KURSSTUFE (4-stündig)

Im vierstündigen Kernfach erfolgt eine tiefer gehende, erweiterte und systematischere Behandlung von Inhalten, die auf ein vertieftes Verständnis und Reflexion abzielt. Entsprechend unterscheiden sich die gestellten Anorderungen im Schwierigkeits- und Komplexitätsgrad sowie in Bezug auf die Selbstständigkeit bei der Bearbeitung. Projektarbeit zu fächerübergreifenden und fächerverbindenden Zusammenhängen spielt eine größere Rolle. Neben die Vertiefung tritt eine Erweiterung der Themen, die den Facettenreichtum des Faches Informatik verdeutlicht.

Die Gliederung erfolgt anhand der Leitideen:
Information und Daten
Algorithmen und Daten
Problemlösen und Modellieren
Sprachen und Automaten
Wirkprinzipien von Informatiksystemen
Informatik und Gesellschaft

1. LEITIDEE „INFORMATION UND DATEN“

Information ist neben Energie und Materie eine der zentralen Erscheinungsformen der realen Welt. Unsere Informations- und Wissensgesellschaft basiert auf der automatisierten Verarbeitung von Informationen. Dazu müssen Informationen durch geeignete Daten repräsentiert werden. Durch Interpretation werden daraus wieder Informationen gewonnen.
Die Digitalisierung erlaubt eine einheitliche Darstellung gänzlich verschiedenartiger Informationen. Digitale Daten lassen sich auf einfache Weise übertragen und weiterverarbeiten. Digitalisierte Daten können zur Einsparung von Speicherplatz und Übertragungszeit komprimiert werden.

Die Schülerinnen und Schüler können

- zwischen Information und Daten unterscheiden;
- Information darstellen und Daten interpretieren;
- Daten in eine digitale Form übertragen;
- Vor- und Nachteile von Datenkompression darlegen;
- ein Verfahren zur Datenkompression beschreiben.

- *Datei, Dokument, Interpretationsvorschrift, zugehöriges Programm*
- *Bit und Byte, Hexadezimalsystem*
- *einfache Formate für Text und Grafik*
- *Codierung*
- *ein Kompressionsverfahren*

2. LEITIDEE „ALGORITHMEN UND DATEN“

Zentral für die Informatik ist die automatische Verarbeitung von Daten. Ein Algorithmus ist die präzise Beschreibung der notwendigen Verarbeitungsschritte. Die elementaren Bausteine von Algorithmen werden an geeigneten Problemen erarbeitet und verwendet. Die Verwendung abstrakter Datentypen zeigt, wie Abstraktion den Entwurf von Algorithmen erleichtert. Zur Realisierung der Problemlösung auf einem Rechner werden die Algorithmen in einer Programmiersprache implementiert. Die Testphase ermöglicht, Ursache, Wirkung und Tragweite von Fehlern zu erkennen. Exemplarische Effizienzbetrachtungen zeigen Möglichkeiten, Algorithmen zu optimieren. Weder Programmiersprache noch Entwicklungsumgebung dürfen dabei die zentrale Rolle im Unterricht spielen.

Die Schülerinnen und Schüler können

- elementare und abstrakte Datentypen sowie Strukturen zur Ablaufsteuerung anwenden;
- Benutzerschnittstellen mit einfachen Komponenten gestalten;
- Algorithmen und Datentypen entwerfen und in Programme umsetzen;
- Techniken zur Modularisierung einsetzen;
- einfache Algorithmen auf Effizienz und Korrektheit analysieren;
- Grenzen des Rechnereinsatzes darlegen.

- *Variablenkonzept, Geltungsbereich*
- *einfache und strukturierte Datentypen*
- *rekursive Datenstrukturen (einfach verkettete Liste, Binärbaum)*
- *abstrakte Datentypen (Schlange, Keller)*
- *Anweisung, Anweisungsfolge, Verzweigung, Wiederholung*
- *Prozeduren und Funktionen, Parameterkonzept*
- *Rekursion als Lösungsprinzip*
- *einfache und komplexe Sortier -und Suchverfahren*
- *Rechnen mit endlicher Stellenzahl*
- *kritisches Laufzeitverhalten, praktische Grenzen der Berechenbarkeit*
- *Untersuchung zur Effizienz von Algorithmen inkl. theoretischer Untersuchung*

3. LEITIDEE „PROBLEMLÖSEN UND MODELLIEREN“

Der Prozess zur Lösung eines hinreichend großen Problems lässt sich gliedern in Analyse, Modellbildung und Implementierung. Ein Modell ist eine abstrahierte Beschreibung eines Systems. Modellieren beziehungsweise Modellbildung ist die Erstellung eines solchen Modells. Abhängig von der Problemstellung kommen verschiedene Modellierungsarten zur Anwendung. Die Implementierung von Modellen sorgt dafür, dass diese veranschaulicht, überprüft und bewertet werden können. In der Softwareentwicklung werden zur Problemlösung zunehmend standardisierte Analyse- und Entwurfsmethoden eingesetzt, die auch dem evolutionären Charakter des zu entwickelnden Produktes Rechnung tragen. Sie ermöglichen es bei geeigneter Auswahl, im Unterricht Modellierung als Lerninhalt und als Methode in einem Projekt zu behandeln.

Die Schülerinnen und Schüler

- kennen grundlegende Prinzipien beim Problemlösen und können diese anwenden;
- können ein Problem arbeitsteilig im Team lösen;
- können den Problemlöseprozess strukturieren;
- können zu realen Problemen ein passendes Modellierungsverfahren auswählen und ihre Wahl begründen;
- können eine Lösung dokumentieren, präsentieren und vertreten;

Objektorientierte Modellierung

- kennen Konzepte der objektorientierten Modellierung;
- können Beziehungen zwischen Klassen und die Kommunikation zwischen Objekten analysieren und beschreiben;
- können ein Modell entwerfen und implementieren;

Datenmodellierung

- können Datenmodelle entwerfen und in ein relationales Datenbankschema übertragen;
- können Abfragen in Datenbanken formulieren;

Zustandsorientierte Modellierung

- können Abläufe mit Hilfe von Zustandsdiagrammen modellieren.

- *Top-down- und Bottom-up-Vorgehensweise*
- *Modularisierung*
- *Problemanalyse, Modellbildung, Implementierung und Bewertung der Lösung*
- *Objektorientierte Modellierung:*
  Geheimnisprinzip
  Objekt, Klasse, Attribut, Methode
  Zustand und Verhalten eines Objektes, Lebenszyklus
  Vererbung, Polymorphie
  UML-Klassendiagramme
- *Datenmodellierung:*
  ER-Modell, Beziehungstypen, Normalisierung
  SQL-Abfragen
- *Zustandsmodellierung:*
  Zustand, Übergang, Zustandsdiagramm

4. LEITIDEE „SPRACHEN UND AUTOMATEN“

Sprachen dienen der Kommunikation zwischen Menschen (natürliche Sprache), aber auch der Mensch-Maschine- und der Maschine-Maschine-Kommunikation (formale Sprachen wie z. B. Programmiersprachen oder Netzwerkprotokolle) und genügen gewissen Regeln zur Bildung von Wörtern und Sätzen. Diese Regeln werden durch Grammatiken und Syntaxdiagramme formalisiert. Erst die Repräsentation von Informationen in einer formalen Sprache macht eine Verarbeitung durch Automaten möglich. Automaten werden in vielen Lebensbereichen eingesetzt (z. B. Fahrkartenautomaten, DVD-Rekorder, usw.). In der Informatik werden Automaten durch Zustandsmodellierung beschrieben. Ihr Einsatzbereich reicht vom Erkennen korrekt gebildeter Wörter einer formalen Sprache bis zur Steuerung von Robotern. Formale Charakterisierungen der Automaten ermöglichen es, prinzipielle Grenzen der Berechenbarkeit aufzuzeigen.

Die Schülerinnen und Schüler

- können zwischen Syntax und Semantik unterscheiden;
- kennen den syntaktischen Aufbau einer formalen Sprache und können einfache formale Sprachen in Syntaxdiagrammen und Grammatiken darstellen;
- können endliche Automaten zur Syntaxprüfung regulärer Sprachen einsetzen;
- können die Grenzen der endlichen Automaten und der algorithmischen Berechenbarkeit aufzeigen.

- *Automaten (endliche erkennende Automaten)*
- *Theoretische Grenzen der Berechenbarkeit (Turingmaschine)*
- *Syntaxdiagramme, Grammatiken*
- *reguläre Sprachen*

5. LEITIDEE „WIRKPRINZIPIEN VON INFORMATIK-SYSTEMEN“

In vielen Lebensbereichen unserer Gesellschaft werden komplexe Informatiksysteme verwendet. Um solche Systeme kompetent zu nutzen, ist ein grundlegendes Verständnis ihres Aufbaus und ihrer Funktionsweise erforderlich. Dazu gehören wesentlich die Organisation großer Datenmengen auf Rechnern, die Kommunikation zwischen Rechnern und die Abläufe innerhalb eines Rechners.
Datenbanksysteme unterstützen das Beschreiben, Bearbeiten, Speichern, Wiedergewinnen und Auswerten umfangreicher Datenmengen.
In lokalen und globalen Netzen wird Informationsaustausch organisiert und Kommunikation ermöglicht. Die dabei anfallenden komplexen Aufgaben werden in aufeinander aufbauende Schichten gegliedert, die unabhängige Teilaufgaben erledigen.
Zum Verständnis der Wirkungsweise eines Rechners gehören Kenntnisse über das Betriebssystem, die Übersetzungsvorgänge zwischen unterschiedlichen Sprachebenen und das Prinzip der Interpretation von Maschinenbefehlen durch den Prozessor.

Die Schülerinnen und Schüler

- kennen den prinzipiellen Aufbau und die Wirkungsweise von Datenbanksystemen;
- kennen Grundlagen der Rechnerkommunikation;
- können das Zusammenspiel der Protokollschichten am Beispiel eines Internetdienstes erläutern;
- gewinnen Einsicht in den Aufbau und die Prinzipien der Arbeitsweise des Rechners;
- können das Zusammenwirken von Rechenwerk, Steuerwerk und Speicher erläutern.

- *Datenbankmodell: Tabellen, Abfragen*
- *Client-Server-Prinzip*
- *Protokoll, Adressierung, einfaches Schichtenmodell: Anwendungsschicht, Transportschicht, Vermittlungsschicht, Netzwerkschicht*
- *Betriebssystem, Compiler, Maschinensprache*
- *Prinzip des Von-Neumann-Rechners*

6. LEITIDEE „INFORMATIK UND GESELLSCHAFT“

Informatiksysteme dienen oft als Grundlage für weitreichende Entscheidungen. Die Zuverlässigkeit der dabei gelieferten Ergebnisse ist abhängig von der Güte der Daten, ihrer fehlerfreien Bearbeitung und ihrer Integrität. Durch die einheitliche Darstellung sowie die globale Vernetzung sind auch unerwünschte Eingriffe von Seiten Dritter möglich. Die einfache Möglichkeit, bestehende auch verteilte Daten zu verknüpfen, birgt die Gefahr einer missbräuchlichen Nutzung. Nur mit Kenntnissen grundlegender informatischer Konzepte und Zusammenhänge lassen sich global vernetzte Systeme verantwortlich einsetzen sowie Chancen und Risiken ihrer Nutzung beurteilen.

Die Schülerinnen und Schüler

- kennen die geschichtliche Entwicklung der Rechenmaschinen und Informationstechnik im Überblick;
- kennen Aspekte der Datensicherheit;
- können moderne Verschlüsselungsverfahren erläutern, beurteilen und anwenden;
- haben Einblick in grundlegende Rechte und Gesetze des Datenschutzes;
- entwickeln ein Bewusstsein für rechtliche und ethische Fragen der Nutzung von Information und Software;
- gewinnen Einsicht in die Verantwortung beim Entwurf und beim Einsatz informationsverarbeitender Systeme.

- *Spuren im Netz, Angriffe aus dem Netz, Schutzmaßnahmen*
- *Verschlüsselungsverfahren (symmetrisch und asymmetrisch), digitale Signatur, Schlüsselmanagement*
- *informationelle Selbstbestimmung, Datenschutzgesetz*
- *Respektierung geistigen Eigentums*
- *Wirtschaftliche und soziale Folgen durch den Einsatz von Informatiksystemen*
- *Verlagerung von Entscheidungen vom Menschen auf Maschinen*

29. Informatik

29.1 Schriftliche Abiturprüfung

29.1.1 Allgemeine Hinweise

Bearbeitungszeit: **240** Minuten

Hilfsmittel:
- siehe 29.2
- Nachschlagewerke zur deutschen Rechtschreibung

Der Fachlehrerin, dem Fachlehrer werden die Aufgabenstellungen

A: eine Aufgabe mit dem Schwerpunkt **Objektorientierte Modellierung und Programmierung**

sowie drei Aufgaben **B1**, **B2** und **B3** mit verschiedenen Schwerpunkten aus den folgenden Themengebieten vorgelegt:

B1: eine Aufgabe mit dem Schwerpunkt **Datenbanken** inklusive Verschlüsselung und Datenschutz
B2: eine Aufgabe mit dem Schwerpunkt **Automaten und formale Sprachen**
B3: eine Aufgabe mit dem Schwerpunkt **Abstrakte Datentypen** inklusive erweiterter Algorithmen/Rekursion

Die Fachlehrerin, der Fachlehrer wählt aus Gruppe B von den drei vorgelegten Aufgaben **zwei** Aufgaben aus.

Die Schülerin, der Schüler

- bearbeitet die **Aufgabe A** und die **beiden** ausgewählten **Aufgaben aus der Gruppe B;**
- vermerkt auf der Reinschrift, welche Aufgaben sie/er bearbeitet hat;
- ist verpflichtet, die Vollständigkeit der vorgelegten Aufgaben vor Bearbeitungsbeginn zu überprüfen (Anzahl der Blätter, Anlagen usw.).

29.1.2 Gegenstand der schriftlichen Prüfung

Die folgenden Themen des Bildungsplans sind **nicht** Gegenstand der schriftlichen Prüfung:

- Rechner (von-Neumann-Rechner)
- Rechnernetze

29.2 Hilfsmittel

Der im jeweiligen Kurs eingeführte **wissenschaftliche Taschenrechner (WTR)** mit dem mitgelieferten Handbuch.
Hierzu sind die Ausführungen in der Anlage des Erlasses des Kultusministeriums vom 26.02.2014 (Az.: 36/45-6624.03-P/234) zu beachten.
Vor Prüfungsbeginn ist sicherzustellen, dass alle Speicherinhalte auf den wissenschaftlichen Taschenrechnern der Schülerinnen und Schüler gelöscht sind.

Die Arbeit an einem PC ist nicht gestattet.

29.3 Auf die gültigen Einheitlichen Prüfungsanforderungen (EPA) unter http://www.kmk.org/fileadmin/veroeffentlichungen_beschluesse/1989/1989_12_01_EPA_Informatik.pdf wird verwiesen.

Programmierung

- lineare Datentypen
- nicht-lineare Datentypen

Aufgabe

- Projekt „Datentypen"
- Objekt/Klasse „Schüler"

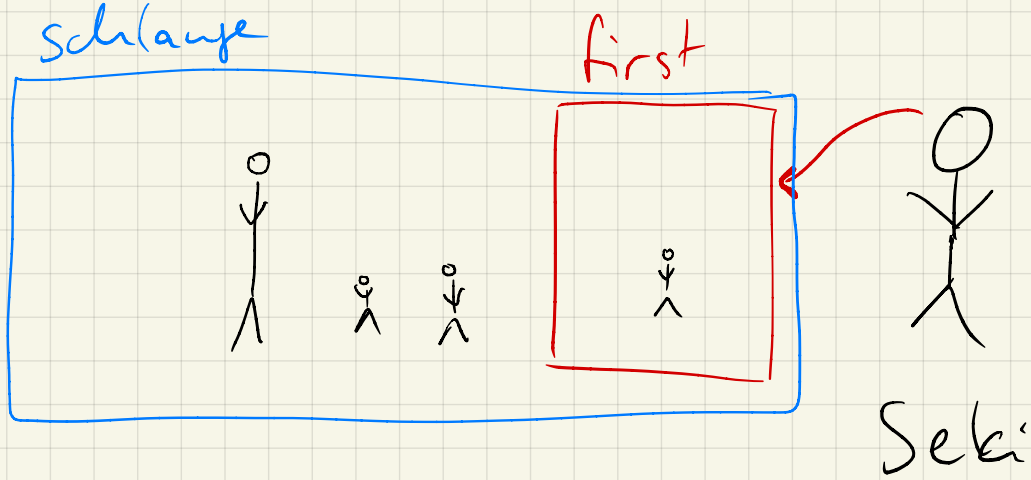
Schüler
name: String männlich: boolean klasse: int
Schüler (name: String, männlich: boolean klasse: int) begrüßung(): void

```
main {

}
```

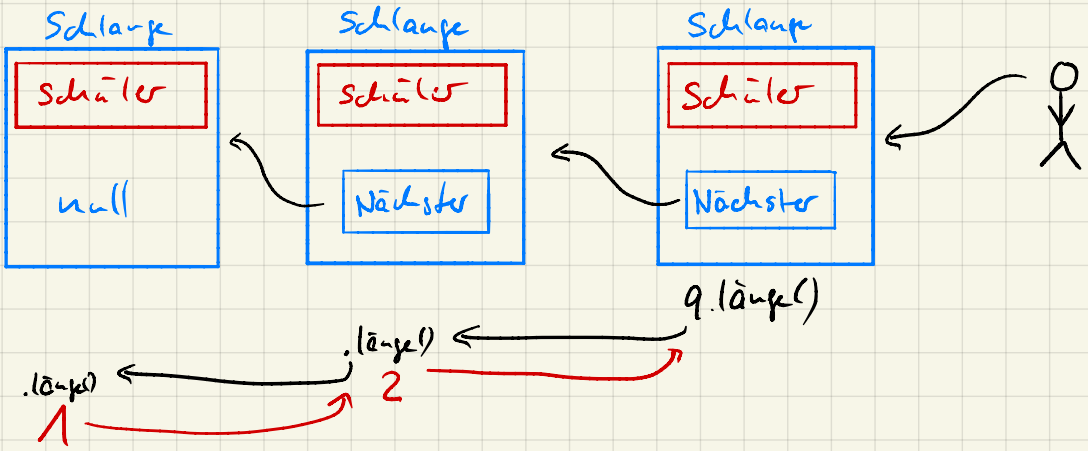
Sekretariat: FIFO (LIFO)

„Queue"



schlange.hinzufügen(Schüler)
schlange.abarbeiten()
schlange.istBelegt()

first.begrüßen();



abarbeiten: 1. Schüler begrüßen

2. vorrücken

q = nächster

LinkedList

Listen

Arrays bieten die Möglichkeit, viele Daten vom selben Typ strukturiert abzuspeichern. Sie haben aber den Nachteil, dass sie in ihrer Größe fest sind und später nicht mehr erweitert werden können. Demgegenüber können Listen auch erweitert werden.

1. LinkedList

Die am einfachsten zu erweiternde Liste ist eine *LinkedList*. Hierbei wird neben dem eigentlichen zu speichernden Wert/Objekt auch die Referenz auf den Nachfolger gespeichert, bzw. `null` falls kein Nachfolger existiert.
Programmiert werden soll ein Programm, in das nacheinander verschiedene (ganze) Zahlen eingegeben werden können. Diese Zahlen sollen in einer Liste gespeichert werden. Wird eine `0` eingegeben, so wird die Eingabe abgebrochen und alle Zahlen der Reihe nach auf der Konsole wieder ausgegeben.
Um die Funktionalität zu erreichen, werden zwei Klassen benötigt:

1.1 Node

Node
zahl: int next: Node
Node(zahl: int)

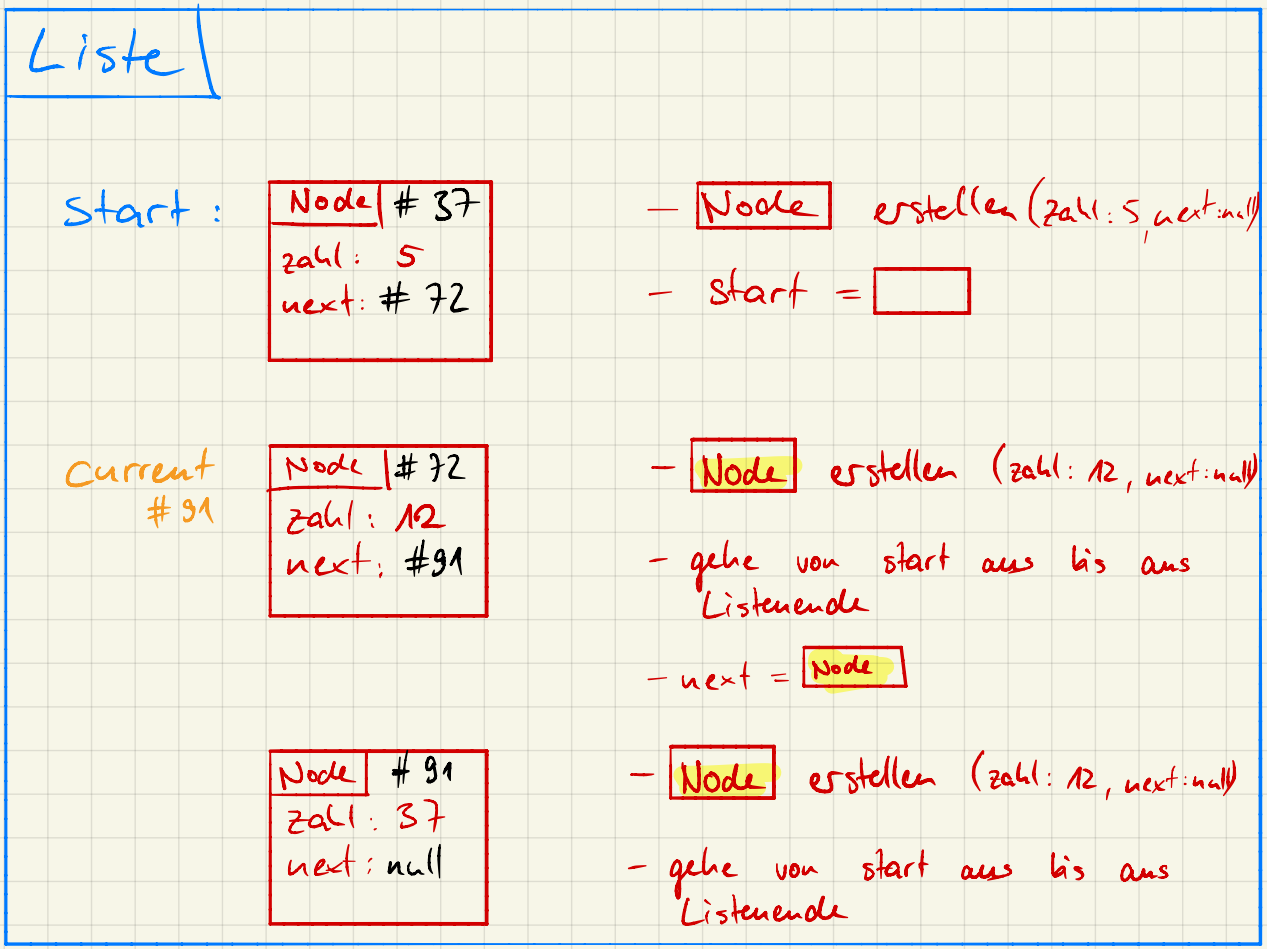
In einem `Node` wird die eigentliche Zahl (`zahl`) und deren Nachfolger (`next`) gespeichert. `next` soll `null` sein, wenn es keinen Nachfolger gibt.

1.2 Liste

Liste
start: Node
einfügen(zahl: int) länge(): int erste(): int

Die `Liste` ist für die Verwaltung der Liste zuständig.

- Bei `start` ist der erste`Node` gespeichert oder `null`, wenn die Liste noch keine Eintäge hat.
- Mit der Methode `einfügen` soll ein neuer `Node` mit der eingegebenen Zahl erzeugt und ans Ende der Liste angehängt werden.
- `länge` gibt die Anzahl der Elemente in der Liste an.
- `erste` gibt die Zahl am Anfang der Liste zurück und entfernt den ersten `Node` aus der Liste.

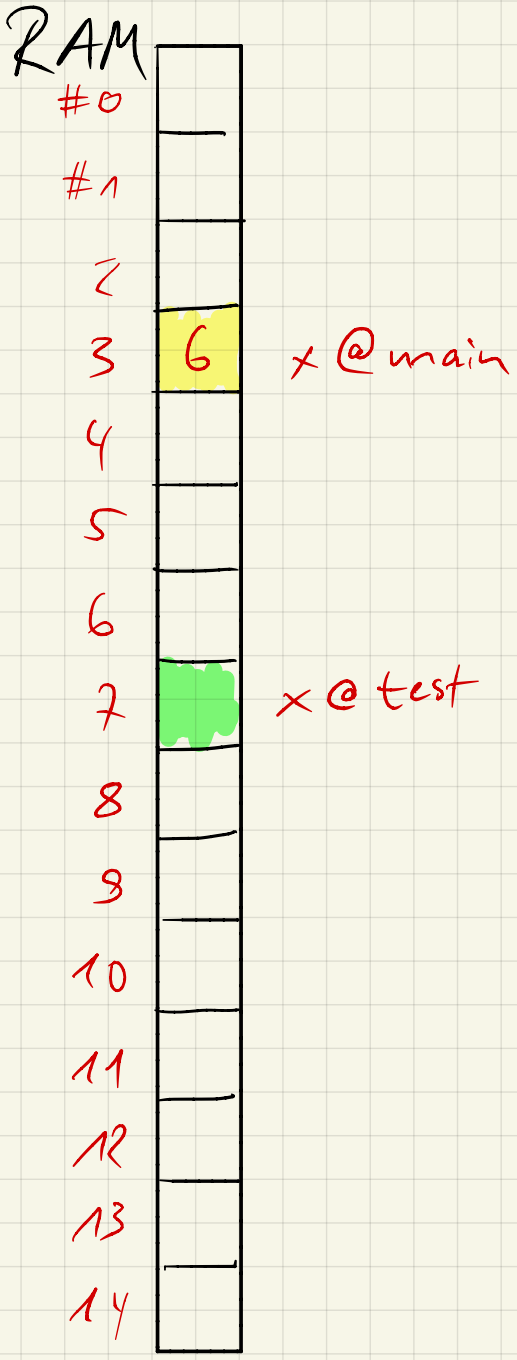


einfügen (5);

einfügen (12);

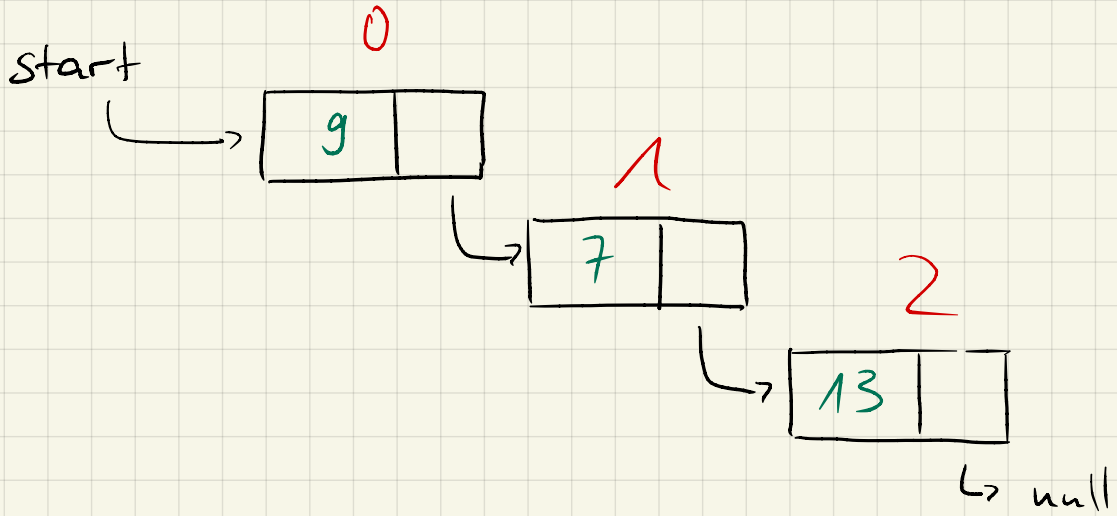
einfügen (37);

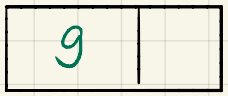
start = new Node (5);




```
public static int test() {
    int x;
    x = 7;
    return x;
}
main () {
    int x = 6;
    int y = test();
}
```

