Einzelprüfungsnummer 66116
Datenbanksysteme / Softwaretechnologie
(vertieft)

Hermine Bschlangaul

Inhaltsverzeichnis

Frühjahr 2012 4

Herbst 2012 18

Frühjahr 2013 31

Herbst 2013 45

Frühjahr 2014 63

Herbst 2014 78

Frühjahr 2015 94

Herbst 2015 107

Frühjahr 2016 120

Herbst 2016 135

Frühjahr 2017 156

Herbst 2017 173

Frühjahr 2018 192

Herbst 2018 214

Frühjahr 2019 235

Herbst 2019 251

Frühjahr 2012

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer
Kennzahl: _____	**Frühjahr 2012**	**66116**
Kennwort: _____		
Arbeitsplatz-Nr.: _____		

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: **13**

Bitte wenden!

Thema Nr. 1

Teilaufgabe 1:

1. Entity-Relationship-Modell

Ein Handelsunternehmen möchte seine Struktur verbessern und ein Datenbanksystem zur Verwaltung seiner Filialen, angebotenen Waren und Kunden erstellen.
Die Basis dieses Systems bilden die Filialen des Unternehmens. Jede Filiale ist eindeutig durch ihre Filialnummer gekennzeichnet und befindet sich in einer Stadt. Außerdem hat jede Filiale einen Filialleiter.
Zu jeder Filiale gehört genau ein Lager mit einer eindeutigen Lagernummer und ebenfalls einem Leiter. Jedes Lager verfügt über eine bestimmte Menge an verschiedenen Waren. Jede Ware kann in mehreren Lagern vorrätig sein und ist über eine Nummer, einen Namen und einen Preis gekennzeichnet.
Ein Kunde kann in einer Filiale des Unternehmens Bestellungen aufgeben. Der Kunde hat eine Kundennummer, einen Namen und eine Adresse. Eine Bestellung enthält dabei jeweils einen Warenartikel, dessen gewünschte Menge und das Datum, an dem die Bestellung abgeholt wird.

a) Erstellen Sie ein Entity-Relationship-Diagramm für obige Datenbank.

b) Setzen Sie das in Teilaufgabe a) erstellte Entity-Relationship-Diagramm in ein Relationenschema um. Relationships sollen mit einer möglichst geringen Anzahl von Relationen realisiert werden. Dabei sind unnötige Redundanzen zu vermeiden. Ein Relationenschema ist in folgender Form anzugeben: `Relation (Attribut1, Attribut2, ...)`. Schlüsselattribute sind dabei zu unterstreichen. Achten Sie bei der Wahl des Schlüssels auf Eindeutigkeit und Minimalität.

2. Normalisierung

Gegeben sei folgende Datenbank für Wareneingänge eines Warenlagers. Die Primärschlüssel-Attribute sind unterstrichen.

ZulieferungsNr	ArtikelNr	Datum	Artikelname	Menge
1	1	01.01.2009	Handschuhe	5
1	2	01.01.2009	Mütze	10
2	3	05.01.2009	Schal	2
2	1	05.01.2009	Handschuhe	8
3	4	06.01.2009	Jacke	2

a) Erläutern Sie, inwiefern obiges Schema die 3. Normalform verletzt.

b) Geben Sie für obige Datenbank alle vollen funktionalen Abhängigkeiten (einschließlich der transitiven) an.

c) Überführen Sie das obige Relationenschema in die 3. Normalform. Erläutern Sie die dazu durchzuführenden Schritte jeweils kurz.

Fortsetzung nächste Seite!

3. SQL

Gegeben sei das folgende Relationenschema:

```
Fahrzeug(MNR:int(3), FZGNR:char(12), Baujahr:int(4),
         KMStand:int(5), Preis:int(5))

Modell(MNR:int(3), HNR:int(3), Typ:char(20), Neupreis:int(5),
       ps:int(3))

Hersteller(HNR:int(3), Name:char(20))
```

Dabei sind die Schlüsselattribute jeweils unterstrichen und zusätzlich für alle Attribute die Typen angegeben. Formulieren Sie die folgenden Anfragen bzw. Anweisungen in SQL.

a) Geben Sie die Anweisungen in SQL-DDL an, die notwendig sind, um die Relationen „Fahrzeug", „Modell" und „Hersteller" zu erzeugen. Achten Sie dabei darauf, die Primärschlüssel der Relationen zu kennzeichnen.

b) Bestimmen Sie die Typen aller Modelle des Herstellers mit Namen BMW.

c) Bestimmen Sie den Mindestpreis, bezogen auf das Attribut „Preis", der Fahrzeuge eines jeden Herstellers.

d) Bestimmen Sie die Namen der Hersteller, für die von jedem ihrer Modelle mindestens ein Fahrzeug in der Datenbank gespeichert ist.

e) Bestimmen Sie die Namen aller Hersteller, von denen mindestens fünf Fahrzeuge eines beliebigen Modells in der Datenbank gespeichert sind.

Fortsetzung nächste Seite!

Teilaufgabe 2:

Aufgabe 1

Welche Tätigkeiten sind in der Anforderungsanalyse für ein zu entwickelndes Softwaresystem durchzuführen? Welche Probleme können typischerweise in der Anforderungsanalyse auftreten?

Aufgabe 2

Gegeben sei das Klassendiagramm in Abb. 1.

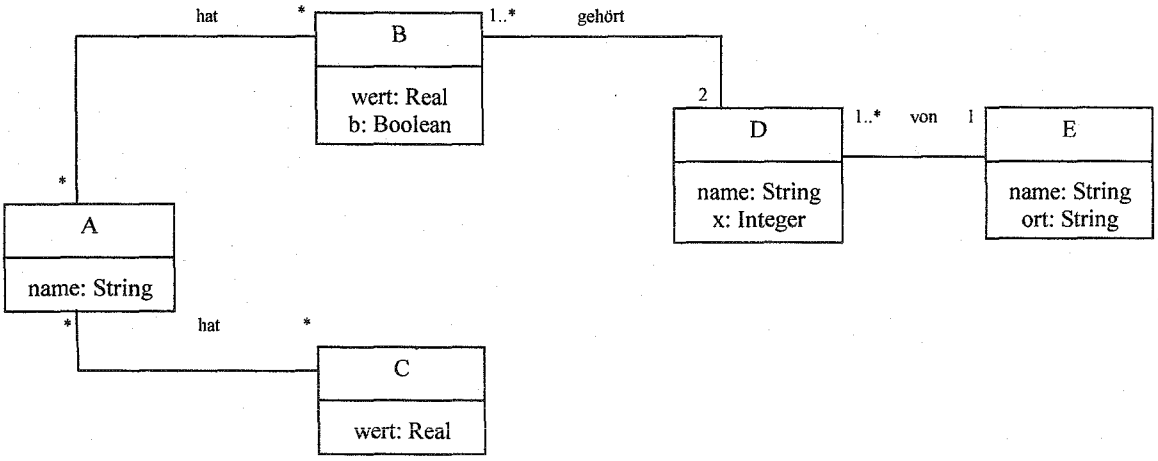


Abbildung 1: Klassendiagramm

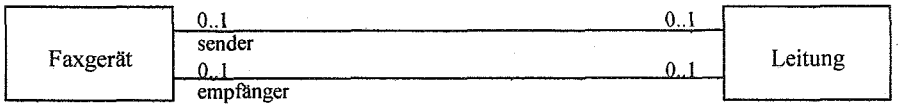
a) Konstruieren Sie ein (gemäß des Klassendiagramms zulässiges) Instanzendiagramm, das möglichst wenige Objekte aber mindestens ein Objekt der Klasse E enthält.

b) Gemeinsame Attribute und Assoziationen können in Oberklassen zusammengefasst werden. Konstruieren Sie auf diese Weise ein Klassendiagramm mit geeigneten Oberklassen und Vererbungsbeziehungen, das genau dieselben Instanzendiagramme zulässt wie das Klassendiagramm in Abb. 1. Falls sich dabei abstrakte Oberklassen ergeben, sind diese zu kennzeichnen.

Fortsetzung nächste Seite!

Aufgabe 3

Die (statistische) Struktur eines einfachen Faxübertragungssystems wird durch das Klassendiagramm in Abb. 2 beschrieben.



Das Sequenzdiagramm in Abb. 3 beschreibt ein Szenario, in dem ein Fax mit zwei Seiten erfolgreich übertragen wird.

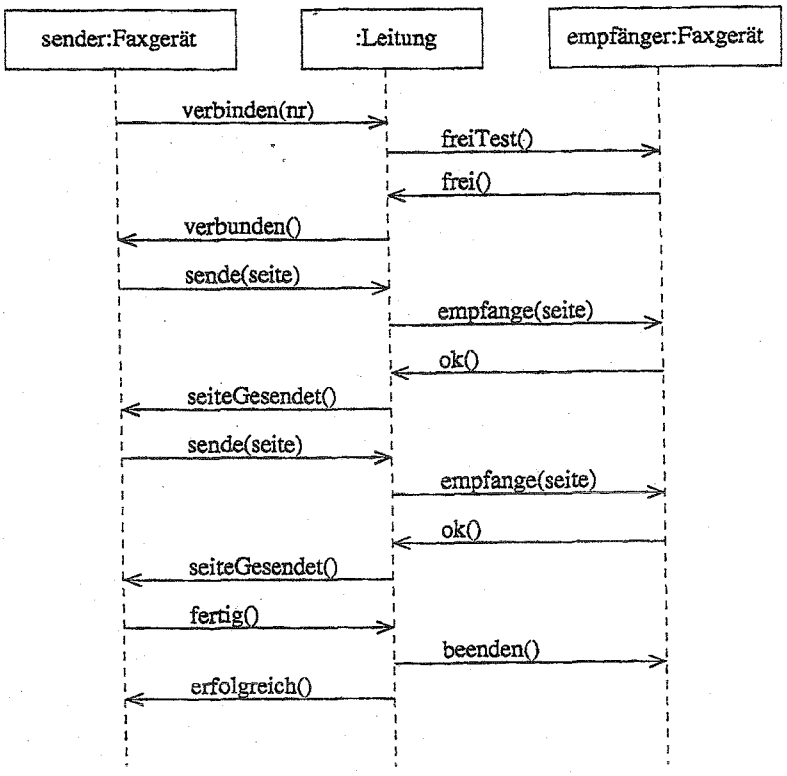


Abbildung 3: Sequenzdiagramm

a) Falls der Empfänger belegt ist oder falls eine Seite nicht richtig beim Empfänger angekommen ist, soll der Vorgang mit entsprechender Rückmeldung an den Sender abgebrochen werden. Erstellen Sie für jedes dieser beiden Szenarien ein Sequenzdiagramm.

b) Konstruieren Sie ein Zustandsdiagramm für die Klasse *Leitung*. Das Zustandsdiagramm soll das in den drei Szenarien (vgl. Abb. 3 und Teil a)) beschriebene Verhalten eines Leitungsobjekts integrieren, so dass beliebig viele Seiten übertragen werden können.

