

Unified Modeling Language (UML)

Kirsten Berkenkötter

Was ist ein Modell?

Warum Modellieren?

Warum UML?

Viele, viele Diagramme

UML am Beispiel

Was ist ein Modell?

Ein Modell:

ist eine abstrakte Repräsentation eines Systems, bzw.

ist eine Vereinfachung der Realität.

stellt die Kernpunkte eines Systems dar.

Abstrakt bedeutet nicht kompliziert, sondern auf das Wesentliche reduziert!

Durch Abstraktion verhindern wir, uns beim Entwurf eines Systems in Details zu verlieren!

Warum Modellieren?

Die meisten Systeme sind für Heldenprogrammierung zu groß.

Teamarbeit → Kommunikation ist notwendig

Nicht nur Programmierer, auch Manager, Vertrieb und Kunden sind in den Entwicklungsprozess involviert →
gemeinsame, verständliche Sprache ist notwendig

Das Rad muss nicht jedesmal neu erfunden werden → Dokumentation

Beschreibung des Systems:

- natürliche Sprache ist mehrdeutig und unpräzise

- Programmiersprachen sind zu präzise und nicht für alle (z.B. Manager) verständlich

- graphische Modellierung mit UML füllt diese Lücke aus

Warum Modellieren?

Software sollte:

- nützlich und nutzbar
- zuverlässig
- flexibel
- kostengünstig und
- verfügbar

sein.

Negative Gegenbeispiele:

- Ariane 5
- Therac 25
- ...

Warum UML?

Unified Modeling Language (UML)

derzeit in Version 1.5, bald 2.0

seit 1995, erfunden von Grady Booch, Jim Rumbaugh und Ivar Jacobson
mittlerweile de facto Standard

graphisch orientierte Modellierungssprache

objektorientiert

unabhängig von Entwicklungsprozessen und -methoden

neun verschiedene Diagramme, um die verschiedenen Aspekte von Softwaresystemen widerzuspiegeln

Viele, viele Diagramme

Sinn und Zweck eines Systems

Ein System sollte nützlich und nutzbar sein, d.h. es soll einen Zweck erfüllen und vom Benutzer verwendet werden können.

→ Anwendungsfalldiagramme

Struktur eines Systems

Ein objektorientiertes System besteht aus Klassen mit Attributen und Methoden, sowie Beziehungen zwischen diesen Klassen (Vererbung, Methodenaufrufe, ...).

→ Klassendiagramme

Das lauffähige Programm besteht aus Objekten mit konkreten Attributwerten, deren Beziehungen sich aus den zugehörigen Klassen ergeben.

→ Objektdiagramme

Viele, viele Diagramme

Verhalten eines Systems - Interaktionen zwischen Objekten

Objekte kommunizieren miteinander: Methodenaufrufe, Erzeugen von Objekten, ...

- Sequenzdiagramme (Schwerpunkt auf zeitlichem Ablauf)
- Kollaborationsdiagramme (Schwerpunkt auf Konstellation der Objekte zueinander)

Verhalten eines Systems - Verhalten eines Objekts

Objekte können verschiedene Zustände haben, Zustandswechsel treten auf.

- Zustandsdiagramme

Verhalten eines Systems - unabhängig von Objekten

In einem System treten viel Abläufe auf, die man allgemein beschreiben möchte (z.B. Anwendungsfälle).

- Aktivitätsdiagramme

Viele, viele Diagramme

Physikalische Struktur eines Systems - Softwarekomponenten

Die physikalischen Teile eines Systems, z.B. verschiedene Dateien, stehen in einem festgelegten Zusammenhang zueinander.

→ Komponentendiagramme

Physikalische Struktur eines Systems - Software und Hardware

Die Software läuft auf Hardware, auch da gibt es eine festgelegte Zuordnung.

→ Verteilungsdiagramme

Und in UML 2.0 gibt es natürlich noch mehr...

UML am Beispiel

MP3-Player

entweder ein Song oder eine ganze Playlist soll abgespielt werden

eine Playlist soll erstellt werden können

eine Playlist soll durch Hinzufügen oder Entfernen von Songs bearbeitet werden können

eine Playlist soll gelöscht werden können

der aktuelle Song soll auf dem Display angezeigt werden können

UML am Beispiel - Anwendungsfalldiagramm

Ein *Anwendungsfalldiagramm* stellt das System aus der Sicht des/der Nutzer(s) dar.

