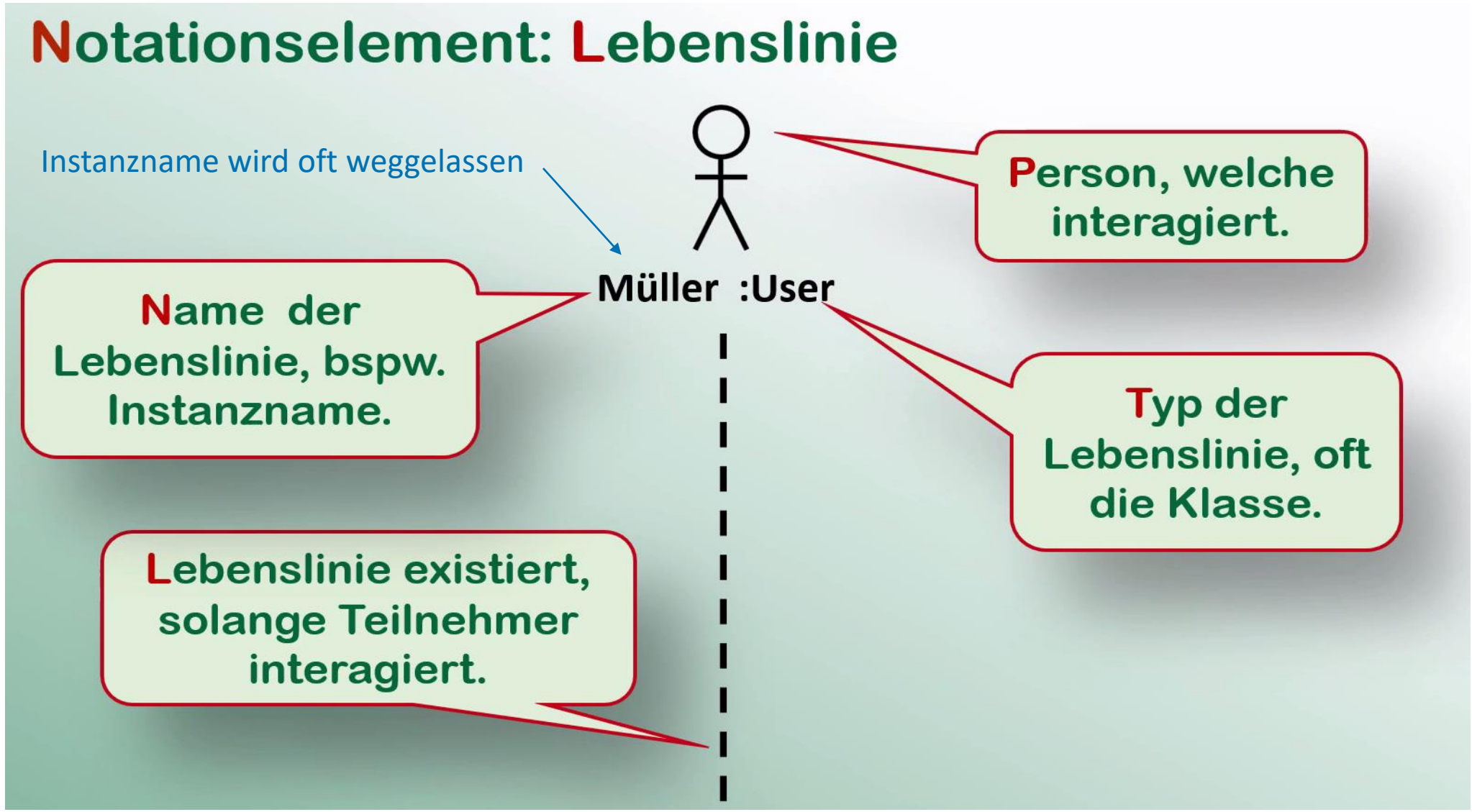
Die Lebenslinie ist eine unterbrochene Linie und zeigt, wie lange ein Objekt existiert.

Auf einer Lebenslinie ist die Ausführung. Die Ausführung gibt an, wann ein Objekt aktiv wird. Ein Objekt ist aktiv, wenn es Methoden aufruft oder Methoden von ihm aufgerufen werden.

Wird ein Objekt zerstört, wird dies durch ein X am Ende der Lebenslinie dargestellt. Eine Objektzerstörung wird nur modelliert, wenn auch tatsächlich eine Objektzerstörung stattfindet.



