Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer

Kennzahl: _____

Kennwort: _____

Arbeitsplatz-Nr.: _____

Herbst 2013

66116

Erste Staatsprüfung für ein Lehramt an öffentlichen Schulen

— Prüfungsaufgaben —

Fach: **Informatik (vertieft studiert)**

Einzelprüfung: **Datenbanksysteme, Softwaretechnologie**

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

Anzahl der Druckseiten dieser Vorlage: **17**

Bitte wenden!

Thema Nr. 1

Teilaufgabe 1:

1. Grundwissen

1.1 Bewerten Sie die folgenden Aussagen.

Geben Sie für die folgenden Aussagen an, ob sie richtig oder falsch sind. Begründen Sie Ihre Antwort kurz.

a) Ein perfekter Datenbankpuffer müsste die zukünftige Seitenreferenzfolge kennen.

b) Ein Hash-Index eignet sich gut zur Beschleunigung von Bereichsanfragen.

c) Fremdschlüsselwerte müssen eindeutig sein.

d) Eine Relation kann mehrere Primärschlüsselattribute besitzen.

e) Die Begriffe Datenbankverwaltungssystem und Datenbank sind äquivalent.

1.2 Erläutern Sie kurz die folgenden Begriffe und Abkürzungen im Kontext von Datenbanksystemen.

a) Index

b) Normalisierung

c) Internes Schema

d) Stored Procedure

e) ORM

Fortsetzung nächste Seite!

2. Transaktionen

a) Erläutern Sie kurz die vier wesentlichen Eigenschaften einer Transaktion.

b) Können zwei parallel ablaufende Transaktionen, unter Garantie der Eigenschaften aus a), schreibend auf dasselbe Datenbankobjekt zugreifen? Begründen Sie Ihre Antwort.

c) Kann bei zwei parallel ablaufenden Transaktionen, unter Garantie der Eigenschaften aus a), eine Transaktion schreibend auf ein Datenbankobjekt zugreifen, das von der anderen Transaktion gelesen wurde? Begründen Sie ihre Antwort.

3. Relationale Algebra

Relationenalgebraische Ausdrücke werden genutzt, um die Bearbeitung von SQL-Anfragen zu optimieren. Die Literatur kennt dabei eine Menge von Transformationsregeln. Untersuchen Sie die folgenden drei Regeln auf Richtigkeit bzw. Nichtrichtigkeit und begründen Sie dies kurz, wobei R (A1, ..., An), S (B1, ..., Bm) und T (C1, ..., Ck) Relationen sind.

a) Der Verbund ist assoziativ, d. h. es gilt:
join(join (R, S),T) = join (R,(join (S, T))

b) Alle Mengenoperationen sind kommutativ.

c) Welche Operation der relationalen Algebra wird durch SQL zunächst nur unvollständig umgesetzt (Verletzung der Relationeneigenschaften bei Anwendung)? Welche Eigenschaft fehlt dieser vereinfachten Umsetzung?

d) Nennen Sie das SQL-Schlüsselwort, welches für die Operation aus c) ein Verhalten gemäß der relationalen Algebra erzwingt. Aus welchem Grund ist nicht Algebra konformes Verhalten in manchen Situationen besser?

Fortsetzung nächste Seite!

4 . Normalformen

a) Die Normalisierung von Relationenschemata dient der Vermeidung von Redundanzen und dadurch bedingter Anomalien. Geben Sie ein Beispiel für eine nicht-normalisierte Relation an und erläutern Sie zwei mögliche Anomalien an diesem Beispiel.

b) Normalisierung beruht auf dem Erkennen und Eliminieren von funktionalen Abhängigkeiten. Erläutern Sie in diesem Zusammenhang kurz die folgenden Begriffe

1. Funktionale Abhängigkeit
2. Voll-funktionale Abhängigkeit
3. Transitive funktionale Abhängigkeit
4. Superschlüssel
5. Determinante

c) In welcher Normalform ist das folgende Beispiel? Zeigen Sie, dass alle Bedingungen für diese Normalform erfüllt sind. Welche Bedingung der nächsthöheren Normalform ist verletzt?

