

## Softwareprojekt

### Übung 3

Bearbeitet von: Thore Brehmer und Jonny Lam

Hinweis: Als Quelle wurden nur die Vorlesungsfolien und  
<https://www.omg.org/spec/UML/2.5.1/PDF/#page=390&zoom=100,152,625> benutzt.

## Aufgabe 3-1: UML-Elemente

Lernziel: Wichtige UML-Modellelemente kennen, lesen und verwenden können.

Betrachten Sie die folgenden UML-Modellelemente:

~~Klassen~~ und ~~Objekte~~, ~~Attribute~~ und ~~Operationen~~, Assoziationen, Multiplicität, Generalisierung, sowie Komposition und Aggregation.

Beantworten Sie für jedes Element die folgenden Fragen:

- Was ist der Zweck bzw. die Semantik des Elements? Was sind die Beziehungen zu den anderen genannten Elementen?
- Wie wird das Element grafisch oder textuell notiert? Welches sind dabei die notwendigen Notationsbestandteile und welches die optionalen? Beachten Sie: Die meisten Modellelemente haben sehr viele optionale Notationsbestandteile!

Sie können für die Bearbeitung dieser Aufgabe die UML-Spezifikation selbst heranziehen (siehe Aufgabe 3-3). Sie dürfen aber auch andere, leichter zu lesende Quellen verwenden. Denken Sie wie immer daran, diese auch anzugeben.

a, b)

**Klassen** - Stellt die Struktur des Systems klar

Notwendig : Klassenname

Optional : Attribute und Operationen

Not: Watch

Opt: Watch

time  
push()

**Objekte** - Wie eine Klasse, jedoch gibt es ein Beispiel anstatt einer allgemeinen Aussage wieder

Notwendig : Klassenname

Optional : Werte für Attribute

Not: Watch

Opt: Watch

time: 9:00

**Attribute** - Sind Teil einer Klasse und geben Typen an

Notwendig : Attributname

Optional : Attribut Type

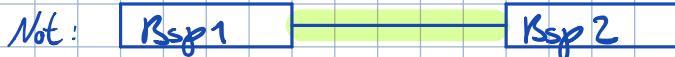
Not: Watch  
time

Opt: Watch  
time: Int

**Assoziationen** - Gibt eine Beziehung zwischen einer oder mehreren Klassen an

Notwendig : Verbindung

Optional : Navigierbarkeits Richtung, Sichtbarkeit  
Ordnung, Changeability Constraint usw...



**Multipizitäten** - Gibt die Anzahlen von Teilnehmern zwischen Beziehungen an

Notwendig : /

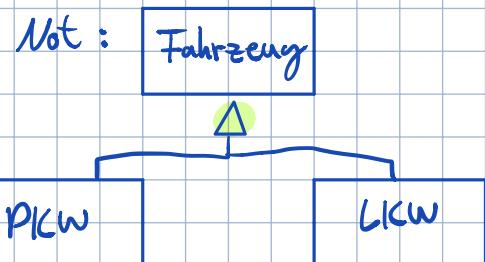
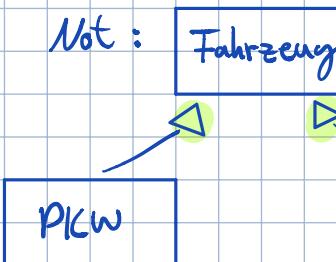
Optional : Zahl, ... , \*



**Generalisierung** - Die Vererbung von Attributen und Operationen einer Elternklasse an Kinderklassen

Notwendig : Pfeil

Optional : Verschiedene Schreibweisen



**Aggregation** - Spezialfall von Assoziationen, welche eine "ist Teil von" Hierarchie erstellt. Dabei sind die Komponenten Teil vom Aggregaten

