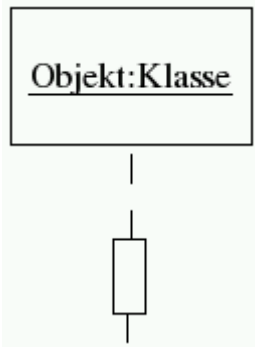
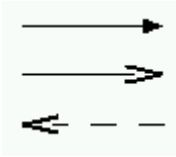
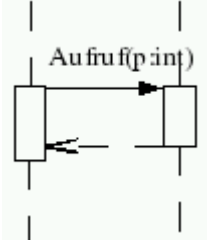
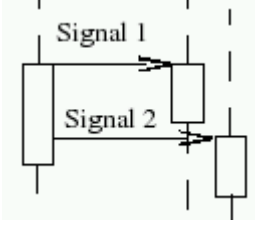
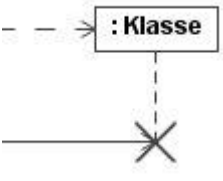


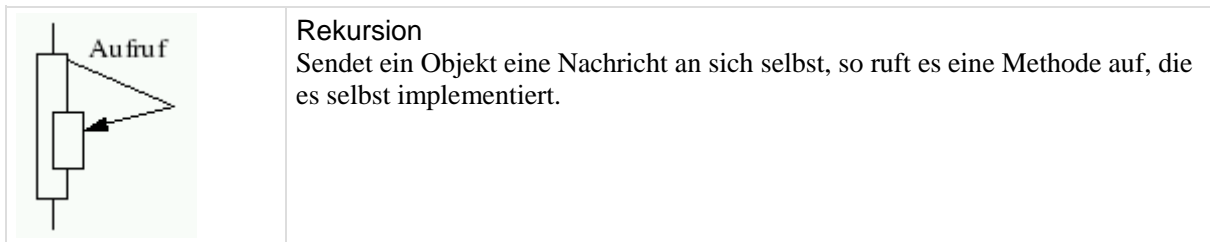
Sequenzdiagramm

Zweck

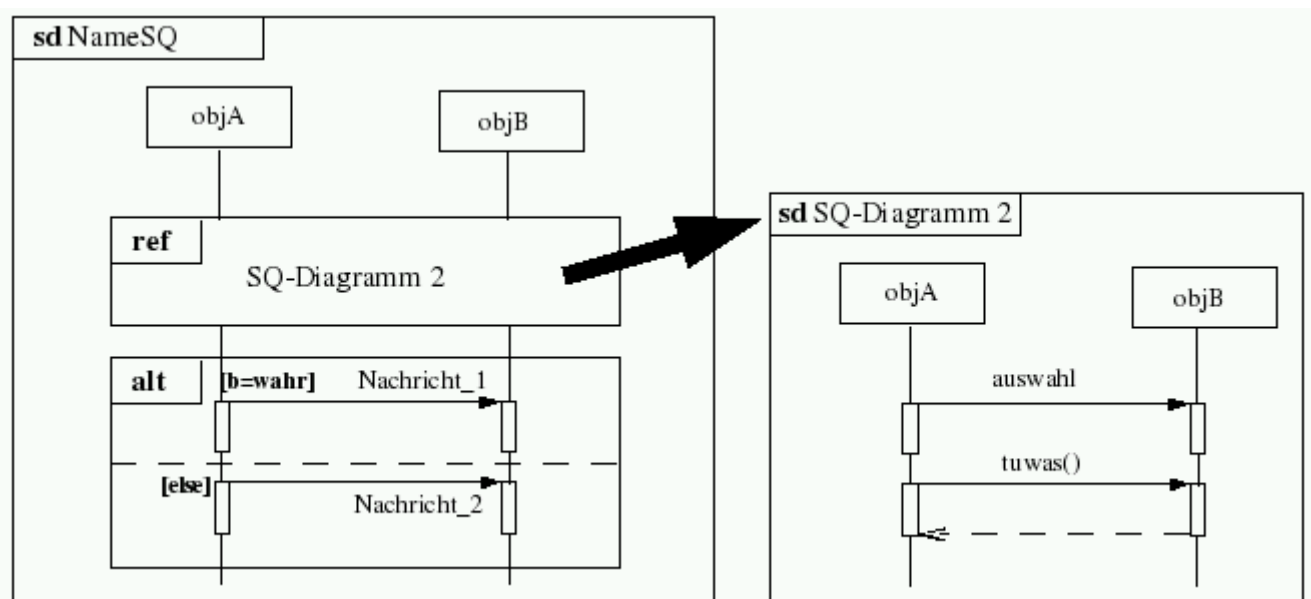
Sequenzdiagramme beschreiben die Kommunikation zwischen Objekten in einer bestimmten Szene. Es wird beschrieben welche Objekte an der Szene beteiligt sind, welche Informationen (Nachrichten) sie austauschen und in welcher zeitlichen Reihenfolge der Informationsaustausch stattfindet. Sequenzdiagramme enthalten eine implizite Zeitachse. Die Zeit schreitet in einem Diagramm von oben nach unten fort. Die Reihenfolge der Pfeile in einem Sequenzdiagramm gibt die zeitliche Reihenfolge der Nachrichten an.

