

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer
Kennzahl: _____	<b>Frühjahr</b>	
Kennwort: _____	<b>2022</b>	<b>66116</b>
Arbeitsplatz-Nr.: _____		

---

**Erste Staatsprüfung für ein Lehramt an öffentlichen Schulen**  
— Prüfungsaufgaben —

---

Fach: **Informatik (Unterrichtsfach)**

Einzelprüfung: **Softwaretechnologie/Datenbanksysteme**

Anzahl der gestellten Themen (Aufgaben): **2**

Anzahl der Druckseiten dieser Vorlage: **33**

---

Bitte wenden!

Thema Nr. 1  
(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!  
Alle Lösungsschritte sind sorgfältig zu begründen!

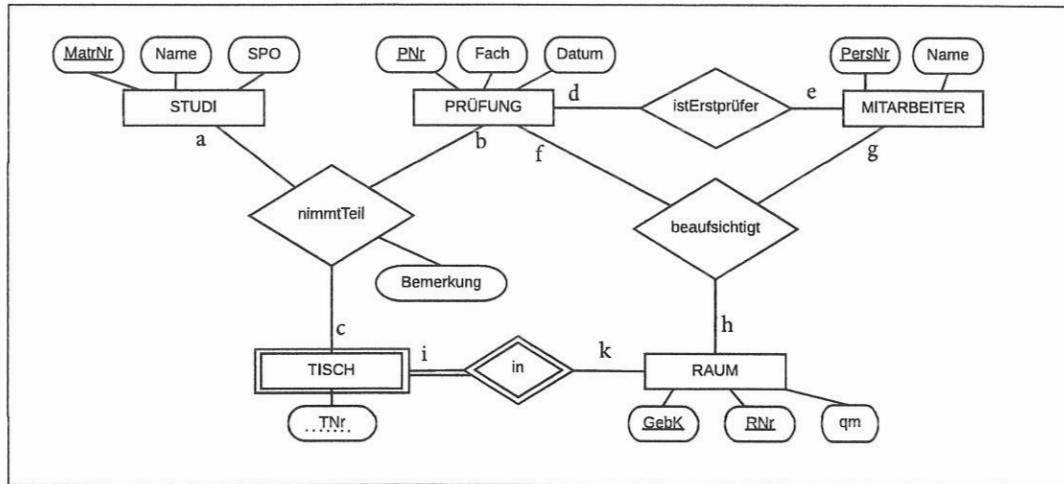
**Teilaufgabe I: Datenbanksysteme**

**Aufgabe 1**

[22 PUNKTE]

Die Planung von Prüfungen soll elektronisch unterstützt werden. Wir gehen davon aus, dass wir nur für einen Prüfungszeitraum planen und ignorieren Wiederholungsklausuren.

- Alle STUDI sind eindeutig durch ihre Matrikelnummer (*MatrNr*) identifiziert. Zu allen STUDI ist der *Name* bekannt, sowie die geltende Studienprüfungsordnung (*SPO*).
  - Jede PRÜFUNG ist eindeutig durch eine Prüfungsnummer (*PrNr*) identifiziert. In einer PRÜFUNG geht es um ein konkretes *Fach* und sie findet an einem bestimmten *Datum* statt.
  - Zu den MITARBEITERn sind die Personalnummer (*PersNr*) und der *Name* bekannt.
  - Manche MITARBEITER sind *Erstprüfer* für eine oder mehrere Prüfungen. Zu jeder Prüfung ist immer genau ein *Erstprüfer* bekannt.
  - Prüfungen finden in Räumen statt. Jeder RAUM ist durch das Gebäudekürzel (*GebK*) und die Raumnummer (*RNr*) eindeutig identifiziert. Zu jedem Raum ist die Größe (in *qm*) bekannt.
  - Ein RAUM, in dem eine Prüfung stattfindet, ist mit Prüfungstischen bestückt. Jeder TISCH ist dabei nummeriert (mit einer *TNr*). Die Nummerierung beginnt in jedem Raum mit 1.
  - Wenn ein STUDI an einer PRÜFUNG *teilnimmt*, wird erfasst, an welchem TISCH er oder sie sitzt. Es gibt nur Einzeltische. Besondere Vorfälle während der Prüfung werden von der Prüfungsaufsicht als *Bemerkung* festgehalten.
  - Manche Mitarbeiter helfen bei der Prüfungsaufsicht mit. Ein Mitarbeiter kann bei mehreren Prüfungen als Aufsicht eingeteilt sein. Mitarbeiter sind während einer Prüfung immer nur für einen Raum zuständig. In großen Räumen werden mehrere Mitarbeiter zur Aufsicht eingesetzt.
- (a) Charakterisieren Sie die Beziehungen in folgendem ER-Diagramm mit Kardinalitäten in Chen-Notation (= Funktionalitäten) entsprechend der Beschreibung: Ordnen Sie dazu den Variablen (a) bis (h) aus dem ER-Diagramm auf Ihrem Lösungsblatt Werte zu, wie z. B. (i) =  $N$ , (k) = 1 für die Charakterisierung des schwachen Entitytypen.



- (b) Übersetzen Sie das ER-Diagramm in ein relationales Schema. Ergänzen Sie die noch fehlenden Relationen, ausgehend von:

STUDI	: {[ <u>MatrNr</u> , Name, SPO ]}
PRÜFUNG	: {[ <u>PNr</u> , Fach, Datum ]}
MITARBEITER	: {[ <u>PersNr</u> , Name ]}

Geben Sie für jede Relation einen Primärschlüssel durch Unterstreichen der beteiligten Attribute an. Falls es weitere Schlüsselkandidaten gibt, geben Sie diese an. Typen brauchen nicht angegeben werden. Fassen Sie noch keine Relationen zusammen.

- (c) Verfeinern Sie Ihren Entwurf, indem Sie zwei Relationen Ihrer Wahl zusammenfassen. Achten Sie darauf, dass die Verfeinerung sinnvoll ist und insbesondere, dass keine unnötigen NULL-Werte entstehen können:

- Benennen Sie die Relationen, die ersetzt werden.
- Kennzeichnen Sie in der entstandenen Relation den von Ihnen gewählten Primärschlüssel durch Unterstreichen der beteiligten Attribute.
- Begründen Sie in ganzen Sätzen, warum durch die von Ihnen gewählte Verfeinerung keine NULL-Werte entstehen können.

**Aufgabe 2****[23 PUNKTE]**

Gegeben ist folgendes Datenbankschema einer Universität. Eine Beispielausprägung finden Sie am Ende dieser Teilaufgabe.

**Professoren:** {[PersNr, Name, Rang, Raum]}

**Studenten:** {[MatrNr, Name, Semester]}

**Vorlesungen:** {[VorlNr, Titel, SWS, gelesenVon]}

Attribut gelesenVon ist Fremdschlüssel auf die Relation Professoren

**prüfen:** {[MatrNr, VorlNr, PersNr, Note]},

Attribut MatrNr ist Fremdschlüssel auf die Relation Studenten und identifiziert den Prüfling,

Attribut VorlNr ist Fremdschlüssel auf die Relation Vorlesungen und identifiziert den Prüfungsstoff,

Attribut PersNr ist Fremdschlüssel auf die Relation Professoren und identifiziert den Prüfer.