Fortsetzung nächste Seite!

Aufgabe 4

a) Was versteht man unter einem Entwurfsmuster bei der Entwicklung von Softwaresystemen? Erläutern Sie kurz die Vorteile von Entwurfsmustern.

Abb. 4 zeigt, in UML-Notation, das statische Modell des Observer-Patterns (aus dem Design-Pattern Katalog von Gamma et al.).

b) Erläutern Sie die Wirkungsweise des Observer-Patterns. Gehen Sie dabei auch auf die Aufgabe der einzelnen Operationen ein.

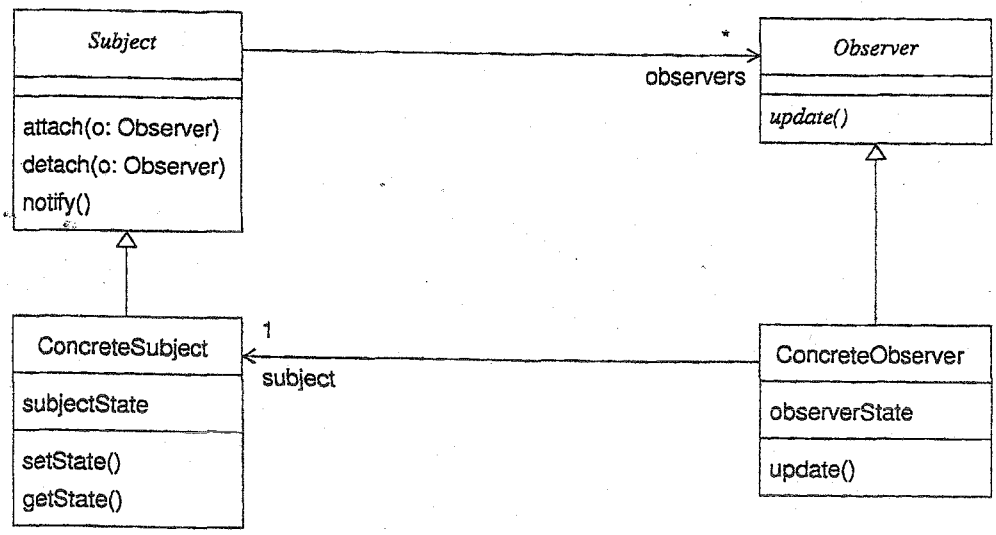


Abbildung 4: Observer-Pattern

c) Schildern Sie eine typische Anwendung des Observer-Patterns und erläutern Sie dessen Nutzen.

Thema Nr. 2

Teilaufgabe 1:

1. Datenbankanfragen in SQL

Wir betrachten die Datenbank FUSSBALL-EUROPAMEISTERSCHAFT mit den folgenden Tabellen:

- Spieler (Rückennummer, Land, Name, Vorname, Länderspiele, Verein, Land-V):
  Spielerinformation, einschließlich des Heimatlandes (Land), der Anzahl der Länderspiele, dem Verein und dem Land des Vereins (Land-V)
- Nationen(Land, Hauptstadt):
  Informationen zu den teilnehmenden Nationen
- nimmt_teil_an(Rückennummer, Land, Spiel_ID):
  Beteiligung eines Spielers an Spielen.
- Spiele(Spiel_ID, Land_Heim, Land_Gast, Datum, Stadion_ID, Tore_Heim, Tore_Gast):
  Informationen zu den Spielen, wobei Land_Heim und Land_Gast die beiden an einem Spiel beteiligten Nationen sind
- Stadion(Stadion_ID, Stadt, Land, Zuschauerkapazität):
  Informationen zu den Stadien

Erstellen Sie in SQL folgende Anfragen:

a) Bestimmen Sie alle teilnehmenden Spieler, die beim "FC Bayern München" (Verein) spielen.

b) Bestimmen Sie alle Spiele, die im Heimatland von einer der beiden beteiligten Nationen stattfinden.

c) An wie vielen Spielen nimmt "Bastian Schweinsteiger" teil?

d) Bestimmen Sie für jede Nation den Spieler mit den meisten Länderspielen. Dazu können Sie eine Unteranfrage oder eine Sicht (View) verwenden, die in der Hauptanfrage aufgerufen wird.

Fortsetzung nächste Seite!

2. Datenmodellierung und Datenbankaufbau

Wir betrachten die Datenbank FUSSBALL-EUROPAMEISTERSCHAFT aus Aufgabe 1.

a) Definieren Sie die Relationenschemata „Spieler" und „nimmt_teil_an" in SQL mittels CREATE TABLE und geben Sie dabei geeignete Schlüssel- und Fremdschlüsselbedingungen an. Nehmen Sie dabei an, dass die anderen Tabellen schon erzeugt sind.

b) Fügen Sie in SQL mittels INSERT in jede der beiden Relationen „Spieler" und „nimmt_teil_an" jeweils mindestens ein Tupel ein, so dass die natürlichen Schlüssel- und Fremdschlüsselbedingungen aus Teilaufgabe a) erfüllt sind. Eventuelle weitere Fremdschlüsselbedingungen auf andere Tabellen müssen nicht berücksichtigt werden.

c) Geben Sie ein ER-Diagramm mit Funktionalitätsbedingungen für das Datenbankschema an, in dem möglichst viele Relationen als Relationships modelliert sind.

3. Funktionale Abhängigkeiten

Gegeben sei das Relationenschema $R = (U, F)$ mit der Attributmenge

$$U = \{A,B,C,D,E,G\}$$

und der folgenden Menge F von funktionalen Abhängigkeiten:

$$F = \{A \rightarrow B,\ AB \rightarrow CD,\ CD \rightarrow E\}.$$

a) Bestimmen Sie die Attributhüllen $\{A\}_F^+$ und $\{C,D\}_F^+$.

b) Erklären Sie, warum die funktionale Abhängigkeit $CD \rightarrow E$ zu Update-Anomalien (d. h. beim Einfügen, Löschen und Ändern) führt.

c) Bestimmen Sie alle Schlüssel von R.

d) Bestimmen Sie eine Basis und eine 3NF-Zerlegung von R.

e) Zerlegen Sie R bezüglich der BCNF-verletzenden funktionalen Abhängigkeit $CD \rightarrow E$.

Fortsetzung nächste Seite!

Teilaufgabe 2:

1. Testen

Gegeben sei folgendes Modul, das einen ganzzahligen Wert *element* in die sortierte, verkettete Liste *list* einfügt. Gleiche Werte werden nur einmal aufgenommen. Der Wert *element* ist bereits im Knoten *newNode* gespeichert.

```
if (list.first == null) {
  list.first = newNode;
} else if (element < list.first.element) {
  newNode.next = list.first;
  list.first = newNode;
} else {
  actualNode = list.first;
  while (actualNode != null) {
    if (actualNode.element < element
        && actualNode.next == null) {
      actualNode.next = newNode;
    } else if (actualNode.element < element
              && element < actualNode.next.element) {
      newNode.next = actualNode.next;
      actualNode.next = newNode;
    }
    actualNode = actualNode.next;
  }
}
```

a) Bestimmen Sie anhand des Kontrollflussgraphen des obigen Code-Fragments die maximale Anzahl linear unabhängiger Programmpfade, also die zyklomatische Komplexität nach McCabe.

b) Kann für dieses Modul eine 100%-ige Pfadüberdeckung erzielt werden? (Grenzen aufgrund physikalischer Gegebenheiten sind dabei nicht zu berücksichtigen). Begründen Sie Ihre Antwort.

c) Geben Sie einen möglichst kleinen Testdatensatz an, der eine 100%-ige Verzweigungsüberdeckung dieses Moduls erzielt.

d) Beschreiben Sie kurz, welche Eigenschaften eine Testfallmenge allgemein haben muss, damit sie das datenflussorientierte Überdeckungskriterium „all-uses“ erfüllt.

e) Welche Anforderungen stellt das kontrollflussorientierte Testkriterium „boundary-interior“ („Schleife-Inneres-Überdeckung“) an den Tester?

f) Skizzieren Sie die drei klassischen Strategien beim Software-Integrationstest: „bottom-up“, „top-down“ und „sandwich“.

Fortsetzung nächste Seite!

