UNIKASSEL
VERSITÄT

SOFTWARE ENGINEERING
UNIVERSITÄT KASSEL

Objektdiagramme

Jens Kosiol

Wintersemester 23/24

Überblick

- Was ist objektorientierte Modellierung und warum wollen wir das?
- Objektdiagramme
  - Rolle im Entwicklungsprozess
  - Grundlegende Syntax
- Entwurf von Objektdiagrammen aus Szenarien
  - Praktischer Enwurf für den Study-Right University-Fall
  - Designentscheidungen

Generelles Vorgehen

Stories/Scenarios/Beispiele

- Stammen vom Kunden
- So konkret wie möglich
- Grundlage für Objektdiagramme und für Tests

Objektdiagramme

konto42: Konto
kunde = "A. Muster" stand = 400

- Dienen der Kommunikation mit Kunden und Entwicklern
- Snapshot des Heap zur Programmlaufzeit
- Grundlage für Klassendiagramm

Klassendiagramme

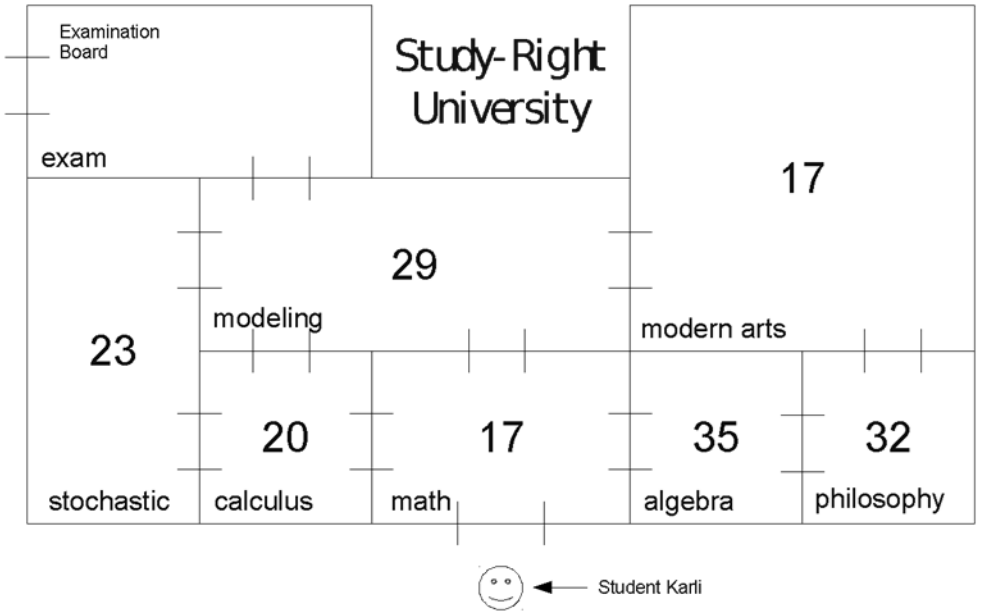
- Dienen der Kommunikation zwischen Entwicklern
- Muster für Datenstruktur

Konto
kunde: String stand: float

Code

```
public class Konto {
    public String kunde;
    public float stand;
}
```

Vorlesungsbeispiel: Study-Right University



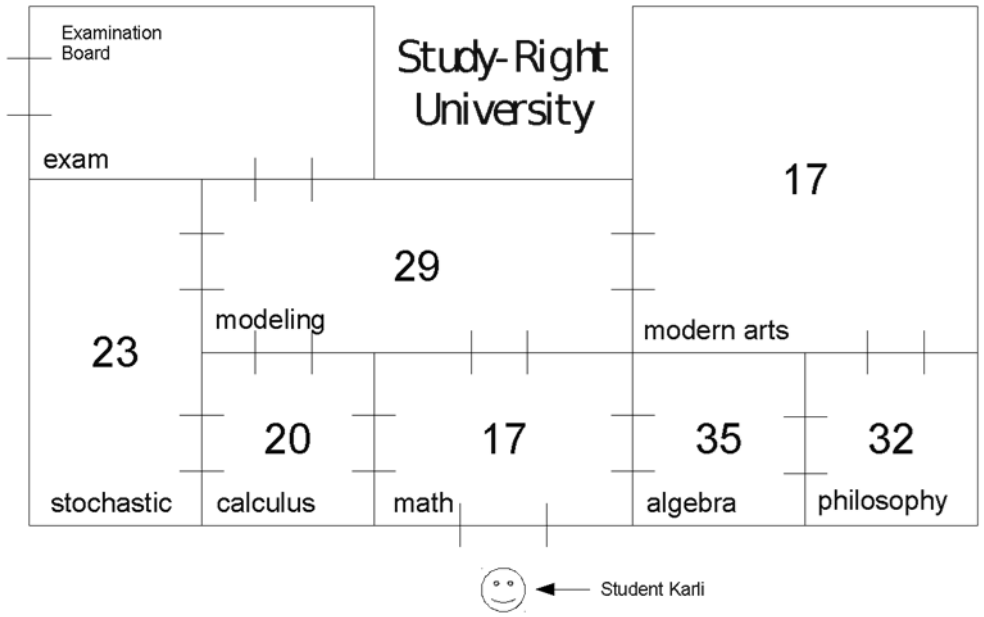
Aufgabe:

- Wegesuche (vom Mathe- zum Examensraum)

Bedingungen:

- Jeder Credit Point kostet einen Motivationspunkt
- 214 Motivationspunkte gegeben und 214 Credit Points zu erreichen
- Modulabhängigkeiten (Türen zwischen Räumen)
- Mehrfachbelegung möglich (bei Mehrfachbetreten eines Raumes wird jeweils ein anderes Modul unterrichtet)

Gutes oder schlechtes Modell?




```
public class StudyRight {
    int [] roomCredits = {17, 20, 23, 29, 17, 32, 35, 0};
    String [] topics = {"math", "calculus", "stochastic",
    "modeling", "modern arts", "philosophy",
    "algebra", "exam"};
    int [][] doors = {{0, 1, 0, 1, 0, 0, 1, 0},
                      {1, 0, 1, 1, 0, 0, 0, 0},
                      {0, 1, 0, 1, 0, 0, 0, 0},
                      {1, 1, 1, 0, 1, 0, 0, 1},
                      {0, 0, 0, 1, 0, 1, 0, 0},
                      {0, 0, 0, 0, 1, 0, 1, 0},
                      {1, 0, 0, 0, 0, 0, 0, 0},
                      {0, 0, 0, 1, 0, 0, 0, 0}};
    int [] mandatoryRooms = {1, 3};
    int studPos = -1;
    int examPos = 7;
    float motivation = 214.0;
    int [] hasMandatoryTopic = {};
    …
}
```

Gutes oder schlechtes Modell?

```
public class StudyRight {
    int [] roomCredits = {17, 20, 23, 29, 17, 32, 35, 0};
    String [] topics = {"math", "calculus", "stochastic",
    "modeling", "modern arts", "philosophy",
    "algebra", "exam"};
    int [][] doors = {{0, 1, 0, 1, 0, 0, 1, 0},
                      {1, 0, 1, 1, 0, 0, 0, 0},
                      {0, 1, 0, 1, 0, 0, 0, 0},
                      {1, 1, 1, 0, 1, 0, 0, 1},
                      {0, 0, 0, 1, 0, 1, 0, 0},
                      {0, 0, 0, 0, 1, 0, 1, 0},
                      {1, 0, 0, 0, 0, 0, 0, 0},
                      {0, 0, 0, 1, 0, 0, 0, 0}};
    int [] mandatoryRooms = {1, 3};
    int studPos = -1;
    int examPos = 7;
    float motivation = 214.0;
    int [] hasMandatoryTopic = {};
    …
}
```

- Wer findet den Fehler?
- Wie entfernt/ergänzt man in diesem Modell eine Tür?
- Wie fügt man in diesem Modell einen Raum hinzu?

Objektorientierte Modellierung

- Objektorientierte Modellierung (OOM) gibt es als Begriff etwa seit dem Aufkommen der OO-Analyse (ca. 1988).
- Objektorientierung dient als durchgängiges Paradigma für die gesamte Softwareentwicklung.

Aspekte der OO-Modellierung

- Ontologischer Aspekt: Die Welt wird als zusammengesetzt aus unterscheidbaren, identifizierbaren und im einzelnen beschreibbaren Objekten aufgefasst.
- Struktureller Aspekt: Objekte können in Beziehungen und Hierarchien stehen und aus einfacheren Objekten zusammengesetzt sein.
- Verhaltensaspekt: Objekte werden sowohl durch statische Merkmale (= Attribute) als auch durch dynamische (= Operationen) beschrieben.

Zentrale Konzepte der objektorientierten Modellierung

- Klasse (auch „Objekttyp“):
  - beschreibt die strukturellen und verhaltensmäßigen Merkmale einer Menge gleichgearteter Objekte. Diese Merkmale werden durch Attribute und Operationen (Methoden) beschrieben.
- „Semantische“ Standardbeziehungen (auf Klassen und Objekten) wie z.B.
  - Instanziierung
  - Generalisierung/Spezialisierung
  - Gruppierung (Aggregation oder Komposition)
  - Assoziation

Objektdiagramme

„An object diagram is a graph of instances, including objects and data values. A static object diagram is an instance of a class diagram; it shows a snapshot of the detailed state of a system at a point in time. The use of object diagrams is fairly limited, mainly to show examples of data structures.”

[OMG UML Specification, Version 1.5; Hervorhebung J.K.]