Entwerfen Sie alle Anfragen so, dass Sie unabhängig von der konkreten Beispielausprägung korrekt ausgewertet werden. Die Ergebnisse sind ohne Duplikate zu berechnen, doch unnötige Duplikatelimination ist zu vermeiden.

Übertragen Sie die untenstehenden Anfragen aus den Teilaufgaben (a) bis (c) auf Ihr Lösungsblatt und ergänzen Sie nur in den Lücken. Lücken können auch leer bleiben.

- (a) Wie viele Studierende haben schon an fortgeschrittenen Prüfungen teilgenommen, d. h. an Prüfungen, die nicht die Vorlesung mit dem Titel "Grundzüge" behandeln?

Vervollständigen Sie folgende SQL-Anfrage mit einer *unkorrelierten* Unteranfrage. (Zur Erinnerung: Eine unkorrelierte Unteranfrage braucht nur einmal ausgewertet werden, das Ergebnis ist während der Auswertung der äußeren Anfrage konstant.)

```
SELECT ..... as AnzahlStudis
FROM   prüfen pr
WHERE  .....
```

- (b) Welche Studierenden haben schon mindestens einmal die Note 1 in einer Vorlesung mit 4 SWS erzielt? Vervollständigen Sie folgende SQL-Anfrage mit einer *korrelierten* Unteranfrage.

```
SELECT ..... s.MatrNr, s.Name
FROM   Studenten s
WHERE  .....
```

- (c) Welche Noten hat Prof. Sokrates schon mindestens 100 Mal vergeben? Vervollständigen Sie folgende SQL-Anfrage.

```
SELECT ..... pr.Note  
FROM   prüfen pr, Professoren p  
WHERE  .....  
GROUP BY .....  
.....
```

- (d) Welche Professoren haben noch nie die Note 1 vergeben? Sie sollten aber schon mindestens 50 Studierende geprüft haben, damit das Ergebnis aussagekräftig ist.

Formulieren Sie eine SQL-Anfrage, welche die Personalnummern und Namen dieser Professoren ausgibt.

**Aufgabe 3**

[18 PUNKTE]

Wir beziehen uns auf das Datenbankschema einer Universität aus Aufgabe 2 mit folgenden Beispieldaten der Relationen Studenten und prüfen:

Studenten		
<u>MatrNr</u>	Name	Semester
24002	Xenokrates	18
25403	Jonas	12
26120	Fichte	10
26830	Aristoxenos	8
27550	Schopenhauer	6
28106	Carnap	3
29120	Theophrastos	2
29555	Feuerbach	2

Da während des Prüfungszeitraums noch nicht alle Noten feststehen, befinden sich aktuell NULL-Werte in der Relation prüfen:

prüfen			
<u>MatrNr</u>	<u>VorlNr</u>	PersNr	Note
28106	5001	2126	<b>NULL</b>
25403	5041	2125	<b>NULL</b>
27550	4630	2137	1
25403	4630	2125	5

Fortsetzung nächste Seite!

Folgende SQL-Anfrage wird gestellt:

```
1 SELECT DISTINCT s.MatrNr, s.Name
2 FROM prüfen pr, Studenten s
3 WHERE s.MatrNr = pr.MatrNr AND NOT (pr.Note <= 4)
```

- (a) Geben Sie das Ergebnis der kanonischen Übersetzung der SQL-Anfrage in die relationale Algebra an.  
(b) Geben Sie das Ergebnis der logischen Optimierung der Anfrage in der relationalen Algebra an.

Beschränken Sie sich auf diese Optimierungen:

- Aufbrechen von Selektionen
- Selection Pushing
- Einführen von Joins

Die Angabe der Zwischenschritte ist nicht nötig.

Obige SQL-Anfrage sollte eigentlich Prüflinge ermitteln, die zu Prüfungen angetreten sind, zu denen aber während des Prüfzeitraums noch nicht alle Noten fest stehen (im konkreten Beispiel die Studierenden mit den Matrikelnummern 28106 und 25403):

- (c) Erklären Sie – Schritt für Schritt – wie die optimierte Anfrage aus Teilaufgabe (b) auf den oben angegebenen Beispieldaten ausgewertet wird und geben Sie das Ergebnis an.  
Legen Sie nachvollziehbar dar, warum die Anfrage nicht das gewünschte Ergebnis berechnet.
- (d) Wie sollte die SQL-Anfrage formuliert werden, damit sie das erwartete Ergebnis liefert?  
Formulieren Sie einen Verbesserungsvorschlag.

**Aufgabe 4**

[35 PUNKTE]

Wir betrachten die Relation ASSIGN. Sie hält fest, welche Piloten (PILOT) welchen Flügen zugewiesen werden. So fliegt Pilot "Cushing" den Flug mit Flugnummer 83 (FLIGHT) am 9. August (DATE) um 10:15 Uhr vormittags (DEPARTS) ab Gate 2 (GATE):

ASSIGN					
PILOT	FLIGHT	DATE	DEPARTS	GATE	
Cushing	83	9 Aug	10:15a	2	
Cushing	116	10 Aug	1:25p	7	
Clark	281	8 Aug	5:50a	2	
Clark	301	11 Aug	6:35a	11	
Clark	83	11 Aug	10:15a	2	
Chin	83	13 Aug	10:15a	2	
Chin	116	12 Aug	1:25p	7	
Copely	281	9 Aug	5:50a	2	
Copely	281	13 Aug	5:50a	2	
Copely	412	15 Aug	1:25p	4	

(a) Es gilt:

- Jede Flugnummer hat eine fixe Abflugzeit (DEPARTS).
- Jede Flugnummer hat ein fixes Gate (d. h. dieses ist an allen Flugtagen gleich).
- Ein Pilot kann nur eine Maschine auf einmal fliegen.
- Ein Flug wird durch genau einen Piloten durchgeführt (natürlich gibt es Co-Piloten, aber diese sind dann auch "nur" Co-Piloten).
- An einem Gate kann zeitgleich immer nur das Boarding für einen Flug stattfinden.

Formulieren Sie jede dieser Bedingungen als eine eigene, funktionale Abhängigkeit.

(b) Geben Sie drei Kandidatenschlüssel für die Relation ASSIGN an.

(c) Begründen Sie, warum das Relationenschema der Realtion ASSIGN mit der funktionalen Abhängigkeit aus Teilaufgabe (a) in der dritten Normalform ist. Diskutieren Sie in Ihrer Begründung jede funktionale Abhängigkeit *einzelnen*.

Fortsetzung nächste Seite!