2. Prozessmodelle

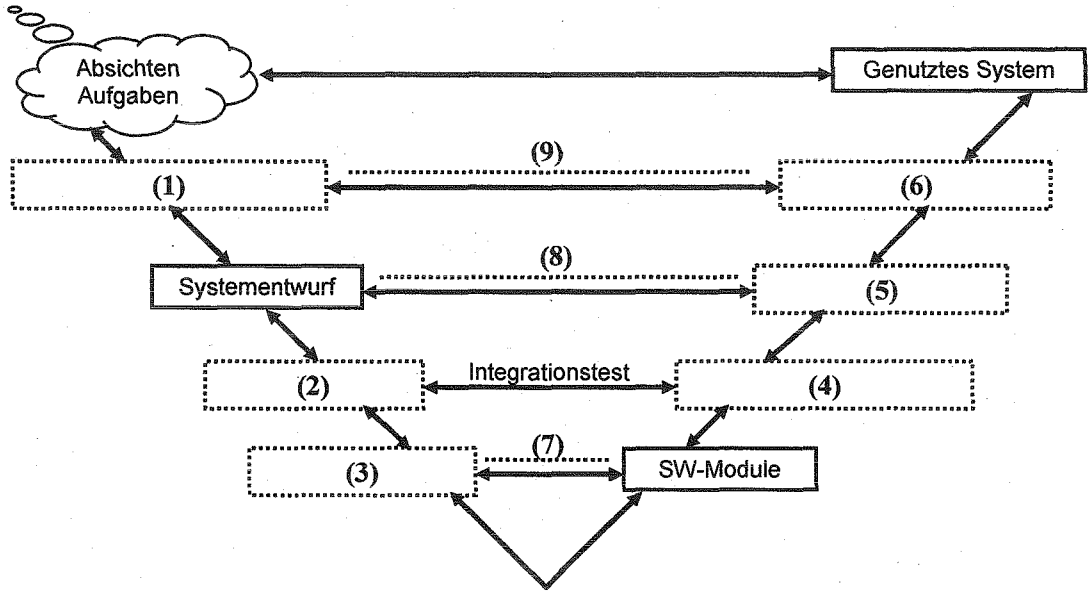
a) Um welche Aspekte erweitert das V-Modell 97 das „ältere“ Wasserfallmodell?

b) Welche grundlegende Phasenart unterscheidet das Spiralmodell von anderen vergleichbaren Modellen wie V-Modell oder Wasserfallmodell?

c) Beschreiben Sie kurz die folgenden vier Kategorien von Anforderungen und geben Sie jeweils ein entsprechendes kurzes Beispiel an: „funktional“, „qualitativ“, „systembezogen“, „prozessbezogen“.

d) Erläutern Sie knapp mit je einem Satz die folgenden vier klassischen Wartungsanlässe und skizzieren Sie dabei jeweils den Unterschied zu den anderen: „corrective“, „adaptive“, „perfective“, „preventive“.

e) Nennen Sie die fehlenden Angaben (Zwischenprodukte und Testarten) - jeweils mit ihrer Nummer - im folgenden, am V-Modell 97 angelehnten Prozessmodell:



Fortsetzung nächste Seite!

3. Formale Verifikation

Gegeben sei das folgende Programmfragment „**MinMax**" mit zwei ganzzahligen Parametern **x** und **y**:

```
int h;
if (x > y) {
    h = x;
    x = y;
    y = h;
}
```

Unter Verwendung des Hoare-Kalküls weisen Sie folgende Aussage nach:

$\{x, y \in N\}$ **MinMax** $\{x \leq y\}$

Hinweise:

Zuweisungsaxiom: $\{q \mid l \text{ ersetzt durch } r\}\ [\ l := r\ ]\ \{q\}$

Bedingungsaxiom: $\{p \wedge B\}\ S1\ \{q\};\ \{p \wedge \neg B\}\ S2\ \{q\} \Rightarrow \{p\}$ if B then S1 else S2 $\{q\}$

4. UML - OOA

Gegeben sei folgende natürlich-sprachliche Spezifikation eines PKI-Systems (Public-Key-Infrastruktur):

Damit Lehrer Anton seiner Kollegin Berta die Aufgaben der Klausur verschlüsselt per eMail übermitteln kann, bedarf es einer entsprechenden Infrastruktur.
Nachdem beide Lehrer die notwendige Software installiert haben, erstellt Berta zunächst ein sogenanntes Schlüsselpaar, bestehend aus einem „öffentlichen" (ÖS) und einem zugehörigen „privaten" Schlüssel (PS). Anschließend veröffentlicht Berta ihren ÖS durch Hochladen auf einen sogenannten Schlüsselverzeichnisdienst (Keyserver). Damit steht er Anton zur Verfügung, so dass dieser den ÖS jederzeit vom Keyserver herunterladen kann.
Anton kann nun seine Nachricht (z. B. das Aufgabenblatt) mit dem ÖS von Berta verschlüsseln und mit einer eMail versenden. Empfängt Berta eine solche eMail, so kann sie (und nur sie) mit ihrem PS die Nachricht wieder entschlüsseln.
Umgekehrt kann Berta mit ihrem PS beliebige Informationen (z. B. die Notenliste) „digital signieren". Diese unterschriebenen Daten übermittelt Berta dann zusammen mit der digitalen Signatur per eMail an Anton. Anton kann mit dem ÖS von Berta prüfen, ob die übermittelte Nachricht unverändert und tatsächlich von Berta stammt.

Fortsetzung nächste Seite!

a) Modellieren Sie die in der Spezifikation beschriebenen Anwendungsfälle zusammen mit den jeweils beteiligten Aktoren in einem Use-Case-Diagramm. Betrachten Sie den *Keyserver* zunächst ebenfalls als Aktor, welcher gegenüber PKI als „externer Vermittler" auftritt.

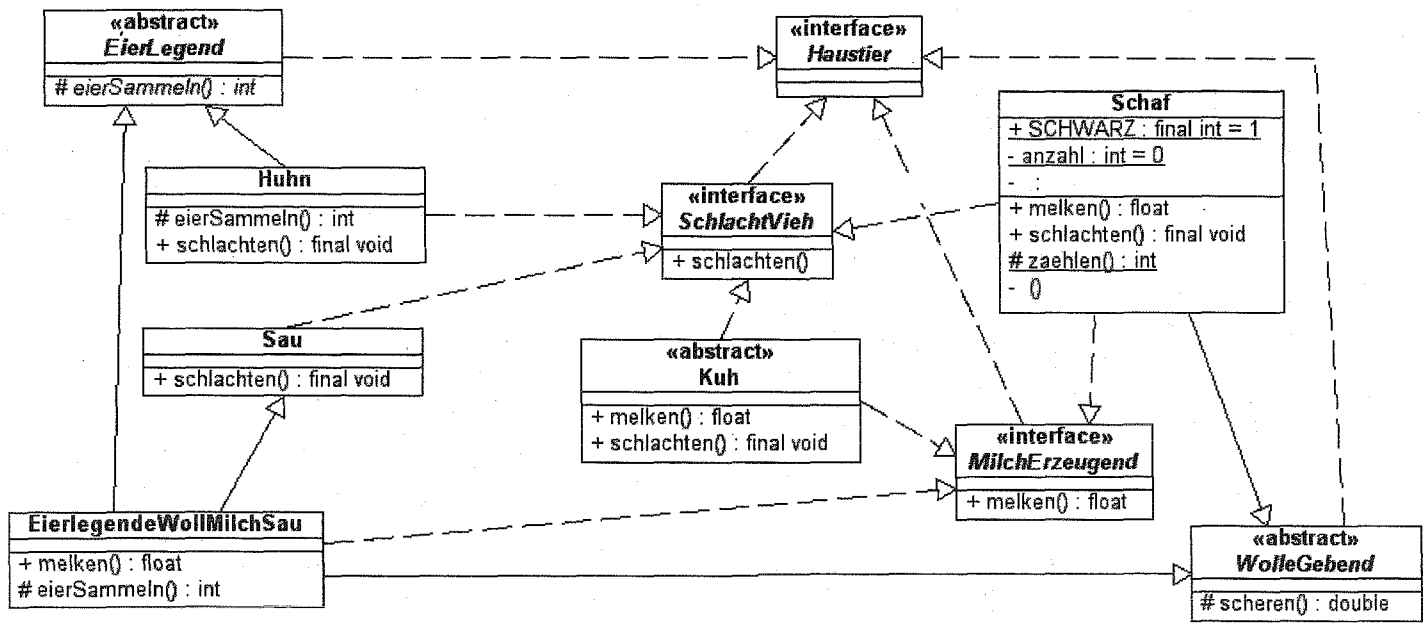
b) Erstellen Sie das Klassendiagramm für die von Ihnen identifizierten Klassen. Berücksichtigen Sie zusätzlich zur verbalen Spezifikation noch folgende Präzisierungen:

1. Es gibt genau einen *Keyserver*; dieser verwaltet aber beliebig viele Schlüssel. Er bietet die entsprechenden Dienste zum Veröffentlichen bzw. Abfragen von Schlüsseln an.
2. Der Schlüssel wird mit der eMail-Adresse von Berta „benannt", damit Anton den richtigen Schlüssel abrufen kann.
3. Jeder Lehrer hat höchstens ein Schlüsselpaar, aber jeder Schlüssel gehört genau einem Lehrer. Anton hingegen kennt außer Berta auch Christian und Dora, denen er auch verschlüsselte Nachrichten schicken möchte.
4. Eine Nachricht kann (muss aber nicht) eine Signatur als „Anhang" zusätzlich zum eigentlichen Inhalt mit sich führen.
5. Für das Signieren bzw. Entschlüsseln ist der PS zuständig. Dafür bekommt er den Inhalt der Nachricht und gibt entsprechend eine Signatur bzw. den entschlüsselten Inhalt zurück.
6. Für das Prüfen der Signatur und das Verschlüsseln ist der ÖS zuständig. Dazu bekommen die Methoden je nach Bedarf den Inhalt der Nachricht und die Signatur und liefern einen Wahrheitswert bzw. den verschlüsselten Inhalt zurück.

Fortsetzung nächste Seite!