Rolle von Objektdiagrammen

abstrakt

konkret

Klassisch

- Abgeleitet aus dem Klassendiagramm
- Eingeschränkte Rolle im Entwicklungsprozess: Präsentieren von Beispielen für die Datenstruktur (zur Laufzeit)

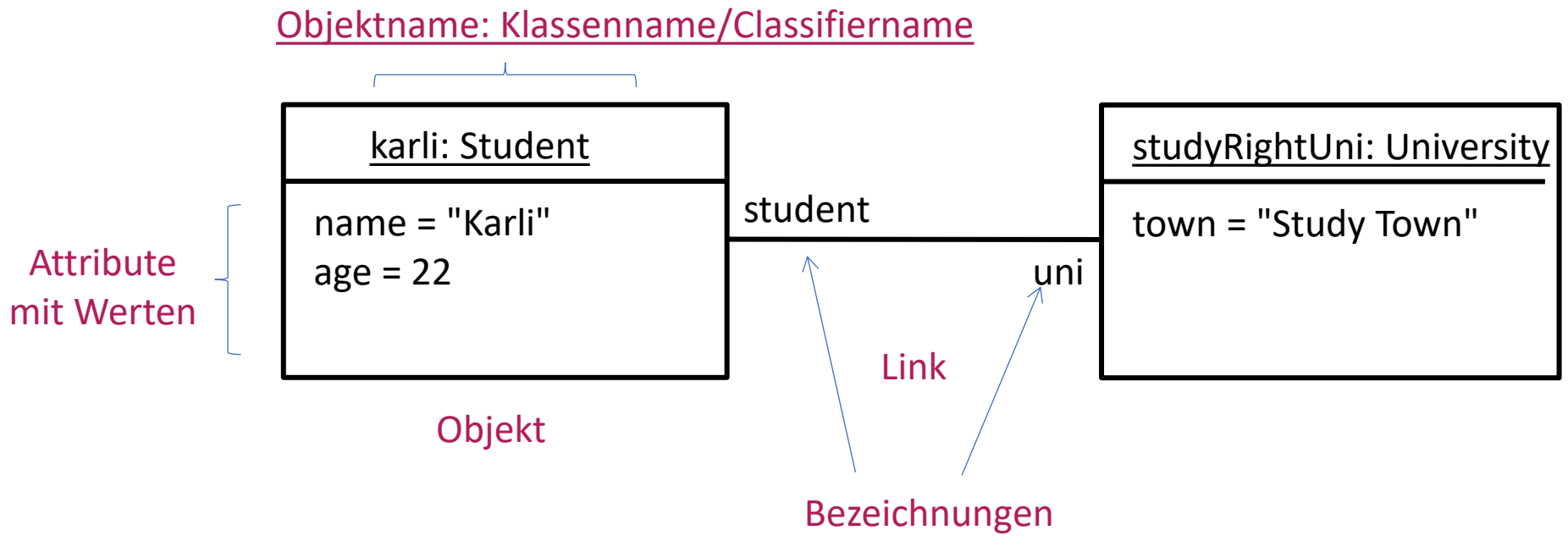
In dieser Vorlesung

- Entwicklungsschritt auf dem Weg zum Klassendiagramm
- Zentrale Rolle im Entwicklungsprozess
  - Präsentieren von Beispielen für die Datenstruktur (zur Laufzeit)
  - Hilfestellung beim Entwurf des Klassendiagramms
  - Hilfsmittel in der Kommunikation mit Kunden/Stakeholdern und zwischen Entwicklern
  - Grundlage für Tests

konkret

abstrakt

Syntax von und Bezeichnungen in Objektdiagrammen



- Sowohl Objektname als auch Klassen/Classifiername sind **optional** und werden unterstrichen; manchmal wird auch zusätzlich **Fettdruck** verwendet!
- Die Classifiernamen dürfen eine kommaseparierte Liste sein.
- Klassennamen beginnen mit **großem Buchstaben**!
- Attributnamen, Objektnamen und Rollennamen beginnen mit **kleinem Buchstaben**!
- Bei Bezeichnungen auf **Einzahl/Mehrzahl** achten!