2.10.19





einfügen (9 , 2)

wert ↓ position ↓

```
public Wagen removeLast() {

    if ( this.getLaenge() == 0 )  return null

    if ( this.getLaenge() == 1 ) {
        LinkedList w = this.first;
        this.first = null;
        return w.getContent();
    }

    LinkedList current = this.first;

    for ( int i = 0; i < this.getLaenge() -1 ; i++ ) {

        current = current.getNext();

    }

    LinkedList w = current.getNext();

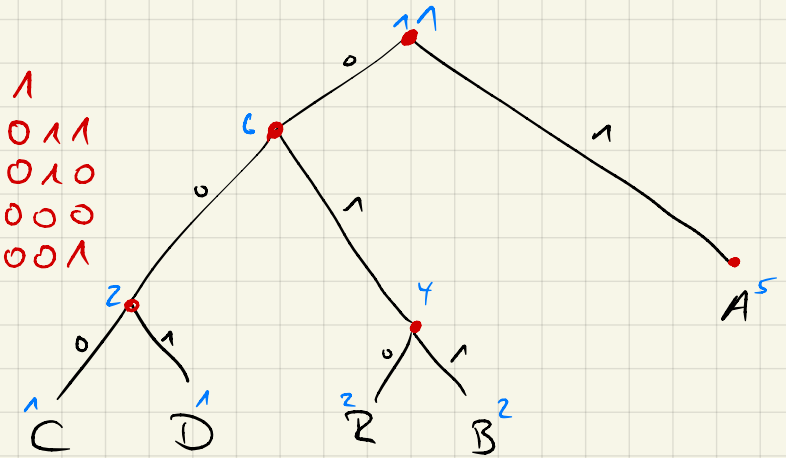
    current.setNext(null);

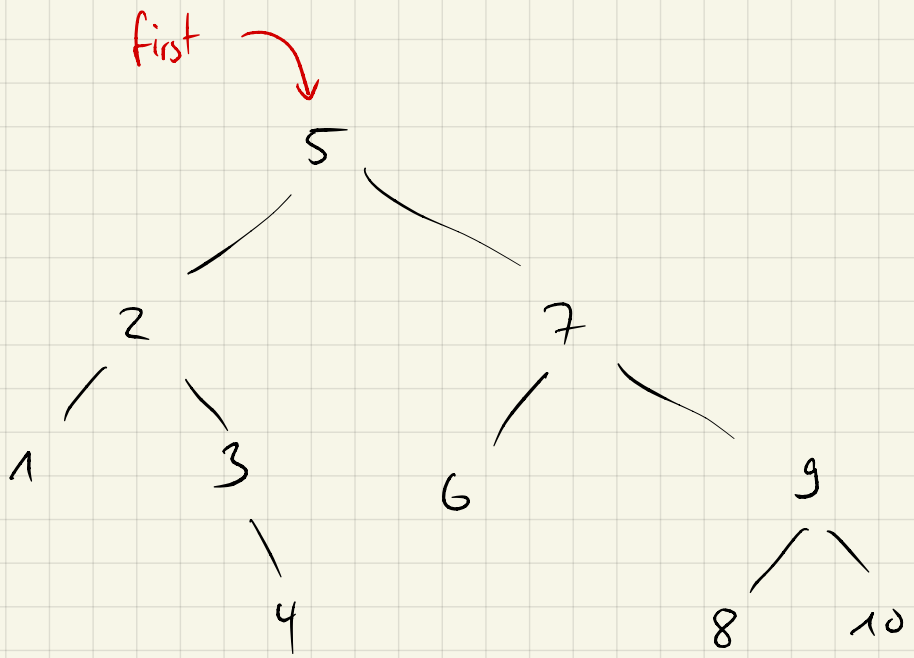
    return w.getContent();

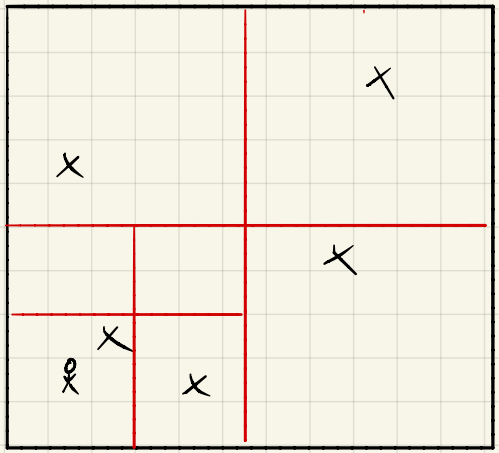

}
```

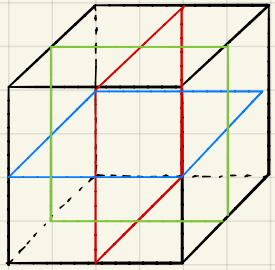
ABRACADABRA

A: 5 1
B: 2 011
R: 2 010
C: 1 000
D: 1 001





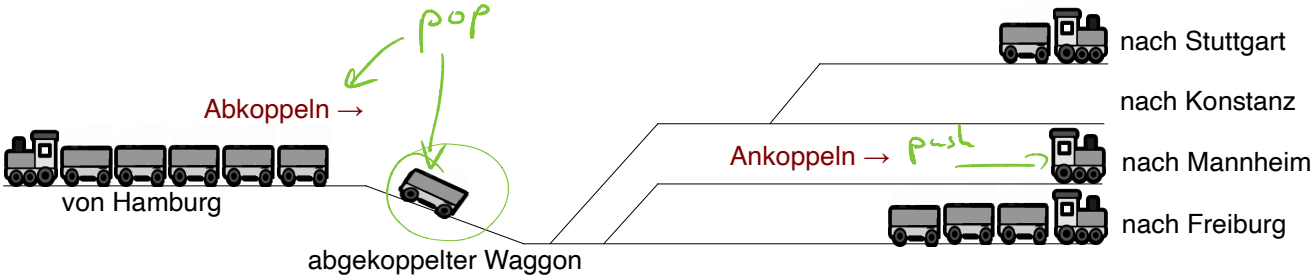




I B3 Abstrakte Datentypen

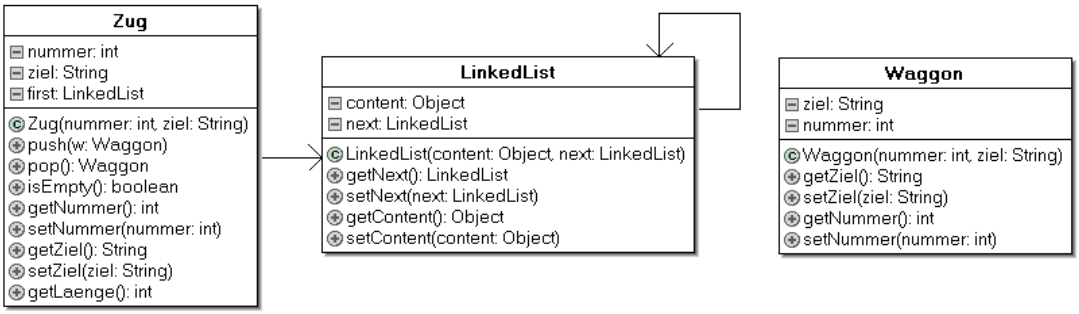
„Kornwestheim Rangierbahnhof 20070713“ von Rosenzweig. Lizenziert unter Creative Commons 3.0 über Wikimedia Commons - http://commons.wikimedia.org/wiki/File:Kornwestheim_Rangierbahnhof_20070713.jpg

Im Güterverkehr der Bahn müssen die Waggons der Güterzüge immer wieder zu neuen Zügen zusammengestellt werden, so dass alle Waggons das gleiche Ziel haben. In Kornwestheim gibt es dafür einen großen Rangierbahnhof. Ankommenden Güterzügen wird dort von hinten betrachtet Waggon für Waggon abgehängt. Diese rollen dann über einen leicht geneigten Hang durch Weichen gesteuert von hinten an Züge heran, die zur Abfahrt auf verschiedenen Gleisen bereitstehen, und werden dort automatisch angekoppelt.



Für den Rangierbahnhof soll eine Software entwickelt werden, mit der diese Prozesse automatisiert werden können. Dazu müssen die Daten der Züge und der Gleise modelliert werden. Dabei hat jeder Zug eine Zugnummer und ein Ziel. Auch die Waggons haben Nummern und Ziele. Ist das Ziel eines Zuges ein Rangierbahnhof, werden die Waggons dort ihrem Ziel entsprechend auf neue Züge verteilt.
Bei den Gleisen sind die Weichen zu modellieren. Jede Weiche kann entweder auf „gerade“ oder auf „abbiegen“ gestellt sein.

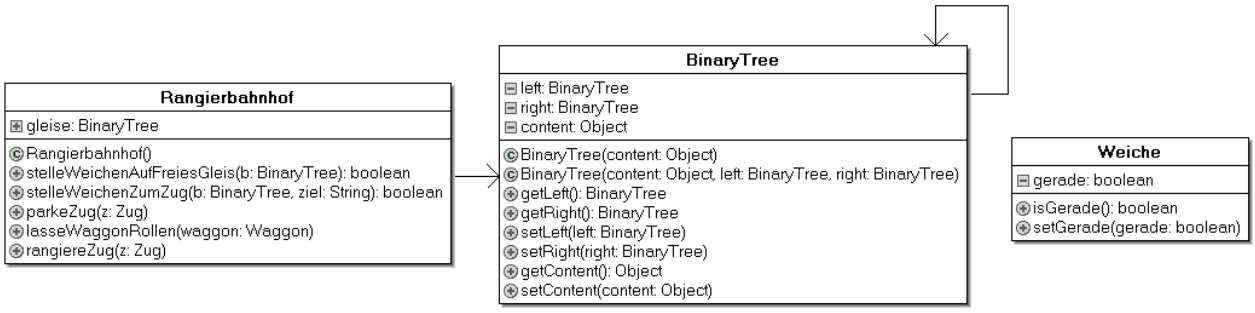
B3.1 Der Softwareentwickler für dieses Projekt schlägt vor, die Züge auf Grundlage des ADT Stapel (Stack) zu implementieren und für die Gleise die Datenstruktur Binärbaum zu verwenden. Das folgende UML-Diagramm stellt die dafür verwendeten Klassen dar:



- Beschreiben Sie die grundlegenden Eigenschaften eines Stapels und eines Binärbaums und erläutern Sie, warum der Vorschlag des Entwicklers sinnvoll ist.
  Erläutern Sie anschließend, wie Anhängen und Abkoppeln eines Waggons an einen Zug realisiert werden können.
- Die Methode `getLaenge():int` der Klasse Zug soll die Anzahl der Waggons zurückliefern. Implementieren Sie diese Methode auf der Basis des dargestellten Klassendiagramms. Sie können davon ausgehen, dass alle anderen Methoden schon korrekt implementiert sind.

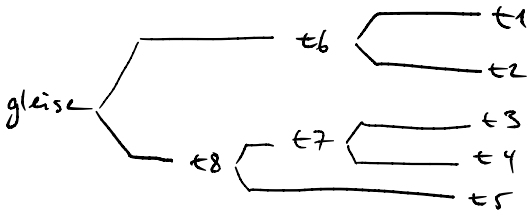
(6 VP)

B3.2 Die Gleise des Rangierbahnhofs werden als Binärbaum gespeichert. Jeder innere Knoten enthält eine Weiche als „content“. An den Blättern können Zug-Objekte (oder nichts) gespeichert sein.

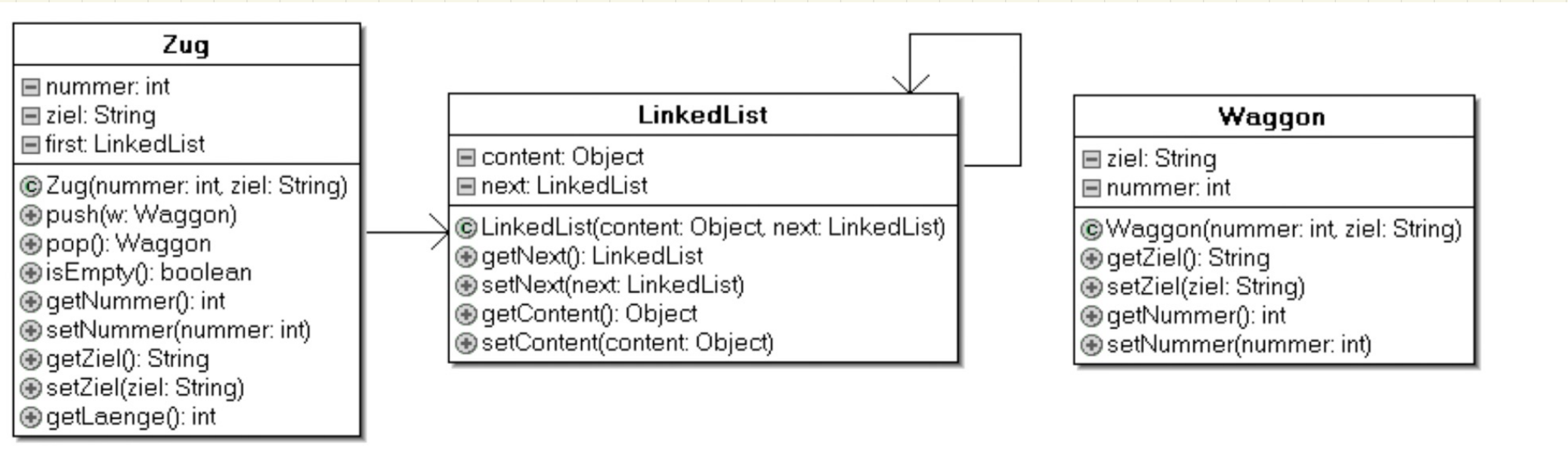


Die Implementation der Klasse Rangierbahnhof beruht auf folgendem Pseudocode.
Er beschreibt einen Ausschnitt des Konstruktors sowie den Algorithmus einer Methode, die die Weichen so stellt, dass sie zu einem freien Gleis führen:

```
t1 = neuer BinaryTree(leer)
t2 = neuer BinaryTree(leer)
t3 = neuer BinaryTree(leer)
t4 = neuer BinaryTree(leer)
t5 = neuer BinaryTree(leer)
```



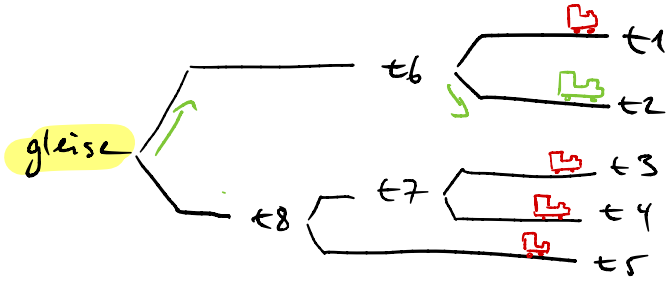

```
t6 = neuer BinaryTree(neue Weiche(), t1, t2)
t7 = neuer BinaryTree(neue Weiche(), t3, t4)
t8 = neuer BinaryTree(neue Weiche(), t7, t5)
gleise = neuer BinaryTree(neue Weiche(), t8, t6)
...
```




```
public int getLaenge() {
    int laenge = 0;
    LinkedList current = this.first;
    while (current != null){
        current = current.getNext();
        laenge++;
    }
    return laenge;
}
```

Methode stelleWeichenAufFreiesGleis(b:BinaryTree):boolean

```
1  wenn (b.getLeft() == leer)
2    wenn (b.getContent() == leer)
3      gib wahr zurück
4    sonst
5      gib falsch zurück
6    ende wenn
7  sonst
8    w = (Weiche) b.getContent()
9    wenn (stelleWeichenAufFreiesGleis(b.getRight()) == wahr)
10     w.setGerade(wahr)
11     gib wahr zurück
12   sonst
13     wenn (stelleWeichenAufFreiesGleis(b.getLeft()) == wahr)
14       w.setGerade(falsch)
15       gib wahr zurück
16     sonst
17       gib falsch zurück
18     ende wenn
19   ende wenn
20 ende wenn
```



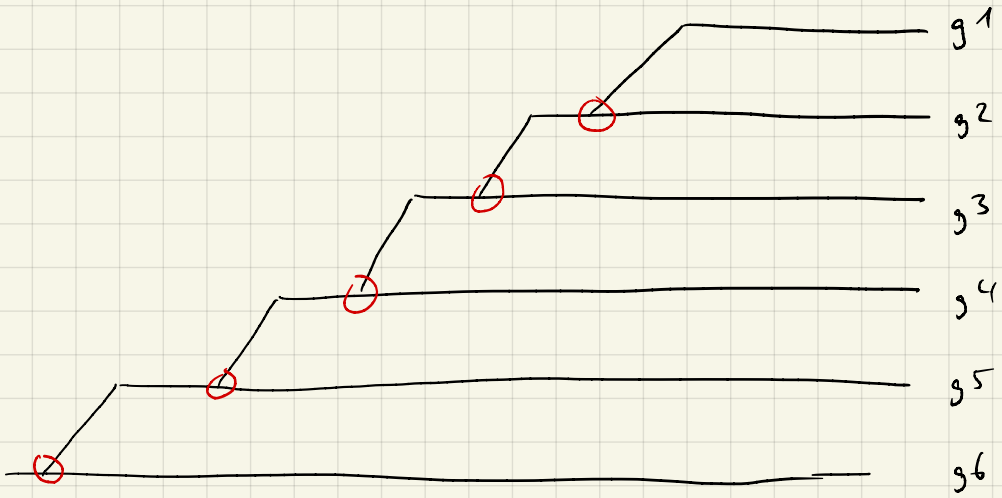
- Zeichnen Sie die Gleisanlage auf, die durch diesen Konstruktor modelliert wird. 2
- Erläutern Sie, wie die Methode `stelleWeichenAufFreiesGleis(..)` dafür sorgt, dass die Weichen so gestellt werden, dass sie zu einem freien Gleis führen (um eine neue Lok dort zu platzieren). Erläutern Sie, was passiert, wenn es kein freies Gleis mehr gibt. 3
- Implementieren Sie die Methode `parkeZug(z:Zug)`, die dafür sorgt, dass der Zug auf dem Gleis (d.h. Blatt des Gleis-Baumes) als content gespeichert wird, das durch die aktuelle Weichenstellung vorgegeben ist. 4
- Untersuchen Sie, welches Laufzeitverhalten für die Methode `parkeZug(..)` im schlimmsten Fall bei n Gleisen zu erwarten ist. Untersuchen Sie, welchen Einfluss die Struktur der Gleisanlage auf Ihre Aussage hat. 2

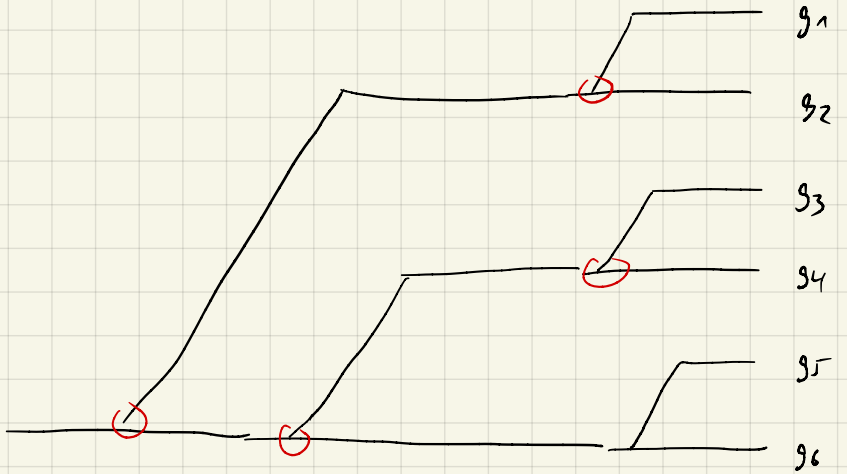
(11 VP)

B3.3 Die Methode `stelleWeichenZumZug(b:BinaryTree; ziel:String): boolean` stellt die Weichen so, dass ein Zug mit dem angegeben Ziel erreicht wird. Gibt es keinen derartigen Zug, gibt die Methode false zurück.
Die Methode `lasseWaggonRollen()` hängt einen Waggon an den Zug an, der durch die aktuelle Weichenstellung erreicht wird

- Implementieren Sie die Methode `rangiereZug(z:Zug)`, die einen ganzen Zug abarbeitet und jeden Wagen des Zuges an den richtigen weiterführenden Zug anhängt. Sie können davon ausgehen, dass es für jedes Ziel eines Waggons einen entsprechenden Zug im Rangierbahnhof gibt (d.h. `stelleWeichenZumZug(..)` findet immer einen passende Weichenstellung).

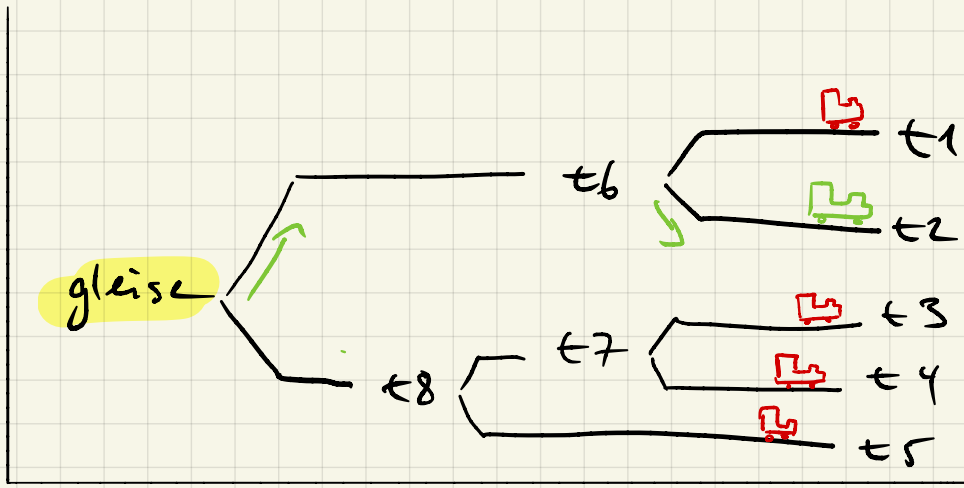
(3 VP)


g^1
g^2
g^3
g^4
g^5
g^6


g_1
g_2
g_3
g_4
g_5
g_6

```
public void parkeZug(Zug z) {
    BinaryTree current = this.gleise;
    while(current.getLeft() != null) {
        Weiche w = current.getContent();
        if(w.isGerade()) {
            current = current.getRight();
        } else {
            current = current.getLeft();
        }

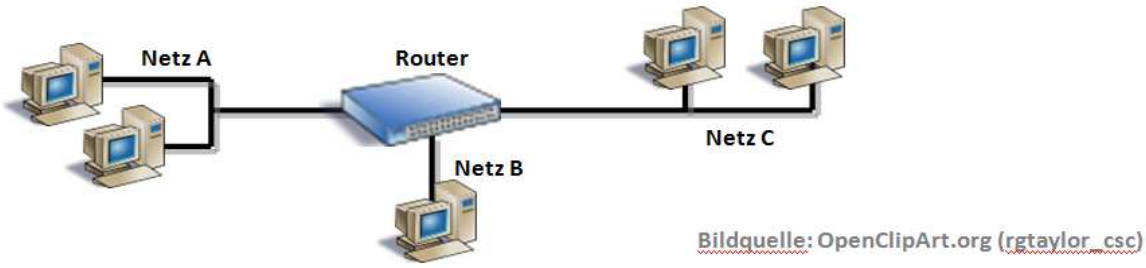
    }
    current.setContent(z);
}
```



- Kräuterbaguette
- Sauce
- Dönermadler
- Bionade
- Salat + Sauce
- Wedges

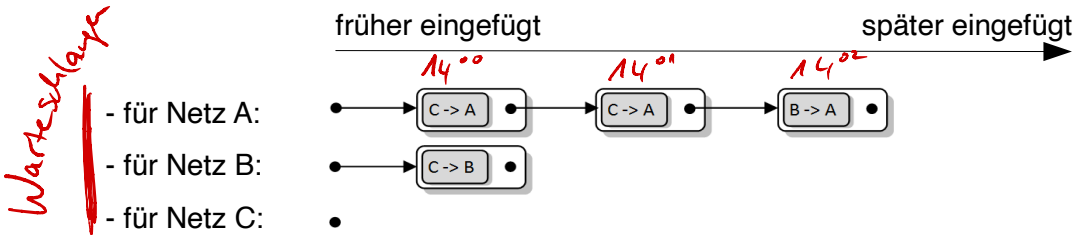
B3 Abstrakte Datentypen

Durch den zunehmenden Datenverkehr im Internet gibt es eine Diskussion wie mit Leitungsengpässen umzugehen ist. Zur Zeit gilt die Netzneutralität, d.h. alle Datenpakete - egal von welchem Absender an welchen Empfänger sie gehen und egal welchen Inhalt sie haben - werden gleich behandelt. Bei einer ausgelasteten Leitung werden ankommende Pakete vom Router zwischengespeichert und dann in der Reihenfolge ihres Eintreffens weitergeleitet. Die Betreiber von Telekommunikationsnetzwerken möchten diese Netzneutralität gerne abschaffen und die Daten in Prioritätsklassen einteilen. Pakete mit höherer Priorität würden dann bevorzugt weitergeleitet.



Zu Beginn sind schon folgende Pakete zwischengespeichert
(B -> A : stellt ein Paket von Netz B nach Netz A dar):

Zwischenspeicher am Anfang



Pro Zeittakt schickt der Router zuerst in jedes Netz jeweils ein Datenpaket. Danach empfängt er gegebenenfalls aus jedem Netz ein Paket (in der Reihenfolge Netz A, Netz B und zuletzt Netz C).

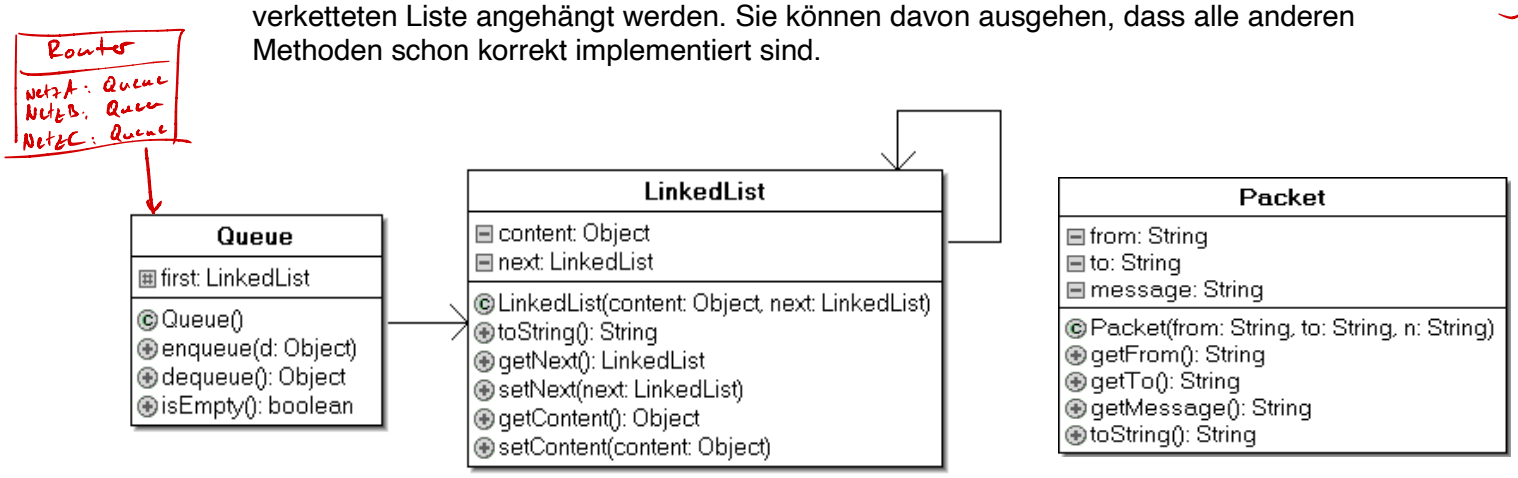
Folgende Pakete erreichen den Router:

Takt 1: Paket von B->A, Paket von C->A, Paket von A->C
Takt 2: Paket von C->A
Takt 3: Paket von A->B, Paket von C->B
Takt 4: Paket von A->B, Paket von C->B, Paket von B->C

B3.1 Das Steuerungsprogramm eines die Netzneutralität wahrenden Routers könnte den ADT Schlange (Queue) zur Verwaltung der Pakete verwenden. Für jedes angeschlossene Netz gibt es dann eine Schlange, die die noch abzuarbeitenden Datenpakete speichert.

- Beschreiben Sie die grundlegenden Eigenschaften einer Schlange und erläutern Sie, warum die Wahl des ADT Schlange für diese Anwendung sinnvoll ist.
- Geben Sie an, welche Operationen der Warteschlange für Netz A ausgeführt werden müssen, um
  - das nächste zu übermittelnde Paket von C->A aus der Warteschlange zu holen,
  - das in Takt 1 ankommende Paket aus Netz B für Netz A zu speichern.
- Stellen Sie die Schlangen nach dem 4. Takt dar, wenn der Router die Netzneutralität wahrt.

- Erläutern Sie die Bedeutung der Beziehungen zwischen den unten dargestellten Klassen. Gehen Sie insbesondere auf die Bedeutung der Beziehung einer Klasse mit sich selbst ein.
- Implementieren Sie die Methode enqueue(Object d) der Klasse Queue auf der Basis des unten dargestellten Klassendiagramms, so dass neue Pakete an das Ende der verketteten Liste angehängt werden. Sie können davon ausgehen, dass alle anderen Methoden schon korrekt implementiert sind.



(12 VP)

B3.2 Um bestimmte Pakete zu bevorzugen, wird nun eine Priorität der Pakete eingeführt. Alle Pakete, die aus Netz C kommen, sollen mit höherer Priorität weitergeleitet werden. Um dies zu realisieren, wird der Datentyp PriorityQueue verwendet, der sich nur in der Methode enqueue von der normalen Schlange unterscheidet: Die Pakete mit der höheren Priorität werden vor den Paketen mit niedrigerer Priorität in der Schlange einsortiert.