5. UML - OOD

Gegeben sei folgendes Klassendiagramm eines Moduls „**bauernhof**“:



a) Geben Sie den Code der Klassen **`EierLegend`** und **`Schaf`** aus dem obigen Klassendiagramm in einer objekt-orientierten Sprache Ihrer Wahl an und annotieren Sie ihn mit verständlichen Kommentaren. Etwaige Methodenrümpfe können Sie leer lassen, auch wenn dies zu einem Compiler-Fehler führen würde. Beachten Sie die Sichtbarkeiten (**#** = **`protected`**) und dass **`SCHWARZ`** sowie **`schlachten()`** *final* sind!

b) Warum kann es die Klasse **`EierlegendeWollMilchSau`** in Java nicht geben? Beschreiben Sie kurz aber vollständig, welche minimalen Änderungen Sie an der obigen Architektur vornehmen würden, damit die Klasse **`EierlegendeWollMilchSau`** in Java doch implementiert werden kann, sie aber weiterhin eine Unterklasse von **`Sau`** bleibt.

c) Stellen Sie sich eine Klasse Bauer mit einer Methode **`alltag()`** vor:
In dieser Methode werden zunächst je zwei Instanzen der Klassen **`Huhn`** bzw. **`Schaf`** erstellt. Danach muss der Bauer bei seinen Hühnern die **`eierSammeln()`**, das erste Schaf **`scheren()`** und das zweite **`melken()`**. Schließlich erstellt er sich noch eine Instanz von **`Sau`**, die er gleich im Anschluss **`schlachten()`** kann.
Mit welcher UML-Diagrammart würden Sie das obige Szenario modellieren? Warum halten Sie diese Diagrammart für besonders geeignet?

d) Modellieren Sie die Methode **`alltag()`** in der vorangehend gewählten Diagrammart Ihrer Wahl.

Herbst 2012

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer
Kennzahl: _____	**Herbst 2012**	**66116**
Kennwort: _____		
Arbeitsplatz-Nr.: _____		

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: **12**

Bitte wenden!

Thema Nr. 1

Teilaufgabe 1:

1. Modellierung

Sie sollen ein System zur Verwaltung von Einkaufszentren entwerfen.
Gehen Sie dabei von folgendem Szenario aus:

- Einkaufszentren haben einen eindeutigen Namen und eine Adresse. Sie bestehen aus mehreren Etagen. Jede Etage hat eine Nummer (z. B. „2. Etage"). Diese ist jedoch nur innerhalb eines Einkaufszentrums eindeutig. In jeder Etage gibt es Räume. Räume werden durch eine Raumnummer identifiziert, die auf einer Etage eindeutig ist (z. B. „Raum 7 der 2. Etage"). Räume haben eine Fläche. Außerdem können Räume neben anderen Räumen liegen. Räume, die nebeneinander liegen, befinden sich in demselben Einkaufszentrum in derselben Etage.
- Ein Einkaufszentrum hat Läden. Der Name eines Ladens ist innerhalb eines Einkaufszentrums eindeutig. Für jeden Laden wird die Verkaufsfläche gespeichert. Jeder Laden belegt ein oder mehrere Räume des Einkaufszentrums, in dem er sich befindet. Es gibt keine Läden, die sich einen Raum teilen.
- Es gibt Warengruppen. Diese haben einen eindeutigen Namen (z. B. „Handtaschen" oder „Schuhe"). Läden können einer beliebigen Anzahl Warengruppen zugeordnet werden.
- Für das Personal, welches sich in Verkäufer und Reinigungskräfte aufteilt, werden ihre eindeutige Personalnummer, ihr Name und ihre Adresse gespeichert. Verkäufer können Auszeichnungen erhalten (z.B. „Verkäufer des Monats November 2010"). Für jede Auszeichnung werden Name und Datum angegeben (z. B. „Verkäufer des Monats" und „November 2010"). Es kann vorkommen, dass sich mehrere Verkäufer eine Auszeichung teilen müssen.
- Läden beschäftigen Personal, d. h. Verkäufer und Reinigungskräfte. Einkaufszentren beschäftigen nur Reinigungskräfte, aber keine Verkäufer.

a) Entwerfen Sie für das beschriebene Szenario ein ER-Modell.
Bestimmen Sie hierzu:

- die Entity-Typen, die Relationship-Typen und jeweils deren Attribute,
- ein passendes ER-Diagramm,
- die Primärschlüssel der Entity-Typen, welche Sie anschließend in das ER-Diagramm eintragen, und
- die Funktionalitäten der Relationship-Typen, welche Sie ebenfalls in das ER-Diagramm eintragen.

b) Überführen Sie das ER-Modell aus Aufgabe a) in ein verfeinertes relationales Modell. Geben Sie hierfür die verallgemeinerten Relationenschemata an. Achten Sie dabei insbesondere darauf, dass die Relationenschemata keine redundanten Attribute enthalten.

Fortsetzung nächste Seite!

2. Anfragen

Gegeben seien die folgenden Relationen:

Menschen : {[MenschId, Name, Alter]}

Berufe : {[BerufId, Name]}

HatBeruf : {[MenschId, BerufId]} — *Ein Mensch kann mehrere Berufe haben.*

FanVon : {[IdolId, FanId]} — *Ein Idol kann mehrere Fans haben.*
Idole und Fans sind Menschen.

a) Formulieren Sie eine SQL-Anfrage, die *MenschId* und *Name* des Rockmusikers (bzw. der Rockmusiker) mit den meisten Fans zurückgibt.
Hinweis: Sie dürfen eigene Views definieren.

b) Geben Sie einen Ausdruck in relationaler Algebra an, der *MenschId* und *Name* der Fußballspieler zurückliefert, die weniger als 20 Jahre alt sind und mindestens einen Fan haben, der älter ist als 60 Jahre.

c) Formulieren Sie einen Ausdruck im relationalen Tupelkalkül, der *BerufId* und *Name* derjenigen Berufe zurückgibt, bei denen *jeder,* der sie ausübt, älter ist als 40 Jahre.
Hinweis: Der Ausdruck darf auch unsicher sein.

d) Geben Sie einen Ausdruck im relationalen Domänenkalkül an, der alle Paare von *MenschIds* von Menschen zurückliefert, die gegenseitig voneinander Fans sind. Der *Name* des Menschen, dessen *MenschId* zuerst genannt wird, soll dabei (als String) kleiner oder gleich dem Namen des Menschen sein, dessen *MenschId* als zweites genannt wird.
Hinweis: Der Ausdruck darf auch unsicher sein.

e) Formulieren Sie eine Datalog-Anfrage, die *MenschId* und *Name* von allen Menschen zurückgibt, die Fans von einem Boxer und von einem Ballettänzer sind.

Fortsetzung nächste Seite!

3. Normalformen und Anomalien

Gegeben sei die Relation R mit Schema R: $\{[A, B, C, D, E]\}$.

Für die Relation R gelten die folgenden Funktionalen Abhängigkeiten:

- $A \rightarrow B$
- $A, C \rightarrow D$
- $B \rightarrow E$

a) Bestimmen Sie alle Schlüsselkandidaten von R.

b) Prüfen Sie, ob R in 1NF, 2NF, 3NF bzw. BCNF ist. Begründen Sie Ihre Antwort.

c) Zeigen Sie an einer geeigneten Instanz, dass in der Relation R eine Update-Anomalie auftreten kann.
Hinweis: Beachten Sie, dass Ihre Instanz alle Funktionalen Abhängigkeiten aus der Aufgabenstellung erfüllt.

d) Betrachten Sie nacheinander die Normalformen 1NF, 2NF, 3NF und BCNF. Gibt es eine Relation, die sich in der betrachteten Normalform befindet und in der eine Update-Anomalie auftreten kann (ohne Begründung)?

Fortsetzung nächste Seite!

Teilaufgabe 2:

Sie sollen für eine Online-Apotheke eine Software zur Verwaltung des Medikamentenbestandes implementieren. Zu jedem Medikament gehört eine Nummer, ein Markenname, ein Hersteller, ein Lieferant, ein Verfallsdatum, ein Einzelpreis und sein Lagerbestand (Anzahl der Einheiten). Dasselbe Medikament kann mit unterschiedlichen Verfallsdaten vorhanden sein. Dazu kommt die Zusatzinformation, etwa ob und in welchem Umfang es vorbestellt wurde.
Es soll möglich sein, den Lagerbestand aufzufüllen und zu reduzieren (zum Beispiel beim Verkauf eines Medikaments). Es soll möglich sein, alle Medikamente, deren Verfallsdatum überschritten ist, zu entfernen. Außerdem soll es möglich sein, Aufträge an Lieferanten zu erstellen.
Die beteiligten Personen können Kunden, Apotheker oder Administratoren sein; der Einfachheit halber können Apotheker und Administratoren nicht Kunden sein.
Beteiligte Personen können sich beim System an- und wieder abmelden.
Kunden können Medikamente kaufen und bezahlen. Außerdem können Kunden den Bestand durchsuchen und erhalten dann für jedes geführte Medikament die Mitteilung, ob es verfügbar ist, oder nicht. In letzterem Falle können sie es vorbestellen. Kunden können aber nicht Stückzahlen oder Verfallsdaten einsehen.
Apotheker können Medikamente verkaufen, den Bestand durchsuchen, verändern und Aufträge an Lieferanten tätigen. Außerdem können sie Kunden neu aufnehmen oder löschen.
Administratoren haben alle Befugnisse von Apothekern und können außerdem Apotheker und Administratoren aufnehmen oder löschen.

a) Entwickeln Sie aus der gegebenen Anwendungsbeschreibung ein UML-Klassendiagramm. Alle Angaben aus dem Text sollen sich in Ihrem Modell wiederfinden; weitere sollten nur dann hinzugenommen werden, wenn es aufgrund der von Ihnen gewählten Modellierung erforderlich erscheint.
Verwenden Sie Vererbung und Interfaces, wo es sinnvoll möglich ist und achten Sie auf eine angemessene Darstellung von Assoziationen und Aggregationen, wo erforderlich. Es geht hier nur um das UML-Klassendiagramm; insbesondere ist eine Modellierung als relationale Datenbank *nicht* verlangt und auch nicht im Sinne der Aufgabenstellung.

b) Modellieren Sie den gesamten Anwendungsfall des Kaufens eines Medikaments (Anmelden, Bestand anzeigen, Aussuchen, Kaufen, Abmelden) zunächst als UML-Anwendungsfall und dann als UML-Aktivitätsdiagramm.
Achten Sie auch auf die Möglichkeit des Fehlschlagens einer Teilaktivität.