Notation

	Klasse, Objekt In dem Rechteck oberhalb der gestrichelten Linie wird der Objektname und der Klassenname angegeben. Der Name wird unterstrichen. Die senkrechte, gestrichelte Linie stellt die Lebenslinie (lifeline) eines Objekts dar. In diesem zeitlichen Bereich existiert das Objekt. Das schmale Rechteck auf der gestrichelten Linie stellt eine Aktivierung dar. Eine Aktivierung ist der Bereich, in dem eine Methode des Objektes aktiv ist (ausgeführt wird). Auf einer Lebenslinie können mehrere Aktivierungen enthalten sein.
	Nachricht, Botschaft Objekte kommunizieren über Nachrichten. Nachrichten werden als Pfeile zwischen den Aktivierungen eingezeichnet. Der Name der Nachricht steht an dem Pfeil. Eine Nachricht liegt immer zwischen einem sendenden und einem empfangenden Objekt.
	Synchrone Nachricht (Aufruf) Der Pfeil mit der ausgefüllten Spitze bezeichnet einen synchronen Aufruf. Der Aufruf erfolgt von der Quelle zum Ziel, d. h. die Zielklasse muss eine entsprechende Methode implementieren. Die Quelle wartet mit der weiteren Verarbeitung bis die Zielklasse ihre Verarbeitung beendet hat und setzt die Verarbeitung dann fort. Ein Aufruf muss einen Namen haben; in runden Klammern können Aufrufparameter angegeben werden. Der gestrichelte Pfeil ist der Return. Die Bezeichnung des Return mit einem Namen ist optional.
	Asynchrone Nachricht Mit einer offenen Pfeilspitze werden asynchrone Nachrichten gekennzeichnet. Der Aufruf erfolgt von der Quelle zum Ziel. Die Quelle wartet mit der Verarbeitung nicht auf die Zielklasse, sondern setzt ihre Arbeit nach dem Senden der Nachricht fort. Asynchrone Aufrufe werden verwendet, um parallele Threads zu modellieren.
	Objekt erzeugen, zerstören Das Erzeugen eines Objektes wird durch eine Nachricht, die im Kopf des Objekts endet dargestellt (gestrichelte Linie). Das Zerstören (Löschen) eines Objektes wird durch ein x auf der Lebenslinie markiert.