- Stellen Sie die Schlangen nach dem 4. Takt dar, wenn der Router diese Priorität beachtet.
- Ergänzen Sie das Klassendiagramm um die neue Klasse PriorityQueue.
  Dabei sollen die nicht veränderten Methoden von der Klasse Queue übernommen werden. Erläutern Sie die dabei neu hinzugekommene Beziehung.

(4 VP)

B3.3 Es gibt Möglichkeiten eine PriorityQueue zu implementieren, so dass die Methoden enqueue und dequeue in einer durchschnittlichen Laufzeit proportional zu log(n) ausgeführt werden (z.B. mittels Heap). Dabei ist n die Anzahl der in der PriorityQueue gespeicherten Pakete.

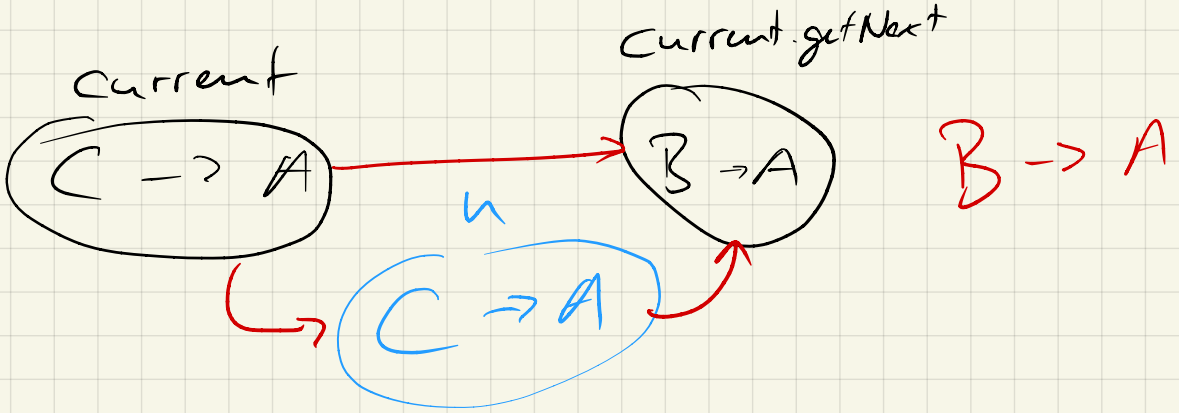
- Analysieren Sie das Laufzeitverhalten von enqueue und dequeue bei einer PriorityQueue auf der Basis einer LinkedList. Beurteilen Sie, ob es bei einem Router im Hinblick auf die Laufzeit sinnvoller ist, einen Heap oder eine LinkedList als Datenstruktur für eine PriorityQueue einzusetzen.

(4 VP)

A C -> A

B A -> B

C A -> C




```
n.setNext(current.getNext());
current.setNext(n);
```

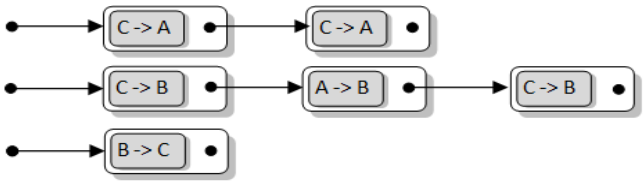
Lösungen:

B3.1

- Eine Schlange arbeitet nach dem FIFO-Prinzip (First-in-first-out), d.h. die zuerst gespeicherten Daten werden als erste aus der Schlange entnommen. Genau dies ist in der Aufgabe gefordert, wenn die Netzneutralität gewahrt bleiben soll.
  Die Schlange kennt nur die Operationen enqueue, dequeue und isEmpty. Diese reichen für eine Routerverwaltung vollkommen aus, da der Router nur neue Pakete zwischenspeichern und das jeweils erste Paket der Schlange abrufen können muss, um es weiter zu versenden.

- Es muss also eine dequeue-Operation ausgeführt werden, um das Paket aus der Warteschlange zu holen und dort zu entfernen.
  Das neue Paket wird mit enqueue(packet) der Warteschlange von Netz A hinzugefügt.

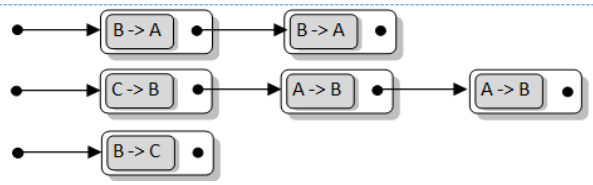
- Schlange A:
  Schlange B:
  Schlange C:



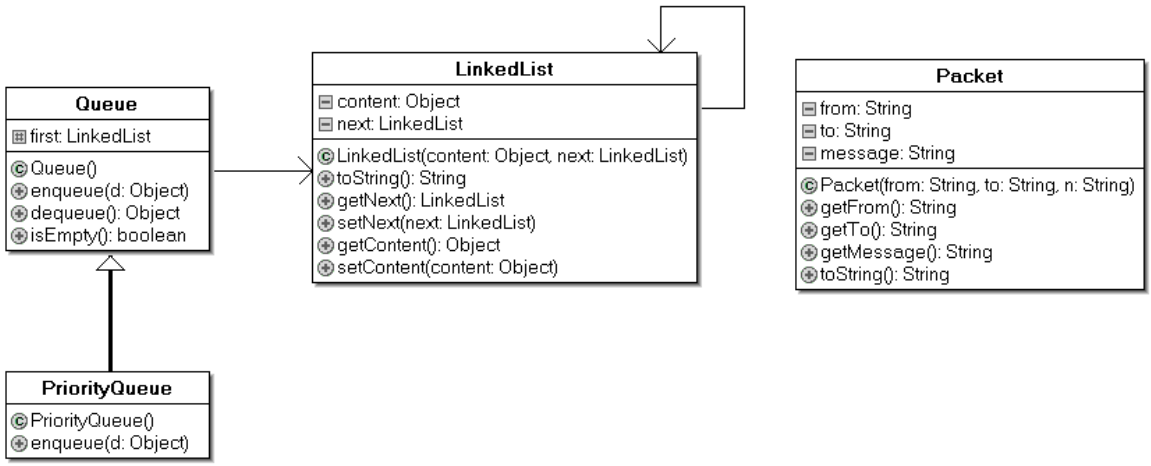
- Die Queue verwaltet die Datenpakete in einer linearen verketteten Liste. Daher muss die Queue das erste Element der verketteten Liste kennen (Assoziation). Eine verkettete Liste lässt sich aufbauen, indem jedes Element der Liste wieder eine Liste oder die leere Liste enthält. Dadurch entsteht eine rekursive Datenstruktur und wird die kennt-Beziehung (Assoziation) der Klasse LinkedList mit sich selbst erklärt.

- ```
  public class Queue {
    private LinkedList first = null;
  ..
    public void enqueue(Object d) {
      LinkedList n = new LinkedList(d, null);
      if (first == null) {
        first = n;
      } else {
        LinkedList l = first;
        while (l.getNext() != null) {
          l = l.getNext();
        } // end of while
        l.setNext(n);
      } // end of if
    }
  ..
  }
  ```
  ```

B3.2 •



•



Die PriorityQueue benötigt die gleichen Methoden wie die Queue. Nur die Implementation der Methode enqueue unterscheidet sich von der ursprünglichen Form in der Queue. Daher ist es sinnvoll die PriorityQueue von der Queue abzuleiten und die Methode enqueue zu überschreiben. Die PrioriyQueue steht also in einer „ist-Beziehung“ (Vererbung) zur Queue.

B3.3 • Bei einer PriorityQueue liegt die Laufzeit für das Einfügen in O(n), da im schlimmsten Fall die ganze Liste durchsucht werden muss. Auch im Durchschnitt wird die Hälfte der Liste durchsucht. Das Entfernen geht in O(1), da nur das 1. Element der Liste entfernt werden muss, auf das man direkten Zugriff hat.
Beim Heap liegen beide Operationen in O(log n).

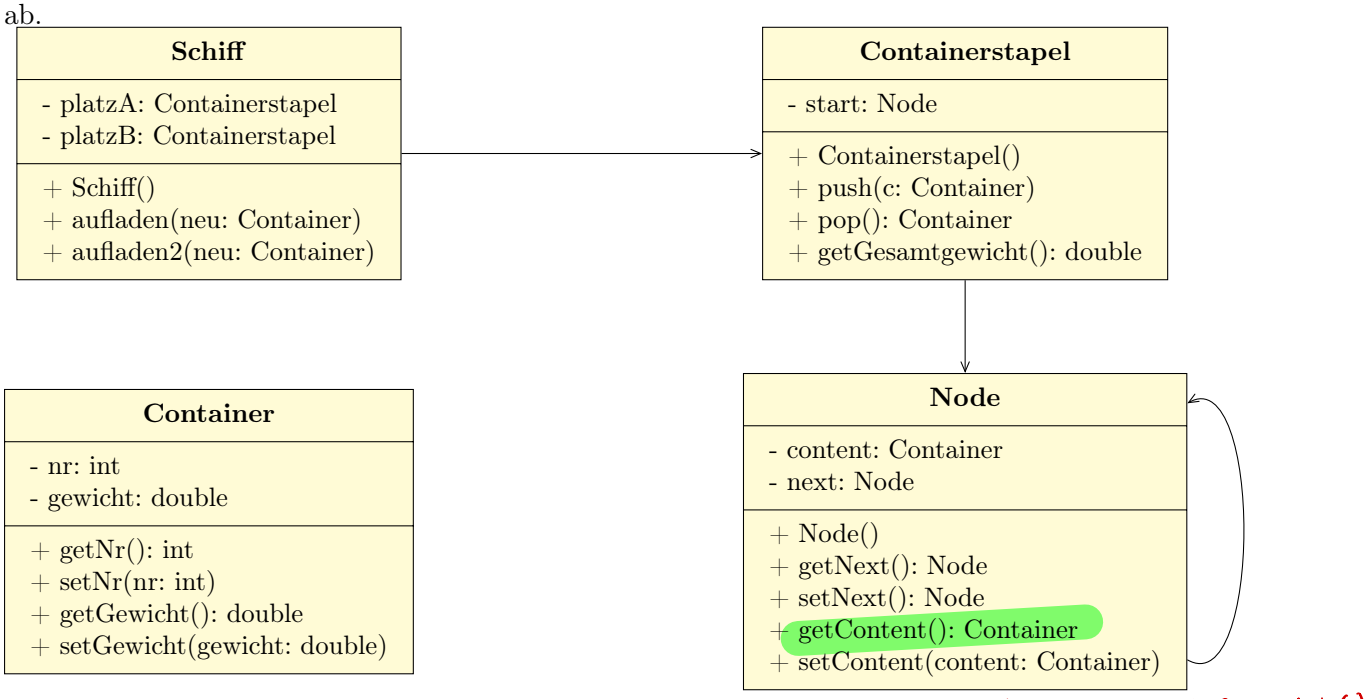
Da gleich viele Einfüge- wie Entferne-Operationen auftreten, ist bei größeren Warteschlangen auf jeden Fall der Heap zu bevorzugen, da 2*log n viel kleiner als 1+n ist.
Sind die Warteschlangen sehr kurz, ist es egal, welche Implementierung man verwendet, da keine großen Unterschiede zu erwarten sind.

Klausur 22.10.2019

Name: _____ VP: _____/20P NP: _____ mündlich: _____

Einleitung

Ein Containerschiff ist ein speziell für den Transport von Containern gebautes Frachtschiff. Die Container sind dabei genormte Behälter, die auf vorgesehenen Plätzen aufeinander gestapelt werden.
Ein besonderer Schiffstyp ist das sogenannte Feederschiff, das nur eine geringe Anzahl von Plätzen zur Aufnahme von Containern hat und deshalb auch kleinere Häfen anfahren kann.
Eine Reederei besitzt mehrere Feederschiffe mit jeweils genau zwei Stapelplätze (`platzA` und `platzB`) an denen Container gelagert werden kann. Ein solcher Container auf diesen Stapeln darf nicht mehr als 28 t wiegen.
Die Reederei lässt sich nun ein Computerprogramm zur Verwaltung neu entwickeln, dieses beruht auf folgenden Klassen. Die Klasse `Containerstapel` bildet dabei einen Stapel (bzw. *Stack* oder *Keller*) ab.



g = g + current.getContent().getGewicht()

1. Aufgabe (6 VP)

a) Begründe, warum diese Aufgabenstellung sinnvollerweise mit einem *Stapel* modelliert wird.

b) Erläutere, welche Auswirkungen die Methoden `push` und `pop` auf einen Stapel haben.

c) Implementiere die Methode `getGesamtgewicht(): double` der Klasse `Containerstapel`.[1] Dabei dürfen keine zusätzlichen Attribute oder Methoden in die Klassen eingefügt werden.

2. Aufgabe (4VP)

Das Schiff soll möglichst ausgeglichen beladen werden. Aus diesem Grund werden neie Container immer auf dem leichteren Stapel abgelegt, bei gleich schweren Stapeln auf `platzA`.

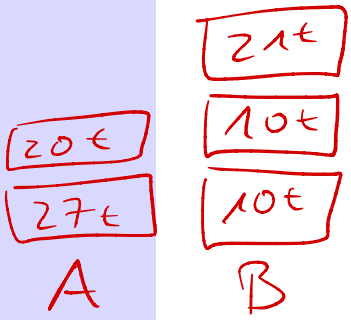
[1] Möglichst kurz und ohne überflüssige Befehle! Für viel unnötigen Code kann es Abzug geben.

a) Implementiere in der Klasse `Schiff` die Methode `aufladen(neu: Container)`, die den neuen Container auf den jeweils leichteren Stapel ablegt.

3. Aufgabe (6VP)

Eine experimentelle Lademethode `aufladen2(neu: Container)` beruht auf nachfolgendem Pseudocode-Algorithmus. Es soll dabei nicht geprüft werden, ob beide Stapel überhaupt Container enthalten.

