Kurs

Datenbankgrundlagen und Modellierung

Sebastian Maneth, Universität Bremen
maneth@uni-bremen.de

Sommersemester 2023

26.6.2023
Vorlesung 9: Sequenzdiagramme (Sequence Diagrams)

Kurs **Datenbankgrundlagen und Modellierung**

17.4. Vorlesung 1 — Intro
24.4. V2 — ER, SQL
1.5. keine Vorlesung
4.5. Fragestunden
8.5. V3 — SQL
10.5. Ü1
11.5. Fragestunden
15.5. V4 — SQL
17.5. Ü2
22.5. V5 — SQL & funct. dependencies
24.5. Ü3
~~29.5. V6 — funct. dependencies~~
31.5. Ü4
5.6. V6 — normal forms
7.6. Ü5

12.6. V7 — modelling Intro & class diagrams
14.6. Ü6
19.6. V8 — state charts
21.6. Ü7
26.6. V9 — sequence diagrams
28.6. Ü8
3.7. V10 — Klausurvorbereitung
5.7. Ü9 — Allgemeine Fragestunden

Kurs **Datenbankgrundlagen und Modellierung**

17.4. Vorlesung 1 — Intro
24.4. V2 — ER, SQL
1.5. keine Vorlesung
4.5. Fragestunden
8.5. V3 — SQL
10.5. Ü1
11.5. Fragestunden
15.5. V4 — SQL
17.5. Ü2
22.5. V5 — SQL & funct. dependencies
24.5. Ü3
~~29.5. V6 — funct. dependencies~~
31.5. Ü4
5.6. V6 — normal forms
7.6. Ü5

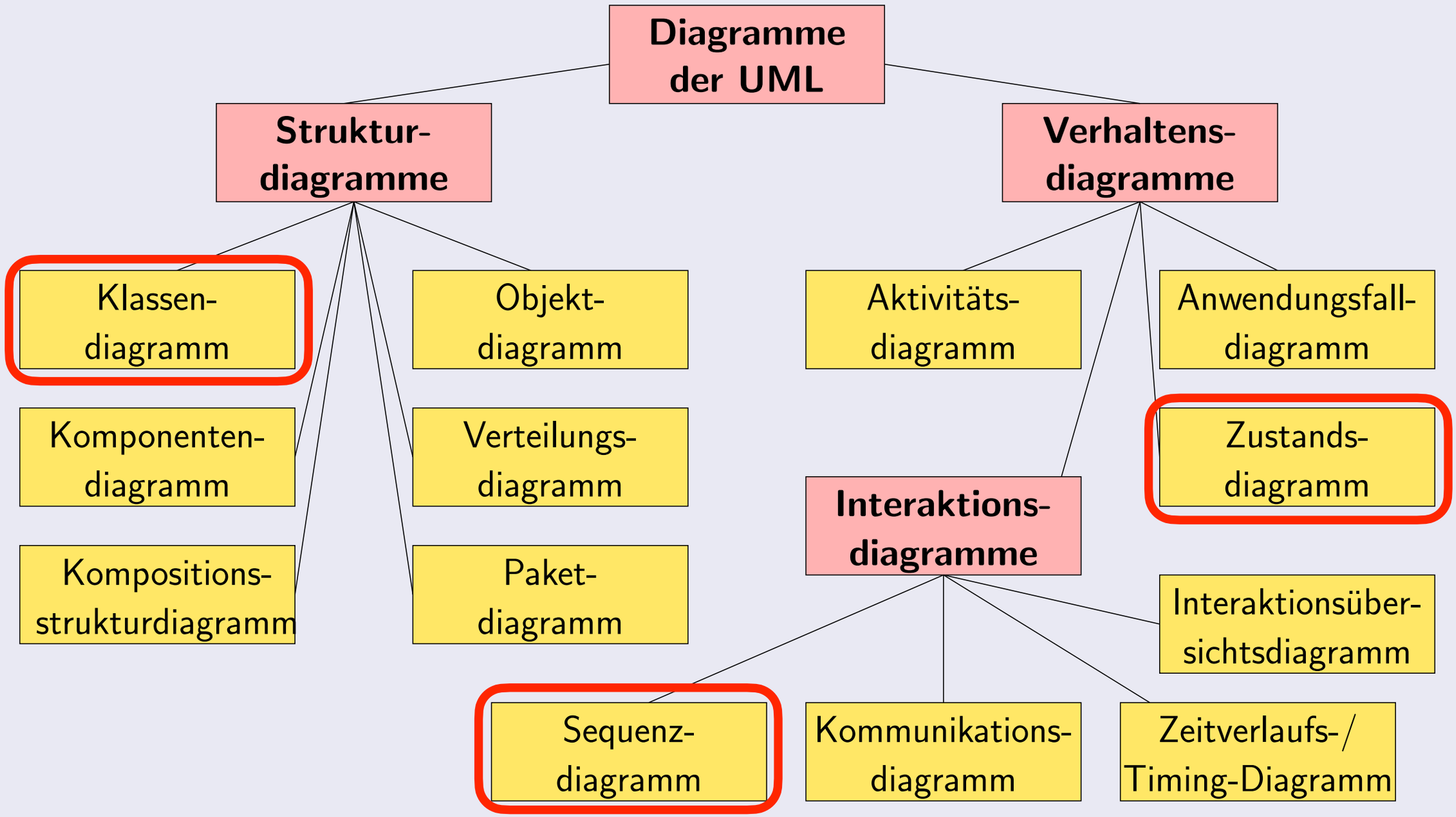
12.6. V7 — modelling Intro & class diagrams
14.6. Ü6
19.6. V8 — state charts
21.6. Ü7
26.6. V9 — sequence diagrams
28.6. Ü8
3.7. V10 — Klausurvorbereitung
5.7. Ü9 — Allgemeine Fragestunden

Klausur:

Dienstag 26.9., 13:30 — 15:00 Uhr
HS 1010 & 2010

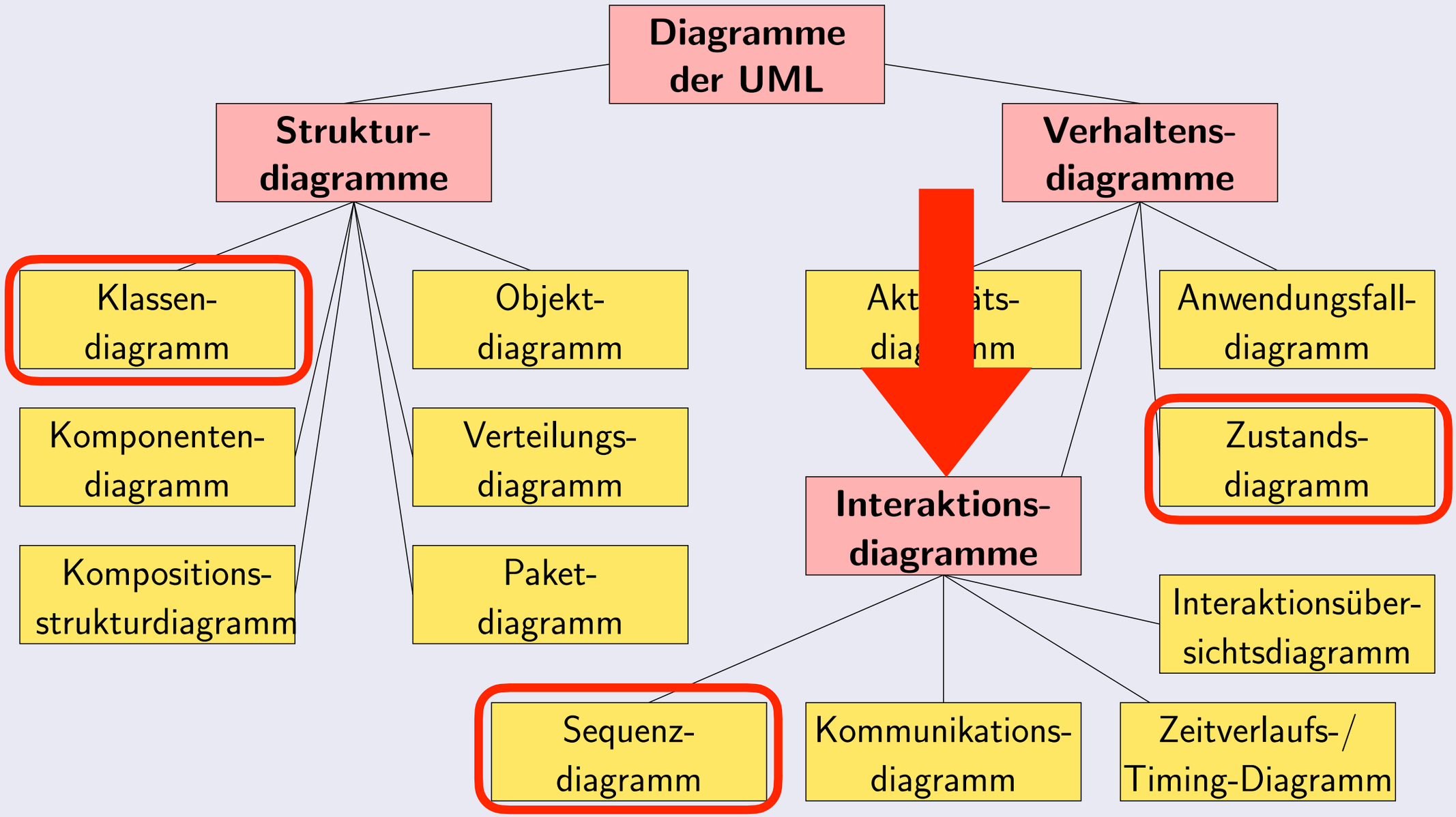
UML: Einführung

UML-Diagramme



UML: Einführung

UML-Diagramme



Sequenzdiagramme

Sequenzdiagramme (sequence diagrams) sind die bekanntesten Vertreter von Interaktionsdiagrammen in UML.

Sie dienen dazu, um Kommunikation und Interaktion zwischen mehreren Kommunikationspartnern zu modellieren und beruhen auf dem Basiskonzept der Interaktion:

Sequenzdiagramme

Sequenzdiagramme (sequence diagrams) sind die bekanntesten Vertreter von Interaktionsdiagrammen in UML.

Sie dienen dazu, um Kommunikation und Interaktion zwischen mehreren Kommunikationspartnern zu modellieren und beruhen auf dem Basiskonzept der Interaktion:

Interaktion

Eine Interaktion ist das Zusammenspiel von mehreren Kommunikationspartnern.

Typische Beispiele: Versenden von Nachrichten, Datenaustausch, Methodenaufruf

Sequenzdiagramme

Sequenzdiagramme (sequence diagrams) sind die bekanntesten Vertreter von Interaktionsdiagrammen in UML.

Sie dienen dazu, um Kommunikation und Interaktion zwischen mehreren Kommunikationspartnern zu modellieren und beruhen auf dem Basiskonzept der Interaktion:

z.B.:
— ein Drucker
— ein Display
— ein Controller
etc.

Interaktion

Eine Interaktion ist das Zusammenspiel von mehreren Kommunikationspartnern.

Typische Beispiele: Versenden von Nachrichten, Datenaustausch, Methodenaufruf

Sequenzdiagramme

Sequenzdiagramme waren bereits vor Aufnahme in die UML unter dem Namen message sequence charts bekannt.

Im Gegensatz zu Aktivitätsdiagrammen oder Zustandsdiagrammen beschreiben sie im allgemeinen nicht alle Abläufe eines Systems, sondern nur einen oder mehrere mögliche Abläufe.

UML und Objekt-Orientierung

Geschichte der Objekt-Orientierung

- Entwicklung von objekt-orientierten Programmiersprachen:
  - 60er Jahre: Simula (zur Beschreibung und Simulation von komplexen Mensch-Maschine-Interaktionen)
  - 80er Jahre: C++
  - 90er Jahre: Java
- Verbreitung von objekt-orientierten Entwurfsmethoden:
  - 70er Jahre: Entity-Relationship-Modell
  - 90er Jahre: Vorläufer von UML:
    - Jacobson → OOSE (Object-Oriented Software Engineering), 1992
    - Rumbaugh → OMT (Object Modeling Technique) 1991
    - Booch Method 1992
  - Seit 1997: UML
  - Seit 2005: UML 2.0

* Latest Version: UML 2.5.1 2017

Z.120 : Message Sequence Cha

https://www.itu.int/rec/T-REC-Z.120

International Telecommunication Union

Français Español

Z.120 : Message Sequence Chart (MSC)

Recommendation Z.120

In force components		
Number	**Title**	**Status**
Z.120 (02/11)	Message Sequence Chart (MSC)	In force
Z.120 Annex B (04/98)	Formal semantics of message sequence charts	In force

Superseded and Withdrawn components		
Number	**Title**	**Status**
Z.120 (03/93)	Messages sequence chart (MSC)	Superseded
Z.120 (1993) Annex B (03/95)	Algebraic semantics of message sequence charts	Superseded
Z.120 (10/96)	Message sequence chart (MSC)	Superseded
Z.120 Annex C (10/96)	Static semantics of message sequence charts Z.120/Annex C was withdrawn on 2002-06-12 since its content is covered by Z.120/Annex B (1998)	Withdrawn
Z.120 (11/99)	Message sequence chart (MSC)	Superseded
Z.120 (1999) Corrigendum 1 (12/01)		Superseded
Z.120 (04/04)	Message Sequence Chart (MSC)	Superseded
Z.120 (2004) Amendment 1 (09/08)	Appendix I - Application of MSC	Superseded
Z.120 (2004) Amendment 2 (09/09)	Revised Appendix I - Application of MSC	Superseded

Contact for this page : ITU-T Publications
Updated : 2014-04-01

INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

Z.120

(03/93)

30 years old

CRITERIA FOR THE USE AND APPLICABILITY OF FORMAL DESCRIPTION TECHNIQUES

MESSAGE SEQUENCE CHART (MSC)

Initial Recommendation is from **1992**.
First scientific papers about MSCs are from **1992**.

ITU-T Recommendation Z.120

(Previously "CCITT Recommendation")

Piotr Kosiuczenko:
Formal Semantics of Basic Message Sequence Charts: an Algebraic Approach. FBT 1997: 135-144

Hanêne Ben-Abdallah, Stefan Leue:
Timing Constraints in Message Sequence Chart Specifications. FORTE 1997: 91-106

André Engels, Sjouke Mauw, Michel A. Reniers:
A Hierarchy of Communication Models for Message Sequence Charts. FORTE 1997: 75-90

Nils Faltin, Lennard Lambert, Andreas Mitschele-Thiel, Frank Slomka:
An annotational extension of message sequence charts to support performance engineering. SDL Forum 1997: 307-322

Sjouke Mauw, Michel A. Reniers:
High-level message sequence charts. SDL Forum 1997: 291-306

Vladimir Levin, Doron A. Peled:
Verification of Message Sequence Charts via Template Matching. TAPSOFT 1997: 652-666

1996

Sjouke Mauw:
The Formalization of Message Sequence Charts. Comput. Networks ISDN Syst. 28(12): 1643-1657 (1996)

Ekkart Rudolph, Peter Graubmann, Jens Grabowski:
Tutorial on Message Sequence Charts. Comput. Networks ISDN Syst. 28(12): 1629-1641 (1996)

Rajeev Alur, Gerard J. Holzmann, Doron A. Peled:
An Analyzer for Message Sequence Charts. Softw. Concepts Tools 17(2): 70-77 (1996)

Stephen G. Eick, Amy Wards:
An Interactive Visualization for Message Sequence Charts. WPC 1996: 2-7

Keiji Ishikawa, Tamio Hoshino:
Rapid protocol prototyping from message sequence chart based specification. RSP 1996: 61-65

1994

Sjouke Mauw, Michel A. Reniers:
An Algebraic Semantics of Basic Message Sequence Charts. Comput. J. 37(4): 269-278 (1994)

Jos C. M. Baeten, Sjouke Mauw:
Delayed choice: an operator for joining Message Sequence Charts. FORTE 1994: 340-354

1993

Peter B. Ladkin, Stefan Leue:
What Do Message Sequence Charts Mean? FORTE 1993: 301-316

1992

Jens Grabowski, Ekkart Rudolph:
Message Sequence Chart (MSC) - A Survey of the new CCITT Language for the Description to Traces within Communications Systems. FBT 1992: 66-87

Peter B. Ladkin, Stefan Leue:
On the Semantics of Message Sequence Charts. FBT 1992: 88-104

Search for “message sequence chart” on dblp.org (accessed June-26-2023)

behaviour of a system. Although the name *Message Sequence Chart* obviously originates from its graphical representation, it is used both for the textual and the graphical representation.

The Message Sequence Chart heading consists of the Message Sequence Chart name and (optionally) a list of the instances being contained in the Message Sequence Chart body.

Abstract grammar

Message-sequence-chart	::	*MSC-name* [*MSC-interface*] *MSC-body*
MSC-name	=	*Name*
MSC-interface	::	*Instance-list*
Instance-list	=	*Instance-declaration+*
Instance-declaration	::	*Instance-name* [*Instance-kind*]
Instance-name	=	*Name*
Instance-kind	=	*System-name* | *Block-name* | *Process-name* | *Service-name* | *Name*
System-name	::	*Name*
Block-name	::	*Name*
Process-name	::	*Name*
Service-name	::	*Name*
MSC-body	::	(*Instance-definition* | *Text-definition*)*
Text-definition	::	*Informal-text*

The *Instance-list* in the *MSC-interface*, if present, must contain the same instances as specified in the *MSC-body*.

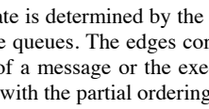
Concrete textual grammar

<msc head> ::=
<message sequence chart name> <end> [<msc interface>]
<msc interface> ::=
inst <instance list> <end>
<instance list> ::=
<instance name> [**:** <instance kind>] [**,** <instance list>]
<instance kind> ::=
[<kind denominator>] <kind name>

<kind denominator> ::=
system | block | process | service
<msc body> ::=
{ <instance definition> | <text definition> }*

Concrete graphical grammar

<msc diagram> ::=
<msc symbol> ***contains*** { <msc heading> <msc body area> }
<msc body area> ::=
{ <instance area> | <external message area> | <text area> }*
<msc symbol> ::=
<frame symbol>
<frame symbol> ::=



T1007630-93/d03

<msc heading> ::=
msc <message sequence chart name>

Semantics

An MSC describes the communication between a number of system components, and between these components and the rest of the world, called environment. For each system component covered by an MSC there is an instance axis. The communication between system components is performed by means of messages. The sending and consumption of messages are two different events. It is assumed that the environment of an MSC is capable of receiving and sending messages from and to the Message Sequence Chart; no ordering of message events within the environment is assumed. Although the behaviour of the environment is non-deterministic, it is assumed to obey the constraints given by the Message Sequence Chart.

No global time axis is assumed for one Message Sequence Chart. Along each instance axis the time is running from top to bottom, however, we do not assume a proper time scale. If no coregion is introduced (see 5.1) a total time ordering of events is assumed along each instance axis. Events of different instances are ordered only via messages: a message must first be sent before it is consumed (see 4.3). No other ordering is prescribed. A Message Sequence Chart therefore imposes a partial ordering on the set of events being contained. A binary relation which is transitive, antisymmetric and reflexive is called partial order.

For the message inputs (labelled by ?mi) and outputs (labelled by !mi) of the Message Sequence Chart in Figure 1a) we derive the following ordering relation: !m2 < ?m2, !m3 < ?m3, !m4 < ?m4, ?m1 < !m2 < !m3 < ?m4, ?m2 < !m4 together with the transitive closure.

The partial ordering can be described in a minimal form (without an explicit representation of the transitive closure) by its connectivity graph [see Figure 1b)].

The semantics of an MSC can be related to the semantics of SDL by the notion of a reachability graph. Each sequentialization of an MSC describes a trace from one node to another node (or a set of nodes) of the reachability graph describing the behaviour of an SDL system specification. The reachability graph consists of nodes and edges. Nodes denote global system states. A global system state is determined by the values of the variables and the state of execution of each process and the contents of the message queues. The edges correspond to the events which are executed by the system, e.g. the sending and the consumption of a message or the execution of a task. A sequentialization of an MSC denotes one total ordering of events compatible with the partial ordering defined by the MSC.

Note that the reachability graph shows more events than the MSC. Thus, the sequentializations of the MSCs are not complete paths of the reachability graph.

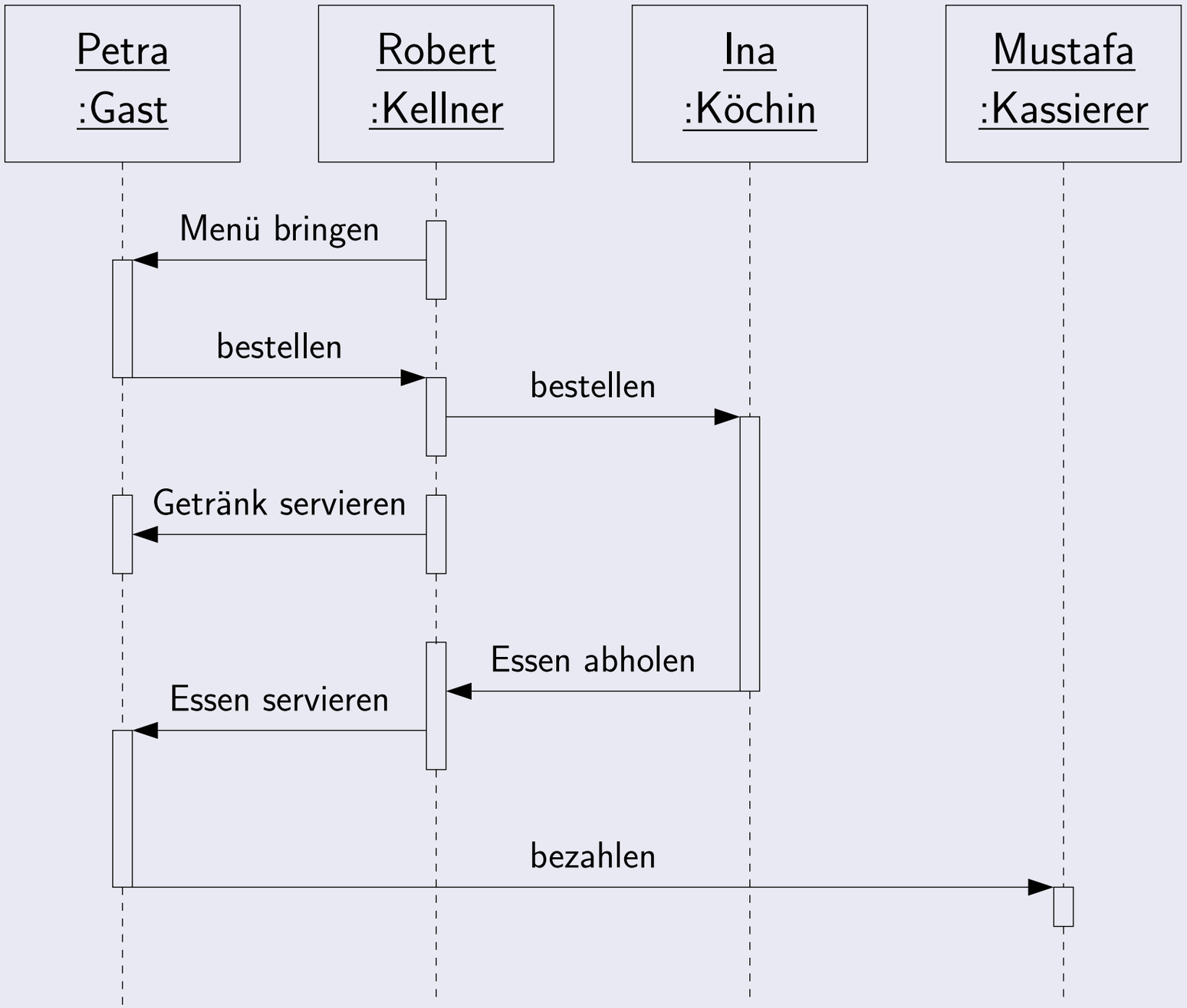
Sequenzdiagramme

Sequenzdiagramme beschreiben Interaktionen in zwei Dimensionen:

- Von links nach rechts: Anordnung der Kommunikationspartner als Lebenslinien
  Oft wird der Partner, der den Ablauf initiiert ganz links angegeben.
- Von oben nach unten: Zeitachse

Sequenzdiagramme

Beispiel Restaurant

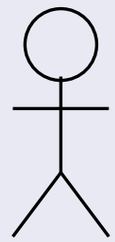


Sequenzdiagramme

Kommunikationspartner

Die Kommunikationspartner in einem Sequenzdiagramm werden ähnlich wie in Objektdiagrammen als Rechtecke notiert.

Manchmal werden die Rechtecke auch weggelassen. Menschliche Partner werden auch durch ein Strichmännchen symbolisiert:

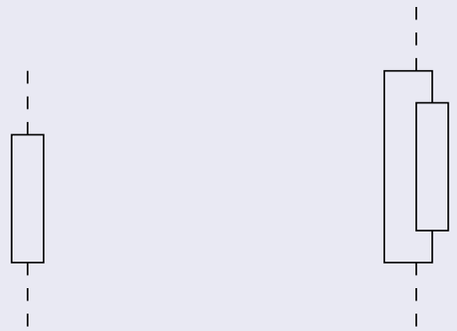


Von jedem Kommunikationspartner führt eine gestrichelte Lebenslinie nach unten.

Sequenzdiagramme

Ausführungsbalken

Aktivitäten eines Kommunikationspartners werden durch sogenannte Ausführungsbalken dargestellt.



Parallele Tätigkeiten eines Kommunikationspartners werden dabei durch übereinander liegende Ausführungsbalken beschrieben (siehe rechts oben).

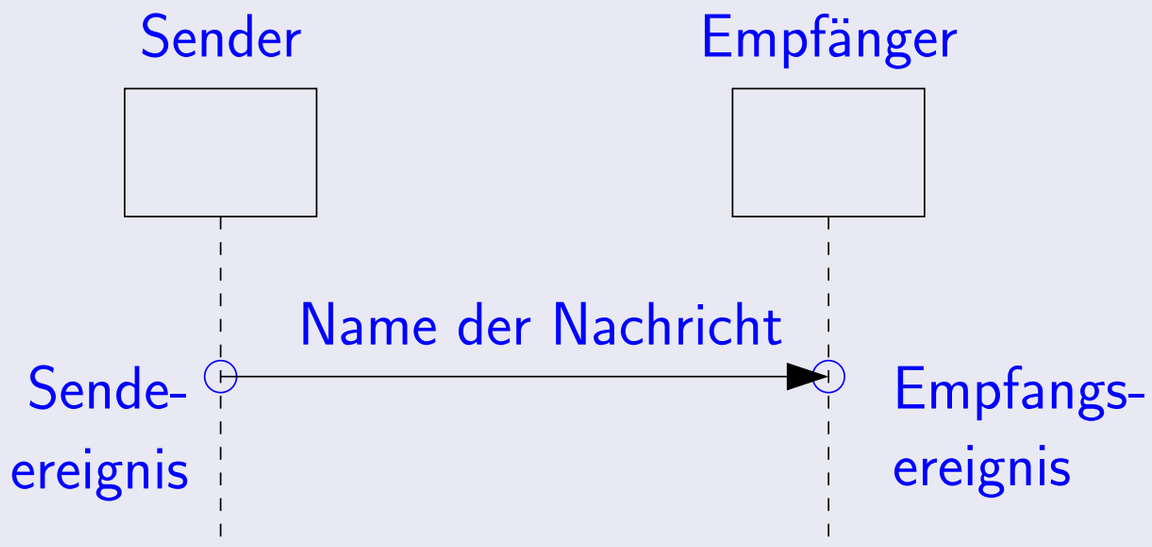
Während die Balken aktive Zeit anzeigen, symbolisieren die gestrichelten Linien passive Zeit.

Sequenzdiagramme

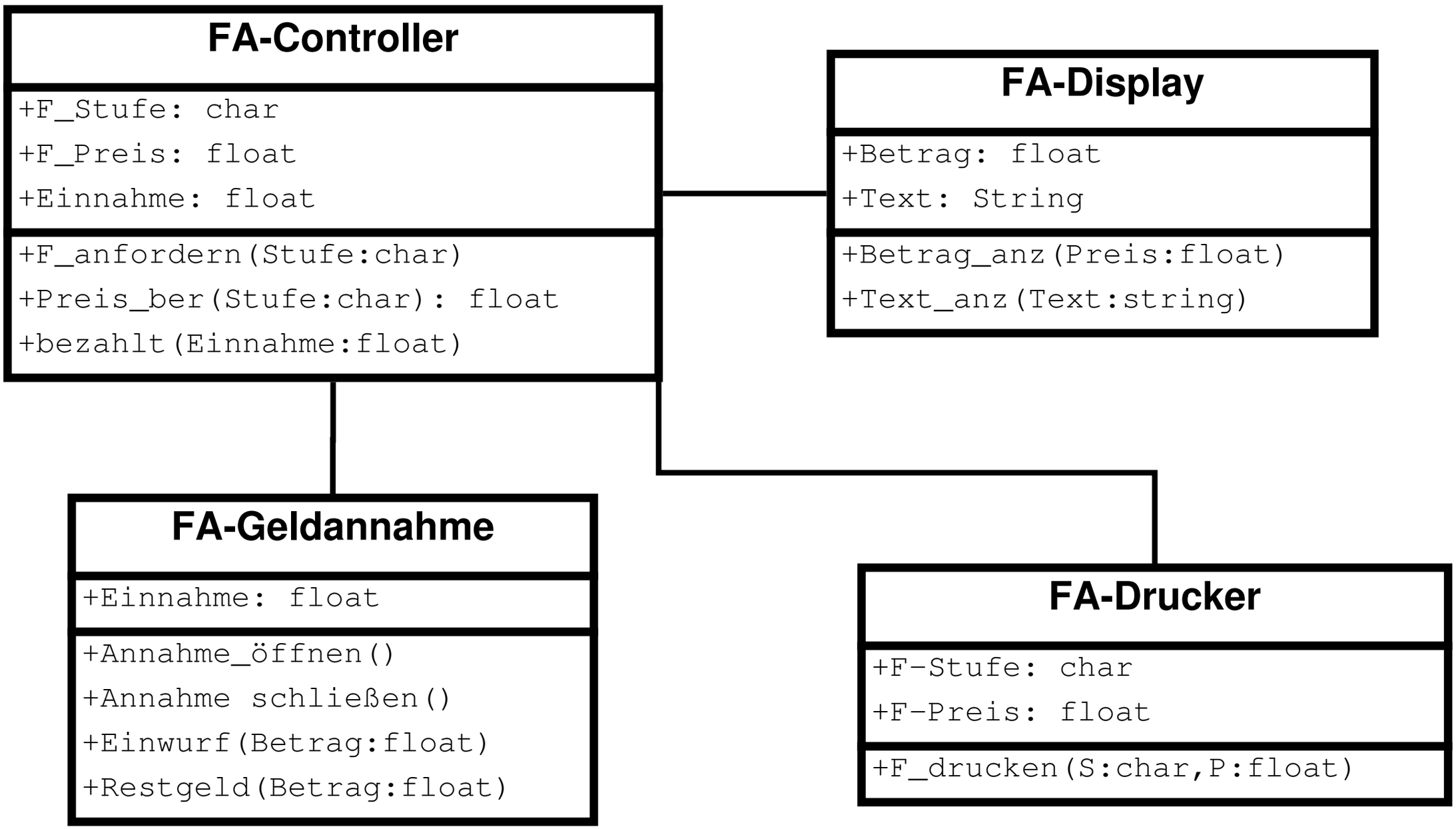
Nachrichten

Die Nachrichten beschreiben die Kommunikationen bzw. Interaktionen der Kommunikationspartner und werden durch Pfeile dargestellt. Eine Nachricht hat einen Sender und einen Empfänger.

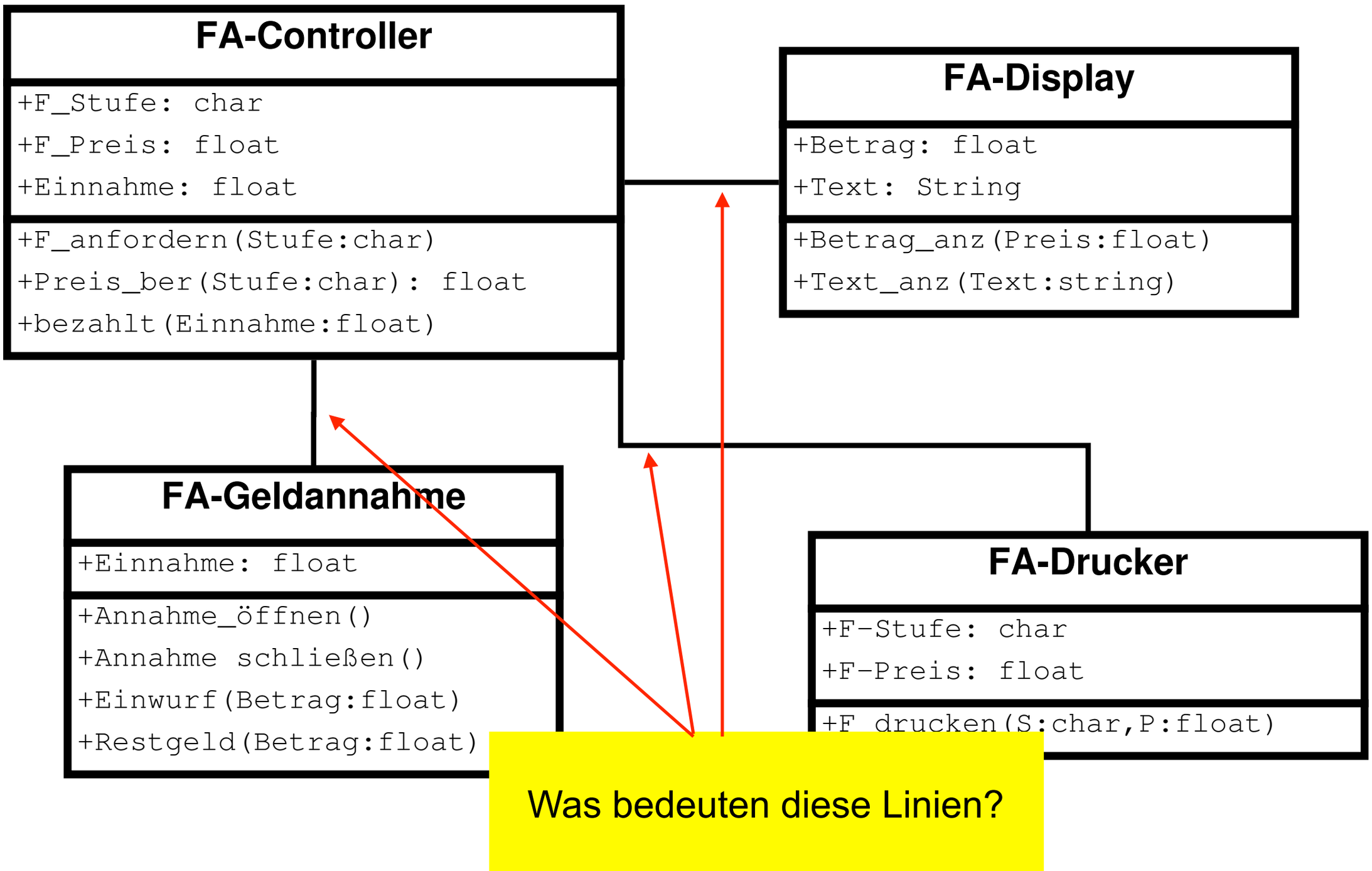
Die Stellen, an denen die Pfeile auf den Lebenslinien auftreffen, nennt man auch Sendeereignis und Empfangsereignis.