- (d) Begründen Sie, warum das Relationsschema der Relation ASSIGN mit der funktionalen Abhängigkeit aus Teilaufgabe (a) nicht in der Boyce-Codd-Normalform (BCNF) ist. Geben Sie in Ihrer Begründung eine konkrete funktionale Abhängigkeit an, die diese Normalform verletzt.
- (e) Geben Sie das Ergebnis der Überführung in BCNF mit Hilfe des Dekompositionsalgorithmus an (Angabe der Zwischenschritte ist nicht nötig).  
Kennzeichnen Sie in jeder entstandenen Relation einen Primärschlüssel durch Unterstreichen der beteiligten Attribute.
- (f) Erläutern Sie zwei mögliche Gründe, die in der Praxis dazu führen könnten, dass Sie sich bei einer bereits in Betrieb genommenen Datenbank gegen eine Zerlegung in die BCNF entscheiden und die dritte Normalform bevorzugen.

**Aufgabe 5**

[12 PUNKTE]

Wir betrachten das folgende Schema. Es beschreibt gerichtete Beziehungen in einem sozialen Netzwerk:

Likes: {[name1, name2]}

Follows: {[name1, name2]}

So hat z. B. die Person mit name1 ein “Like” auf die Person mit name2 vergeben.

Schreiben Sie jeweils eine Anfrage, um die Namen aller Personen zu finden, die eine Person “liken”, die wiederum (mindestens) einer Person folgt.

- (a) In SQL (ohne Duplikate)
- (b) In der relationalen Algebra
- (c) Im Tupelkalkül

**Aufgabe 6**

[10 PUNKTE]

Die folgenden Teilaufgaben beziehen sich auf das Zwei-Phasen-Sperrprotokoll (2PL).

Folgende Notation wird für das Anfordern und Freigeben von Lese-/Schreibsperren durch Transaktion  $l_i, i \in \{1, 2, 3\ldots\}$ , für das Objekt  $x$  verwendet:

- Lesesperre anfordern:  $rl_i(x)$
- Lesesperre freigeben:  $ru_i(x)$
- Schreibsperre anfordern:  $wl_i(x)$
- Schreibsperre freigeben:  $wu_i(x)$
- Transaktion blockiert nach Anforderung einer Sperre:  $block_i$

a) Gegeben ist folgender Schedule:

	$T_1$	$T_2$	$T_3$
		$rl_2(x)$	
			$rl_3(x)$
$wl_1(x)$			
$block_1$			
$\downarrow$		$rl_2(y)$	
		$r_2(y)$	
			$r_3(x)$
			$ru_3(x)$
		$r_2(x)$	
		$ru_2(x)$	
		$ru_2(y)$	
$wl_1(x)$			
		$c_2$	
$w_1(x)$			
$wu_1(x)$			
$c_1$			
		$wl_3(z)$	
		$w_3(z)$	
		$wu_3(z)$	
			$c_3$

Ist dieser Schedule im 2PL Protokoll möglich? Begründen Sie Ihre Entscheidung kurz.

b) Gegeben ist folgender Schedule, allerdings ist dieser unvollständig.

$T_1$	$T_2$
--(1)--	
$w_1(z)$	
	$wl_2(z)$
	--(4)--
	--(5)--
--(2)--	
$c_1$	
--(3)--	
	$wl_2(z)$
	$w_2(z)$
	--(6)--
	$r_2(x)$
	$ru_2(x)$
	$wu_2(z)$
	$c_2$

Vervollständigen Sie die Lücken (\_\_\_\_) so, dass der Schedule einem 2PL-Schedule gehorcht:

- Geben Sie **auf Ihrem Lösungsblatt** die Nummer der Lücke und Ihren Eintrag an.
- Geben Sie explizit an, welche Lücken keine Einträge benötigen.

## Beispielausprägung

Professoren				Studenten		
PersNr	Name	Rang	Raum	MatrNr	Name	Semester
2125	Sokrates	C4	226	24002	Xenokrates	18
2126	Russel	C4	232	25403	Jonas	12
2127	Kopernikus	C3	310	26120	Fichte	10
2133	Popper	C3	52	26830	Aristoxenos	8
2134	Augustinus	C3	309	27550	Schopenhauer	6
2136	Curie	C4	36	28106	Carnap	3
2137	Kant	C4	7	29120	Theophrastos	2
				29555	Feuerbach	2

Vorlesungen				voraussetzen	
VorlNr	Titel	SWS	gelesenVon	Vorgänger	Nachfolger
5001	Grundzüge	4	2137	5001	5041
5041	Ethik	4	2125	5001	5043
5043	Erkenntnistheorie	3	2126	5001	5049
5049	Mäeutik	2	2125	5041	5216
4052	Logik	4	2125	5043	5052
5052	Wissenschaftstheorie	3	2126	5041	5052
5216	Bioethik	2	2126	5052	5259
5259	Der Wiener Kreis	2	2133		
5022	Glaube und Wissen	2	2134		
4630	Die 3 Kritiken	4	2137		

hören		Assistenten			
MatrNr	VorlNr	PersNr	Name	Fachgebiet	Boss
26120	5001	3002	Platon	Ideenlehre	2125
27550	5001	3003	Aristoteles	Syllogistik	2125
27550	4052	3004	Wittgenstein	Sprachtheorie	2126
28106	5041	3005	Rhetikus	Planetenbewegung	2127
28106	5052	3006	Newton	Keplersche Gesetze	2127
28106	5216	3007	Spinoza	Gott und Natur	2134
28106	5259	prüfen			
29120	5001	MatrNr	VorlNr	PersNr	Note
29120	5041	28106	5001	2126	1
29120	5049	25403	5041	2125	2
29555	5022	27550	4630	2137	2
25403	5022	25403	4630	2137	5
29555	5001				

Abbildung 1: Beispielausprägung für eine Universitäts-Datenbank

Fortsetzung nächste Seite!

**Teilaufgabe II: Softwaretechnologie****Aufgabe 1 (Vermischte Aufgaben)**

[15 PUNKTE]

- a) Definieren Sie den Begriff *Refactoring* und geben Sie zwei Beispiele dafür an.
- b) Gegeben sei folgende Anforderungsbeschreibung: „Das Programm soll auf allen Computersystemen laufen“. Beschreiben Sie ein Problem, das aus dieser Anforderungsbeschreibung entstehen kann. Formulieren Sie beispielhaft eine verbesserte Anforderungsbeschreibung, bei der dieses Problem nicht mehr auftreten kann.
- c) Erklären Sie einen Vorteil, den das Programmierkonzept *Vererbung* mit sich bringt. Zeichnen Sie zusätzlich ein UML-Diagramm, in dem Vererbung zum Einsatz kommt.

**Aufgabe 2 (Sequenzdiagramm)****[25 PUNKTE]**

Vergleichsportale, die Preise im Internet vergleichen und den günstigsten Anbieter ermitteln, erfreuen sich immer größerer Beliebtheit. In dieser Aufgabe soll ein solches Vergleichsportal namens Compariso betrachtet werden, welches seinen Kunden für eine beliebige Auswahl an Produkten eine PDF-Datei mit den jeweils günstigsten Anbietern erstellen kann.