Notationselement: Lebenslinie



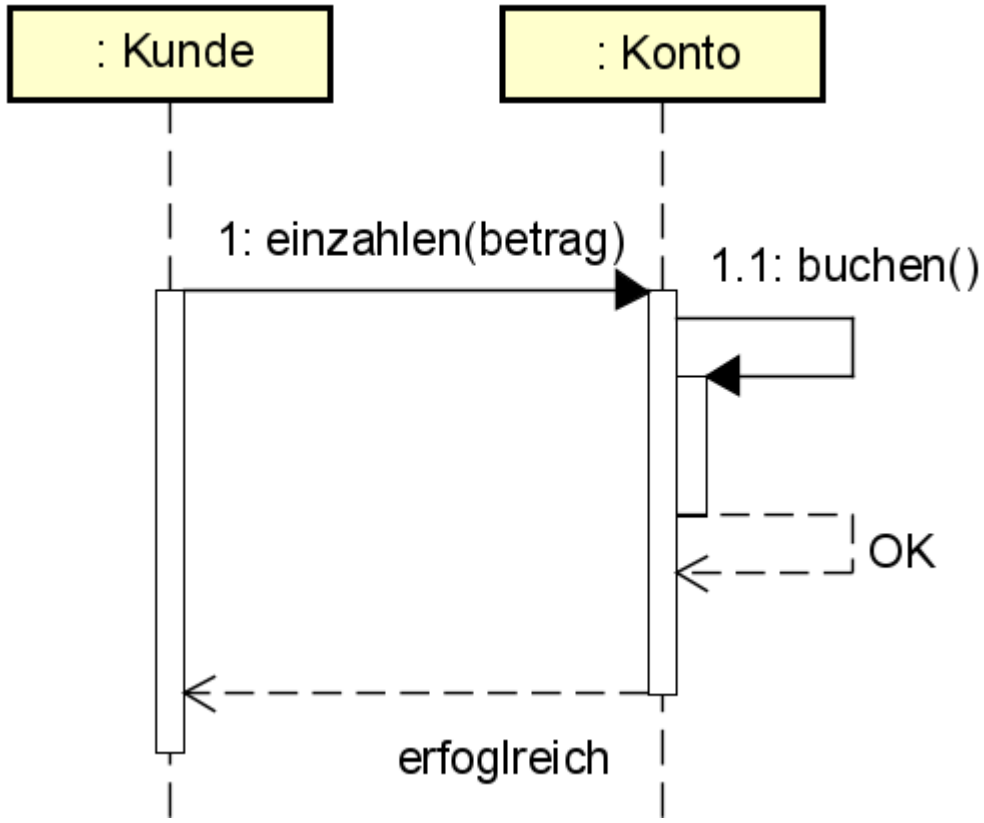
Syntax

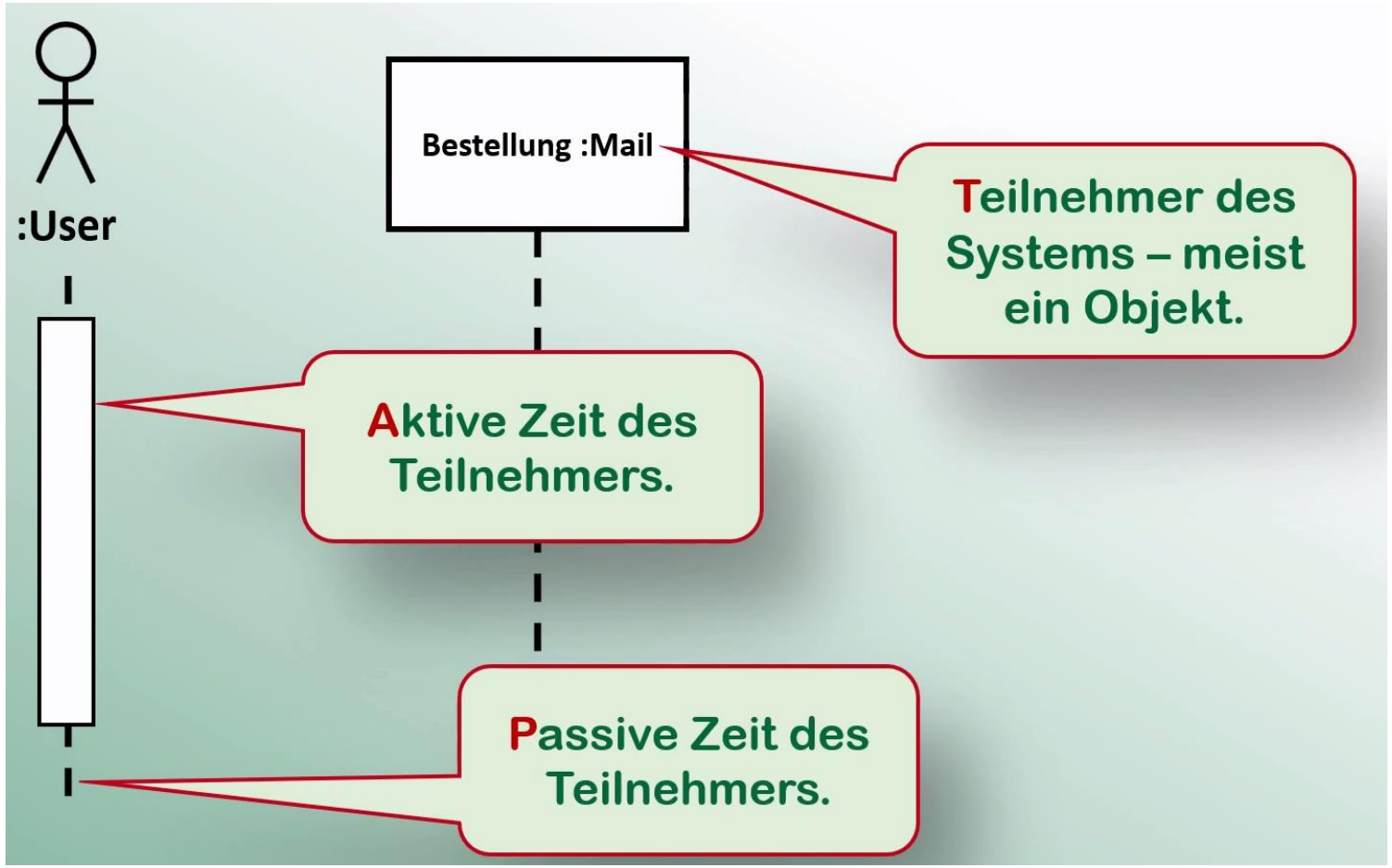
Nachrichten und Selbstaufruf

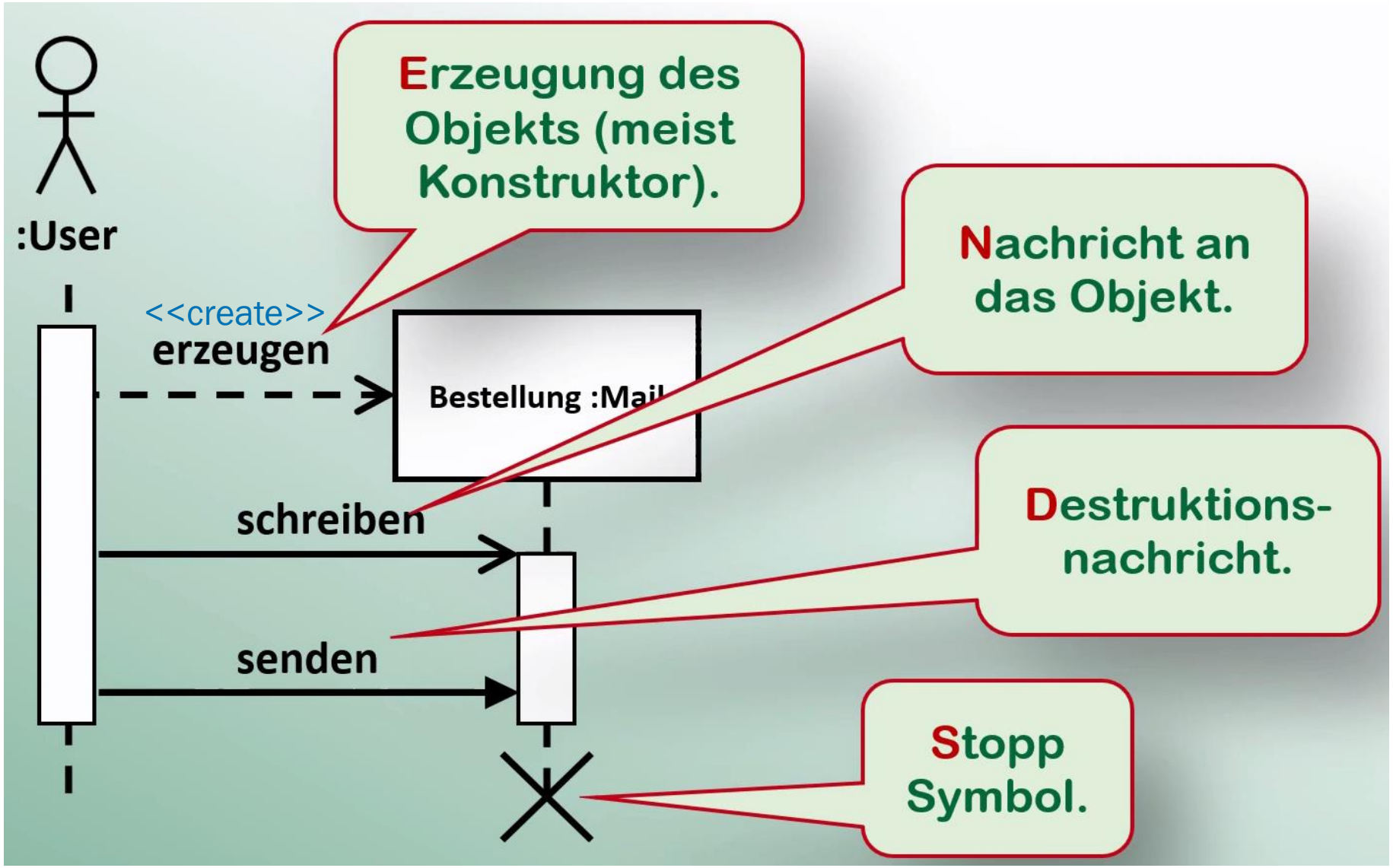
In einem Sequenzdiagramm kommunizieren Objekte miteinander. Diese Kommunikation stellt den zeitlichen Ablauf der Methodenaufrufe dar.
Der zeitliche Ablauf ist dabei immer von oben nach unten.