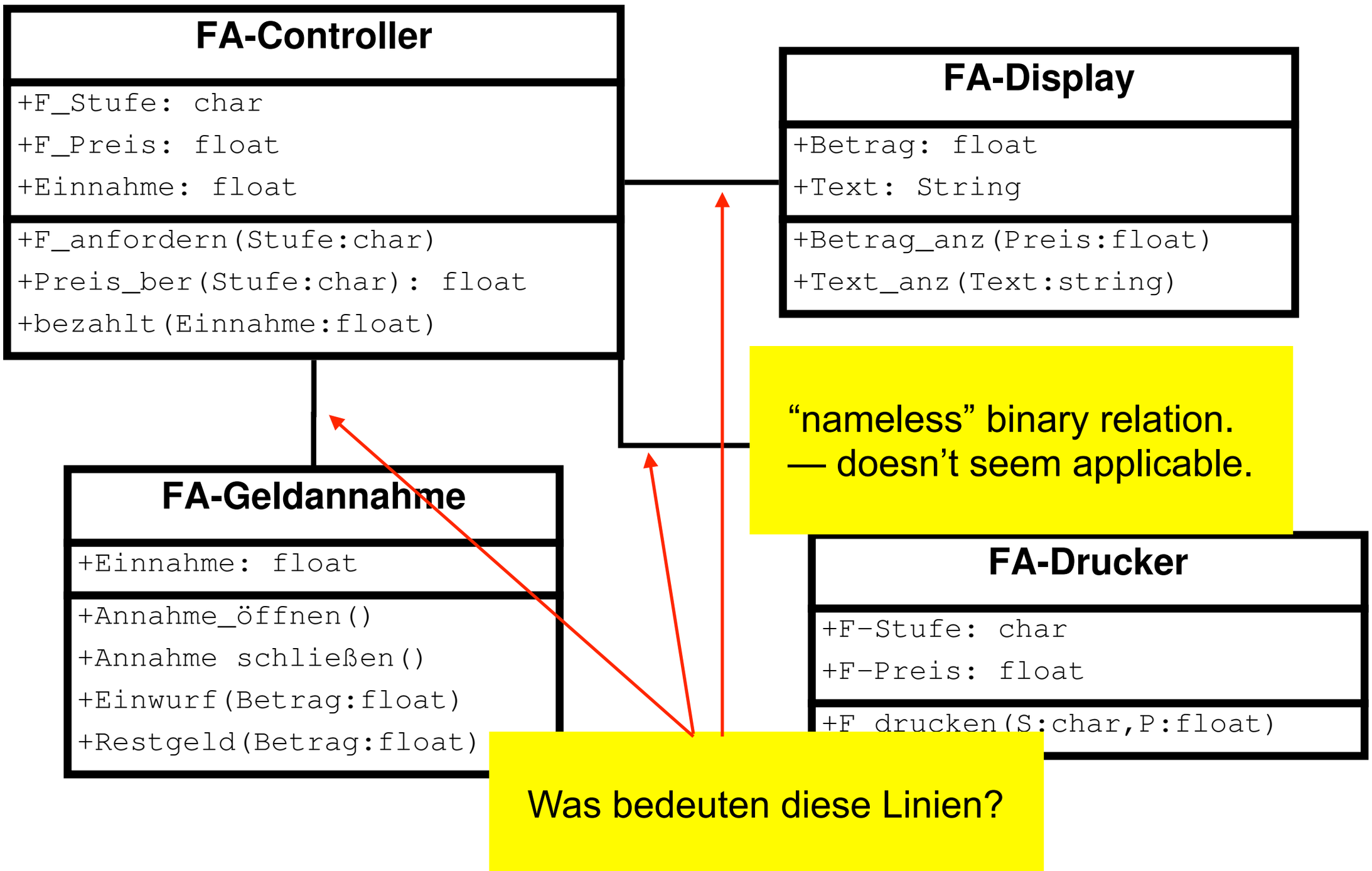
Fallstudie: Fahrkartenautomat - Klassendiagramm



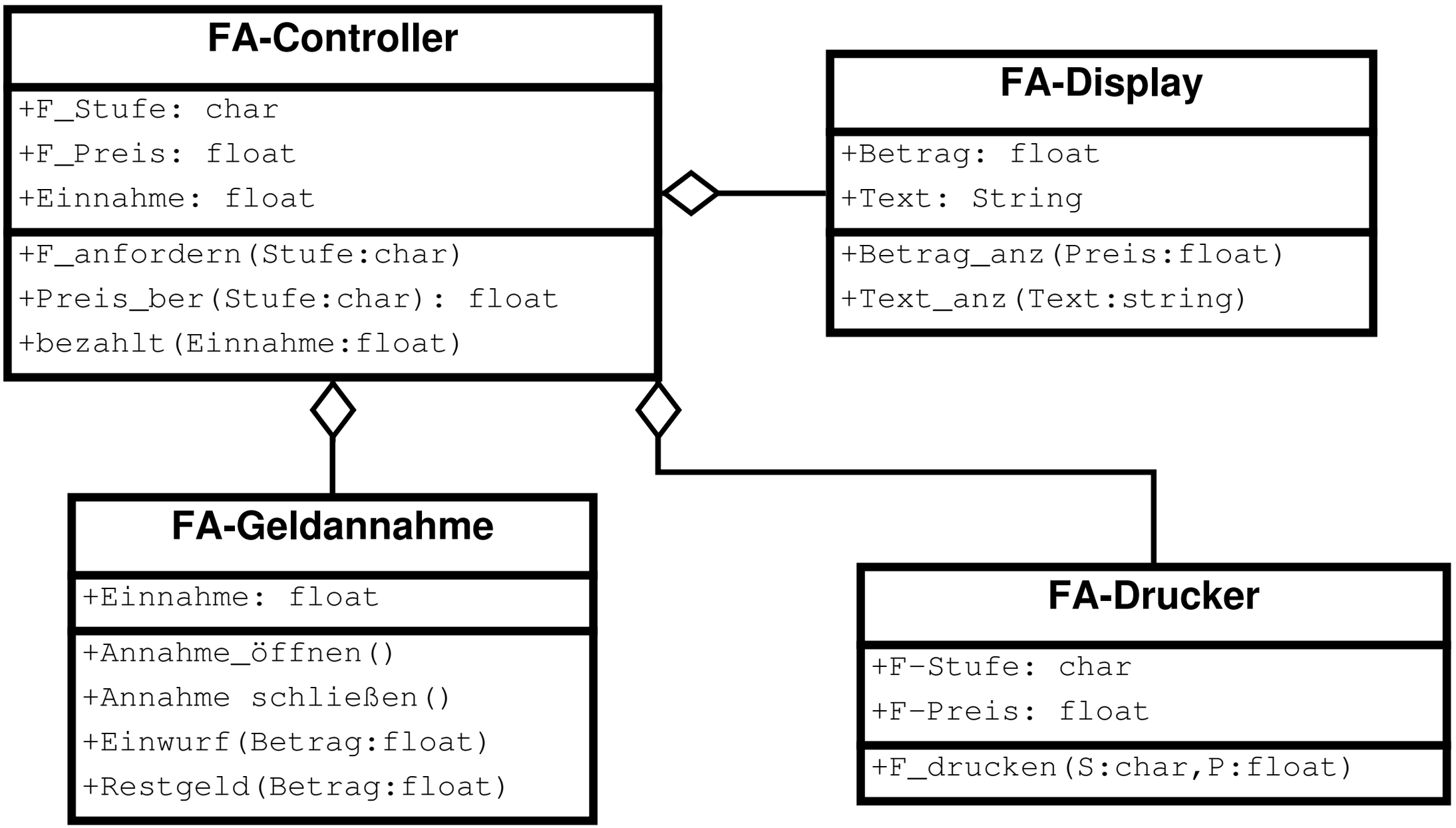
Fallstudie: Fahrkartenautomat - Klassendiagramm



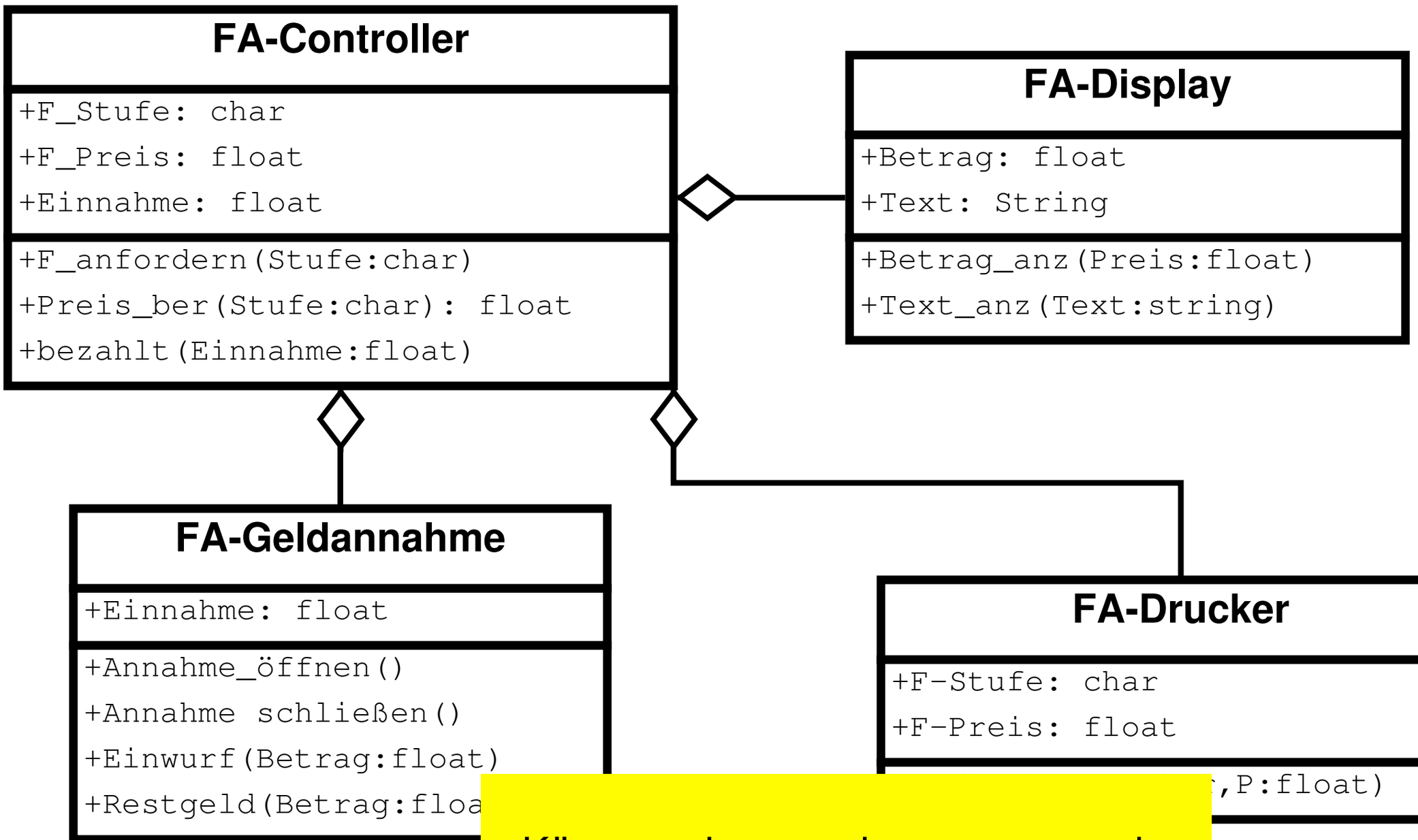
Fallstudie: Fahrkartenautomat - Klassendiagramm



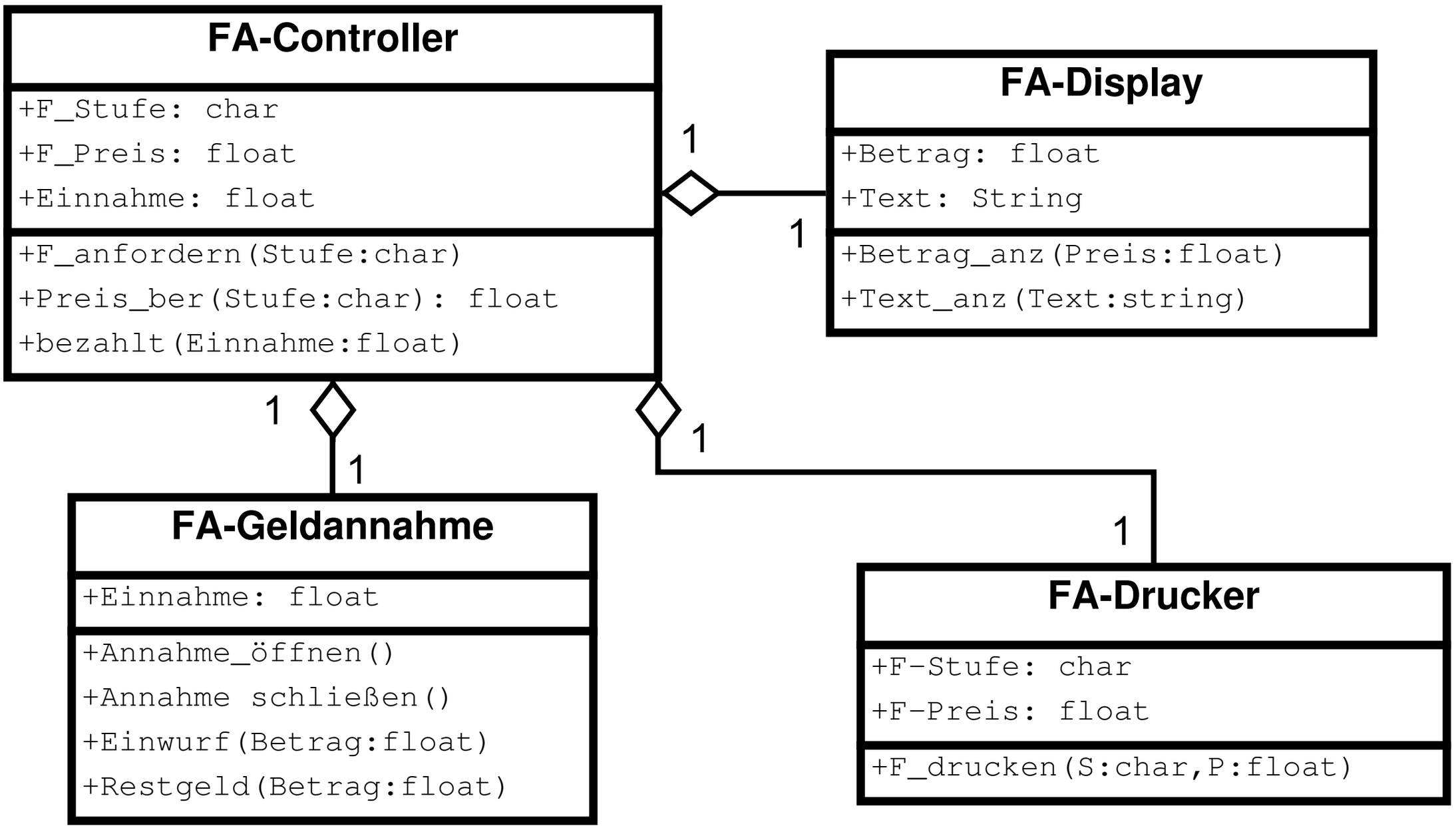
Fallstudie: Fahrkartenautomat - Klassendiagramm



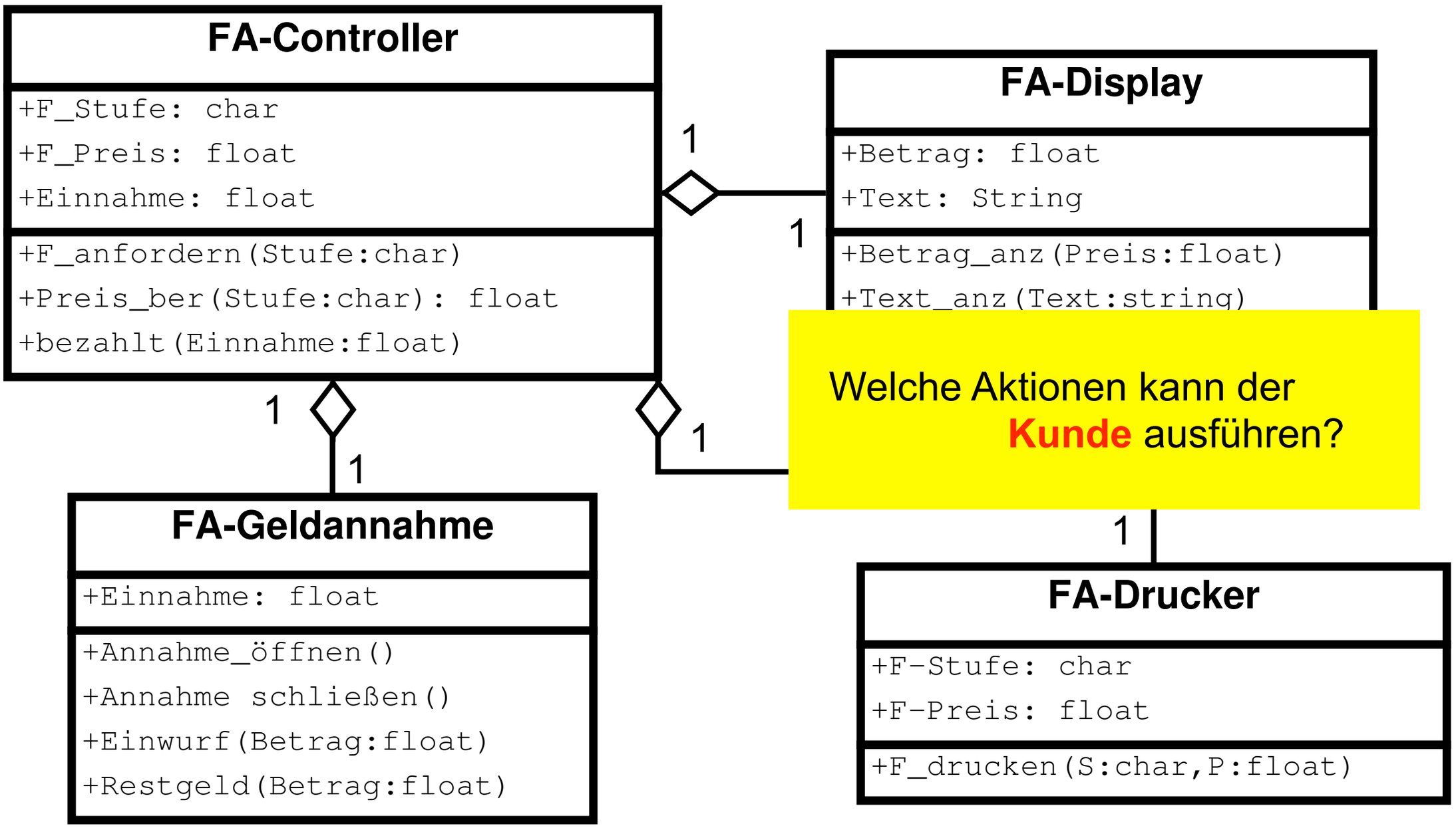
Fallstudie: Fahrkartenautomat - Klassendiagramm



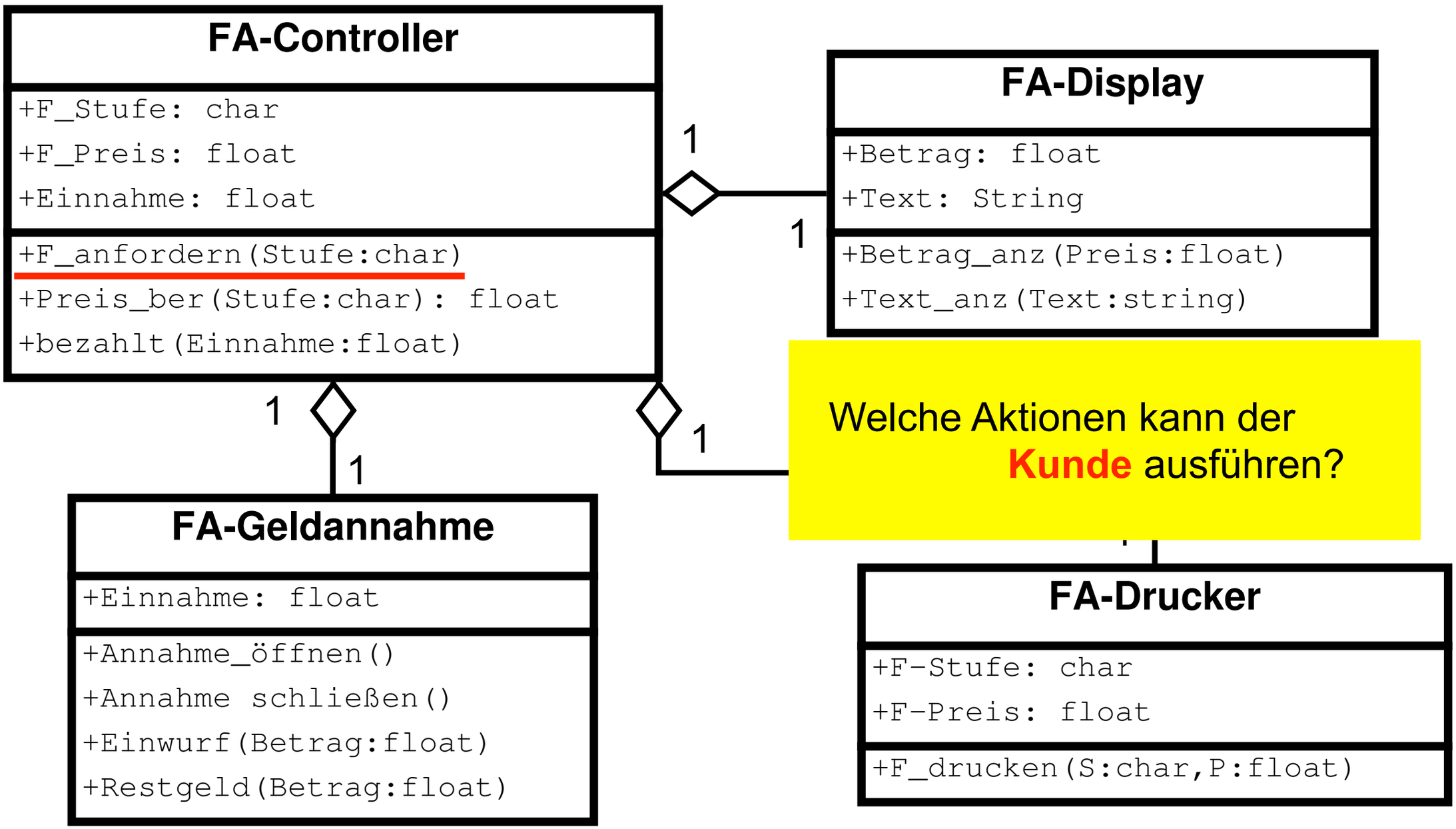
Fallstudie: Fahrkartenautomat - Klassendiagramm



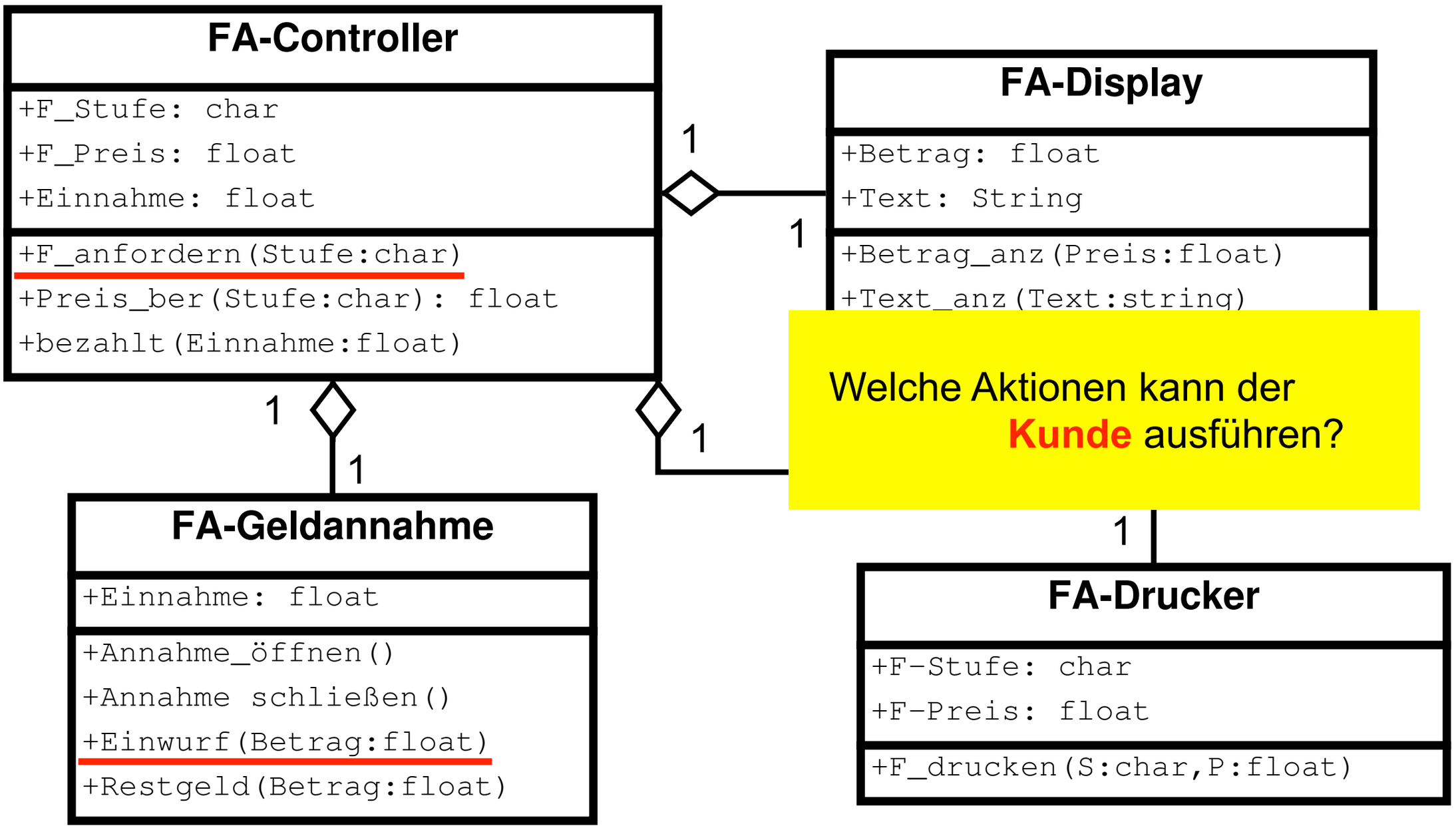
Fallstudie: Fahrkartenautomat - Klassendiagramm



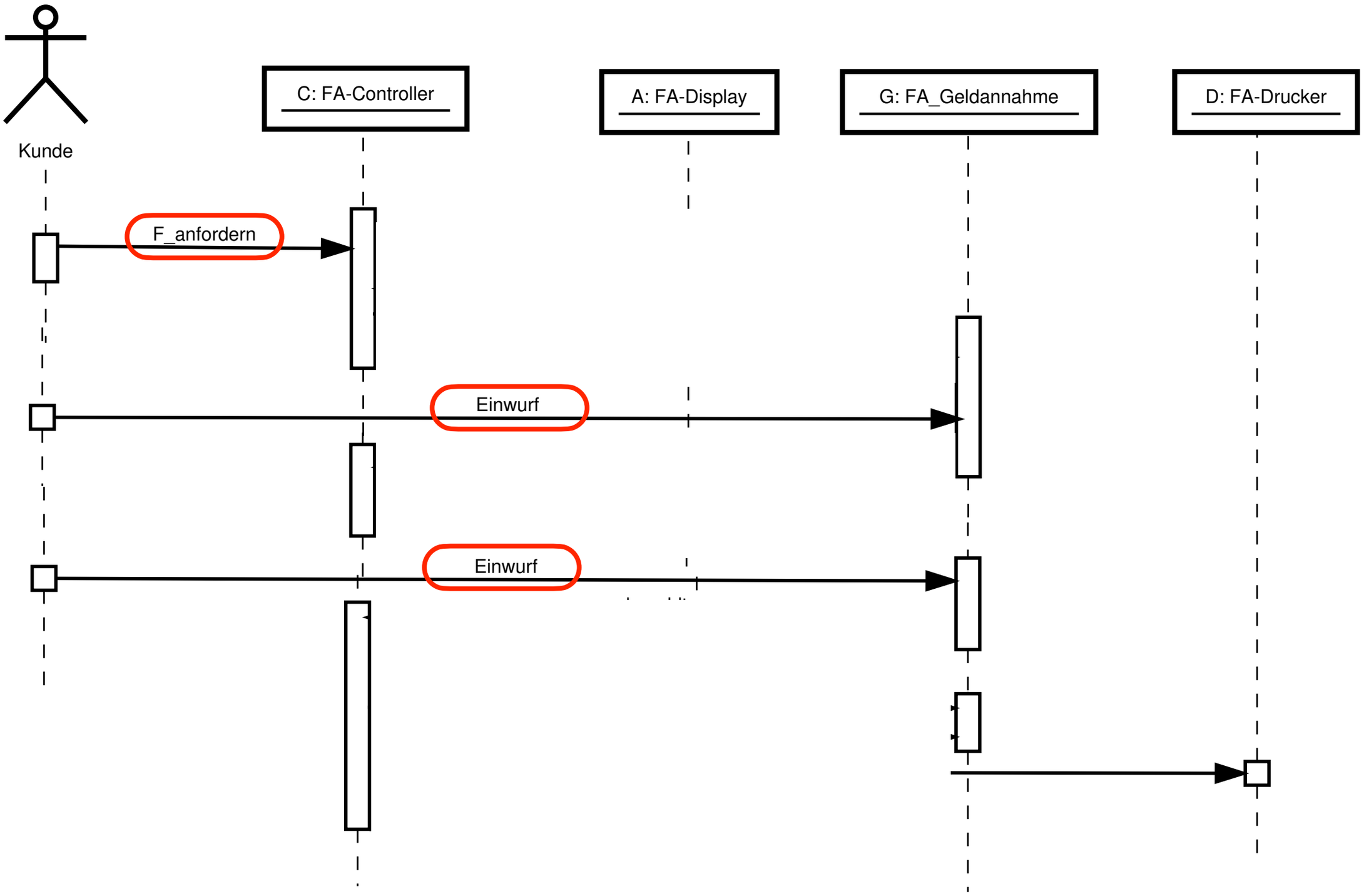
Fallstudie: Fahrkartenautomat - Klassendiagramm



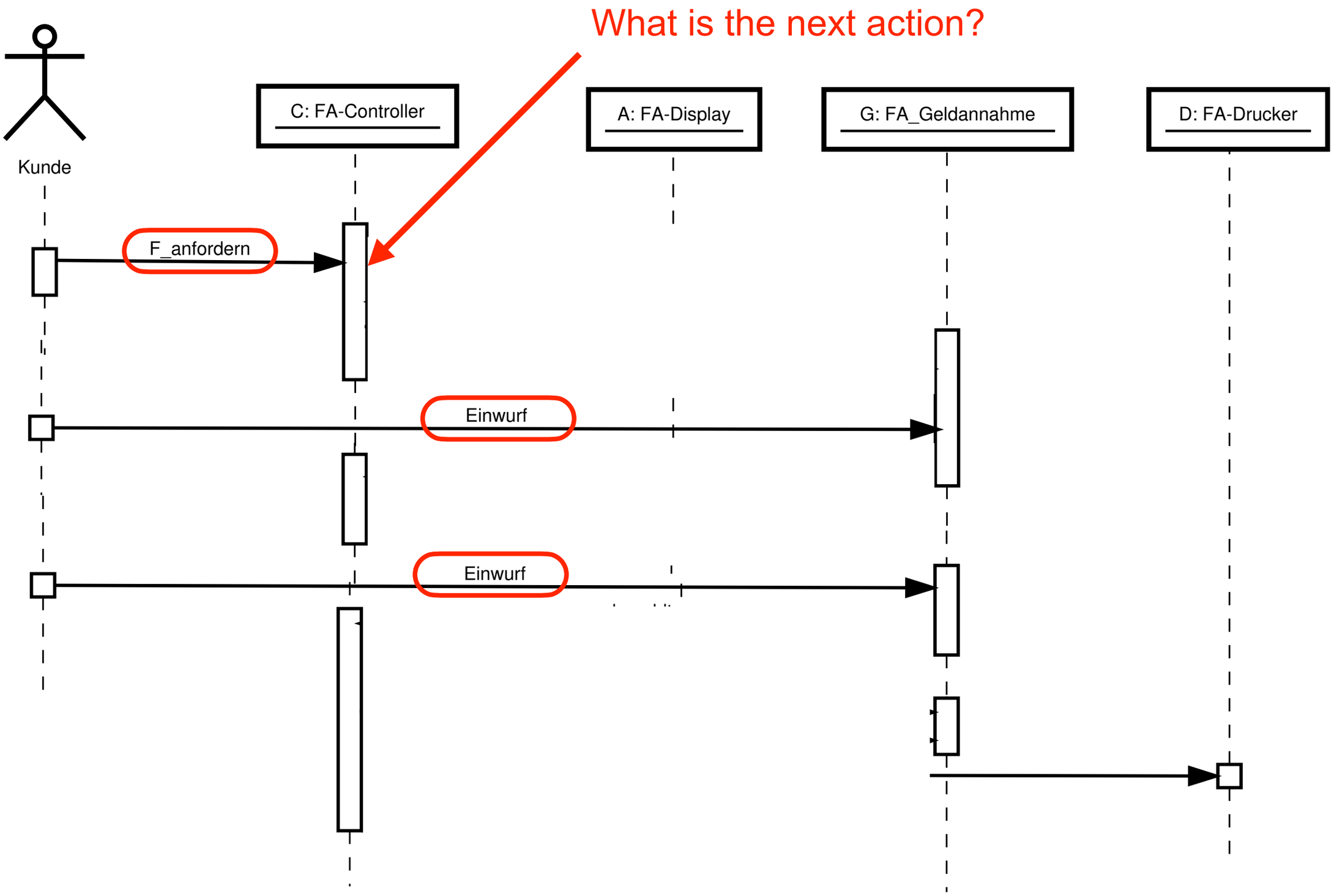
Fallstudie: Fahrkartenautomat - Klassendiagramm