Erstellen Sie ein **Sequenzdiagramm**, das den folgenden Vorgang des Vergleichportals Compariso modelliert:

- Der Nutzer startet den Anmeldevorgang bei Compariso.
- Nachdem der Nutzer den Anmeldevorgang gestartet hat, fordert das Vergleichsportal den Nutzer auf, seine Kombination aus Namen und Passwort einzugeben.
- Sollte der Nutzer eine falsche Kombination aus Namen und Passwort eingeben, fordert das Vergleichsportal den Nutzer so lange erneut auf, bis er die richtige Kombination eingibt.
- Nach dem erfolgreichen Anmelden kann der Nutzer nach einer Angeboteübersicht eines bestimmten Produkts suchen.
- Um dem Nutzer die Angebotsübersicht anzeigen zu können, fragt Compariso die aktuellste Angebotsübersicht von einer Datenbank eines Drittanbieters ab.
- Aus den verfügbaren Angeboten kann der Nutzer eine beliebige Auswahl davon zu seinen Favoriten hinzufügen.
- Wenn der Nutzer seine Auswahl abgeschlossen hat, kann der Vorgang auf zwei Arten beendet werden.
  - Erstens: der Nutzer lädt sich seine Auswahl als PDF-Datei herunter und meldet sich ab.
  - Zweitens: der Kunde meldet sich direkt ab.

In beiden Fällen bestätigt Compariso dem Nutzer seinen Abmeldevorgang.

**Aufgabe 3 (Testen)**

[35 PUNKTE]

Gegeben ist die folgende Funktion *Search* (in Java).

```
1 public static int search(int x, int v[]){  
2     int low, high, mid;  
3     low = 0;  
4     high = v.length;  
5  
6     for(int i=1;i<v.length;i++){  
7         if(v[i-1]>v[i]){  
8             return -2;  
9         }  
10    }  
11  
12    while(low <= high){  
13        mid = (low + high)/2;  
14        if(x<v[mid]){  
15            high = mid - 1;  
16        }  
17        else if(x > v[mid]){  
18            low = mid + 1;  
19        }  
20        else{  
21            return mid;  
22        }  
23    }  
24  
25    return -1;  
26 }
```

- Beschreiben Sie kurz, was beim Aufruf der die Funktion *Search* ausgegeben wird.
- Zeichnen Sie den Kontrollflussgraphen für die Funktion *Search*. Beschriften Sie die Knoten mit den jeweiligen Zeilen des Programmcodes.
- Geben Sie eine minimale Testmenge an, die das Kriterium der Zweigüberdeckung (Kantenüberdeckung) erfüllt.

Hinweis: Eine Testmenge ist minimal, wenn es keine Testmenge mit einer kleineren Zahl von Testfällen gibt. Sie müssen die Minimalität nicht beweisen.

- Definieren Sie den Begriff *vollständige Anweisungsüberdeckung*.

**Aufgabe 4 (Entwurfsmuster)**

[30 PUNKTE]

- a) Nennen Sie vier verschiedene Entwurfsmuster und geben Sie zusätzlich für jedes genannte Entwurfsmuster an, zu welchem Zweck dieses verwendet wird.

Hinweis: Betrachten Sie etablierte Entwurfsmuster und nicht lediglich *Interface* bzw. *Vererbung*.

- b) Geben Sie für zwei der in der vorherigen Teilaufgabe genannten Entwurfsmuster das dazugehörige UML Diagramm an.

**Aufgabe 5 (Prozessmodelle)**

[15 PUNKTE]

- a) Geben Sie die Phasen des Prozessmodells namens V-Modell in der richtigen Reihenfolge an.
- b) Nennen Sie ein weiteres Prozessmodell.
- c) Vergleichen Sie das V-Modell mit dem von Ihnen in der vorherigen Teilaufgabe genannten Prozessmodell. Nennen Sie dafür jeweils zwei allgemeine Vorteile und zwei allgemeine Nachteile der beiden Prozessmodelle. (Insgesamt werden vier Vorteile und vier Nachteile erwartet.)

Thema Nr. 2  
 (Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!  
 Alle Lösungsschritte sind sorgfältig zu begründen!

**Teilaufgabe I: Datenbanksysteme**

**Aufgabe 1 (Datenbankmodellierung und Entwurf)**

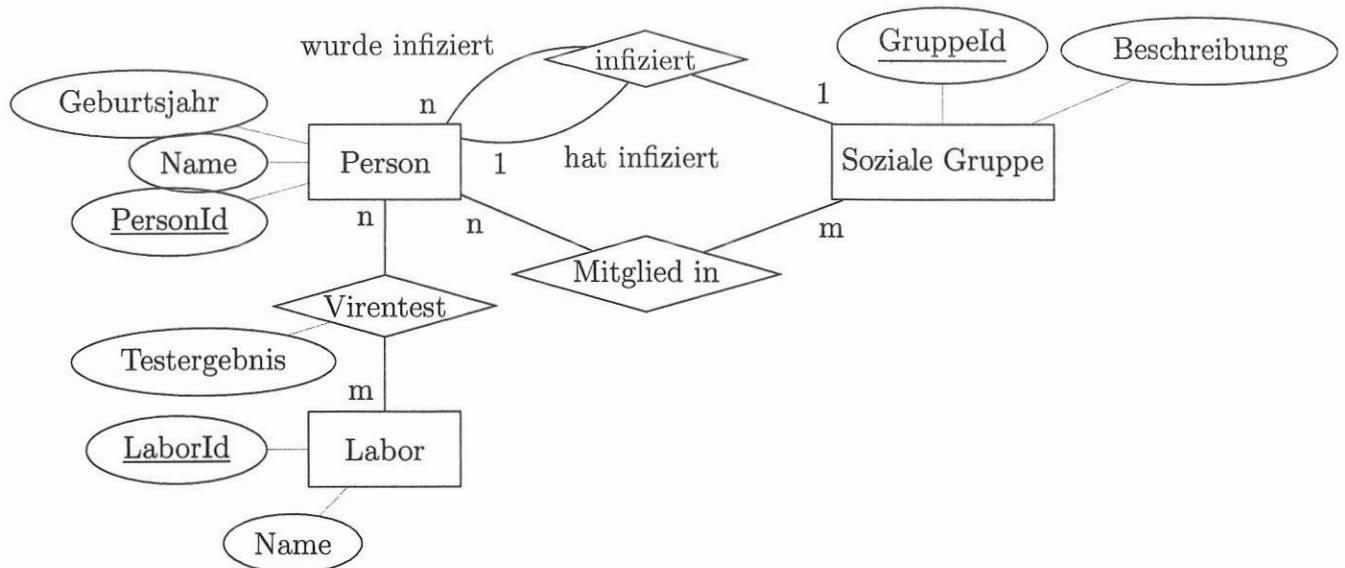
[12 PUNKTE]

Zur Nachverfolgung von Infektionen soll ein Schema für eine Datenbank modelliert werden.