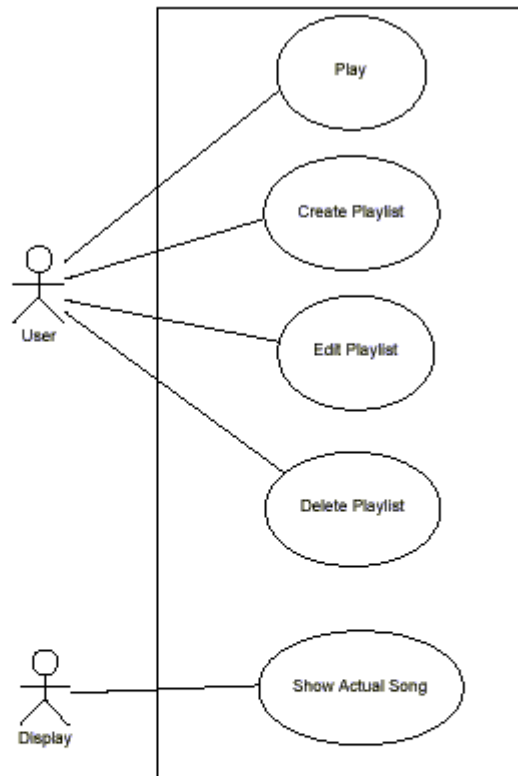
Zunächst müssen die Nutzer des Systems identifiziert werden. Das können sowohl Menschen also auch andere Systeme, z.B. Sensoren sein. Jeder Nutzer wird in UML *Akteur* genannt.



UML am Beispiel - Anwendungsfalldiagramm

Für jeden Akteur muss dargestellt werden, was das System für ihn leistet. Daraus ergeben sich die *Anwendungsfälle*. Akteur und Anwendungsfall werden durch eine Linie verbunden.

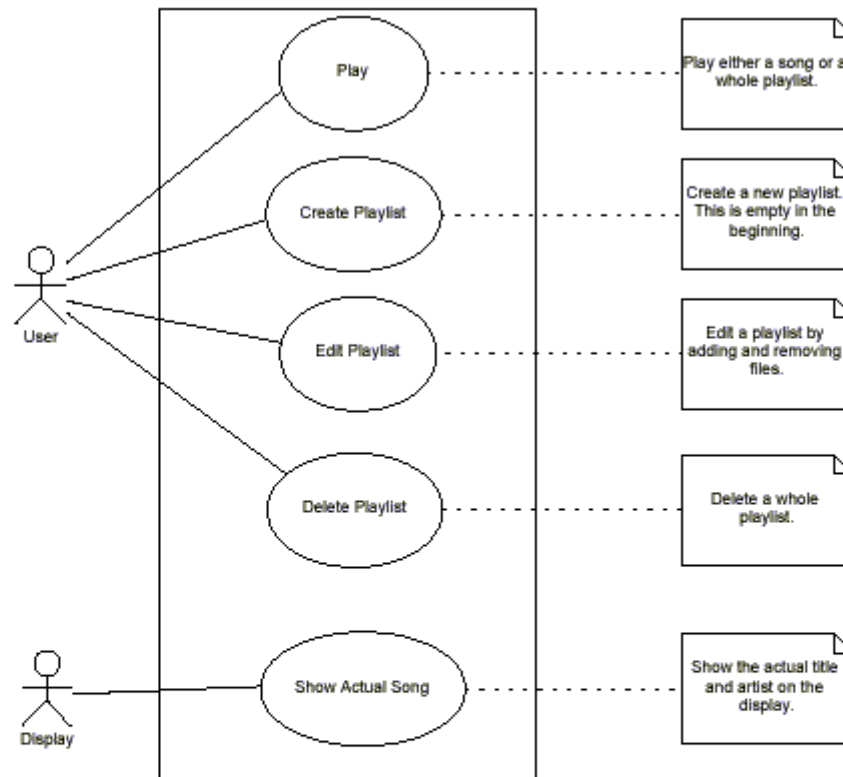
Außerdem können die *Systemgrenzen* als Rechteck eingezeichnet werden.



UML am Beispiel - Anwendungsfalldiagramm

Die Anwendungsfälle müssen beschrieben werden (z.B. in einem Kommentar oder einer eigenen Datei).

Das Anwendungsfalldiagramm ist nur die übersichtliche Darstellung.



UML am Beispiel - Klassendiagramm

Ein *Klassendiagramm* stellt die statische Struktur eines Systems dar.

Zunächst einmal müssen die *Klassen* identifiziert werden, ebenso wie ihre *Attribute* und *Operationen*. Klassen, Attribute und Operationen haben Namen. Attribute können auch Arrays sein (übliche Arrayschreibweise mit eckigen Klammern).