Vom Szenario zum Objektdiagramm

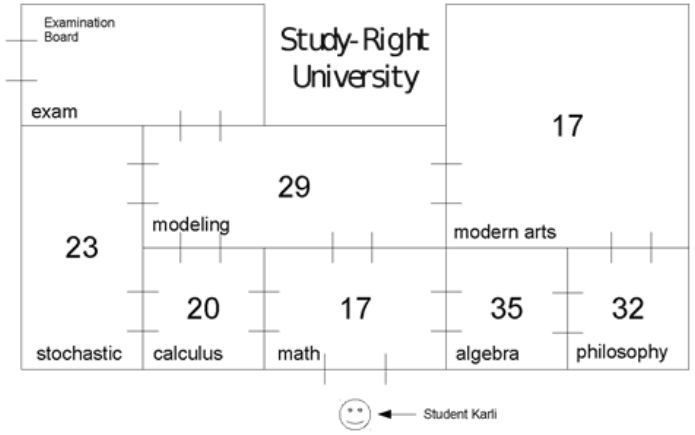
- Welche konkreten Objekte tauchen im Beispielszenario (wann) auf?
  - Häufig als Nomen im Text
  - Das werden die Objekte im Objektdiagramm.
- Welche Eigenschaften haben diese Objekte?
  - Häufig auch als Nomen, Zahlenwerte usw. im Text
  - Das werden die Attributwerte der zugehörigen Objekte.
- Zwischen welchen Objekten bestehen welche Beziehungen?
  - Zum Beispiel durch Subjekt–Objekt–Beziehungen in Sätzen erkennbar
  - Daraus werden die Links zwischen den Objekten.
- Zusätzliche Quellen von Information berücksichtigen (z.B. informelle Diagramme)
- Filter: Welche dieser Informationen muss ich überhaupt repräsentieren, um den Zweck des Programms erfüllen zu können? Welche kann ich weglassen?

Beispiel: Vom Szenario zum Objektdiagramm

Titel: Karli versucht zu graduieren und scheitert

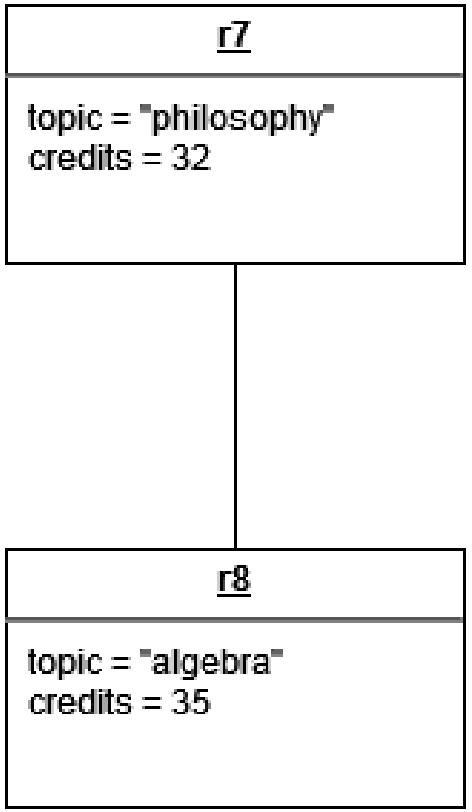
Beschreibung:

1. Ausgangssituation: Karli möchte einen Abschluss an der Study-Right University erwerben. Zur Zeit braucht er dafür 214 Credit Points (CP). Karli startet mit 0 CP und einer Motivation von 214 Punkten. Er befindet sich außerhalb der Universität.

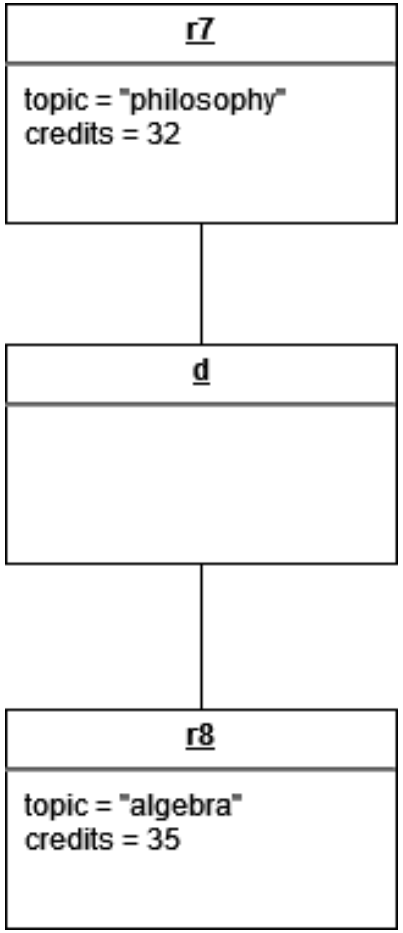


- Wo überall wird hier wichtige Information repräsentiert?
- Welche Objekte tauchen auf? Welche davon müssen wir modellieren?
- Welche Attribute haben diese Objekte?
- Welche Beziehungen gibt es zwischen den Objekten?