Fortsetzung nächste Seite!

Aufgabe 2:

Das Entwurfsmuster *Beobachter* (Observer) kommt zum Einsatz, wenn auf Änderungen einer Datenstruktur an verschiedenen Stellen im Programm reagiert werden soll.

a) Beschreiben Sie eine mögliche konkrete Anwendung dieses Musters.

b) Geben Sie ein allgemeines UML-Klassendiagramm für dieses Entwurfsmuster an.

c) Skizzieren Sie, wie die beteiligten Klassen und Interfaces in Java oder einer ähnlichen Sprache implementiert werden können.

d) Nennen Sie zwei weitere Entwurfsmuster und beschreiben Sie ihre Einsatzmöglichkeiten in jeweils ca. drei Sätzen.

Thema Nr. 2

Teilaufgabe 1:

1 Allgemeinwissen

1.1 Erläutern Sie kurz die folgenden Begriffe im Kontext von föderierten Datenbanksystemen.

a) Komponentenschema
b) Lokales Schema
c) Föderiertes Schema
d) Externes Schema
e) Exportschema

2 Relationale Algebra
2.1 Nennen Sie die fünf Grundoperationen der relationalen Algebra.
2.2 Warum wird in gewöhnlichen Datenbanken die relationale Algebra nicht strikt eingehalten? Geben Sie ein Beispiel an und erklären Sie wie man die relationale Eigenschaft erzwingen kann.

3 ER-Modellierung und Relationenmodell
3.1 Nennen und erläutern Sie kurz vier Beispiele für Integritätsbedingungen.
3.2 Sie sollen die Verwaltung von TÜV-Stützpunkten modellieren. Erstellen Sie ein ER-Diagramm bestehend aus Entitäts- und Beziehungstypen sowie Attributen. Geben Sie auch die Kardinalitäten mit an.
Verwenden Sie bei Entitäten und Attributen ausschließlich die beschriebenen, soweit dies möglich ist. Es ist ebenfalls nicht erlaubt künstliche Schlüssel zu erfinden. Die Kardinalitätseinschränkungen können Sie entweder in der (min,max)-Notation oder der einfachen Notation nach Chen (1:1, 1:N, M:N) angeben.

In der Formulierung der Beziehungen sind die Kardinalitätseinschränkungen genau zu beachten. Im Allgemeinen gilt: alles was nicht explizit eingeschränkt ist, muss im Modell auch uneingeschränkt bleiben. (Keine Einschränkungen in eine Beschreibung hineininterpretieren). Alles was explizit eingeschränkt ist, darf im Modell nicht uneingeschränkt bleiben. (Alle formulierten Einschränkungen sollen erkennbar sein).

1. Es gibt TÜV-Stationen. Diese sind an einer Adresse ansässig und besitzen einen Namen. Eine TÜV-Station ist offiziell unter einer eindeutigen ID registriert.
2. Jede TÜV-Station ist in Abteilungen organisiert. Eine Abteilung hat einen Namen und eine Spezialisierung (z.B. "Bremsen", "Flüssigkeiten", etc.). Eine Abteilung soll durch einen künstlichen Schlüssel (ID) eindeutig identifiziert werden.
3. Es gibt Angestellte, die über ihre Sozialversicherungsnummer (SVN) eindeutig identifiziert sind. Angestellte haben einen Vor- und einen Nachnamen. Außerdem sind sie irgendwann geboren und haben ein Geschlecht. Darüber hinaus ist ihnen ein Gehalt sehr wichtig.
4. Prüfingenieure besitzen eine eindeutige Lizenznummer. In der Regel haben sie ein Spezialisierungsfach. Darüber hinaus können sie eine Zusatzqualifikation (Sicherheitsingenieur etc.) besitzen.

Fortsetzung nächste Seite!

5. Prüfingenieure sind Angestellte: Ein Angestellter kann ein Prüfingenieur sein; ein Prüfingenieur ist stets ein Angestellter.
6. Eine Abteilung beschäftigt beliebig viele Angestellte. Ein Angestellter kann in einer oder mehreren Abteilungen arbeiten.
7. Jeder Prüfingenieur ist in genau einer TÜV-Station tätig. Eine Station kann mehrere Prüfingenieure beschäftigen.
8. Keine TÜV-Station ohne Fahrzeuge: Sie werden durch die Fahrgestellnummer identifiziert und es werden weitere Daten wie Hersteller, Typ und Baujahr gespeichert.
9. Ein Prüfingenieur kann mehrere Fahrzeuge untersuchen.
10. Eine TÜV-Station muss mindestens eine Abteilung haben. Jede Abteilung gehört zu genau einer TÜV-Station.
11. "Gebühren": Prüfungsleistungen werden in Rechnung gestellt. Zu jeder Leistung speichern wir die Beschreibung der Leistung und den jeweiligen Preis. Identifiziert werden sie durch Leistungsnummer-Codes (nach DIN). Rechnungen haben eine eindeutige Rechnungsnummer. Jede Rechnung umfasst mindestens eine Leistung.
12. Angestellte leiten andere Angestellte
13. Einem Chef können andere Angestellte untergeben sein.
14. Jeder Angestellte ist Untergebener von maximal einem Angestellten.

3.3 Erstellen Sie zu dem E/R-Diagramm aus Aufg. 3.2 ein möglichst vollständiges Relationenschema – bspw. Berücksichtigung totaler Partizipationen. Vermeiden Sie unnötiges Ausprägen von Relationen bei allen Beziehungen.

Beispiel für die Notation:

```
Relationenname (Primärschlüssel, Attribut1, Attribut2, …,
Fremdschlüssel[AndereRelation],(FKAttrA,FKAttrB)[DritteRelation])
Attribut2 NOT NULL
```

4 Normalformen

4.1 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.

4.2 In welcher Normalform ist das folgende Beispiel? Zeigen Sie, dass alle Bedingungen für diese Normalform erfüllt sind. Welche Bedingung der nächsthöheren Normalform ist verletzt? Berücksichtigen Sie, dass eine Relation nur dann in n-ter Formalform ist, wenn sie die Bedingungen aller m-ten Normalformen mit $m \leq n$ erfüllt.

AbtNr	Name	Strasse	Ort	PLZ	Vorwahl
1	Entwicklung	Abbestr.	München	80999	089
2	Forschung	Allacher Str.	München	80997	089
3	Entwicklung	Zieglhof	Regensburg	93055	0941
4	Fertigung	Bucher Hauptstr.	Nürnberg	90427	0911
5	Logistik	Osserweg	Passau	94034	0851
6	Vertrieb	Alfred-Nobel-Str.	Würzburg	97080	0931

Fortsetzung nächste Seite!

5 Schlüsselzugriffe

5.1 Wozu ist die Möglichkeit des Satzzugriffes über Schlüssel wünschenswert?

5.2 Grenzen Sie die Begriffe Primärorganisation und Sekundärorganisation voneinander ab.

5.3 Gegeben ein leerer B-Baum (maximale Anzahl der Einträge pro Knoten ist 2k, k=1). Die Zahlen 13, 17, 21, 11, 5, 15 und 12 sind in dieser vorgegebenen Reihenfolge in den B-Baum einzufügen. Löschen Sie dann den Schlüssel 15. Zeichnen Sie die Zustände des B-Baums nach jeder Strukturänderung. Falls sich durch eine Operation die Struktur des Baumes nicht ändert, muss er nicht neu gezeichnet werden, sondern der einzufügende Wert kann einfach in den Baum eingetragen werden. Wenn eine Operation mehrere Strukturänderungen erfordert, so ist der Baum nach jeder Änderung neu zu zeichnen.

5.4 Nennen Sie zwei Unterschiede zwischen B-Baum und B*-Baum sowie je einen Vor- und Nachteil beider Varianten.

Fortsetzung nächste Seite!

Teilaufgabe 2:

1 Objektorientierung

Ein Getränkelieferservice verwaltet die Bestellungen verschiedener Kunden. Die folgenden Teilaufgaben sind in einer objektorientierten Programmiersprache zu lösen (die verwendete Sprache ist vorab anzugeben.).

1. Implementieren Sie eine Klasse Kasten zur Beschreibung eines Getränkekastens mit den folgenden Eigenschaften. Entscheiden Sie dabei jeweils ob eine Realisierung als Objekt- oder Klassenfeld sinnvoll ist.

   - Es existiert ein einheitliches Kastenpfand in Höhe von 1,50 Euro.
   - Für alle Flaschen in einem Kasten gelte ein einheitliches Flaschenpfand, das jedoch von Kasten zu Kasten verschieden sein kann.
   - Während das Flaschenpfand für alle Flaschen eines Kastens gleich ist, sind die Einzelpreise der Flaschen je nach Inhalt unterschiedlich. Die Einzelpreise (ohne Flaschenpfand) der im Kasten enthaltenen Flaschen sollen in einem 2-dimensionalen Array abgelegt werden.

   Geben Sie für die Klasse Kasten einen geeigneten Konstruktor an.
   Ergänzen Sie in der Klasse Kasten eine Objektmethode zur Berechnung des Gesamtpreises des Getränkekastens inklusive Kasten- und Flaschenpfand. (12 Punkte)

2. Schreiben Sie eine Klasse Bestellung. Jeder Bestellung soll eine eindeutige Bestellnummer zugeordnet werden, die über den Konstruktoraufruf erstellt wird. Außerdem soll zu jeder Bestellung der Name des Kunden gespeichert werden, sowie eine einfach verkettete Liste der bestellten Getränkekästen. Die Klasse Bestellung soll weiterhin eine Methode beinhalten, die den Gesamtpreis der Bestellung ermittelt. (10 Punkte)

3. Schreiben Sie ein kleines Testprogramm, das eine Bestellung erstellt, die zwei Getränkekästen umfasst. Der erste Kasten soll ein 1 x 1 Getränkekasten mit einer Flasche zu 0,75 Euro sein, der zweite Kasten soll - wie in Abbildung 1 dargestellt - ein 3 x 3 Getränkekasten mit 3 Flaschen zu 0,7 Euro auf der Diagonalen und 3 weiteren Flaschen zu je 1 Euro sein. Das Flaschenpfand beider Kästen beträgt 0,15 Euro pro Flasche, das Kastenpfand 1,50 Euro. Anschließend soll der Preis der Bestellung berechnet und auf der Standardausgabe ausgegeben werden. (8 Punkte)

1,0	1,0	0,7
1,0	0,7	0
0,7	0	0

Abbildung 1: Belegung des zweiten Getränkekastens aus Teilaufgabe (3)

Fortsetzung nächste Seite!