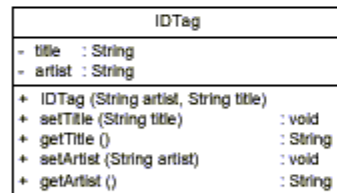
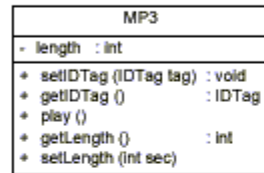
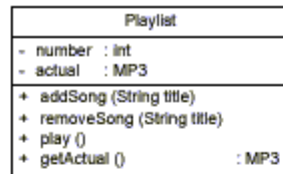
Parameter und *Rückgabewert* von Operationen werden wie folgt angegeben:

operation(param1 : Typ, param2 : Typ) : Typ

Bei Attributen und Operationen kann die *Sichtbarkeit* angegeben werden. Ohne Angabe wird *public* angenommen.

- + public
- private
- # protected
- ~ package

UML am Beispiel - Klassendiagramm



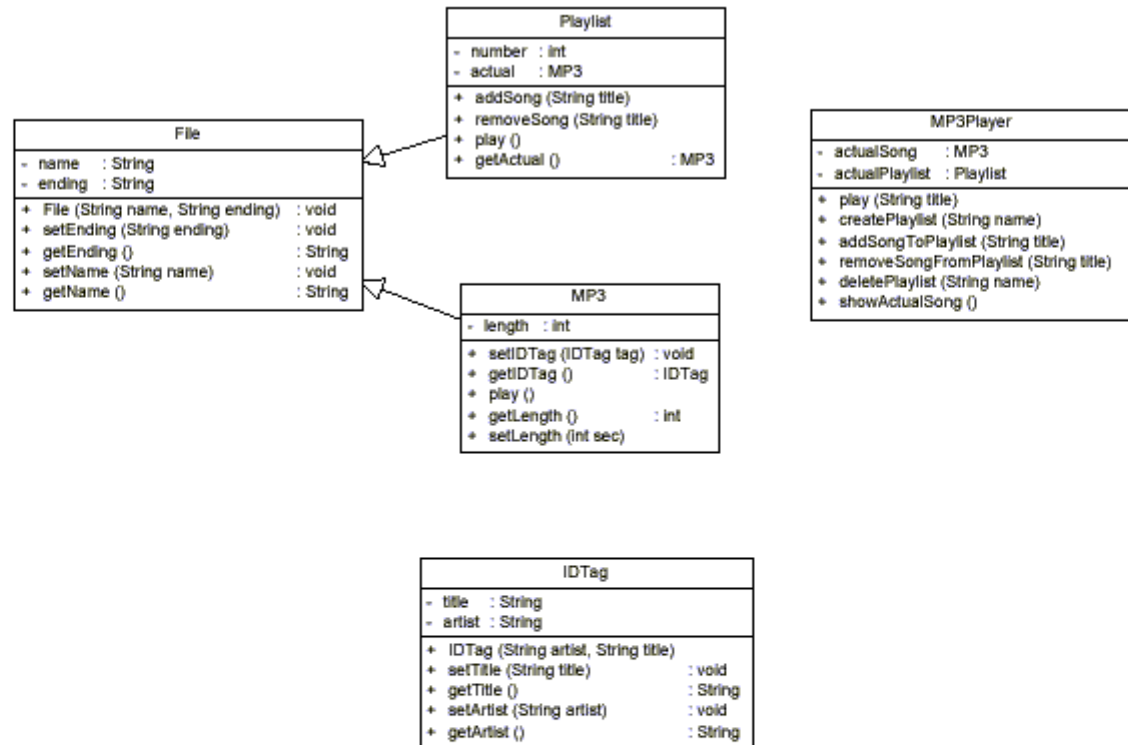
UML am Beispiel - Klassendiagramm

Vererbung wird durch eine sogenannte *Generalisierung* dargestellt.

Abstrakte Klassen

werden dadurch gekennzeichnet, dass ihr Name kursiv dargestellt wird.

(Das kommt hier allerdings nicht vor, da man an der Tafel so schlecht kursiv schreiben kann.)