```
aufladen2(Container neu) {
        a = platzA.getGesamtgewicht();
        b = platzB.getGesamtgewicht();
        c = neu.getGewicht();
        wenn (a < b) {
                wenn (a+c > b+20) {
                        platzA.push(platzB.pop());
                        platzB.push(neu);
                } sonst {
                        platzA.push(neu);
                }
        } sonst {
                wenn (b+c > a+20) {
                        platzB.push(platzA.pop());
                        platzA.push(neu);
                } sonst {
                        platzB.push(neu);
                }
        }
}
```



Listing 1: experimentelle Lademethode

Bei einem Ladevorgang befinden sich auf `platzA` und `platzB` je ein Container mit 10 t. Es sollen nun drei Container mit den Gewichten in der Reihenfolge 21 t, 27 t und 20 t geladen werden.

a) Stelle das Ergebnis des Ladevorgangs in einer Skizze dar.

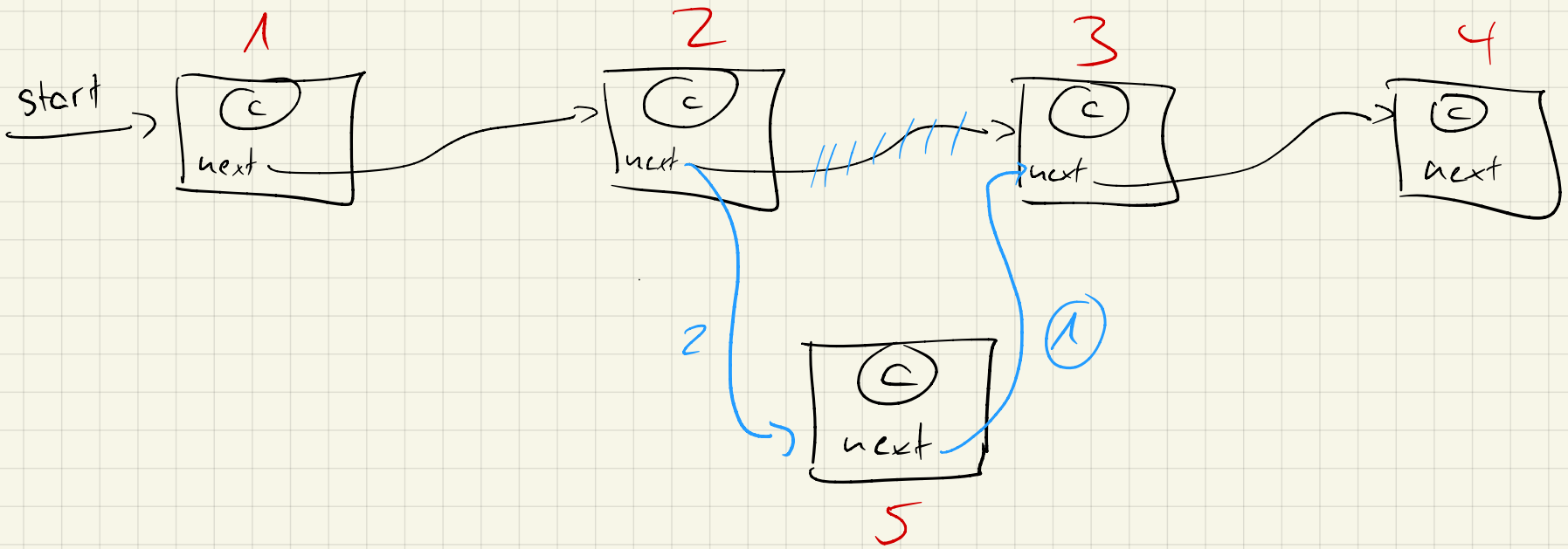
b) Analysiere, welche Vorteile dieser neue Algorithmus gegenüber dem herkömmlichen Algorithmus aus Aufgabe 2 hat.

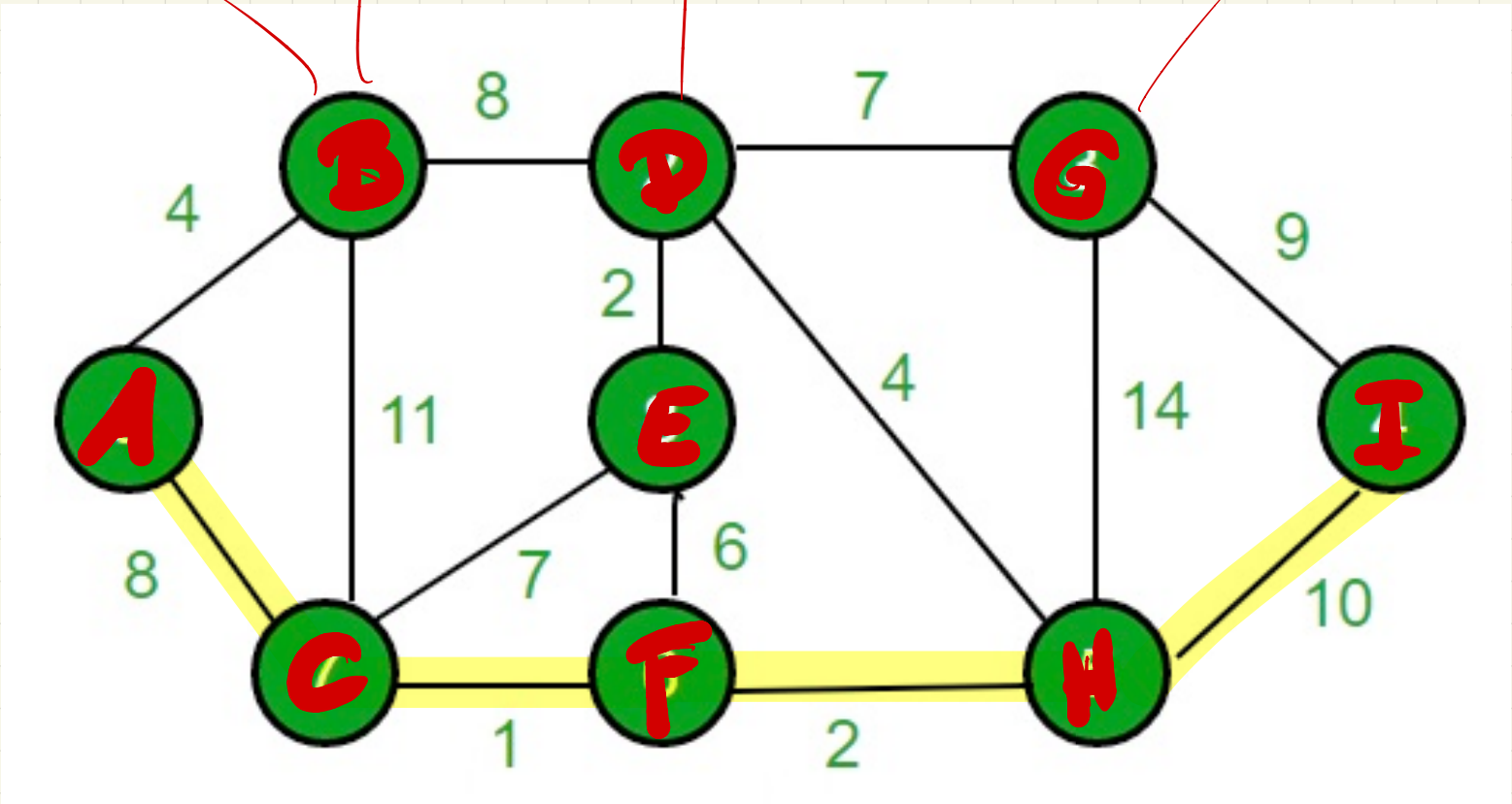
c) Beurteile, ob dieser experimentelle Ladevorgang in den Produktivbetrieb übernommen werden sollte.

4. Aufgabe (4VP)

Zur Vorbereitung des Ladevorgangs wird eine **neue** Ladeliste erstellt (ebenfalls vom Typ `Containerstapel`), in der die Container – bezogen auf die Zielhäfen – in umgekehrter Reihenfolge notiert werden, damit die Container, die zuerst abgeladen werden müssen, ganz oben stehen.
Neu hinzukommende Container werden an der richtigen Stelle in diese neue Liste eingefügt. Dafür wird in der `Containerstapel`-Klasse eine neue Methode `einfuegenAnKorrekterPosition(c: Container)` implementiert.

a) Beschreibe, wie innerhalb der Methode die Verkettung der Nodes verändert werden muss, um eine neue Node innerhalb dieser Verketteten Liste einzufügen. Veranschauliche die Beschreibung mit einer aussagekräftigen Skizze.

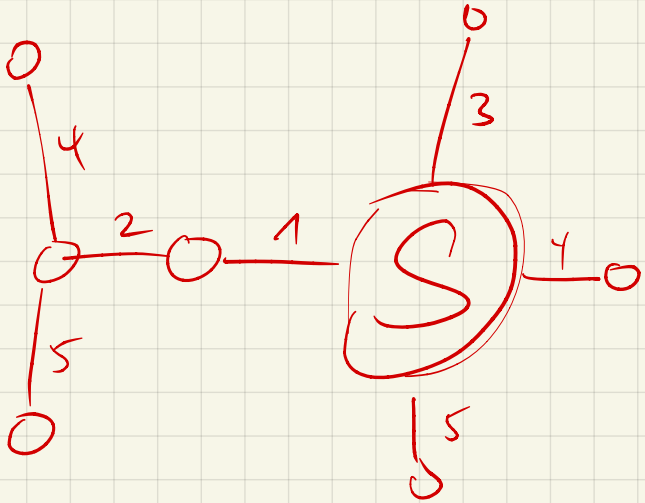

start
1
2
3
4
5
c
next
2
1

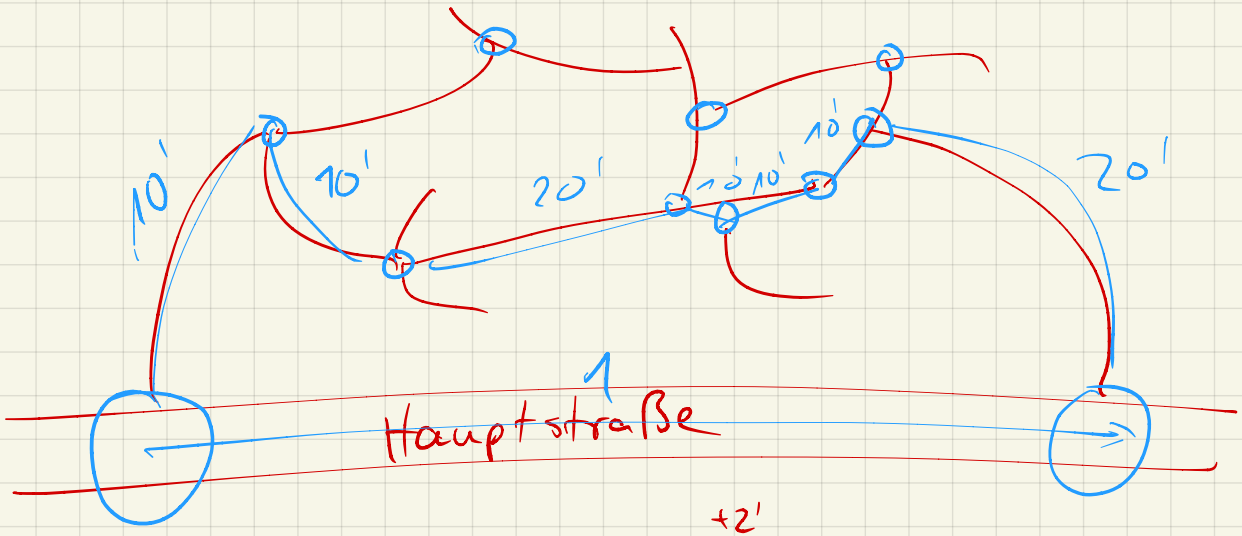


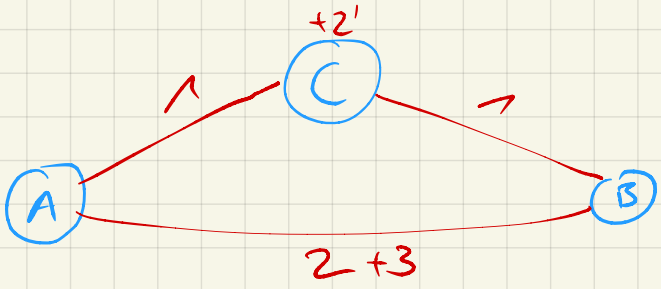
Dijkstra

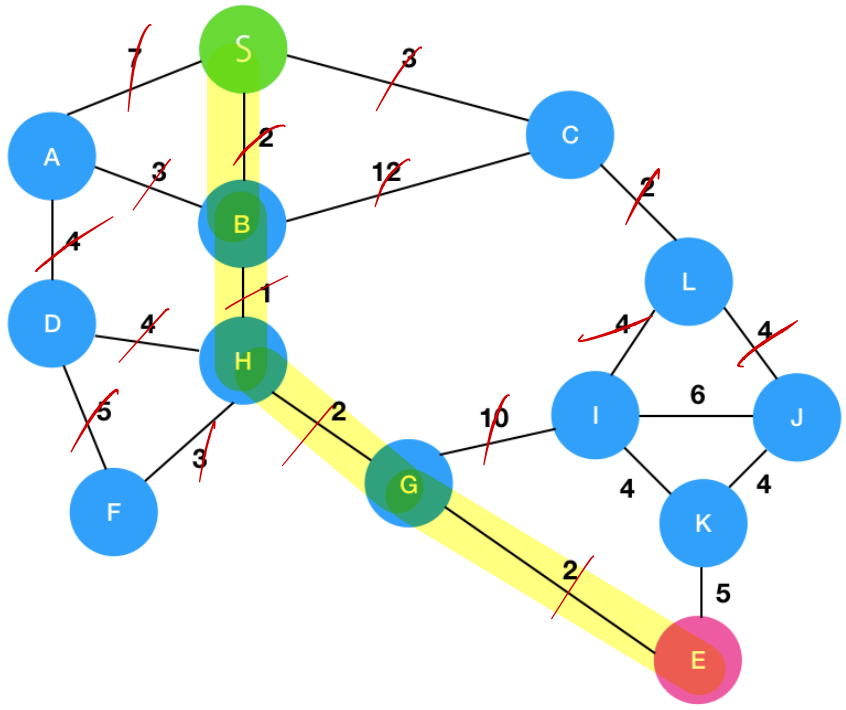
Knoten		Kürzester Weg zu A	Vorgänger
A	✓	0	/
B	✓	4	A
C	✓	8	A
D	✓	12	B

E	✓	~~15~~ 14	~~E~~ D
F	✓	9	C
H	✓	11	~~F~~
G	✓	~~25~~ 19	~~H~~ D
I	✓	21	H

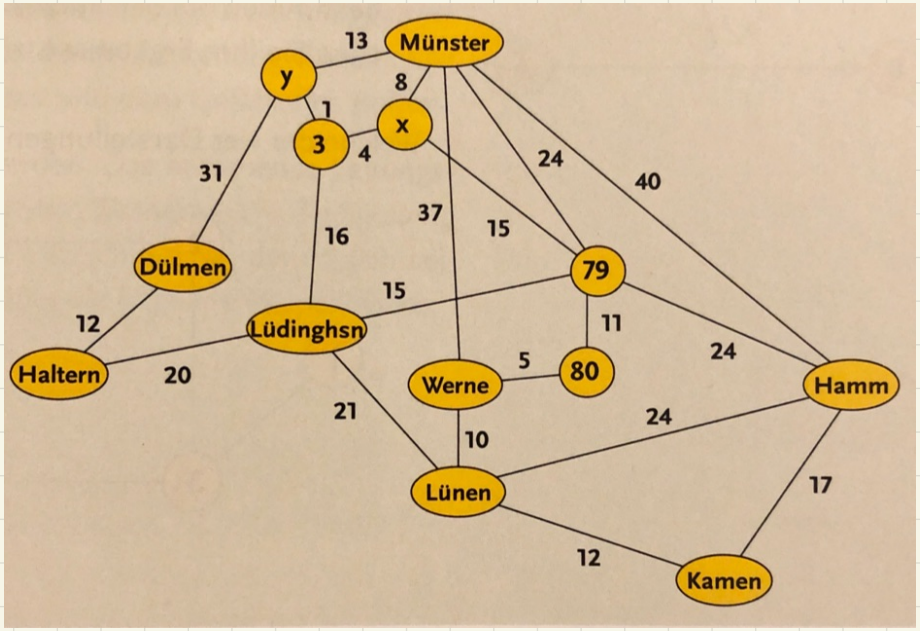








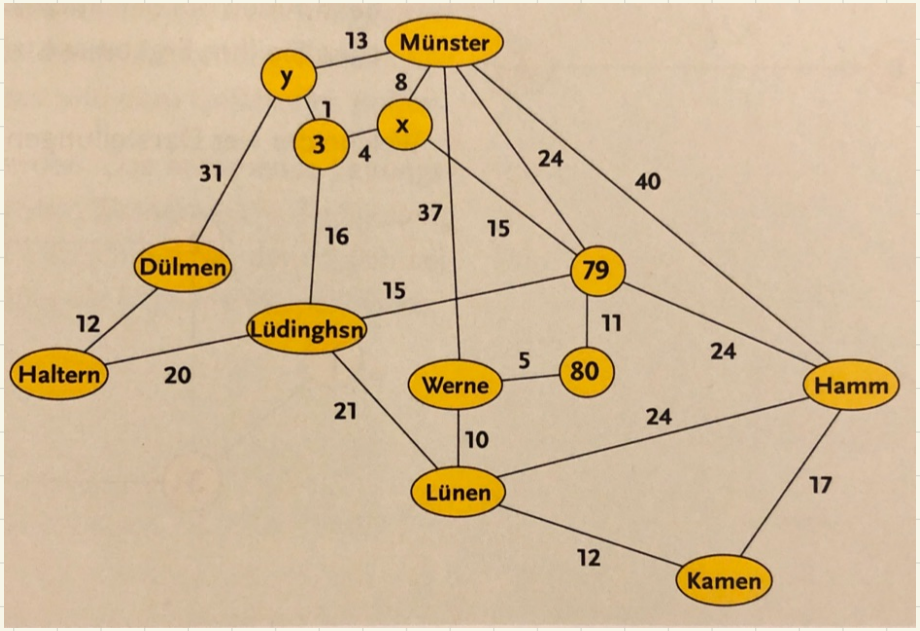
Knoten	Kürzeste Distanz zu S	Vorgänger
S ✓	0	-
A ✓	~~7~~ 5	~~S~~ B
B ✓	2	S
C ✓	3	S
D ✓	7	H
F ✓	6	H
G ✓	5	H
H ✓	3	B
I	~~15~~ 9	~~G~~ L
J	9	L
K		
L ✓	5	C
E	7	G



Dijkstra

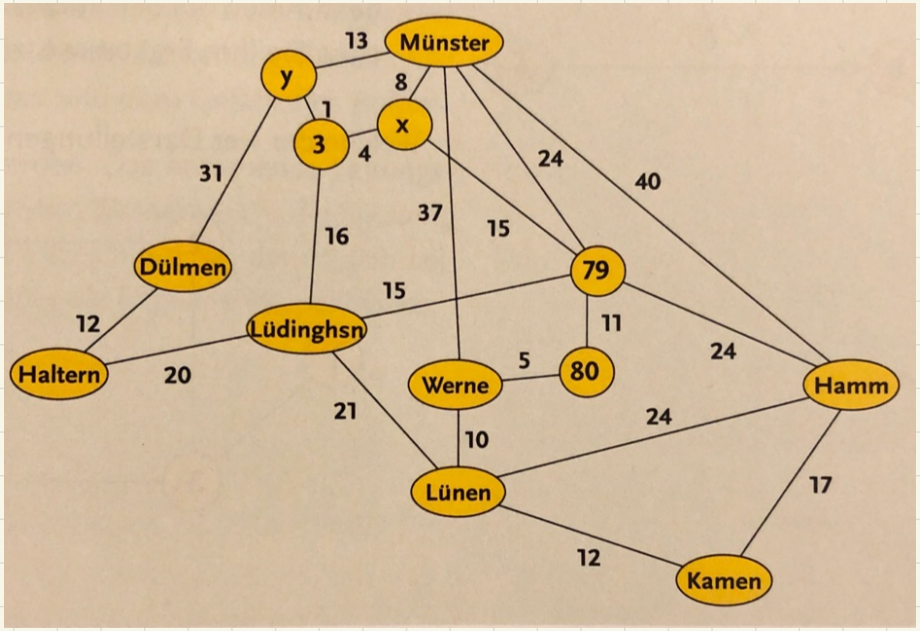
Graph

- gerichtet / ungerichtet
(- sortiert / unsortiert)
- gewichtet / ungewichtet
- vollständig
- zusammenhängend



Adjazenzmatrix

von \ zu	Münster	y	3	x	Dülmen	Lüdinghsn.	79	Hamm	Werne	80	Haltern	Lünen	Kamen
Münster	X	13	X										
y	13												
3	X												
x													
Dülmen													
Lüdinghsn.													
79													
Hamm													
Werne													
80													
Haltern													
Lünen													
Kamen													



List

Münster → Hamm (40) | 79 (24) | x (8) | ...

Werne → 80 (5) | Lünen (10) | ...

Lünen → Werne (10) | Hamm (24)

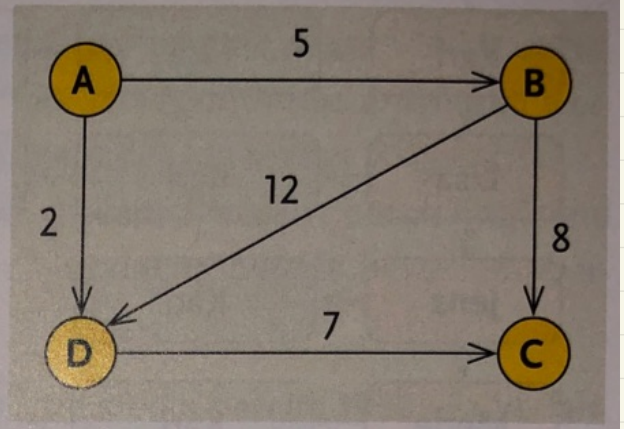
Kamen

⋮

a) Skizzieren Sie eine Übersicht über die Knotenliste und die Kantenlisten des nebenstehenden Graphen ähnlich wie auf S. 211.

b) Geben Sie die Adjazenzmatrix des Graphen an.

c) Um welche besondere Art von Graph handelt es sich hier?



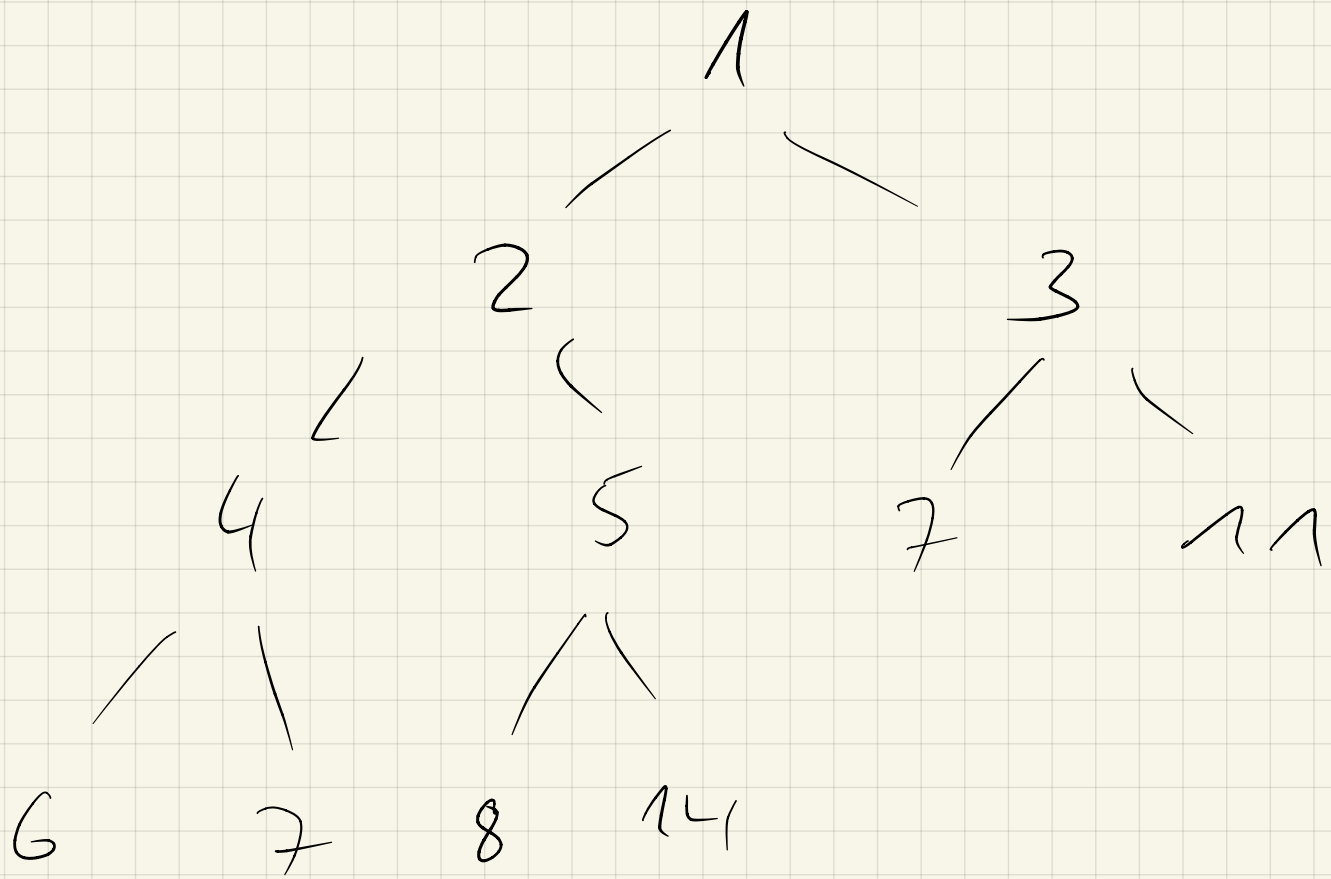
a)

A → B (5) → D (2)
↓
B → D (12) → C (8)
↓
C
↓
D → C (7)

b)

von \ zu	A	B	C	D
A		5	×	2
B	×		8	12
C	×	×		×
D	×	×	7	

c) zusammenhängend (?), gerichtet, gewichtet


1
2
3
4
5
7
11
6
7
8
14

Zeichnen Sie den Graphen zu den folgenden Adjazenzmatrizen. Geben Sie jeweils an, welche der Eigenschaften für den Graphen zutreffend sind und welche nicht.

gerichtet | gewichtet | vollständig | zusammenhängend | ist ein Baum

a)

	A	B	C	D	E
A		1		1	1
B	1				
C				1	
D	1		1		
E	1				

zusammenhängend

b)

	A	B	C	D	E
A		1	1	1	1
B	1		1	1	1
C	1	1		1	1
D	1	1	1		1
E	1	1	1	1	

zus.hgd. / vollständig

c)

	A	B	C	D	E
A				4	
B			1		
C		2			
D	4				0,5
E	2				

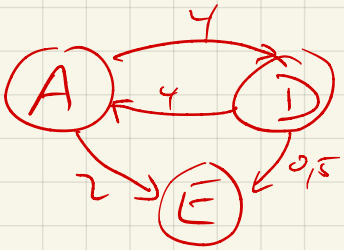
gerichtet, gewichtet

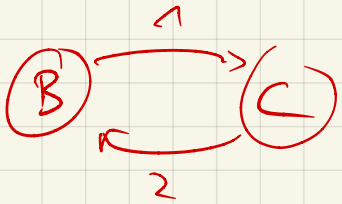
d)

	A	B	C	D	E
A		1			
B	1		2		
C		2		3	
D			3		4
E				4	

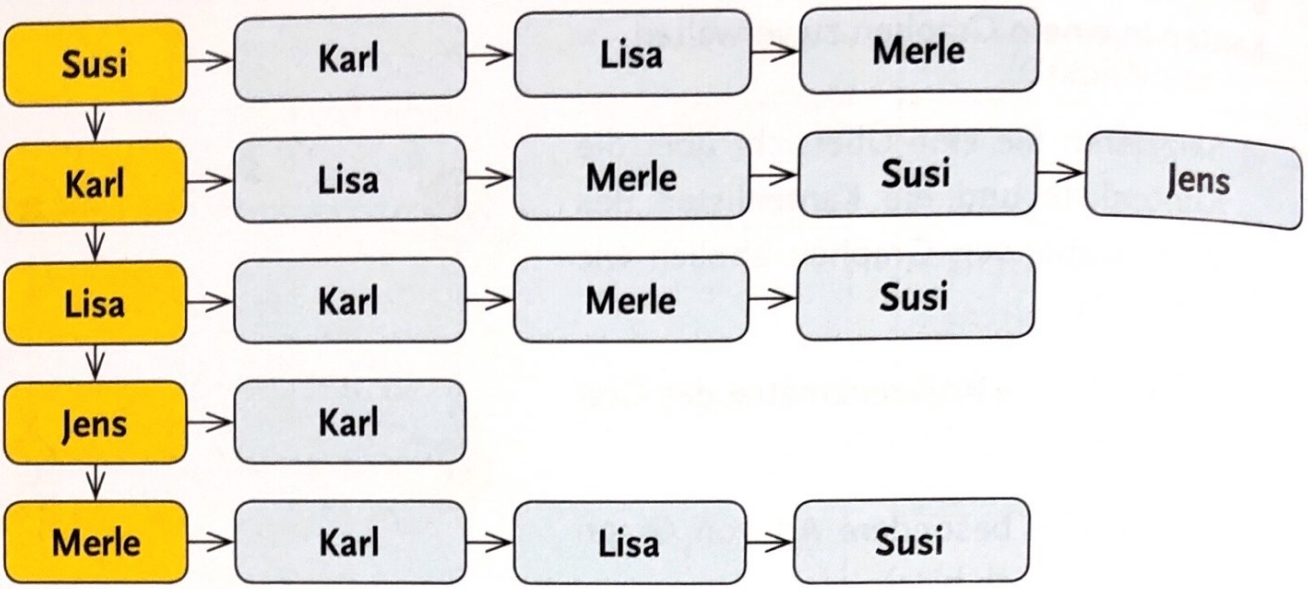
gewichtet, zus.hgd.

e) Formulieren Sie einen Algorithmus, der feststellt, ob ein Graph vollständig ist. Der Graph wird dabei als Adjazenzmatrix übergeben.





4. Auch Freundschaftsstrukturen können in einem Graphen dargestellt werden. Dabei sind die Personen als Knoten und die Freundschaftsbeziehungen als Kanten modelliert.



Da hier nur modelliert wurde, ob eine Freundschaft zwischen zwei Personen besteht oder nicht, haben die Kanten in diesem Fall kein Gewicht. Es handelt sich um einen sog. ungewichteten Graphen. Es besteht immer dann eine Freundschaft zwischen zwei Personen, wenn eine Kante zwischen den entsprechenden Knoten existiert.

a) Gehört Jens zu Lisas Clique? Begründen Sie.

b) Skizzieren Sie den zugehörigen Freundschaftsgraphen.

c) Implementieren Sie eine Methode *freundschaftsGraphErzeugen()*, die den in der Abbildung dargestellten Graphen erzeugt. Verwenden Sie die Methoden aus dem Implementationsdiagramm von Seite 212.

d) Formulieren Sie einen Algorithmus, der auf den Adjazenzlisten arbeitet und die Frage aus a) beantwortet.

e) Der Grad eines Knotens ist die Anzahl seiner Nachbarknoten, also die Anzahl der Kanten, die sich in diesem Knoten treffen.
   1. Erläutern Sie, welche Bedeutung der Grad eines Knotens für das Beispiel des Freundschaftsgraphen hat.
   2. Implementieren Sie eine Methode *getGrad(n: GraphNode): int*, welche den Grad des Knotens n bestimmt.
   3. Den kleinsten Grad eines Knotens in einem Graphen bezeichnet man als Minimalgrad des Graphen. Implementieren Sie eine Methode *minimalgrad(g : Graph) : int*, welche den Minimalgrad des Graphen g ermittelt.

11.11.19

Datenschutz Datensicherheit Datensicherung

RAID: redundant array of independant disks

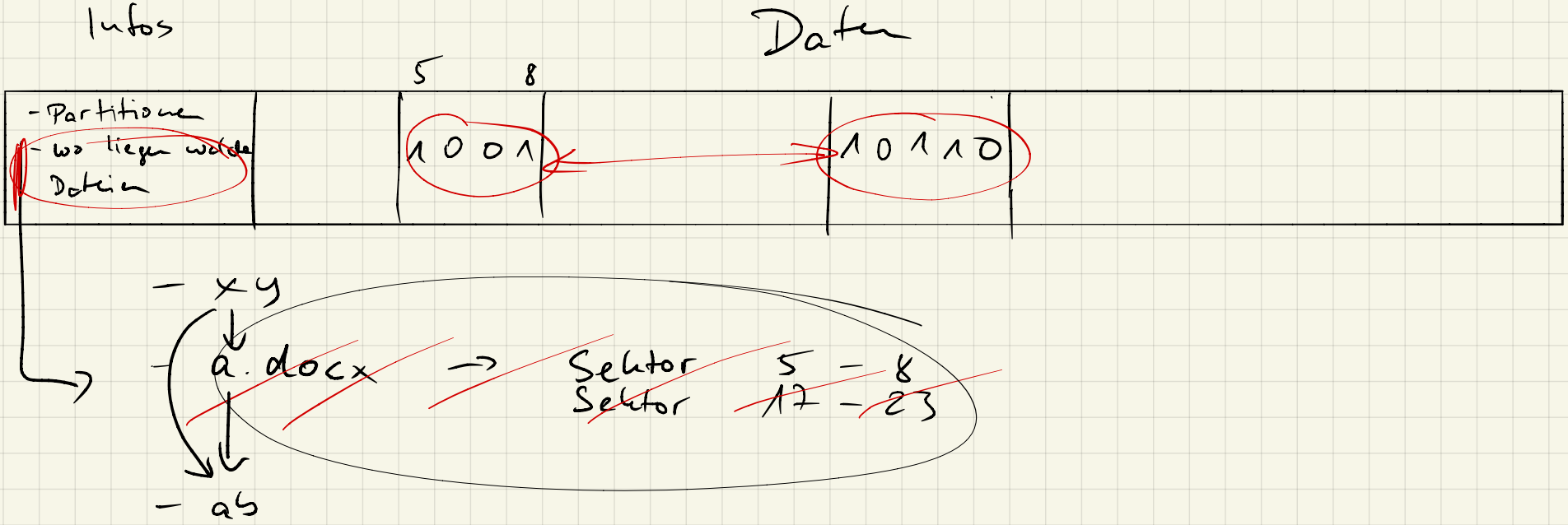
DE ☐ 642 500 40 0123456789

S.M.A.R.T.

1 TB → 1000 000 000 000 B
→ 976 562 500 kiB
→ 953 674 MiB
931,3 GiB

1099511628...

1 kB → 1000 B
1 kiB → 1024 B


Infos
Daten
5
8
- Partitionen
- wo liegen welche Dateien
1001
10110
- xy
a.docx → Sektor 5 - 8
Sektor 17 - 23
- ab

Datenschutz | Datensicherheit | Datensicherung

Datensicherung: z.B. Backups schützen vor Datenverlust

→ Strategien:
- RAID: mehrere Festplatten
- unterschiedliche Orte
- unterschiedliche Versionen / Generationen

Datensicherheit: schützt vor fremdem Zugriff
- Firewall
- Rechte (Software + räumlich)
- Verschlüsselung (Daten + Übertragung)
- Sicherheitsupdates
- Passwörter → gehasht speichern mit salt

Datenschutz: regelt wer welche persönlichen Daten haben darf

DSG-VO → Datenschutz-Grundverordnung (EU)

BDSG → Bundes-Datenschutzgesetz

GG → Persönlichkeitsrechte

KUG → Kunsturhebergesetz

Anlage 2

Fall A1

Da mittlerweile die kompletten Personaldaten per Computer verwaltet werden, haben die alten Personalbögen ausgedient. Herr Krohn, ein Mitarbeiter der Personalabteilung, nimmt den Inhalt mehrer Ordner und wirft die Unterlagen in den Papiermüll, der direkt vor dem Gebäude steht.

a) Liegt ein Verstoß gegen das Datenschutzgesetz vor? Begründen Sie Ihre Antwort!
b) Welche Daten sind betroffen und in welche Teilbereiche lassen sich diese gliedern? Nennen Sie bitte jeweils Beispiele.
c) Bei welchen Tätigkeiten findet das Datenschutzgesetz Anwendung?
d) Wie hätte Herr Krohn Ihrer Meinung nach mit den Akten umgehen sollen?

Fall B2

Stefan Schwarz, ein ehemaliger Mitarbeiter der Personalabteilung, trifft am Abend seinen Finanzberater Herrn Gierig. Dieser ist hoch interessiert an einer Liste mit Namen, Adressen und Gehältern der ehemaligen Kollegen von Herrn Schwarz und zahlt ihm 1.000,- € für die Informationen.

a) Unterliegt die Liste dem Datenschutz? Begründen Sie Ihre Antwort!
b) Verstößt Herr Schwarz gegen das Datengeheimnis?
c) Wann hätte Herr Schwarz die Liste weitergeben dürfen?

Fall C3

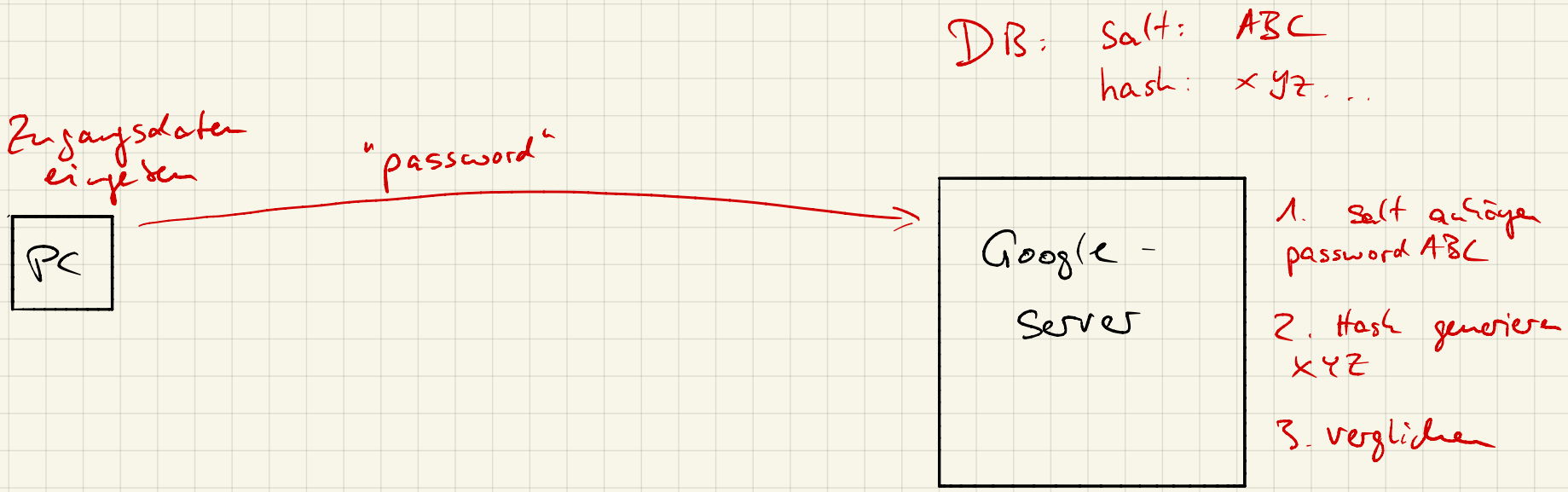
Der Personalchef wendet sich aufgrund der datenschutzrechtlichen Probleme an die Geschäftsleitung. Dort lagen bisher nie Beschwerden vor und man ist nicht gewillt wegen sieben Mitarbeitern in der Personalabteilung großen Aufwand zu betreiben. Datenschutz ist nicht so wichtig, hier kennt doch jeder jeden, Kontrolle ist da unnötig.

a) Muss aufgrund der Mitarbeiterzahl der Personalabteilung eine besondere Datenschutzmaßnahme getroffen werden, wenn ja welche?
b) Wann muss ein Datenschutzbeauftragter eingesetzt werden?
c) Welche Bestimmungen sind bzgl. eines Datenschutzbeauftragten zu beachten?
d) Welche Aufgaben hat ein Datenschutzbeauftragter? Nennen Sie Beispiele.

Fall D4

Der Personalchef beschwert sich:
„Immer wieder kommen Mitarbeiter und wollen Auskunft über ihre Personalakte. Zu jenem Eintrag etwas wissen, dort etwas ändern, und der Gipfel war die Forderung, Informationen zu löschen. Alles Verstöße gegen den Datenschutz!“

Erklären Sie ausführlich die Rechte eines betroffenen Mitarbeiters.



1234 -> 10

3331 -> 10

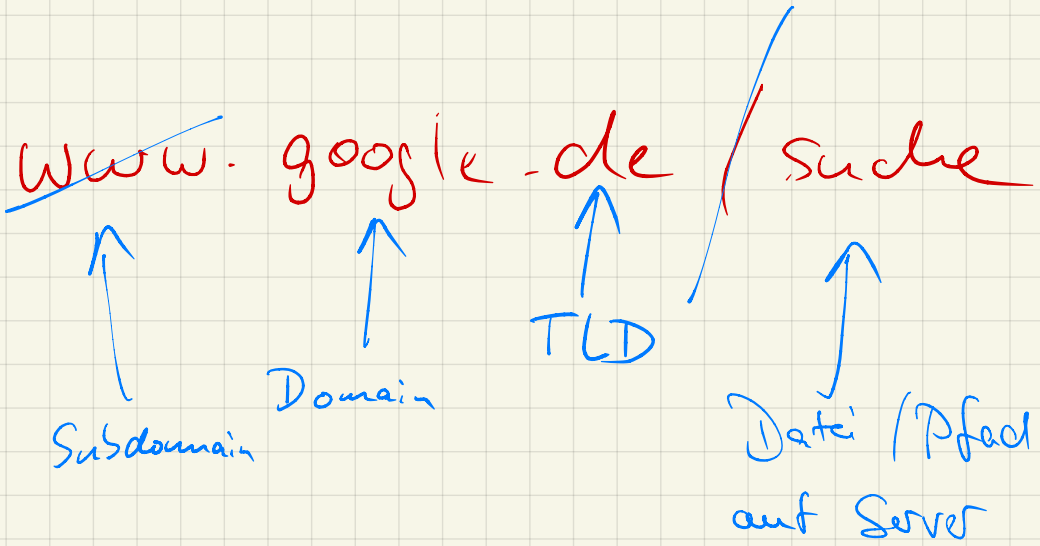
7210 -> 10

Anlage 9
Der neue Datenschutzbeauftragte stellt bei einer Prüfung folgende Zustände fest:

Damit sich die Mitarbeiter Passwörter besser merken können, vergibt man im ganzen Unternehmen für alle Programme das gleiche Passwort.
Gefahr: unsicher, einmal knacken alles machen unberechtigter Zugriff
Maßnahmen: - eigene Passwörter - PW regelmäßig ändern