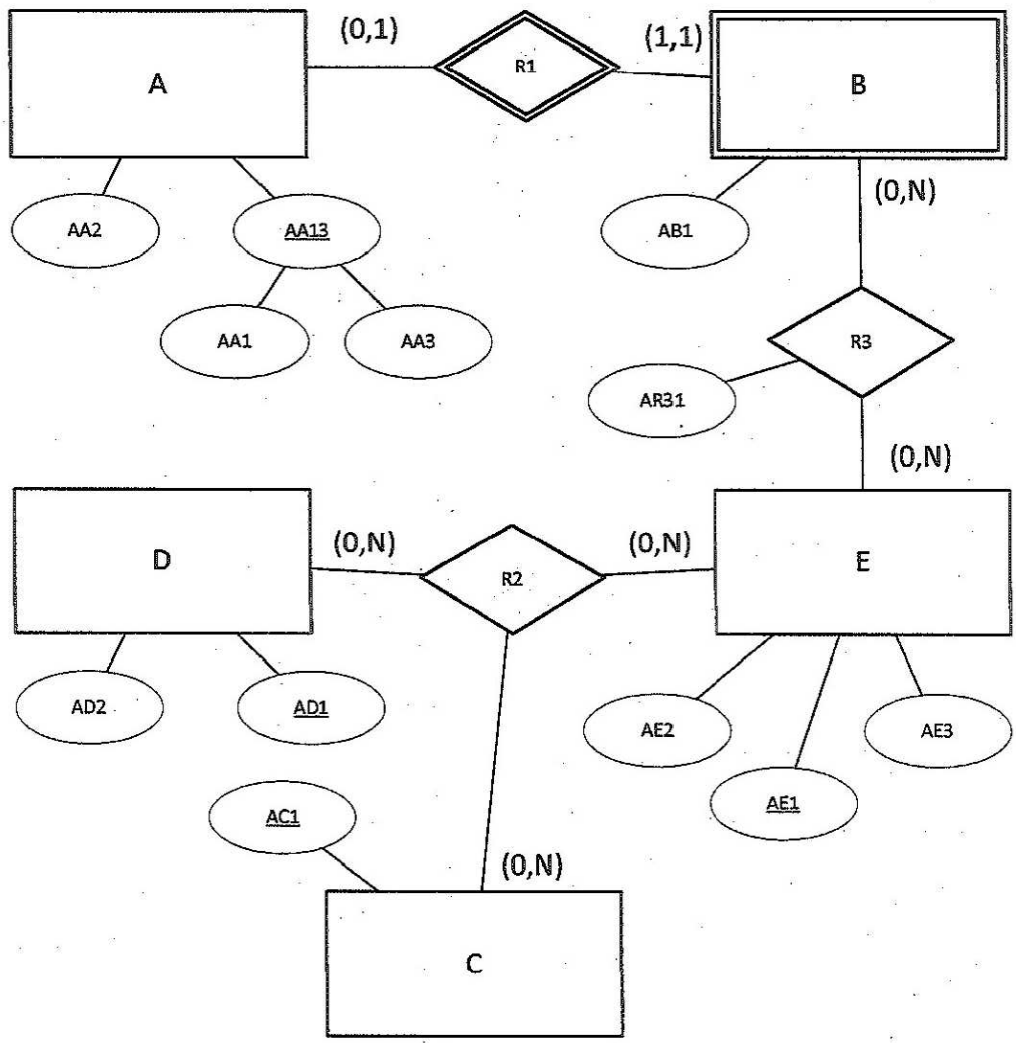
PersNr	Name	Land	Landesvorwahl	Vorwahl	Rufnummer
1	Maier	Frankreich	33	556	56881
2	Maier	Deutschland	49	821	4563
3	Pascal	Frankreich	33	556	56881
4	Konrad	Deutschland	49	841	923476
5	Schmidt	USA	1	0212	928357
6	Berner	Frankreich	33	556	183258

d) Erklären Sie die Begriffe Verlustlosigkeit und Abhängigkeitsbewahrung bei der Zerlegung von Relationen.

Fortsetzung nächste Seite!