2 Projektmanagement

Abbildung 2 stellt ein CPM-Netzwerk dar. Die Ereignisse sind fortlaufend nummeriert (Nummer im Inneren der Kreise) und tragen keine Namen. Gestrichelte Linien stellen Pseudo-Aktivitäten mit einer Dauer von 0 dar.

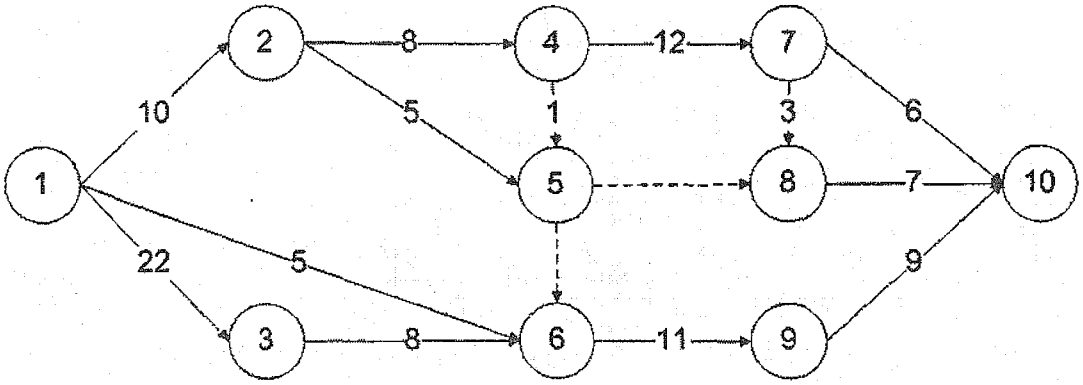


Abbildung 2: CPM-Netzwerk.

1. Berechnen Sie die früheste Zeit für jedes Ereignis, wobei angenommen wird, dass das Projekt zum Zeitpunkt 0 startet! (5 Punkte)
2. Setzen Sie anschließend beim letzten Ereignis die späteste Zeit gleich der frühesten Zeit und berechnen Sie die spätesten Zeiten! (5 Punkte)
3. Berechnen Sie nun für jedes Ereignis die Pufferzeiten! (5 Punkte)
4. Bestimmen Sie den kritischen Pfad! (2 Punkte)
5. Konvertieren Sie das Gantt-Diagramm aus Abbildung 3 in ein CPM-Netzwerk! (10 Punkte)

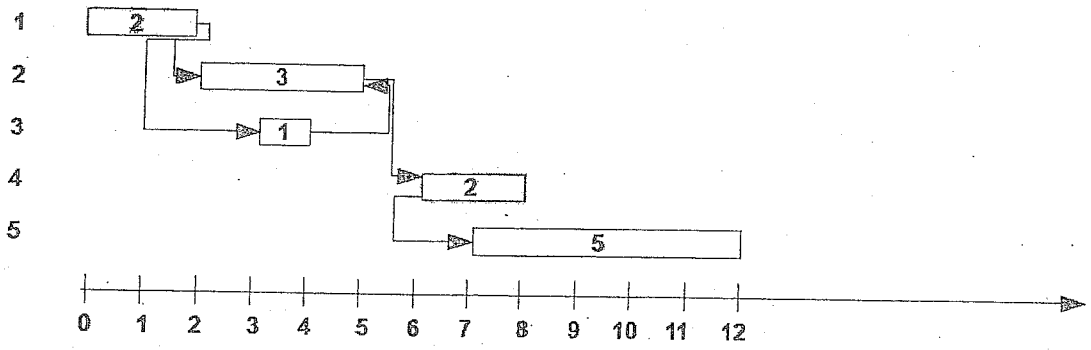


Abbildung 3: Gantt-Diagramm

Fortsetzung nächste Seite!

3 UML-Klassendiagramme

Die UEFA möchte zur kommenden Europa League und Champions League Saison ein neues System zur Verwaltung der entsprechenden Partien einsetzen. Erstellen Sie zu der folgenden Beschreibung ein Klassendiagramm, das Attribute und Assoziationen mit Kardinalitäten sowie die Methoden enthält. Setzen Sie dabei das Konzept der Vererbung sinnvoll ein.

- Ein Fussballspiel wird durch Zeit und Ort beschrieben.
- Ein Spiel findet immer zwischen zwei Teams statt.
- Teams werden durch den Namen beschrieben.
- An einem Spiel sind unterschiedliche Personen beteiligt, die ebenfalls durch einen Namen unterschieden werden:
    - Schiedsrichter, die sich in Hauptschiedsrichter, Ersatzschiedsrichter und Linienrichter untergliedern
    - Spieler
    - Trainer
- Ein Team besteht aus einem Kader von 23 Spielern, wobei die Startelf nur aus 11 Spielern besteht. Pro Team gibt es genau einen Kapitän, der als Spielführer bezeichnet wird.
- Das Team wird von genau einem Trainer betreut.
- Ein Spieler kann eine bestimmte Anzahl an Toren erzielen, die ausgegeben werden kann.
- Jede Person kann rennen.
- Darüber hinaus kann ein Schiedsrichter einen Regelverstoß anzeigen.
- Der Ersatzschiedsrichter kann noch die Tafel heben.
- Der Hauptschiedsrichter kann pfeifen, eine Karte zeigen und nur er kann das Spiel unterbrechen.
- Der Linienrichter kann mit der Fahne winken.
- Spieler können kicken.
- Es gibt eine Spezialform eines Spielers - den Torwart - dieser kann zusätzlich noch fangen.
- Der Trainer kann schreien und einen Spieler für einen anderen Spieler auswechseln.
- Das System erfasst, welcher Spieler für einen anderen Spieler eingewechselt wurde.
- Jeder Schiedsrichter kann Personen ermahnen, aber nur der Hauptschiedsrichter kann Personen auch verwarnen oder des Feldes verweisen. Die ermahnten bzw. verwarnten und des Feldes verwiesenen Personen werden erfasst. (30 Punkte)

Frühjahr 2013

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer
Kennzahl: _____	**Frühjahr 2013**	**66116**
Kennwort: _____		
Arbeitsplatz-Nr.: _____		

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: **13**

Bitte wenden!

Thema Nr. 1

Teilaufgabe 1:

1. Allgemeinwissen

a) Definieren Sie den Begriff Datenbanksystem und grenzen Sie die Funktionalitäten eines Datenbanksystems von denen eines Dateisystems ab.
b) Nennen Sie die Phasen des Datenbankentwurfsprozesses und charakterisieren Sie diese kurz.
c) Nennen und beschreiben Sie kurz die vier wesentlichen Eigenschaften von Transaktionen.
d) Nennen und erläutern Sie kurz vier Beispiele für Integritätsbedingungen.

2. ER-Modellierung und Relationenmodell

a) Erstellen Sie ein ER-Diagramm bestehend aus Entitäts- und Beziehungstypen sowie Attributen für folgendes Szenario. Geben Sie auch die Kardinalitäten mit an. Verwenden Sie bei Entitäten und Attributen ausschließlich die beschriebenen, soweit dies möglich ist. Verwenden Sie keine künstlichen Schlüssel, es sei denn, diese sind der Aufgabenstellung eindeutig zu entnehmen. Die Kardinalitätseinschränkungen können Sie entweder in der (min,max)-Notation oder der einfachen Notation nach Chen (1:1, 1:N, M:N) angeben.

Als **Szenario** dient eine Mitarbeiterverwaltung, die folgendermaßen beschrieben ist:

1. Mitarbeiter werden über ihre Sozialversicherungsnummer identifiziert. Sie haben einen Namen, bestehend aus Vor- und Nachnamen, sowie ein Geburtsdatum und ein Alter, das aus dem Geburtsdatum errechnet wird.
2. Ein Mitarbeiter kann der Vorgesetzte anderer Mitarbeiter sein.
3. Jeder Mitarbeiter ist einer Abteilung zugeordnet. Eine Abteilung wird über ihren Namen und eine Nummer identifiziert (z.B. „Entwicklung", „3"). Zu einer Abteilung werden mehrere Orte gespeichert, an denen die Abteilung angesiedelt ist.
4. Mitarbeiter können Abteilungen leiten. Jede Abteilung wird von einem Mitarbeiter geleitet, aber nicht jeder Mitarbeiter muss eine Abteilung leiten.
5. Abteilungen können Projekte realisieren. Ein Projekt wird durch eine Nummer identifiziert und findet an einem Ort statt.
6. Jeder Mitarbeiter arbeitet an einem oder mehreren Projekten. An jedem Projekt muss mindestens ein Mitarbeiter arbeiten. Zusätzlich wird gespeichert, ab wann ein Mitarbeiter an einem Projekt arbeitet.

Hinweis: In der Formulierung der Beziehungen sind die Kardinalitätseinschränkungen genau zu beachten. Im Allgemeinen gilt: Alles was nicht explizit eingeschränkt ist, muss im Modell auch uneingeschränkt bleiben (keine Einschränkungen in eine Beschreibung hineininterpretieren). Alles was explizit eingeschränkt ist, darf im Modell nicht einfach uneingeschränkt bleiben (alle formulierten Einschränkungen sind zu erkennen).

b) Erstellen Sie zu dem E/R-Diagramm aus Teil (a) ein Relationenschema. Seien Sie dabei so vollständig wie möglich – bspw. Berücksichtigung totaler Partizipationen. Vermeiden Sie unnötiges Ausprägen von Relationen bei allen Beziehungen.