UML am Beispiel - Klassendiagramm

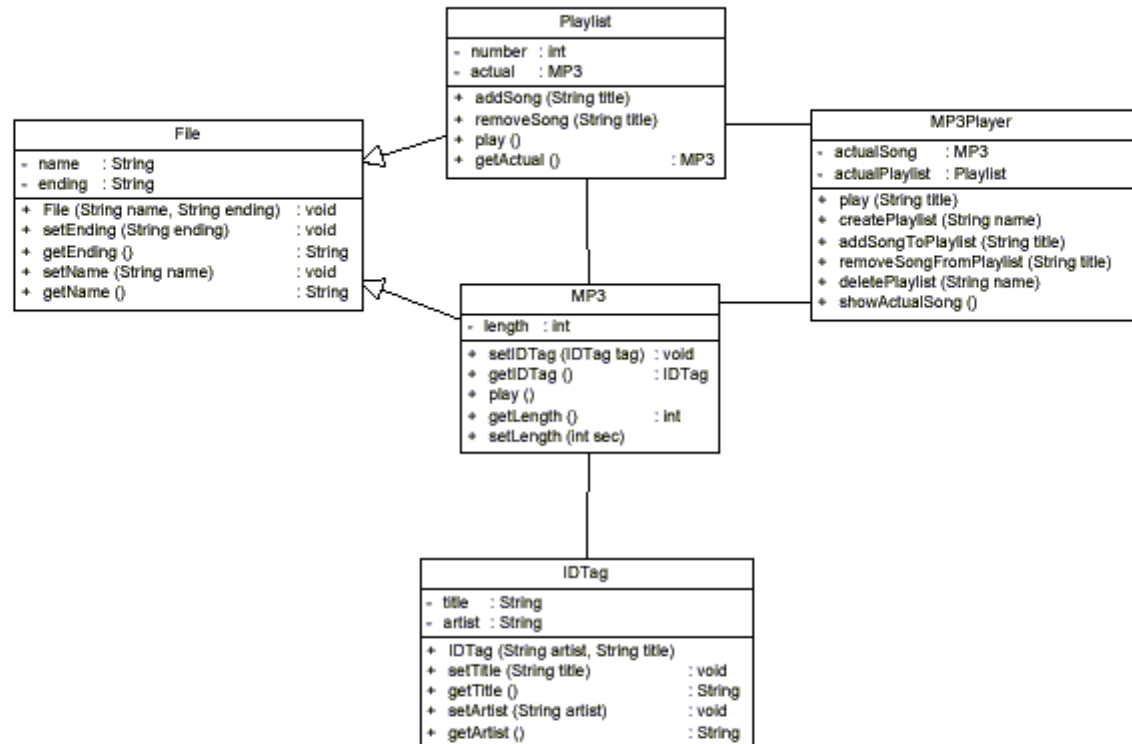
Verbindungen
zwischen Klassen
heißen *Assoziationen*.

Verbindungen sind
beispielsweise:

Objekt der Klasse A
erzeugt Objekt der
Klasse B

Objekt der Klasse A
hat ein Attribut der
Klasse B

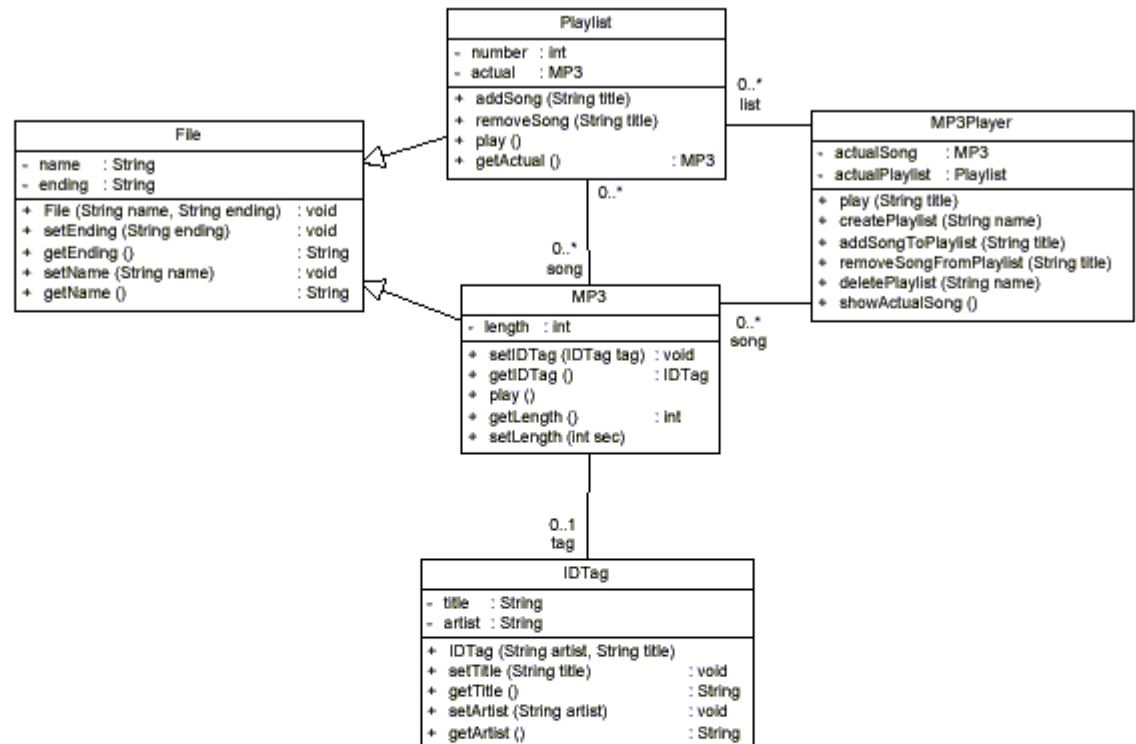
Objekt der Klasse A
ruft Methode der
Klasse B auf



UML am Beispiel - Klassendiagramm

Am Ende einer Verbindung kann ein *Rollenname* stehen, um die dortige Klasse zu identifizieren. Ohne Angabe ist der Name der Klasse der Rollenname.