Reifizierung

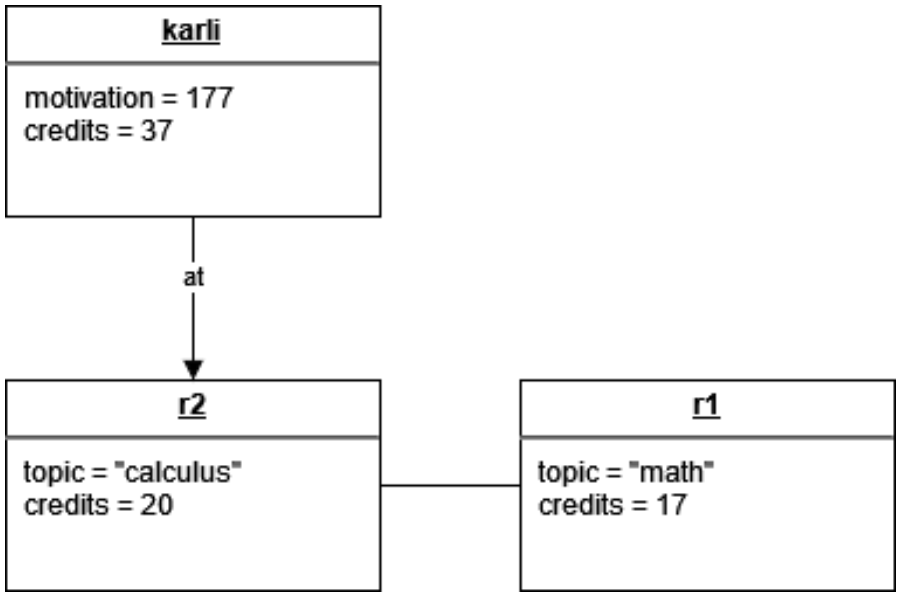


oder

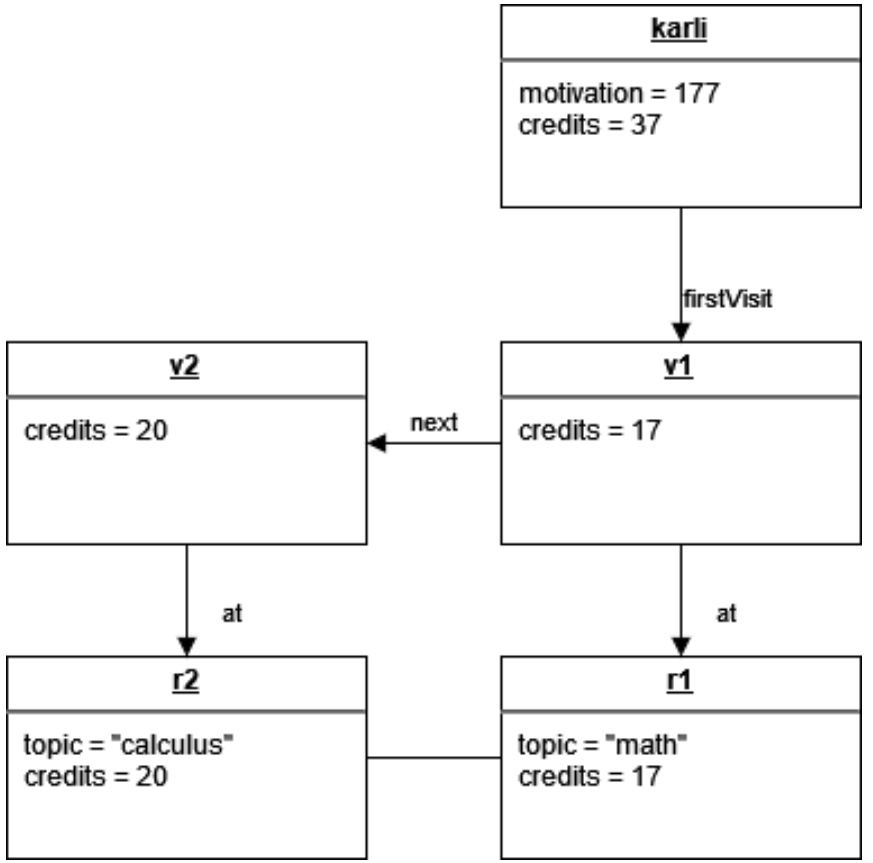


?

Was modellieren wir alles?