Die Installation von unlizenzierter Software führt immer wieder zu Systemabstürzen, auch während der Erfassung neuer Mitarbeiterdaten.
Gefahr: – Malware – Cryptotrojaner – Spyware – Datenverlust
Maßnahmen: - Zugriffsberechtigungen - Administrator(en) benennen - keine betriebsfremden Programme

Einmal im Quartal werden die Daten der Personalabteilung auf CD gebrannt. Die Cds stehen auf dem Fensterbrett, bis ein besserer Platz gefunden ist.
Gefahr: – Datenverlust (z.B. durch Sonneneinstrahlung) – einfache Kopien - Beschädigung der CDs – Diebstahl - altes Medium
Maßnahmen: – sichere Aufbewahrung - Backups



www. google.de . suche . schoen . dhg-rw.de

Die neue Drohnen-Verordnung

1 **Kennzeichnungspflicht:** Ab 0,25 kg muss eine Plakette mit Namen und Adresse des Eigentümers angebracht werden – auch auf Modellfluggeländen.

2 **Kenntnisnachweis:** Ab 2,0 kg müssen besondere Kenntnisse nachgewiesen werden.

3 **Erlaubnispflicht:** Ab 5,0 kg wird eine spezielle Erlaubnis der Landesluftfahrtbehörde benötigt.

4 **Grundsätzlich verboten:** Ab 100m dürfen Drohnen nur fliegen, wenn eine behördliche Ausnahmeerlaubnis bei den Landesluftfahrtbehörden eingeholt wurde.

Weitere Überflugverbotsbereiche siehe: www.bmvi.de/drohnen

Netzwerke: Begriffe und Grundlagen

1. Begriffe – „Netzwerk“

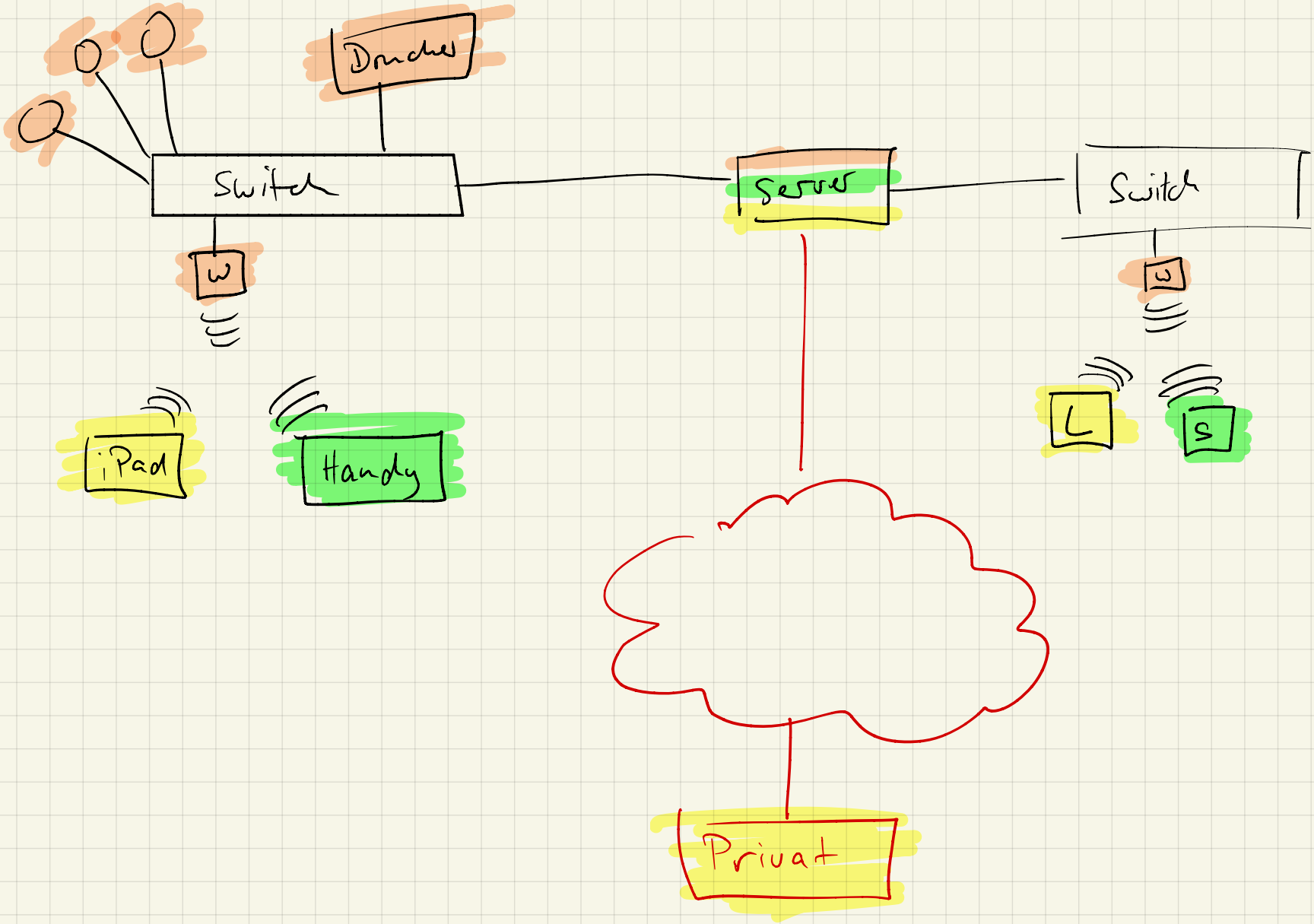
Beschreibe die folgenden Begriffe und gib, sofern möglich, Einsatzzwecke davon an:

- Netzwerk (allgemein):
  miteinander verbundene Geräte, zur Kommunikation, Datenaustausch, um Ressourcen gemeinsam nutzen zu können
- LAN / WLAN:
  (wireless) local area network, räumlich begrenztes Netzwerk
- WAN:
  wide area network, weitläufiges Netzwerk, besteht aus mehreren LANs
- ARPANET: advanced research projects agency network
  Vorläufer des Internets, entwickelt vom US-Militär
- Internet:
  Ansammlung von vielen miteinander verbundenen Netzwerke, meist synonym zum WWW, öffentlich einfach zugänglich

2. Begriffe – „Netzwerktopologie“

Beschreibe die folgenden Begriffe:

- Netzwerktopologie (allgemein):
  Topologie = Struktur = Aufbau
- physische Topologie:
  physikalisch
  konkrete Vernetzung

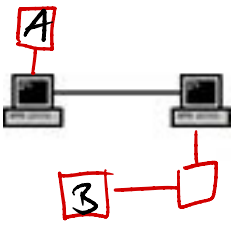
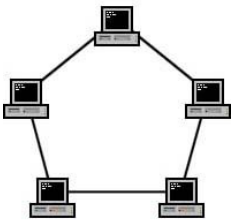
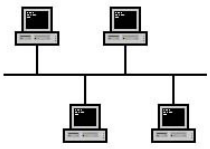
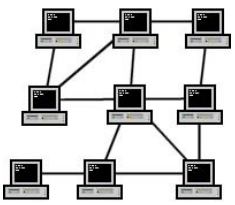
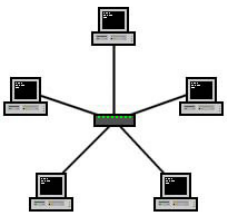

Drucker
Switch
Server
Switch
W
W
iPad
Handy
L
S
Privat

virtuelle
- logische Topologie:

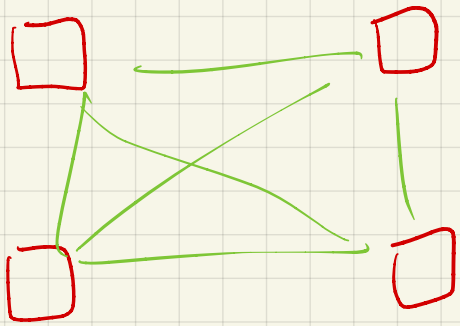
Softwareseitige Konfiguration / Aufteilung des Nutzwertes

3. Topologiearten

Benenne die folgenden Topologiearten und gib einige Vor- und Nachteile an:

	Peer-to-Peer (P2P) / Direkt / Linie Verkabelung + einfach	Erweiterbarkeit + gut	Ausfallsicherheit - nicht ausfallsicher	Datensicherheit - jeder PC dazwischen kann abhören
	Ring Verkabelung • etwas mehr	Erweiterbarkeit • relativ gut	Ausfallsicherheit • etwas besser	Datensicherheit - jeder PC dazwischen kann abhören
	Bus Verkabelung • ähnlich Ring	Erweiterbarkeit + sehr einfach	Ausfallsicherheit + PC egal - Hauptleitung blöd	Datensicherheit - jeder PC kann abhören
	Vermascht Verkabelung + sehr aufwändig	Erweiterbarkeit • relativ einfach	Ausfallsicherheit • recht gut	Datensicherheit • relativ gut
	Stern Verkabelung + sehr einfach	Erweiterbarkeit + easy	Ausfallsicherheit + Kabel / PC egal - Switch blöd	Datensicherheit + recht gut - Switch kann alles abhören

Bildquellen: Wikipedia: Topologie, https://de.wikipedia.org/wiki/Topologie_(Rechnernetz), abgerufen am 26.5.2016

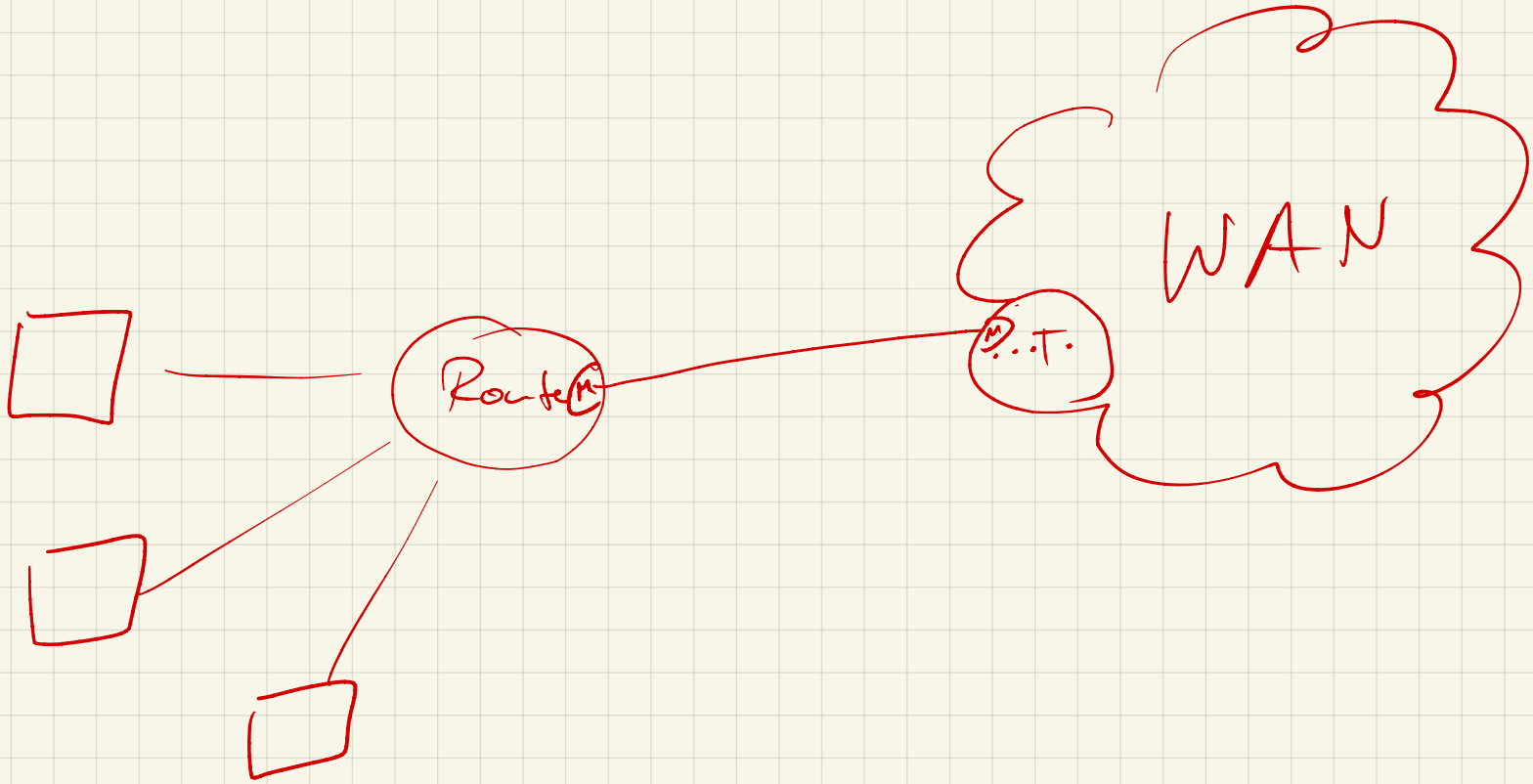


PCs	Kabel
2	1
3	3
4	6
5	10
6	15
7	21
8	28
9	36

4. Begriffe – Hardware und Software

Beschreibe die folgenden Begriffe und deren Einsatzzweck:

- Hub:
  Verteiler für Netzwerkdaten, schickt Daten an alle angeschlossenen Geräte
- Switch:
  Verteiler für Netzwerkdaten, schickt Daten nur an den richtigen Empfänger weiter
- Modem:
  Modulator - Demodulator, es werden damit Daten über eine Leitung übertragen, die ursprünglich nicht für Daten vorgesehen war.
- Router:
  Verbindet mehrere Netzwerke
- MAC-Adresse:
  Gerätebezogene, individuelle Nummer
  Hardware-/Adapterbezogen
- IP-Adresse:
  eindeutige Adresse in einem Netzwerk, logische Adresse


WAN
Router
N.A.T.

- Client-Server-Modell:

  Es gibt einen zentralen Computer ("Server") über den alles läuft, der Dienste anbietet die Clients benutzt werden

- Peer-to-Peer-Modell (P2P):

  Direkte Verbindung zwischen 2 Geräten

- Port:

  Eingang zu einem Rechner hinter dem ein Dienst liegt

- Domain:

  „sprechender" Name für eine IP

- TLD:

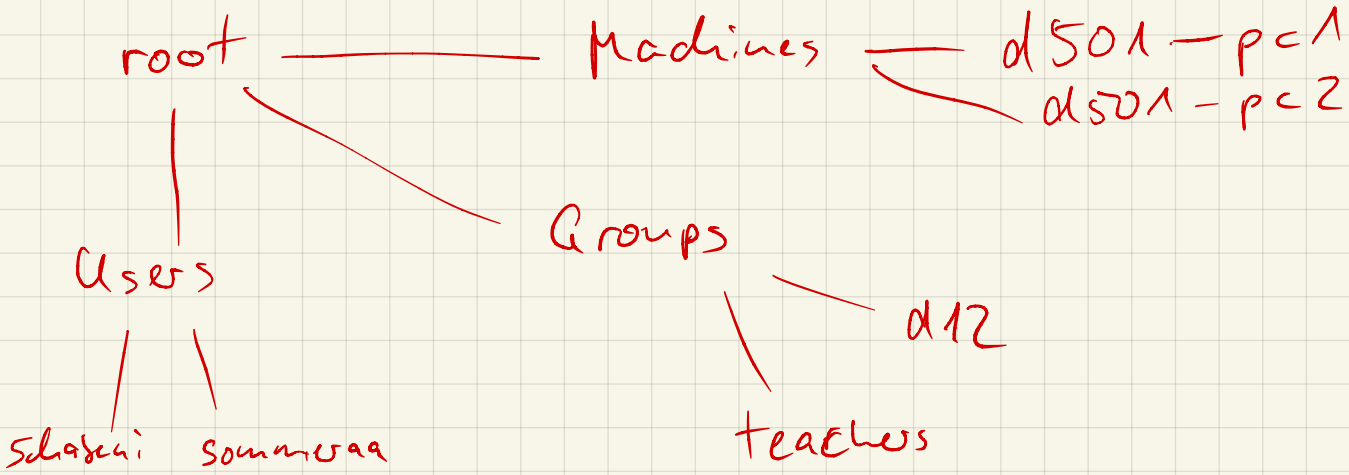
  Top-Level-Domain: Domain-Endung

- Protokoll: HTTP / FTP / ICMP / ARP / IP / TCP / UDP

  Gibt die Struktur von Nachrichten an (= Codierung)

- DNS:

  Domain Name System, löst Domain-Namen in IP-Adressen auf.



---

64 32 16 8 4 2 1
1 0 1 1 0 1 0_2 = 90_{10}

51 = 0 · 64 + 51
51 = 1 · 32 + 19
19 = 1 · 16 + 3
3 = 0 · 8 + 3
3 = 0 · 4 + 3
3 = 1 · 2 + 1
1 = 1 · 1 + 0

51 = 00110011

91

128 64 32 16 8 4 2 1
0 1 0 1 1 0 1 1

Netzwerke: Adressierung

1. IP-Adressen

Zur Kommunitation innerhalb eines Netzwerkes wird jedem Rechner eine eindeutige IP-Adresse zugewiesen. Diese sind in der noch gebräuchlichsten Version (IPv4) in 4 Blöcke mit je 8 Bit aufgeteilt. Es sind also theoretisch die Adressen von 0.0.0.0 bis 255.255.255.255 möglich.
Zum besseren Verständnis ist eine Darstellung im Binärsystem hilfreich.

2. Übung: Umrechnung Dezimalsystem ↔ Binärsystem

Wiederhole die Umrechung zwischen dem Binärsystem und dem Dezimalsystem. Wandle anschließend in das jeweils andere Stellenwertsystem um.

a) $1101\ 1110_2$ 222 c) $1111\ 1101_2$ 253 e) 13_{10} 0000 1101 g) 254_{10} 1111 1110

b) $0011\ 1111_2$ 63 d) $0101\ 1010_2$ 90 f) 96_{10} 0110 0000 h) 127_{10} 0111 1111

3. Subnetzmaske

Eine *Subnetzmaske* ist in IPv4 ebenfalls eine 32-Bit-Zahl, welche eine IP-Adresse in *Netzwerkteil* und *Geräteteil* trennt.
Die Subnetzmaske gibt quasi an, welche Geräte direkt miteinander kommunizieren können, bzw. welche Geräte in einem *logischen* Netz verbunden sind.
Durch UND-Verknüpfung der IP mit der Subnetzmaske erhält man den Netzwerkteil. Alle Geräte mit dem selben Netzwerkteil gehören zum selben logischen Netzwerk.
Durch UND-Verknüpfung mit der invertierten Subnetzmaske erhält man den Geräteteil.
Mit dem Netzwerkteil ergibt sich dann auch die kleinstmögliche und größtmögliche IP-Adresse. Diese beiden Adressen dürfen nicht an Geräte vergeben werden, sie sind für die *Netzwerkadresse* (identisch zum Netzwerkteil) bzw. die *Broadcastadresse* reserviert.
Beispiel:

IP-Adresse:	192.145.96.201	=	11000000.10010001.01100000.11001001
Subnetzmaske:	255.255.255.240	=	11111111.11111111.11111111.11110000
Netzwerkteil:	192.145.96.192	=	11000000.10010001.01100000.11000000
invertierte Subnetzmaske:	0.0.0.15	=	00000000.00000000.00000000.00001111
Geräteteil:	0.0.0.9	=	00000000.00000000.00000000.00001001
Netzwerkadresse:	192.145.96.192	=	11000000.10010001.01100000.11000000
Broadcastadresse:	192.145.96.207	=	11000000.10010001.01100000.11001111
kleinste nutzbare Adresse:	192.145.96.193	=	11000000.10010001.01100000.11000001
größte Nutzbare Adresse:	192.145.96.206	=	11000000.10010001.01100000.11001110

192.145.96.201/28

Vervollständige die Tabelle:

IP	Subnetzmaske	Netzwerkteil	Geräteteil	Broadcast	1. Adresse	letzte Adresse
192.168.213.15	255.255.255.192	192.168.213.0	0.0.0.15	192.168.213.63	192.168.213.1	192.168.213.62
172.16.5.254	255.255.255.0	172.16.5.0	0.0.0.254	172.16.5.255	172.16.5.1	172.16.5.254
172.254.13.8	255.255.248.0	172.254.8.0	0.0.5.8	172.254.15.255	172.254.8.1	172.254.15.254
10.18.51.5	255.240.0.0	10.16.0.0	0.2.51.5	10.31.255.255	10.16.0.1	10.31.255.254
10.0.0.15	255.0.0.0	10.0.0.0	0.0.0.15	10.255.255.255	10.0.0.1	10.255.255.254

IP:

10.18.51.1 = 00001010.00010011.00110011.00000001

Subnetzmaske:

255.240.0.0 = 11111111.11110000.00000000.00000000

Netzwerkteil bzw. Netzwerkadresse:

10.16.0.0 = 00001010.00010000.00000000.00000000

Geräteteil

0.2.51.1 = 00000000.00000010.00110011.00000001

erste Adresse

10.16.0.1 = 00001010.00010000.00000000.00000001

letzte Adresse

10.31.255.254 = 00001010.00011111.11111111.11111110

10.10.2.5 = 00001010.00001010.00000010.00000101

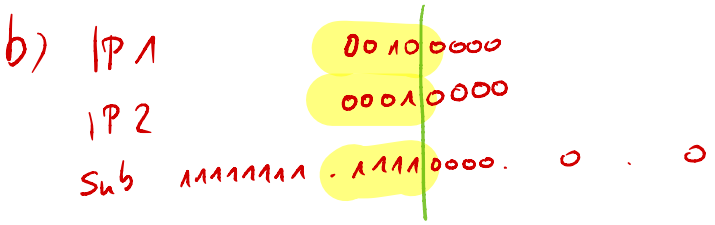
255.255.0.0 = 11111111.11111111.00000000.00000000

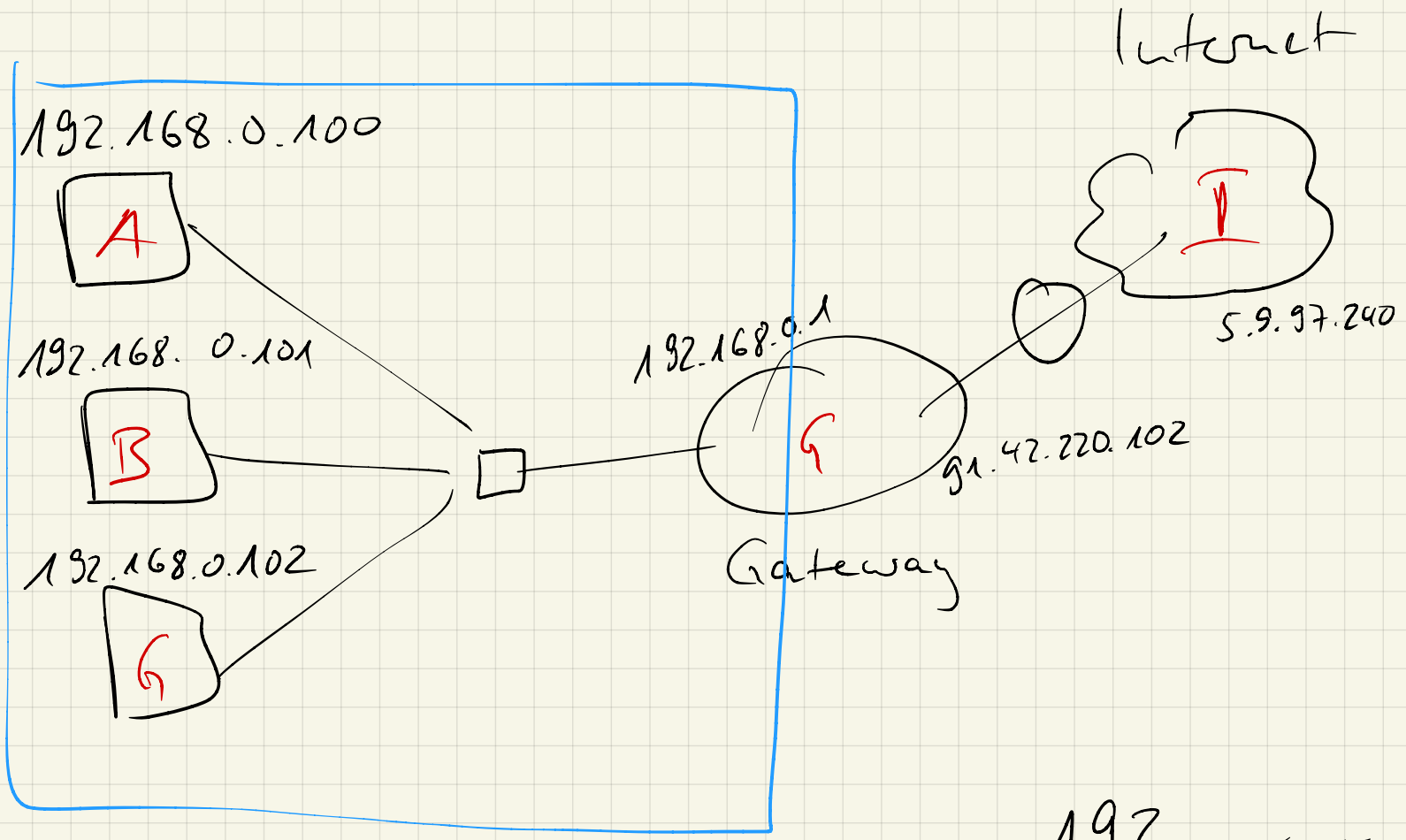
10.10.0.0 = 00001010.00001010.00000000.00000000

4. Übungen

a) Eine Nachricht wird im Netzwerk mit der Subnetzmaske 255.255.248.0 von einem Rechner mit der IP 192.168.203.15 an einen Rechner mit der IP 192.168.200.65 geschickt. Bleibt die Nachricht im Netzwerk oder muss sie über das Internet verschickt werden?

b) Eine Nachricht wird im Netzwerk mit der Subnetzmaske 255.240.0.0 von einem Rechner mit der IP 10.32.100.12 an einen Rechner mit der IP 10.16.1.1 geschickt. Bleibt die Nachricht im Netzwerk oder muss sie geroutet werden?




Internet
192.168.0.100
A
192.168.0.101
B
192.168.0.102
G
192.168.0.1
G
91.42.220.102
I
5.9.97.240
Gateway

192 . . . . .

10 . . . . . .

OSI +Layer -Modell

Please Do Not Throw Salami Pizza Away

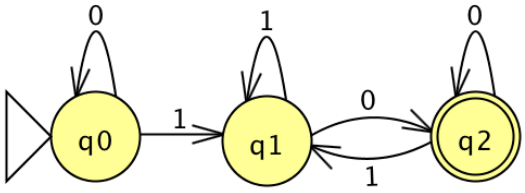
1. Physical / Bitübertragung
2. Data Link / Sicherung — MAC
3. Network / Vermittlung — IP
4. Transport — TCP / UDP
5. Session / Sitzung
6. Presentation / Darstellung
7. Application / Anwendung

} Anwendung: HTTP / FTP / SMTP

- Automaten
- Datenbanken
- OOP
- Abstrakte Datentypen

I B2 Automaten und formale Sprachen

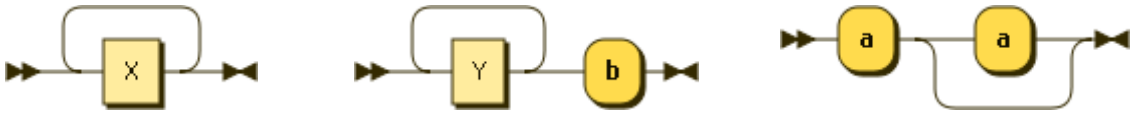
B2.1 Gegeben ist der endliche Automat M mit dem Eingabealphabet $\Sigma = \{0, 1\}$ durch das folgende Zustandsübergangsdiagramm:



- Überprüfen Sie, ob M die Wörter
  i. 1100
  ii. 1011
  akzeptiert und geben Sie jeweils die Folge der durchlaufenen Zustände an.
- Geben Sie ein Wort der Länge 5 an, das der Automat nicht akzeptiert.
- Beschreiben Sie die Menge der Wörter, die der Automat akzeptiert.
- Überführen Sie den Automaten in eine reguläre Grammatik, welche die von ihm erkannte Sprache erzeugt.

(5 VP)

B2.2 Gegeben ist das folgende Syntaxdiagramm der Sprache L:



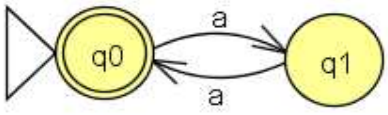
S X Y

- Geben Sie drei Wörter an, die in L liegen.
- Erläutern sie an Hand des Syntaxdiagramme die Begriffe Terminalsymbol und Nichtterminalsymbol.
- Geben Sie an, welche Sprache L dadurch erzeugt wird.
- Begründen Sie, dass Wort „aabbaa“ nicht in L liegt.
- Entwerfen Sie einen endlichen Automaten, der genau L akzeptiert.
- Geben Sie eine reguläre Grammatik an, die L erzeugt.

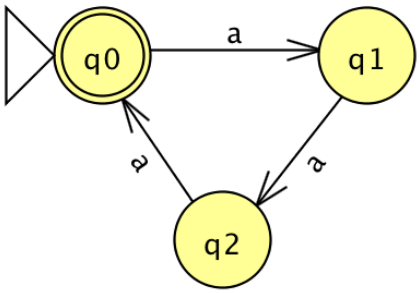
(7 VP)

B2.3 M_1 und M_2 sind zwei endliche Automaten mit dem Eingabealphabet $\Sigma = \{a\}$:

M_1:



M_2:

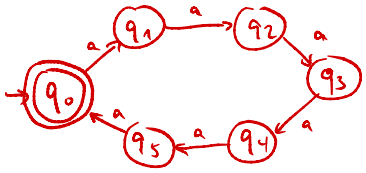


M_1 erkennt die Sprache L_1 und M_2 erkennt die Sprache L_2.

- Beschreiben Sie die Menge aller Wörter in L_1 und L_2.

M_3 ist ein Automat, der genau die Wörter erkennt, die sowohl von M_1 als auch von M_2 erkannt werden. Es sei L_3 die von M_3 erkannte Sprache.

- Beschreiben Sie die Menge der Wörter in L_3.
- Entwerfen Sie den Automaten M_3.



(4 VP)

B2.4 Eine Ohrmarke dient der amtlichen Kennzeichnung und Registrierung von Haus- und Nutztieren. Diese besteht aus dem zweistelligen Ländercode `de` gefolgt von dem Bundesland, das durch eine Ziffernfolge `01,02,03,…,15,16` gekennzeichnet ist sowie einer weitere individuellen achtstelligen Zahl, die auch führende Nullen enthalten kann.

- Geben Sie eine reguläre Grammatik (Σ, V, S, P) über dem Terminalsymbolalphabet $\Sigma = \{0,1,2,3,4,5,6,7,8,9,d,e\}$ an, welche die oben beschriebene Sprache erzeugt.
- Begründen Sie an Hand Ihrer Grammatik, ob sich die folgenden Wörter aus Ihrer Grammatik ableiten lassen:
  i. `de0412345678`
  ii. `de1753198124`
  iii. `de107610358`
- Entwerfen Sie ein den Übergangsgraphen eines deterministischen endlichen Automaten, der die Sprache der korrekten Ohrmarkennummern erkennt.

(4 VP)

P: $S \to deB$

$B \to 01 \mid 02 \mid 03 \mid \dots \mid 16Z$

$Z \to 00000000 \mid 00000001 \mid \dots \mid 99999999$

$T = \{de, 01, 02, \dots, 16, 00000001, \dots, 99999999\}$

$N = \{S, B, Z\}$

$S = S$

$S \to d\ A$

$A \to e\ B$

$B \to 0\ C \mid 1\ D$

$C \to 1 \mid 2 \mid \ldots \mid 9\ E$

$D \to 0 \mid 1 \mid \ldots \mid 6\ E$

$E \to 0 \mid \ldots \mid 9\ F$

$F \to 0 \mid \ldots \mid 9\ G$

$G \to 0 \mid \ldots \mid 9\ H$

$H \to 0 \mid \ldots \mid 9\ I$

$I \to 0 \mid \ldots \mid 9\ J$

$J \to 0 \mid \ldots \mid 9\ K$

$K \to 0 \mid \ldots \mid 9\ L$

$L \to 0 \mid \ldots \mid 9$

1.

2.

3.

4.

5.

6.

7.

8.

II B2 Automaten und formale Sprachen

B2.1 Ein einfacher Fahrscheinautomat kann wie folgt beschrieben werden:

- Eine Fahrkarte für Erwachsene (E) kostet 4.- €, eine Kinderfahrkarte (K) 2.- €.
- In den Automaten können 1.- € und 2.- € Münzen eingeworfen werden.
- Werden mehr Münzen eingeworfen, verbleiben diese zunächst im Automaten.
- Drückt man den Knopf für die Geldrückgabe (R), erhält man alle Münzen zurück.
- Wählt man eine der Fahrkartentasten (E, K), so wird die entsprechende Fahrkarte ausgedruckt und eventuelles Rückgeld ausgeworfen, sofern genügend eingezahlt wurde. Andernfalls passiert nichts.

Dieser Fahrscheinautomat wird durch einen endlichen Automaten modelliert.

- Geben Sie jeweils eine geeignete passende Eingabemenge E und Ausgabemenge A an.
- Welche Informationen über zurückliegende Handlungen, muss der Automat speichern, um korrekt zu reagieren? In welcher Form geschieht dies im Modell?

  Am Anfang (z. B. nach dem Einschalten) ist der Automat in einem definierten Anfangszustand z_0. Dieser ist zugleich auch ein Endzustand. Wenn man darauf verzichtet, die Art der eingeworfenen Münzen zu speichern, kann die Zustandsmenge $S = \{z_0, z_1, z_2, z_3, z_4, z_5\}$ mit den weiteren Zuständen z_1 bis z_5 wie folgt definiert werden:

  z_0: kein Geld eingeworfen
  z_1: 1.- € eingeworfen
  z_2: 2.- € eingeworfen
  z_3: 3.- € eingeworfen
  z_4: 4.- € eingeworfen
  z_5: um 1.- € überzahlt (weitere Überzahlungen sollen nicht möglich sein)

  Jedem Zustand $s \in S$ und jeder Eingabe $e \in E$ wird durch die Übergangsfunktion δ ein neuer Zustand $s' \in S$ zugeordnet.

- Geben Sie die Übergangsfunktion δ in Form einer Tabelle an.
- Zeichnen Sie das vollständige Zustandsübergangsdiagramm für diesen Fahrscheinautomaten, das auch die Ausgaben enthält.
- Begründen Sie, warum es sich bei diesem Modell eines Fahrkartenautomaten um einen deterministischen endlichen Automaten handelt.

(10 VP)

(bitte wenden ↳)

B2.1:

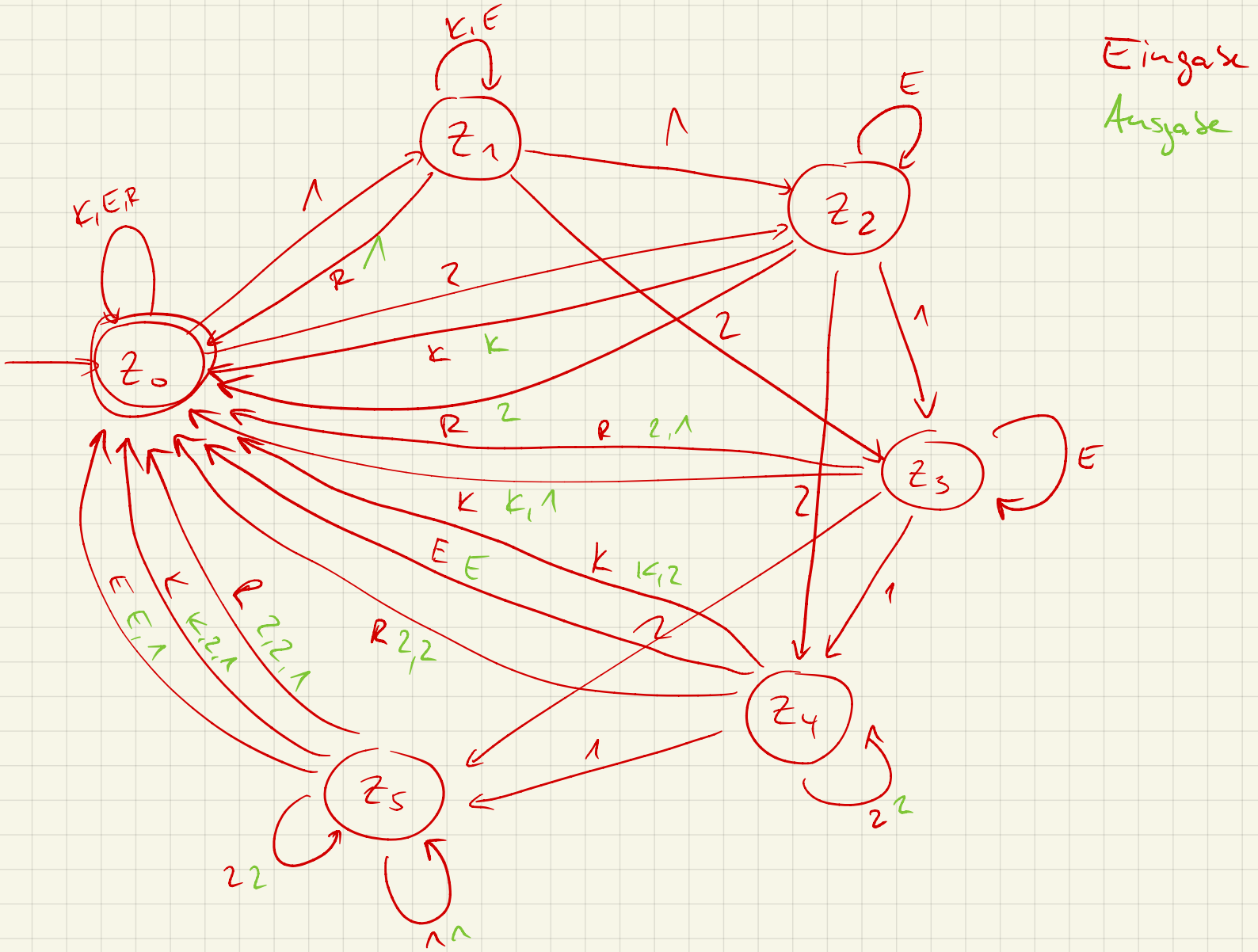
$E = \{1, 2, E, K, R\}$

$A = \{1, 2, E, K, -\}$

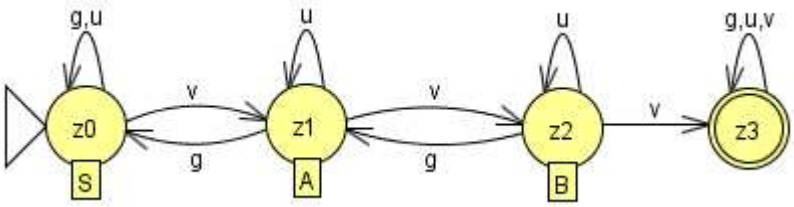
δ:

z_{alt}	Eingabe	z_{neu}
z_0	1	z_1
z_0	2	z_2
z_0	E	z_0
z_0	K	z_0
z_0	R	z_0

z_{alt} \ E	1	2	E	K	R
z_0	z_1 -	z_2 -	z_0 -	z_0 -	z_0 -
z_1	z_2 -	z_3 -	z_1 -	z_1 -	z_0 1
z_2	z_3 -	z_4 -	z_2 -	z_0 K	z_0 2
z_3	z_4 -	z_5 -	z_3 -	z_0 K,1	z_0 2,1
z_4	z_5 -	z_4 2	z_0 E	z_0 K,2	z_0 2,2
z_5	z_4 2	z_5 2	z_0 E,1	z_0 K,2,1	z_0 2,2,1


Eingabe
Ausgabe
z_0
z_1
z_2
z_3
z_4
z_5
K,E
K,E,R
E
1
R
2
K
R 2
R 2,1
K K,1
E E
K K,2
E E,1
K K,2,1
R 2,2,1
R 2,2
2 2
1 1

B2.1 Der Vorstand einer Fußballvereins möchte jeglichen Diskussionen über eine mögliche Trainerentlassung nach verlorenen Spielen entgegenwirken. Er hat daher ein Regelwerk aufgestellt, aus dem eindeutig hervorgeht, unter welchen Bedingungen ein Trainer entlassen wird. Der folgende Übergangsgraph eines endlichen Automaten modelliert dieses Regelwerk. Das Eingabealphabet ist $\Sigma = \{g, u, v\}$. (g: gewonnen, u: unentschieden, v: verloren)



- Geben Sie eine Folge von Zeichen aus dem Eingabealphabet Σ der Länge 10 an, die von dem Automaten akzeptiert wird.
- Erläutern Sie anhand des Übergangsgraphen, unter welche Bedingungen der Trainer entlassen wird.
- Entwickeln Sie eine Grammatik, die genau diejenigen Wörter erzeugt, die von dem obigen Automaten akzeptiert werden.
  Bestimmen sie für das Wort vvguvv die Ableitung aus dieser Grammatik.

Der Vereinsvorstand musste zurücktreten. Der neue Vorstand möchte ein neues Regelwerk für eine Trainersuspendierung einführen. Man ist sich einig, dass jeder der beiden folgenden Umstände zu einer Trainerentlassung führt.

- Das Team hat dreimal hintereinander verloren.
- Das Team hat viermal hintereinander nicht gewonnen.

- Geben Sie zu jedem der beiden Fälle eine Spielergebnisfolge aus {g, u, v] an, die zu einer Trainerentlassung nach 6 Spielen führt.
- Entwerfen Sie den Übergangsgraphen eines deterministischen endlichen Automaten, der das Eingabealphabet $\Sigma = \{g, u, v]$ besitzt und genau die Folgen von Spielausgängen akzeptiert, die zu einer Trainerentlassung nach dem neuen Regelwerk führen?
- Geben Sie für die Abarbeitung der folgenden Eingaben die zugehörigen Zustandsfolgen an und geben Sie begründet an, ob der Automat jeweils die Eingabe akzeptiert oder nicht.

  i: gugvvv

  ii: uvugvvu

(10 VP)

$N = \{S, A, B\}$

$T = \{u, v, g\}$

$S = S$

$P =$

$S \to uS$
$S \to gS$
$S \to vA$
$A \to uA$
$A \to vB$
$A \to gS$
$B \to uB$
$B \to gA$
$B \to v$

$S \to uS \mid gS \mid vA$

$A \to uA \mid vB \mid gS$

$B \to uB \mid gA \mid v$

vvguvv

S
vA
vvB
vvgA
vvguA
vvguvB
vvguvv

vguvugv

S
vA
vgS
vgvA
vguB
vguvuB
vguvugA
vguvugvB

27.1.20

B2.1

a) i) ichwillabi

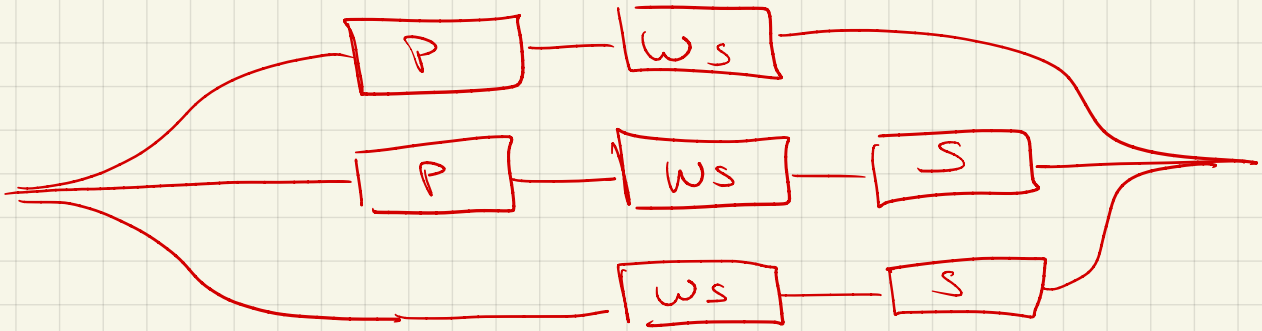
ii) Kabinett_17

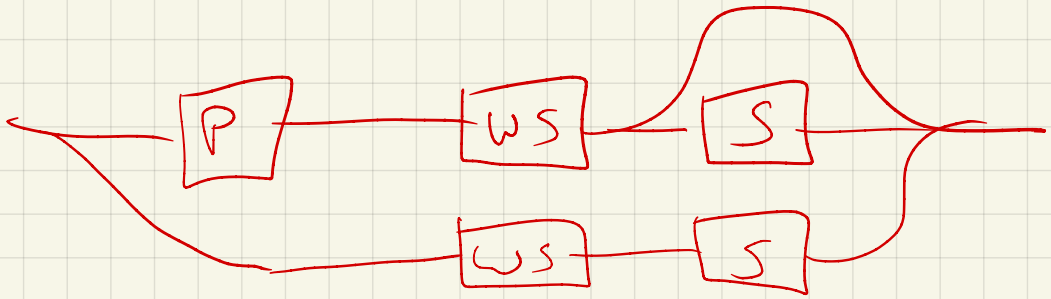
iii) abgeschrieben_88

iv) abitur_1

b) rechtschreibungistunwichtig_12

c)





2.2

Σ = Vokabular

T = Terminale

V = N = Nichtterminale

S = Start

P = Produktionsregeln

S = 0A | 1B | 2B | 3C

A = 1.D | 2.D | 3.D | 4.D | 5.D | 6.D | 7.D | 8.D | 9.D

B = 0.D | 1.D | 2.D | 3.D | 4.D | 5.D | 6.D | 7.D | 8.D | 9.D

C = 0.D | 1.D

⋮

oder

T = {01, 02, ..., 31, 1900, 1901, ..., 1999, .}

N = {T, M, J}

T = 01.M | ... | 31.M

M = 01.J | ... | 12.J

J = 1900 | ... | 1999

oder

$N = \{S, T, M, J, Z\}$

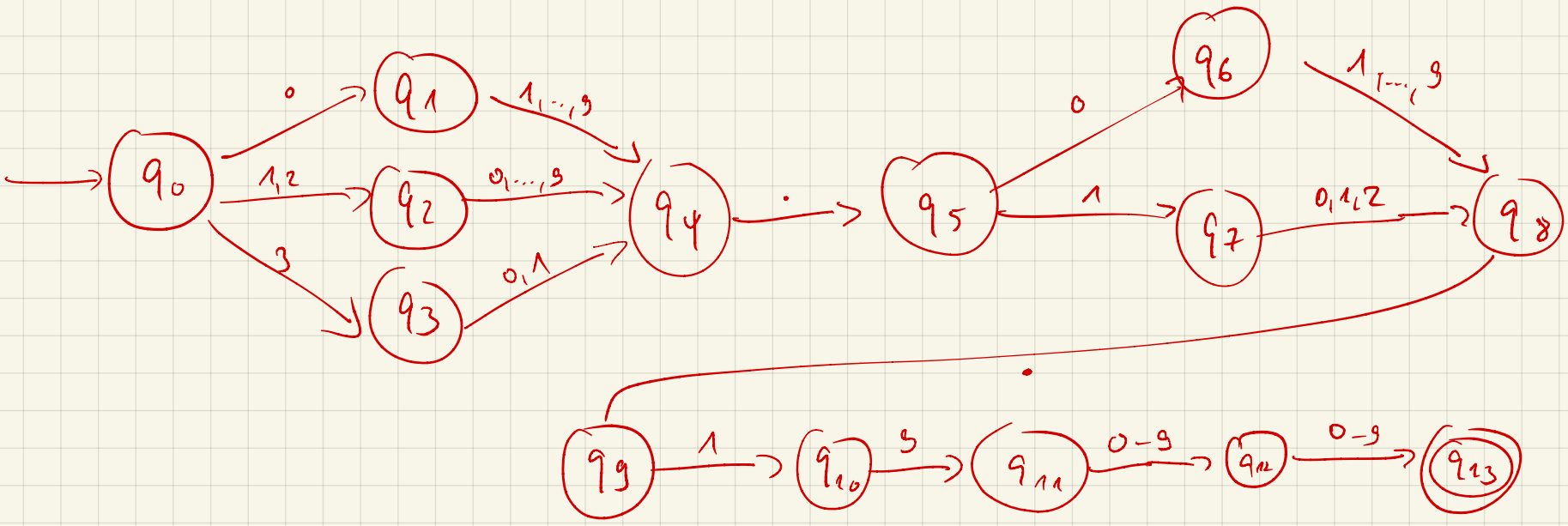
$S \rightarrow T.M.J$

$T \rightarrow 01 \mid 02 \mid \ldots \mid 31$

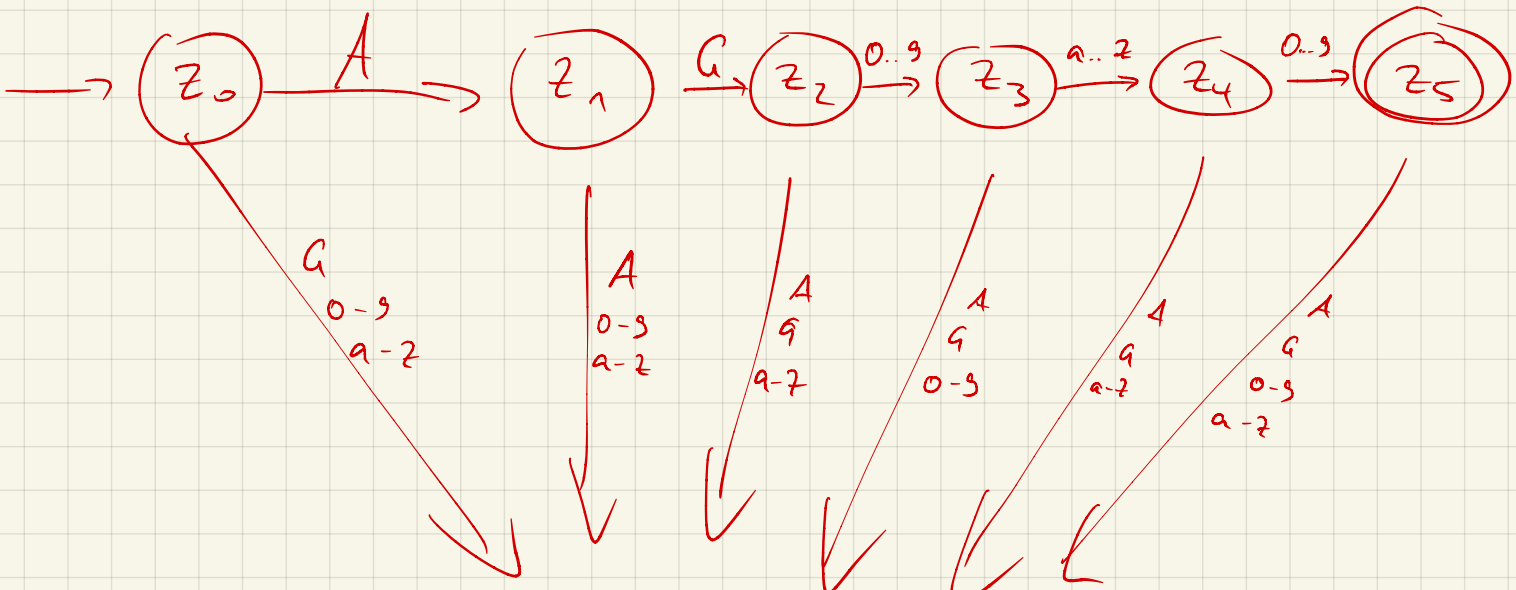
$M \rightarrow 01 \mid \ldots \mid 12$

$J \rightarrow 19ZZ$

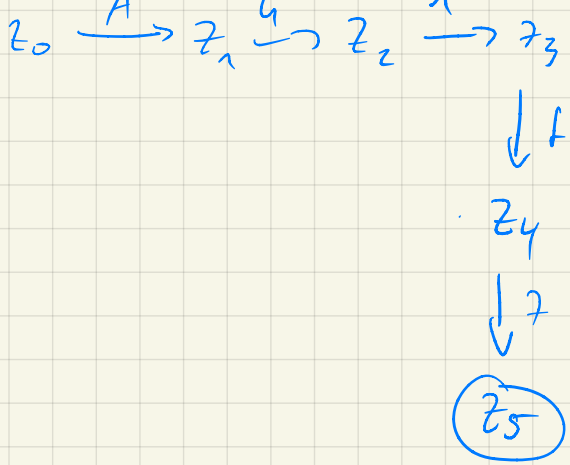
$Z \rightarrow 0 \mid \ldots \mid 9$



2.3

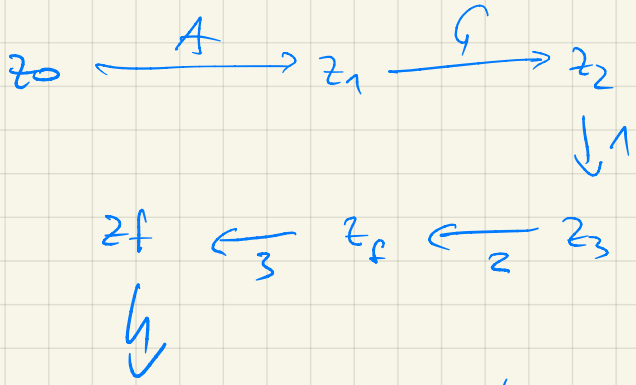


AG1f7



✓ befindet sich im Endzustand

AG123



↯ befindet sich nicht im Endzustand

I B1 Datenbanken

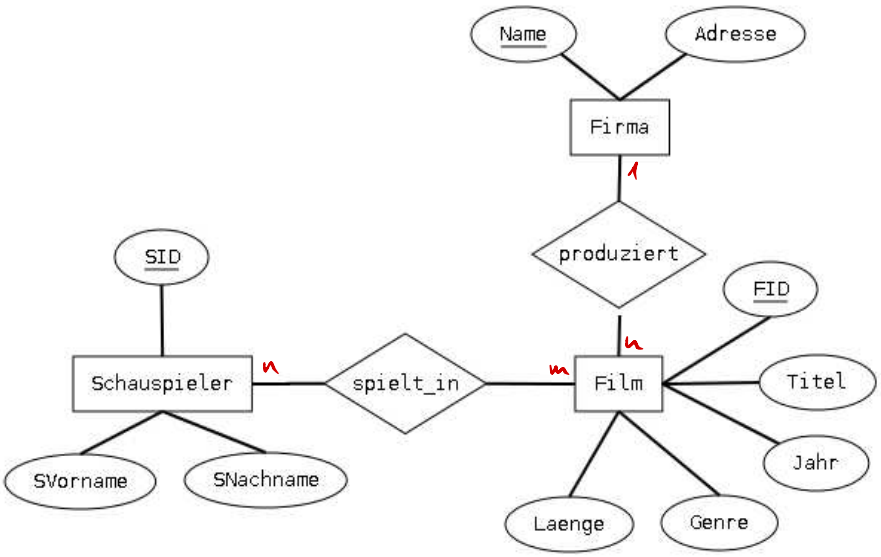
B1.1 Gegeben ist die folgende Tabelle einer Datenbank, mit der ein Elektronik-Händler seine verkauften Smartphones verwaltet:

Hersteller	Bezeichnung	Preis in €	Kaufdatum	Kunde
Tony	Expert3	539,50	10.10.2014	Harry Baumeister, Gmünd
Samson	Merkur4	682,85	22.10.2014	Robert Fruchtig, Semmelrode
Samson	Sun5	456,90	15.09.2014	Maria Lustig, Freiburg
Samson	Sun5	456,90	15.09.2014	Maria Lustig, Freiburg
Pear	xPhone3	889,00	11.10.2014	Harry Baumeister, Gmünd
Samson	Sun5	406,90	14.10.2014	Rita Richter, Freiburg
...	...	...	...	...

- Erläutern Sie, welche Probleme sich bei der Verwaltung der Datenbank ergeben können. Gehen Sie dabei insbesondere auf die Begriffe „Redundanz“ und „Dateninkonsistenz“ ein.
- Entwerfen Sie für die Datenbank ein optimiertes, relationales Datenbankschema, das diese Probleme vermeidet.

(6 VP)

B1.2 Gegeben ist folgendes Entity-Relationship-Diagramm:



- Geben Sie alle Beziehungskardinalitäten an und begründen Sie Ihre Wahl!