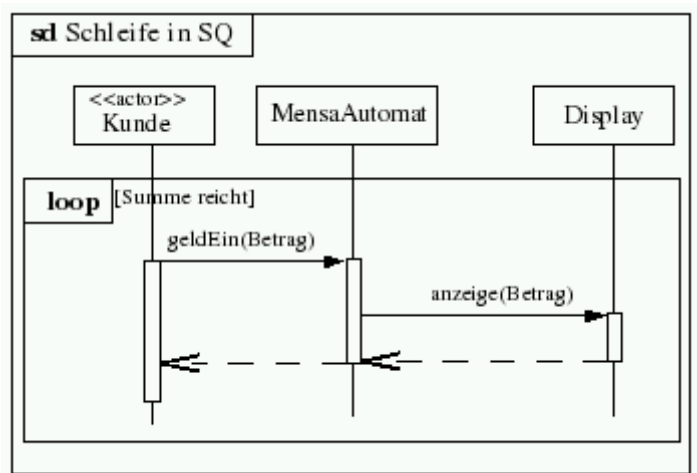
Ein Sequenzdiagramm enthält Objekte, Lebenslinien, Aktivierungen, Nachrichten und kombinierte Fragmente (combined fragment). Mit kombinierten Fragmenten können in Sequenzdiagrammen die Kontrollstrukturen höherer Programmiersprachen ausgedrückt werden. Jedes kombinierte Fragment wird als rechteckiger Block dargestellt. In der linken, oberen Ecke trägt er eine Bezeichnung, die den Typ der Kontrollstruktur angibt (interaction operator). Die wichtigsten kombinierte Fragmente der UML 2.0 sind weiter unten erläutert.



Diese Abbildung stellt das Prinzip des Sequenzdiagramms in UML 2.0 dar. In der linken, oberen Ecke steht der Diagrammtyp (sd) und der Name des Diagramms (NameSQ). Im Diagramm sind die beiden Objekte objA und objB enthalten. Sie existieren während der gesamten Szene; das ist daran ersichtlich, dass die Lebenslinien bis zum unteren Bildrand durchgezogen sind und kein x enthalten, das das Zerstören eines Objektes ausdrückt.

Sequenzdiagramme können in der UML 2.0 verschachtelt sein. Aus einem Sequenzdiagramm kann auf Sequenzen in einem anderen Diagramm verwiesen werden. Der mit ref bezeichnete Block stellt einen Verweis auf ein Diagramm mit dem Namen "SQ-Diagramm 2" dar, das eine Verfeinerung des Blockes enthält.

Der mit alt markierte Block beschreibt eine Alternative (Verzweigung). Ist die Bedingung [b=wahr] erfüllt, wird der Bereich oberhalb der gestrichelten Linie in dem alt-Block ausgeführt. Im else-Fall wird der Bereich unterhalb der gestrichelten Linie ausgeführt.



Diese Abbildung zeigt eine Schleife in einem Sequenzdiagramm. Die modellierte Szene ist eine Variation des Beispiels Mensaautomat. Es sind drei Objekte an der Szene beteiligt. Das Objekt Kunde ist vom Stereotyp <<actor>>. Er ist ein Akteur (aus dem Use Case Diagramm) und greift auf den MensaAutomat zu, indem er ihm die Botschaft "geldEin" sendet; diese Botschaft übergibt den Parameter "Betrag". Danach sendet der "MensaAutomat" die Botschaft "anzeige(Betrag)" an das "Display". Das "Display" führt die Methode aus und gibt die Kontrolle über den Return wieder an den "MensaAutomat" zurück. Der "MensaAutomat" gibt die Kontrolle wieder an den "Kunden" zurück. Diese Sequenz von Botschaften wird solange ausgeführt bis die Abbruchbedingung [Summe reicht] erfüllt ist. Das Prüfen des Geldes auf Echtheit ist in diesem Beispiel nicht modelliert worden.

Anwendungsbereich

Sequenzdiagramme werden zur Darstellung der Kommunikation von Objekten eingesetzt. Sie können in der Analyse und im Design eingesetzt werden. Sequenzdiagramme werden häufig verwendet, um Szenarien eines Systems zu modellieren. Ein Szenario (Szene) ist die Beschreibung einer Sicht auf einen Teil eines Systems. Sequenzdiagramme können auch zur Modellierung von komplexen Operationen eines Systems verwendet werden und detaillierte Designinformationen enthalten.

Ein System wird in der Regel nicht vollständig durch Sequenzdiagramme spezifiziert. Es werden nur die Szenen modelliert, die häufig vorkommen oder besonders wichtig sind.