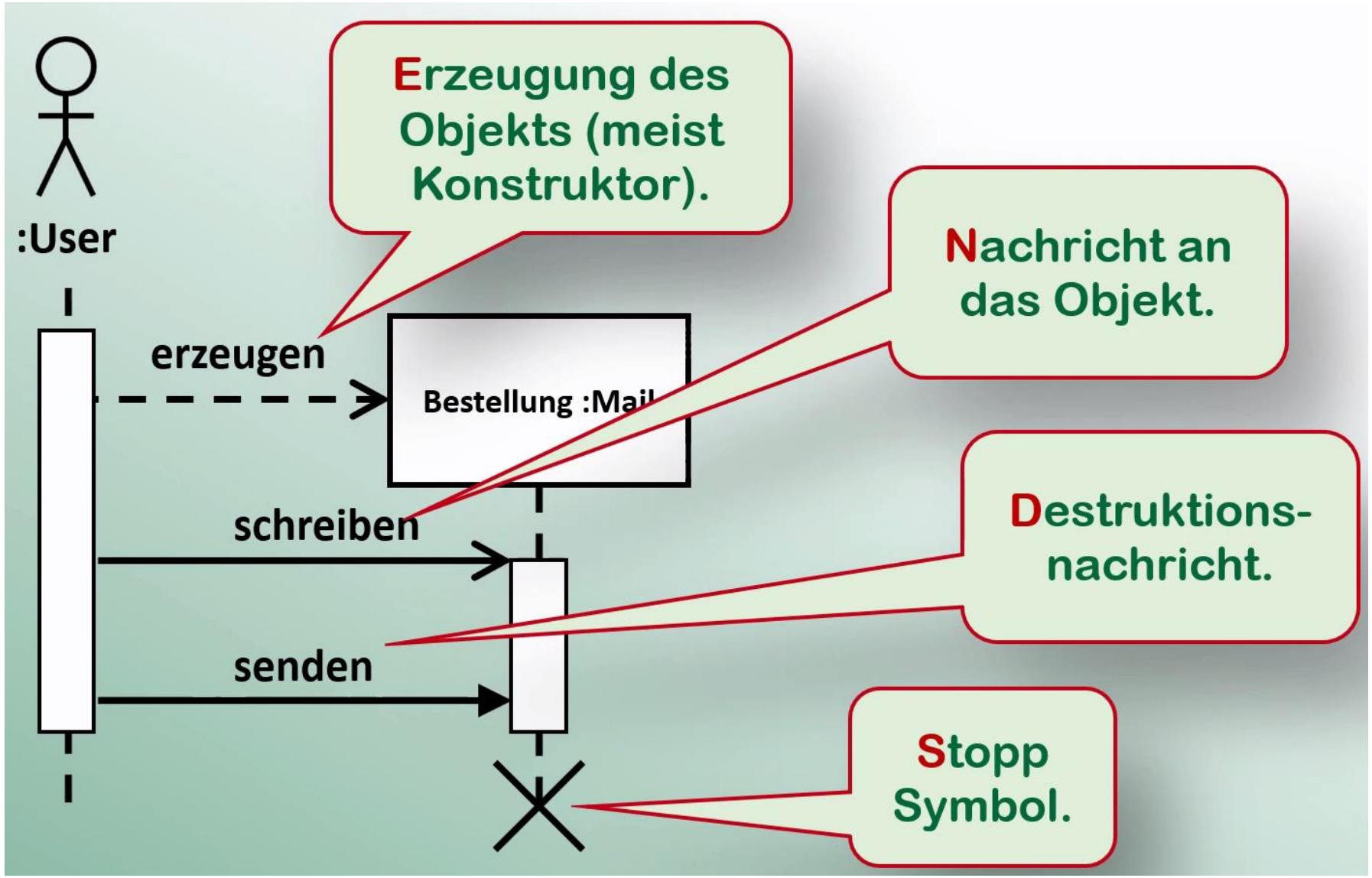
Objekte können Methoden von anderen Objekten aufrufen, aber auch Methoden von sich selbst. Der Selbstaufruf ist dabei mit einer erneuten (quasi verschachtelten) Aktivität des Objektes verbunden.

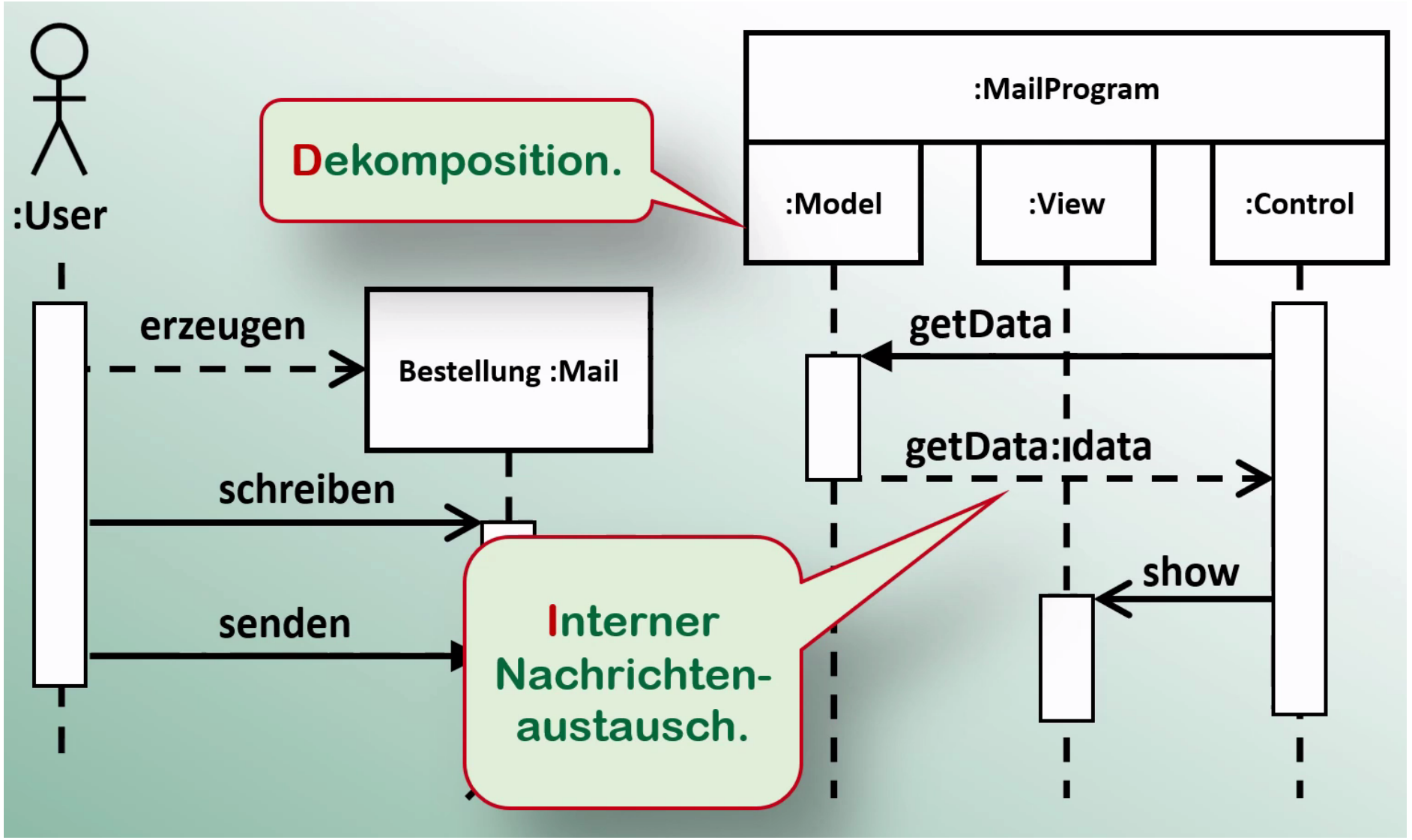
Ist die Verarbeitung eines Methodenaufrufes abgeschlossen, erfolgt in der Regel auch eine Antwort der Methode.




:User
Bestellung :Mail
Teilnehmer des Systems – meist ein Objekt.
Aktive Zeit des Teilnehmers.
Passive Zeit des Teilnehmers.


Erzeugung des Objekts (meist Konstruktor).
:User
<<create>>
erzeugen
Bestellung :Mail
Nachricht an das Objekt.
schreiben
Destruktions-nachricht.
senden
Stopp Symbol.


Erzeugung des Objekts (meist Konstruktor).
:User
erzeugen
Bestellung :Mail
Nachricht an das Objekt.
schreiben
Destruktions-nachricht.
senden
Stopp Symbol.


:User
:MailProgram
:Model
:View
:Control
Dekomposition.
erzeugen
Bestellung :Mail
getData
getData:idata
schreiben
show
senden
Interner
Nachrichten-
austausch.