5. E/R-Modellierung und Relationenmodell

Entwerfen Sie zum untenstehenden E/R-Diagramm ein Relationenschema in dritter Normalform (3. NF) mit möglichst wenigen Relationen.
Verwenden Sie dabei folgende Notation:
Primärschlüssel werden durch Unterstreichen gekennzeichnet und Fremdschlüssel durch die Nennung der Relation auf die sie verweisen in eckigen Klammern hinter dem Fremdschlüsselattribut. Attribute zusammengesetzter Fremdschlüssel werden durch runde Klammern als zusammengehörig markiert.
Beispiel: Relation1 (Primärschlüssel, Attribut1, Attribut2, Fremdschlüssel[Relation2])



Fortsetzung nächste Seite!

6. Relationen und SQL

Gegeben ist folgendes Relationenschema zur Verwaltung von Studenten und Kursen:
`STUDENT (Matrikelnummer, Name, Geburtsdatum)`
`KURS (Kursnummer, Kursname, Kursleiter)`
Die Primärschlüssel der Relationen sind wie üblich durch Unterstreichung gekennzeichnet.
Verwenden Sie für die zu formulierenden SQL-Anfragen nur standardkonformes SQL.

a) Erweitern Sie das obige Schema so, dass eine m:n-Beziehung zwischen STUDENT und KURS dargestellt werden kann: Ein Kurs kann aus mehreren Studenten bestehen, und ein Student kann an mehreren Kursen teilnehmen. Es ist darauf zu achten, dass alle Primärschlüssel nach wie vor genau ein Tupel identifizieren. Die Ausgangsrelationen STUDENT und KURS dürfen nicht verändert werden.

b) Formulieren Sie ein SQL-Statement, welches die Namen der Kursleiter alphabetisch sortiert ausgibt (keine Mehrfachnennung).

c) Formulieren Sie ausgehend von Ihrem in Teilaufgabe a) erweiterten Schema eine Anfrage in SQL, die die Namen und Leiter aller Kurse liefert, in denen Studenten mit Namen „Maier" teilnehmen.

d) Formulieren sie eine Anfrage in SQL, die die Namen und Geburtsdaten aller Kommilitonen liefert, die der Student mit der Matrikelnummer „12345678" in mindestens einem seiner Kurse außer sich selbst noch trifft? (keine Mehrfachnennungen)

e) Formulieren Sie eine SQL-Abfrage, die die Namen der 10 ältesten Studenten ausgibt, absteigend sortiert nach Alter. Gehen Sie davon aus, dass es keine zwei Studenten genau gleichen Alters gibt.

Fortsetzung nächste Seite!

Teilaufgabe 2:

1. Verifikation

Diese Frage befasst sich mit dem Begriff der Abstiegsfunktion zum Beweis der Terminierung von Rekursionen.

a) Begriffsdefinition. Definieren Sie den Begriff der Abstiegsfunktion durch die Angabe folgender Dinge:

   i. ihrer Signatur (d.h. ihrer Urbild- und Bildmenge),

   ii. ihrer Eigenschaften.

b) Beispiel. Geben Sie eine Abstiegsfunktion für die folgende Funktionsdefinition an:

```
even(x) = if x<0 then even (-x)
                 else if x==0 then True
                              else if x==1 then False
                                           else even (x-2)
```

Verfahren Sie dazu, wie folgt:

   i. Definieren Sie die Abstiegsfunktion durch Angabe von Signatur und Gleichungen.

   ii. Weisen Sie mathematisch nach, dass die Funktion die Anforderungen an eine Abstiegsfunktion erfüllt.

Fortsetzung nächste Seite!

2. Softwareentwicklung

Welche Phasen werden in jedem Prozess-Modell der Softwareentwicklung durchlaufen?
Nennen Sie zwei Vorteile und zwei Nachteile des Wasserfallmodells, jeweils mit Begründung!
Nennen Sie zwei Vorteile und zwei Nachteile der agilen Softwareentwicklung, jeweils mit Begründung!
Nennen Sie zwei Maßnahmen, mit denen agile Methoden versuchen, eine frühe Validierung zu erreichen. Erläutern Sie zusätzlich kurz, warum eine frühe Validierung von Vorteil ist!
Bewerten Sie folgende Aussage eines Projektmanagers: „Falls wir in Verzug geraten, dann können wir jederzeit neue Programmierer hinzuholen und die Deadline doch noch einhalten."
Nennen Sie Vorteile des Einsatzes von Versionsverwaltungssoftware!