In der Datenbank sollen hauptsächlich *Personen* und *soziale Gruppen* gespeichert werden. Eine soziale Gruppe beschreibt ein Umfeld, in dem sich mehrere Personen häufiger regelmäßig treffen. Eine Person kann dabei Mitglied in mehreren verschiedenen sozialen Gruppen sein. Auch sollen bestätigte *Infektionen* festgehalten werden. Das Modell soll dabei abbilden, welche Personen welche anderen Personen durch welche soziale Gruppe infiziert haben.

Zusätzlich sollen noch *Virentests* modelliert werden. Eine Person kann einen Virentest in einem *Labor* machen, dessen Ergebnis entweder positiv oder negativ ist.

Ein mögliches ER-Diagramm für dieses System könnte wie folgt aussehen:



Fortsetzung nächste Seite!

a) Beantworten Sie die folgenden Fragen zu der Modellierung in diesem Diagramm in jeweils **ein bis zwei Sätzen**:

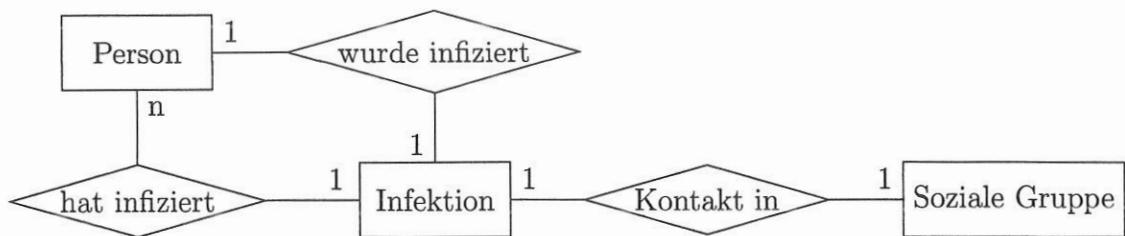
1. Wodurch ermöglicht das modellierte Schema, dass eine Person Mitglied in mehreren Gruppen sein kann?
2. Kann es nach diesem Diagramm Personen geben, die jemanden infiziert haben, aber negativ getestet wurden? Begründen Sie!
3. Stellt das modellierte Schema sicher, dass es keine Zyklen in der Infektionskette geben kann, also beispielsweise Person A steckt Person B an, Person B steckt Person C an und Person C steckt wieder Person A an? Begründen Sie!

b) Hier sind zwei weitere ER-Diagramme abgebildet, die Personen, soziale Gruppen und Infektionen modellieren:

**Diagramm 1:**



**Diagramm 2:**



1. Beschreiben Sie die Unterschiede in der Modellierung beider Diagramme (zwei bis drei Sätze).
2. Bewerten Sie die allgemeinen Vor- und Nachteile der zwei gezeigten Arten der Modellierung (zwei bis drei Sätze).

**Aufgabe 2 (Relationales Modell)****[8 PUNKTE]**

Verwenden Sie für diese Aufgabe das Datenbankschema einer Universität. Sie finden eine Beispieldaten ausprägung am Ende dieser Teilaufgabe.

1. Finden Sie die Professoren mit einer zweistelligen Raumnummer (10-99). Geben Sie nur die Namen aus. Formulieren Sie die Anfrage im **Tupelkalkül**.
2. Finden Sie die Studenten, die mindestens eine Vorlesung hören. Geben Sie nur die Namen aus. Formulieren Sie die Anfrage im **Tupelkalkül**.

**Aufgabe 3 (SQL)****[29 PUNKTE]**

Die folgenden Fragen beziehen sich auf das Universitäts-Beispiel (vgl. Aufgabe 2), welches in Form einer Beispieldaten ausprägung auf dem Beiblatt am Ende dieser Teilaufgabe zu finden ist. Die Antworten sind in SQL zu formulieren. Die Ergebnisse sind ohne Duplikate zu berechnen, wobei unnötige Duplikatelimination zu vermeiden ist.

In jeder der Teilaufgabe finden Sie das erwartete Ergebnis für die Beispieldaten ausprägung. Ihre jeweilige Anfrage muss natürlich auch dann funktionieren, wenn die Ausprägung der Relationen anders ist als die Beispieldaten ausprägung.

1. Geben Sie die Matrikelnummern und Namen aller Studenten aus, die mindestens eine Vorlesung, die von Professor Sokrates gelesen wird, hören.

Matrnr	Name
27550	Schopenhauer
28106	Carnap
29120	Theophrastos

2. Geben Sie die Namen der Professoren aus, die mindestens zwei Assistenten haben.

Name
Kopernikus
Sokrates

3. Geben Sie aus, wie viele verschiedene Studenten jeder Professor in seinen Vorlesungen unterrichtet. Studenten, die mehrere Vorlesungen eines Professors hören, sollen dabei nur einmal gezählt werden. Professoren, die keine Hörer haben, sollen nicht ausgegeben werden.

Name	AnzStudenten
Kant	4
Augustinus	2
Russel	1
Popper	1
Sokrates	3

4. Formulieren Sie die Antwort zu Teilaufgabe 3 so um, dass auch die Professoren ausgegeben werden, die keine Hörer haben. Tipp: Sie können dazu LEFT (OUTER) JOINS verwenden.

Name	AnzStudenten
Kopernikus	0
Curie	0
Kant	4
Augustinus	2
Russel	1
Popper	1
Sokrates	3

5. Betrachten Sie folgende SQL-Anfrage:

```
SELECT Name FROM Professoren
WHERE PersNr NOT IN (SELECT gelesenVon FROM Vorlesungen)
```

- a) Beschreiben Sie textuell (in einem Satz), welches Ergebnis diese Anfrage liefert.  
 b) Formulieren Sie eine semantisch äquivalente SQL-Anfrage, die statt einem NOT IN ein NOT EXISTS verwendet.  
 Hinweis: Sie können davon ausgehen, dass es keine NULL-Werte in den Daten gibt.

