UML Sequenzdiagramm

Einleitung

Schon in der Einleitung zum Klassendiagramm wurde gezeigt, dass zur Laufzeit eines objektorientierten Systems sehr viele Objekte erzeugt werden. Diese sind eine Weile da und verschwinden später wieder. Während Ihrer Lebenszeit werden auf diesen Objekten Methoden aufgerufen. In der Regel führt dies zu weiteren Methodenaufrufen bei anderen Objekten. Aus einem einzigen Aufruf kann so eine ganze Kette von Aufrufen entstehen. Eine ziemlich dynamische und auf den ersten Blick verwirrende Sache. Sie können dies mit einem Unternehmen vergleichen, wo ein Projekt von vielen Mitarbeitern bearbeitet wird. Informationen fliessen zwischen diesen Mitarbeitern hin und her, verschiedene Resultate werden erzeugt und weitergegeben.

Unser Ziel ist es diese Abläufe im Überblick zu behalten. Ein Mittel dazu ist das *Sequenzdiagramm*. Es ist eines der Diagramme, welche unter dem Oberbegriff *Verhaltensdiagramme* segeln.

Bei der Arbeit mit Sequenzdiagrammen geht es nie darum, alle möglichen Abläufe eines Systems in ihrer Gesamtheit zu erfassen. Dazu eignet sich das Sequenzdiagramm nicht. Vielmehr geht es darum auf einer angemessenen Abstraktionsebene die wichtigsten oder kompliziertesten Abläufe beispielhaft darzustellen. Ein Sequenzdiagramm zeigt deshalb einen bestimmten Ausschnitt aus dem Leben des Systems.

Sequenzdiagramm im Überblick

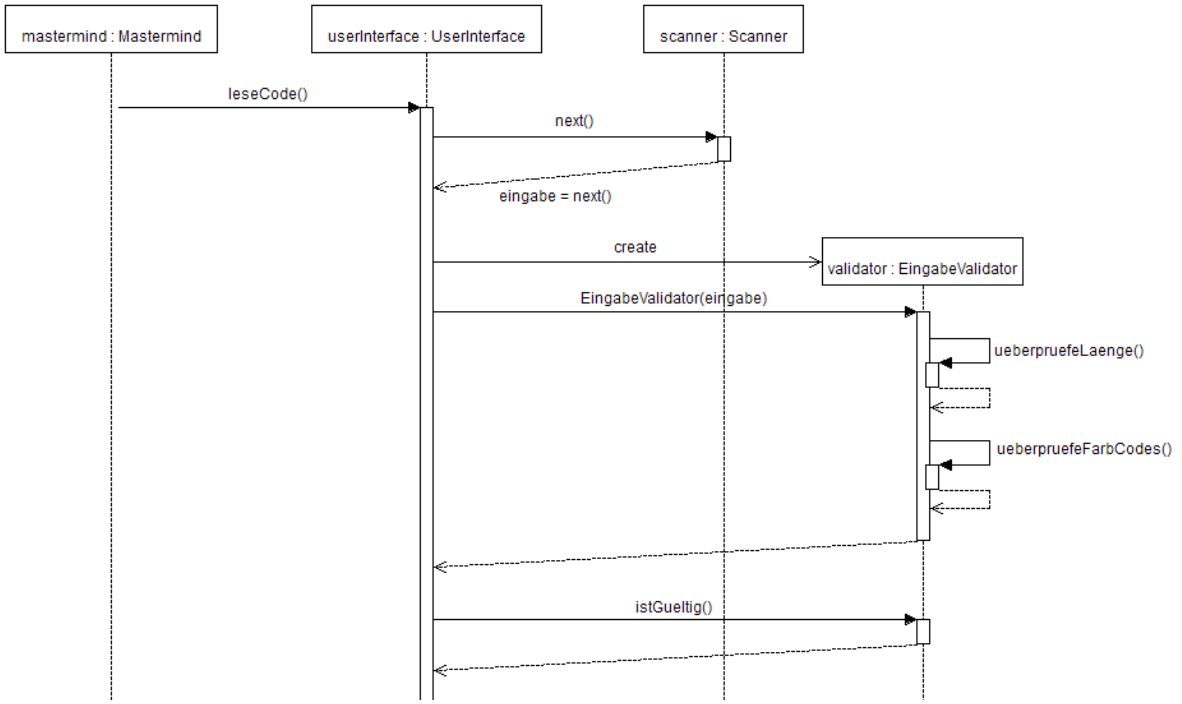
Am einfachsten ist es, ein Sequenzdiagramm auf der Basis eines Code-Ausschnittes zu erstellen. Zu diesem Zweck wird hier der Code einer klassenbasierten Version des Mastermind-Spiels verwendet. Sie finden diesen Code in den Ressourcen der Lektion 11 (MastermindBeispielCode.docx).