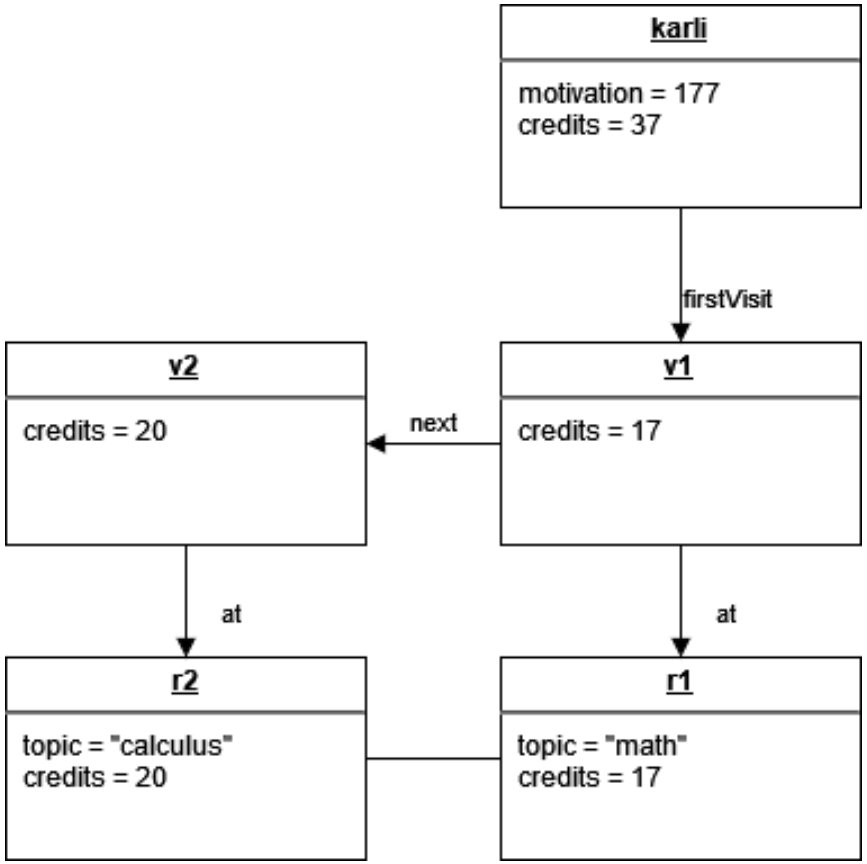


oder



?

Referenzen oder Attribute?



oder

?

Welche Objektdiagramme?

Gegeben eine Menge Szenarien/User Stories, welche Objektdiagramme erstellen wir?

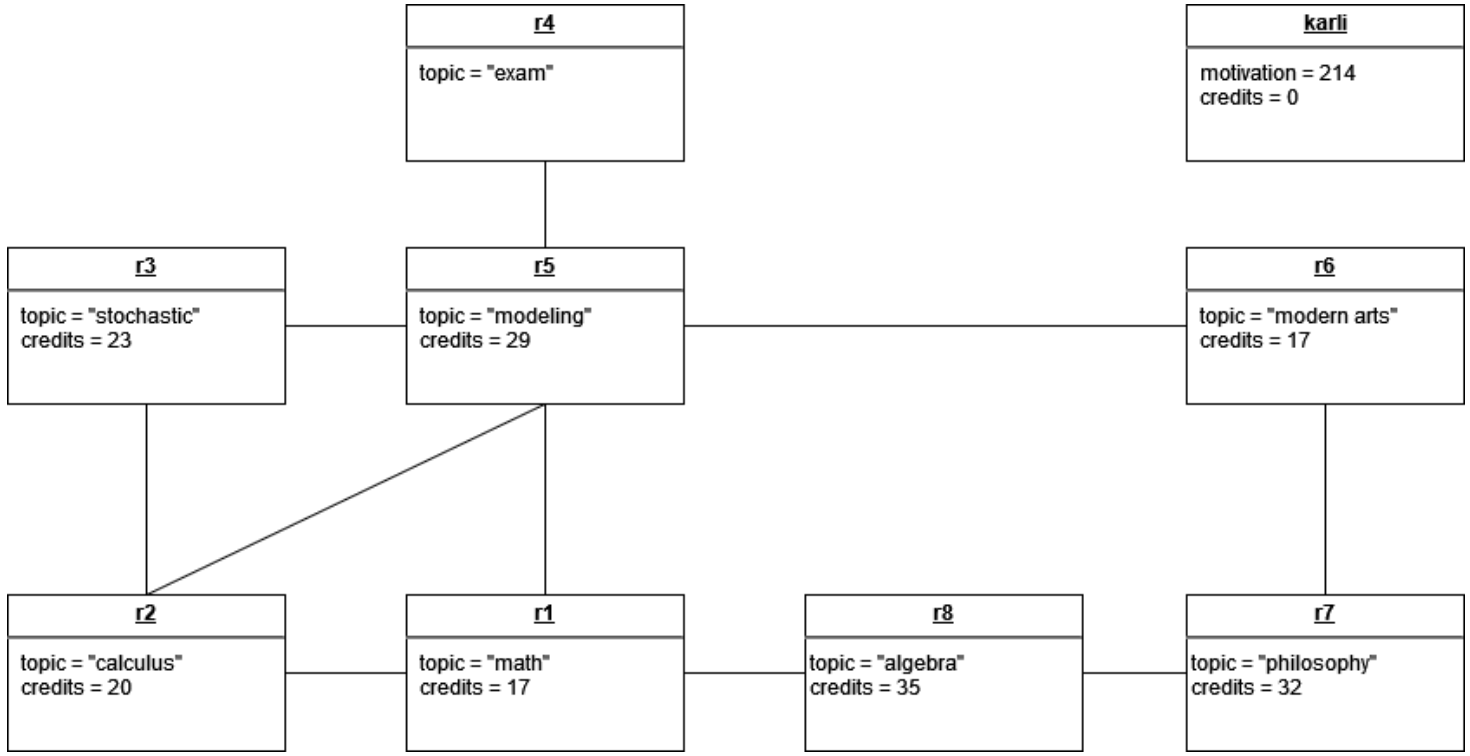
- Für jedes Szenario ein Objektdiagramm für die Ausgangs- und eins für die Endsituation
- Objektdiagramme für die Zwischensituationen können hilfreich oder nötig sein:
  - als Hilfe bei der Entwicklung von Algorithmen;
  - wenn in Zwischensituationen Objekte und/oder Links auftauchen, die weder in der Ausgangs- noch in der Endsituation vorhanden sind.