**Aufgabe 4 (Relationale Entwurfstheorie)**

[34 PUNKTE]

1. Gegeben sei ein abstraktes Relationenschema  $\mathcal{R} = \{A, B, C, D, E, F\}$  mit den funktionalen Abhängigkeiten

$$FD = \{A \rightarrow DF, BF \rightarrow CDE, C \rightarrow ADF, E \rightarrow BF, F \rightarrow AB\}.$$

- a) Berechnen Sie die Attributhülle von  $\{A\}$ .
- b) Geben Sie die vier Kandidatenschlüssel an.
- c) Berechnen Sie die kanonische Überdeckung von  $FD$ . Benennen Sie dabei jeden Schritt und geben Sie das Zwischenergebnis nach jedem Schritt (Linksreduktion, Rechtsreduktion, leere Menge entfernen, funktionale Abhängigkeiten zusammenfassen) an.
2. Gegeben sei nun folgende Beispieldaten

ProfessorenAllerlei								
PersNr	Name	Rang	Raum	VorlNr	VorlTitel	VorlTag	VorlZeit	Hörsaal
2138	Codd	C4	123	5432	Datenbanken	Di	8:15	D003
2138	Codd	C4	123	5432	Datenbanken	Do	10:00	D002
...	...	...	...	...	...	...	...	...

mit den funktionalen Abhängigkeiten in kanonischer Überdeckung

- $PersNr \rightarrow Name, Rang, Raum$
- $Raum \rightarrow PersNr$
- $VorlNr \rightarrow PersNr, VorlTitel$
- $VorlNr, VorlTag \rightarrow VorlZeit, Hörsaal$
- $VorlTag, VorlZeit, Hörsaal \rightarrow VorlNr$

- a) Bestimmen Sie den oder die Kandidatenschlüssel (ohne Begründung).
- b) Begründen Sie kurz, warum das Schema nicht in zweiter Normalform ist.

Fortsetzung nächste Seite!

- c) Überführen Sie das Schema mit Hilfe des Synthesealgorithmus in die dritte Normalform. Benennen Sie dabei jeden Schritt des Algorithmus und geben Sie das Zwischenergebnis nach jedem Schritt an. Beachten Sie, dass die funktionalen Abhängigkeiten bereits in der kanonischen Überdeckung vorliegen und daher nicht weiter reduziert werden (Links-/Rechtsreduktion) müssen.  
Kennzeichnen Sie im Endergebnis in jeder Relation einen Primärschlüssel durch Unterstreichen.

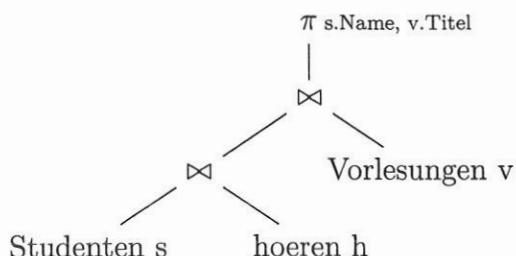
**Aufgabe 5 (Anfrageoptimierung)**

[19 PUNKTE]

1. Betrachten Sie die folgende SQL-Anfrage:

```
SELECT DISTINCT s1.Name
FROM Studenten s1, hören h1, hören h2, Studenten s2
WHERE s1.MatrNr = h1.MatrNr AND h1.VorlNr = h2.VorlNr AND
      h2.MatrNr = s2.MatrNr AND
      s2.Name = 'Theophrastos';
```

- a) Übersetzen Sie diese SQL-Anfrage gemäß der kanonischen Übersetzung in die relationale Algebra (Operatorbaum-Darstellung). Etwaige Umbenennungsoperatoren brauchen dabei nicht angegeben werden. Führen Sie noch keine Optimierungen durch.
- b) Führen Sie nun die Optimierung durch (ohne *Projections Pushing*). Achten Sie dabei auch auf die optimale Join-Reihenfolge.
- c) Ermitteln Sie für den optimalen Auswertungsplan aus Teilaufgabe b) die Größen der beteiligten Basisrelationen, der Zwischenergebnisse und des Endergebnisses für die am Ende dieser Teilaufgabe zu findende Beispieldarstellung des Datenbankschemas einer Universität. Tragen Sie diese Werte in den Operatorbaum ein.
2. Gegeben sei nun die folgende Anfrage in relationaler Algebra. Geben Sie das Ergebnis von *Projections Pushing* (Entfernung von nicht benötigten Attributen durch frühe Einführung von Projektionen  $\pi$ ) an.



Fortsetzung nächste Seite!

**Aufgabe 6 (Transaktionsmanagement)**

[18 PUNKTE]

Auf einem Datenbanksystem laufen zwei Transaktionen  $T_1$  und  $T_2$ . Ohne Synchronisation passiert dabei Folgendes:

1.  $T_1$  startet.
  2.  $T_1$  liest das Datum  $A$ .
  3.  $T_1$  schreibt das Datum  $A$ .
  4.  $T_2$  startet.
  5.  $T_2$  liest das Datum  $A$ .
  6.  $T_2$  schreibt das Datum  $A$ .
  7.  $T_2$  committed.
  8.  $T_1$  liest das Datum  $B$ .
  9.  $T_1$  schreibt das Datum  $B$ .
  10.  $T_1$  committed.
- a) Ist diese Historie serialisierbar? Ist sie rücksetzbar?  
Geben Sie jeweils eine kurze Begründung an.
- b) Charakterisieren Sie das *Zwei-Phasen-Sperrprotokoll* (2PL). Gehen Sie dabei darauf ein, nach welchem Schema Sperren angefordert und freigegeben werden und worin der Unterschied zwischen dem “normalen”/konservativen 2PL und dem strengen 2PL im Hinblick auf das Freigeben von Sperren liegt.

- c) Welche Historie erzeugt das *strenge Zwei-Phasen-Sperrprotokoll* aus obigem Ablauf?

Geben Sie hierfür alle Lese- und Schreiboperationen der beteiligten Transaktionen, Beginn und Ende der Transaktionen sowie das Anfordern und Freigeben von Sperren in der richtigen Reihenfolge an. Geben Sie unter „Kommentar“ an, wenn Transaktionen blockiert werden und wann sie weiterlaufen können. Orientieren Sie sich beim Aufbau Ihrer Antwort an folgender Tabelle. Übertragen Sie diese auf Ihren Bearbeitungsbogen.

Fordern Sie immer gleich das strengste Lock an, das die Transaktion in Zukunft benötigen wird.

$T_1$	$T_2$	Kommentar
BEGIN  lock <sub>X</sub> (A)  read(A)		

Professoren			
PersNr	Name	Rang	Raum
2125	Sokrates	C4	226
2126	Russel	C4	232
2127	Kopernikus	C3	310
2133	Popper	C3	52
2134	Augustinus	C3	309
2136	Curie	C4	36
2137	Kant	C4	7

Studenten		
MatrNr	Name	Semester
24002	Xenokrates	18
25403	Jonas	12
26120	Fichte	10
26830	Aristoxenos	8
27550	Schopenhauer	6
28106	Carnap	3
29120	Theophrastos	2
29555	Feuerbach	2

Vorlesungen			
VorlNr	Titel	SWS	gelesen Von
5001	Grundzüge	4	2137
5041	Ethik	4	2125
5043	Erkenntnistheorie	3	2126
5049	Mäeutik	2	2125
4052	Logik	4	2125
5052	Wissenschaftstheorie	3	2126
5216	Bioethik	2	2126
5259	Der Wiener Kreis	2	2133
5022	Glaube und Wissen	2	2134
4630	Die 3 Kritiken	4	2137

voraussetzen	
Vorgänger	Nachfolger
5001	5041
5001	5043
5001	5049
5041	5216
5043	5052
5041	5052
5052	5259

hören	
MatrNr	VorlNr
26120	5001
27550	5001
27550	4052
28106	5041
28106	5052
28106	5216
28106	5259
29120	5001
29120	5041
29120	5049
29555	5022
25403	5022
29555	5001

Assistenten			
PersNr	Name	Fachgebiet	Boss
3002	Platon	Ideenlehre	2125
3003	Aristoteles	Syllogistik	2125
3004	Wittgenstein	Sprachtheorie	2126
3005	Rhetikus	Planetenbewegung	2127
3006	Newton	Keplersche Gesetze	2127
3007	Spinoza	Gott und Natur	2134

prüfen			
MatrNr	VorlNr	PersNr	Note
28106	5001	2126	1
25403	5041	2125	2
27550	4630	2137	2
25403	4630	2137	5

Fortsetzung nächste Seite!