Als Ausgangspunkt nehmen wir den Aufruf der Methode `leseCode()` beim `UserInterface`-Objekt.

```
public class Mastermind {
    …
    public UserInterface userInterface = new UserInterface();
    …

    private void spielen() {
        geheimCode = CodeGenerator.erzeugeZufallsCode();
        Versuch versuch;
        do {
            userInterface.zeigeEingabeAufforderung();
            Code code = userInterface.leseCode();
            …
```

Auf der folgenden Seite sehen Sie ein mögliches Sequenzdiagramm, welches sich aus diesem Aufruf ergibt. Beachten Sie, dass nicht alle Details des Codes im Diagramm sichtbar sind. Es geht hier lediglich darum, einen Überblick der Elemente eines Sequenzdiagramms zu zeigen.

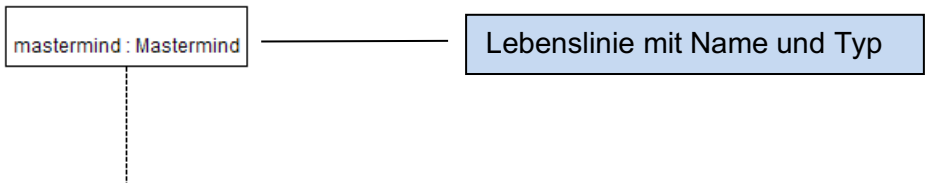


Das Sequenzdiagramm hat zwei Achsen:

- Auf der horizontalen Achse werden die beteiligten Objekte aufgelistet (dargestellt durch ein Rechteck mit dem Namen und dem Typ des Objektes). Die Reihenfolge ist dabei unbedeutend. Unterhalb von jedem Objekt hängt die Lebenslinie des Objektes (gestrichelte Linie). Zwischen den Lebenslinien werden die ausgetauschten *Meldungen*, eingezeichnet. Für unseren Gebrauch repräsentiert eine Meldung einen *Methodenaufruf*.
- Die vertikale Achse ist die Zeitachse. Die Zeit läuft von oben nach unten. Das heisst, je weiter unten im Diagramm eine Meldung steht, umso später wird sie versandt. Objekte, welche zu Beginn des dargestellten Ablaufs schon existieren, werden durch Lebenslinien am oberen Rand dargestellt. Die Lebenslinien von Objekten, welche erst im Lauf der Zeit erzeugt werden, erscheinen entsprechend weiter unten.

Lebenslinien

Objekte werden im Sequenzdiagramm als *Lebenslinien* dargestellt. Der Begriff ist verwirrend, da damit das Rechteck für das Objekt *samt* der gestrichelten Linie gemeint ist.



Die Lebenslinie zeigt den Zeitabschnitt an, während welchem das Objekt existiert.

Innerhalb der Lebenslinie werden ein *Name* und der *Typ* des Objektes in der Form

```
name: typ
```

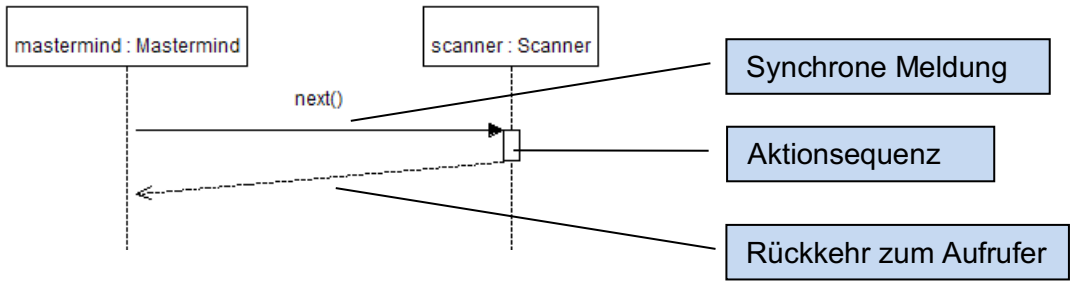
notiert.

Der Name ist optional und kann fehlen. Oft wird der Name einer Referenz, welche auf das Objekt verweist, als Name des Objektes notiert. Mit dem Typ ist in der Regel die Klasse des Objektes gemeint. Der Doppelpunkt ist insofern wichtig, als er bei fehlendem Namen der einzige Hinweis ist, dass es sich beim Rechteck um ein Objekt und nicht um eine Klasse handelt.