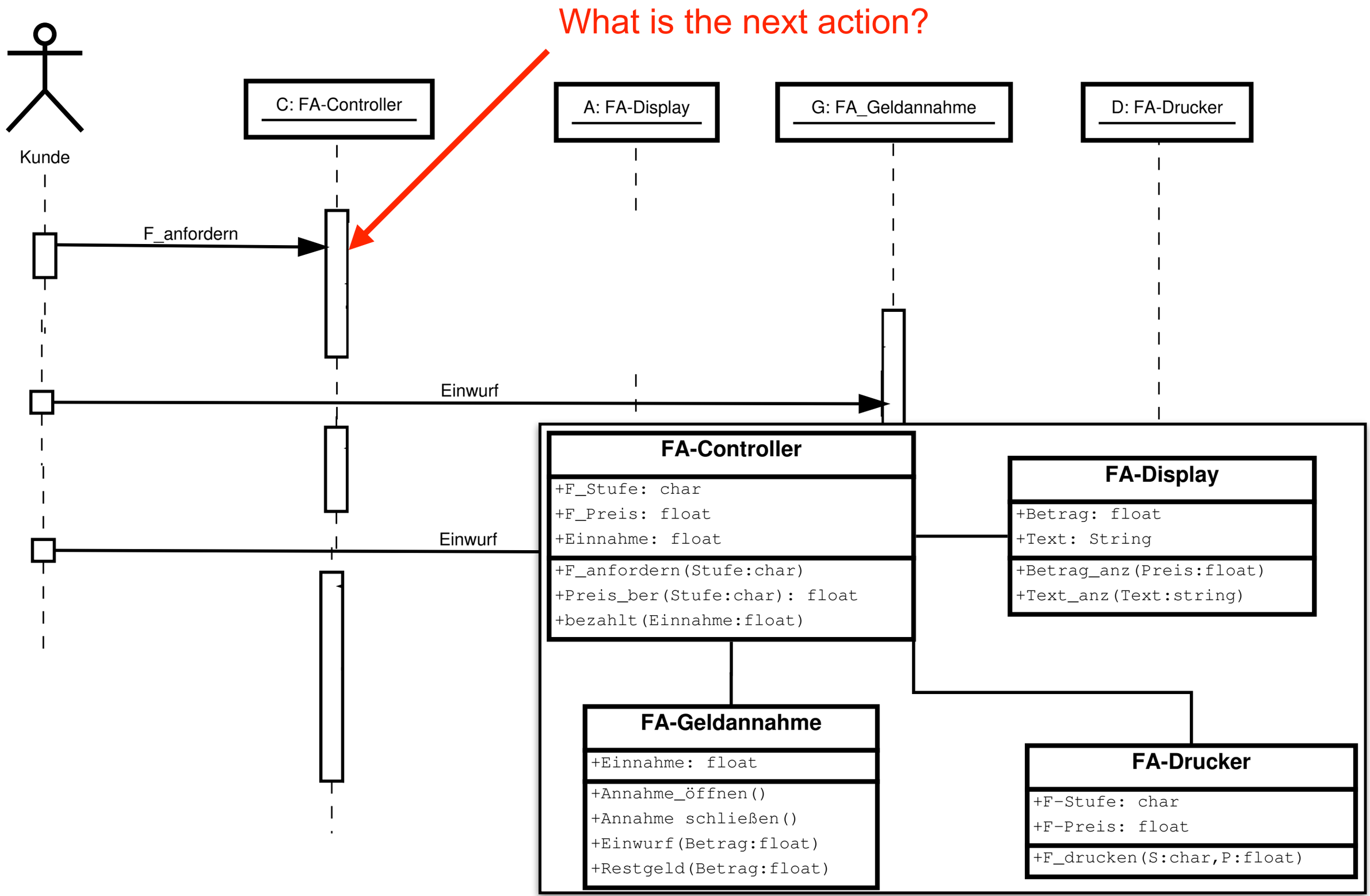
Fallstudie: Fahrkartenautomat - Sequenzdiagramm



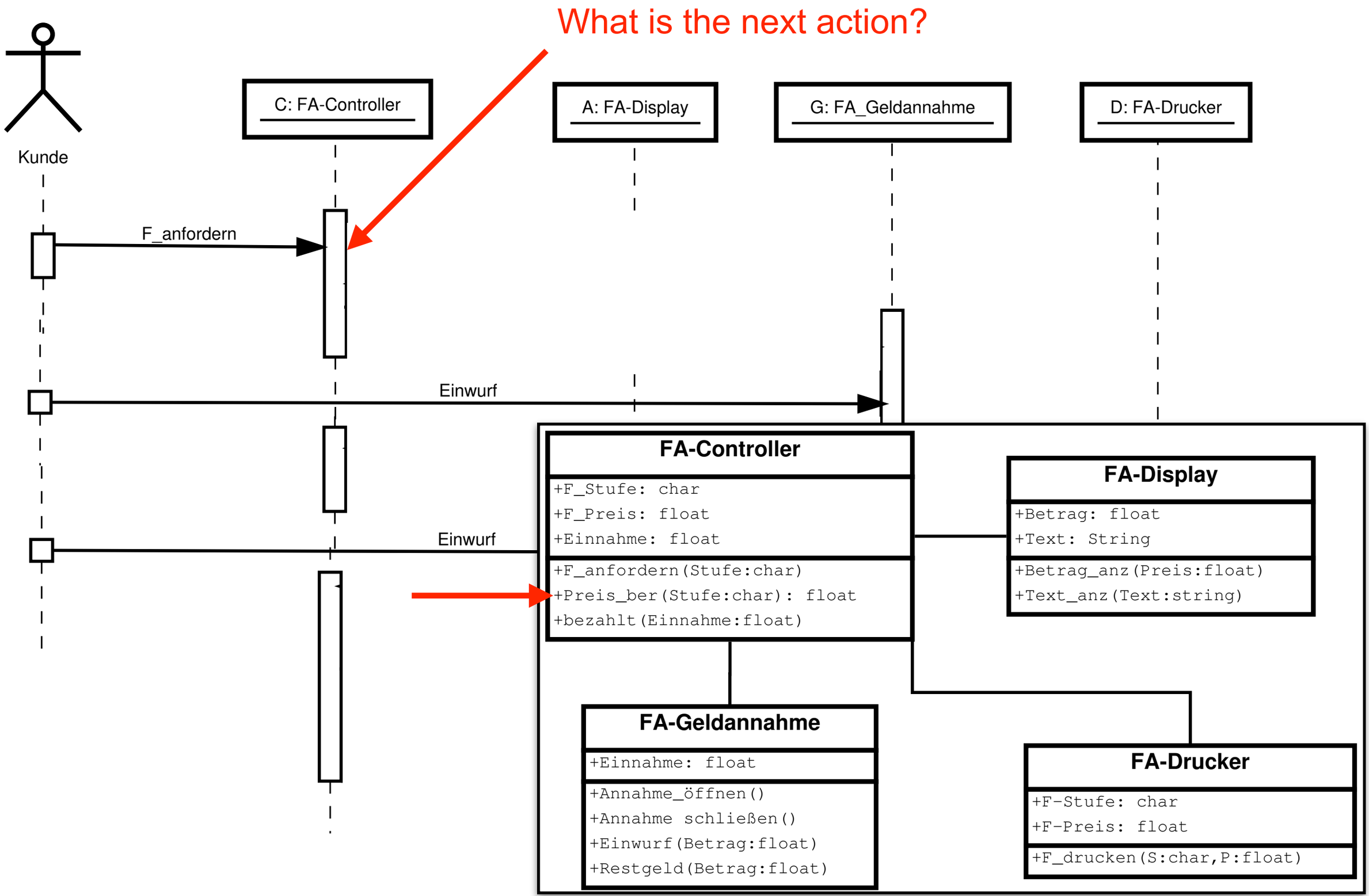
Fallstudie: Fahrkartenautomat - Sequenzdiagramm



Fallstudie: Fahrkartenautomat - Sequenzdiagramm

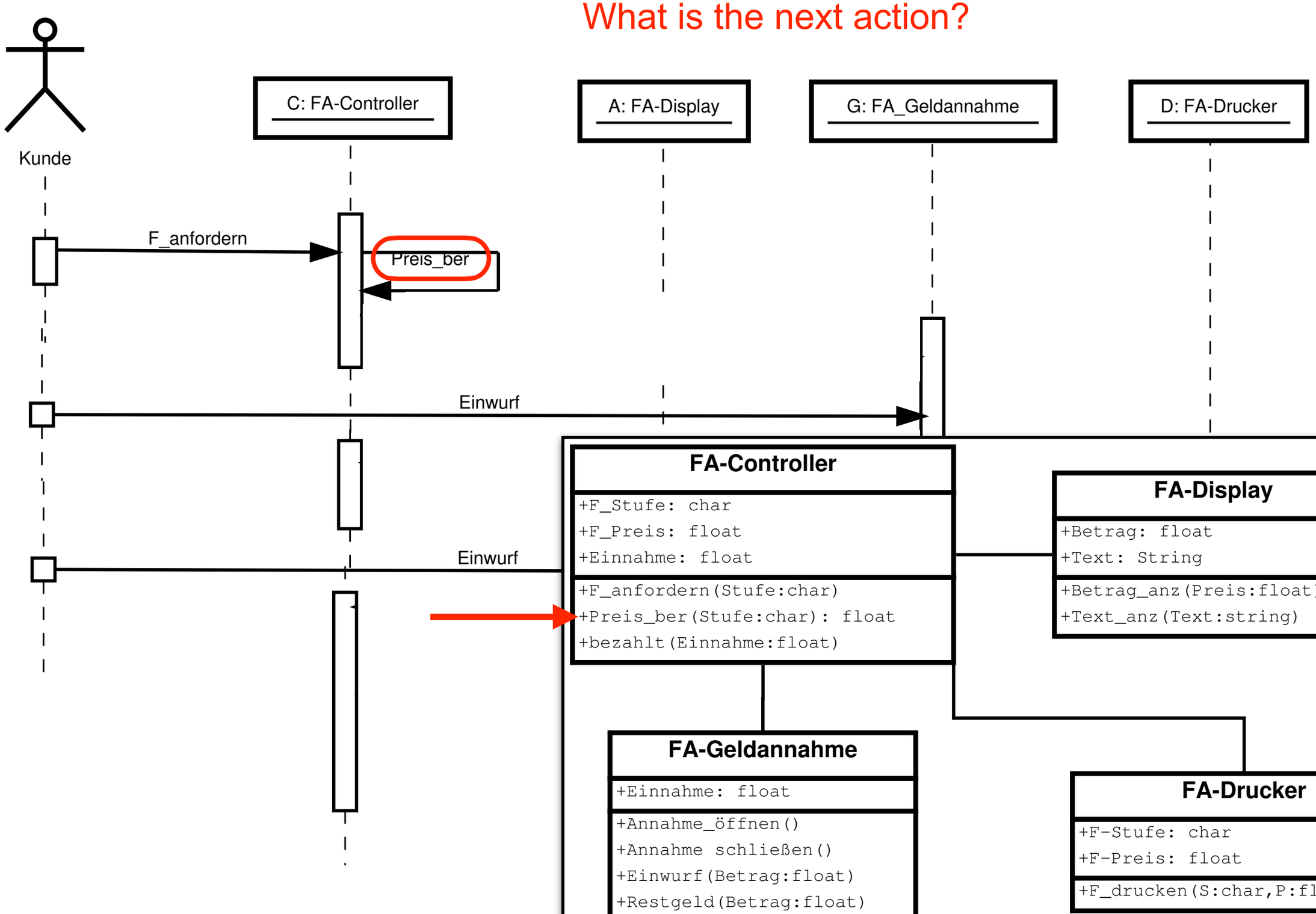


Fallstudie: Fahrkartenautomat - Sequenzdiagramm



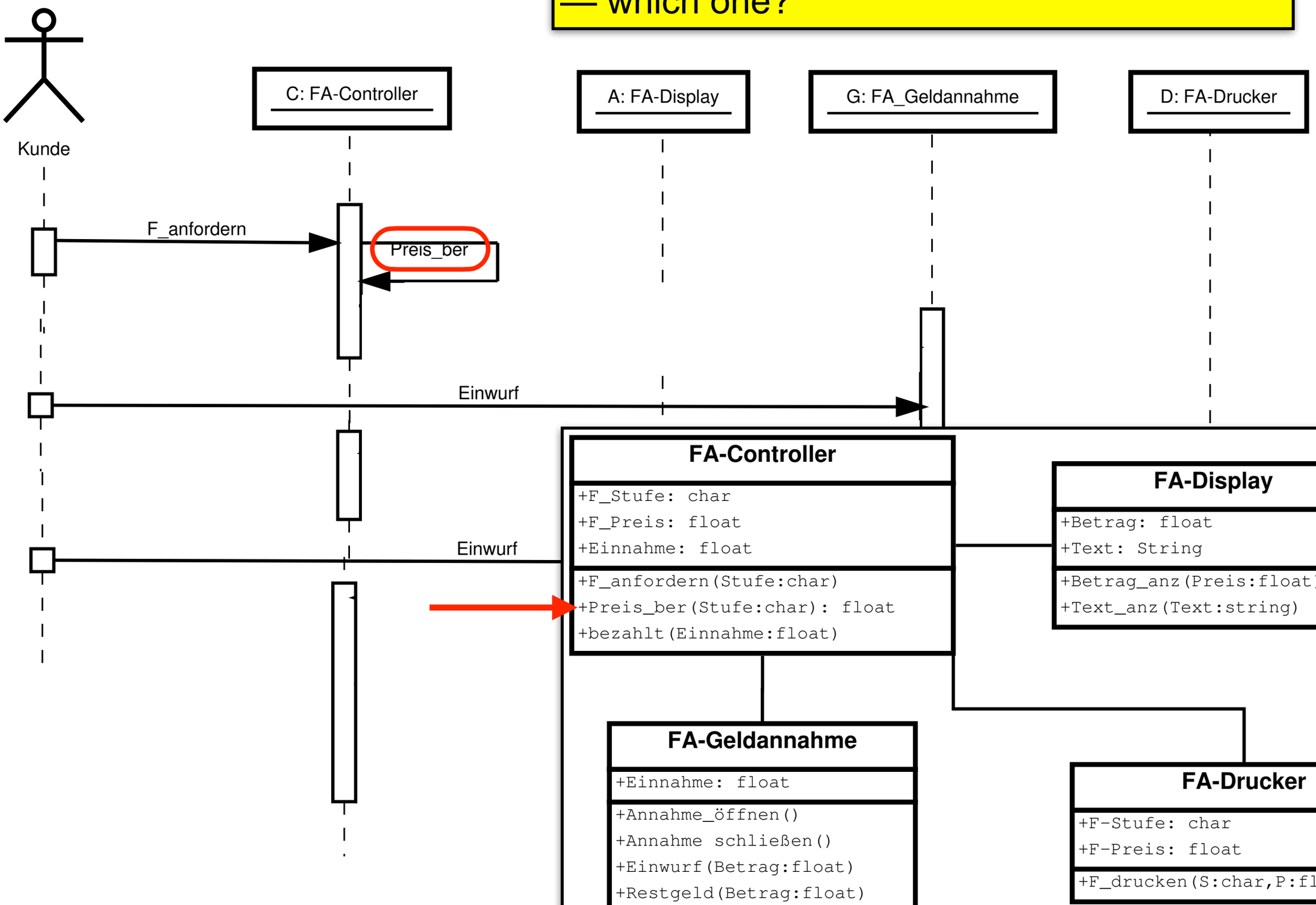
Fallstudie: Fahrkartenautomat - Sequenzdiagramm

What is the next action?



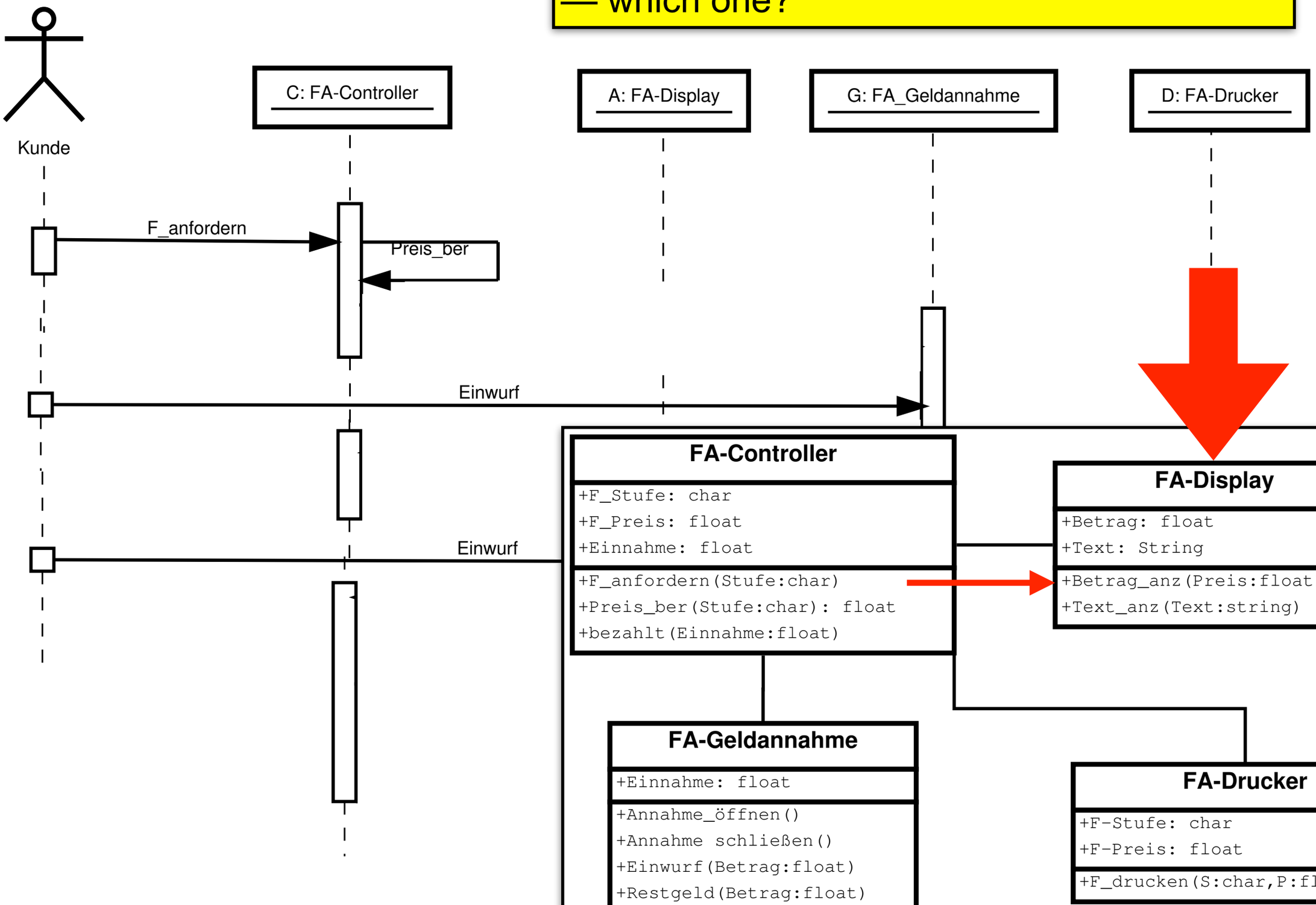
Fallstudie: Fahrkartena

What is the next action?
FA-Controller calls method of **another class.**
— which one?



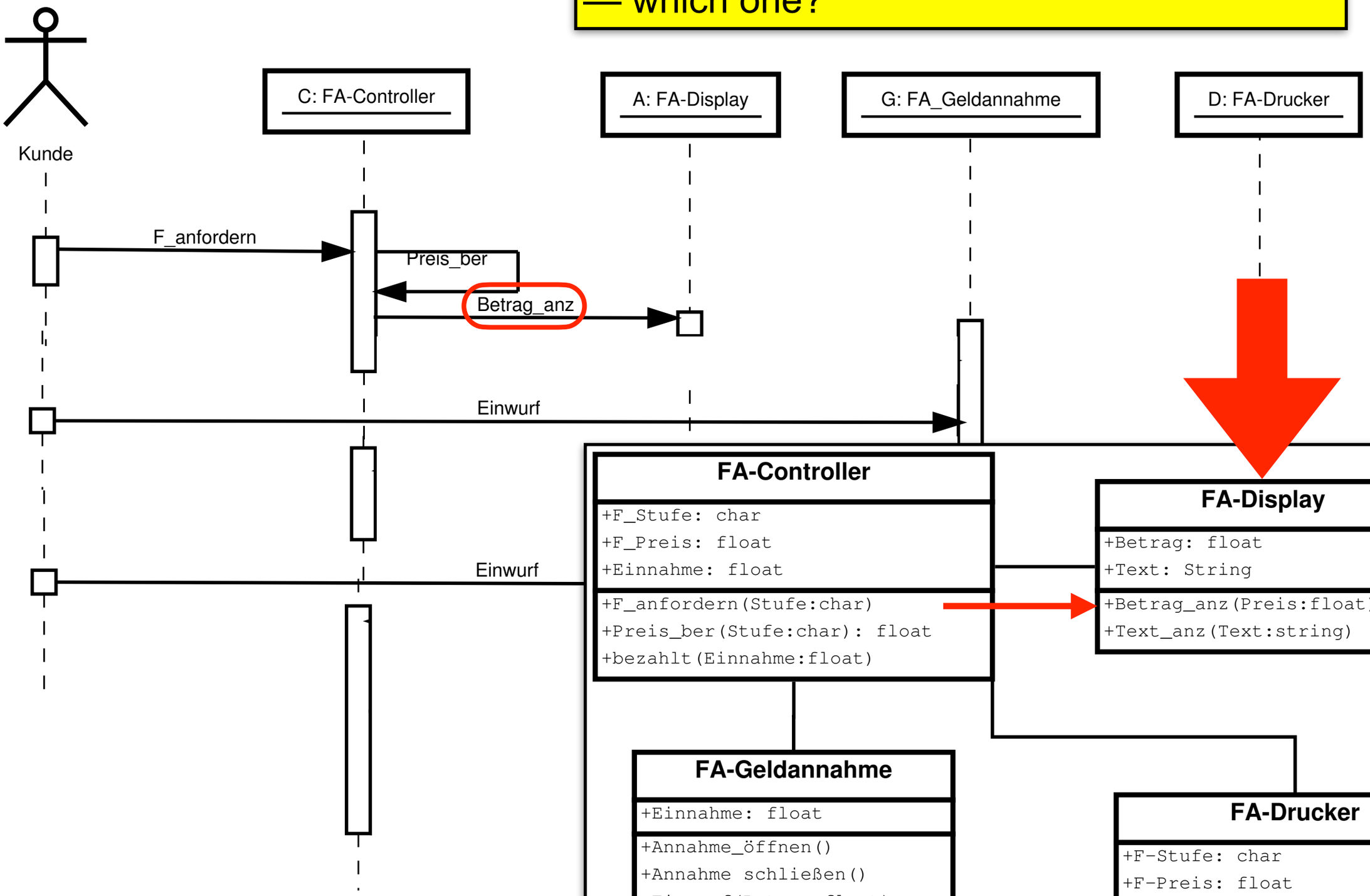
Fallstudie: Fahrkartena

What is the next action?
FA-Controller calls method of **another class.**
— which one?

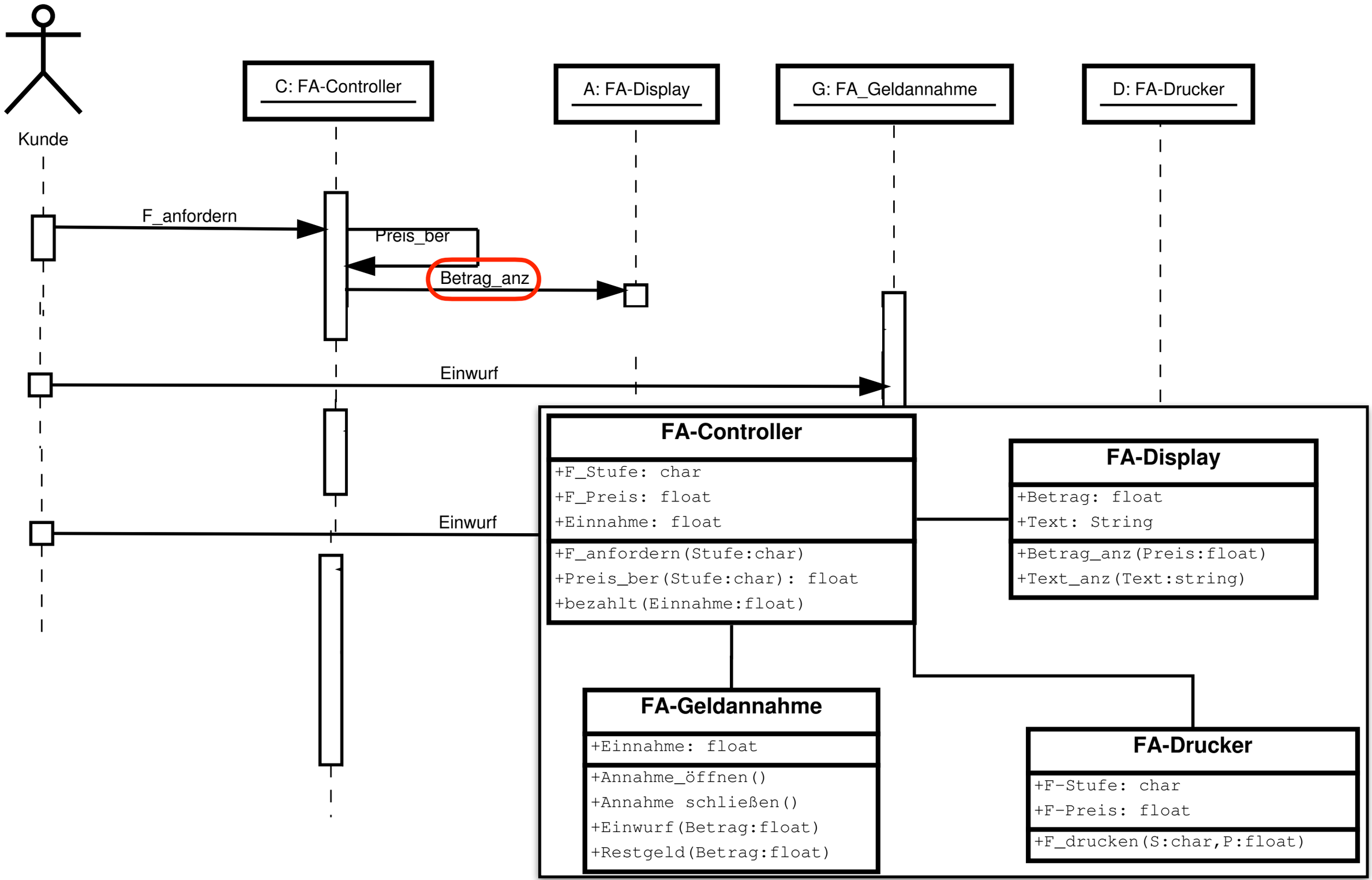


Fallstudie: Fahrkartena

What is the next action?
FA-Controller calls method of **another class.**
— which one?

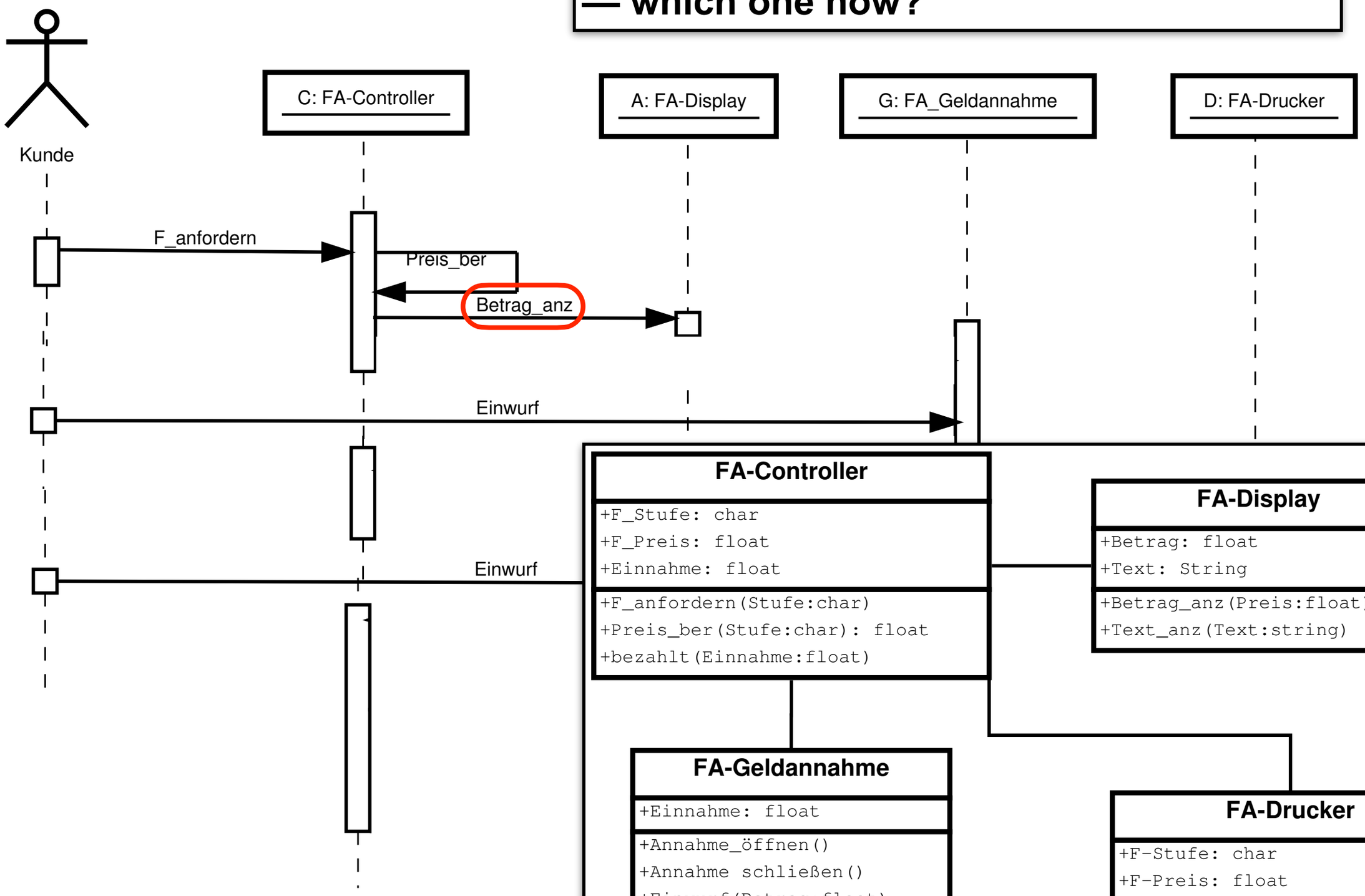


Fallstudie: Fahrkartenautomat - Sequenzdiagramm



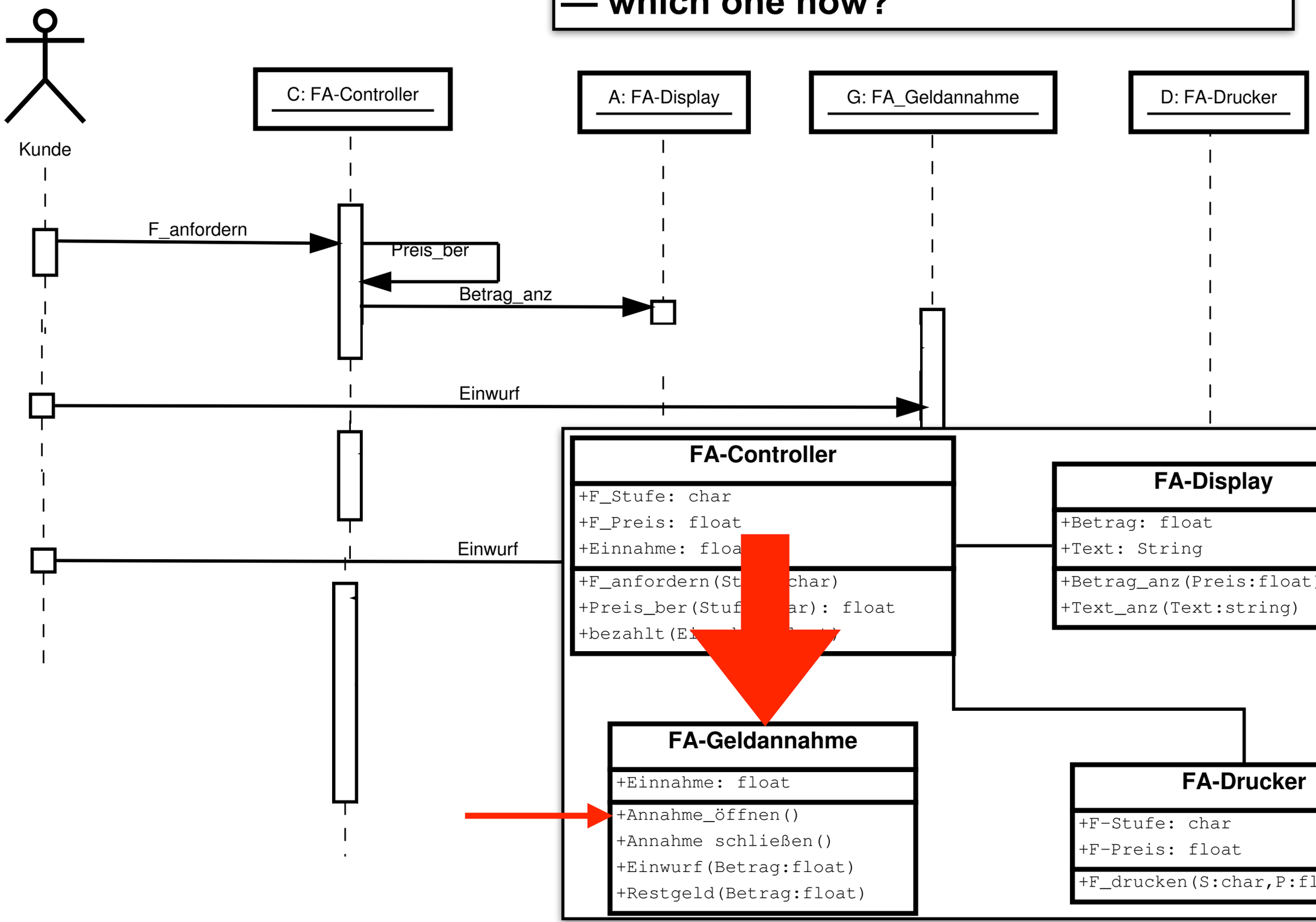
Fallstudie: Fahrkartena

What is the next action?
FA-Controller calls method of **another class**!
— which one now?



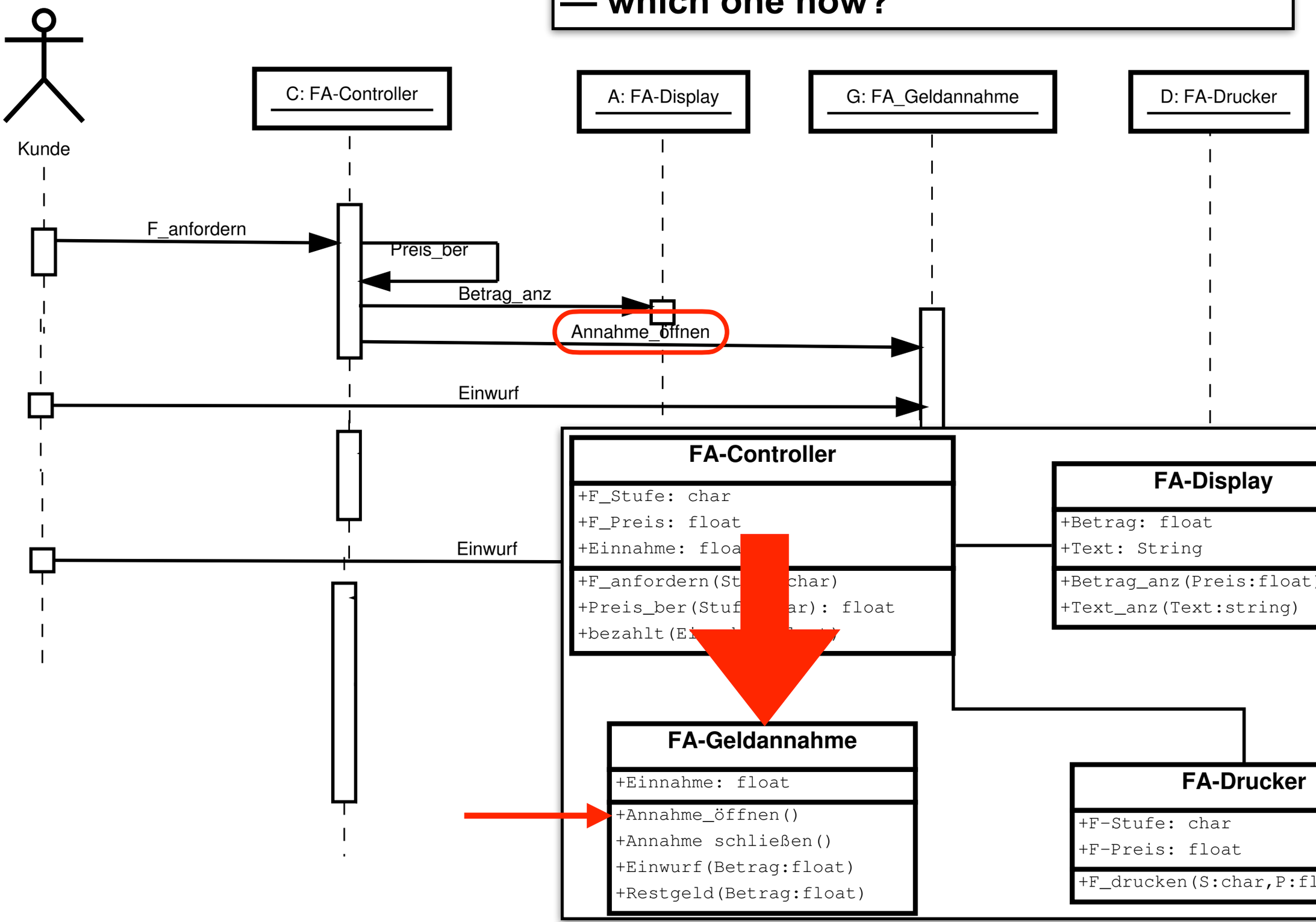
Fallstudie: Fahrkartena

What is the next action?
FA-Controller calls method of **another class**!
— which one now?