Zusammenhang

Sequenzdiagramme können sich auf Use Cases beziehen. Da Use Case Diagramme aber nicht in jedem Projekt verwendet werden, können Sequenzdiagramme auch unabhängig von Use Case Diagrammen existieren. Die Klassen der Objekte in Sequenzdiagrammen müssen in den Klassendiagrammen enthalten sein. Ereignisse und Nachrichten erfordern entsprechende Methoden bei den Zielklassen.

Alternativ zu den Sequenzdiagrammen können auch Interaktionsübersichtdiagramme, Kommunikationsdiagramme, Zeitverlaufdiagramme oder Aktivitätsdiagramme verwendet werden.

Hinweise für die Praxis

Sequenzdiagramme werden zur Modellierung von Interaktionen gewählt, wenn die Darstellung der Reihenfolge des Nachrichtenaustauschs wichtig ist.

Bevor Sequenzdiagramme modelliert werden können, müssen die an der Szene beteiligten Klassen identifiziert werden. Hinweise zur Identifikation von Klassen befinden sich im Kapitel Klassendiagramm.

Die in Sequenzdiagrammen modellierten Objekte und Nachrichten müssen konsistent zu den Klassendiagrammen gehalten werden.

Da Sequenzdiagramme dynamische Aspekte eines Systems darstellen, müssen normalerweise Instanznamen und Klassennamen angegeben werden. Häufig vernachlässigt man jedoch den Instanznamen und gibt nur den Klassennamen an. Ein solches anonymes Objekt steht stellvertretend für alle Objekte der Klasse und impliziert, dass alle Objekte der Klasse sich in der Szene gleich verhalten.