```
SELECT * FROM Film WHERE Genre='Action' AND Jahr='2012'

SELECT * FROM Film WHERE Laenge >='90' AND Laenge <='120'

SELECT COUNT(*) FROM Film WHERE Jahr ='2013'

SELECT Titel FROM Film JOIN spielt_in ON (spielt_in.FID=Film.FID)
                       JOIN Schauspieler ON (Schauspieler.SID=spielt_in.SID)
                       WHERE Svorname = 'Bruce' AND SNachname='Willy'
                       ORDER BY Jahr
```

Geben Sie jeweils die passende SQL-Abfrage an:

- Welche Filme aus dem Jahr 2012 sind Action-Filme?
- Welche Filme haben eine Länge zwischen 90 und 120 Minuten?
- Wie viele Filme wurden im Jahr 2013 produziert?
- In welchen Filmen wirkt der Schauspieler „Bruce Wally“ mit?
  Lassen Sie nur die Filmtitel geordnet nach dem Produktionsjahr ausgeben.

(7 VP)

B1.3 Für einen Baumarkt soll eine Datenbank entworfen werden, die sowohl den Artikelbestand als auch die Artikelverkäufe erfasst.

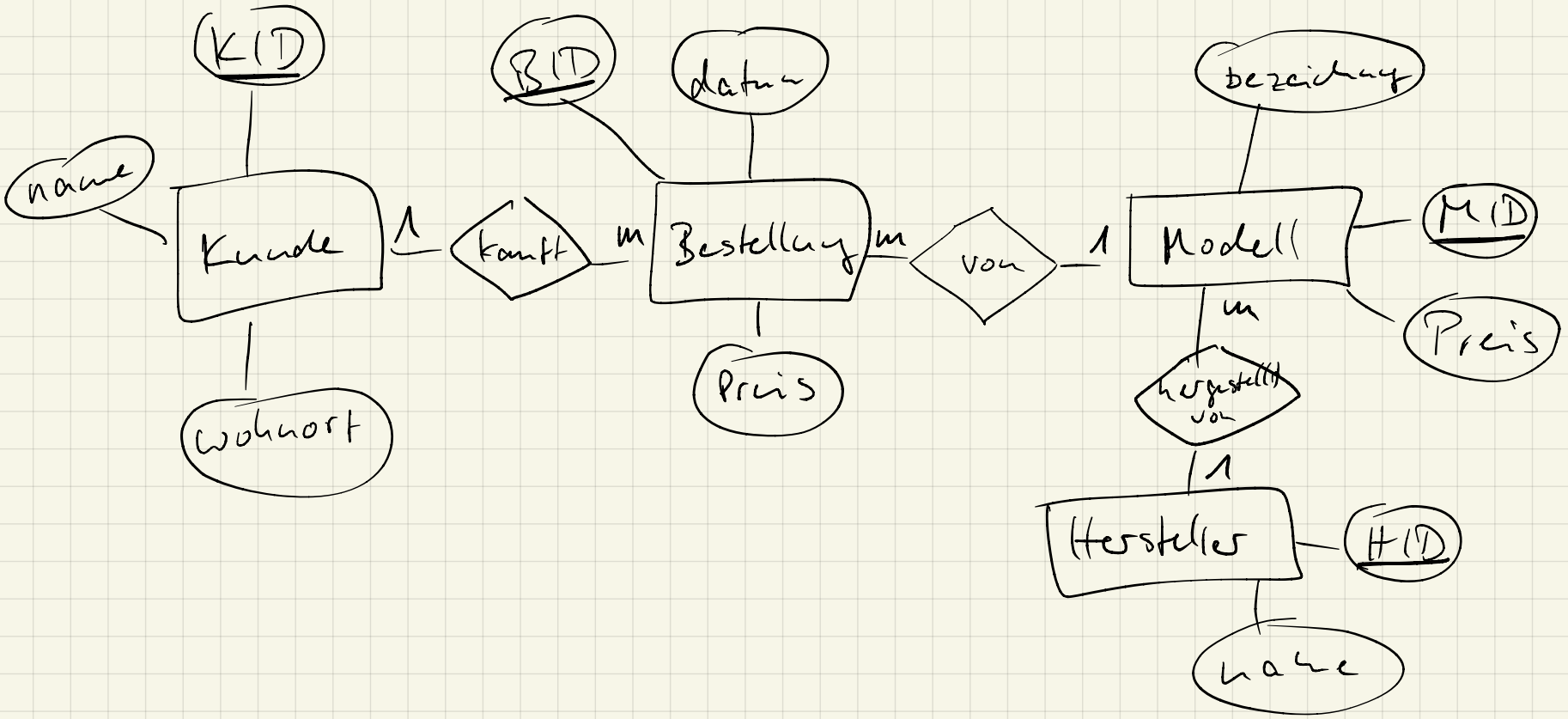
Folgende Anforderungen werden an die Datenbank gestellt:

– Zu jedem Artikel werden der Name, der Preis pro Einheit und wie viele Einheiten im Regal sind gespeichert.
– Viele Kunden haben mindestens eine Kundenkarte des Baumarkts. Für jeden Kunden sind die jeweilige Kartennummer sowie der Name und die Adresse des Kunden gespeichert.
– Kommt ein Kunde nach dem Einkauf an die Kasse, werden alle Artikel eingescannt und über ihren Barcode erfasst.
– Der Kunde erhält einen Kassenbon, auf dem alle Artikel und die Anzahl der erworbenen Einheiten aufgelistet sind. Diese Daten werden mit der eindeutigen Kassenbon-Nummer des Einkaufs gespeichert.
– Zu jedem Einkauf werden auch Datum und Uhrzeit erfasst.
– Bezahlt der Kunde mit EC-Karte oder bar, werden keine weiteren Daten gespeichert, bezahlt er mit seiner Kundenkarte, wird die Nummer der entsprechenden Karte gespeichert.

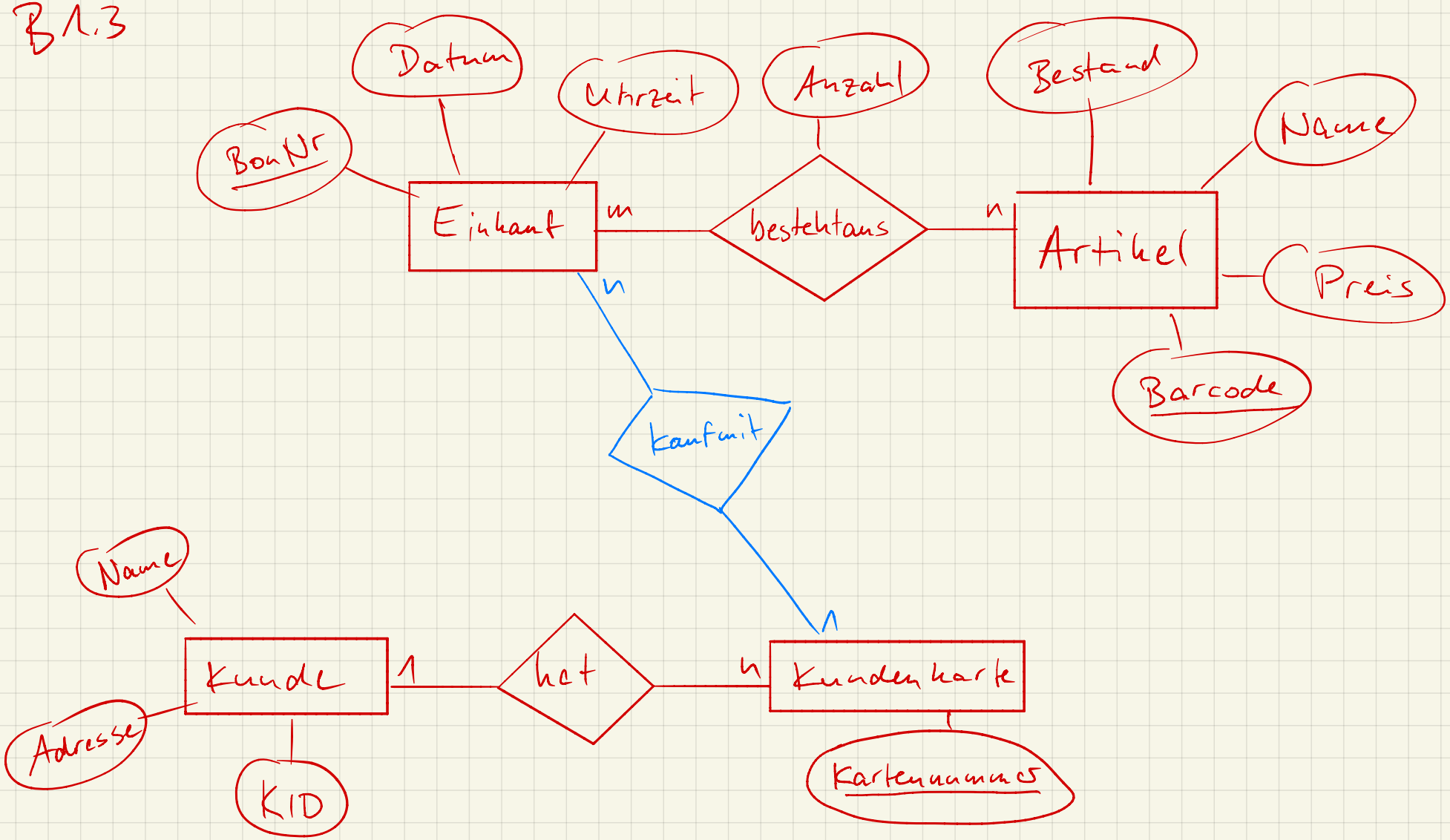
- Entwickeln Sie ein Entity-Relationship-Diagramm für die Datenbank!

- Kennzeichnen Sie die Schlüsselattribute und geben Sie die Beziehungskardinalitäten an!

(7 VP)


KID
name
Kunde
Wohnort
1
kauft
m
BID
datum
Bestellung
Preis
m
von
1
bezeichnung
Modell
MID
Preis
m
hergestellt von
1
Hersteller
HID
name

B1.3



Einkauf → kauft → Aaron

Einkauf → kauft → 1

Aaron → Kuka 1

Nico → Kuka 2

6.	03.02.20	04.02.20 fachpr. Abi Mu	05.02.20 fachpr. Abi Mu	06.02.20	07.02.20
7.	10.02.20	11.02.20	12.02.20	13.02.20	14.02.20
8.	17.02.20	18.02.20	19.02.20	20.02.20 Schmotziger	21.02.20 Bewegl. Ferientag
9.	24.02.20 Bewegl. Ferientag	25.02.20 Bewegl. Ferientag	26.02.20 Bewegl. Ferientag	27.02.20 Bewegl. Ferientag	28.02.20 Bewegl. Ferientag
10.	02.03.20	03.03.20	04.03.20 fachpr. Abi BK	05.03.20	06.03.20 D (1.-6.)
11.	09.03.20 ~~gk 1 / gk 3~~	10.03.20 rel / eth	11.03.20	12.03.20 E	13.03.20
12.	16.03.20 BK/GK/GEO/GNF/MU/REL/SP	17.03.20	18.03.20	19.03.20	20.03.20 F / bk 1 / gk 4
13.	23.03.20	24.03.20 CH 1 / PH / g 2 / ph 1	25.03.20 BIO 1 / BIO 2 / CH 2 / L	26.03.20 M	27.03.20
14.	30.03.20 gk 1+3 ~~bio 2 / ch 2 / mu 1~~	31.03.20	01.04.20 psy	02.04.20	03.04.20
Osterferien vom 06.04. bis einschließlich 17.04.2020					
17.	20.04.20 unterrichtsfrei	21.04.20 unterrichtsfrei	22.04.20 Abitur Neigungsf.	23.04.20 Abitur Italienisch	24.04.20 Abitur English
18.	27.04.20 Abitur Latein	28.04.20 Abitur Franz.	29.04.20	30.04.20 Abitur Deutsch	01.05.20 Feiertag
19.	04.05.20	05.05.20 Abitur Mathe	06.05.20	07.05.20 Musik-Exkursion	08.05.20 Musik-Exkursion
20.	11.05.20 bio 2 Sp ch 2 / mu 1	12.05.20	13.05.20	14.05.20 g 3 / bio 1	15.05.20 bk 2 / ch 1 / g 4 / ph 2
21.	18.05.20 sp	19.05.20 gk 1 / g 1 / mu 2	20.05.20 inf	21.05.20 Christi Himmelfahrt	22.05.20 Bewegl. Ferientag
22.	25.05.20 fachpr. Abi Sp	26.05.20 fachpr. Abi Sp	27.05.20	28.05.20 Kom.prüfung E	29.05.20 Kom.prüfung F
Pfingstferien vom 02.06. bis einschließlich 12.06.20					
25.	15.06.20	16.06.20	17.06.20	18.06.20	19.06.20
26.	22.06.20 unterrichtsfrei	23.06.20 unterrichtsfrei	24.06.20 unterrichtsfrei	25.06.20 unterrichtsfrei	26.06.20 unterrichtsfrei
27.	29.06.20 mdl. Abitur	30.06.20	01.07.20	02.07.20	03.07.20
28.	06.07.20	07.07.20	08.07.20	09.07.20	10.07.20
29.	13.07.20	14.07.20	15.07.20	16.07.20	17.07.20
30.	20.07.20	21.07.20	22.07.20	23.07.20	24.07.20
Sommerferien vom 30.07. bis einschließlich 11.09.20					

EINWURF

II B1 Datenbanken

B1.1 Das Verkaufsportal Paybay verkauft DVDs und speichert Datensätze in den Tabellen *Kunden*, *Filme* und *bestellt*:

Kunden:

KID	Name	Vorname	Land
3767	Epli	Horst	CH
4521	Schmidt	Antonia	DE
5488	Müller	Hans	DE

Filme:

FID	Film	Land	Jahr	Preis	Bestand
1000	Avatar	USA	2009	7,99	900
1001	Titanic	USA	1997	4,99	100
1002	Das Boot	DE	1982	6,99	200
1003	Das Wunder von Bern	DE	2003	6,99	100
1004	Monsieur Claude und seine Töchter	FR	2014	9,99	800
1005	Fack ju Göhte	DE	2013	11,99	1500

bestellt:

FID	KID	Zeitpunkt	Anzahl
1001	4521	2014-10-21 10:39:19	1
1001	3767	2014-10-19 14:02:01	2
1002	3767	2014-10-19 14:02:33	1

- Geben Sie eine SQL-Abfrage an, um alle Filme samt Verkaufspreis aufzulisten, die nach dem Jahr 2000 erschienen sind.
- Geben Sie die Ergebnistabelle der SQL-Abfrage
```
SELECT Land, COUNT(*) AS Anzahl
FROM Filme
GROUP BY Land
```
an.
- Geben Sie eine SQL-Abfrage an, die auflistet, welche Filme der Kunde mit der Nummer 3767 im Jahr 2014 bestellt hat.
- Der Manager von Paybay benötigt eine Liste mit dem Gesamtumsatz seines Unternehmens in jedem Land. Geben Sie hierfür eine SQL-Abfrage an.
- Welche Attribute der Tabelle b*estellt* bilden den Primärschlüssel? Begründen Sie Ihre Antwort.

(9 VP)

- SELECT Name, Preis FROM Filme WHERE Jahr > 2000

USA	2
DE	3
FR	1

- SELECT * FROM Filme JOIN bestellt ON (bestellt.FID = Filme.FID)
  WHERE KID = '3767' AND YEAR(Zeitpunkt) = '2014'

  Zeitpunkt LIKE '2014%'

-
```
SELECT Kunden.Land, SUM(Preis * Anzahl)
    FROM Kunden
    JOIN bestellt ON (bestellt.KID = Kunden.KID)
    JOIN Filme ON (Filme.FID = bestellt.FID)
    GROUP BY Kunden.Land
```

B1.2 Ein Wahlforschungsinstitut möchte die Ergebnisse, die politische Parteien bei Landtagswahlen erzielt haben, in einer relationalen Datenbank speichern.

– Für jedes Land ist der Name, die Einwohnerzahl, die Anzahl der Sitze im Landesparlament und der Regierungssitz zu speichern. Weiterhin nehmen wir an, dass die Länder durch ihre Namen eindeutig identifiziert sind.

– Jede Partei hat einen voll ausgeschriebenen und einen abgekürzten Namen.

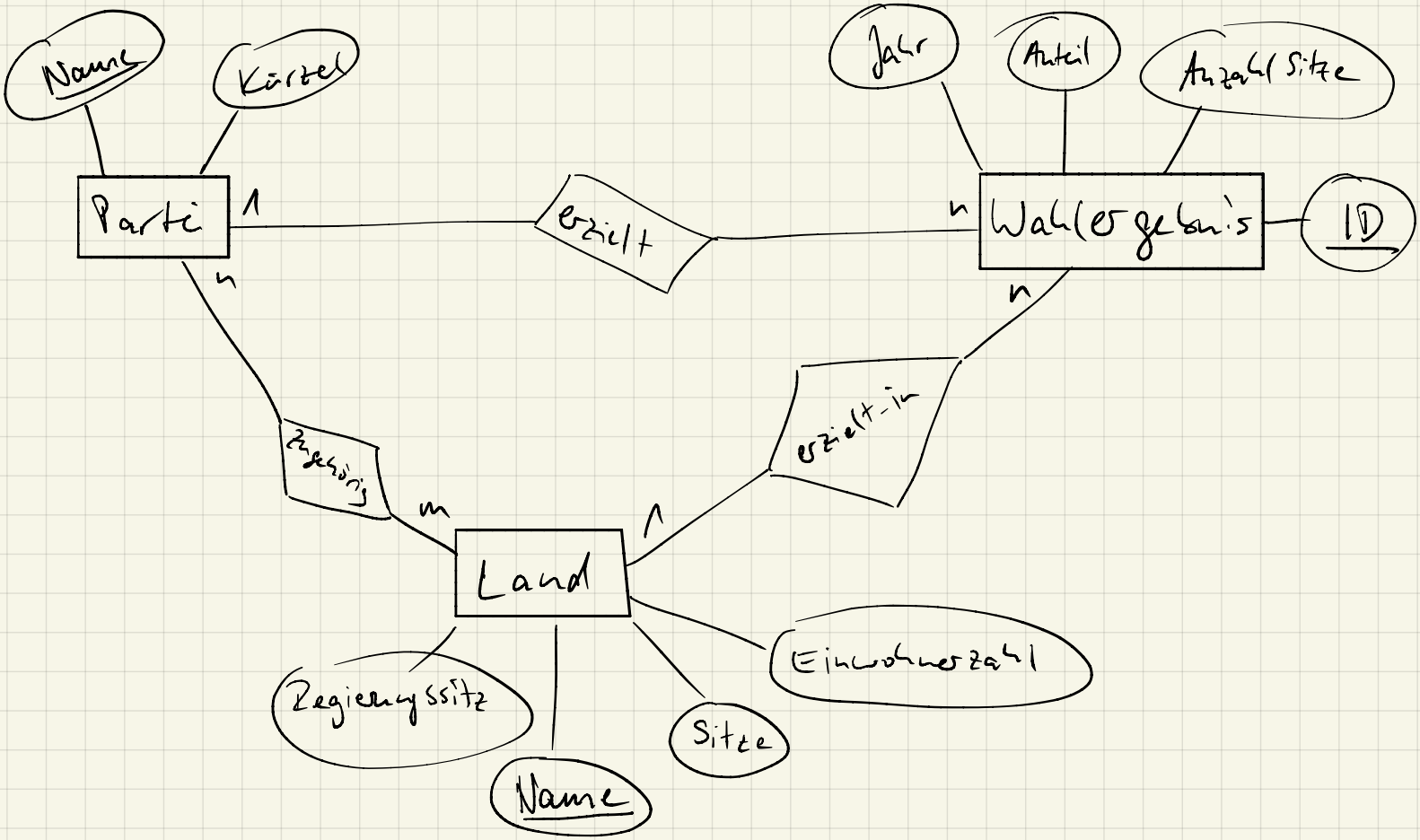
– Folgende bei Landtagswahlen erzielten Ergebnisse werden gespeichert:
- das Jahr, in dem die Wahl stattfand,
- der Stimmenanteil je Partei,
- die Anzahl der Sitze, welche eine Partei erringen konnte.

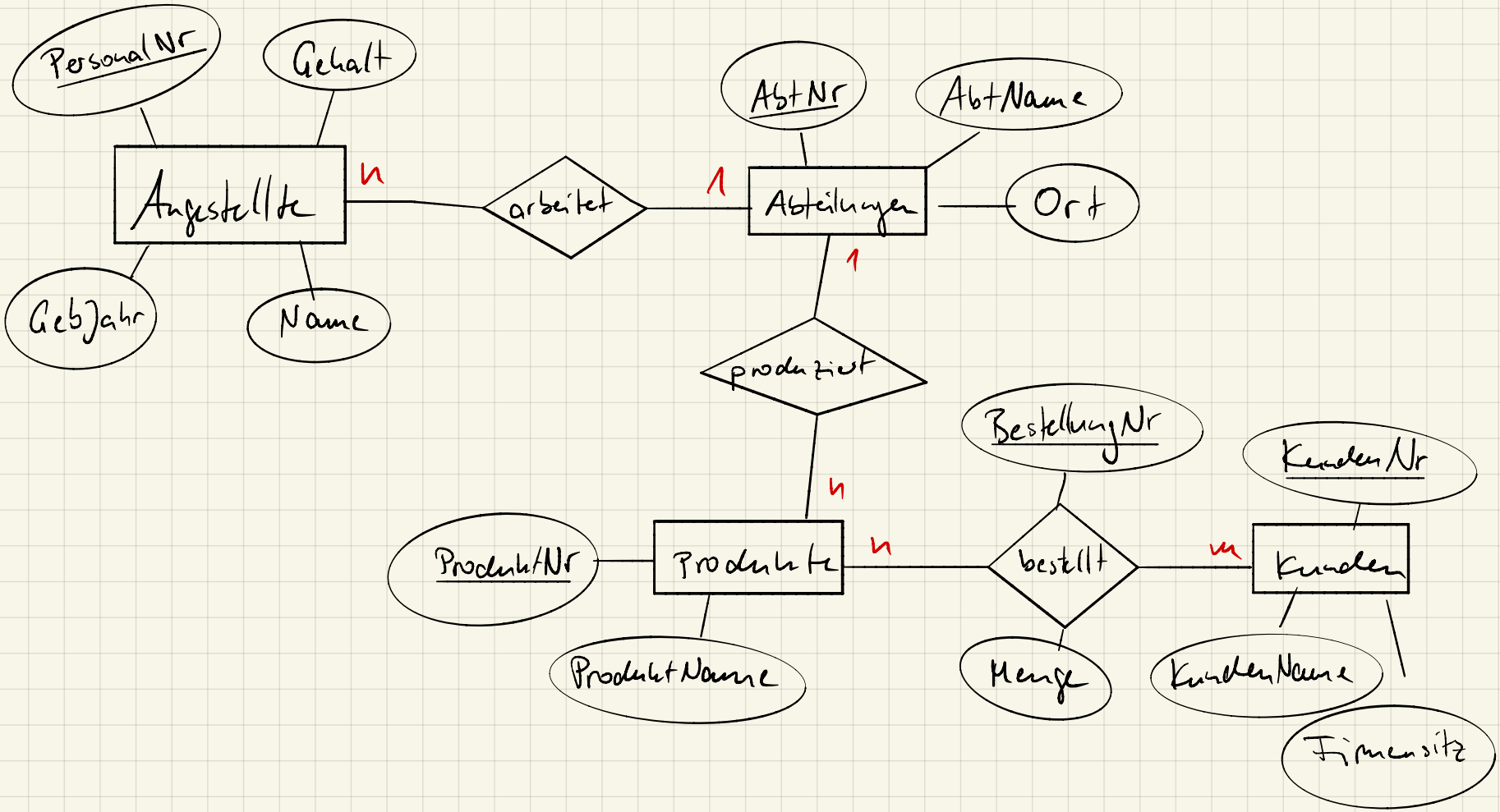
- Entwerfen Sie ein geeignetes ER-Modell mit passenden Relationen, das die Attribute, die Schlüssel und die Kardinalitäten enthält.
- Welche Zugriffsrechte benötigt ein Mitarbeiter des Instituts, um die Daten nach der nächsten Wahl erfassen zu können?

Aus Sicherheitsgründen muss man sich an der Datenbank anmelden, um den Datensatz zu bearbeiten. Das Kennwort darf nur die Kleinbuchstaben a..z enthalten und wird nach Vigenère verschlüsselt auf dem Server abgelegt: Der Schlüssel lautet „abi“.

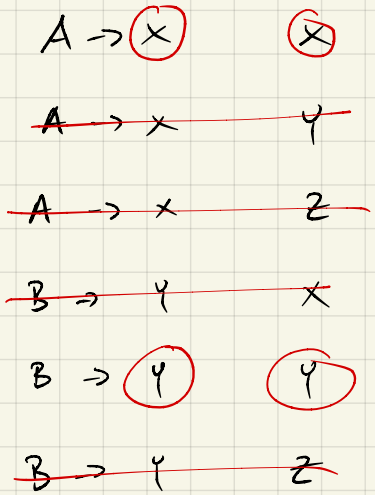
- Verschlüsseln Sie das Kennwort „wahlurne“ des Mitarbeiters. wbpLvznf
  abiabiab
- Erklären Sie, wie man einen mit der Vigenère-Chiffre verschlüsselten Text bei bekannter Schlüssellänge vollständig entschlüsseln kann.

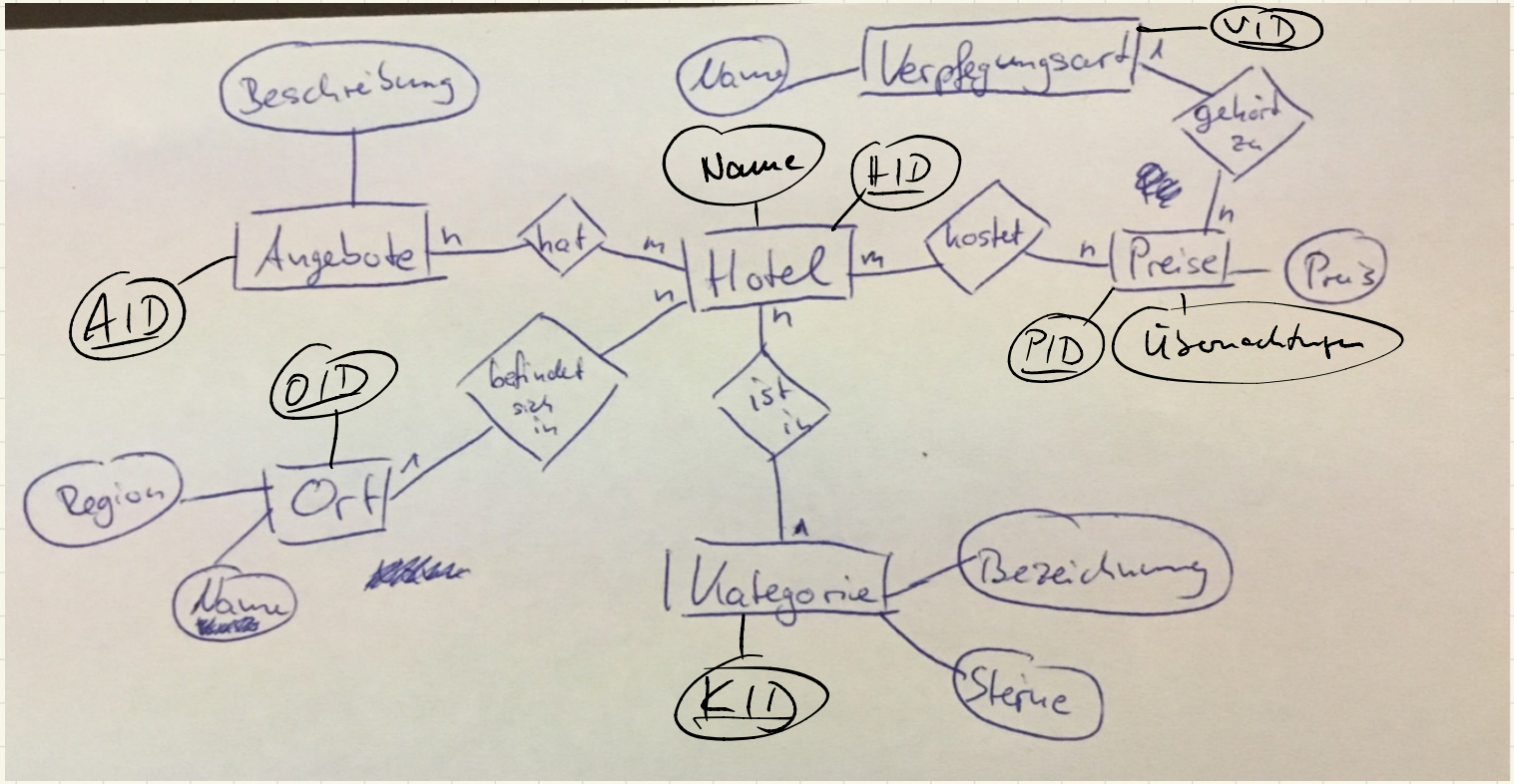
(11 VP)

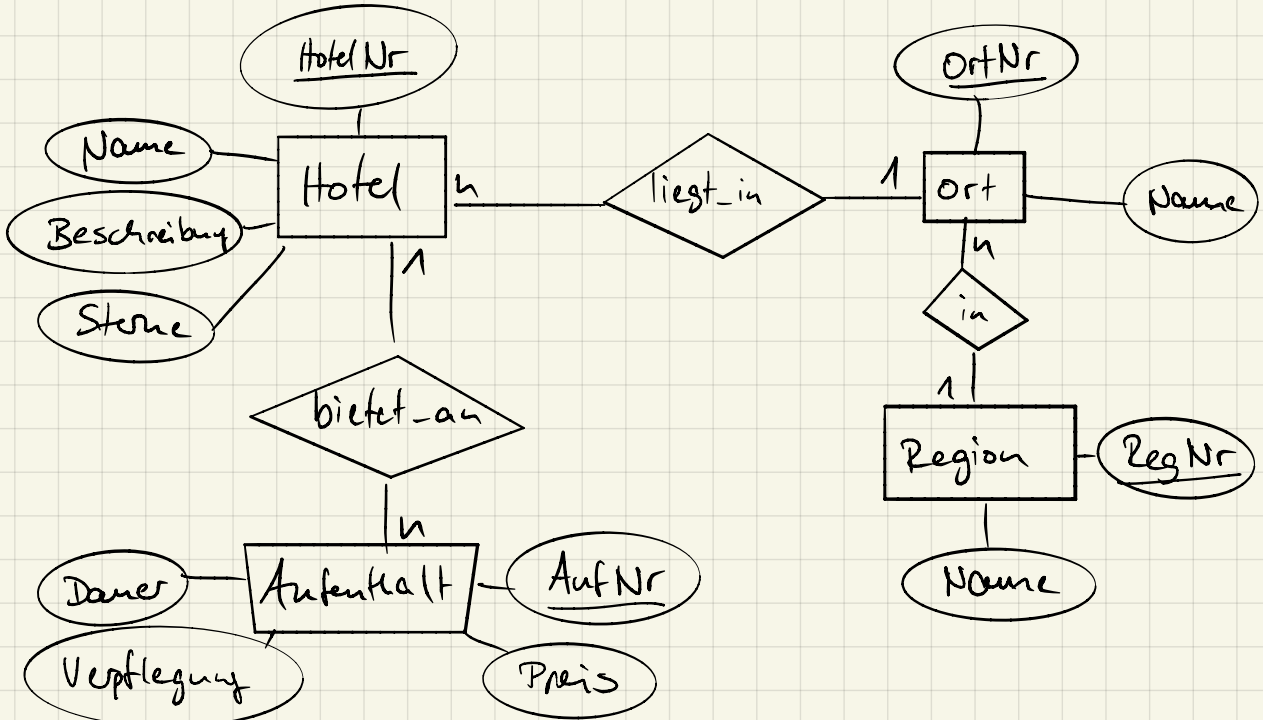

Name
Kürzel
Partei
1
n
erzielt
n
Wahlergebnis
ID
Jahr
Anteil
Anzahl Sitze
n
zugeordnet
m
Land
1
erzielt_in
Regierungssitz
Name
Sitze
Einwohnerzahl


PersonalNr
Gehalt
Angestellte
n
arbeitet
1
AbtNr
AbtName
Abteilungen
Ort
Gebjahr
Name
1
produziert
BestellungNr
KundenNr
n
ProduktNr
Produkte
n
bestellt
m
Kunden
ProduktName
Menge
KundenName
Firmensitz

Prod	Act
A → X	X
B → Y	Y
	Z




Beschreibung
Name
Verpflegungsart
VID
gehört zu
Name
HID
Angebote
n
hat
m
n
Hotel
m
kostet
n
Preise
Preis
AID
PID
Übernachtungen
OID
befindet sich in
n
ist in
Region
Ort
1
Name
Kategorie
Bezeichnung
KID
Sterne


HotelNr
OrtNr
Name
Hotel
n
liegt_in
1
Ort
Name
Beschreibung
n
in
Sterne
1
bietet_an
1
Region
RegNr
n
Dauer
Aufenthalt
AufNr
Name
Verpflegung
Preis

CD_ID	Album	Gründungsjahr	Erscheinungsjahr	Titelliste
4711	Anastacia – Not That Kind	1999	2000	{1. Not That Kind, 2. I'm Outta Love, 3. Cowboys & Kisses}
4712	Pink Floyd – Wish You Were Here	1965	1975	{1. Shine On You Crazy Diamond}
4713	Anastacia – Freak of Nature	1999	2001	{1. Paid my Dues}

Einträge in Tabellen sollten „atomar" sein

→ nur eine einzelne Information in einer Spalte

CD_ID	Albumtitel	Interpret	Gründungsjahr	Erscheinungsjahr	*Track*	Titel
4711	Not That Kind	Anastacia	1999	2000	1	Not That Kind
4711	Not That Kind	Anastacia	1999	2000	2	I'm Outta Love
4711	Not That Kind	Anastacia	1999	2000	3	Cowboys & Kisses
4712	Wish You Were Here	Pink Floyd	1965	1975	1	Shine On You Crazy Diamond
4713	Freak of Nature	Anastacia	1999	2001	1	Paid my Dues

Einträge sind abhängig von ID → Inkonsistenz möglich

→ separate Tabelle

CD

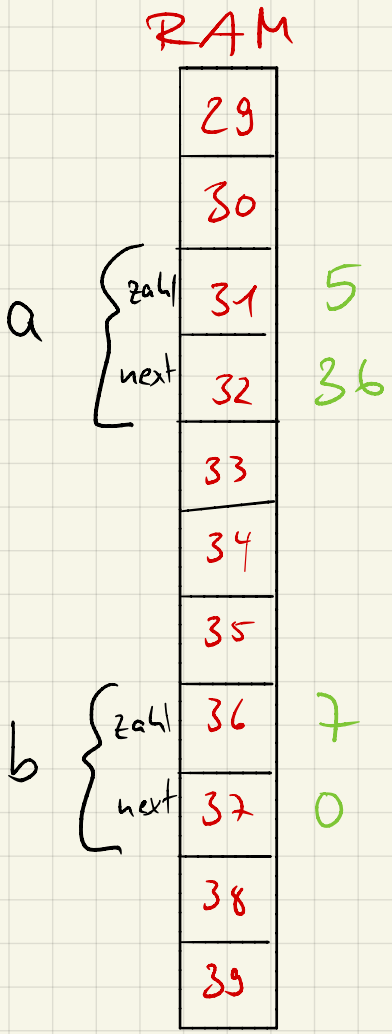
CD_ID	Albumtitel	Interpret	Gründungsjahr	Erscheinungsjahr
4711	Not That Kind	Anastacia	1999	2000
4712	Wish You Were Here	Pink Floyd	1965	1975
4713	Freak of Nature	Anastacia	1999	2001

Lied

CD_ID	*Track*	Titel
4711	1	Not That Kind
4711	2	I'm Outta Love
4711	3	Cowboys & Kisses
4712	1	Shine On You Crazy Diamond
4713	1	Paid my Dues

Noch 9!

19.2.20

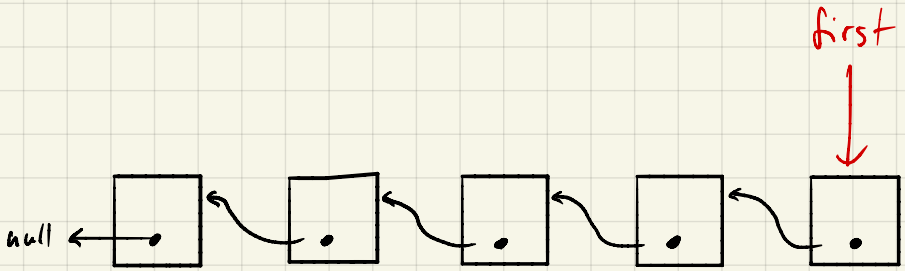