Syntax

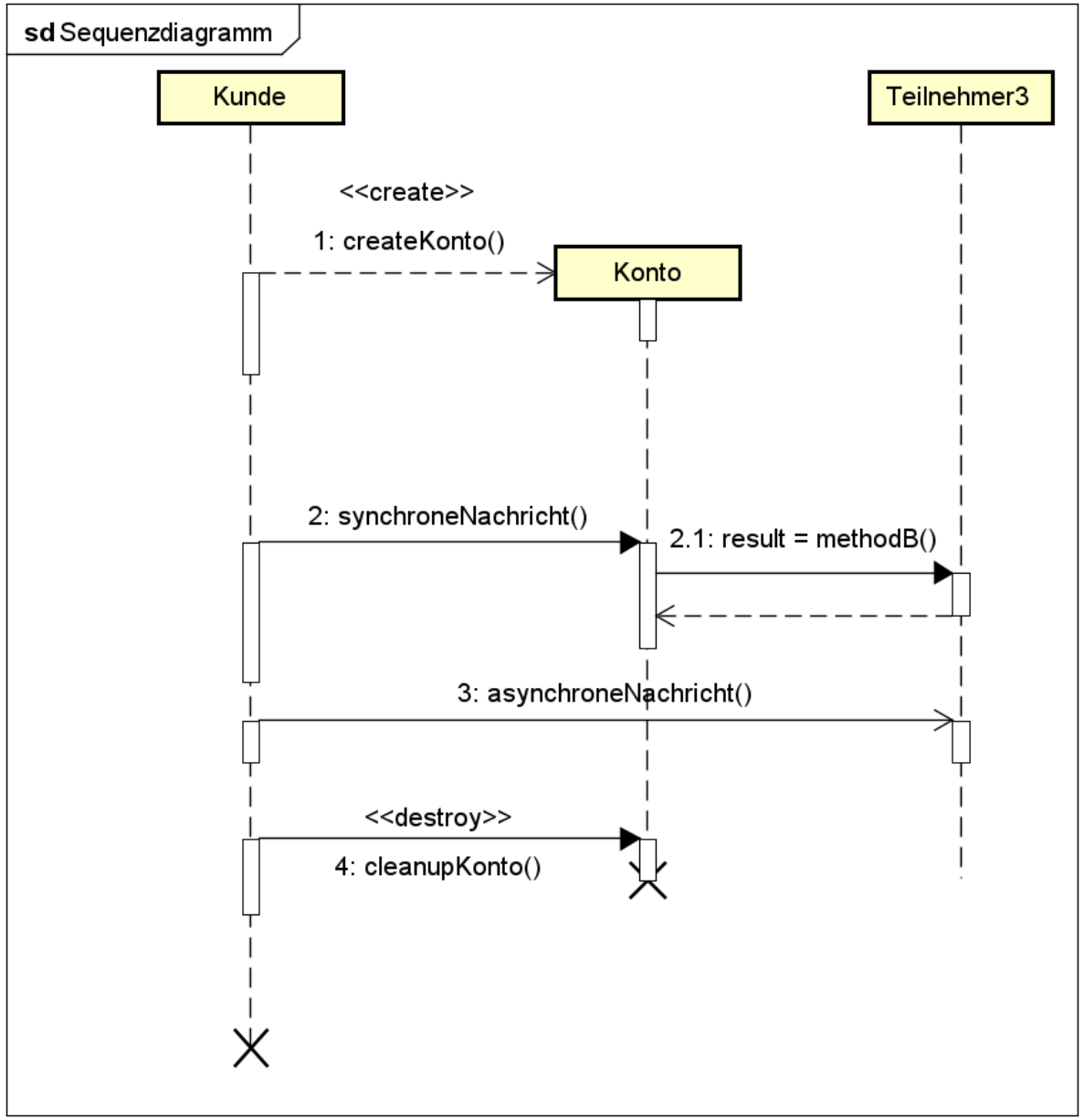
Nachrichten und Selbstaufruf

Es gibt unterschiedliche Arten von Nachrichten.

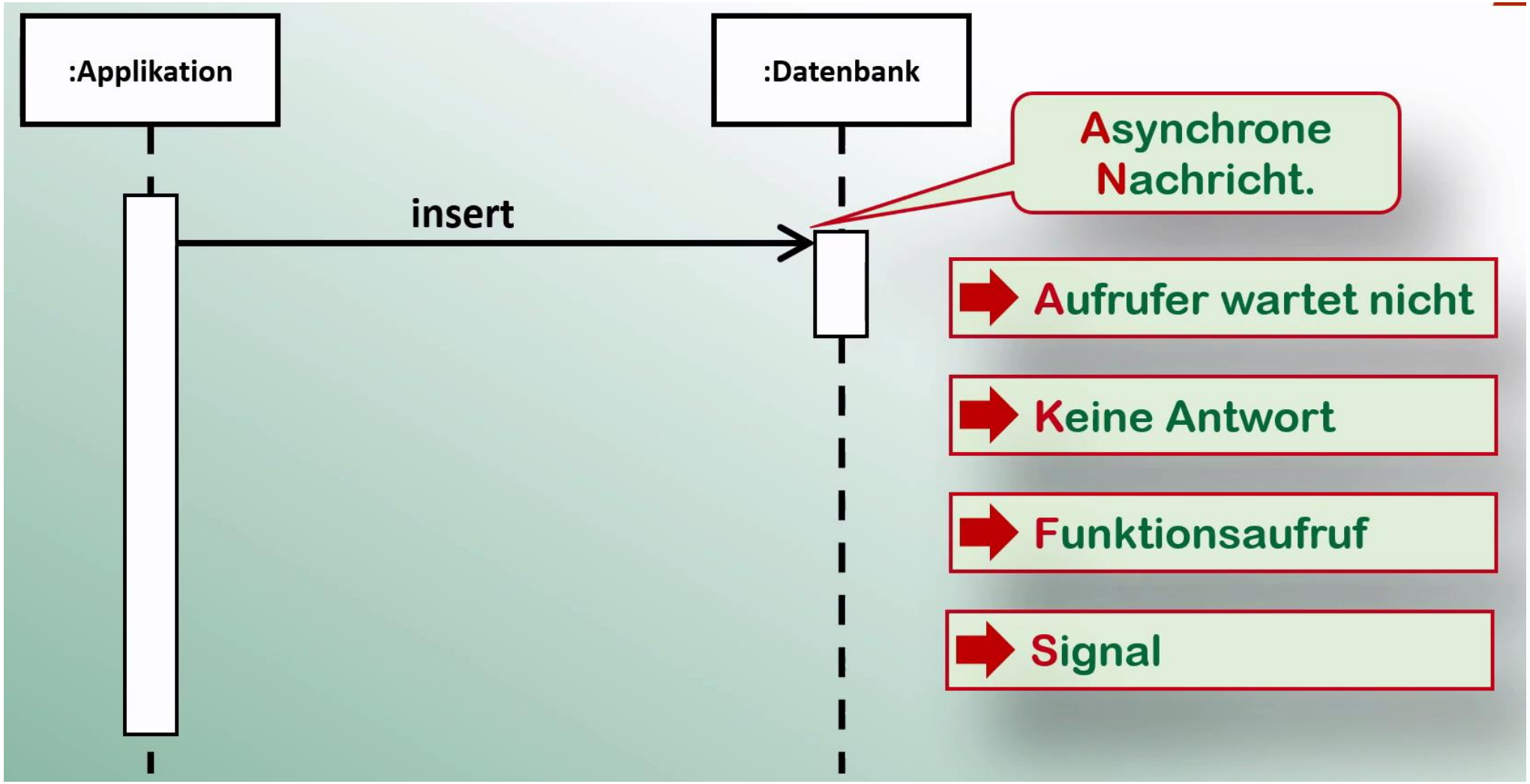
<<create>> erzeugt ein Objekt. Es findet eine Instanziierung einer Klasse statt.

Bei einer synchronen Nachricht muss der Sender auf die Antwort warten.