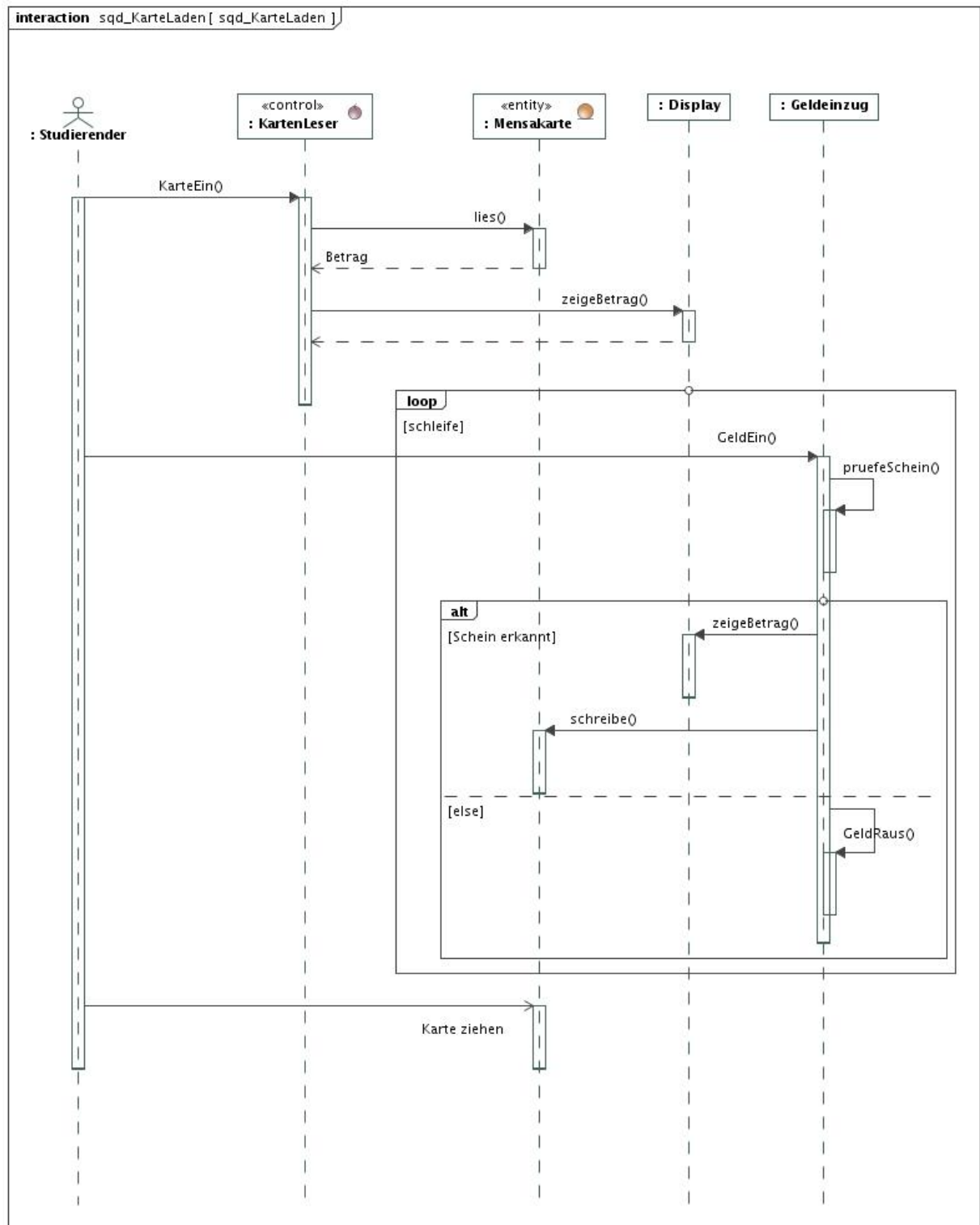
Style Guide

- Objekte sollten so angeordnet werden, dass in einem Diagramm möglichst wenig Kreuzungen entstehen.
- Nachrichten sollten von links nach rechts zeigen, Returns von rechts nach links.
- Nachrichtennamen sollten an der Spitze der Pfeile stehen.
- Beteiligte Akteure sollten am linken Rand der Diagramme stehen.
- Namen der Akteure sollten konsistent zu den Use Case Diagrammen sein.
- Klassennamen sollten konsistent zu dem Klassendiagramm sein.

Beispiel

Sequenzdiagramm der Szene Karte laden

Dieses Sequenzdiagramm modelliert die Szene "Karte laden". Es bezieht sich auf den Use Case "KarteLaden" des Use Case Diagramms "Mensa".



Die Objektnamen wurden vernachlässigt.