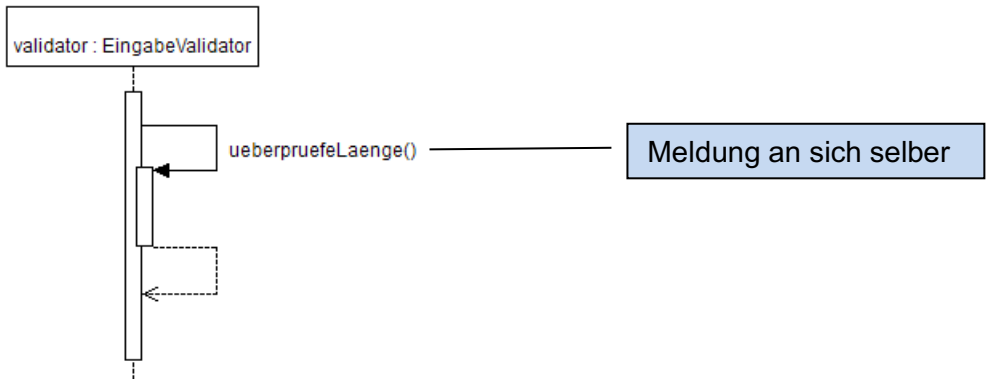
Sollen *Klassenmethoden* (statische Methoden) im Sequenzdiagramm dargestellt werden, so wird an Stelle eines Objektes die Klasse eingezeichnet

Meldungen

Meldungen zwischen Objekten / Lebenslinien werden durch Pfeile dargestellt. Wenn eine Meldung bei einem Objekt eintrifft wird dieses *aktiv*, das heisst, es führt die zur empfangenen Meldung passenden *Aktionen* aus. Im Diagramm wird der Zeitabschnitt während welchem das Objekt aktiv ist als schmales vertikales Rechteck, der *Aktionssequenz*, dargestellt.



Mehrere Aktionssequenzen können bei Bedarf verschachtelt gezeichnet werden. Dies ist dann der Fall, wenn ein Objekt eine Meldung an sich selber sendet.

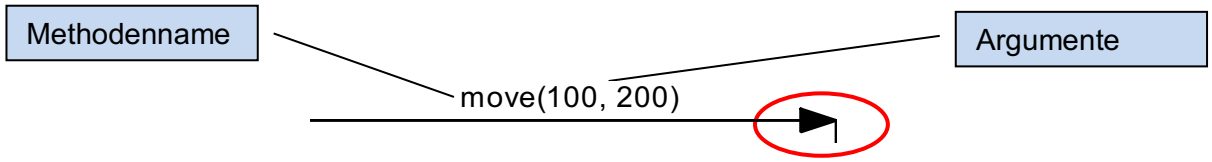


Die Aktionssequenz (auch Aktivitätsbox oder *execution sequence* genannt) ist gemäss UML-Standard optional. D.h. sie kann auch weggelassen werden.

Synchrone Meldungen (Methodenaufruf)

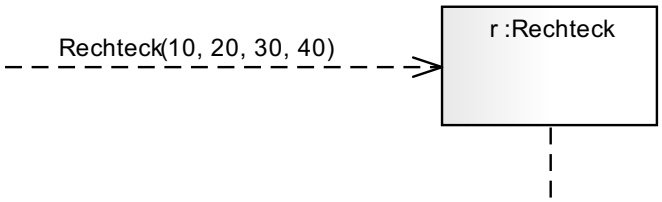
Sendet ein Objekt eine *synchrone* Meldung an ein anderes Objekt, so wartet das sendende Objekt mit weiteren Aktionen, bis das aufgerufene Objekt geantwortet hat. Dies entspricht einem ganz gewöhnlichen *Methodenaufruf*.

Der Meldungspfeil hat eine *geschlossene* Pfeilspitze (im Gegensatz zur *asynchronen* Meldung). Der Name der aufgerufenen Methode wird am Pfeil notiert. Die beim Aufruf mitgegebenen *Argumente* (Parameterwerte beim Aufruf) werden in Klammern angefügt.



Objekterzeugung

Die *Erzeugung* von Objekten ist im UML-Standard nicht ausreichend genau beschrieben. Häufig wird sie durch einen gestrichelten Pfeil angezeigt. Der Pfeil wird gemäss dem verwendeten *Konstruktor* (Klassenname und Argumente) beschriftet.



Eine andere Darstellungsart ist im Überblick zu sehen. Hier werden die Erzeugung und der Aufruf des Konstruktors durch separate Meldungen dargestellt.