Fortsetzung nächste Seite!

Beispiel für die Notation:

```
Relationenname (Primärschlüssel, Attribut1, Attribut2, ...,
    Fremdschlüssel[AndereRelation],
    (FremdschlüsselattributA,
    FremdschlüsselattributB)[DritteRelation])
    Attribut2 NOT NULL
```

3. Normalformen

Gegeben sei die nachfolgende relationale Datenbank mit unterstrichenen Schlüsselattributen. Sie enthält Informationen über Opernaufführungen. Relation "Opernvorstellungen":

Oper	Komponist	Datum	Ort	Opernhaus	Plätze	Dirigent	Dauer	Solist
Rigoletto	Verdi	08.04.2013	Wien	Staatsoper	2298	Lopez-Cobos	2:00	Polenzani
Don Carlos	Verdi	09.04.2013	Wien	Staatsoper	2298	de Billy	4:00	Youn
Rigoletto	Verdi	11.04.2013	Dresden	Semperoper	1300	Fisch	2:00	Berrugi
Macbeth	Verdi	16.04.2013	Mailand	Scala	2000	Gergiev	2:45	Vassallo
Carmen	Bizet	20.05.2013	Wien	Staatsoper	2298	de Billy	2:45	Garanca
Carmen	Bizet	02.06.2013	Wien	Staatsoper	2298	de Billy	2:45	Alagna
Don Carlos	Verdi	12.10.2013	Mailand	Scala	2000	Luisi	4:00	Pape

Gegeben seien ebenso die folgenden funktionalen Abhängigkeiten:

FD1: Oper → Komponist, Dauer
FD2: Ort → Opernhaus, Plätze
FD3: Oper, Ort → Dirigent
FD4: Oper, Datum, Ort → Solist

a) Beschreiben Sie kurz, welche Redundanzen in der Datenbank vorhanden sind und welche Anomalien auftreten können. Geben Sie ein Beispiel für jede Anomalie an.

b) Welcher Normalform genügt das angegebene Relationenschema? Begründen Sie Ihre Entscheidung.

c) Erläutern Sie kurz, welchen Nachteil Normalisierung allgemein für die Anfragebearbeitung haben kann.

4. Anfrageverarbeitung

a) Zeichnen Sie für folgendes SQL-Statement einen nicht-optimierten Anfragegraphen.

```
SELECT e1.name AS ename, e1.salary, dept.name AS dname,
       AVG (e2.salary) AS avg_salary
  FROM employee e1, employee e2, dept
 WHERE e1.dept_id = dept.id
   AND e2.dept_id = e1.dept_id
   AND dept.name LIKE 'Sales %'
 GROUP BY e1.name, e1.salary, dept.name
   HAVING AVG(e2.salary) > 2*e1.salary
```

b) Wie kann die relationale Algebra zur Optimierung von SQL-Anfragen eingesetzt werden?

Fortsetzung nächste Seite!

Teilaufgabe 2:

Aufgabe 1:

Sie sollen für eine Tankstelle eine Software zur Verwaltung von DVDs entwickeln. Jede DVD hat eine Nummer, einen Titel, eine Altersbeschränkung und kann entweder ausleihbar, für einen Kunden reserviert, oder von einem Kunden ausgeliehen sein. Eine Reservierung trägt ein Verfallsdatum; eine Ausleihe trägt ein Rückgabedatum.

Die beteiligten Personen können Kunden, Kassierer, oder Administratoren sein; der Einfachheit halber können Kassierer und Administratoren nicht Kunden sein.

Beteiligte Personen können sich beim System an- und wieder abmelden.

Kunden können DVDs reservieren, ausleihen und zurückgeben, und außerdem Gebühren schulden und bezahlen. Außerdem können Kunden den Bestand durchsuchen und Verfügbarkeitsdaten einsehen.

Kassierer können DVDs verleihen, sowie Rückgaben und Reservierungen entgegennehmen.

Außerdem können sie Kunden und DVDs neu aufnehmen oder löschen. Auch Kassierer können den Bestand durchsuchen, dabei aber im Gegensatz zu Kunden auch die ausleihenden Personen einsehen.

Administratoren haben alle Befugnisse von Kassierern und können außerdem Kassierer und Administratoren aufnehmen oder löschen.

- Entwickeln Sie aus der gegebenen Anwendungsbeschreibung ein UML-Klassendiagramm. Alle Angaben aus dem Text sollen sich in Ihrem Modell wiederfinden; weitere sollten nur dann hinzugenommen werden, wenn es aufgrund der von Ihnen gewählten Modellierung erforderlich erscheint.
- Verwenden Sie Vererbung und Interfaces, wo es sinnvoll möglich ist, und achten Sie auf eine angemessene Darstellung von Assoziationen und Aggregationen, wo erforderlich. Es geht hier nur um das UML-Klassendiagramm; insbesondere ist eine Modellierung als relationale Datenbank nicht verlangt und auch nicht im Sinne der Aufgabenstellung.
- Modellieren Sie den gesamten Anwendungsfall des Ausleihens einer DVD (Anmelden, Bestand anzeigen, Aussuchen, Datum eingeben, Bestätigen der Ausleihe, Abmelden) zunächst als UML-Anwendungsfall und dann als UML-Aktivitätsdiagramm. Achten Sie auch auf die Möglichkeit des Fehlschlagens einer Teilaktivität.
- Modifizieren Sie Ihr Modell so, dass Personen sowohl Administratoren bzw. Kassierer als auch Kunden sein können.

Fortsetzung nächste Seite!

Aufgabe 2:

Das Entwurfsmuster Kompositum dient der objektorientierten Modellierung hierarchischer, insbesondere rekursiver Datenstrukturen.

- Geben Sie ein allgemeines UML-Klassendiagramm für dieses Entwurfsmuster an.
- Geben Sie für ein konkretes Anwendungsbeispiel dieses Entwurfsmusters UML-Klassendiagramm und UML-Objektdiagramm für eine repräsentative Instanz an. Ihre Klassen sollten mindestens eine Methode beinhalten; Ihr Objektdiagramm sollte mindestens drei Objekte umfassen.
- Nennen Sie zwei weitere Entwurfsmuster und beschreiben Sie ihre Einsatzmöglichkeiten in jeweils ca. drei Sätzen.

Thema Nr. 2

Teilaufgabe 1:

Aufgabe 1: Relationale Algebra und SQL

Gegeben sei folgendes Schema:

- Züge:
  {[ZugNr: integer, AnzahlWagons : integer]}
- Städte:
  {[StadtNr: integer, Name: string]}
- Bahnhöfe:
  {[BahnhofNr: integer, Name: string, LiegtInStadtNr: integer]}
- Verbindungen:
  {[ZugNr: integer, BahnhofNrVon: integer, BahnhofNrNach: integer, AbfahrtsZeit: date, AnkunftsZeit: date]}

Hinweis:
Ein Element ∈ date hat der Einfachheit halber die Form: „5:32PM". Wir nehmen an, dass Verbindungen jeden Tag gelten.

Aufgabe 1.1:
Geben Sie tabellarisch eine Beispielausprägung mit mindestens zwei Zeilen pro Tabelle des angegeben Schemas an.

Aufgabe 1.2

1. Finden Sie die Züge, welche morgens um 5 Uhr von dem Bahnhof „Berlin Ostbahnhof" zu dem Bahnhof „München Hauptbahnhof" fahren **(in relationaler Algebra).**
2. Finden Sie die Städte, von denen mindestens ein Zug mit mehr als drei Wagons zu einem Bahnhof in Mannheim fährt **(in relationaler Algebra).**
3. In welcher Stadt liegt der Bahnhof „Coesfeld Schulzentrum"? **(in relationaler Algebra)**
4. Finden Sie die Namen der Städte an denen zur gleichen Uhrzeit mehr als ein Zug ankommt **(in relationaler Algebra).**

Fortsetzung nächste Seite!

Aufgabe 1.3:

Beantworten Sie folgende Fragen in SQL:

1. An welchen Uhrzeiten fahren Züge von „München Hauptbahnhof" nach „Berlin Ostbahnhof"?
2. Finden Sie den Zug oder die Züge mit den meisten Wagons.
3. Welche Bahnhöfe (BahnhofNr angeben) sind mit dem Bahnhof "München Hauptbahnhof" direkt verbunden? Geben Sie die Bahnhöfe duplikatfrei aus. „München Hauptbahnhof" selbst darf nicht im Ergebnis vorkommen.
4. Finden sie alle Bahnhöfe (BahnhofNr), die mit **genau zweimal** Umsteigen von dem Bahnhof „München Hauptbahnhof" aus erreicht werden können.
5. Geben Sie die Namen der Bahnhöfe an, die überdurchschnittlich oft als Ziel vorkommen. Geben Sie außerdem zu jedem der Bahnhöfe die Anzahl der ankommenden Züge an. Beachten Sie hierbei, dass auch Bahnhöfe, die von keinem Zug erreicht werden können, den Durchschnitt beeinflussen.

Aufgabe 1.4:

Gegeben sei folgende SQL Anfrage:

```
SELECT z.ZugNr FROM
        Zug z, Bahnhof b1, Bahnhof b2, Verbindung v
WHERE
        v.BahnhofNrVon = b1.BahnhofNr AND
        v.BahnhofNrNach = b2.BahnhofNr AND
        b1.Name = "München" AND
        b2.Name = "Berlin" AND
        z.ZugNr = v.ZugNr
```

Übertragen Sie das gegebene SQL Statement in die relationale Algebra und optimieren Sie den Ausdruck logisch. Verwenden Sie die Operatorbaum-Syntax.

Fortsetzung nächste Seite!