Fortsetzung nächste Seite!

3. Softwarequalität

Nennen Sie jeweils einen Vorteil und einen Nachteil für Qualitätssicherung durch "Testing" bzw. durch "Model Checking".

Definieren Sie den Begriff "Refactoring".

Begründen Sie, warum bei der Entwicklung nach der Methode des "eXtreme Programming" langfristig gesehen Refactorings zwingend notwendig werden.

Wie wird in der Praxis während und nach erfolgtem Refactoring sichergestellt, dass keine neuen Defekte eingeführt werden bzw. wurden?

Worin besteht der Unterschied zwischen "White-Box-Testing" und "Black-Box-Testing"?

Nennen Sie vier Qualitätsmerkmale von Software.

Worin besteht der Unterschied zwischen funktionalen und nicht-funktionalen Anforderungen?

Was verbirgt sich hinter dem Begriff "Continuous Integration"?

Nennen Sie sechs Herausstellungsmerkmale des "eXtreme Programming" Ansatzes.

Was versteht man unter einem Unit-Test? Begründen Sie, warum es unzureichend ist, wenn eine Test-Suite ausschließlich Unit-Tests enthält.

Nennen Sie jeweils eine Methodik mit welcher in der Praxis die Prozesse der "Validierung" und der "Verifikation" durchgeführt werden.

Grenzen Sie die Begriffe "Fault" und "Failure" voneinander ab.

Fortsetzung nächste Seite!

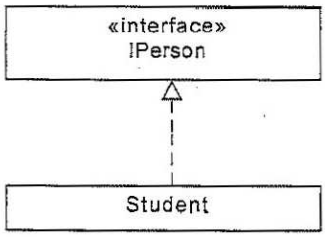
4. Objektorientierte Programmierung

Sie wollen in einer neuen Klasse Funktionalitäten einer anderen Klasse nutzen? Wann bietet sich Vererbung an, wann ist eine Komposition zu bevorzugen?

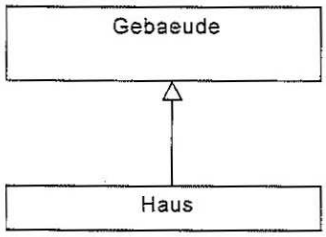
Was versteht man im Kontext der objektorientierten Programmierung unter dem Begriff "Kapselung". Nennen Sie zwei Vorteile die sich aus der konsequenten Einhaltung des Prinzips der Kapselung ergeben.

Geben Sie an, ob die folgenden UML-Diagramme die reale Welt korrekt modellieren. Begründen Sie Ihre Entscheidung kurz.

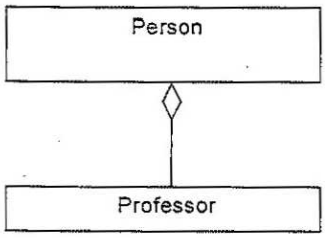
a)



b)



c)



Thema Nr. 2

Teilaufgabe 1:

1. ER-Modellierung

Entwerfen Sie ein ER-Modell für die folgende Miniwelt, die sich mit Konzerten, Veranstaltern, Bands usw. befasst. Geben Sie Kardinalitäten in (*min*, *max*)-Notation an. Sie brauchen Schlüsselattribute nicht zu kennzeichnen.

- Es gibt *Bands*, die jeweils einen Namen tragen. Zu jeder Band soll ein Gründungs- und ein Auflösungsdatum vermerkt werden.
- *Musiker* haben einen Namen. Musiker spielen *Instrumente* in Bands. Dabei können sie in verschiedenen Zeitintervallen (von-bis) verschiedene Instrumente in wechselnden Bands spielen, auch mehrere zur gleichen Zeit.
- *Instrumente* haben einen Namen.
- Bands spielen *Konzerte*. Jedes Konzert wird von genau einer Band gespielt, zusätzlich kann es noch bis zu zwei Vorgruppen geben. Konzerte finden zu einem bestimmten Datum an einem bestimmten Ort statt. Weitere Musiker, die nicht zur Band gehören, können bei Konzerten als Gast auftreten.
- *Veranstalter* haben einen Namen und veranstalten Konzerte. Jedes Konzert hat genau einen Veranstalter.
- *Besucher* gehen zu Konzerten. Ein Konzert hat mindestens 100 und höchstens 100.000 Besucher. Zu einem Konzertbesuch soll notiert werden, wie hoch der Eintrittspreis war. Dieser kann für jeden Besucher verschieden sein.