**Teilaufgabe II: Softwaretechnologie****Aufgabe 1 (Agile Softwareentwicklung)**

[20 PUNKTE]

- (a) Grenzen Sie die Begriffe *Sprint Backlog* und *Product Backlog* im Kontext von SCRUM voneinander ab. Geben Sie dabei für beide Begriffe auch an, in welchen SCRUM-Aktivitäten neue Einträge hinzugefügt werden.
- (b) Beschreiben Sie zwei Vorteile und zwei Nachteile des *Prototyping*, jeweils mit Begründung.
- (c) Erklären Sie den Unterschied zwischen einem *vertikalen* und einem *horizontalen* Prototypen.
- (d) Agile Methoden beschreiben allgemeine Vorgehensweisen und Techniken, die in agilen Prozessen (z.B. Extreme Programming, SCRUM) zum Einsatz kommen. Nennen und beschreiben Sie vier agile Methoden.  
Hinweis: Gemeint sind *nicht agile* Prinzipien oder Leitsätze.
- (e) Nennen Sie fünf der zwölf im Agilen Manifest aufgelisteten Prinzipien.

**Aufgabe 2 (Anforderungsanalyse)**

[25 PUNKTE]

Gegeben ist die folgende Beschreibung für ein Krankenkassen-System:

Eine Krankenkasse bietet unterschiedliche Dienstleistungen für ihre Kunden an. Hierbei hat ein Versicherter Anspruch auf folgende Leistungen: Eine Krankmeldung kann sowohl vor Ort als auch digital eingereicht werden. Des Weiteren kann man sich für Kurse eintragen. Diese unterteilen sich in Fitness- und Entspannungskurse. Ebenso kann eine Kur beantragt oder auch eine Rechnung eingereicht werden. Für Premium-Versicherte besteht zudem die Möglichkeit, sich für Impfungen vorab anzumelden oder eine persönliche Beratung zu vereinbaren. Eine Angestellte der Krankenkasse kann Videos auf die Homepage hochladen sowie Rundmails verfassen.

Das Einreichen einer Rechnung schließt sowohl eine maschinelle Prüfung als auch eine manuelle Prüfung durch eine Angestellte mit ein. Weiterhin muss bei einer Kursteilnahme eine Gebühr entrichtet werden. Premium-Versicherte können bei der Begleichung der Kursgebühr eine Vergünstigung anwenden. Außerdem können Premium-Versicherte neben den bereits angebotenen Kursen an Fortbildungskursen teilnehmen. Die Krankenkasse bietet *keine* Krankenhaussuche als Dienstleistung für ihre Versicherten an.

- (a) Erstellen Sie eine *User Story*, bestehend aus einer Beschreibung und drei Akzeptanzkriterien, für einen Premium-Versicherten.

Fortsetzung nächste Seite!

- (b) Erstellen Sie ein UML *Use-Case Diagramm*, das die im beschriebenen System enthaltenen Use Cases und Akteure in Verbindung setzt.

**Hinweise:** Verwenden Sie Vererbung, Spezialisierung und Generalisierung, wo sinnvoll. Kennzeichnen Sie außerdem *include-* bzw. *extend*-Beziehungen entsprechend. Auch Bedingungen müssen hervorgehoben werden.

**Aufgabe 3 (Entwurfsmuster)****[25 PUNKTE]**

Einfache arithmetische Terme über ganze Zahlen können wie folgt rekursiv definiert werden:

- Eine Zahl ist ein Term.
- Eine Variable ist ein Term.
- Eine binäre Operation ist ein Term. Sie besitzt zwei Operanden, die wiederum Terme sind, und einen Operator aus der Menge  $\{+, -, \times, \div\}$ .

Für jeden Term gibt es die Möglichkeit, *ausgewertet* zu werden. Dabei wird der Wert des Terms rekursiv bestimmt und zurückgegeben. Ferner besteht bei einer binären Operation die Möglichkeit, jeden der Terme abzufragen.

- (a) Modellieren Sie die angegebenen Terme mit Hilfe eines UML-Klassendiagramms. Verwenden Sie dabei ein etabliertes Entwurfsmuster, wie es in der Literatur zu finden ist. Benennen Sie das von Ihnen verwendete Entwurfsmuster.
- (b) Terme in dieser Struktur sind natürlich kompliziert zu verstehen. Daher ist es notwendig, die Terme in einer menschenlesbaren Darstellung bereitzustellen. Hierzu verwenden wir das so genannte *Pretty Printing*.

Um dies umzusetzen sollen alle Terme einen Besucher akzeptieren. In unserem Beispiel soll das ein Besucher sein, der die einzelnen Termbestandteile in die gewöhnliche, menschenlesbare Form überführt.

Erstellen Sie, basierend auf Ihrem Klassendiagramm aus Teilaufgabe a) ein neues Klassendiagramm, welches Sie um die entsprechend notwendigen Klassen und Methoden ergänzen. Verwenden Sie dabei ebenfalls ein bekanntes Entwurfsmuster und benennen Sie dieses anschließend.

Diesem Entwurfsmuster liegt ein bestimmtes Aufrufprinzip für Methoden zu Grunde. Nennen und erklären Sie dieses Aufrufprinzip.

- (c) Die Implementierung der arithmetischen Terme soll nun in eine größere Anwendung eingebettet werden. Hierzu soll es einen Monitor geben, der über jede Veränderung an einem Term informiert wird und die Veränderung an den Benutzer des Systems weitergeben kann. Erstellen Sie, basierend auf Ihrem Klassendiagramm aus Teilaufgabe a), ein neues Klassendiagramm, welches Sie um die entsprechend notwendigen Klassen und Methoden ergänzen. Verwenden Sie dabei ebenfalls ein etabliertes Entwurfsmuster und benennen Sie dieses anschließend.