Notwendig : weißer Diamant

Optional : /



**Komposition** - Stärkere Form von Aggregation.

Wobei die Komponenten nur existieren sofern der Aggregat existiert

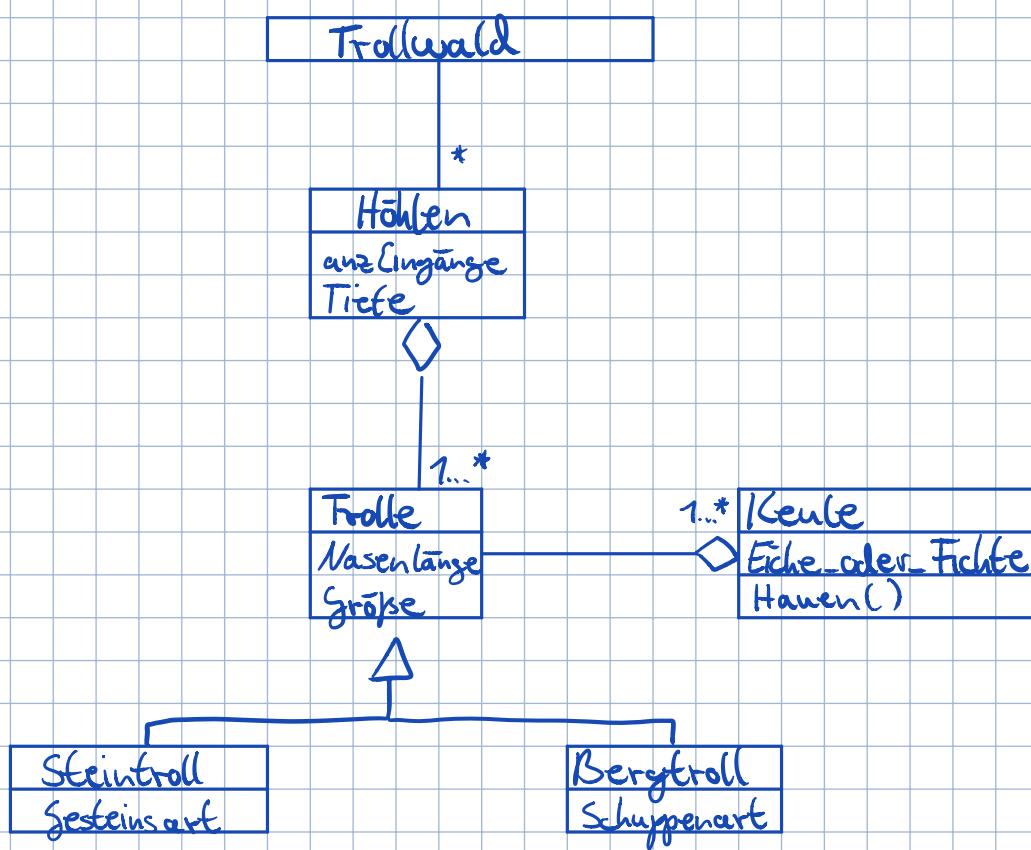
Notwendig : schwarzer Dia.

Optional : /



2

Im gefährlichen Trollwald gibt es viele Höhlen. Jede Höle hat eine bestimmte Anzahl von Eingängen und eine bestimmte Tiefe. Jede dieser Höhlen dient als Behausung für einen oder mehrere Trolle. Es gibt im Trollwald zwei verschiedene Arten von Trolle: Steintrolle und Bergtrolle. Alle Trolle wohnen in Höhlen und sind durch ihre Nasenlänge und ihre Größe unterscheidbar. Steintrolle bestehen aus einer bestimmten Gesteinsart. Bergtrolle bestehen zwar nicht aus Stein, jedoch haben sie Schuppen in den schillerndsten Farben. Jeder Troll hat natürlich eine Keule, mit der er hauen kann und die sein Eigentum ist. Tolle Trolle haben sogar mehrere solcher Keulen. Die Keulen sind entweder aus Eiche oder aus Fichte.



Annahmen:

Ohne Höhle oder Keule sind Trolle keine Trolle

Trolle können nur mit einer Keule hauen

### Aufgabe 3-3: UML-Metamodell

a)

UML-Modellelement	Name der Metaklasse	Relevante Abschnitte
Klasse	Class	11.4 und 11.8.3
Objekt	Instances	9.8
Generalisierung	Generalization	9.7 und 9.9.7
Attribut	Properties	9.5
Operation	Operations	9.6 und 9.9.11
Komposition & Aggregation	Associations	11.5
Schnittstelle	Interfaces	10.4 und 10.5.5
Implementierung (einer Schnittstelle)	InterfaceRealization	10.5.6
Assoziation	Associations	11.5

b)

Modelelement	Diagrammtyp	Metaklasse	Relevante Abschnitte
Methodenaufruf	Sequenzdiagramm	Behaviors	13.2
Interner Übergang	Zustandsdiagramm	TransitionKind	14.5.12
Externer Übergang	Zustandsdiagramm	TransitionKind	14.5.12

c) Die Knoten in Aktivitätsdiagrammen können entweder atomare Aktionen oder zusammengesetzte Aktivitäten sein. In UML sind eine Reihe von Aktionen vordefiniert. Hinweis: Kapitel 16 sollte hierbei besonders hilfreich sein. Finden Sie zu drei dieser Aktionen jeweils die Metaklasse heraus und beschreiben in eigenen Worten, was diese Aktion jeweils tut.