Ihr Modell sollte folgendes ausdrücken können. Sie brauchen diese Fakten nicht zu modellieren, sie dienen nur zur Kontrolle, ob Ihr Modell die o. g. Anforderungen erfüllt:

- Der Musiker Eddie van Halen spielte vom 1.1.1977 bis zum 6.6.1994 Gitarre und Keyboard in der Band "Van Halen"; vom 3.4.1986 bis zum 12.8.1988 spielte er Bass in der "Sammy Hagar Band".
- Am 3.6.2007 spielte die Band "Linkin Park" ein Konzert in Hamburg, das 7.000 Besucher hatte. Vorgruppe war "P.O.D.", als Gast trat Fred Durst auf. Das Konzert wurde veranstaltet von Undercover Entertainment.
- Der Besucher Karl Napf war bei diesem Konzert und hat dafür 53 Euro Eintritt gezahlt.
- Die Besucherin Liese Laut war auch beim o. g. Konzert, hat aber nur 48 Euro Eintritt gezahlt.

Fortsetzung nächste Seite!

2. Relationale Algebra

Entsprechend dem Modell aus der vorigen Aufgabe sei eine Miniwelt über Konzerte, Veranstalter usw. entsprechend den folgenden Relationen modelliert:

- Besucher(persnr, name)
- Konzerte(konzertnr, bandnr, datum, ort, veranstalternr)
- Veranstalter(veranstalternr, name)
- KonzertBesuche(persnr, konzertnr, preis)
- Vorgruppen(bandnr, konzertnr)
- Bands(bandnr, name, gründungsdatum, auflösungsdatum)
- Musiker(musikernr, name)
- Instrumente(instrumentnr, name)
- Spielt(musikernr, bandnr, instrumentnr, von, bis)
- Gast(musikernr, konzertnr)

Geben Sie je einen Ausdruck in relationaler Algebra für folgende Anfragen an:

2.1 In welchen Bands (Nummern sind gesucht) hat Musiker Nr. 12 gespielt?

2.2 Wer spielte schon (Musikernummern sind gesucht) in der Band "The Hooters"?

2.3 Welche Musiker (Nummern sind gesucht) waren schon einmal Gast bei einem Konzert der "Dave Matthews Band"?

Fortsetzung nächste Seite!

3. SQL

Für diese Aufgabe betrachten wir eine relationale Datenbank in Anlehnung an das Schema aus Aufgabe 2.

3.1 Geben Sie je ein `CREATE-TABLE`-Statement zum Erzeugen der Relationen *Bands* und *Spielt* an. Achten Sie auf den Primärschlüssel und auf möglichst restriktive Fremdschlüssel und Integritätsbedingungen:
Ein Musiker soll ein bestimmtes Instrument in einer gegebenen Band nur einmal spielen können (nicht mehrere Zeitintervalle).
Wenn ein Instrument verschwindet, soll das entsprechende Attribut in *Spielt* auf `NULL` gesetzt werden. Ein Musiker soll nicht gelöscht werden können, solange er noch in einer Band spielt. Wenn Bands gelöscht werden, sollen die entsprechenden Tupel in *Spielt* auch verschwinden.

3.2 Geben Sie SQL-Statements für folgende Anfragen an:

3.2.1 In welchen Bands (Nummern sind gesucht) hat Musiker Nr. 28 gespielt?

3.3 In welchen Bands (Nummer gesucht) hat der Musiker namens "Phil Collins" gespielt?

3.4 Erzeugen Sie eine Sicht, die eine Übersicht über die Veranstalter und die Anzahlen ihrer Konzerte enthält.
Die Sicht soll also Tupel der Form (*Veranstalternr, Veranstaltername, Anzahl Konzerte*) enthalten.

3.5 Geben Sie eine Anfrage an, die für jede Band den Namen der Band und die Gesamtzahl der jemals dort spielenden Musiker ausgibt. (Vorsicht, Musiker mit mehreren Instrumenten nicht mehrfach zählen.)