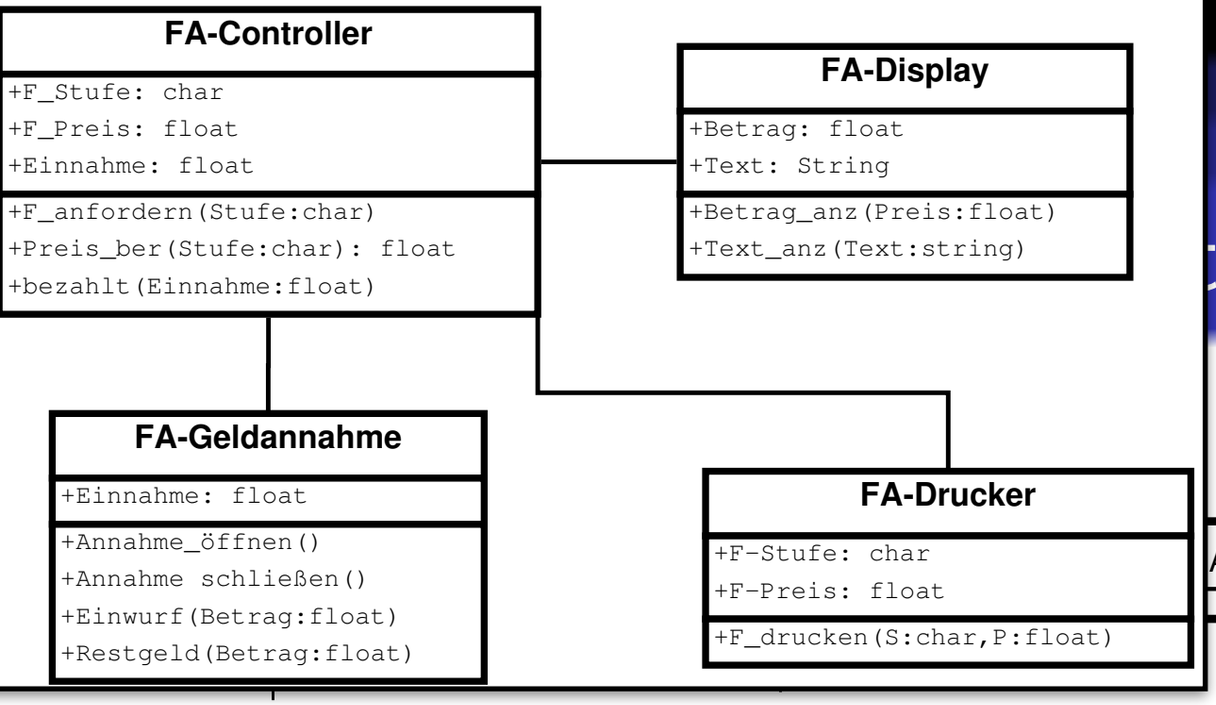


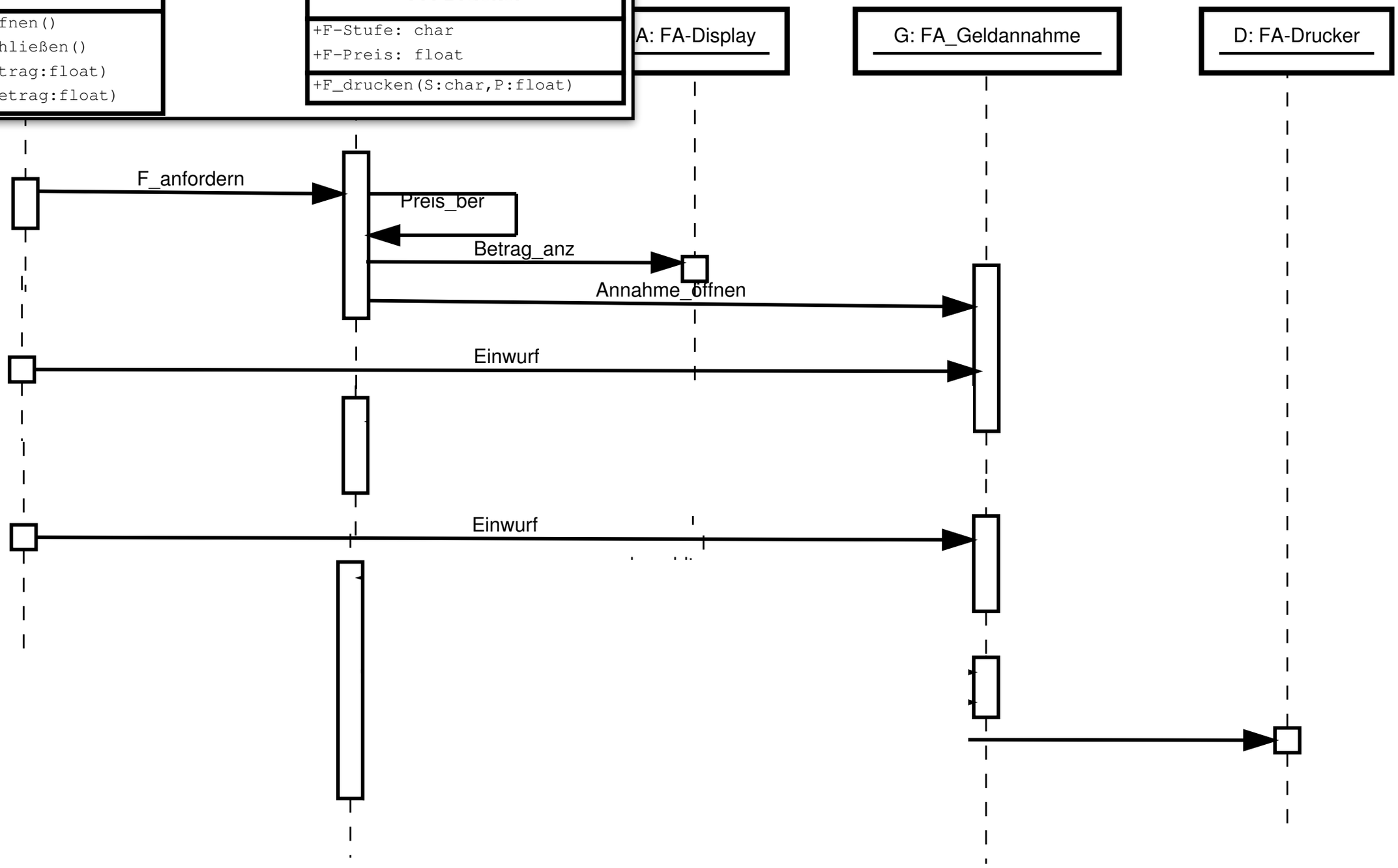
Fallstudie: Fahrkartena

What is the next action?
FA-Controller calls method of **another class**!
— which one now?

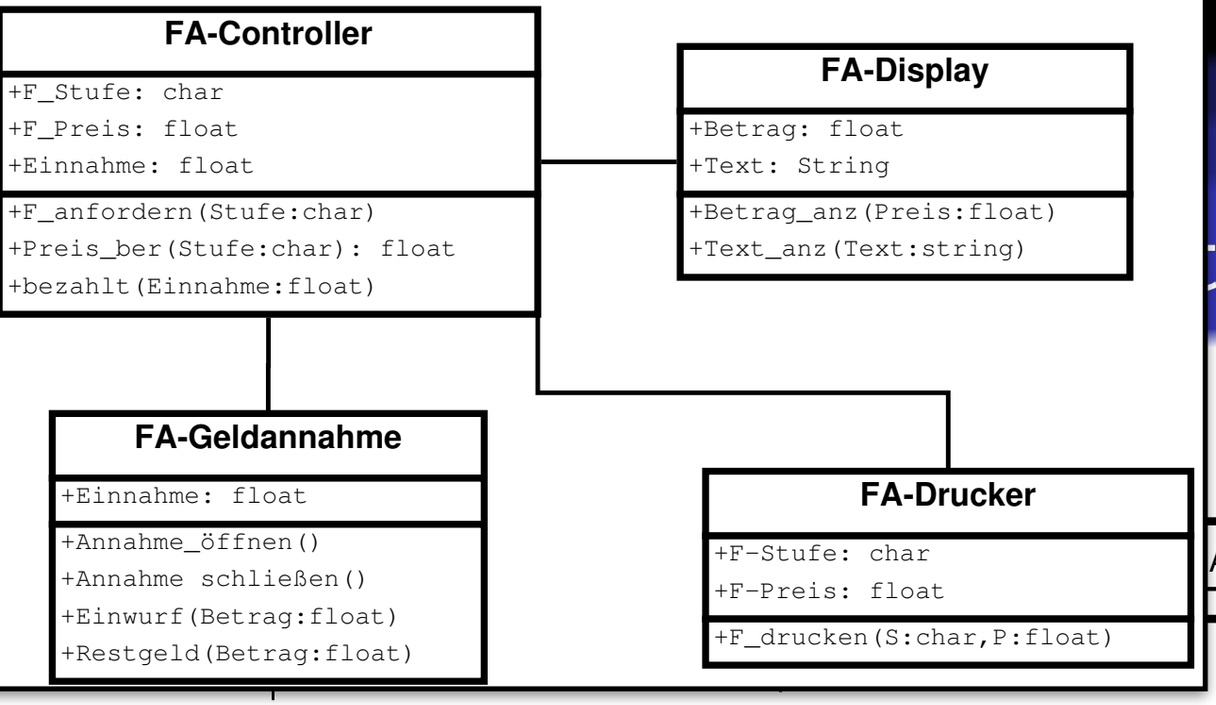


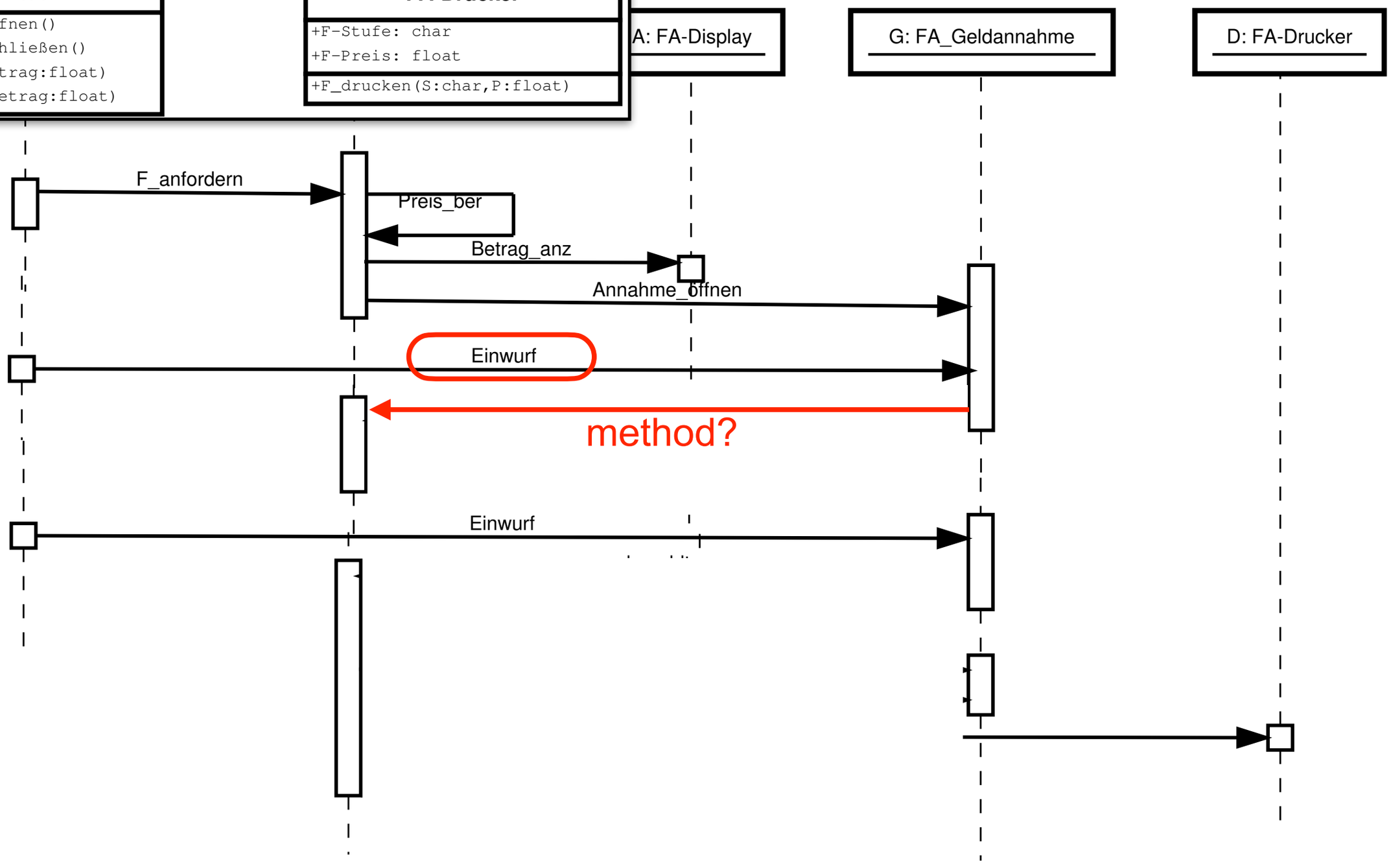
tomat - Sequenzdiagramm



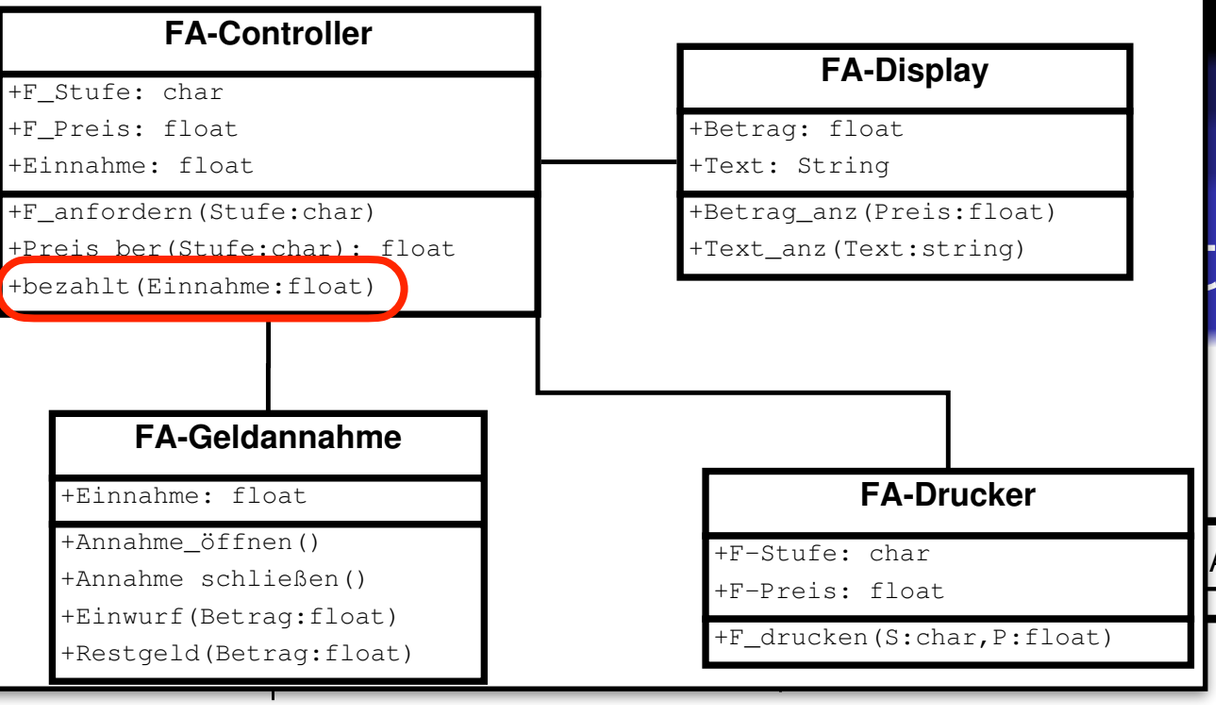


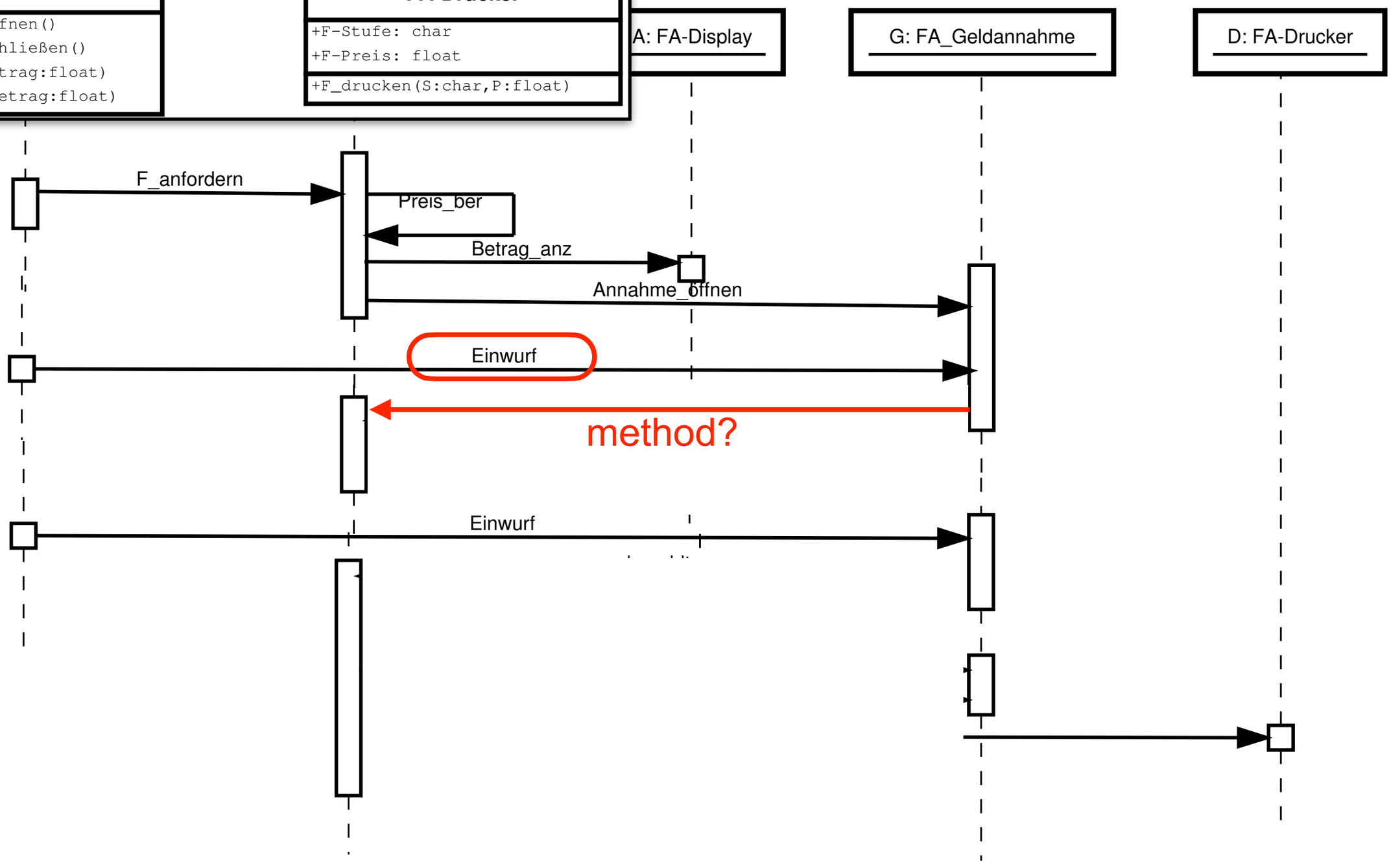
tomat - Sequenzdiagramm



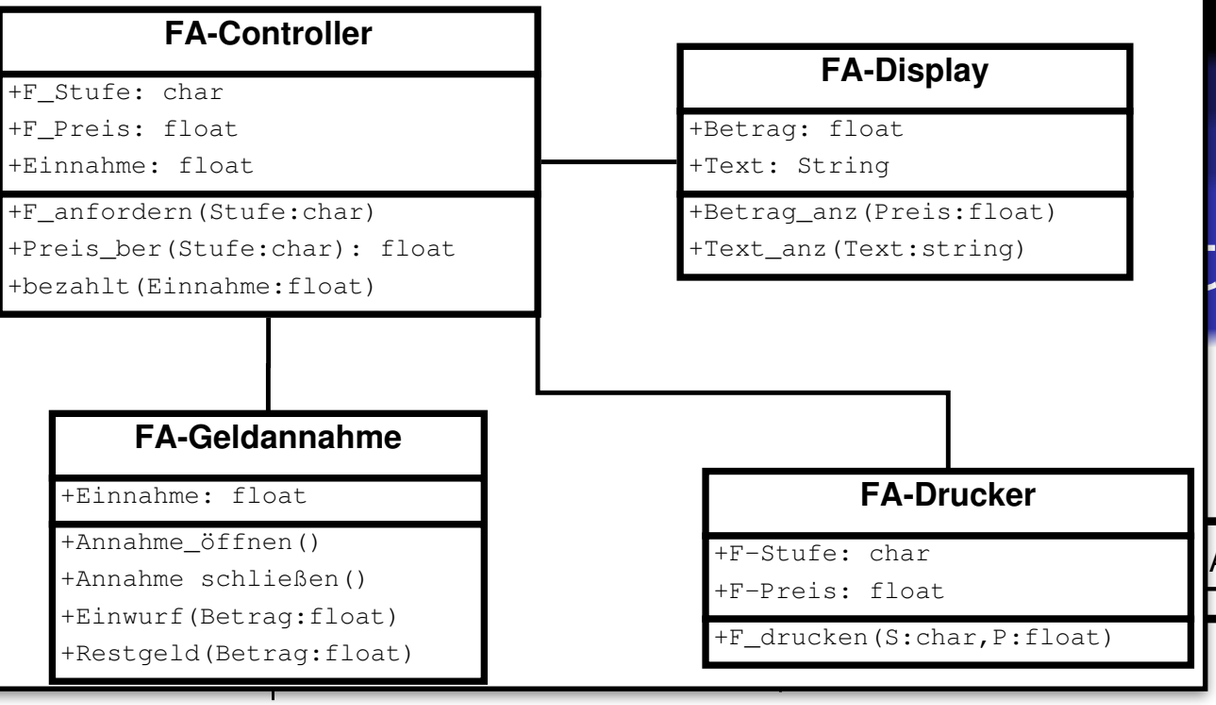


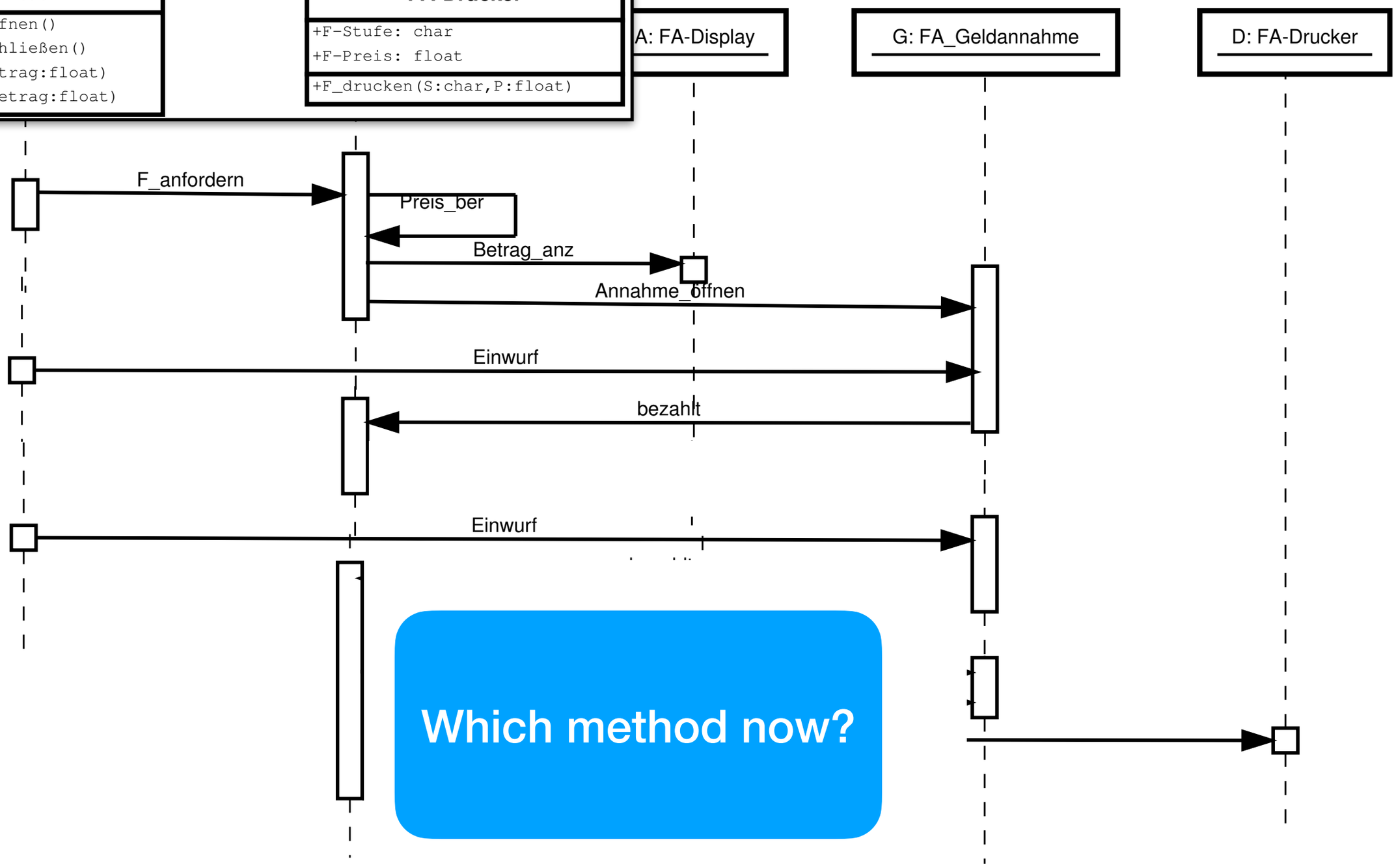
tomat - Sequenzdiagramm





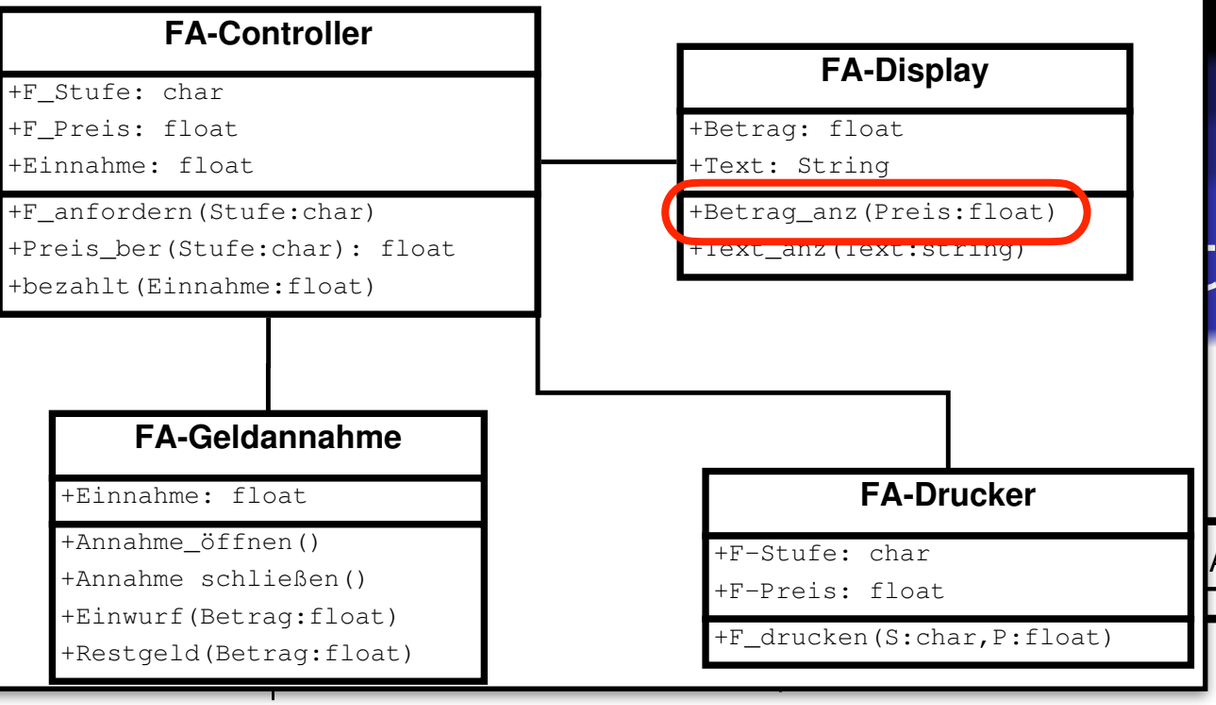
tomat - Sequenzdiagramm

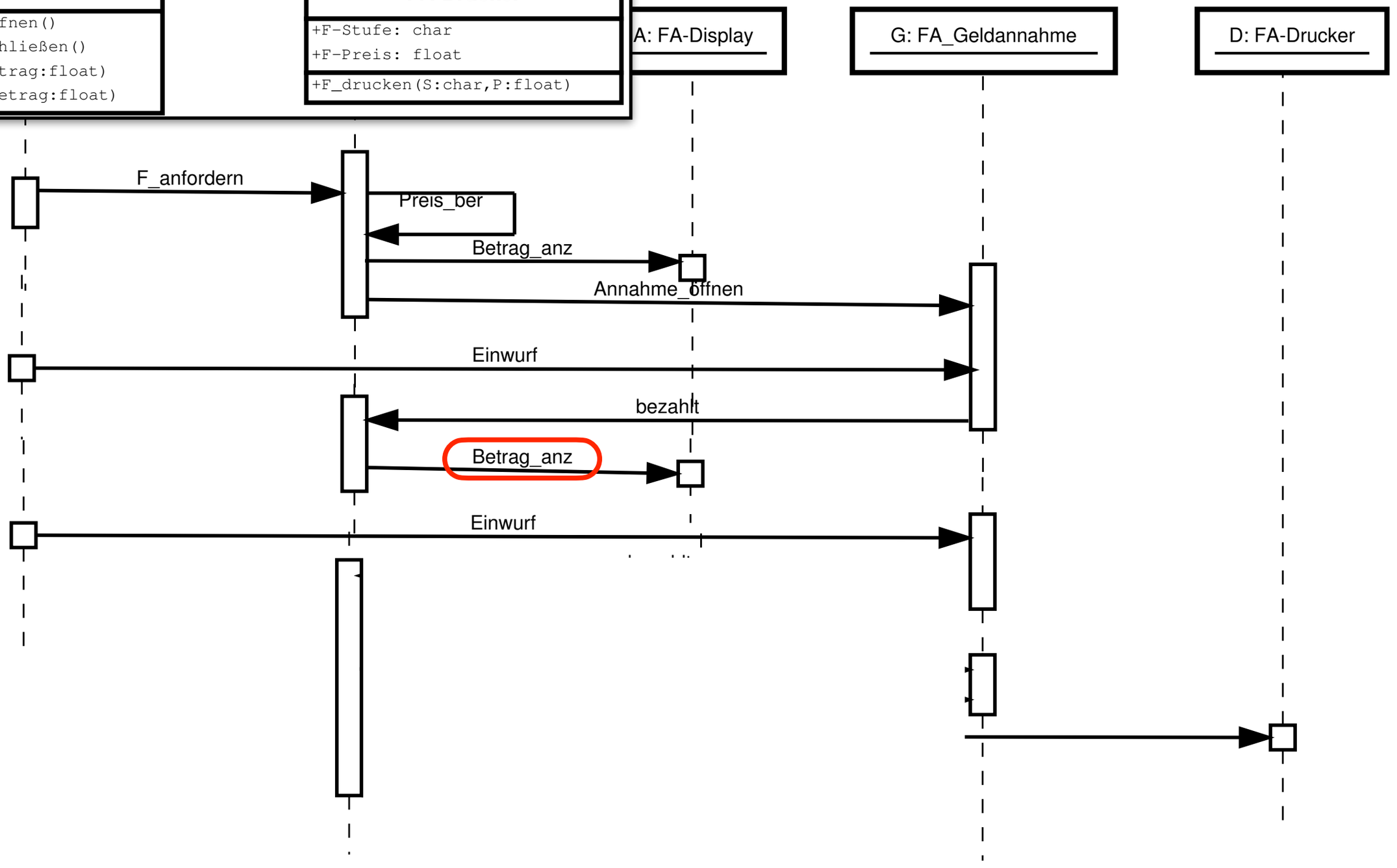




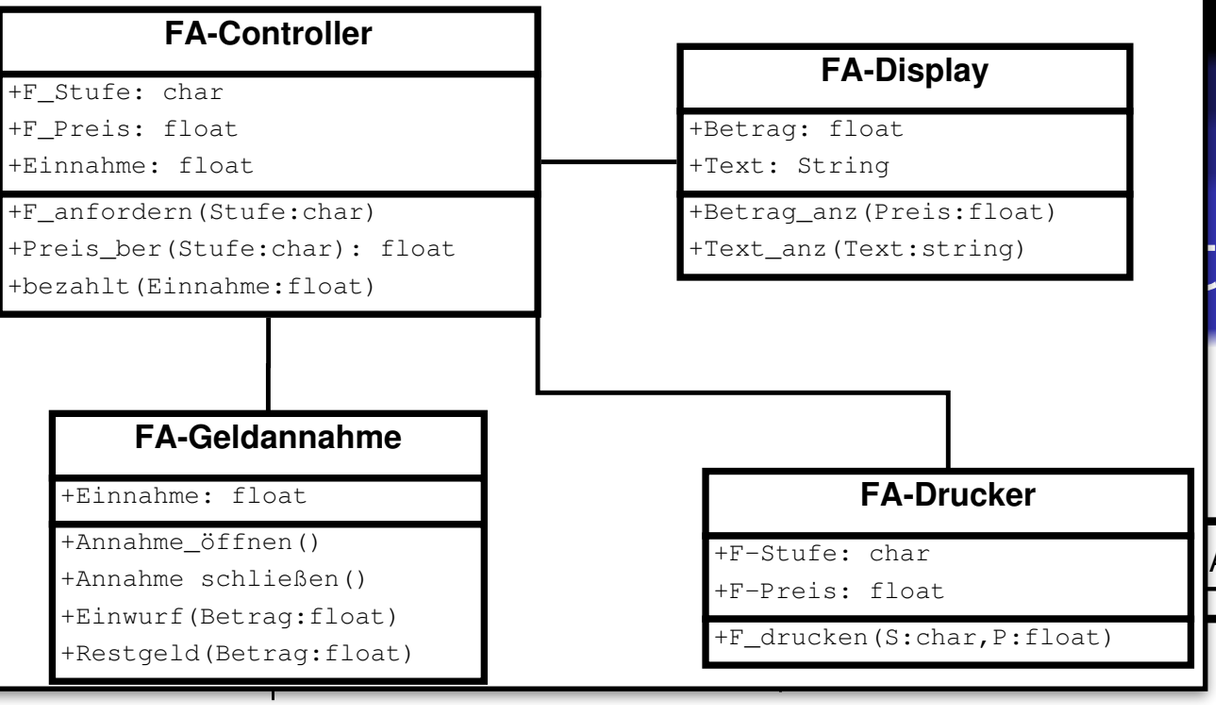
Which method now?