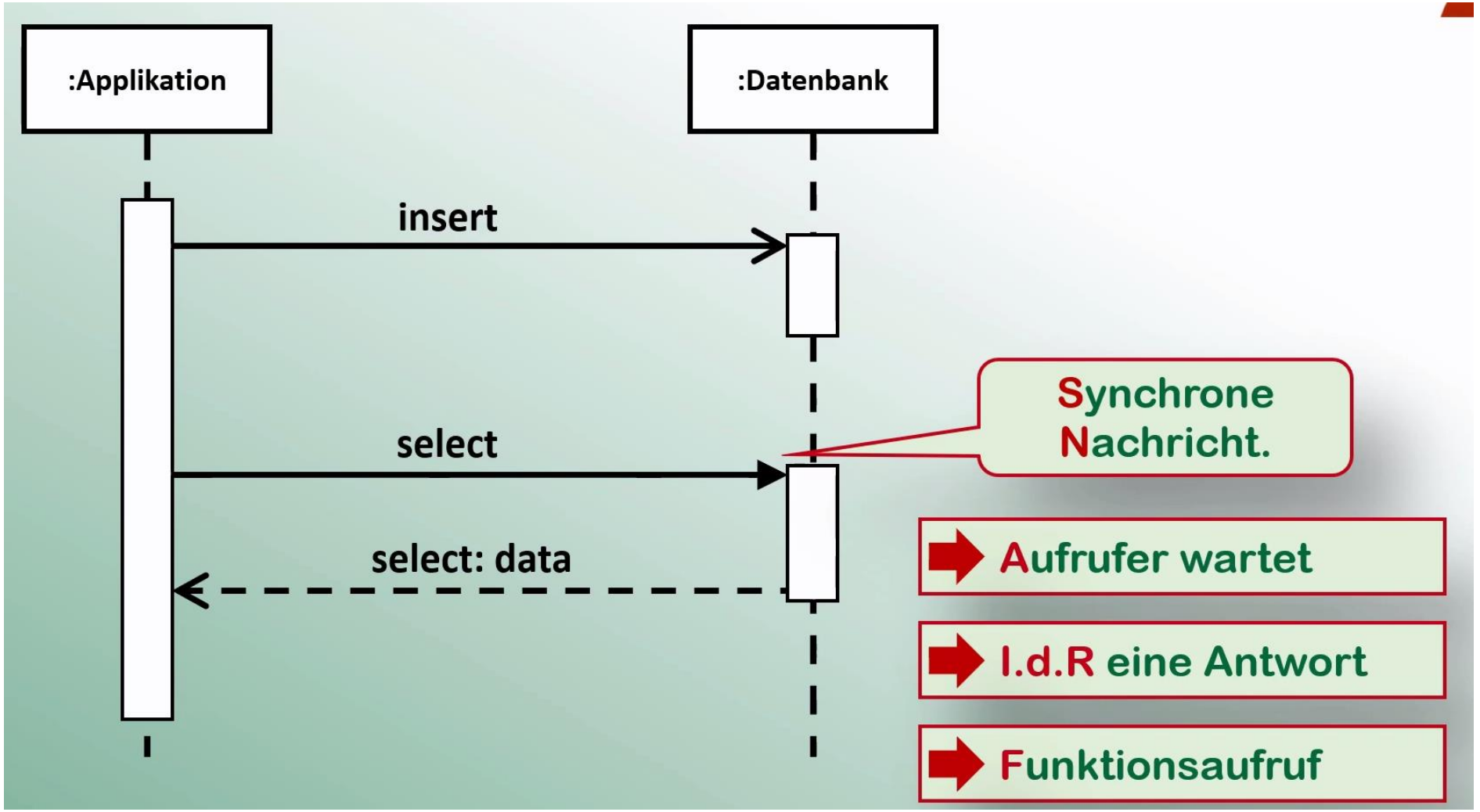
Bei einer asynchronen Nachricht muss der Sender nicht auf die Antwort warten.