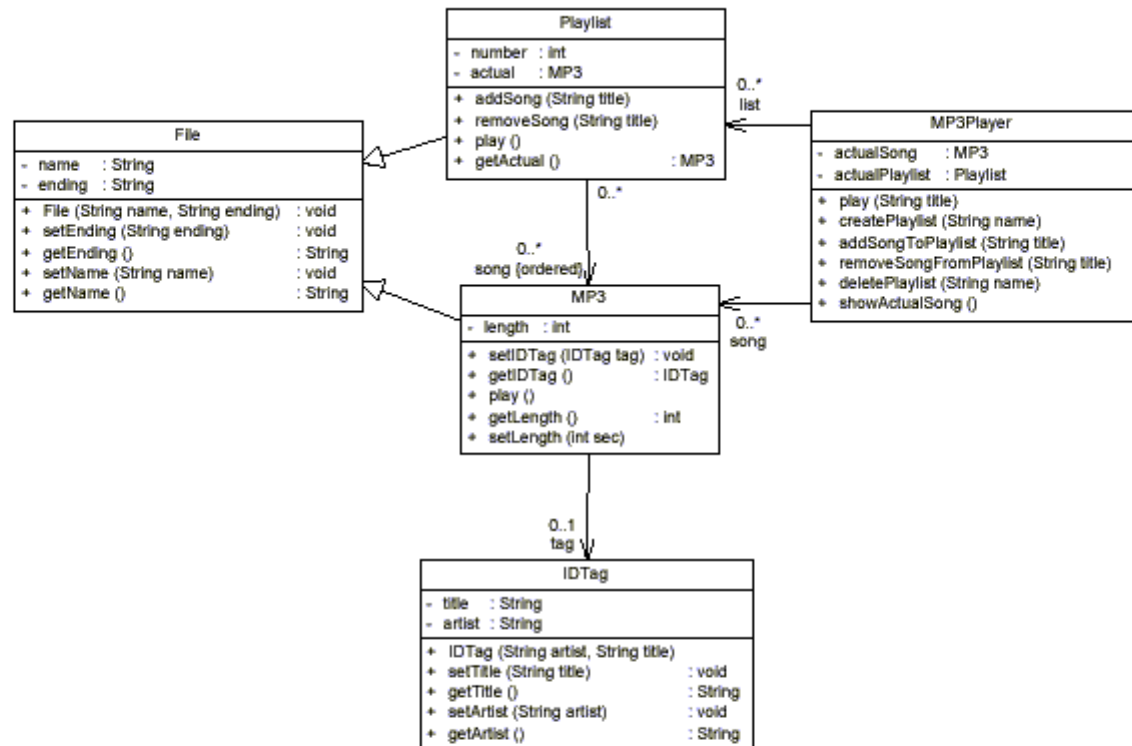
Außerdem kann angegeben werden, wieviele Objekte dieser Klasse durch die Assoziation angesprochen werden. Das ist die sogenannte *Multiplizität* (ohne Angabe immer 1).



UML am Beispiel - Klassendiagramm

Assoziationen können ausschließlich in eine Richtung gelten. Dies wird durch einen Pfeil angezeigt, der die *Navigierbarkeit* darstellt