3.6 Was leistet folgende SQL-Anfrage:

```
select v.name
from konzerte k, konzertbesuche b, veranstalter v
where k.konzertnr = b.konzertnr
and v.veranstalternr = k.veranstalternr
group by v.veranstalternr
having sum(b.preis) >= all (
    select sum(bb.preis)
    from konzerte kk, konzertbesuche bb
    where kk.konzertnr = bb.konzertnr
    group by kk.veranstalternr
)
```

3.7 Ist die Unteranfrage in 3.6 korreliert oder nicht? (Begründung.)

Fortsetzung nächste Seite!

4. Normalisierung

Gegeben seien die funktionalen Abhängigkeiten

$$
\begin{aligned}
F = \{ \quad & ABC \rightarrow E, && \text{(i)} \\
& B \rightarrow AC, && \text{(ii)} \\
& E \rightarrow BCD, && \text{(iii)} \\
& F \rightarrow ABE, && \text{(iv)} \\
& D \rightarrow EF \quad \} && \text{(v)}
\end{aligned}
$$

der Relation $R(A, B, C, D, E, F)$.
Berechnen Sie eine kanonische Überdeckung F_C von F. Geben Sie alle Zwischenschritte und die angewendeten Transformationen an.

5. Normalisierung

Gegeben sei folgendes Schema, das die Bücher eines Verlages repräsentiert:

Bücher (ISBN, AutorNr, Autorname, Autoradresse, LektorNr, Lektorname, Titel, Seiten, Preis, Auflage, Jahr)

mit den funktionalen Abhängigkeiten

$$
\begin{aligned}
F \quad = \{ \quad & \text{AutorNr} \rightarrow \text{Autorname, Autoradresse}, && \text{(i)} \\
& \text{LektorNr} \rightarrow \text{Lektorname}, && \text{(ii)} \\
& \text{ISBN} \rightarrow \text{Titel, AutorNr, LektorNr}, && \text{(iii)} \\
& \text{ISBN, Auflage} \rightarrow \text{Seiten, Preis, Jahr} \quad \} && \text{(iv)}
\end{aligned}
$$

Hinweis: Sie können folgende Abkürzungen verwenden:

I	ISBN	T	Titel
A#	AutorNr	S	Seiten
An	Autorname	P	Preis
L#	Lektornr	J	Jahr
Ln	Lektorname	Au	Auflage
		Ad	Autoradresse

5.1 In welcher Normalform ist dieses Schema? (Begründung)

5.2 Nennen Sie je ein Beispiel für eine Einfüge-, Update- und Löschanomalie in diesem Schema.

Fortsetzung nächste Seite!

Teilaufgabe 2:

1. „OOA/OOD - UML“

Gegeben sei folgende Beschreibung einer Aufzuganlage in einem Gebäude:

„Eine Aufzuganlage besteht aus mindestens einem Aufzugschacht und genau einer Steuerung. Jeder Aufzugschacht hat genau eine Kabine, hat genau einen Motor und kennt alle Stockwerke, die er mittels der Kabine bedienen kann. Ein Stockwerk kann hierbei von mehreren Aufzugschächten bedient werden. Ein Stockwerk verfügt über eine eindeutige Nummer (z.B. 0, 1, 2, ...), eine Bezeichnung (z.B. „Tiefgarage“, „Keller“, „Erdgeschoss“, etc.) und Fahrtwünsche nach oben oder unten (beides gleichzeitig auch möglich). Eine Kabine kennt ihre Position (Stockwerk) und ihre Fahrtziele (Stockwerke). Sie kennt weiterhin ihren Türzustand (z.B. offen/geschlossen, true/false) und kann diesen verändern. Ein Motor kann mit einer Richtung (z.B. vorwärts/rückwärts, auf/ab, true/false) und einer Geschwindigkeit (z.B. 10 für 10cm/s) gestartet werden. Ein Motor kann jederzeit gestoppt werden. Ein Aufzugschacht erlaubt es, die aktuelle Position (Stockwerk) seiner Kabine und alle für ihn relevanten Fahrtziele (Stockwerke) (= Fahrtwünsche aller Stockwerke + Fahrtziele der Kabine) auszulesen. Er bedient weiterhin mit seiner Kabine selbstständig ein von der Steuerung übermitteltes Fahrtziel (Stockwerk). Die Steuerung wiederum bekommt bei ihrer Initialisierung von der Aufzuganlage alle Aufzugschächte übergeben, die sie verwalten muss.“