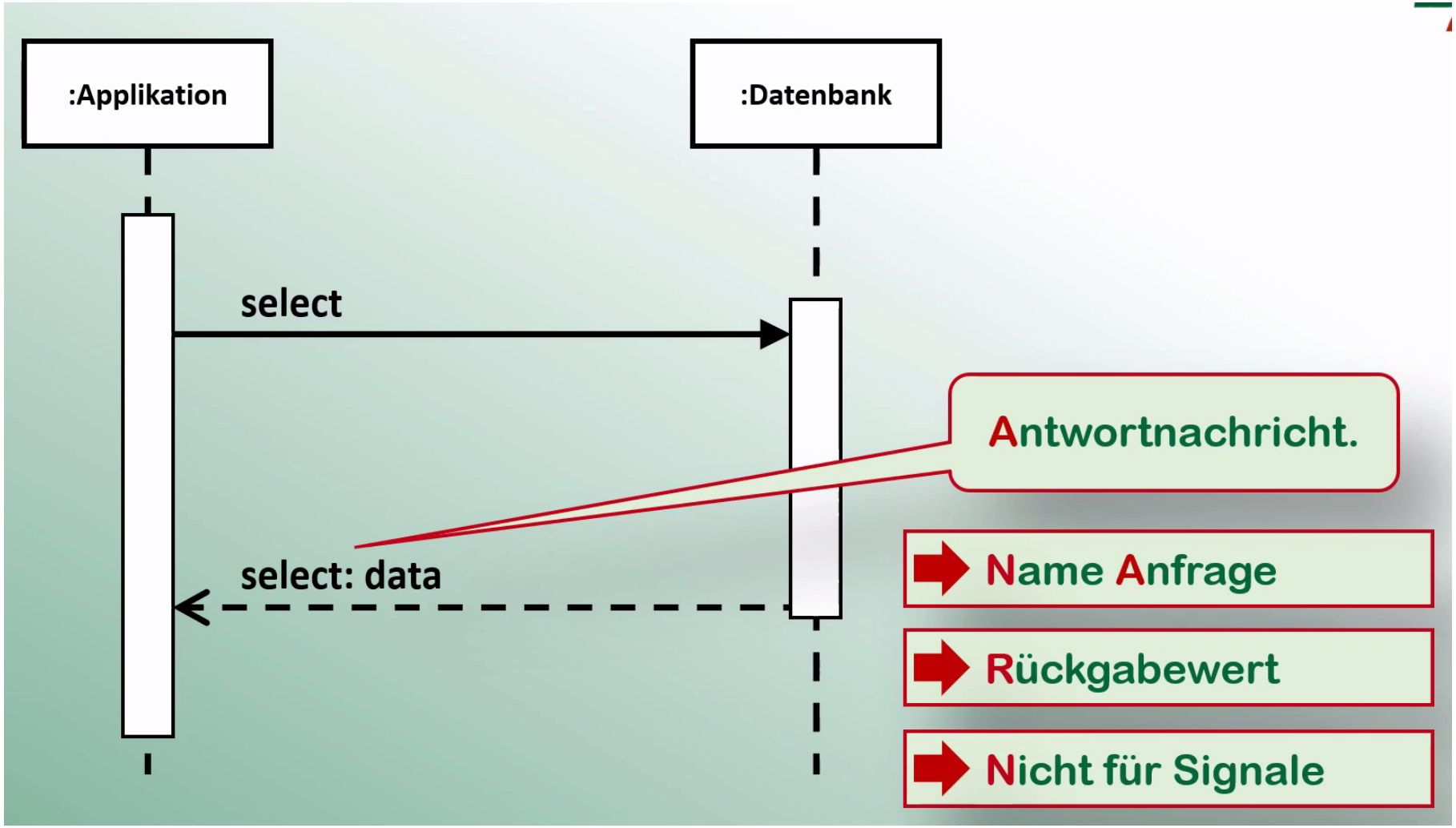
Asynchrone Nachricht

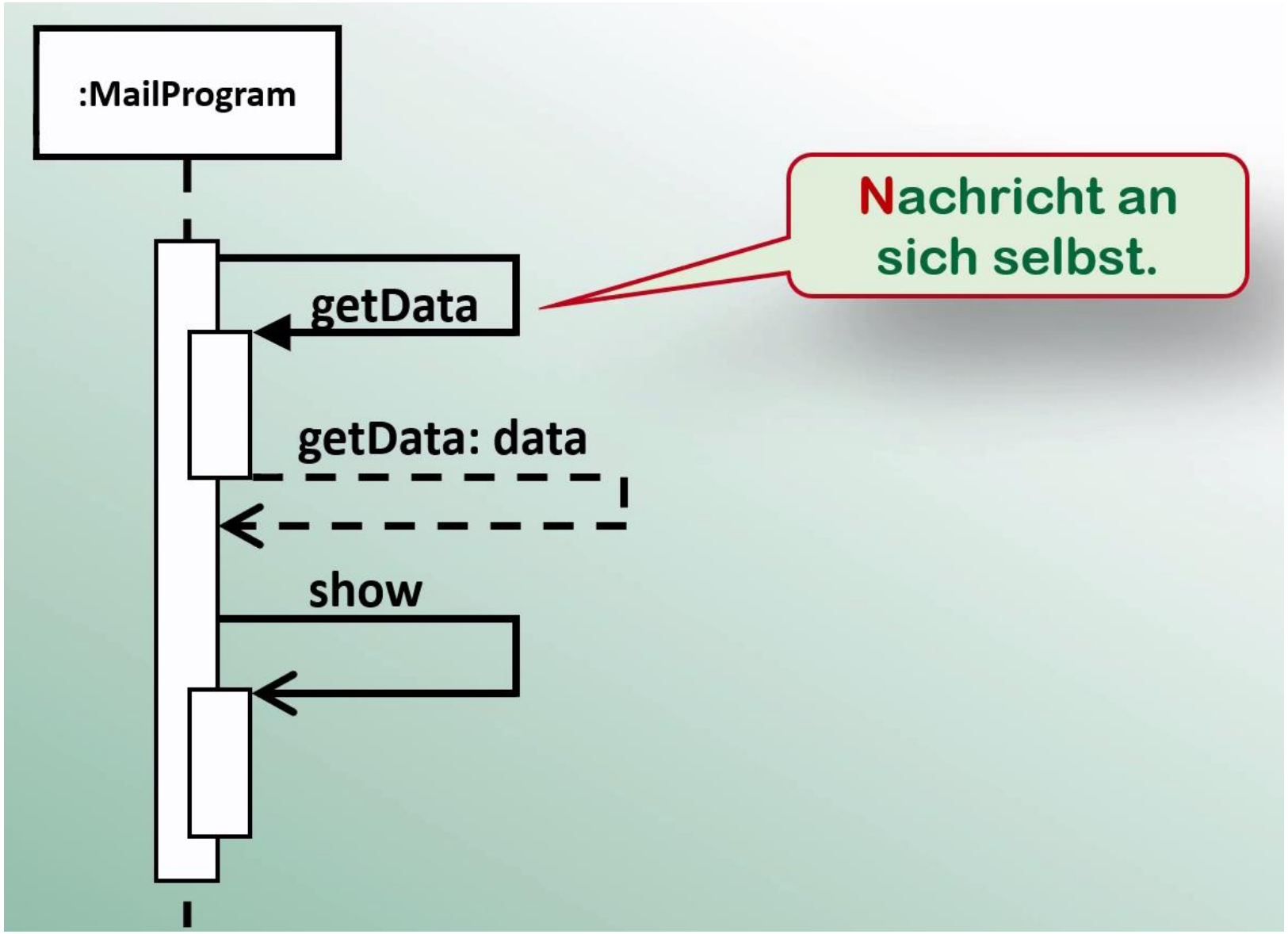


Synchrone Nachricht

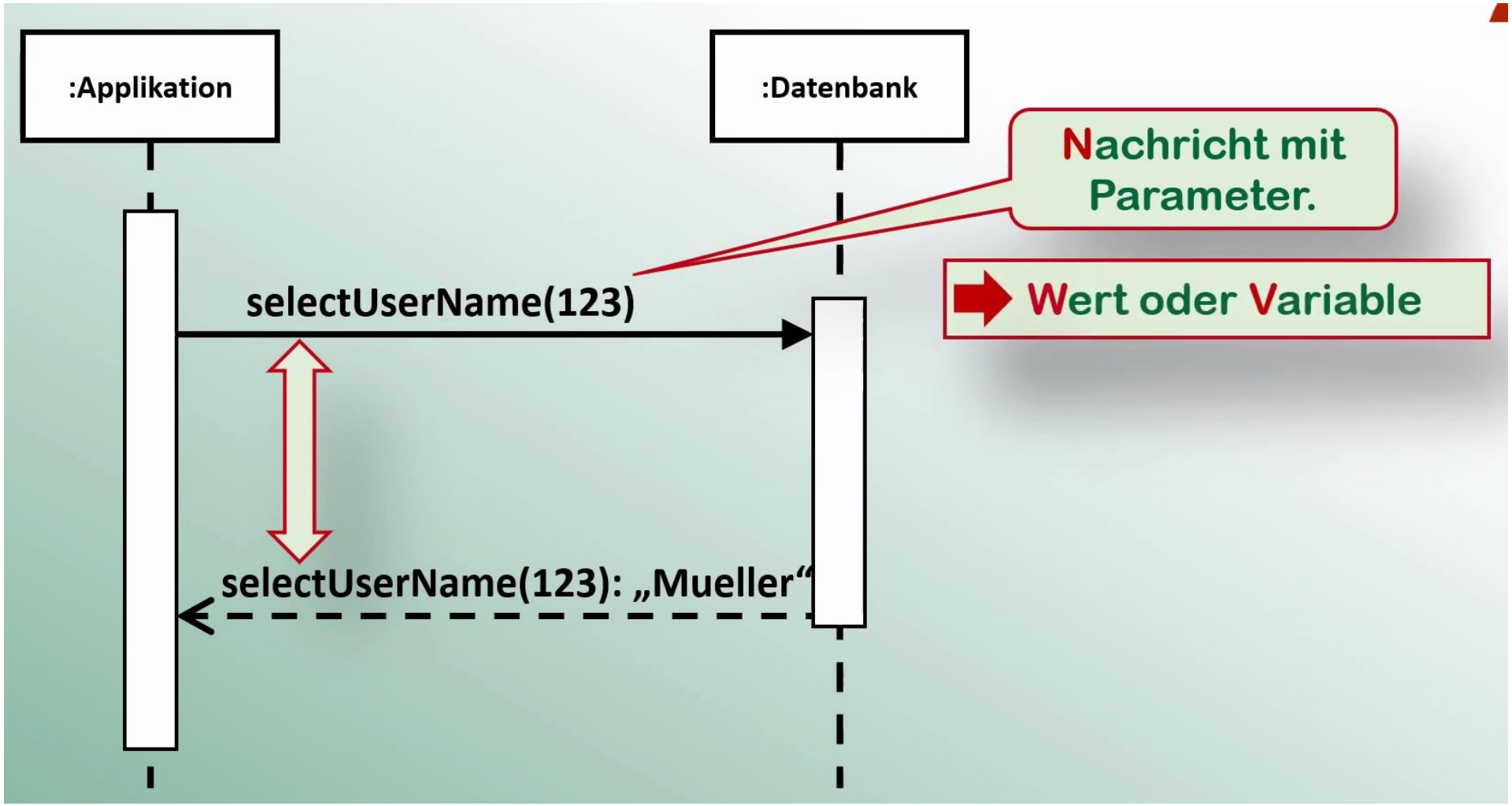


Antwort-Nachricht

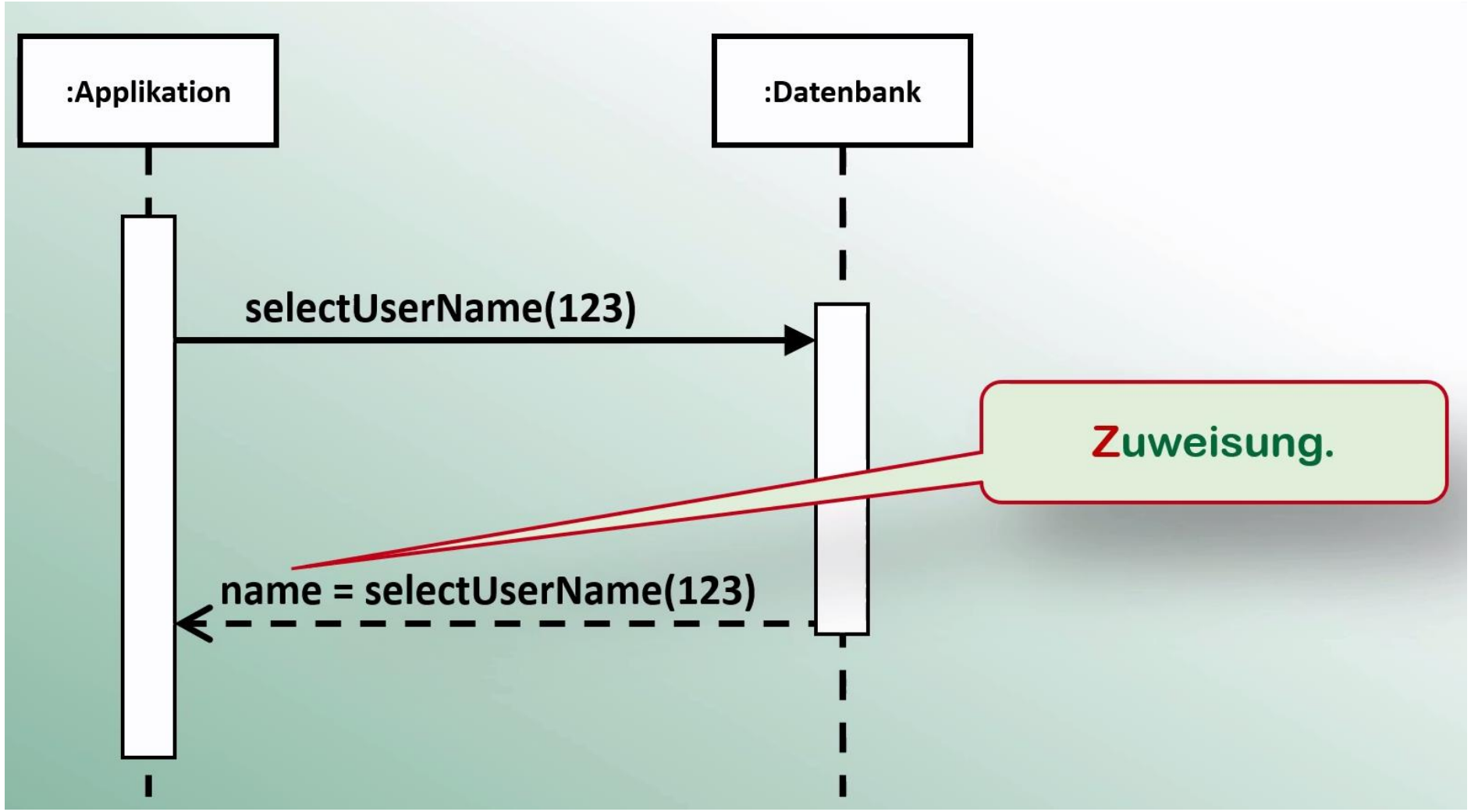



:MailProgram
Nachricht an sich selbst.
getData
getData: data
show

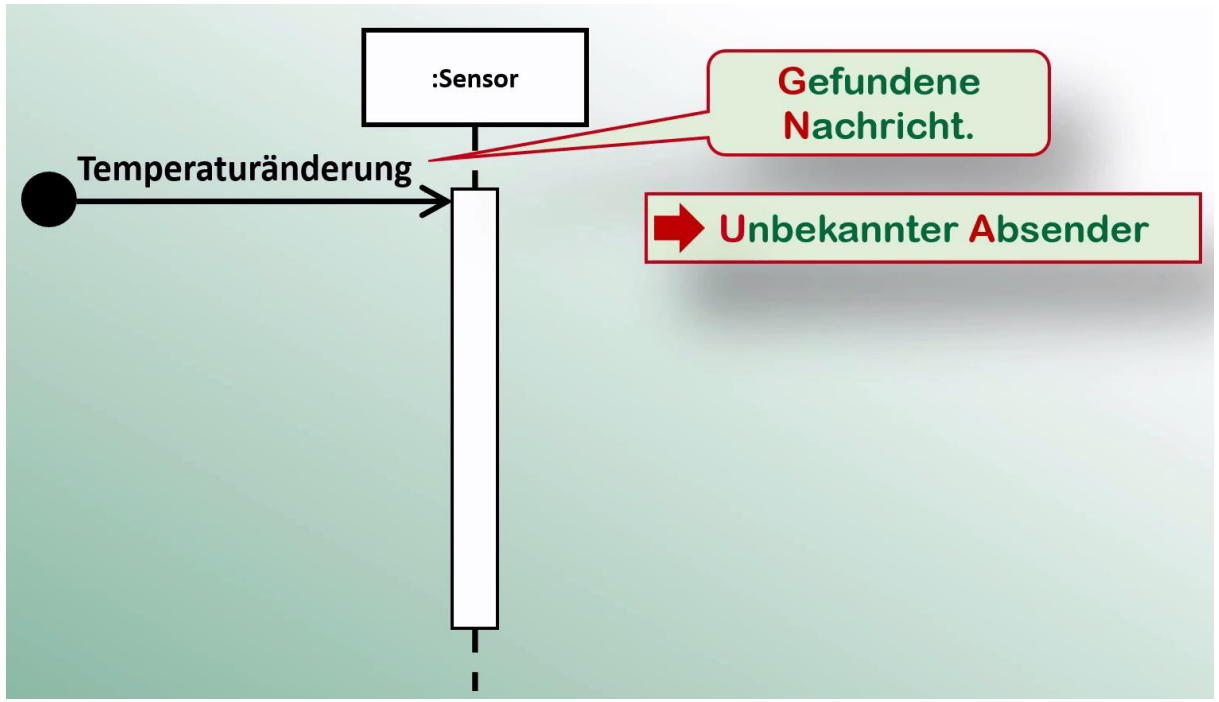
Nachricht mit Parameter

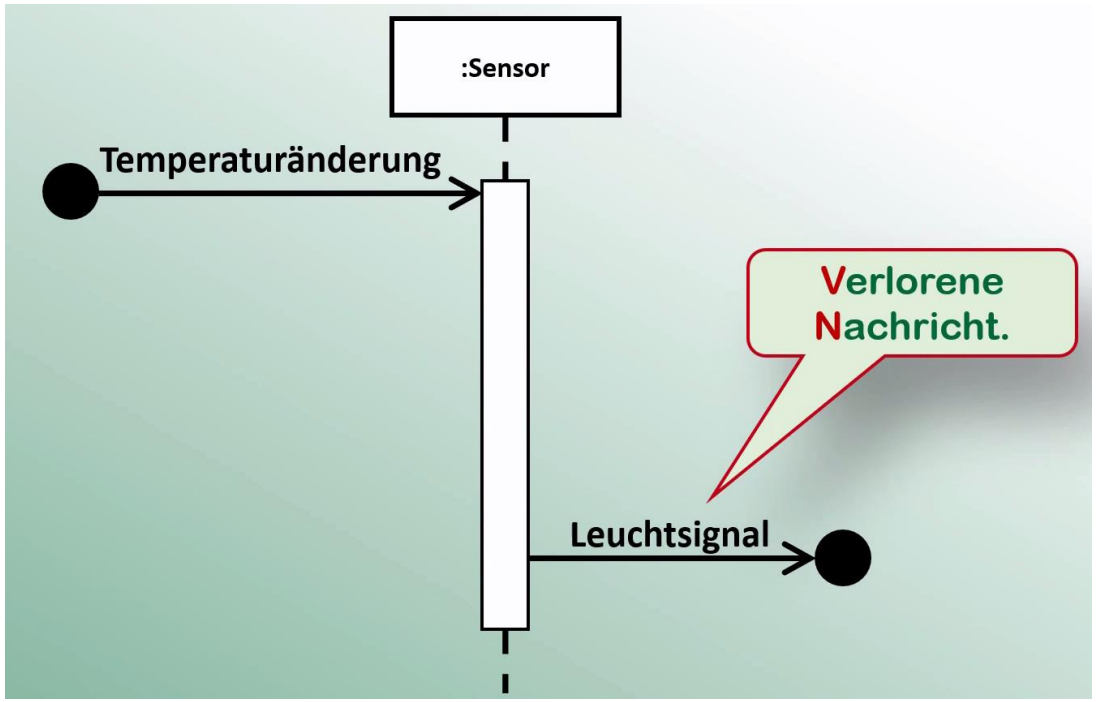


Zuweisung

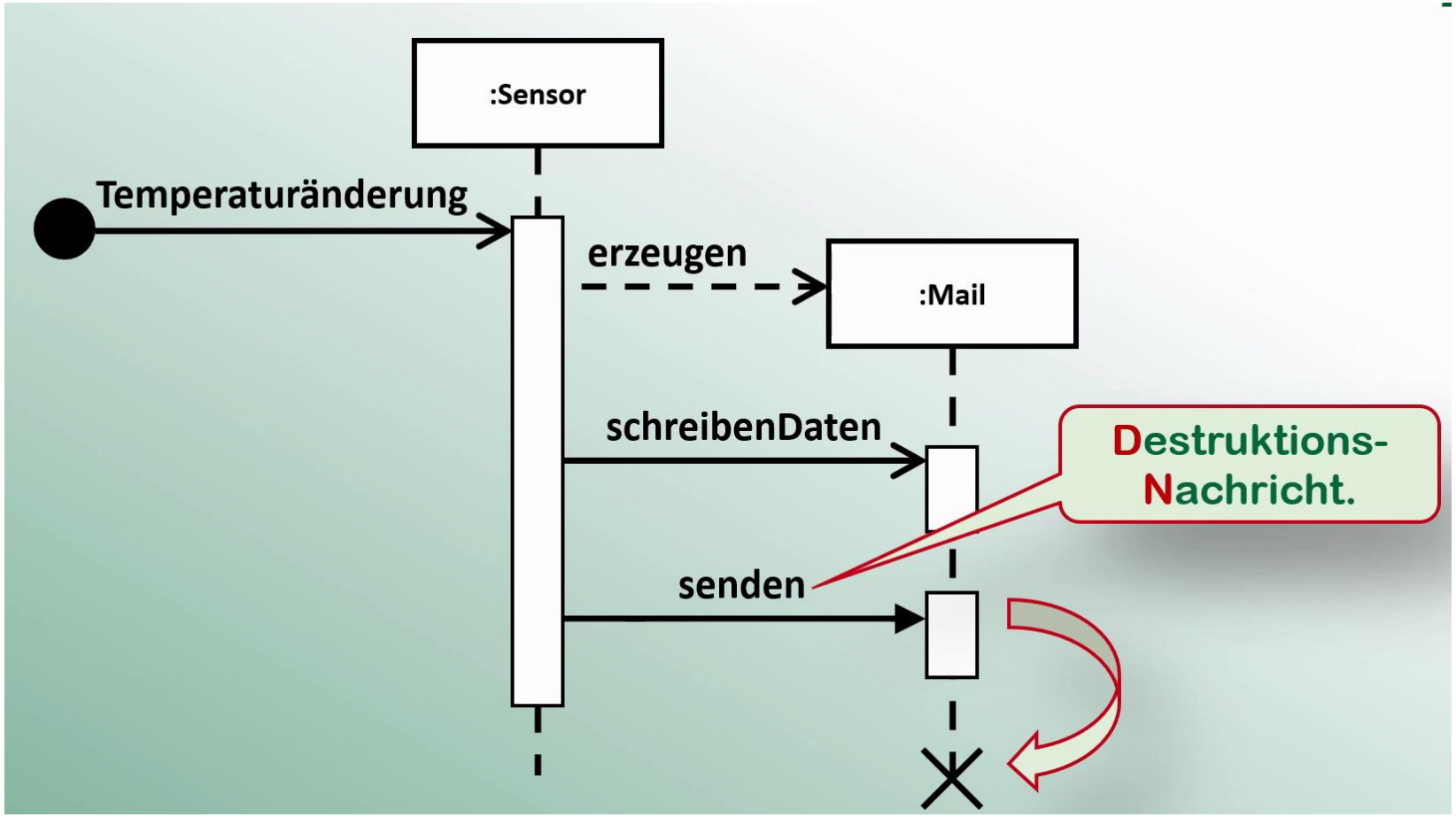

:Applikation
:Datenbank