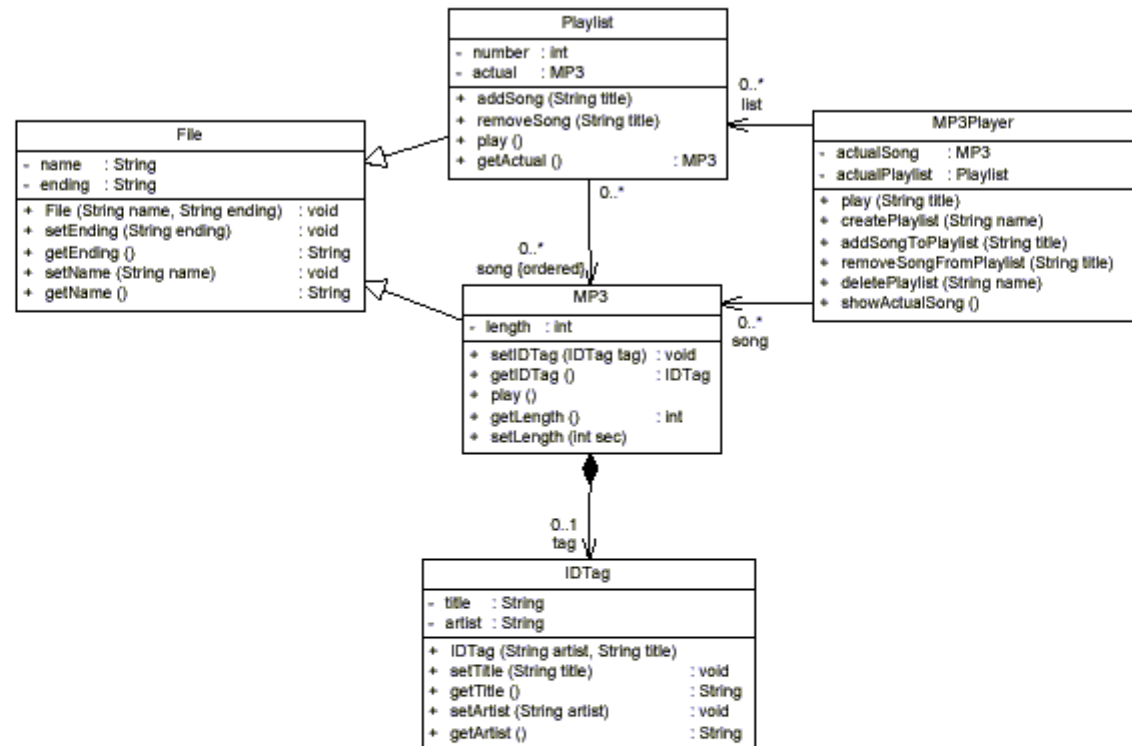
Außerdem können in geschweiften Klammern *Einschränkungen* angegeben werden.



UML am Beispiel - Klassendiagramm

Wenn eine Klasse aus anderen Klassen besteht (d.h. diese sind eigenständig nicht denkbar), wird eine *Komposition* verwendet.

Anmerkung: Es gibt auch eine schwächere Form, die *Aggregation*. Leider kennt niemand den genauen Unterschied, außer dass der Diamant am Ende der Linie weiß ist.



UML am Beispiel - Objektdiagramm

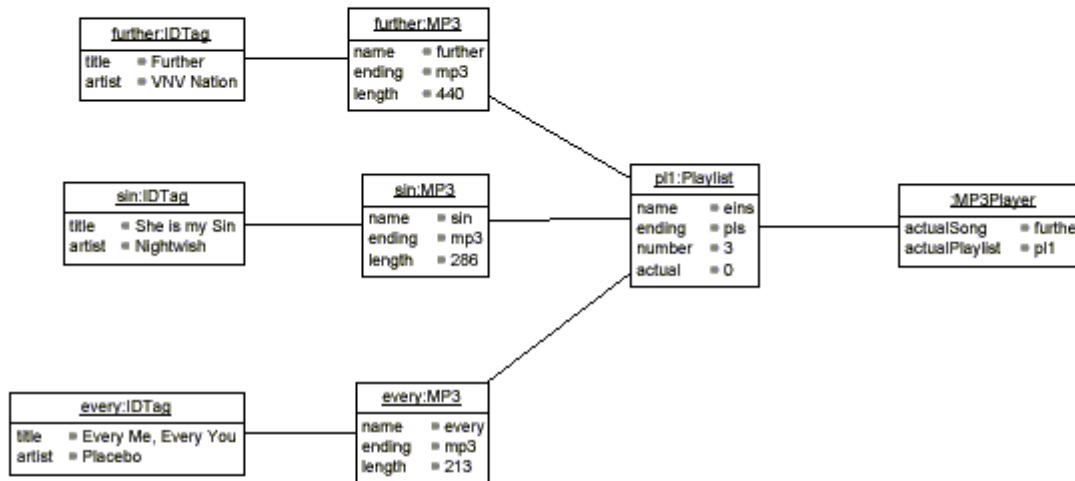
Ein *Objektdiagramm* stellt die Anordnung von Objekten (genannt Kollaboration) zur Laufzeit dar, d.h. es ist eine Art Schnappschuss des tatsächlichen Systems zu einem Zeitpunkt.

Objekte sind *Instanzen* von Klassen und haben (optional) einen Namen und immer einen *Typ* (die Klasse, zu der sie gehören).

Optional können konkrete Werte für die Attribute eines Objekts angegeben werden.