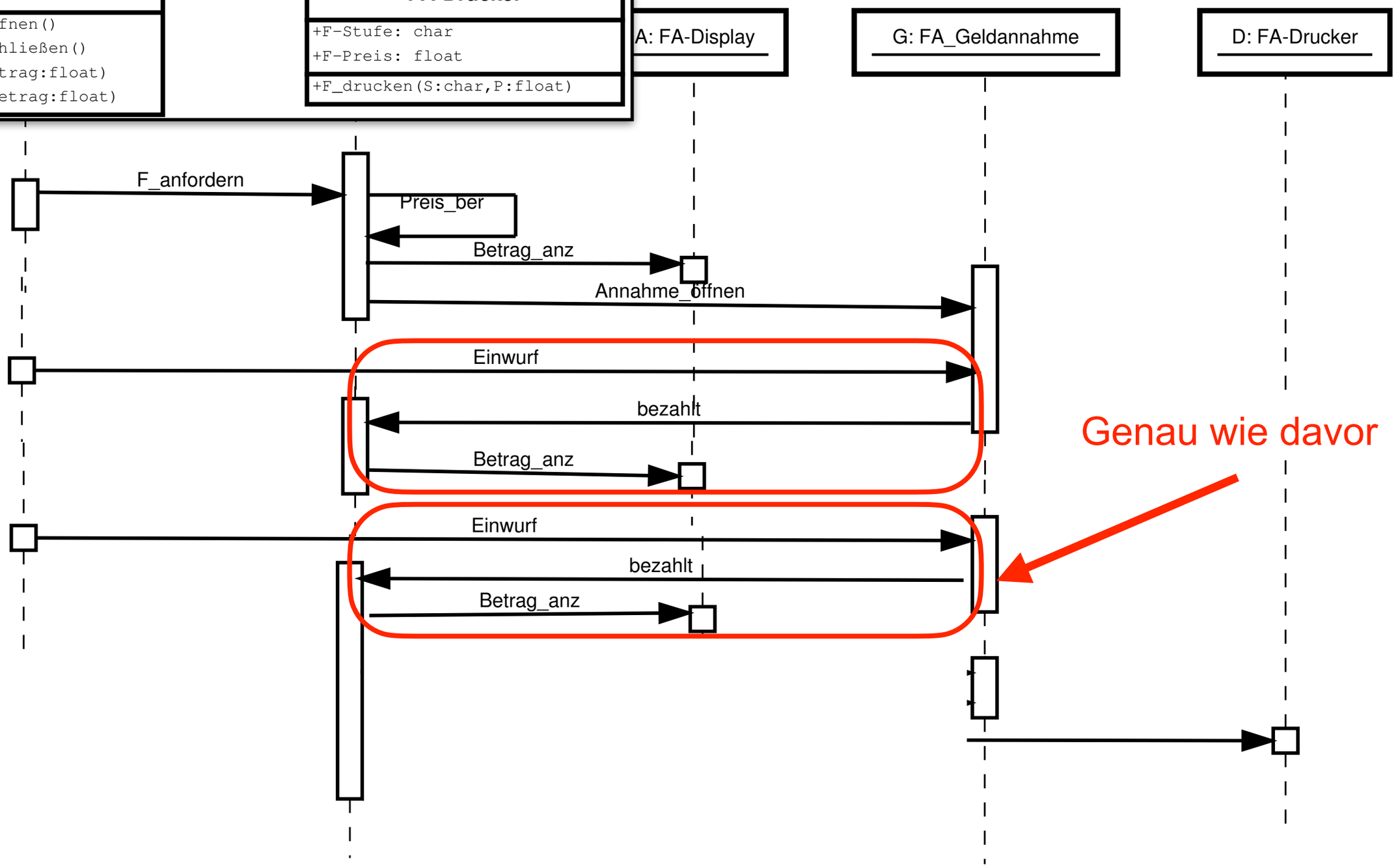
tomat - Sequenzdiagramm



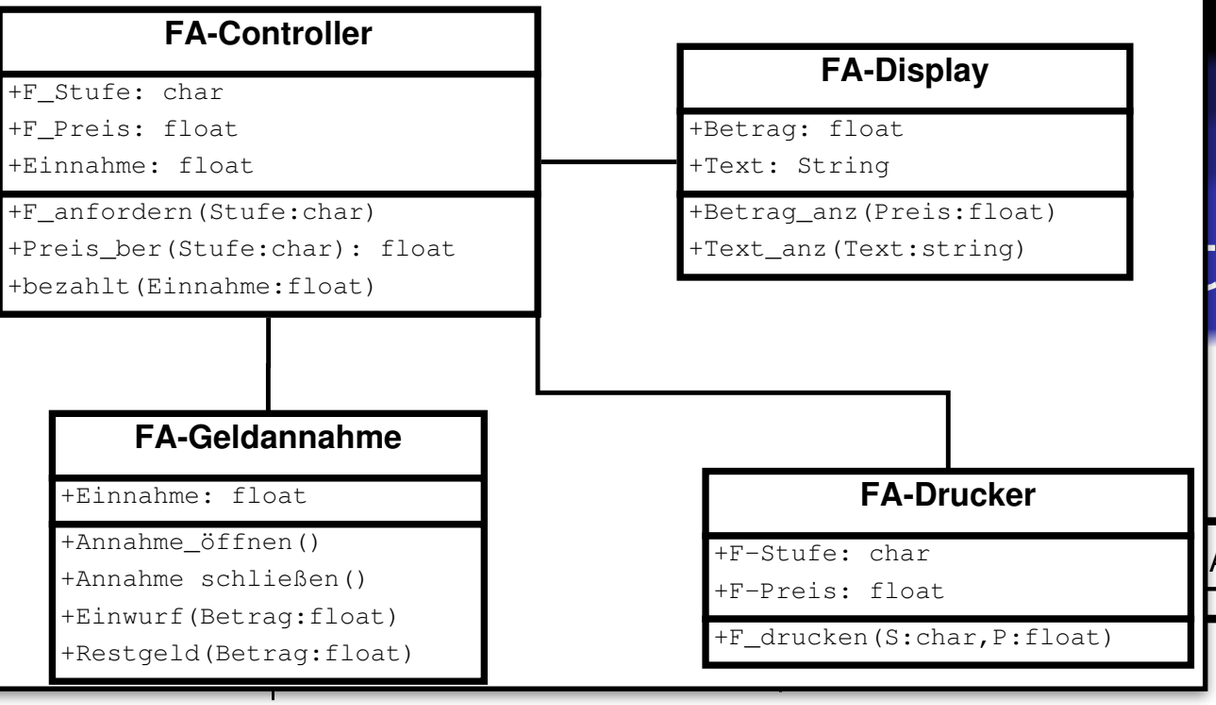


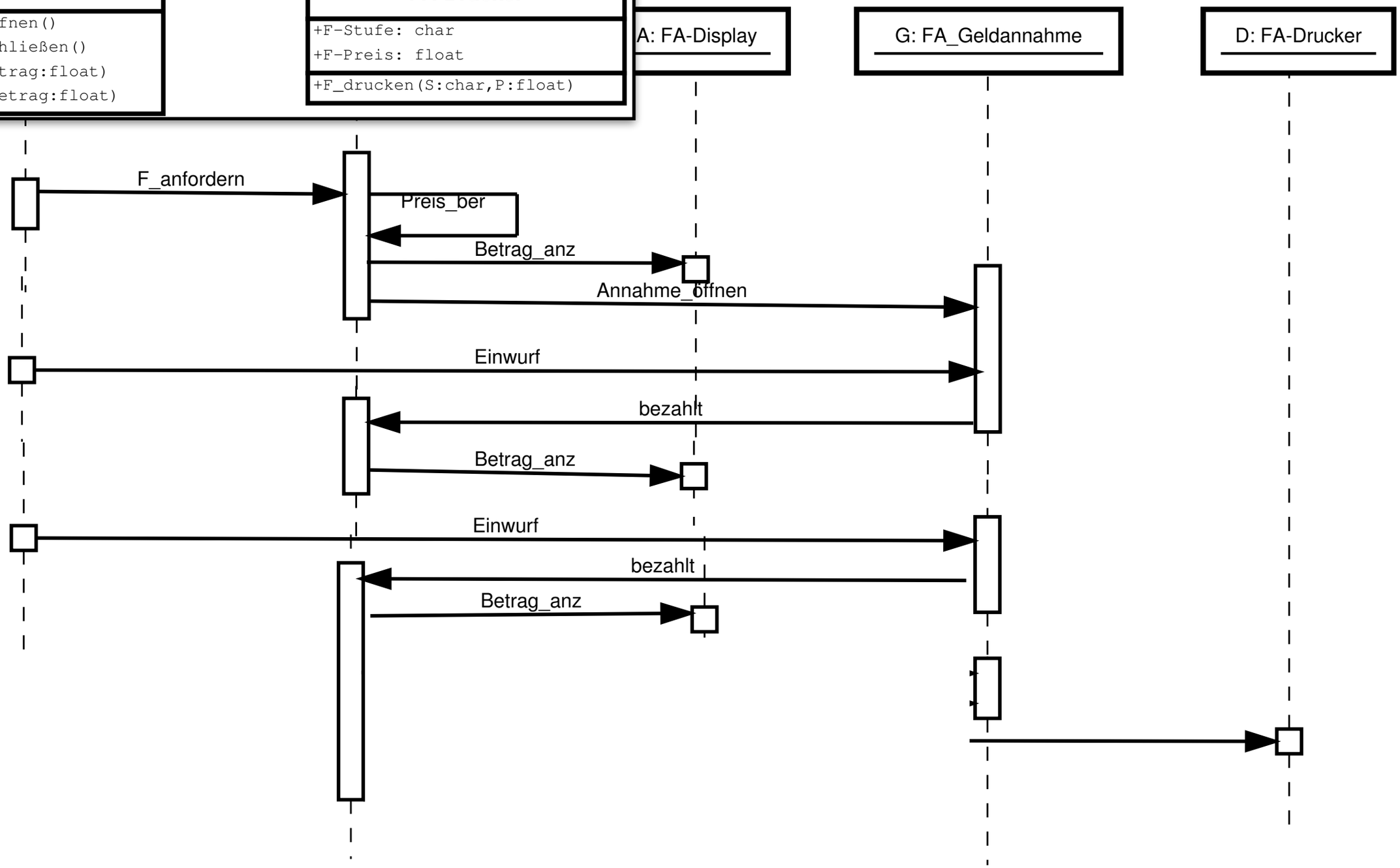
tomat - Sequenzdiagramm



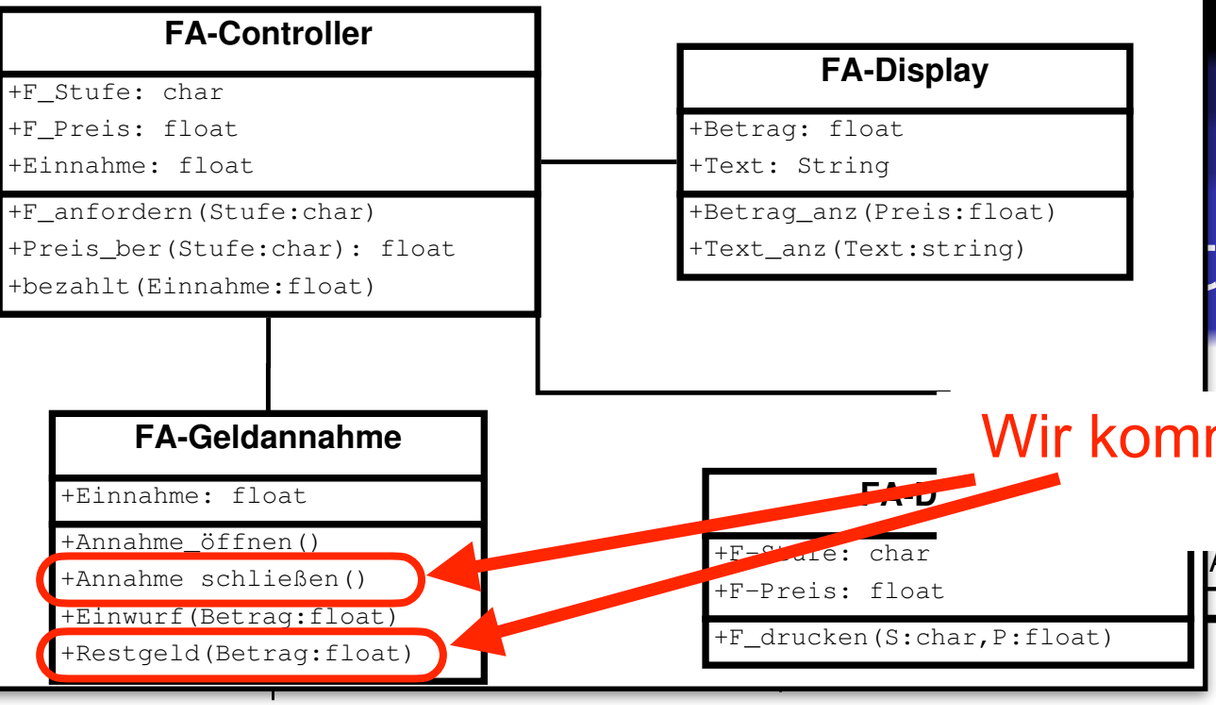


tomat - Sequenzdiagramm

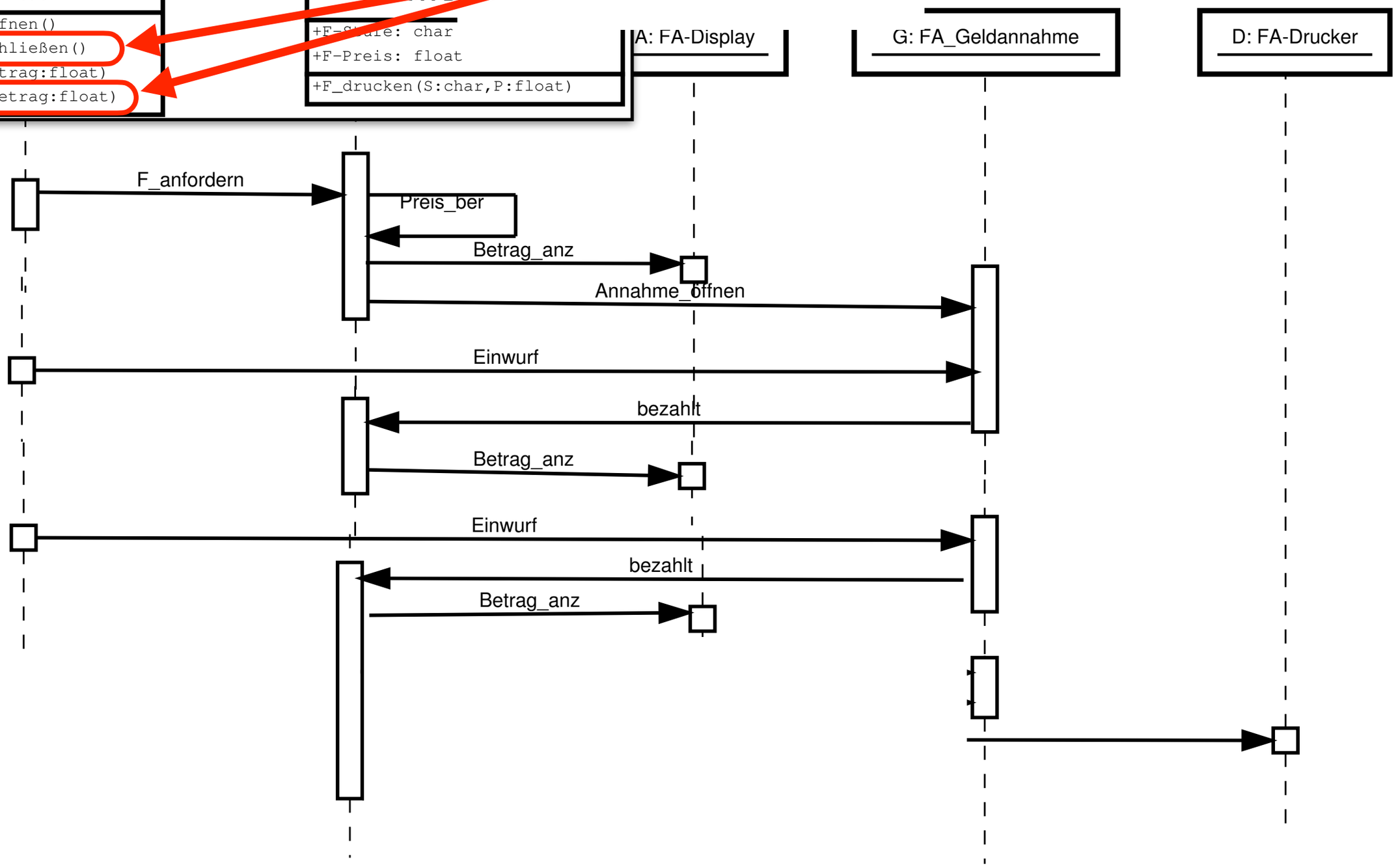




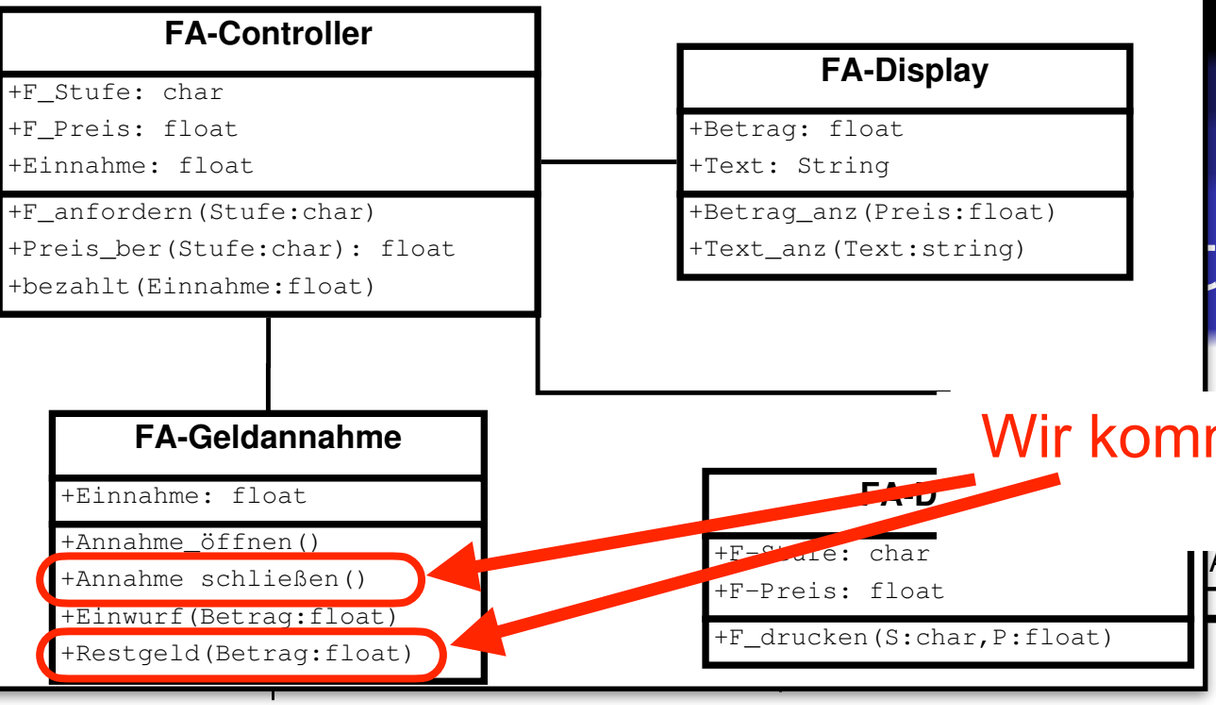
tomat - Sequenzdiagramm



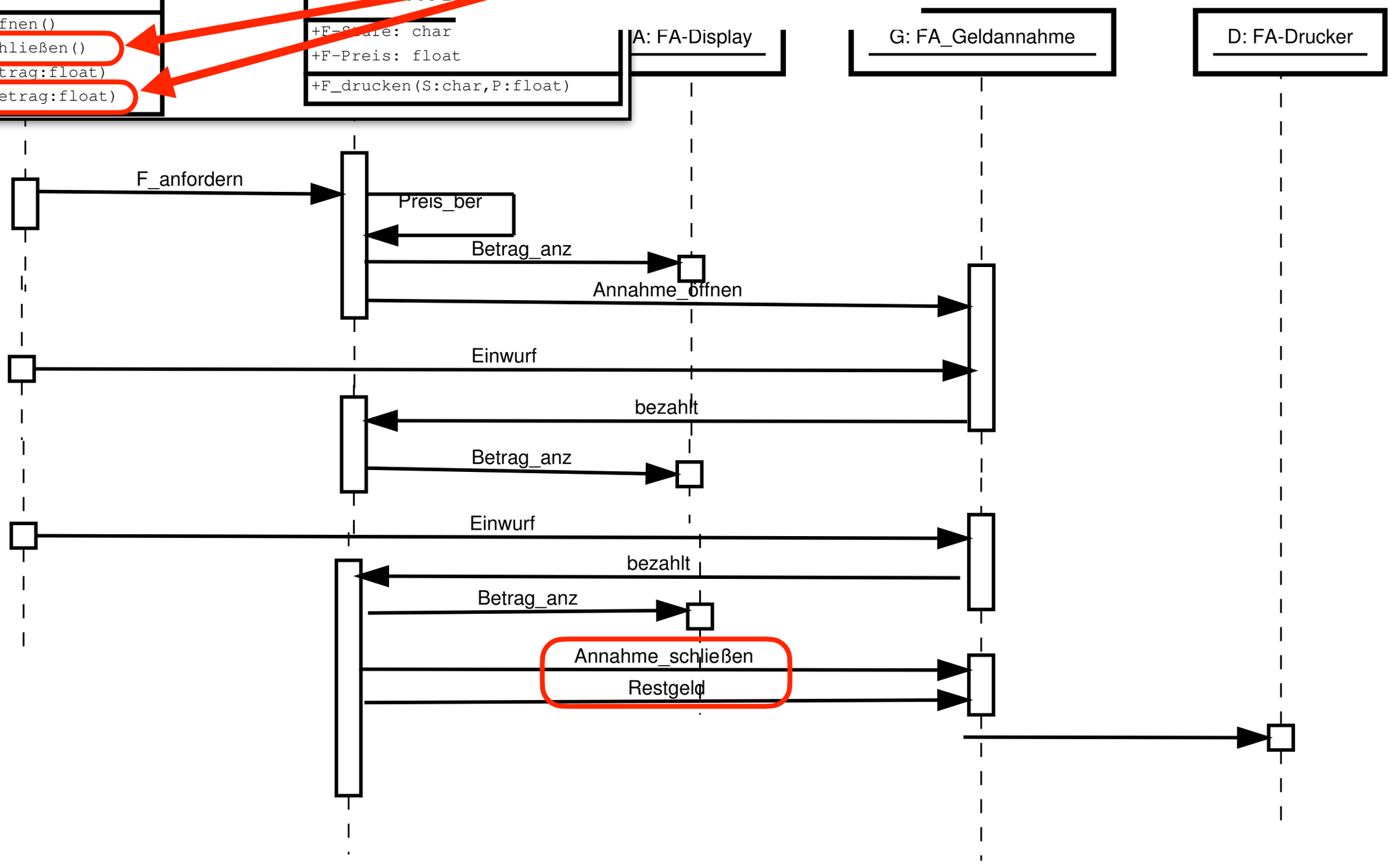
Wir kommen zum Ende..



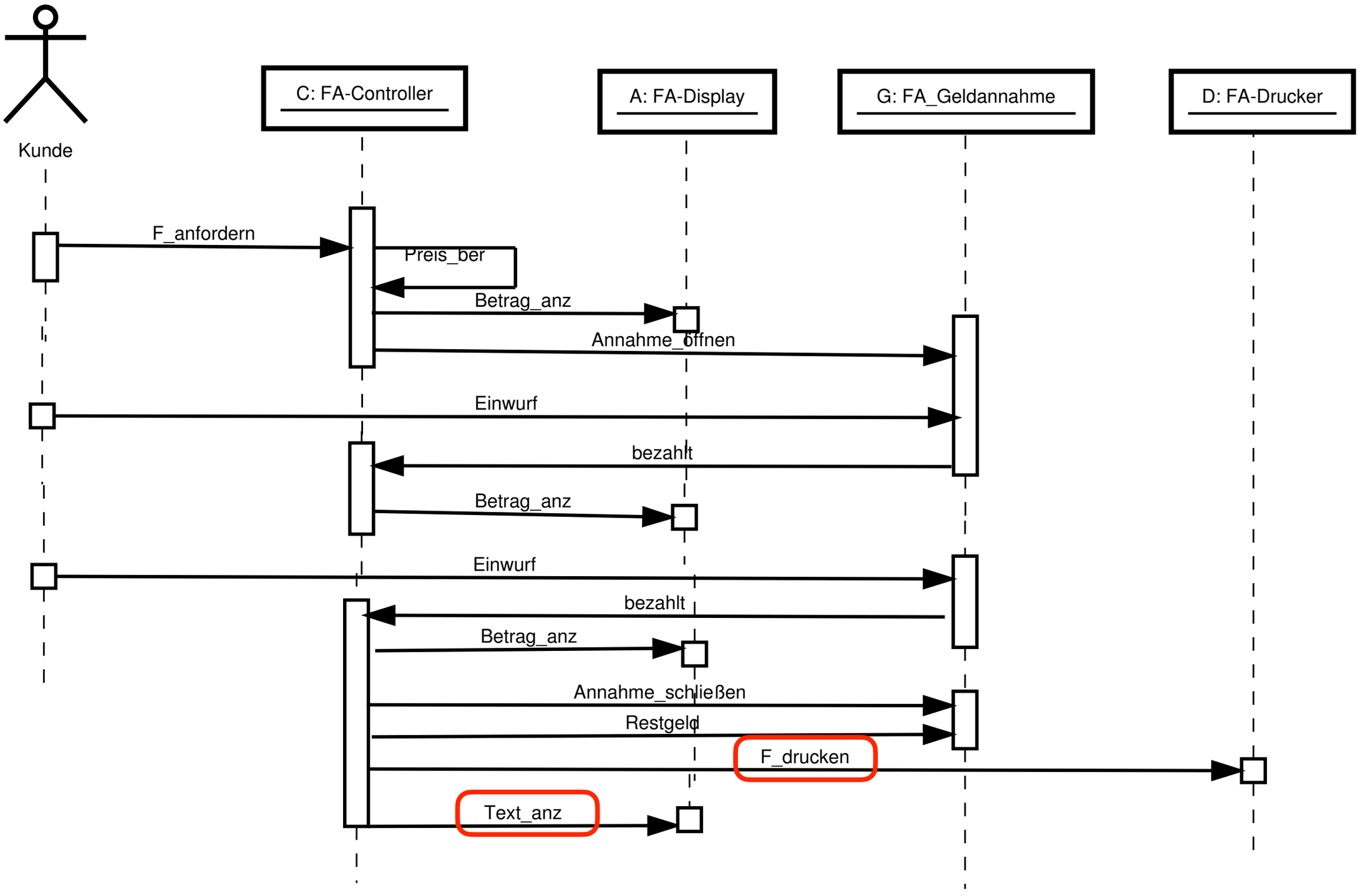
tomat - Sequenzdiagramm



Wir kommen zum Ende..



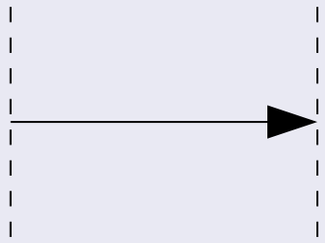
Fallstudie: Fahrkartenautomat - Sequenzdiagramm



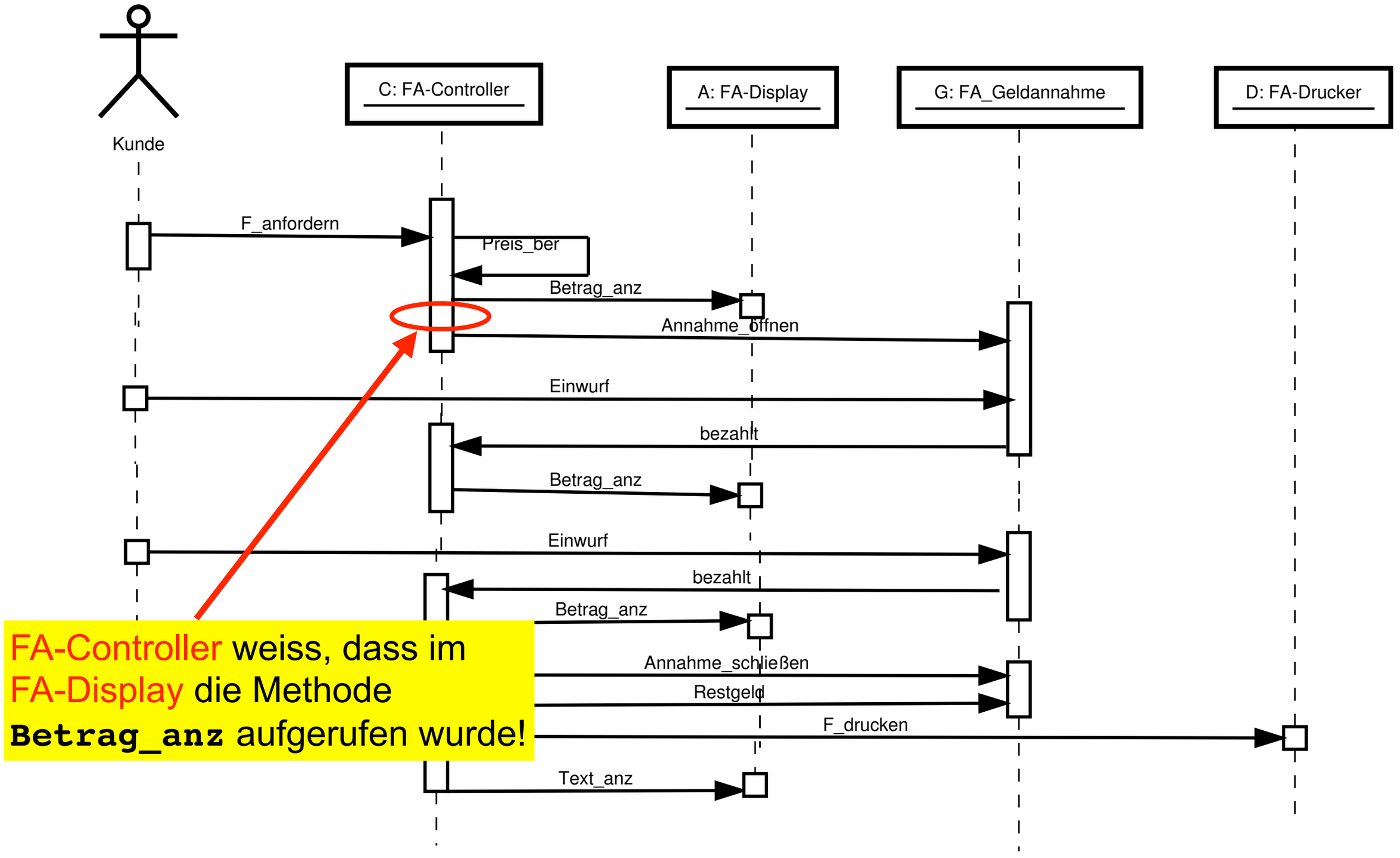
Sequenzdiagramme

Synchrone und asynchrone Nachrichten

Bei synchroner Kommunikation warten Sender und Empfänger aufeinander. Der Sender macht erst dann weiter, wenn er weiß, dass der Empfänger die Nachricht erhalten hat. Sie wird durch eine schwarze ausgefüllte Pfeilspitze dargestellt.



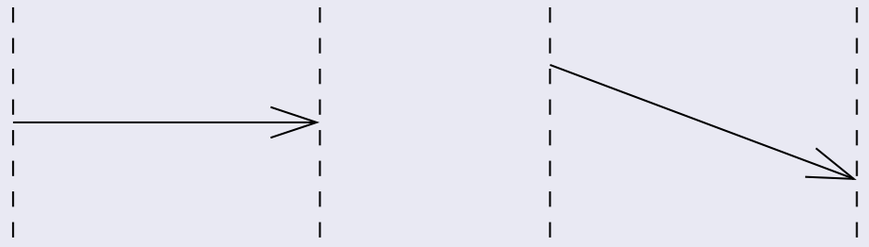
Fallstudie: Fahrkartenautomat - Sequenzdiagramm



Sequenzdiagramme

Synchrone und asynchrone Nachrichten (Fortsetzung)

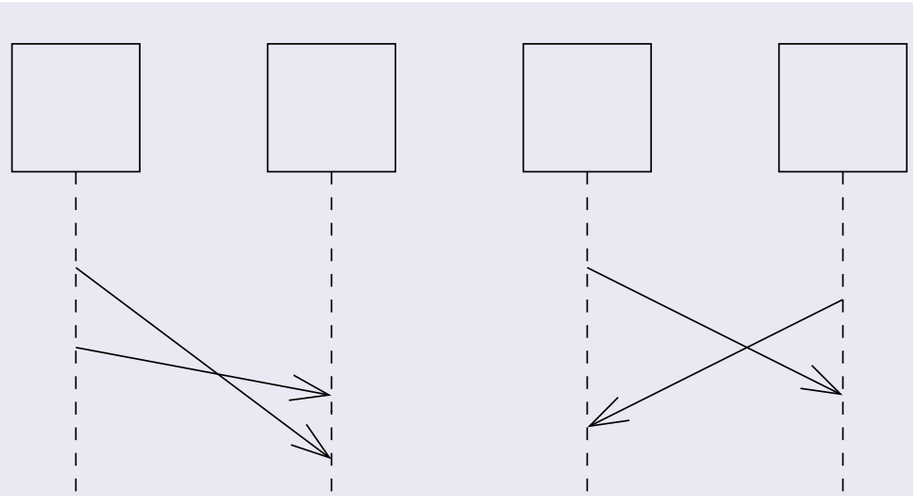
Bei asynchroner Kommunikation wartet der Sender nicht darauf, dass der Empfänger die Nachricht erhalten hat. Er arbeitet einfach weiter. Die Nachricht wird durch eine einfache Pfeilspitze wird dargestellt.