**Accept Event Action:** Ist eine Aktion, die darauf wartet, ob ein oder mehrere Ereignisse auftreten. Falls ein Ereignis auftritt, kann darauf geantwortet werden, z.B. wird ein Attributwert geändert, kann aber auch später erfolgen.

Quelle: Abschnitt 16.10.1

**Object Actions:** Ist für verschiedene Aktionen auf ein Objekt zuständig, kann z.B. ein Objekt erstellen oder löschen, sowie Objekte vergleichen oder auch bestimmen zu welcher Klasse ein Objekt gehört.

Quelle: Abschnitt 16.4.1

**Link Actions:** Ist für die Assoziation zuständig, kann Objekte einer Klasse zuordnen bzw. auch entfernen, oder auch überschreiben und auch andere Assoziationen zwischen Klassen erstellen.

Quelle 16.6.1

### Aufgabe 3-4★: UML-Sequenzdiagramme

- a) Was ist ein Sequenzdiagramm und wozu dient es im Kontext der Softwareentwicklung?
- b) Welche Modellelemente werden sowohl in Klassendiagrammen als auch in Sequenzdiagrammen benutzt?
- c) Was ist in der Vertikalen dargestellt?
- d) Wie wird ein Aufruf einer Methode eines Objektes dargestellt?
- e) Ist eine Nachricht das gleiche wie ein Methodenaufruf?
- f) Wie kommen die Objekte "aneinander heran", also worin besteht die Verbindung zwischen Sender-Objekt und Empfänger-Objekt, über die eine Nachricht gesendet wird?
- g) Woran ist klar zu erkennen, dass Sequenzdiagramme immer nur einen von vielen möglichen Abläufen darstellen?
- h) Geben Sie für die 15 mit Pfeilen gekennzeichneten Diagrammelemente jeweils die Bezeichnung an bzw. wofür diese stehen. Beispiel: 1 → *Diagrammname*

- a) Ein Sequenzdiagramm gibt eine dynamische Sicht auf Interaktionen von Objekten
- Es dient dazu mithilfe des Diagrammes einfacher Ideen erklären zu können
  - Außerdem zur Dokumentation (jedoch Zeitaufwendig)
  - (- Zeigt anders als beim Klassend. eine dynamische Sicht )
- b) Objekte, Methoden (Operationen)  
Nachrichten könnten als eine spezielle Art von Assoziationen gesehen werden.
- c) In der Vertikalen wird die Zeitachse und Lebenslinien von Objekten dargestellt
- d) Mithilfe eines Pfeiles, welches auf die Lebenslinie des Objektes zeigt und mit den Namen der Methode trägt
- e) Ja, da mit Nachrichten Methoden von Objekten aufgerufen werden können
- f) Aus einem horizontalen Pfeil
- g) Sofern im Modell ein Loop oder eine alt (Fallunterscheidung) vorhanden ist

h)

- 1 - Diagrammname
- 2 - Objekt Mailserver
- 3 - Actor
- 4 - Nachricht Methode create() (erstellung von EmailClient)
- 5 - Object EmailClient
- 6 - Lebenslinie vom Actor
- 7 - Nachricht Methode fetchMail (von Actor zu t6)
- 8 - Nachricht Methode readC() (von t6 auf sich selbst)
- 9 - Asynchrone Nachricht
- 10 - Synchrone Nachricht
- 11 - Aktiver Zeitraum (in welchen authenticate() bearbeitet wird)
- 12 - Antwort Nachricht (auf authenticate())
- 13 - Optional Bereich (sofern Bed.=True passiert nichts,) gehe den Bed. Pfad
- 14 - Abfrage von !(leave MessageOnServer
- 15 - Destruction (von t6)