Außer Objekten können (Instanzen von) Akteuren dargestellt werden.

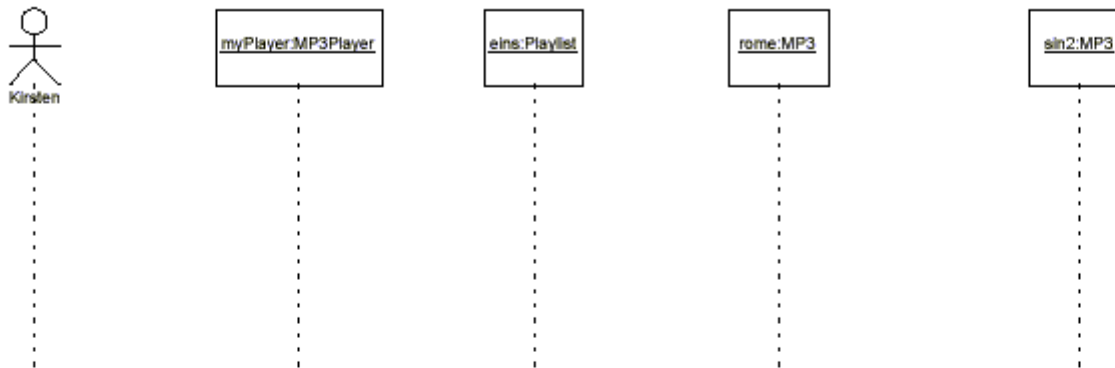
UML am Beispiel - Objektdiagramm



UML am Beispiel - Sequenzdiagramm

Ein *Sequenzdiagramm* stellt Interaktionen zwischen konkreten Akteuren und Objekten dar.

Jeder Akteur und jedes Objekt hat eine Lebenslinie.

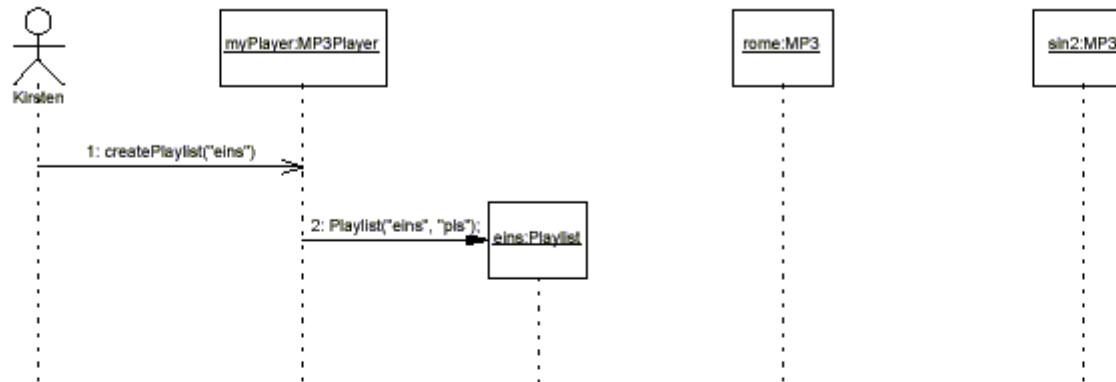


UML am Beispiel - Sequenzdiagramm

Nachrichten werden als Pfeil von einer Lebenslinie zur anderen dargestellt.

Ein Objekt kann auch sich selbst Nachrichten schicken!

Ausnahme: Die *Erzeugung* eines Objekts wird durch eine Nachricht von der Lebenslinie des erzeugenden Objekts zum Kopf des erzeugten Objekts gezogen.



UML am Beispiel - Sequenzdiagramm

2. Ausnahme: Das *Löschen* eines Objekts wird durch eine Nachricht von der Lebenslinie des löschenden Objekts zu einem Kreuz am Ende der Lebenslinie des zu löschenden Objekts dargestellt (nicht im Beispiel verwendet).

Die vertikale Anordnung der Nachrichten beschreibt auch ihre *zeitliche Abfolge*. Außerdem können sie *durchnummeriert* werden.

Wie im Klassendiagramm können *Einschränkungen* angegeben werden, z.B. um die verstreichende Zeit zwischen zwei Nachrichten anzugeben.