Vom Szenario zum Objektdiagramm: Ausgangssituation

Titel: Karli versucht zu graduieren und scheitert

Beschreibung:

1. Ausgangssituation: Karli möchte einen Abschluss an der Study-Right University erwerben. Zur Zeit braucht er dafür 214 Credit Points (CP). Karli startet mit 0 CP und einer Motivation von 214 Punkten. Er befindet sich außerhalb der Universität.

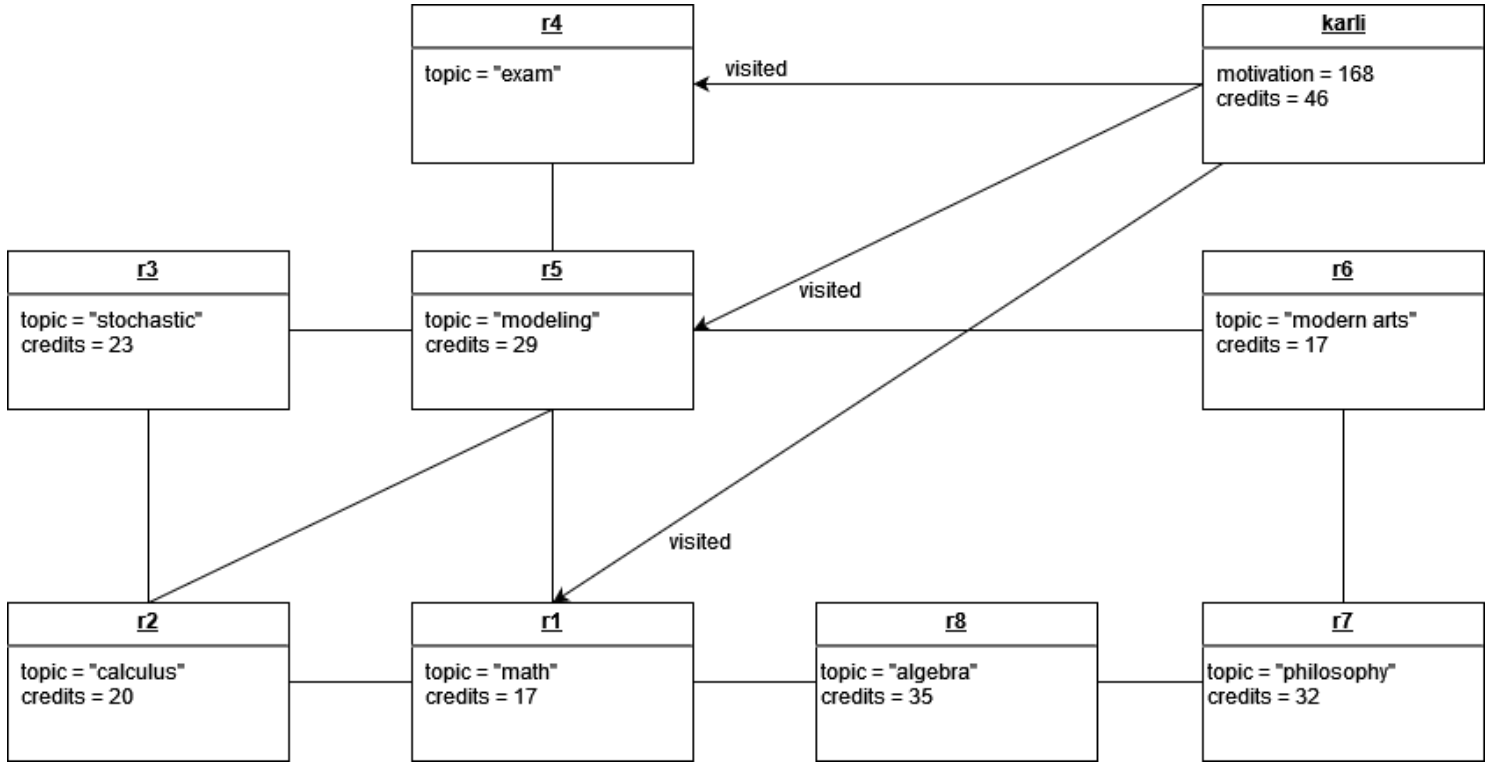


Vom Szenario zum Objektdiagramm: Schlusssituation

Titel: Karli versucht zu graduieren und scheitert

Beschreibung:

4. Schlusssituation: Karli betritt den Raum *examination*. Er wird aufgefordert nachzuweisen, welche Kurse er belegt hat und wie viele CP erworben. Da Karli über weniger als 214 CP verfügt, darf er nicht graduieren, muss die Universität verlassen und neu anfangen.