Bei asynchronen Nachrichten kann es zu einer Zeitverzögerung zwischen Sende- und Empfangsereignis kommen, die durch einen geneigten Pfeil beschrieben wird (siehe oben rechts).

Sequenzdiagramme

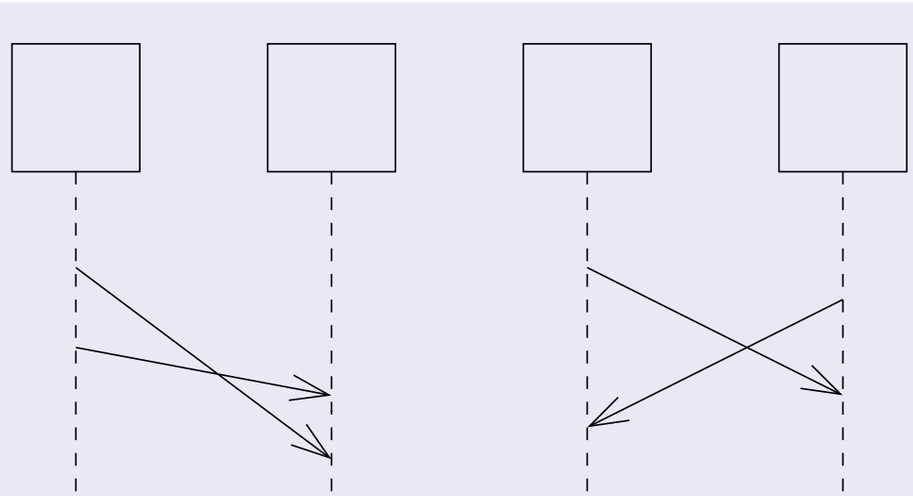
Bei asynchroner Kommunikation (aber *nicht* bei synchroner Kommunikation) kann auch der Fall eintreten, dass sich Nachrichten überholen oder kreuzen.



Das Überholen der Nachrichten (oben links) ist nur dann nicht möglich, wenn man explizit einen FIFO-Kanal fordert.

Sequenzdiagramme

Bei asynchroner Kommunikation (aber *nicht* bei synchroner Kommunikation) kann auch der Fall eintreten, dass sich Nachrichten überholen oder kreuzen.

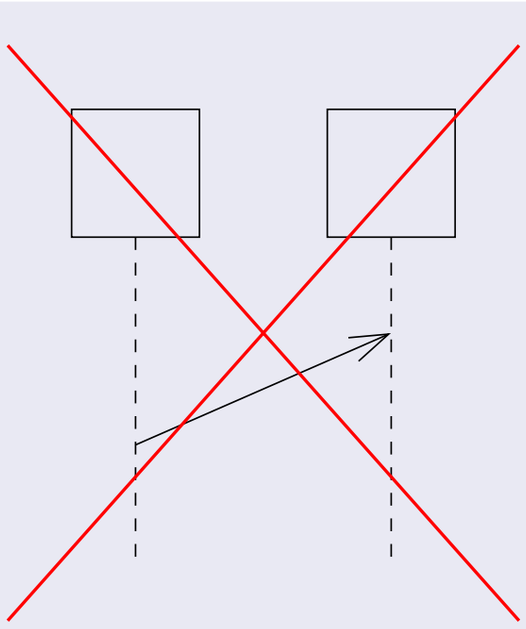


Das Überholen der Nachrichten (oben links) ist nur dann nicht möglich, wenn man explizit einen FIFO-Kanal fordert.

First **I**n **F**irst **O**ut

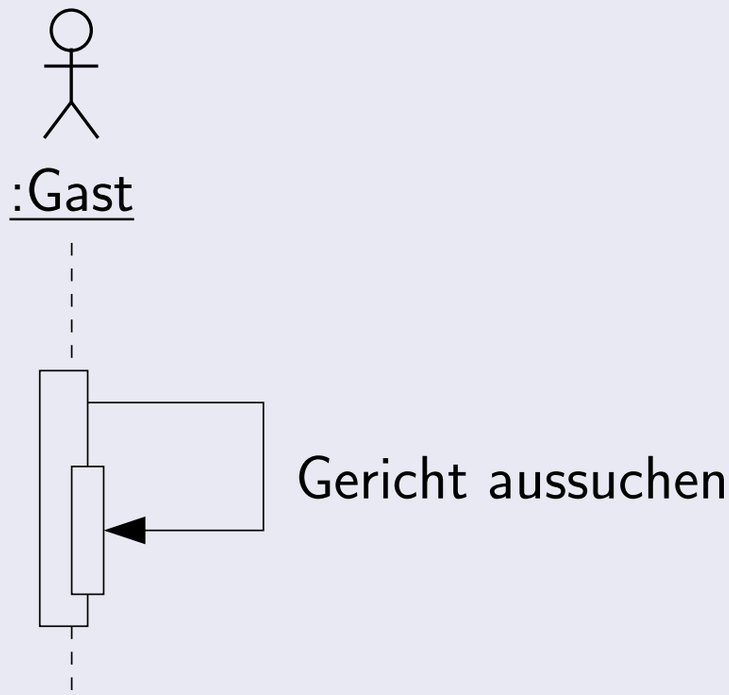
Sequenzdiagramme

Was jedoch nicht möglich ist, ist eine Nachricht, die “rückwärts” in der Zeit läuft und vor dem Senden ankommt.



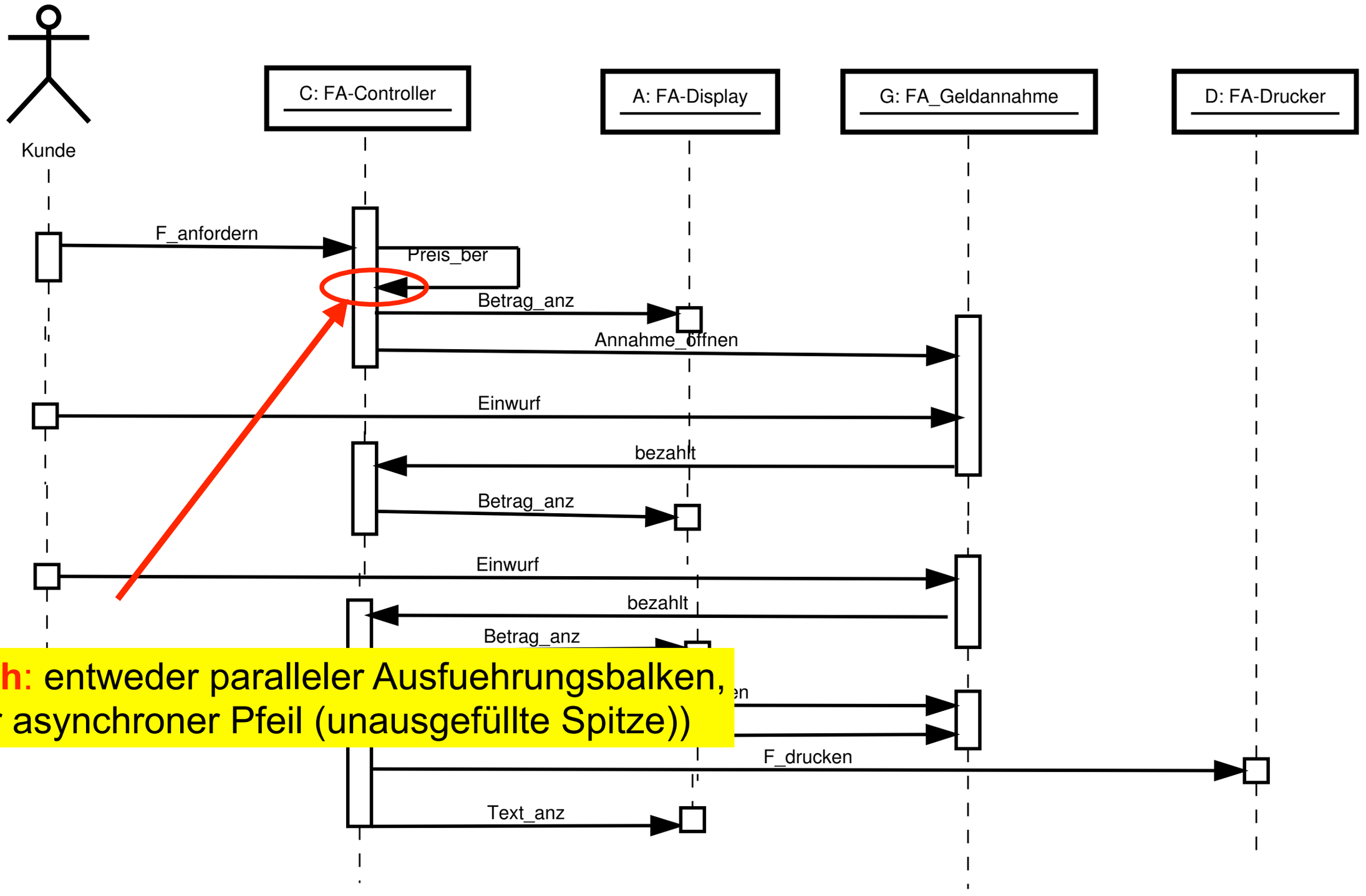
Sequenzdiagramme

Es ist jedoch möglich, eine Nachricht an sich selbst zu schicken.



Bei einer synchronen Nachricht sollte man dabei parallele Ausführungsbalken verwenden, da das Sende- und Empfangsereignis parallel stattfinden müssen.

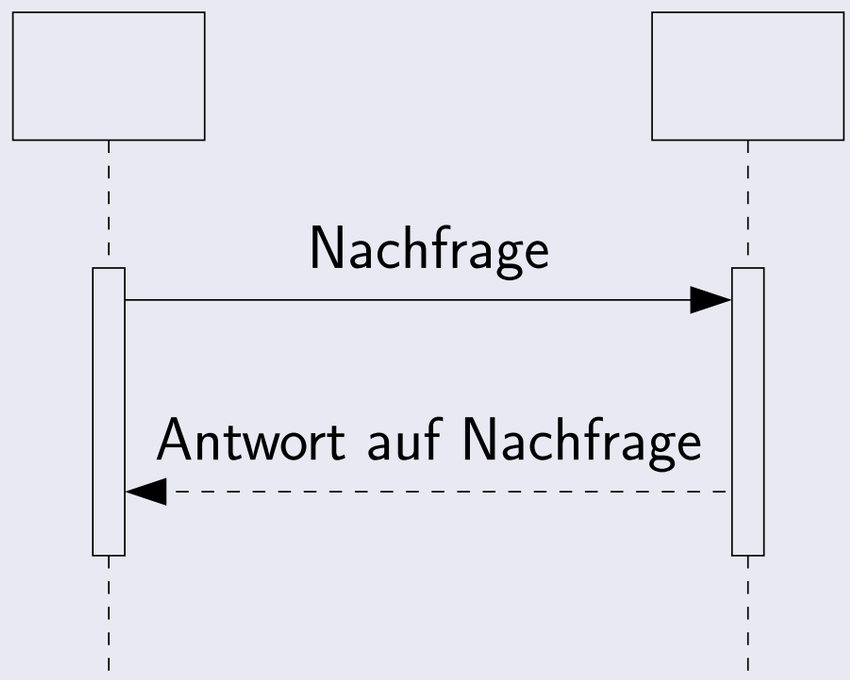
Fallstudie: Fahrkartenautomat - Sequenzdiagramm



falsch: entweder paralleler Ausfuehrungsbalken, (oder asynchroner Pfeil (unausgefüllte Spitze))

Sequenzdiagramme

Nachrichten, die als Antworten auf frühere Nachrichten gedacht sind, werden durch gestrichelte Pfeile notiert. Dabei sollte der Name der ursprünglichen Nachricht wiederholt werden.



Sequenzdiagramme

Wir machen uns nun Gedanken über die Reihenfolge der Ereignisse in einem Sequenzdiagramm. Dazu ist es nützlich sich klarzumachen, was eine (partielle) Ordnung ist.

Partielle Ordnung

Gegen sei eine Menge X. Eine partielle Ordnung auf X ist eine Relation $R \subseteq X \times X$ mit folgenden Eigenschaften:

- Reflexivität: für jedes $x \in X$ gilt $(x, x) \in R$
- Transitivität: aus $(x_1, x_2) \in R$ und $(x_2, x_3) \in R$ für $x_1, x_2, x_3 \in X$ folgt $(x_1, x_3) \in R$.
- Antisymmetrie: aus $(x_1, x_2) \in R$ und $(x_2, x_1) \in R$ für $x_1, x_2 \in X$ folgt $x_1 = x_2$.

Sequenzdiagramme

Weitere Bezeichnungen:

- Partielle Ordnung bezeichnet man oft mit dem Zeichen $\leq$ und notiert in Infix-Schreibweise ($x_1 \leq x_2$ statt $(x_1, x_2) \in R$).
- Wir schreiben $x_1 < x_2$, falls $x_1 \leq x_2$ und $x_1 \neq x_2$.

Vorsicht: Die Relation $<$ ist selbst keine partielle Ordnung.

Sequenzdiagramme

Totale Ordnung

Eine partielle Ordnung $\leq$ heißt total, wenn für zwei beliebige Elemente $x_1, x_2 \in X$ immer $x_1 \leq x_2$ oder $x_2 \leq x_1$ gilt.

Beispiele:

- Die $\leq$-Relation auf den ganzen Zahlen $\mathbb{Z}$ ist eine totale Ordnung.

Sequenzdiagramme

Totale Ordnung

Eine partielle Ordnung $\leq$ heißt total, wenn für zwei beliebige Elemente $x_1, x_2 \in X$ immer $x_1 \leq x_2$ oder $x_2 \leq x_1$ gilt.

Beispiele:

- Die $\leq$-Relation auf den ganzen Zahlen $\mathbb{Z}$ ist eine totale Ordnung.
- Ein typisches Beispiel für eine partielle Ordnung, die nicht total ist, ist die Inklusionsordnung $\subseteq$ auf Mengen.

Sequenzdiagramme

Reflexiv-transitive Hülle

Für eine gegebene Relation $R \subseteq E \times E$ beschreiben wir deren reflexiv-transitive Hülle R^*. Man erhält sie aus R, indem man

- alle Paare (e, e) mit $e \in E$ zu R hinzufügt *und*
- bei $(e_1, e_2), (e_2, e_3) \in R$ auch (e_1, e_3) zu R hinzufügt. Dieser Prozess wird so lange wiederholt, bis keine neuen Paare mehr hinzukommen.

- Die reflexiv-transitiv Hülle R^* einer Relation R ist immer reflexiv und transitiv. Sie ist jedoch nicht immer antisymmetrisch und daher nicht notwendigerweise eine Ordnung.
- Die Relation R^* ist die kleinste reflexive und transitive Relation, die R enthält.

Sequenzdiagramme

Beispiel:

Die reflexiv-transitive Hülle von

$$R = \{(1,2),(2,3),(3,4),(3,5)\}$$

ist

$$R^* = \{(1,2),(2,3),(3,4),(3,5),$$

???

Poll:
Wieviele Paare kommen hinzu?

Sequenzdiagramme

Beispiel:

Die reflexiv-transitive Hülle von

$$R = \{(1,2),(2,3),(3,4),(3,5)\}$$

ist

$$\begin{aligned} R^* \quad = \quad & \{(1,2),(2,3),(3,4),(3,5), \\ & (1,1),(2,2),(3,3),(4,4),(5,5), \\ & (1,3),(1,4),(1,5),(2,4),(2,5)\} \end{aligned}$$

Sequenzdiagramme

Beispiel:

Die reflexiv-transitive Hülle von

$$R = \{(1,1),(3,3)\}$$

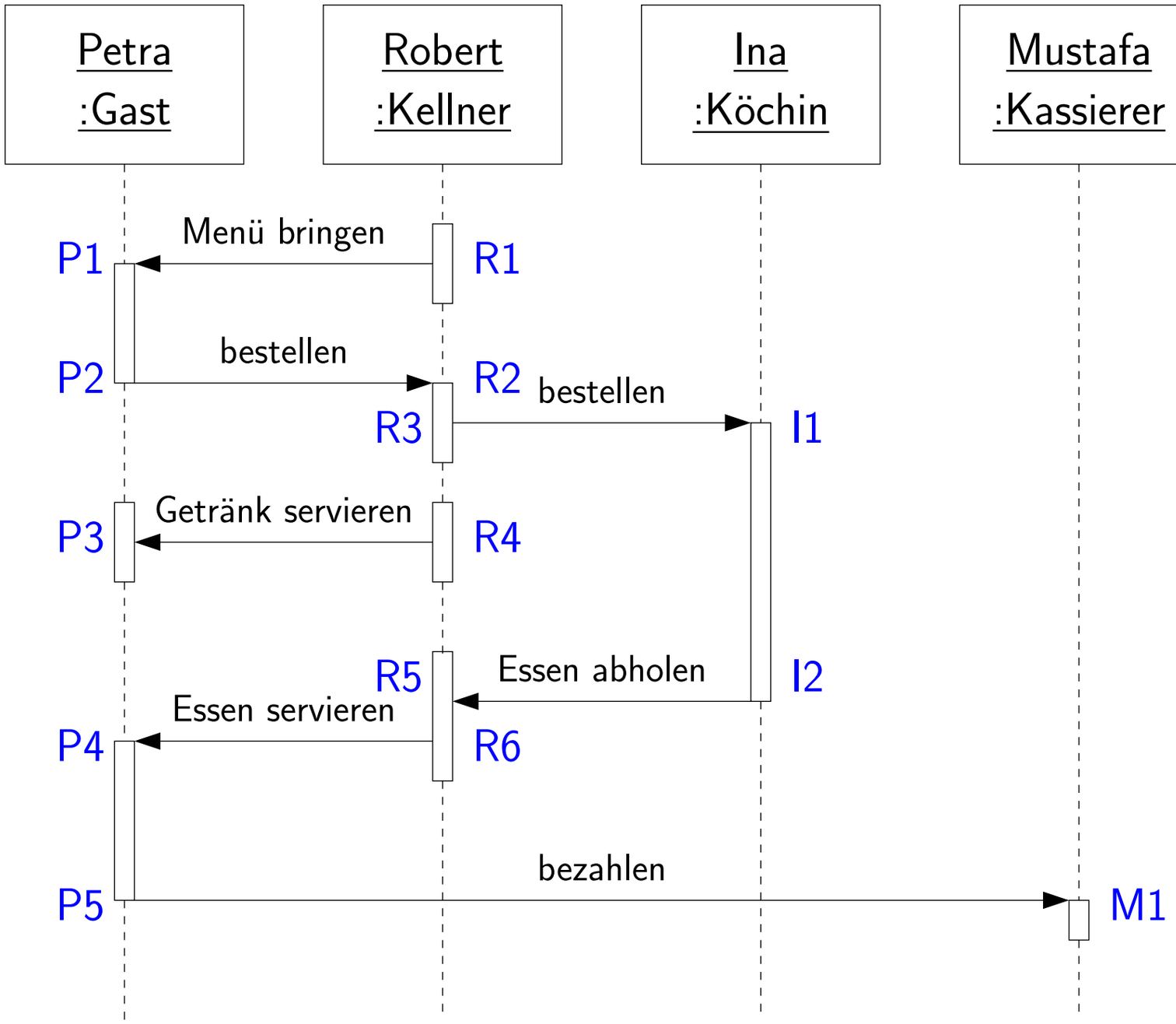
Sequenzdiagramme

Die Elemente von Sequenzdiagrammen, die nun geordnet werden, sind die Ereignisse, d.h., Sende- und Empfangsereignisse.

In den Sequenzdiagrammen, wie wir sie bisher kennengelernt haben, sind die Ereignisse immer total geordnet, nach folgendem Schema:

- (Sendeereignis der Nachricht N) $<$ (Empfangsereignis von N)
- Die restlichen Ereignisse sind entsprechend dem zeitlichen Verlauf geordnet.

Sequenzdiagramme

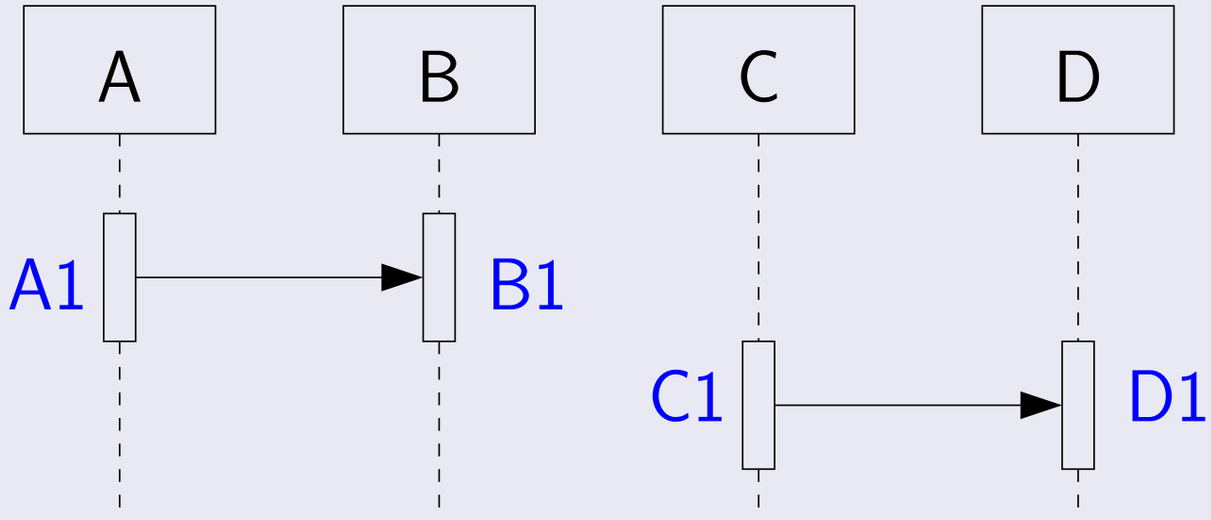


$$
\begin{aligned}
&R1 < P1 < P2 < \\
&R2 < R3 < I1 < \\
&R4 < P3 < I2 < \\
&R5 < R6 < P4 < \\
&P5 < M1
\end{aligned}
$$

Sequenzdiagramme

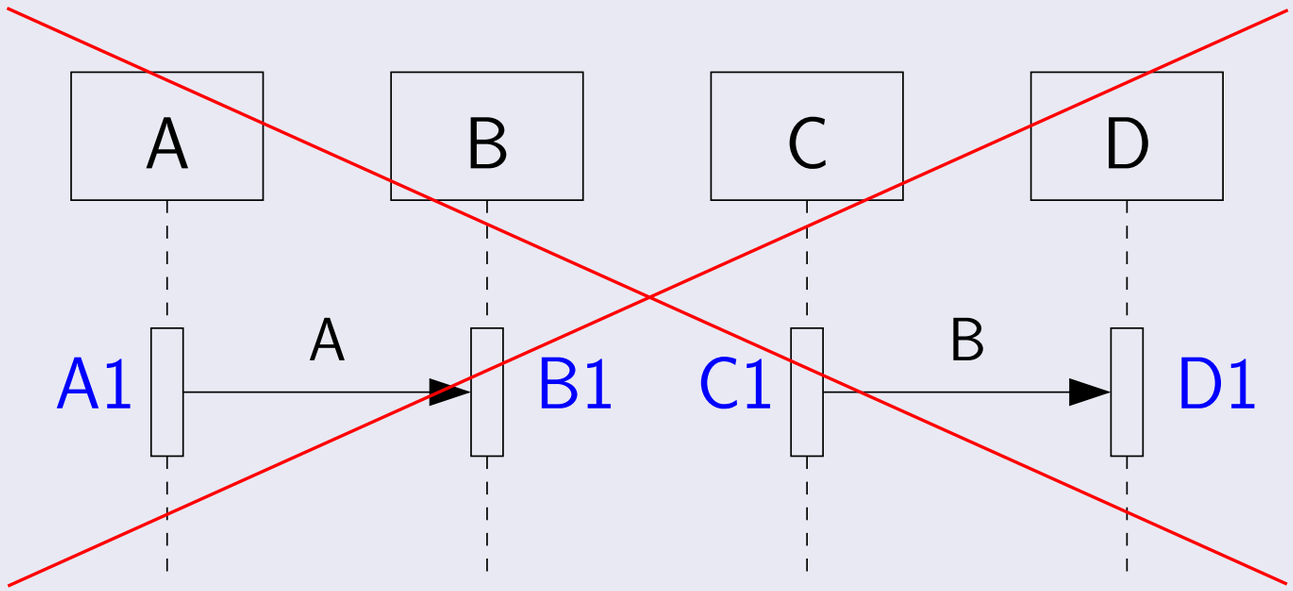
In einem einfachen Sequenzdiagramm (Erweiterungen werden noch besprochen) werden daher die Ereignisse immer in eine Reihenfolge gebracht. Die Ordnung ist daher total. Im folgenden Diagramm gilt beispielsweise:

$$A1 < B1 < C1 < D1$$



Sequenzdiagramme

Nicht so günstig sind Diagramme, aus denen die Reihenfolge von Nachrichten nicht klar hervorgeht:



Insbesondere ist die Darstellung echter Parallelität in Sequenzdiagrammen nicht vorgesehen.

Sequenzdiagramme

Äquivalenz von Sequenzdiagrammen

Zwei Sequenzdiagramme sind äquivalent, wenn sie die gleichen Ereignisse enthalten und die Reihenfolge der Ereignisse identisch ist.

Dabei können die Diagramme durchaus verschieden gezeichnet sein (andere Reihenfolge der Kommunikationspartner, verschiedene Abstände zwischen den Ereignissen, ...).

Sequenzdiagramme

Bisher haben wir gesehen, wie man mit einem Sequenzdiagramm *einen* möglichen Ablauf beschreiben kann. In manchen Fällen möchte man jedoch *mehrere* (vielleicht sogar alle) Abläufe beschreiben.

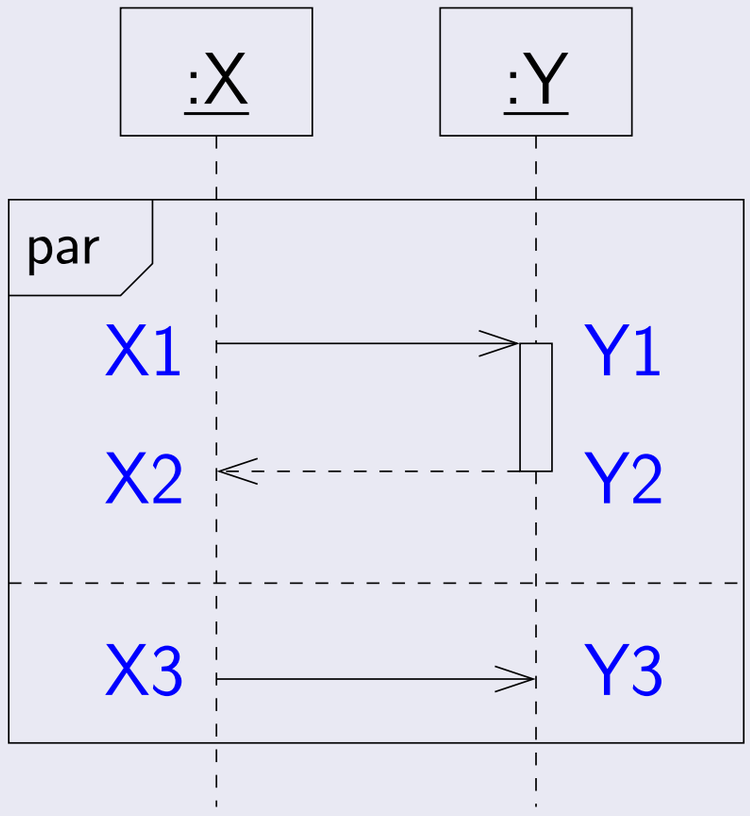
Dazu gibt es die Möglichkeit, kombinierte Fragmente zu verwenden, bei denen mehrere (Interaktions-)Operanden (= Teil-Sequenzdiagramme) mit Hilfe von Interaktions-Operatoren zusammengesetzt werden.

Wir betrachten im folgenden einige dieser Interaktions-Operatoren.

Sequenzdiagramme

Interaktionsoperator par (Parallelität)

Hier sind die Operanden in beliebiger Reihenfolge ausführbar. Die Reihenfolge der Ereignisse *in* den Operanden muss aber gewahrt werden, ansonsten gibt es keine Bedingungen.



Dabei wird der Operator par links oben in der Ecke angegeben.

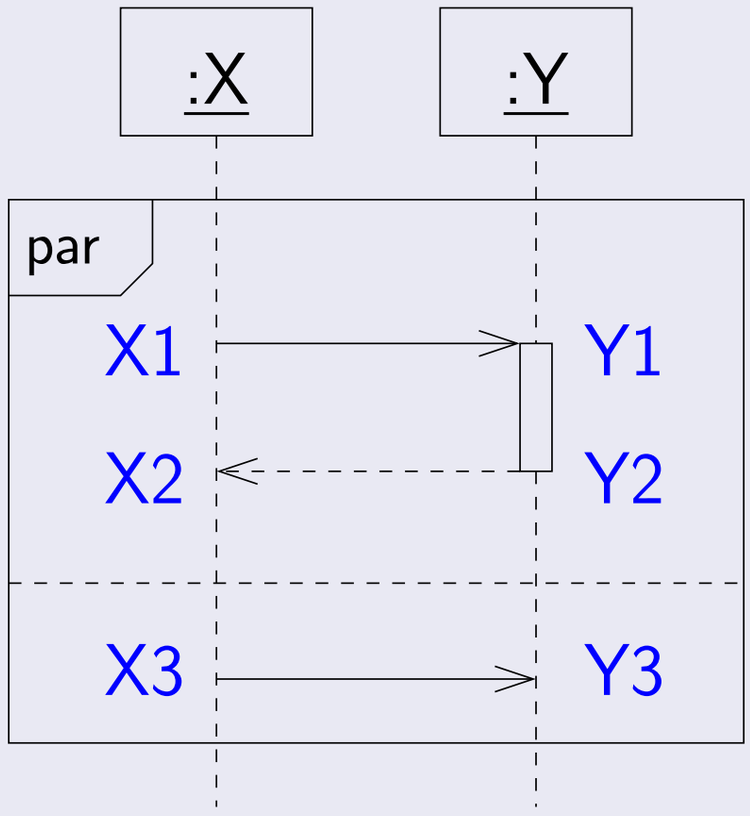
Eine gestrichelte waagrechte Linie trennt die verschiedenen Operanden.

Sequenzdiagramme

Beispiel einer erlaubte Ausführungs-reihenfolgen angeben?

Interaktionsoperator par (

Hier sind die Operanden i
Reihenfolge der Ereignisse
werden, ansonsten gibt es keine Bedingungen.



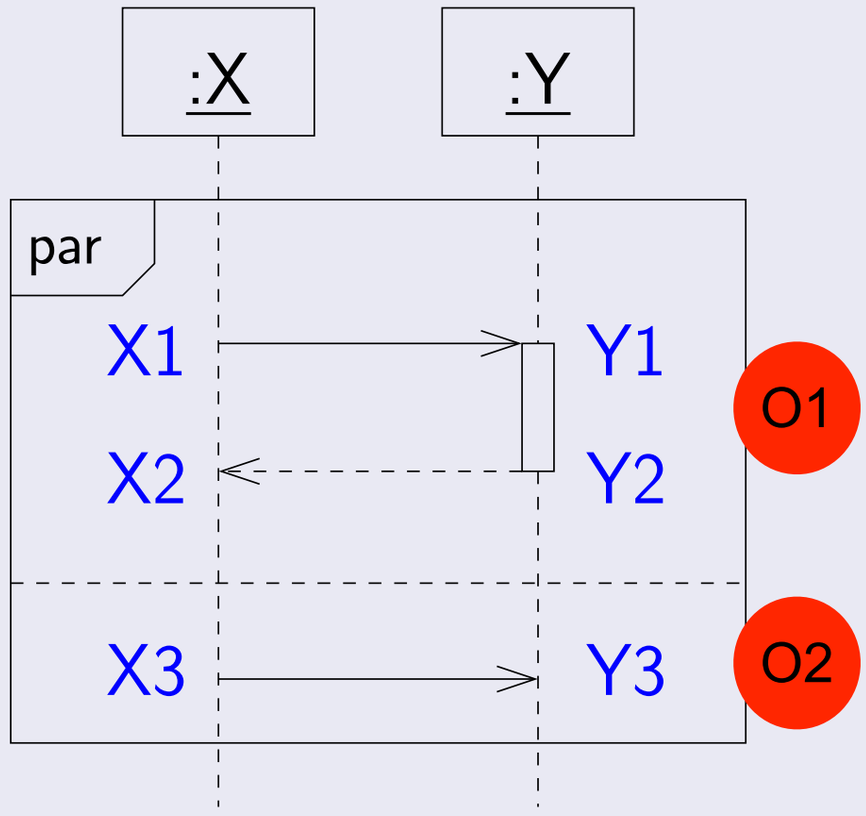
Dabei wird der Operator par links oben in der Ecke angegeben.

Eine gestrichelte waagrechte Linie trennt die verschiedenen Operanden.

Sequenzdiagramme

Interaktionsoperator par (

Hier sind die Operanden i
Reihenfolge der Ereignisse
werden, ansonsten gibt es keine Bedingungen.



Dabei wird der Operator par links oben in der Ecke angegeben.

Eine gestrichelte waagrechte Linie trennt die verschiedenen Operanden.

Beispiel einer erlaubte Ausführungs-
reihenfolgen angeben?

O1 O2

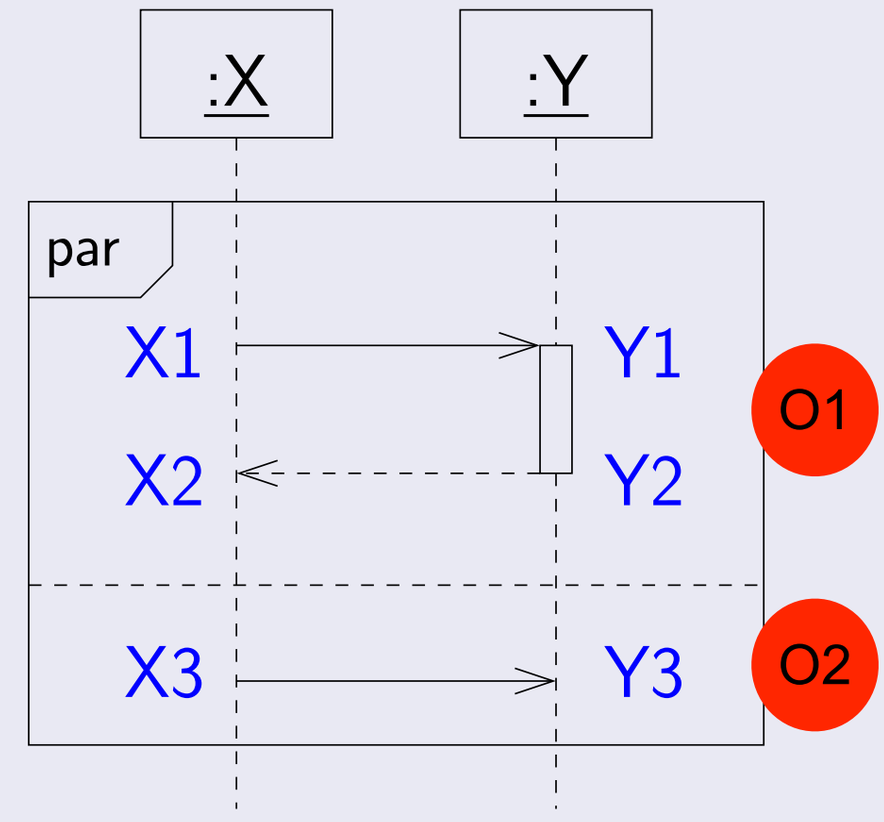
X1 < Y1 < Y2 < X2 < X3 < Y3
X3 < Y3 < X1 < Y1 < Y2 < X2

O2 O1

Sequenzdiagramme

Interaktionsoperator par (

Hier sind die Operanden i
Reihenfolge der Ereignisse
werden, ansonsten gibt es keine



Beispiel einer erlaubte Ausführungs-
reihenfolgen angeben?