```
class Node {
    public int zahl;
    public Node next;
}
```

```
Node 31 a = new Node();
a.zahl = 5;
Node 36 b = new Node();
b.zahl = 7;
a.next = b;
```

LIFO → Stack / Stapel / Keller

1. Vorne anhängen, vorne wegnehmen
2. hinten anhängen, hinten wegnehmen

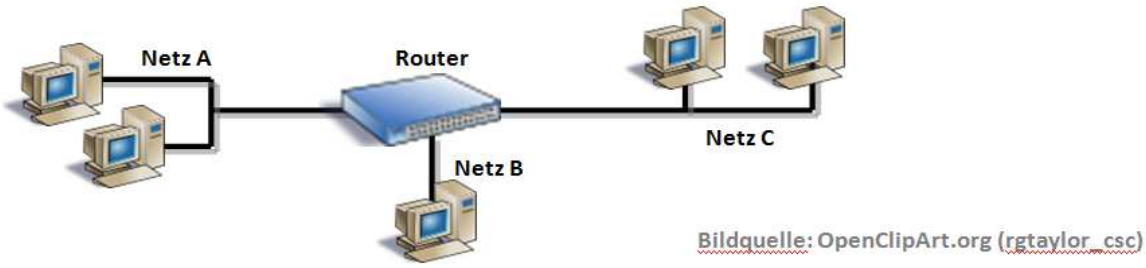


FIFO → Queue / Schlange

1. Hinten anhängen, vorne wegnehmen
2. Vorne anhängen, hinten wegnehmen

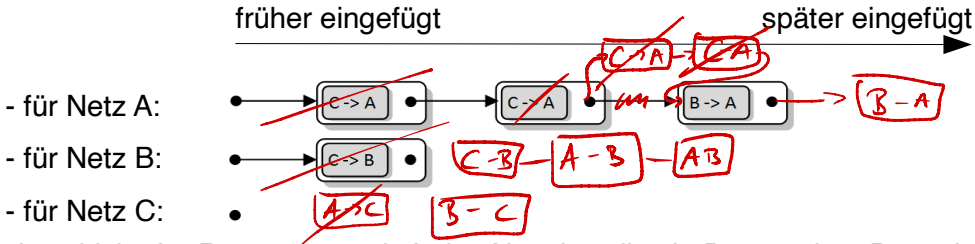
B3 Abstrakte Datentypen

Durch den zunehmenden Datenverkehr im Internet gibt es eine Diskussion wie mit Leitungsengpässen umzugehen ist. Zur Zeit gilt die Netzneutralität, d.h. alle Datenpakete - egal von welchem Absender an welchen Empfänger sie gehen und egal welchen Inhalt sie haben - werden gleich behandelt. Bei einer ausgelasteten Leitung werden ankommende Pakete vom Router zwischengespeichert und dann in der Reihenfolge ihres Eintreffens weitergeleitet. Die Betreiber von Telekommunikationsnetzwerken möchten diese Netzneutralität gerne abschaffen und die Daten in Prioritätsklassen einteilen. Pakete mit höherer Priorität würden dann bevorzugt weitergeleitet.



Zu Beginn sind schon folgende Pakete zwischengespeichert
(B -> A : stellt ein Paket von Netz B nach Netz A dar):

Zwischenspeicher am Anfang



Pro Zeittakt schickt der Router zuerst in jedes Netz jeweils ein Datenpaket. Danach empfängt er gegebenenfalls aus jedem Netz ein Paket (in der Reihenfolge Netz A, Netz B und zuletzt Netz C).

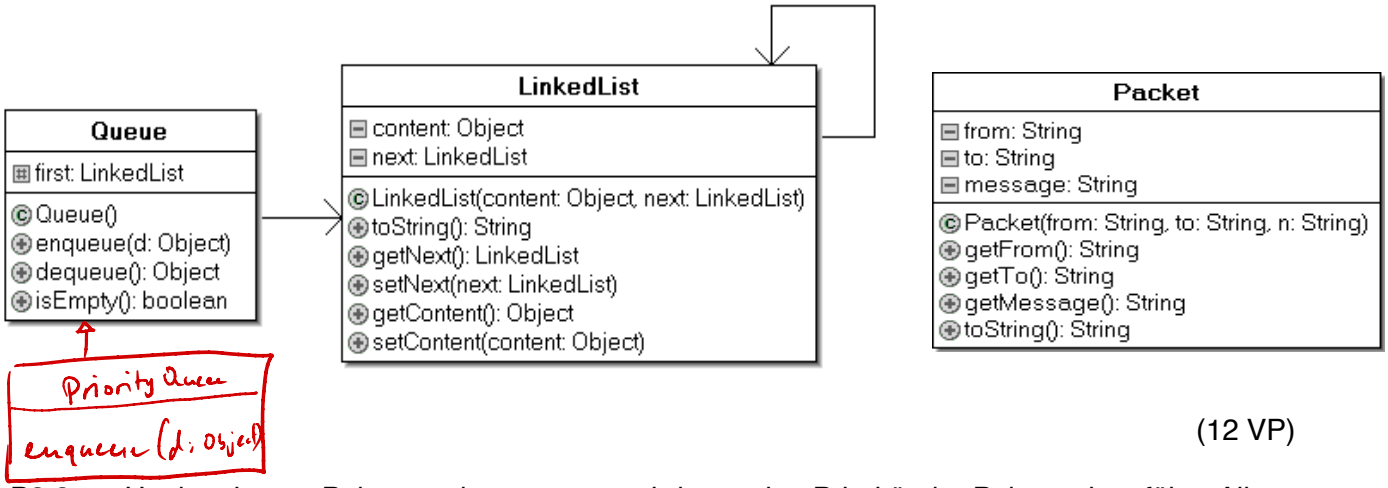
Folgende Pakete erreichen den Router:

Takt 1: Paket von B->A, Paket von C->A, Paket von A->C
Takt 2: Paket von C->A
Takt 3: Paket von A->B, Paket von C->B
Takt 4: Paket von A->B, Paket von C->B, Paket von B->C

B3.1 Das Steuerungsprogramm eines die Netzneutralität wahrenden Routers könnte den ADT Schlange (Queue) zur Verwaltung der Pakete verwenden. Für jedes angeschlossene Netz gibt es dann eine Schlange, die die noch abzuarbeitenden Datenpakete speichert.

- Beschreiben Sie die grundlegenden Eigenschaften einer Schlange und erläutern Sie, warum die Wahl des ADT Schlange für diese Anwendung sinnvoll ist.
- Geben Sie an, welche Operationen der Warteschlange für Netz A ausgeführt werden müssen, um
  - das nächste zu übermittelnde Paket von C->A aus der Warteschlange zu holen,
  - das in Takt 1 ankommende Paket aus Netz B für Netz A zu speichern.
- Stellen Sie die Schlangen nach dem 4. Takt dar, wenn der Router die Netzneutralität wahrt.

- Erläutern Sie die Bedeutung der Beziehungen zwischen den unten dargestellten Klassen. Gehen Sie insbesondere auf die Bedeutung der Beziehung einer Klasse mit sich selbst ein.
- Implementieren Sie die Methode enqueue(Object d) der Klasse Queue auf der Basis des unten dargestellten Klassendiagramms, so dass neue Pakete an das Ende der verketteten Liste angehängt werden. Sie können davon ausgehen, dass alle anderen Methoden schon korrekt implementiert sind.



(12 VP)

B3.2 Um bestimmte Pakete zu bevorzugen, wird nun eine Priorität der Pakete eingeführt. Alle Pakete, die aus Netz C kommen, sollen mit höherer Priorität weitergeleitet werden. Um dies zu realisieren, wird der Datentyp PriorityQueue verwendet, der sich nur in der Methode enqueue von der normalen Schlange unterscheidet: Die Pakete mit der höheren Priorität werden vor den Paketen mit niedrigerer Priorität in der Schlange einsortiert.

- Stellen Sie die Schlangen nach dem 4. Takt dar, wenn der Router diese Priorität beachtet.
- Ergänzen Sie das Klassendiagramm um die neue Klasse PriorityQueue.
  Dabei sollen die nicht veränderten Methoden von der Klasse Queue übernommen werden. Erläutern Sie die dabei neu hinzugekommene Beziehung.

(4 VP)

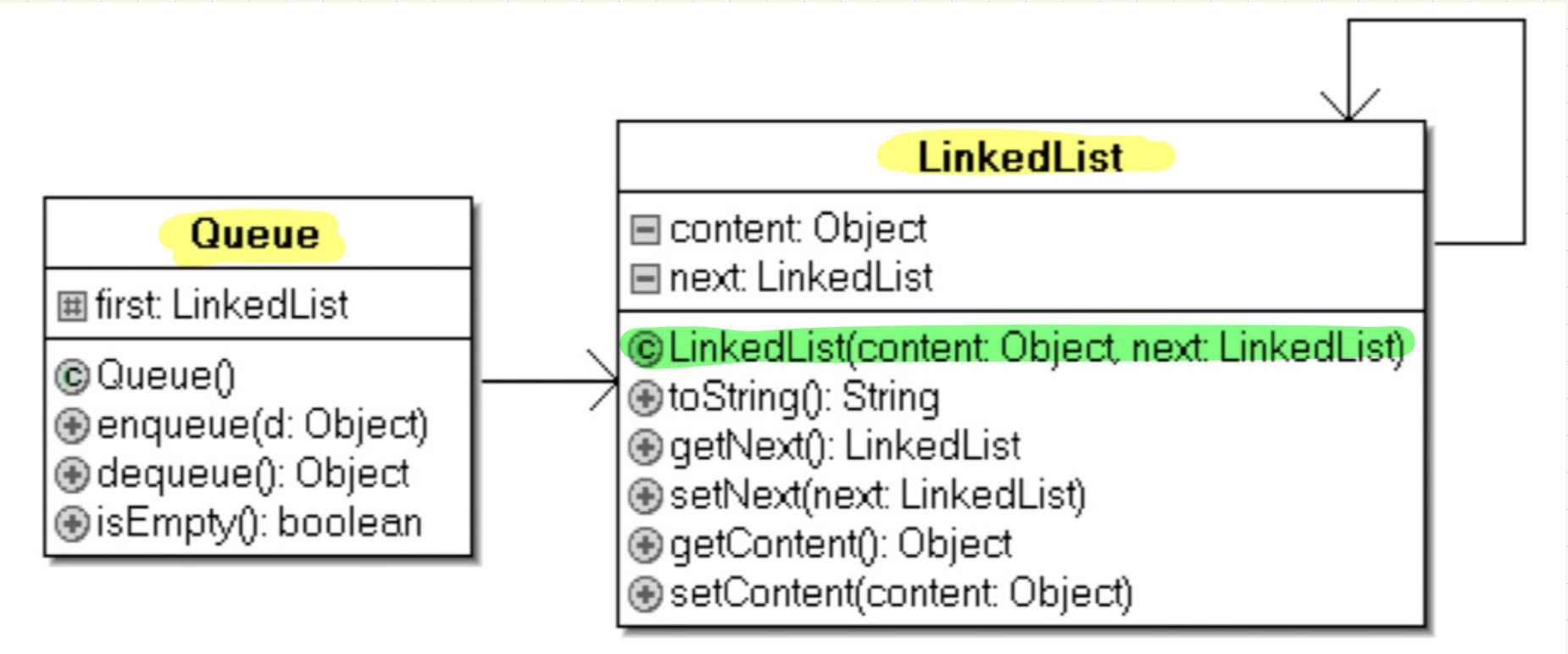
B3.3 Es gibt Möglichkeiten eine PriorityQueue zu implementieren, so dass die Methoden enqueue und dequeue in einer durchschnittlichen Laufzeit proportional zu log(n) ausgeführt werden (z.B. mittels Heap). Dabei ist n die Anzahl der in der PriorityQueue gespeicherten Pakete.

- Analysieren Sie das Laufzeitverhalten von enqueue und dequeue bei einer PriorityQueue auf der Basis einer LinkedList. Beurteilen Sie, ob es bei einem Router im Hinblick auf die Laufzeit sinnvoller ist, einen Heap oder eine LinkedList als Datenstruktur für eine PriorityQueue einzusetzen.

(4 VP)

Linked List:
enqueue → n Schritte
dequeue → 1 Schritt
Ø $\frac{n}{2}$

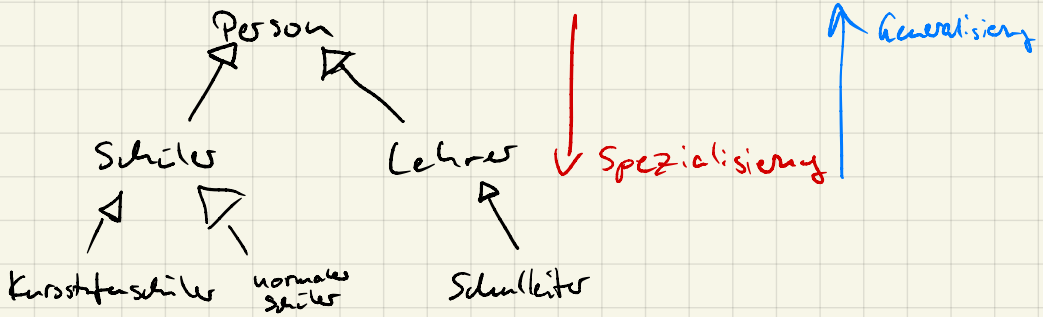
Heap
enqueue → log n
dequeue → log n
Ø log n




```
public void enqueue(Object d) {
    LinkedList new = new LinkedList(d, null);
    if(this.first == null) {
        this.first = new;
    } else{
        LinkedList temp = this.first;
        while (temp.getNext() != null)
            temp = temp.getNext();
        temp.setNext(new);
    }
}
```

Vererbung: „genauere" Klasse, spezifischer

„ist-eine-Art" Beispiel

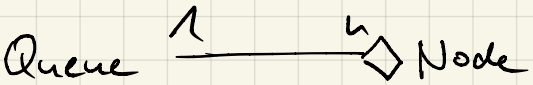


Assoziation:

„hat-Beziehung"

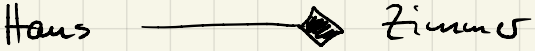
Aggregation:

„hat-Beziehung"

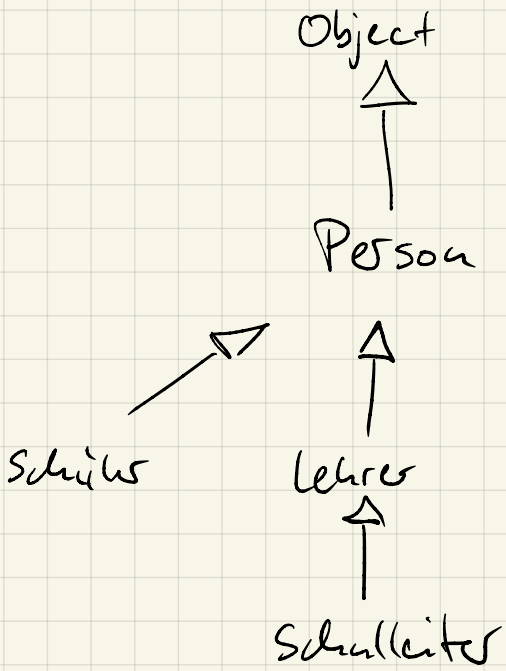


Komposition

„hat-Beziehung"



Teil kann nicht alleine existieren




```
Schüler  a = new Schüler(...)
Lehrer   b = new Lehrer(...)

Person   c = new Schüler(...)

Object   d = new Schüler(...)
```

```
private Fach[] faecher;

public Lernbox() {
    faecher = new Fach[5];
    for (int i=0; i<5; i++) {
        faecher[i] = new Fach();
    }
}
```

Schüler

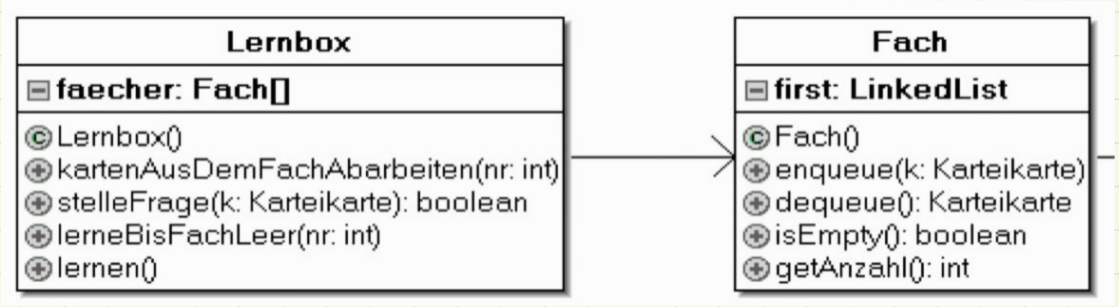
vorname
nachname

Schüler (vorname, nachname)

```
Schüler a = new Schüler("Aaron", "Sommer");

Schüler[] s = new Schüler[5];

s[0] = new Schüler("Patrick", "Mayer");
```



Die Methode `stelleFrage(k: Karteikarte): boolean` bewirkt, dass dem Nutzer der App die Frage angezeigt und seine Antwort kontrolliert wird. Wurde die Frage korrekt beantwortet, wird `true` zurückgegeben, sonst `false`.

- Implementieren Sie unter Verwendung der Methode `stelleFrage(k: Karteikarte): boolean` die Methode `kartenAusDemFachAbarbeiten(nr: int)`, die alle Karten des angegebenen Fachs durchgeht, bis das Fach geleert ist. Richtig beantwortete Fragen werden ins nächste, falsch beantwortete Fragen ins erste Fach befördert.
  Hinweis: Eine Fehlerbehandlung für unzulässige Werte des Parameters `nr` kann entfallen.

```
public void kartenAusDemFachAbarbeiten(int nr){
    Karteikarte aktuell = faecher[nr].dequeue();
    while (aktuell != null) {
        boolean x = stelleFrage(aktuell);
        if(x) {
            faecher[nr+1].enqueue(aktuell);
        }else {
            faecher[0].enqueue(aktuell);
        }
        aktuell = faecher[nr].dequeue();
    }
}
```

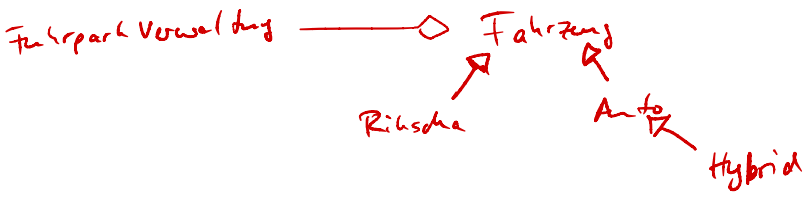
A Objektorientierte Modellierung und Programmierung

Ein Taxiunternehmen unterhält einen Fuhrpark mit verschiedenen Fahrzeugen zur Personenbeförderung, u.a. mit Autos und Fahrradrikschas (kurz: Rikschas). Das Unternehmen beauftragt Sie mit der Erstellung einer Software zur Verwaltung der Fahrzeugdaten und des täglichen Fahrgeschäfts.
Mit dem zu erstellenden Programm sollen eine ganze Reihe von Fahrzeugen verwaltet werden. Mit den Fahrzeugdaten werden -sofern vorhanden - auch deren Verbrauch, die gefahrenen Wegstrecken und die Einnahmen, die mit dem jeweiligen Fahrzeug erwirtschaftet werden, erfasst. Jedes Fahrzeug wird eindeutig über sein Kennzeichen (z.B: S-AB-123) identifiziert. Bei der Neuaufnahme der Fahrzeugdaten mit der Software werden neben dem Kennzeichen auch die Anzahl der Sitzplätze, der Grundpreis und der Kilometerpreis eingegeben.

Rikschas werden mit Muskelkraft angetrieben. Sie haben drei Sitzplätze, der Grundpreis beträgt 2,70 €, der Kilometerpreis 0,50 €.
Autos werden mit Benzin betrieben. Sie haben je nach Typ vier bis neun Sitzplätze. Um den Benzinverbrauch der Autos kontrollieren zu können, sollen bei der Neuerfassung der Autodaten zusätzlich das Volumen des Benzintanks und der durchschnittliche Benzinverbrauch auf 100 km erfasst werden. Der Grundpreis für ein Auto beträgt einheitlich 5,80 €, der Kilometerpreis 1,60 €. Bei allen Fahrzeugen sind die ersten drei gefahrenen Kilometer im Grundpreis enthalten.
Für den Stadtverkehr hat das Taxiunternehmen Hybrid-Fahrzeuge (Autos mit einem gewöhnlichen Verbrennungsmotor, der von einem Elektromotor unterstützt wird) angeschafft. Der Strom für den Elektromotor wird einer Batterie entnommen, die mit Hilfe des Verbrennungsmotors geladen wird. Bei Betriebsbeginn übernehmen die Fahrer die Fahrzeuge, Autos sind dann vollgetankt. Nach jeder Fahrt wird in Abhängigkeit von der zurückgelegten Wegstrecke für den Kunden der Fahrpreis berechnet. Für jedes Fahrzeug werden dabei der Kilometerstand, die Tagesfahrstrecke, die Tageseinnahmen und bei Autos der Tankinhalt aktualisiert.

A1
- Stellen Sie die Beziehungen zwischen den Klassen `FuhrparkVerwaltung`, `Fahrzeug`, `Rikscha`, `Auto` und `HybridFahrzeug` in einem Klassendiagramm dar. Die Klassen sollen nur den Klassennamen, keine Attribute oder Methoden enthalten.
- Entwerfen Sie ein UML-Klassendiagramm für die Klasse `Fahrzeug` unter Berücksichtigung des Geheimnisprinzips. Die Sichtbarkeit von allen Attributen, Konstruktoren und Methoden, sowie Parameterlisten und eventuelle Rückgabewerte sollen mit angegeben werden.
- Implementieren Sie für die Klasse `Fahrzeug` einen geeigneten Konstruktor.
- Implementieren Sie eine Methode `berechneFahrpreis(...)`, die in Abhängigkeit von der gefahrenen Strecke den Fahrpreis zurückgibt und zusätzlich den Kilometerstand, die Tagesfahrstrecke und die Tageseinnahmen aktualisiert.

(7 VP)



Fahrzeug

- kennzeichen: String - sitzplätze: int - grundpreis: double - kmpreis: double - kmstand: double - tagesstrecke: double - tageseinnahmen: double
c Fahrzeug(kennzeichen: String, sitzplätze: int, grundpreis: double, kmpreis: double) + getKennzeichen(): String + getGrundpreis(): double + getKmPreis(): double + getKmstand(): double + getTagesstrecke(): double + getTageseinnahmen(): double + berechneFahrpreis(strecke: double): double

```
public Fahrzeug(String kennzeichen, int sitzplätze, double grundpreis, double kmpreis){
    this.kennzeichen = kennzeichen;
    this.sitzplätze = sitzplätze;
    this.grundpreis = grundpreis;
    this.kmpreis = kmpreis;

    this.kmstand = 0;
    this.tagesstrecke = 0;
    this.tageseinnahmen = 0;

}
```

```
public Rikscha(String kennzeichen){
    super(kennzeichen, 3, 2.7, 0.5)
}
```

```
public double berechneFahrpreis (double strecke) {
    this.kmstand = this.kmstand + strecke;
    this.tagesstrecke += strecke;

    if(strecke < 3) {this.tageseinnahmen += this.grundpreis;}
    else{ this.tageseinnahmen += this.grundpreis + this.kmpreis * (strecke-3); }

    if (strecke <3) { return this.grundpreis; }
    return this.grundpreis + this.kmpreis * (strecke-3);
}
```

```
double preis = this.grundpreis;
if (strecke >3) { preis += this.kmpreis * (strecke-3); }
this.tageseinnahmen += preis;
return preis
```

A2
- Implementieren Sie je einen Konstruktor für die Klassen `Rikscha` und `Auto`.
- Für die Autos muss die Methode `berechneFahrpreis(...)` aus der Klasse Fahrzeug überschrieben werden. Bei Eingabe der zurückgelegten Wegstrecke soll anhand des Durchschnittsverbrauchs der Tankinhalt aktualisiert werden.
  Implementieren Sie eine neue Methode `berechneFahrpreis(...)` für die Klasse `Auto`.
- Implementieren Sie für die Klasse `Auto` die Methode `volltanken()`, die in Abhängigkeit vom aktuellen Tankinhalt und vom Tankvolumen die getankte Benzinmenge in Litern zurückgibt.

(6 VP)

Bei Betriebsende werden per Software die Tagesdaten der Fahrzeuge ausgelesen, ausgewertet und auf Tages-Startwerte zurückgesetzt.

A3
- Geben Sie die Deklaration einer geeigneten Datenstruktur für die Liste der Fahrzeuge in der Klasse `FuhrparkVerwaltung` an.
- Implementieren Sie eine Methode, die es erlaubt, die gesamte gefahrene Strecke aller Fahrzeuge des Fuhrparks zu ermitteln.
- Beschreiben Sie, welche Programmerweiterungen Sie vornehmen müssen, damit Ihr Programm die Gesamteinnahmen sowie die gesamte getankte Benzinmenge am Betriebsende ermitteln kann.
- Die Liste der Fahrzeuge sei vorsortiert. Implementieren Sie eine Methode, die es ermöglicht, ein neues Fahrzeug anhand seines Kennzeichens an der alphabetisch korrekten Stelle einzufügen.

(7 VP)

Hinweise:

Die Implementierung von Programmen oder Programmteilen erfolgt jeweils in der in Ihrem Kurs eingeführten Programmiersprache.

Bei der Programmierung ist davon auszugehen, dass sämtliche erforderlichen Daten in einem vernünftigen Rahmen eingegeben werden. An eine Fehlerbehandlung von Falscheingaben ist nicht gedacht.