selectUserName(123)
Zuweisung.
name = selectUserName(123)

Gefundene Nachricht / Verlorene Nachricht





Destruktions-Nachricht



...

Syntax

Verzweigung und Schleifen

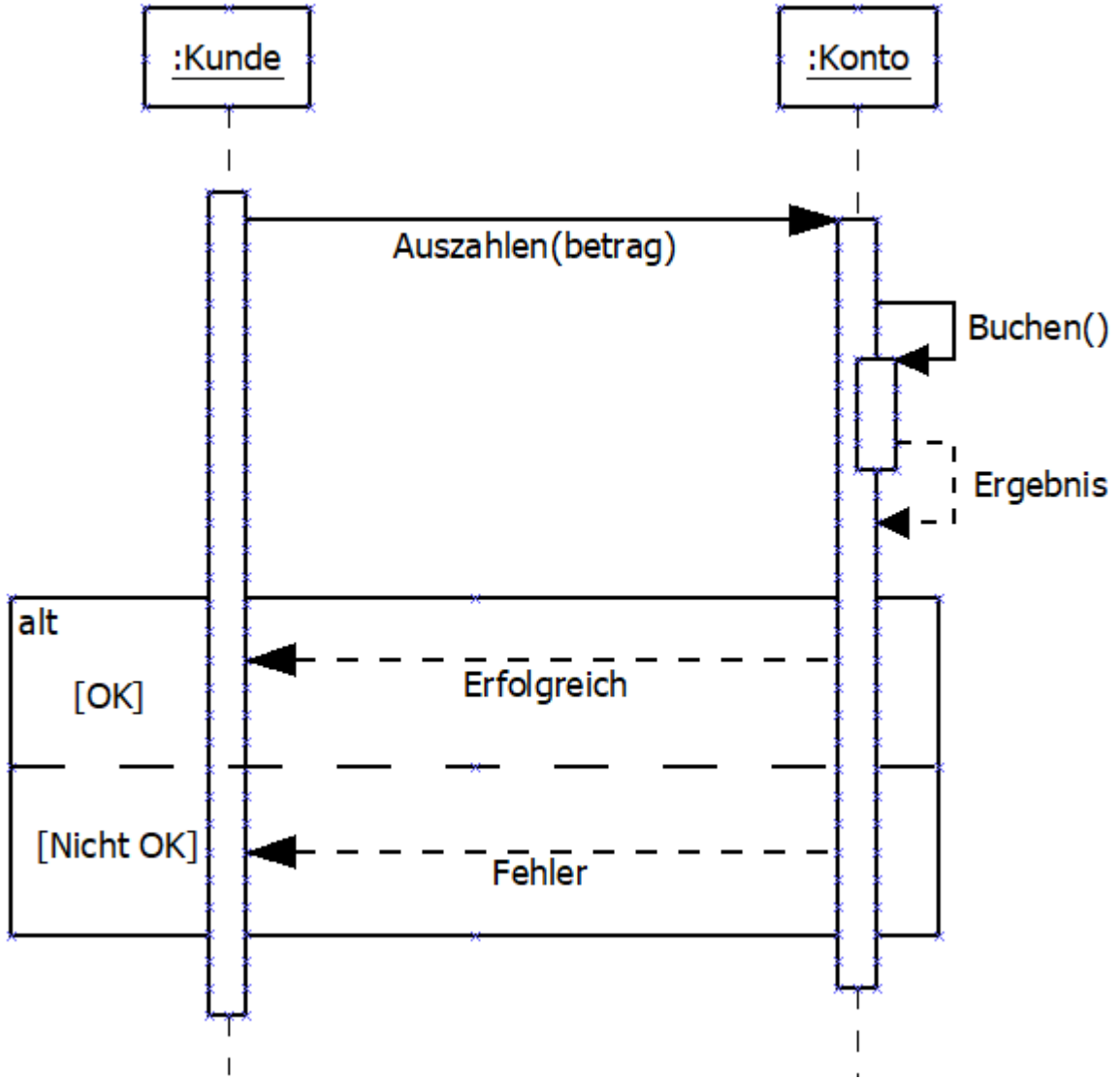
In einem Sequenz-Diagramm können Verzweigungen und Schleifen modelliert werden.

alt: steht für Alternative und stellt If-Else dar

opt: steht für Optional und stellt If dar

loop: modelliert eine Schleife

Bedingungen werden in [ ] geschrieben



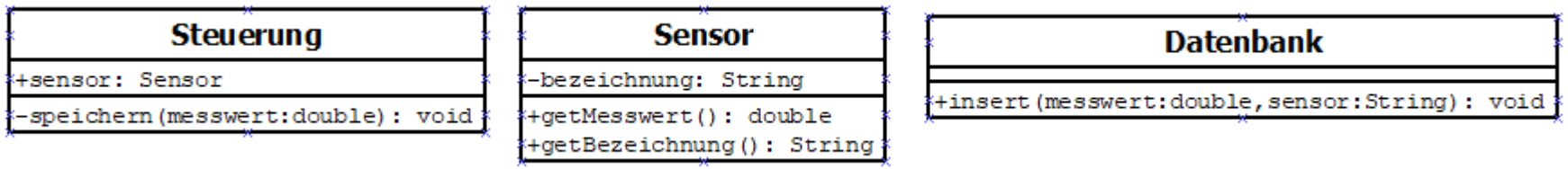
Übungsaufgabe

Übungsaufgabe

Erstellen Sie zum folgenden Programmablauf ein Sequenz-Diagramm:

Die Steuerung fragt jede Minute vom Sensor über getMesswert() den aktuellen Messwert ab.
Anschließend ruft die Steuerung speichern() auf.
speichern() fragt vom Sensor die Bezeichnung ab und ruft auf der Datenbank die insert()-Methode auf.
Am Ende kehrt der Ablauf zum Aufrufer zurück.

Folgendes Klassendiagramm liegt Ihnen vor:



Vielen Dank für Ihre Aufmerksamkeit!