a) Identifizieren Sie in der obigen Beschreibung alle Substantive und Verben, die für die Modellierung des Systems mit UML relevant sind (z. B. mögliche Kandidaten für Klassen, Attribute, Assoziationen und Methoden bzw. Methodenparameter). Geben Sie jeweils an, welche davon Sie wofür geeignet halten und begründen Sie bei den anderen kurz, weshalb Sie sie eher verwerfen würden.

b) Erstellen Sie aus den von Ihnen vorangehend als geeignet identifizierten Kandidaten ein UML-Klassendiagramm einschließlich etwaiger Assoziationen zwischen den Klassen. Beschriften Sie die Assoziationsenden mit den richtigen Multiplizitäten.

Fortsetzung nächste Seite!

2. „Formale Verifikation"

Sie dürfen im Folgenden davon ausgehen, dass keinerlei Under- bzw. Overflows sowie keine Rundungsfehler auftreten.

Gegeben sei folgender Codeausschnitt in Java mit Vorbedingung (pre-condition) $P := d \geq 0$ und Nachbedingung (post-condition) $Q := |h - \sqrt{d}| < epsilon$:

```
public static double epsilon = 0.000001;

public static double sqrt(final double d) {
        // P
        double u = 0;
        double h = d > 1 ? d : 1;
        while ((h - u) / epsilon >= 1) {
                double m = (h + u) / 2;
                if (m * m > d)
                        h = m;
                else
                        u = m;
        }
        // Q
        return h;
}
```

Betrachten Sie dazu die folgenden fünf Prädikate:

1. $I_1 := u^2 \leq d = h^2$
2. $I_2 := d = h^2$
3. $I_3 := 0 \leq u \leq \sqrt{d} \leq h$
4. $I_4 := true$
5. $I_5 := u + 1 \leq h$

a) Welche der potentiellen Schleifeninvarianten $I_1, \ldots, I_5$ ist für den Beweis der (totalen) Korrektheit der Methode `sqrt` geeignet? Begründen Sie jeweils kurz, weshalb die anderen es nicht sind.

b) Weisen Sie *formal* (mittels *wp-Kalkül*) nach, dass die von Ihnen vorangehend gewählte Invariante unmittelbar vor Betreten der Schleife in `sqrt` gilt.

c) Beweisen Sie *formal*, dass dieselbe Invariante auch nach jedem Durchlaufen des Schleifenrumpfs in `sqrt` gilt.

d) Vervollständigen Sie den Beweis der partiellen Korrektheit, indem Sie formal nachweisen, dass die Nachbedingung Q aus Ihrer Invarianten folgt.

e) Geben Sie eine geeignete Terminierungsfunktion (Schleifenvariante) für `sqrt` an.

Fortsetzung nächste Seite!

3. Syntax und Semantik

a) Syntax. Definieren Sie eine Grammatik $G = (N, \Sigma, P, s)$ für die folgende Sprache:

$$L = \{a^i b^j \mid i, j \in \mathbb{N} \wedge i \geq j\}$$

Geben Sie alle Bestandteile von G an.

b) Semantik. Gegeben ist folgendes Programm in Java-Syntax:

```
public class ParameterPassing {
  public static int[] is = {    40,    41,    42 };
  //                         is[0], is[1], is[2]
  public static int r = 0;
  public static void callee(int s, int t) {
    r = -1;
    s =  0;
    t =  1;
    System.out.println("callee: " + is[0] + "/" + is[1] + "/" + is[2]);
  }
  public static void main(String[] arguments) {
    int s = r+1;
    callee(is[r+1], is[s+1]);
    System.out.println("main: " + is[0] + "/" + is[1] + "/" + is[2]);
  }
}
```

Geben Sie die Ausgaben des Programms in den folgenden beiden Fällen an:

i. Bei jedem Methodenaufruf wird *call by value-result* verwendet.

ii. Bei jedem Methodenaufruf wird *call by name* verwendet.