**Aufgabe 4 (Sequenzdiagramm)**

[20 PUNKTE]

Sie sind *Premium-Versicherter* (= *Premium-Benutzer*) bei Ihrer Krankenkasse. Über das Webportal wollen Sie sich in einen *Fitnesskurs* einschreiben. Dazu sei folgender Programmcode (in Java) gegeben:

```
1 class Krankenkasse {
2     int MAXIMALE_TEILNEHMER_ANZAHL = 10;
3
4     String einschreiben(Kurs kurs, Benutzer benutzer) {
5         DBWrapper db = new DBWrapper();
6         int teilnehmerAnzahl = db.ermittleTeilnehmerAnzahl(kurs);
7         if (teilnehmerAnzahl <= MAXIMALE_TEILNEHMER_ANZAHL) {
8             int preis = db.ermittleKursPreis(kurs);
9             if (istPremiumBenutzer(benutzer)) {
10                 preis = preis / 2; // 50 % Rabatt für Premium-Benutzer
11             }
12             Zahlstelle zahlstelle = new Zahlstelle();
13             if (zahlstelle.ueberweisen(preis)) {
14                 return "Erfolgreich in Kurs eingeschrieben!";
15             } else {
16                 return "Zahlungsvorgang fehlgeschlagen!";
17             }
18         } else {
19             return "Leider kein Platz mehr im aktuellen Kurs!";
20         }
21     }
22
23     boolean istPremiumBenutzer(Benutzer benutzer) { /* ... */ }
24 }
```

```
1 enum Kurs {
2     FITNESSKURS,
3     ENTSPANNUNGSKURS,
4     KOCHKURS,
5     // viele weitere Kurse
6 }
```

Fortsetzung nächste Seite!

```
1 class DBWrapper {  
2     int ermittleTeilnehmerAnzahl(Kurs kurs) {  
3         if (kurs == Kurs.FITNESSKURS) {  
4             return 8;  
5         } else if (kurs == Kurs.ENTSPANNUNGSKURS) {  
6             return 5;  
7         } else if (kurs == Kurs.KOCHKURS) {  
8             return 9;  
9         } else { /* sonstiger Kurs */ }  
10    }  
11  
12    int ermittleKursPreis(Kurs kurs) {  
13        if (kurs == Kurs.FITNESSKURS) {  
14            return 60;  
15        } else if (kurs == Kurs.ENTSPANNUNGSKURS) {  
16            return 40;  
17        } else if (kurs == Kurs.KOCHKURS) {  
18            return 120;  
19        } else { /* sonstiger Kurs */ }  
20    }  
21}
```

```
1 class Zahlstelle {  
2     boolean ueberweisen(int betrag) { /* ... */ }  
3 }
```

Modellieren Sie den Sachverhalt, der im obigen Einführungstext beschrieben wurde, als Sequenzdiagramm.

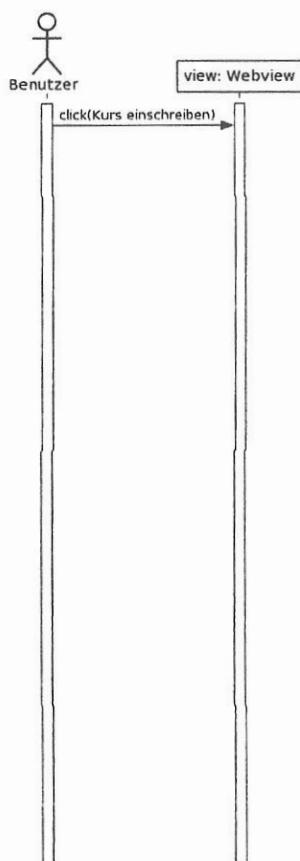
Beachten Sie dabei:

Nachdem ein Benutzer in der WebView auf "Kurs einschreiben" geklickt hat, wird von der WebView die Methode `einschreiben(Kurs kurs, Benutzer benutzer)` der Klasse *Krankenkasse* aufgerufen. Sie dürfen annehmen, dass eine Überweisung an die Zahlstelle immer erfolgreich ist.

Hinweise:

- Übertragen Sie das nachfolgende Diagramm als Grundlage für Ihre Lösung auf Ihren Bearbeitungsbogen.
- Sie dürfen annehmen, dass die Klasse *WebView* bereits eine Instanz der Klasse *Krankenkasse* verwaltet, d.h. Sie müssen den Konstruktor nicht explizit aufrufen.
- Ihre Lösung soll die vorgegebene **konkrete** Ausführung widerspiegeln, d.h. Sie sollen *if-Anweisungen* auflösen.
- Inkludieren Sie die konkreten Rückgabewerte der Methodenaufrufe im Sequenzdiagramm.

Ausgangslage:



**Aufgabe 5 (Testing)**

[30 PUNKTE]

- (a) Nennen Sie die verschiedenen *Test Levels* (Teststufen) nach dem V-Modell.
- (b) Beschreiben Sie die Testverfahren Black-Box-Testing und White-Box-Testing, und nennen Sie für jedes einen Vorteil gegenüber dem anderen.
- (c) Gegeben sei die Funktion `pow`, welche die Potenzfunktion der Eingabewerte  $x^y$  berechnet:

```
1 double pow(int x, int y) {
2     double z;
3     int pow = y;
4     if (y < 0)
5         pow = -pow;
6
7     z = 1;
8
9     while (pow != 0) {
10         z = z * x;
11         pow = pow - 1;
12     }
13
14     if (y < 0)
15         z = 1 / z;
16
17     return z;
18 }
```

Zeichnen Sie den Kontrollflussgraphen für die Funktion `pow`. Verwenden Sie Zeilennummern um die Knoten des Graphen zu beschriften. Verwenden Sie einen expliziten Start- und End-Knoten und sonst nur Knoten für Anweisungen.

- (d) Wie groß ist die zyklomatische Komplexität nach McCabe (cyclomatic complexity) der Methode `pow`? Geben Sie hierzu die Formel sowie auch Ihre Berechnungen mit an.
- (e) Wählen Sie eine *minimale* Menge von Programmeingaben in der Form `pow(..., ...)`, die zu 100% Anweisungsüberdeckung führt. Ermitteln Sie auch die Zweigüberdeckung dieser Test Suite. Geben Sie jeweils die allgemeine Formel zur Berechnung der Überdeckung an, entnehmen Sie die konkreten Werte dieser Aufrufe aus dem Kontrollflussgraphen und berechnen Sie schließlich die erreichte Überdeckung. Brüche brauchen dabei nicht gekürzt werden. Beziehen Sie *Start-* und *Endknoten* in Ihre Berechnung ein.
- (f) Gegeben ist der Testfall `pow(0, 0) == 0`. Berechnen Sie die Anweisungsüberdeckung und Zweigüberdeckung dieses Testfalls. Wenn die Überdeckung noch nicht 100% ist, ergänzen Sie sie durch weitere Tests (Programmeingaben) in der Form `pow(..., ...)`, die zu 100% Anweisungsüberdeckung und 100% Zweigüberdeckung führen.

Hinweis: "Minimal" bedeutet in diesem Fall, dass kein Test aus dieser Menge weggelassen werden kann ohne die Überdeckung zu verringern.