O1 O2

X1 < Y1 < Y2 < X2 < X3 < Y3
X3 < Y3 < X1 < Y1 < Y2 < X2

O2 O1

Es sind auch alle möglichen "Verzahnungen" erlaubt (so lange die Reihenfolgen von O1 und O2 erhalten bleiben)

X1 < Y1 < X3 < Y2 < X2 < Y3
X3 < X1 < Y1 < Y2 < X2 < Y3

…

Frage: wie viele Reihenfolgen gibt es??

Die Menge aller “interleavings” von zwei Sequenzen der Laengen m und n ist

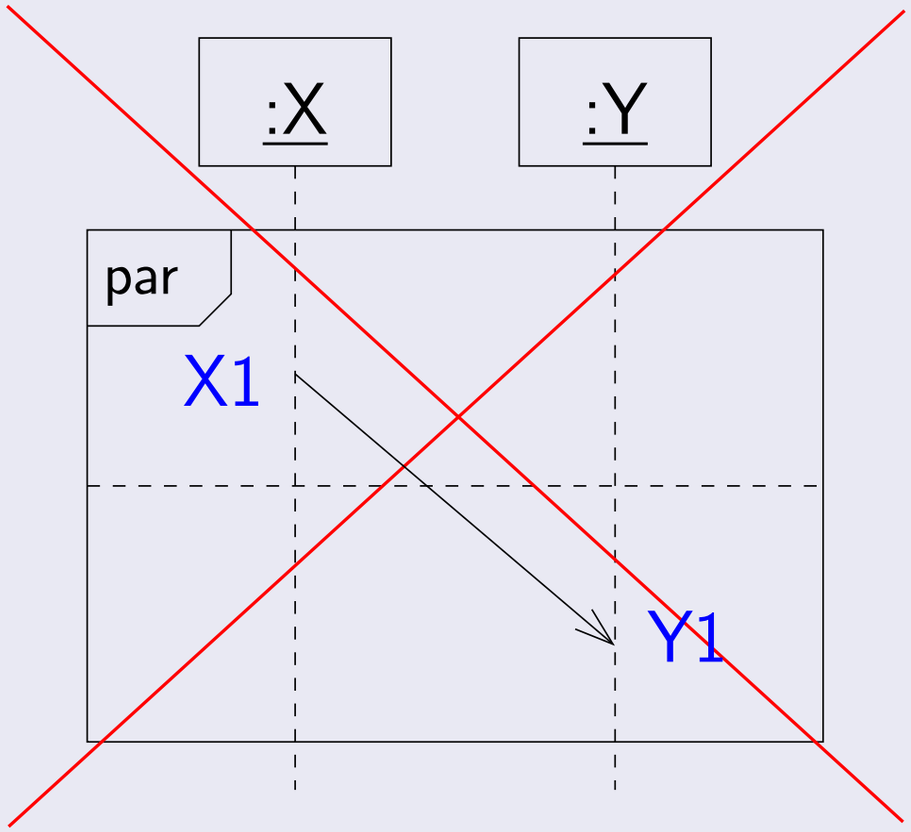
$$\frac{(m+n)!}{m!n!} = \left(\begin{array}{c} m \\ n \end{array} \right)$$

In unserem Fall:

$$\frac{(4+2)!}{4!2!} = \frac{6 \cdot 5}{2} = \underline{15}$$

Sequenzdiagramme

Nachrichten, die von einem Operanden in einen anderen laufen, sind nicht zugelassen.



Sequenzdiagramme

Ordnung auf den Ereignissen:

- X1 $<$ Y1 $<$ Y2 $<$ X2 (Operand 1)
- X3 $<$ Y3 (Operand 2)

Ansonsten gibt es keine weiteren Einschränkungen

Dieses Sequenzdiagramm beschreibt insgesamt fünfzehn Abläufe, beispielsweise:

- X1, Y1, X3, Y3, Y2, X2
- X3, X1, Y1, Y2, X2, Y3
- ...

Sequenzdiagramme

Weitere Bemerkungen: um die Ereignis-Ordnung innerhalb eines par-Operators zu bestimmen ...

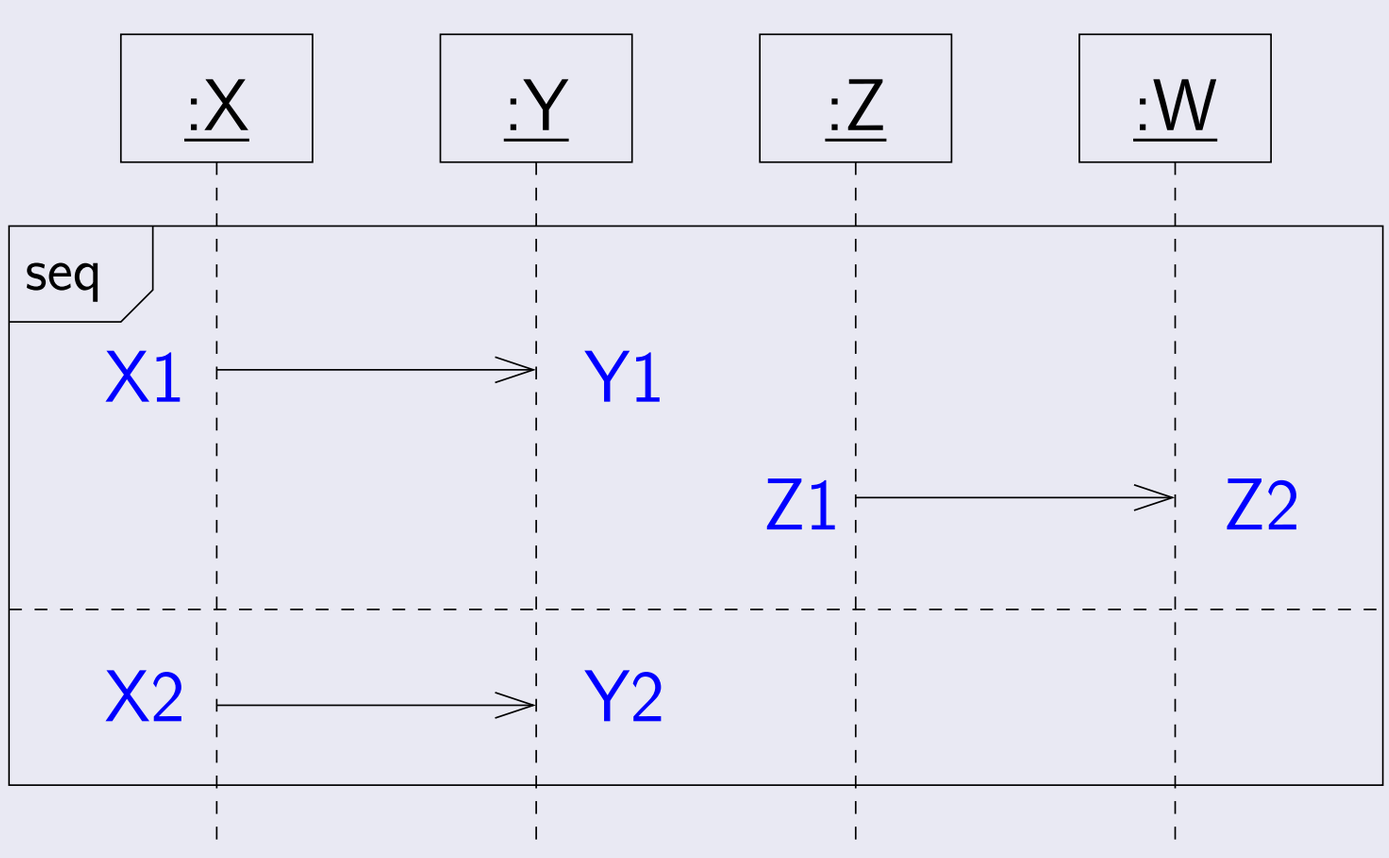
- verwendet man die partiellen Ordnungen der einzelnen Operanden ($\leq_1$, $\leq_2$)
- *und* vereinigt diese. Die Vereinigung $\leq_1 \cup \leq_2$ ist ebenfalls eine partielle Ordnung und beschreibt alle möglichen Reihenfolgen von Ereignissen.

Jede Ereignisreihenfolge, bei der ein Ereignis X immer vor einem Ereignis Y auftritt, falls X$<$Y gilt, ist eine mögliche Reihenfolge.

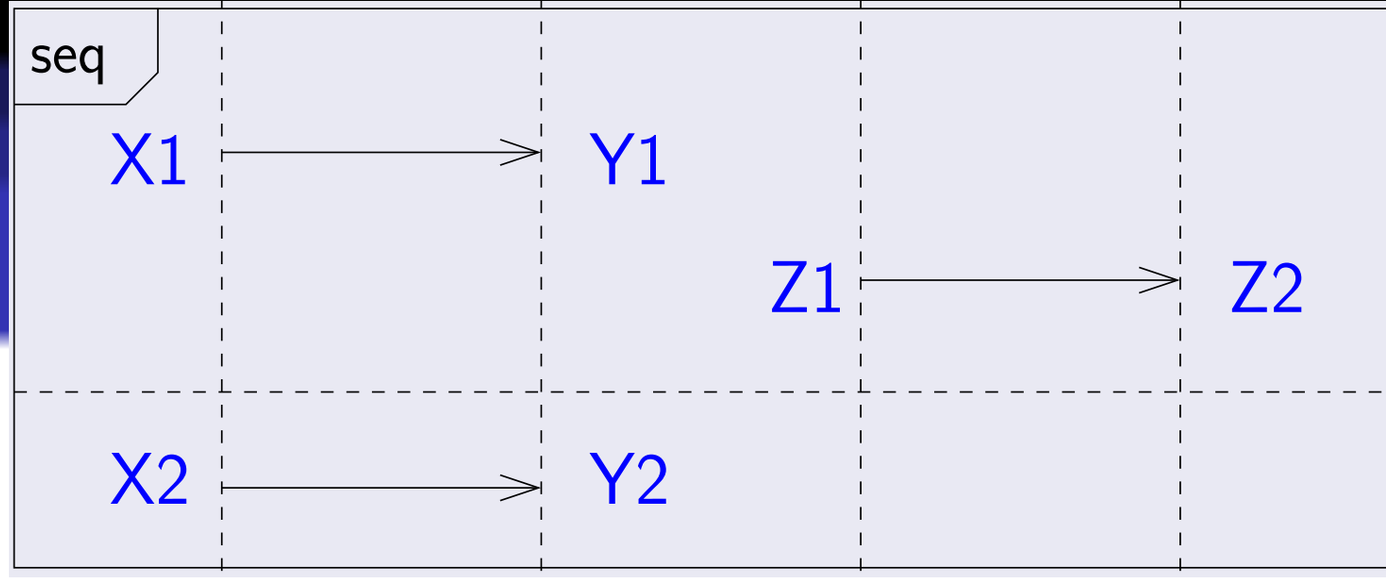
Sequenzdiagramme

Interaktionsoperator seq (schwache Sequenz)

Die Reihenfolge der Ereignisse *in* den Operanden muss wieder gewahrt werden, außerdem ist die Reihenfolge der Ereignisse auf den Lebenslinien (von oben nach unten) festgelegt.



Sequenzdiagramme

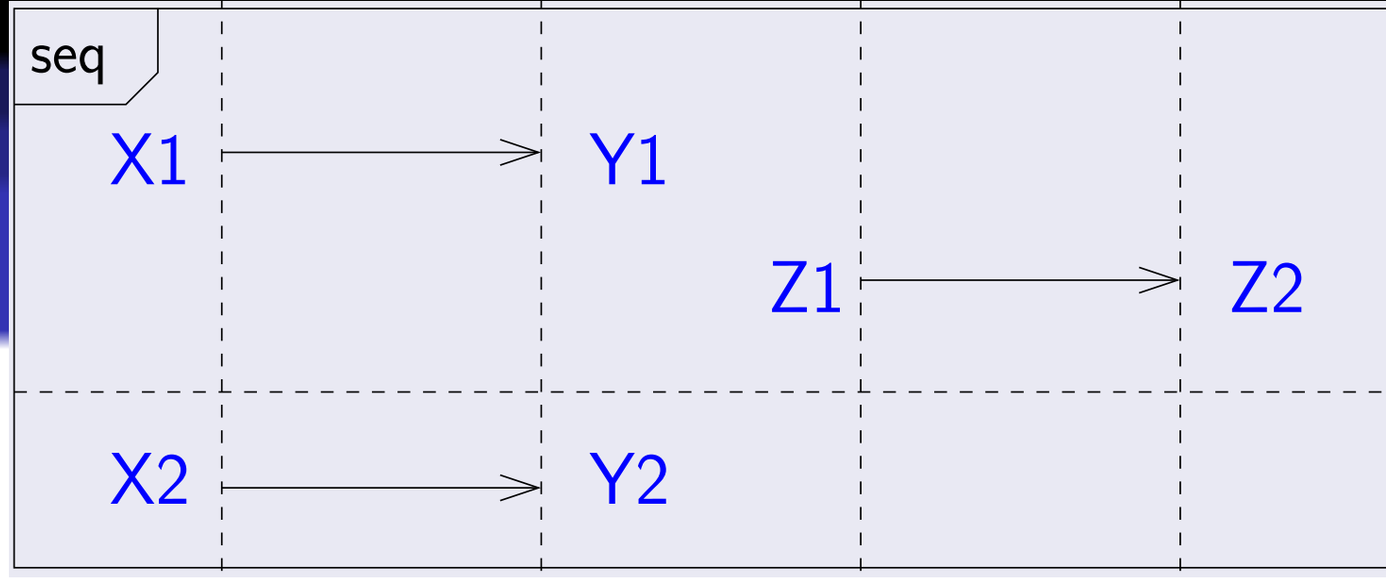


Ordnung auf den Ereignissen:

- X1 $<$ Y1 $<$ Z1 $<$ Z2 (Operand 1)
- X2 $<$ Y2 (Operand 2)
- X1 $<$ X2 (Lebenslinie von X)
- Y1 $<$ Y2 (Lebenslinie von Y)

==> wie viele Abläufe gibt es?

Sequenzdiagramme



Ordnung auf den Ereignissen:

- X1 $<$ Y1 $<$ Z1 $<$ Z2 (Operand 1)
- X2 $<$ Y2 (Operand 2)
- X1 $<$ X2 (Lebenslinie von X)
- Y1 $<$ Y2 (Lebenslinie von Y)

Dieses Sequenzdiagramm beschreibt neun verschiedene Abläufe, beispielsweise:

- X1, X2, Y1, Z1, Z2, Y2
- X1, Y1, X2, Y2, Z1, Z2
- ...

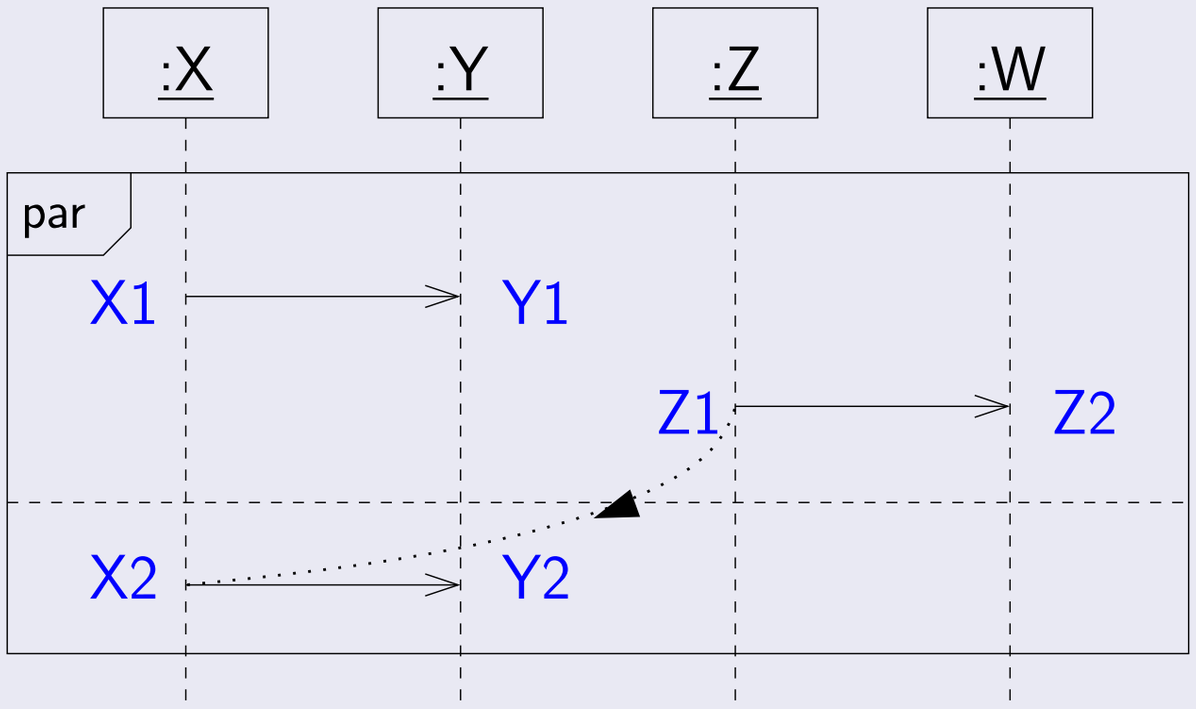
Sequenzdiagramme

Weitere Bemerkungen: um die Ereignis-Ordnung innerhalb eines seq-Operators zu bestimmen ...

- verwendet man die partiellen Ordnungen der einzelnen Operanden ($\leq_1$, $\leq_2$),
- außerdem die partiellen Ordnungen der einzelnen Lebenslinien ($\leq_X$, $\leq_Y$, ...),
- vereinigt diese und bestimmt die reflexiv-transitive Hülle. Dabei entsteht eine partielle Ordnung, die alle möglichen Reihenfolgen von Ereignissen beschreibt.

Sequenzdiagramme

Bei Interaktions-Operanden ist es außerdem möglich, zusätzliche Ordnungsbeziehungen einzuführe, die ansonsten nicht abgedeckt sind. (Durch gestrichelte Linien mit Pfeil in der Mitte.)



Die neue partielle Ordnung kann man hier dadurch bestimmen, indem man die neuen Paare hinzufügt und die reflexiv-transitive Hülle der Gesamtrelation bildet.

Sequenzdiagramme

Weitere Interaktions-Operatoren (Liste ist nicht vollständig):

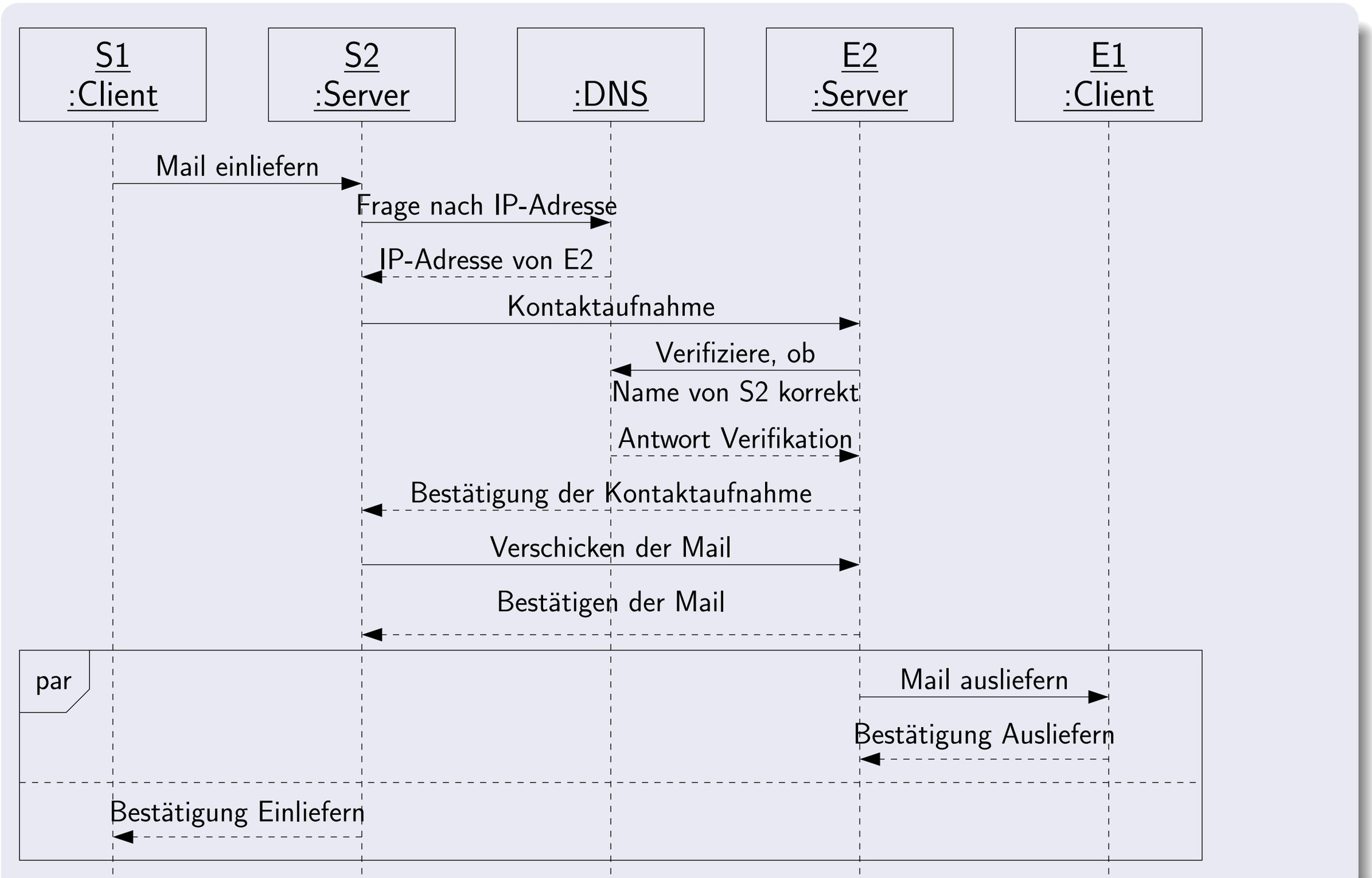
- strict (Strikte Sequenz): Die Reihenfolge der Ereignisse wird von oben nach unten strikt eingehalten (wie in einem einzelnen Operanden).
- alt (Alternative): entsprechend einer If-Then-Else-Anweisung (mit Bedingungen)
- neg (Negation): beschreibt eine ungültige Nachrichtenreihenfolge
- loop (Schleife): beschreibt die Wiederholung eines Operanden

Sequenzdiagramme

Beispiel: wir modellieren ein Protokoll, bei dem ein Client-Rechner S einen E-Mail an einen anderen Client R verschickt.

Weitere Beteiligte sind der Mail-Server von S, der Mail-Server von R und der DNS-Server, der benötigt wird, um Mail-Adressen in die IP-Adressen des entsprechenden Servers umzuwandeln (DNS = domain name system).

Sequenzdiagramme



Sequenzdiagramme

Bemerkung: dieses Sequenzdiagramm ist an den Ablauf im SMTP-Protokoll angelehnt (SMTP = send mail transport protocol), ist jedoch erheblich vereinfacht.

Sequenzdiagrammen zu vielen TCP/IP-Netzwerkprotokollen findet man unter:

`http://www.eventhelix.com/Realtimemantra/Networking/`

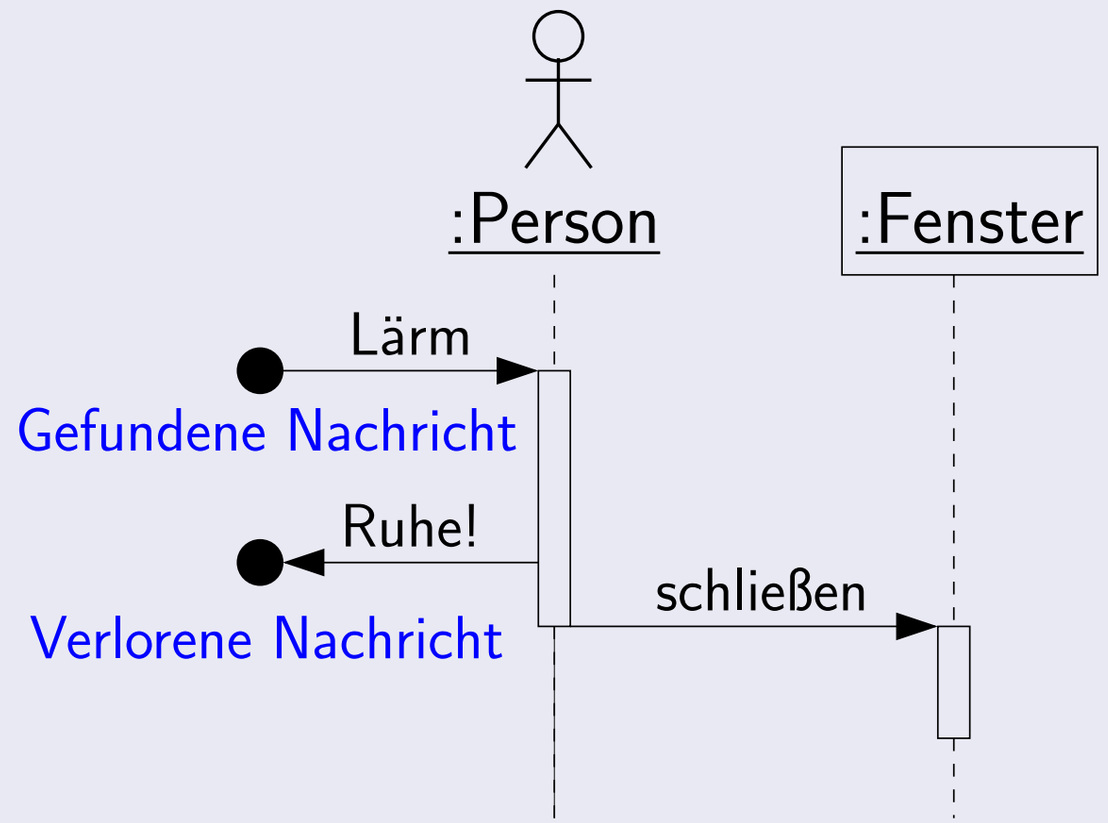
Sequenzdiagramme

Es gibt noch einige weitere Arten von Nachrichten:

Gefundene und verlorene Nachrichten

Bei gefundenen und verlorenen Nachrichten werden das Sende- bzw. das Empfangsereignis nicht explizit modelliert. Die Nachrichten tauchen quasi aus der Umgebung auf und verschwinden wieder dorthin.

Solche Nachrichten werden benötigt, wenn die entsprechenden Kommunikationspartner nicht mitmodelliert werden.

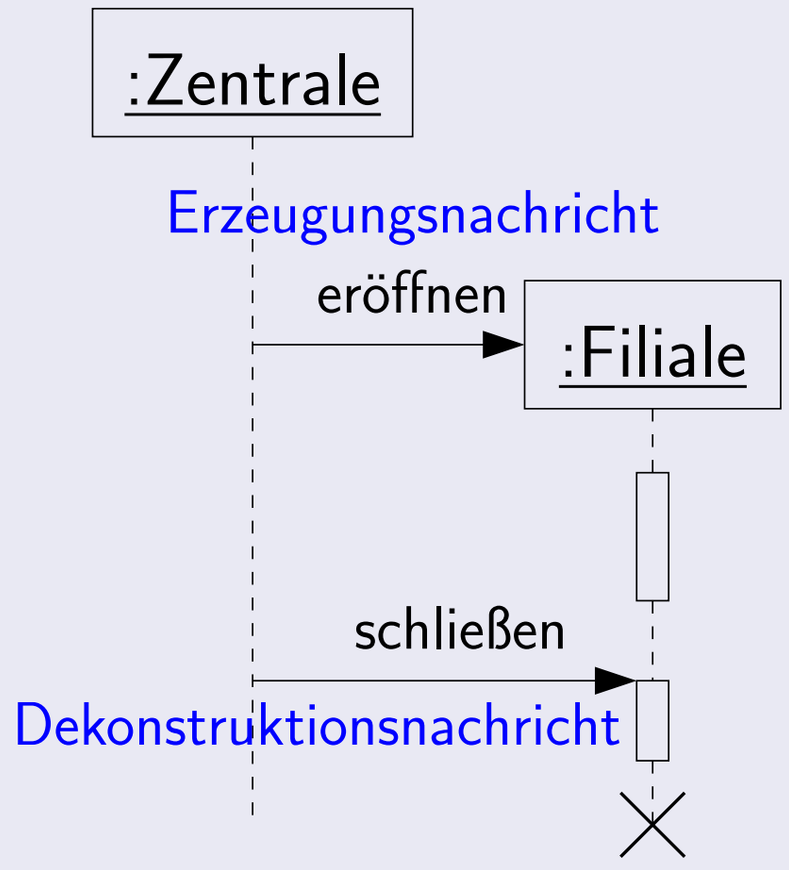


Sequenzdiagramme

Erzeugung und Dekonstruktion von Objekten

Außerdem kann es passieren, dass Objekte nicht während des ganzen Ablaufs zur Verfügung stehen. Sie können während des Ablaufs dynamisch erzeugt und wieder zerstört werden.

Dies erfolgt zumeist durch sogenannte Erzeugungs- und Dekonstruktionsnachrichten und wird folgendermaßen dargetellt:



Beispiele

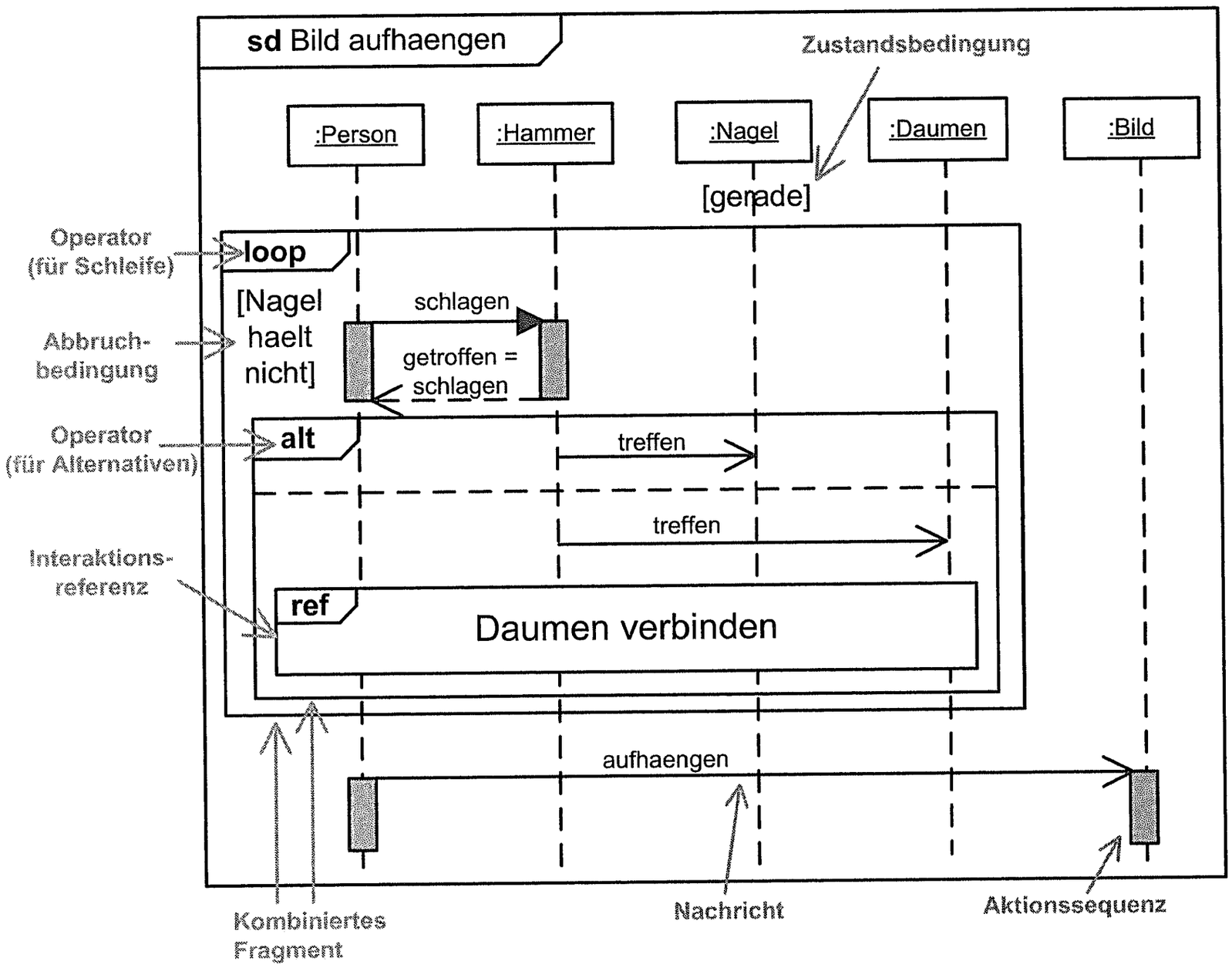
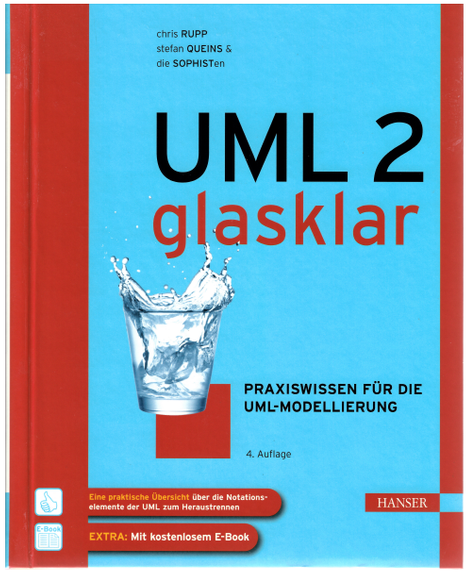


Diagramm ist diesem Buch entnommen:



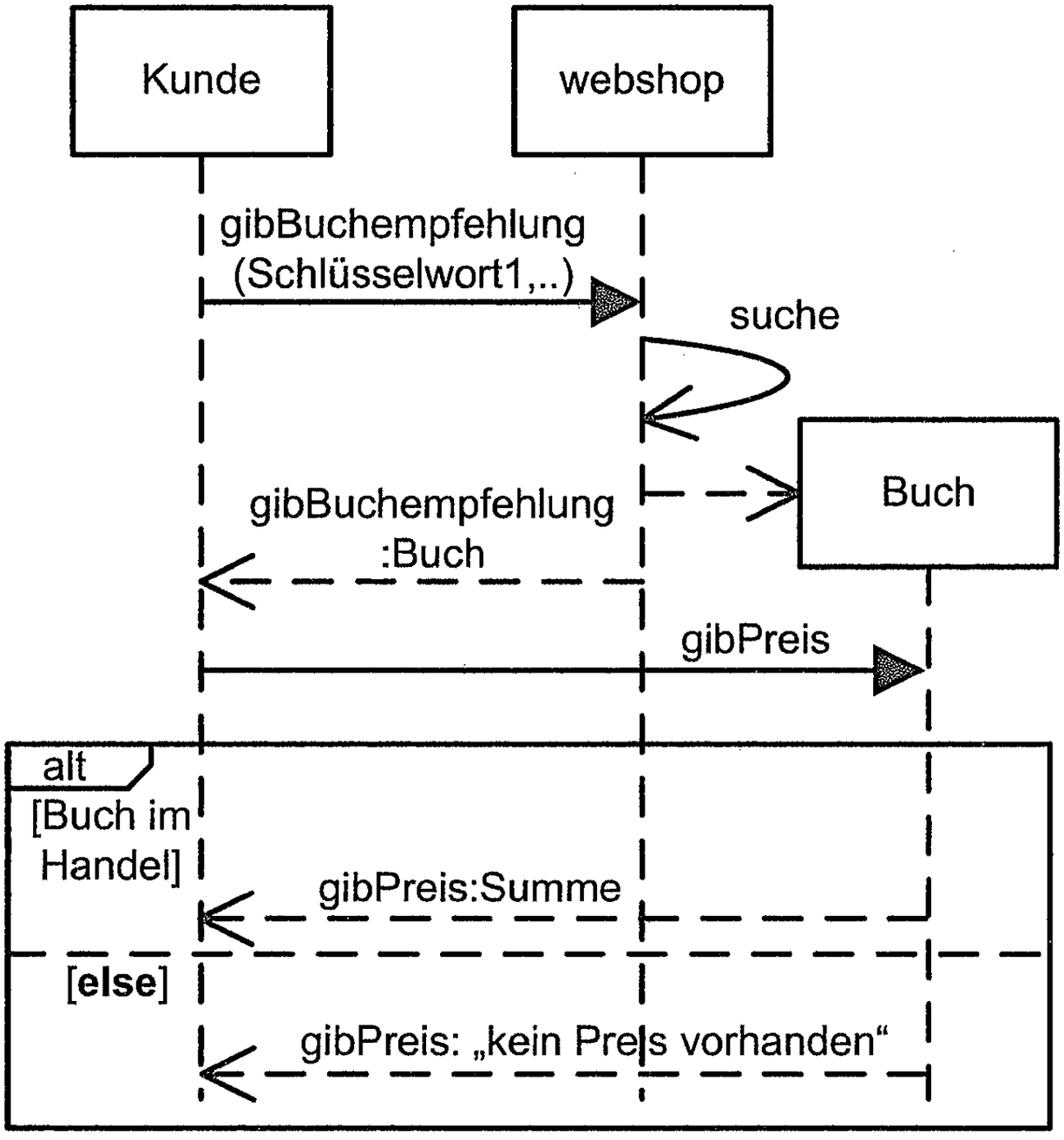
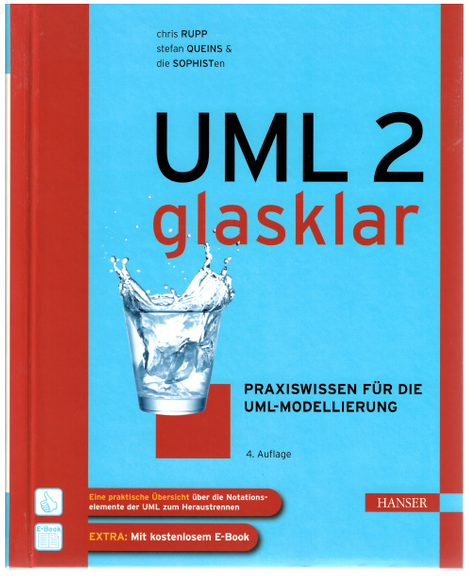


Diagramm ist diesem Buch entnommen:



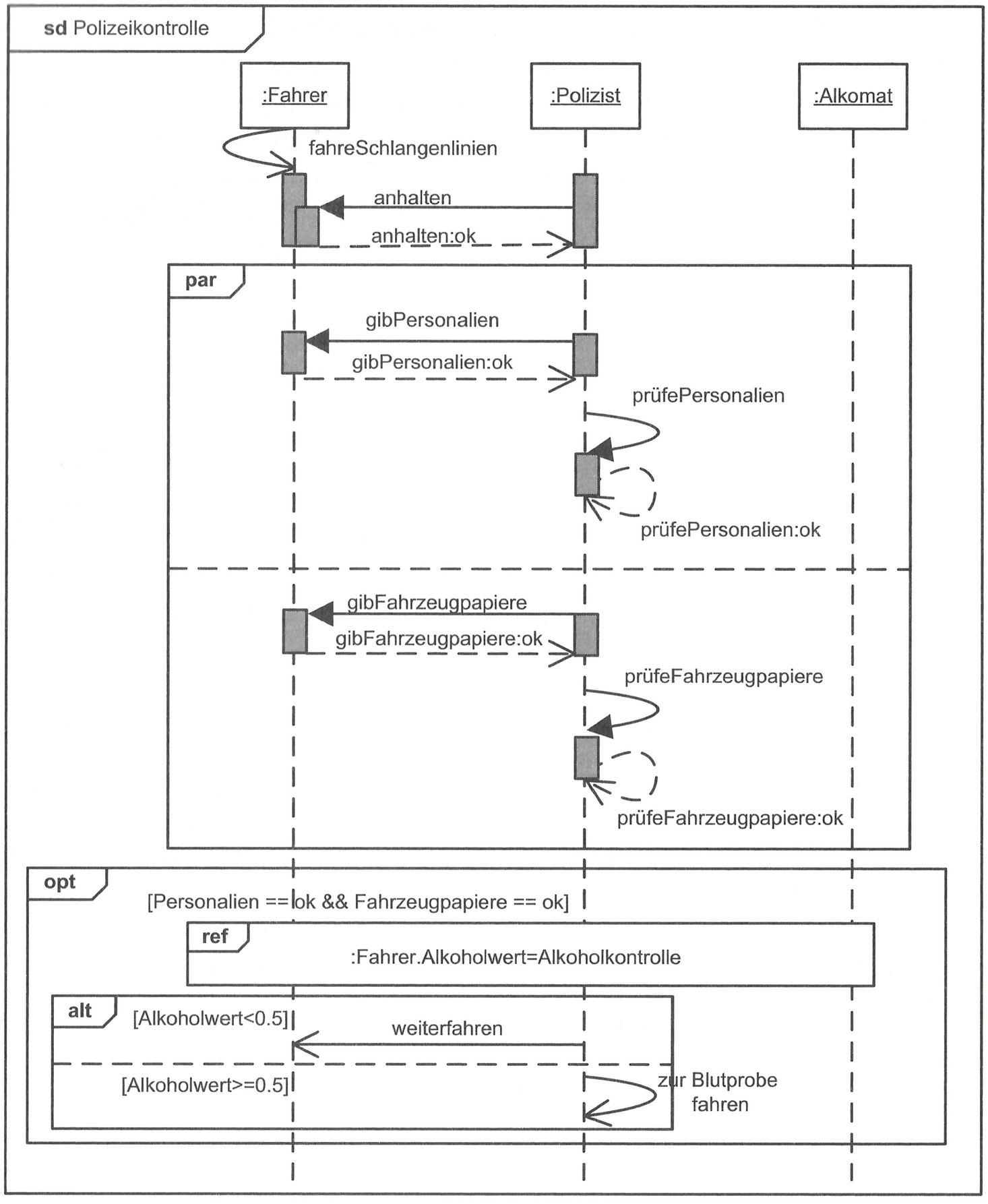
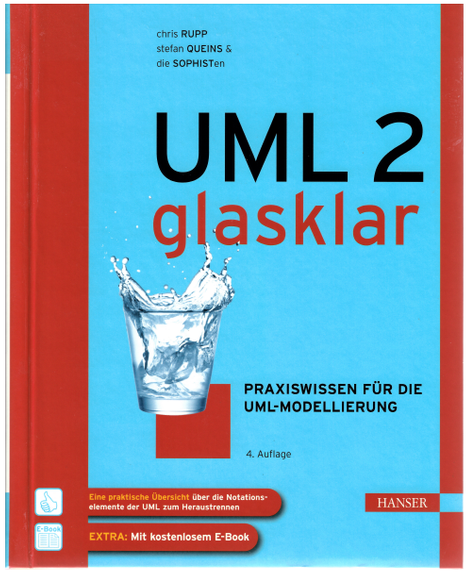


Diagramm ist diesem Buch entnommen:



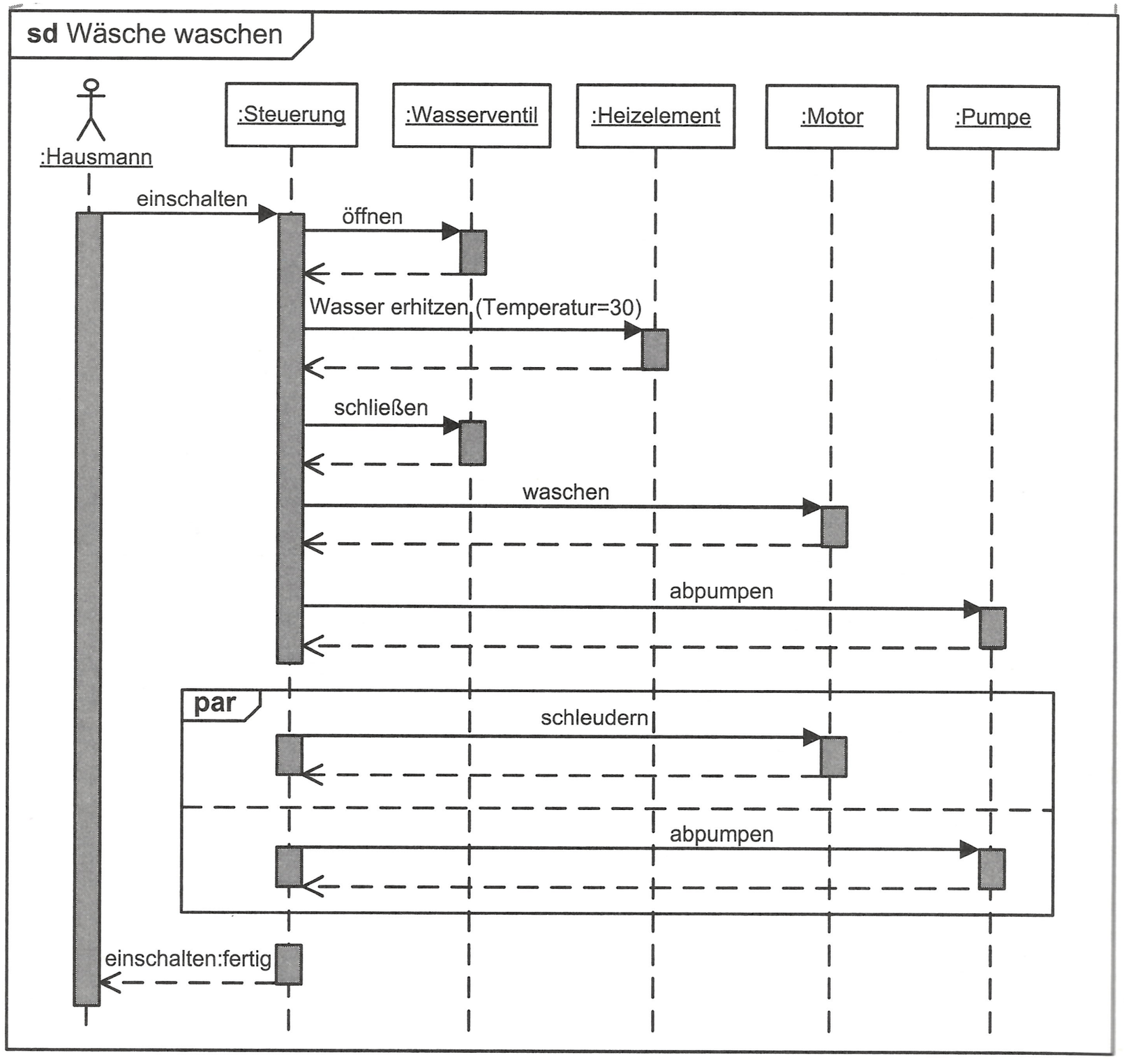
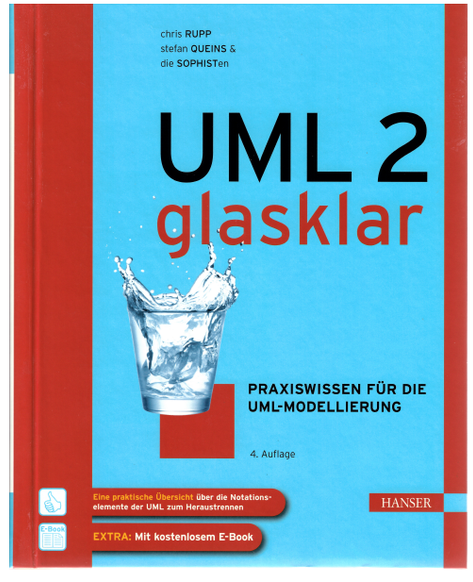
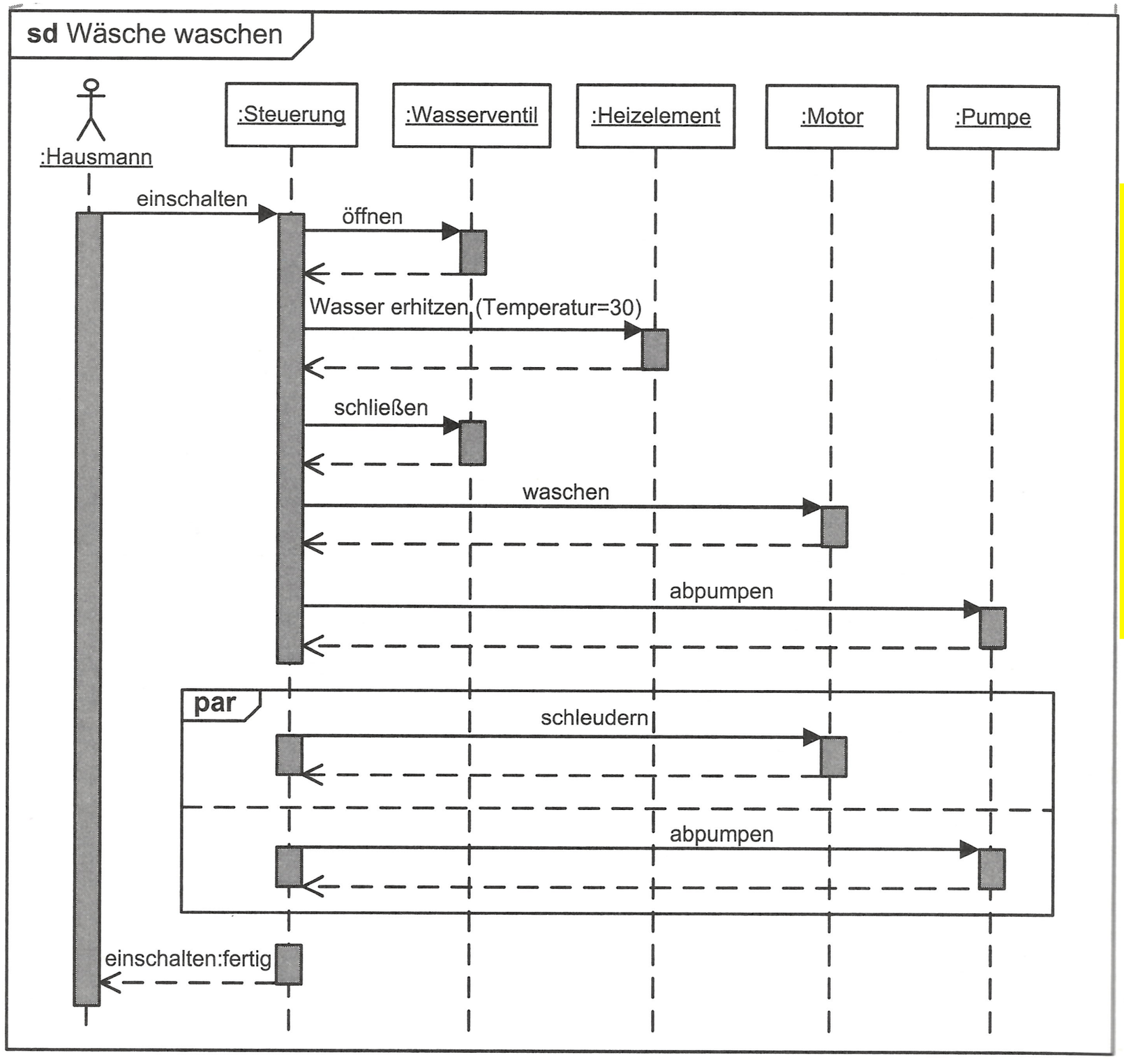


Diagramm ist diesem Buch entnommen:



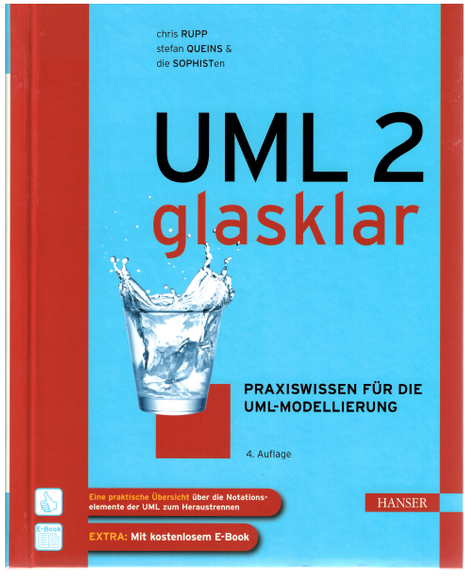


Fragen:

— schleudern und abpumpen soll gleichzeitig passieren (nicht wirklich modellierbar)

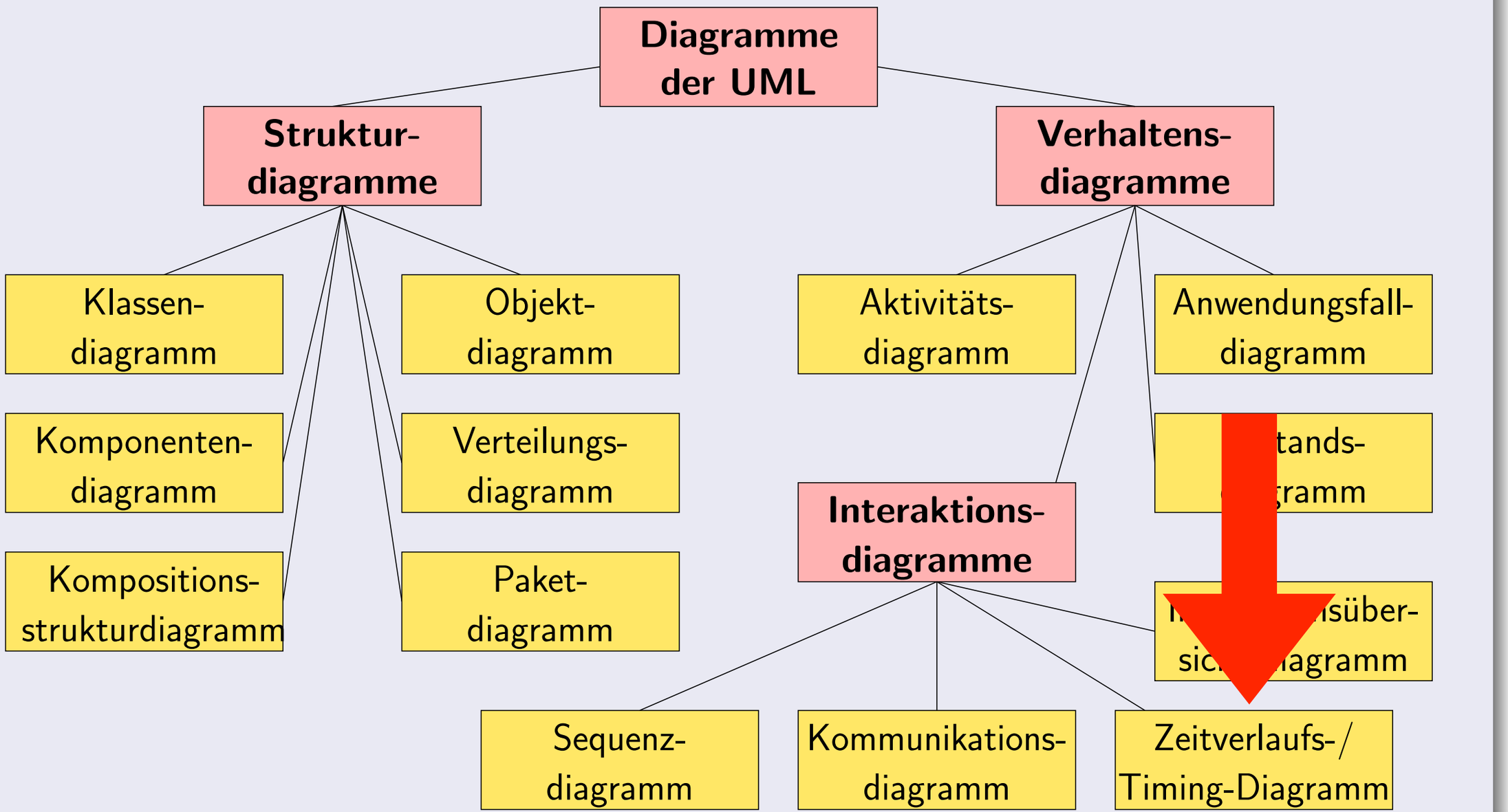
— warum kann man nicht genaue Zeitdauern mit angeben?

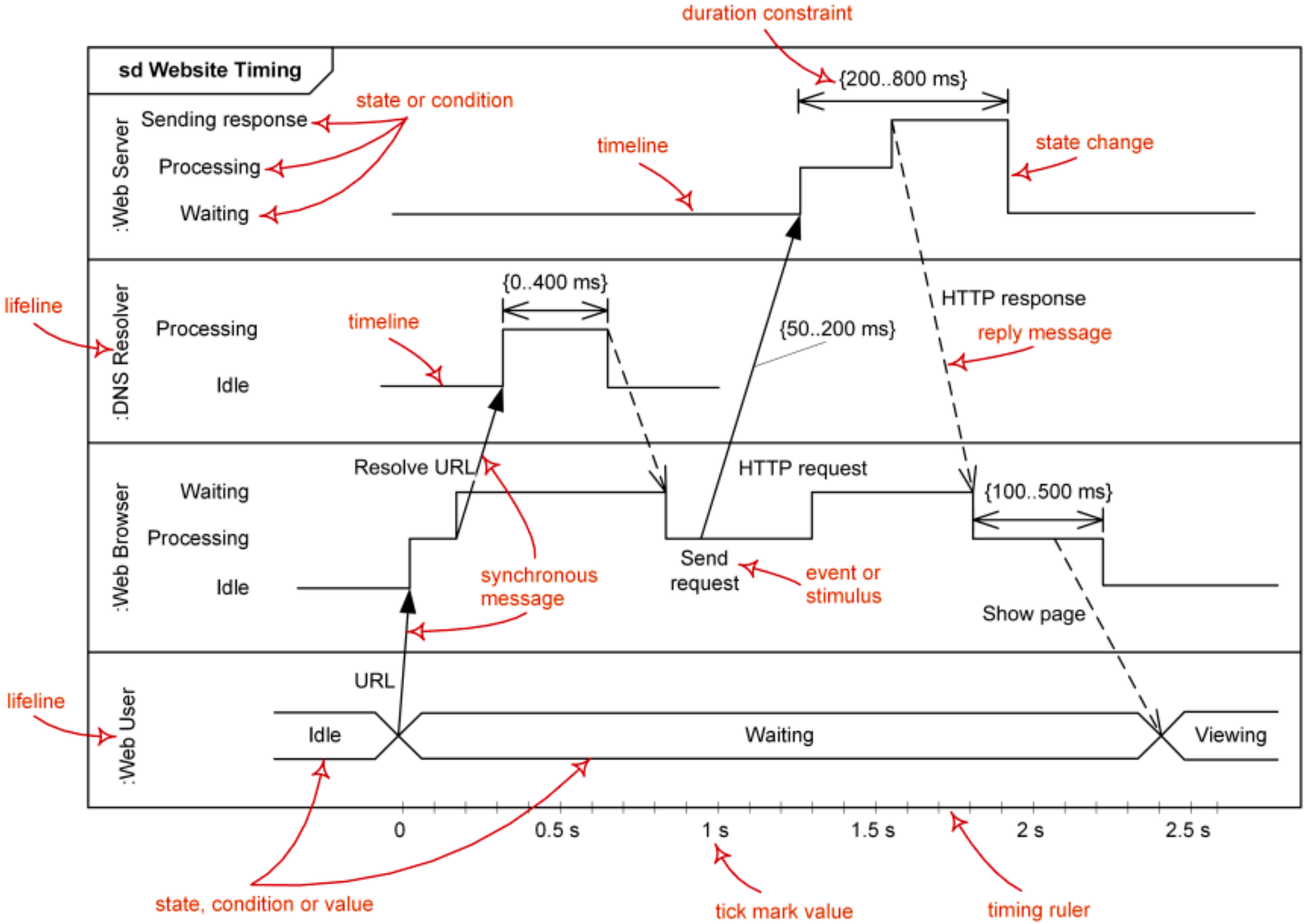
Diagramm ist diesem Buch entnommen:

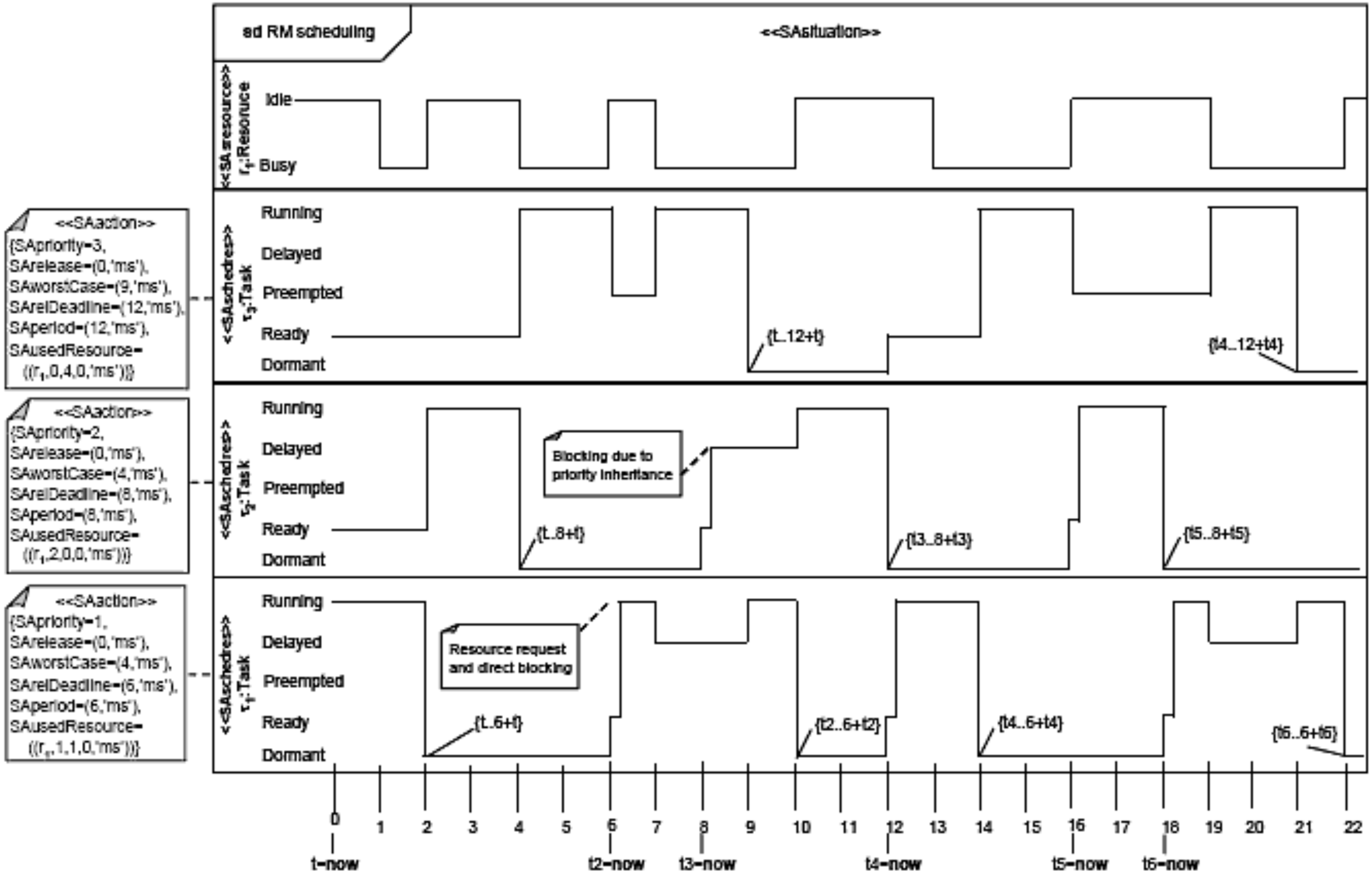


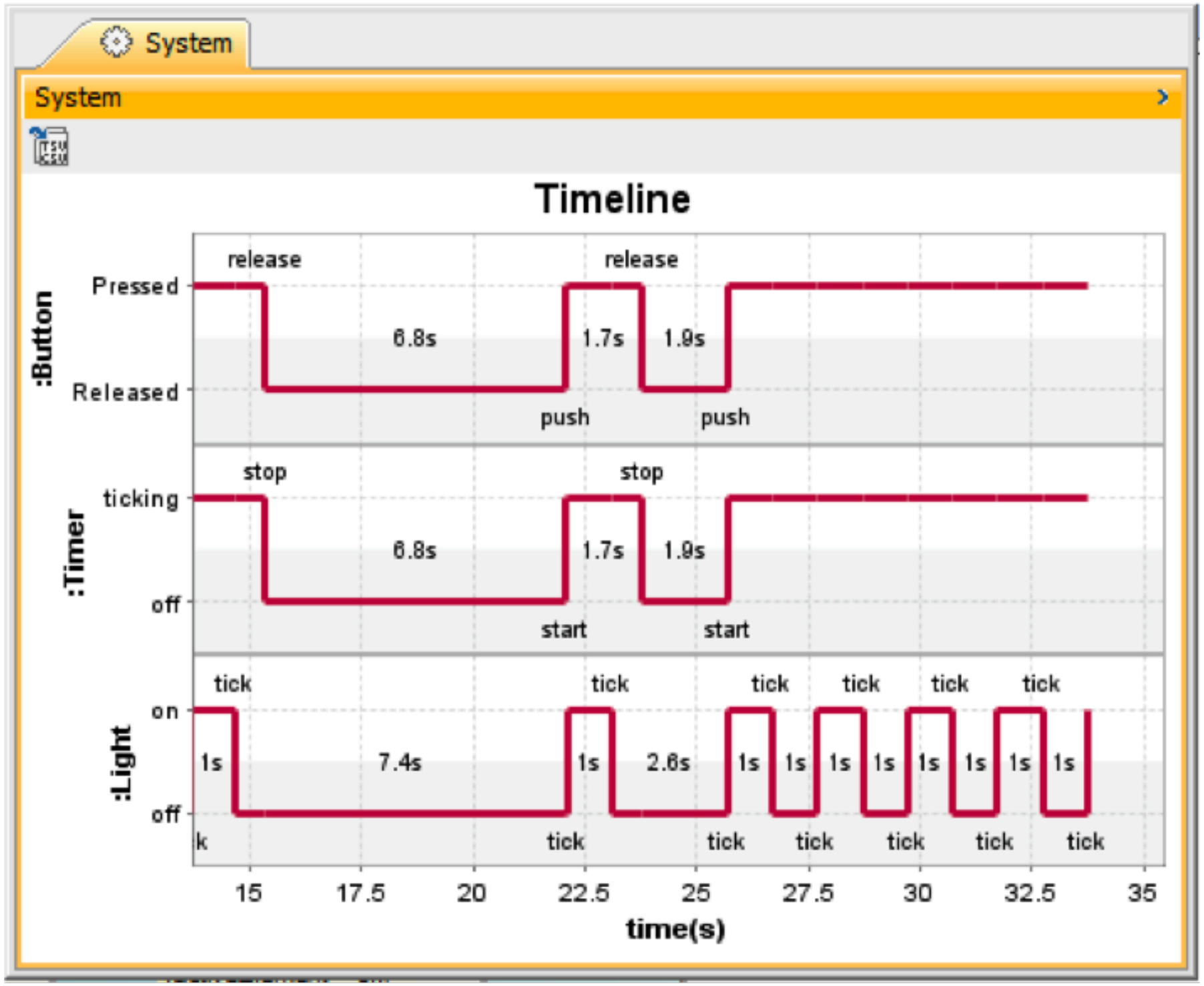
UML: Einführung

UML-Diagramme




sd Website Timing
duration constraint
{200..800 ms}
state or condition
:Web Server
Sending response
Processing
Waiting
timeline
state change
lifeline
:DNS Resolver
Processing
Idle
{0..400 ms}
timeline
{50..200 ms}
HTTP response
reply message
:Web Browser
Waiting
Processing
Idle
Resolve URL
HTTP request
{100..500 ms}
synchronous message
Send request
event or stimulus
Show page
URL
lifeline
:Web User
Idle
Waiting
Viewing
0
0.5 s
1 s
1.5 s
2 s
2.5 s
state, condition or value
tick mark value
timing ruler


sd RM scheduling
<<SAsituation>>
<<SAresource>>
r1:Resource
Idle
Busy
<<SAschedres>>
τ3:Task
Running
Delayed
Preempted
Ready
Dormant
{t..12+t}
{t4..12+t4}
<<SAschedres>>
τ2:Task
Blocking due to priority inheritance
{t..8+t}
{t3..8+t3}
{t5..8+t5}
<<SAschedres>>
τ1:Task
Resource request and direct blocking
{t..6+t}
{t2..6+t2}
{t4..6+t4}
{t5..6+t5}
<<SAaction>>
{SApriority=3,
SArelease=(0,'ms'),
SAworstCase=(9,'ms'),
SArelDeadline=(12,'ms'),
SAperiod=(12,'ms'),
SAusedResource=
((r1,0,4,0,'ms'))}
<<SAaction>>
{SApriority=2,
SArelease=(0,'ms'),
SAworstCase=(4,'ms'),
SArelDeadline=(8,'ms'),
SAperiod=(8,'ms'),
SAusedResource=
((r1,2,0,0,'ms'))}
<<SAaction>>
{SApriority=1,
SArelease=(0,'ms'),
SAworstCase=(4,'ms'),
SArelDeadline=(6,'ms'),
SAperiod=(6,'ms'),
SAusedResource=
((r1,1,1,0,'ms'))}
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
t=now
t2=now
t3=now
t4=now
t5=now
t6=now


System
System
Timeline
:Button
Pressed
Released
release
release
6.8s
1.7s
1.9s
push
push
:Timer
ticking
off
stop
stop
6.8s
1.7s
1.9s
start
start
:Light
on
off
tick
tick
tick
tick
tick
tick
1s
7.4s
1s
2.6s
1s
1s
1s
1s
1s
1s
1s
1s
k
tick
tick
tick
tick
tick
tick
15
17.5
20
22.5
25
27.5
30
32.5
35
time(s)

Kurs
Datenbankgru
und Mod

S
sität Bremen
remen.de

ersemester 2023

26.6.2023
Vorlesung 9: Sequenzdiagramme
(Sequence Diagrams)