Bei synchronen Nachrichten, gibt es immer eine Antwort, auch wenn sie nicht explizit dargestellt wird. Bei asynchronen Nachrichten nicht.



a) Synchronous Message

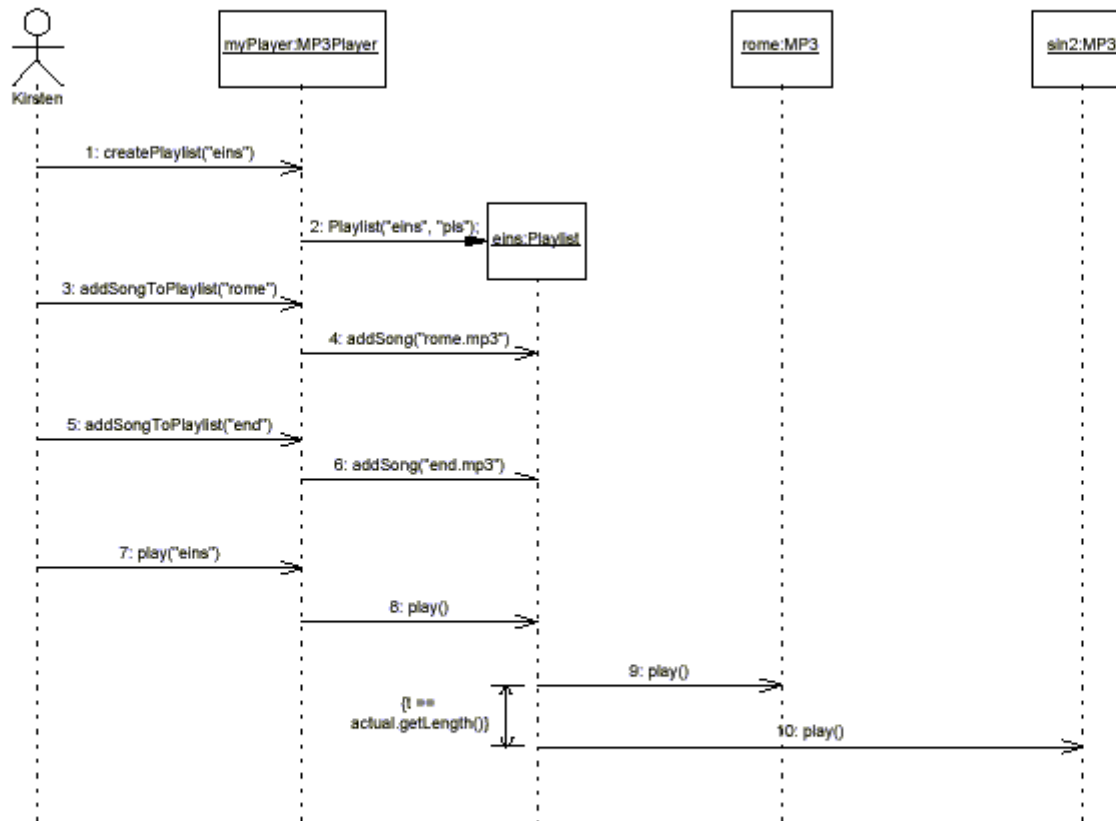


b) Return Message



c) Asynchronous Message

UML am Beispiel - Sequenzdiagramm



UML am Beispiel - Zustandsdiagramm

Ein *Zustandsdiagramm* beschreibt das Verhalten konkreter Objekte einer Klasse.

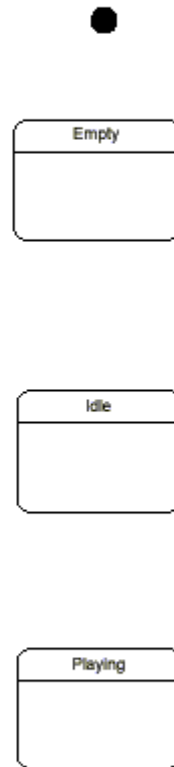
Es gibt *Zustände* und *Zustandsübergänge*.

Am Anfang ist das Objekt im *Initialzustand* (schwarzer Kuller),
bei Terminierung im *Endzustand* (schwarzer Kuller mit Kreis drumherum).



UML am Beispiel - Zustandsdiagramm

Jeder Zustand hat einen Namen.



UML am Beispiel - Zustandsdiagramm

Jeder Zustandsübergang wird mit mindestens einer der folgenden Anweisungen bezeichnet:

Bedingung, z.B. $[x < 10]$

Der Zustandsübergang kann nur stattfinden, wenn diese Bedingung erfüllt ist. Möchte man verstreichende Zeit modellieren, wird dies üblicherweise durch das Schlüsselwort *after* gekennzeichnet.

Ereignis, z.B. `play()`

Der Zustandsübergang kann nur stattfinden, wenn dieses Ereignis eintritt.

Aktion, z.B. `x = 10;`

Wenn dieser Zustandsübergang stattfindet, wird diese Aktion ausgeführt. Aktionen können z.B. Zuweisungen oder Methodenaufrufe sein.

UML am Beispiel - Zustandsdiagramm