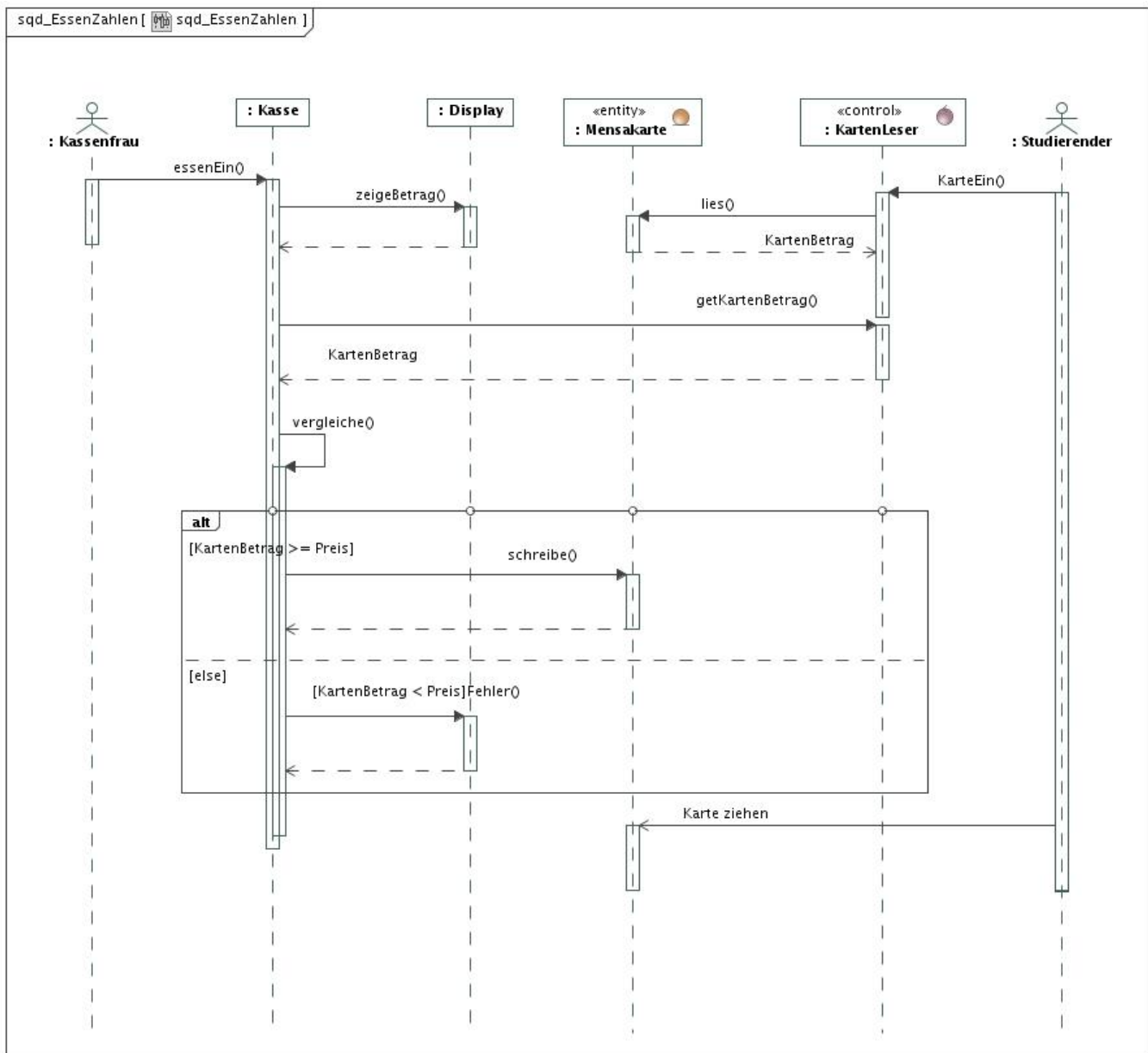
Es wurden die beiden kombinierten Fragmente loop (Schleife) und alt (Alternative, Auswahl) benutzt.

Der Akteur "Studierender" greift auf das Objekt der Klasse "Kartenleser" zu, indem er die Nachricht "KarteEin()" sendet. Der "Kartenleser" liest die "Mensakarte" über den Aufruf "lies()" und erhält als Rückgabewert den "Betrag". Der "Kartenleser" sendet die Nachricht "zeigeBetrag" an das Display.

Über die Nachricht "GeldEin()" wird der "Geldeinzug" aktiviert. Er zieht den eingelegten Schein ein und ruft seine eigene Methode "pruefeSchein()" auf. Der Return gibt TRUE oder FALSE zurück. Ist die Bedingung "[Schein erkannt]" erfüllt, ruft der "Geldeinzug" die Methode "GeldRaus()" auf und gibt den Schein zurück. Trifft die Bedingung "[Schein erkannt]" sendet er dem Display die Nachricht "zeigeBetrag()"; der Betrag wird auf dem Display angezeigt. Die Nachricht "" und danach auf die "MensaKarte" gespeichert. Da mehrere Scheine hintereinander "schreibe()" an "Mensakarte" speichert den neuen Betrag auf der Karte. Trifft die Bedingung "[Schein erkannt]" nicht zu gibt der "Geldeinzug" den Schein über seine eigene Methode "GeldRaus()" wieder zurück.

Durch das kombinierte Fragment "loop" wird angedeutet, dass mehrere Scheine eingelegt werden können. Das Fragment "alt" kennzeichnet die verschiedenen Verhaltensweisen bei erkanntem und nicht erkanntem Geldschein.

Sequenzdiagramm der Szene Essen zahlen



Dieses Sequenzdiagramm modelliert die Szene "Essen zahlen". Es bezieht sich auf den Use Case "EssenZahlen" des Use Case Diagramms "Mensa".

Quelle: <https://www.fbi.h-da.de/labore/case/uml/sequenzdiagramm.html>, besucht am 30.01.17