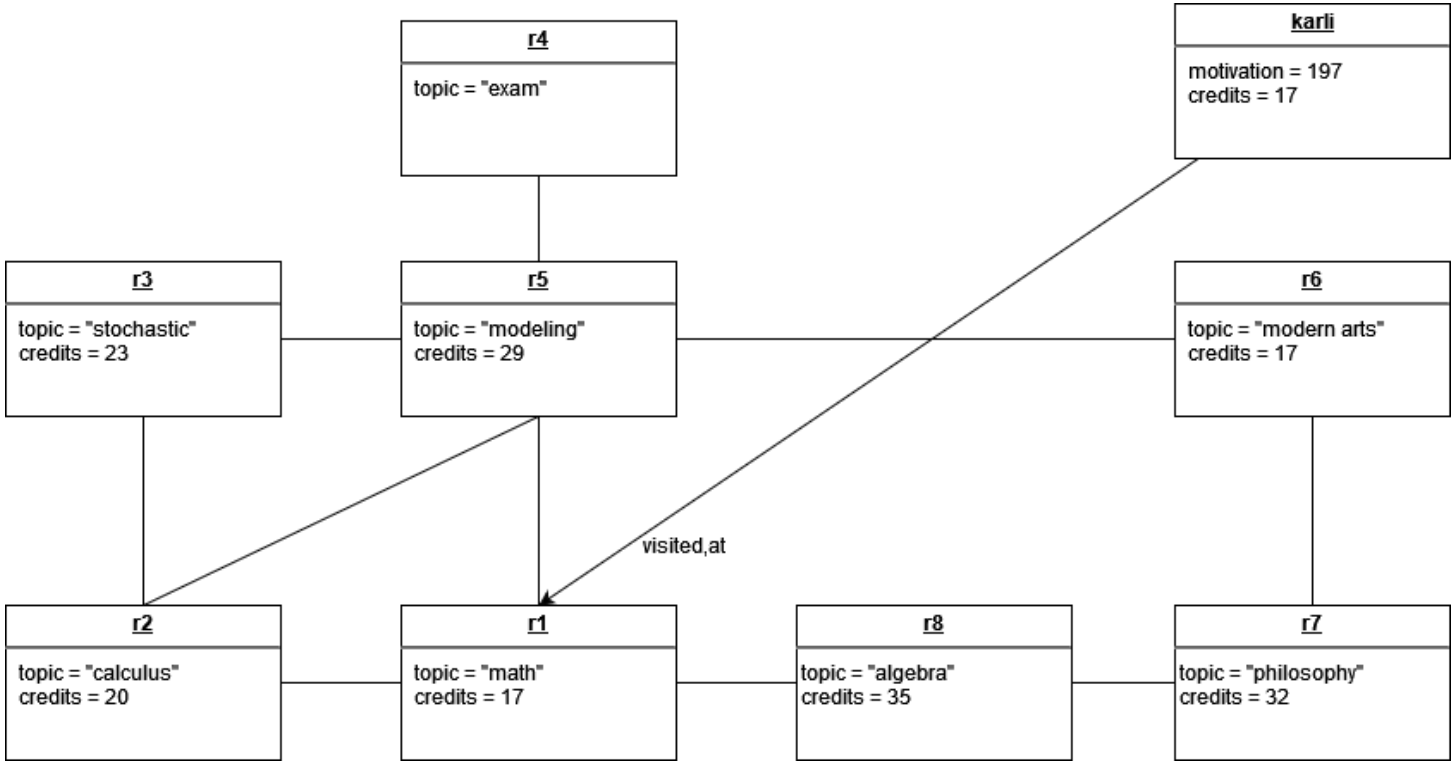


Vom Szenario zum Objektdiagramm: Zwischenschritt I

Titel: Karli versucht zu graduieren und scheitert

Beschreibung:

2. Karli betritt den Raum *math*, nimmt an der Vorlesung teil, erhält automatisch 17 CP und verliert 17 Motivationspunkte. Er hat nun 17 CP und 197 Motivationspunkte.



Vom Szenario zum Objektdiagramm: Zwischenschritt II

Titel: Karli versucht zu graduieren und scheitert

Beschreibung:

3. Karli betritt den Raum *modeling*, nimmt an der Vorlesung teil, erhält automatisch 29 CP und verliert 29 Motivationspunkte. Er hat nun 46 CP, 168 Motivationspunkte und an den Vorlesungen *Mathematik* und *Modellierung* teilgenommen.