Aufgabe 2: ER Modellierung

Sie sind beauftragt, bei der Entwicklung des ersten Prototypen eines sozialen Netzwerkes die Datenmodellierung zu übernehmen. Der erste Prototyp soll folgende Anforderungen erfüllen:

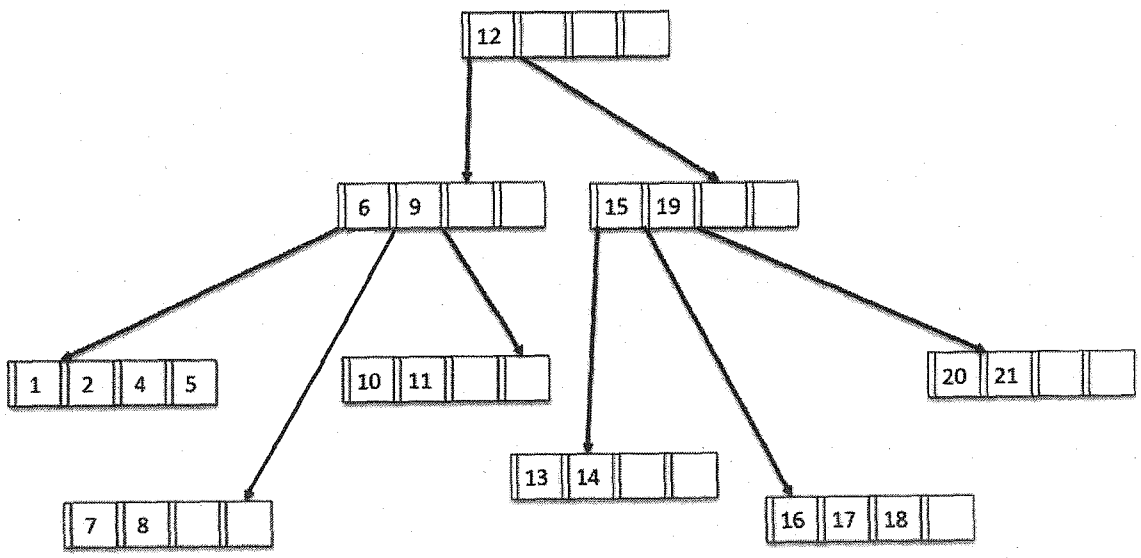
Benutzer können sich mit ihrem Nutzernamen und Kennwort anmelden. Ein Benutzer hat eine sogenannte Wall, auf die andere Benutzer Texte schreiben können. Benutzer können Freunde von anderen Benutzern sein. Außerdem können Benutzer sich in verschiedenen Gruppen anmelden. Gruppen haben ebenfalls eine sogenannte Wall, auf der Gruppenmitglieder Nachrichten hinterlassen können.

Erstellen Sie ein ER-Diagramm, welches die obengenannten Anforderungen erfüllt. Denken Sie daran Ihrem ER-Diagramm Funktionalitätsangaben hinzuzufügen.

Fortsetzung nächste Seite!

Aufgabe 3: B-Baum

Gegeben sei der folgende B-Baum



1. Was bedeutet *k* bei einem B-Baum mit Grad *k*? Geben Sie *k* für den obigen B-Baum an.
2. Was sind die Vorteile von B-Bäumen im Vergleich zu binären Baumen?
3. Wozu werden B-Bäume in der Regel verwendet und wieso?
4. Fügen Sie den Wert **3** in den B-Baum ein, und zeichnen Sie den vollständigen B-Baum nach dem Einfügen und möglichen darauf folgenden Operationen.
5. Entfernen Sie aus dem **ursprünglichen** B-Baum den Wert **19**. Zeichnen Sie das vollständige Ergebnis nach dem Löschen und möglichen darauf folgenden Operationen. Sollte es mehrere richtige Lösungen geben, reicht es eine Lösung zu zeichnen.

Fortsetzung nächste Seite!

Teilaufgabe 2:

Aufgabe 1: Allgemeines

a) Was ist unter Softwaretechnik zu verstehen?

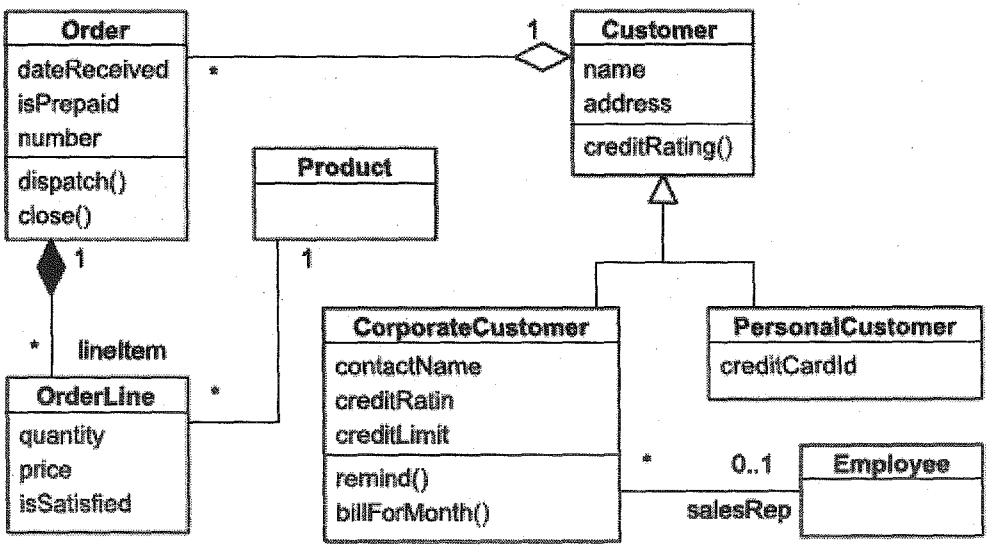
b) Nennen sie ein Vorgehensmodell (oder auch Prozess-Modell) in der Softwareentwicklung. Welcher ist der wesentliche Vorteil dieses Modells gegenüber anderen?

c) Nennen sie alle am Softwareentwicklungsprozess beteiligten Rollen (Akteure) und ordnen sie diesen mindestens eine Phase zu.

Fortsetzung nächste Seite!

Aufgabe 2: UML

Erstellen Sie ein Klassendiagramm, das UML-Klassendiagramme modelliert, d.h. ein Meta-Modell für UML-Klassendiagramme. Ein Klasse besitzt einen Namen, Methoden und Attribute. Bilden Sie mindestens die in dem unten angegebenen Klassendiagramm verwendeten UML-Klassendiagramm-Elemente ab. Die Modellierung von Typen und Anfangswerten ist nicht erforderlich.



Aufgabe 3 (Architekturbeschreibung)

Erstellen Sie eine Architekturbeschreibung für ein einfaches Studentenverwaltungssystem, das die Prüfungsbeteiligung (mit Status und Note) von Studenten (mit Name, Vorname, Matrikelnummer) an angebotenen Prüfungen (mit Bezeichnung, Ort, Zeit) verwalten soll. Eine Komponente „Prüfungsbeteiligung" soll folgende typische Aufgaben unterstützen:

1. Anmeldung und Abmeldung eines Studenten zu einer bestimmten Prüfung,
2. Abfrage des Status einer Prüfungsbeteiligung (angemeldet, abgemeldet, teilgenommen, bestanden, durchgefallen) zu einer bestimmten Prüfung durch einen Studenten,
3. Abfrage der Note eines Studenten zu einer Prüfung,
4. Eintragung der Teilnahme eines Studenten an einer Prüfung durch die Übungsleitung,
5. Eintragung der Note eines Studenten an einer Prüfung durch die Übungsleitung.

Nicht zu berücksichtigen sind dabei Authentifizierung und Autorisierung.

Geben Sie folgende Bestandteile einer Architekturbeschreibung des Systems an:

1. Datenmodell (als ER- oder Klassendiagramm),
2. Spezifikation der Methoden der Komponente „Prüfungsbeteiligung" nach „design by contract" (Vorbedingung, Nachbedingung, Invariante).

Fortsetzung nächste Seite!

Aufgabe 4: Funktionale Programmierung

Wenn Sie die Sprache Haskell kennen, benutzen Sie sie zur Beantwortung dieser Frage. Ansonsten benutzen Sie die funktionale Sprache, in der Sie ausgebildet wurden. Bleiben Sie aber rein funktional, also vermeiden Sie Seiteneffekte oder wiederholte Zuweisungen.

In dieser Frage betrachten wir Funktionen, die eine Liste revertieren. D.h. die Ausgabeliste enthält dieselben Elemente wie die Eingabeliste, nur in umgekehrter Reihenfolge.

a) Einfache Verkettung. Benutzen Sie hier den in der Sprache mitgelieferten Listentyp (in Haskell `[a]`, in ML `'a list`, etc.). Definieren Sie eine Funktion `reverse`, die eine Liste revertiert. Dazu geben Sie

   i. die Signatur (also den Typ) der Funktion und

   ii. ihre definierenden Gleichungen an.

   iii. Welche asymptotische Zeitkomplexität in der Länge der Eingabeliste hat Ihre Funktion?

b) Doppelte Verkettung.

   i. Definieren Sie einen neuen Datentyp für Listen mit Elementen vom Typ `a`, bei denen Elemente sowohl am Anfang als auch am Ende eingefügt werden können, in Haskell `DList a`. Die entsprechenden Konstruktoren sollen `Cons` (für Anhängen am Anfang) und `Snoc` (für Anhängen am Ende) heißen.

   ii. Definieren Sie (mit Signatur und Gleichungen) eine Funktion `reverseDList`, die eine `DList a` revertiert.

   iii. Welche asymptotische Zeitkomplexität in der Länge der Eingabeliste hat diese Funktion?

Fortsetzung nächste Seite!

Aufgabe 5: 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.

Herbst 2013

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer
Kennzahl: _____	**Herbst 2013**	**66116**
Kennwort: _____		
Arbeitsplatz-Nr.: _____		

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?

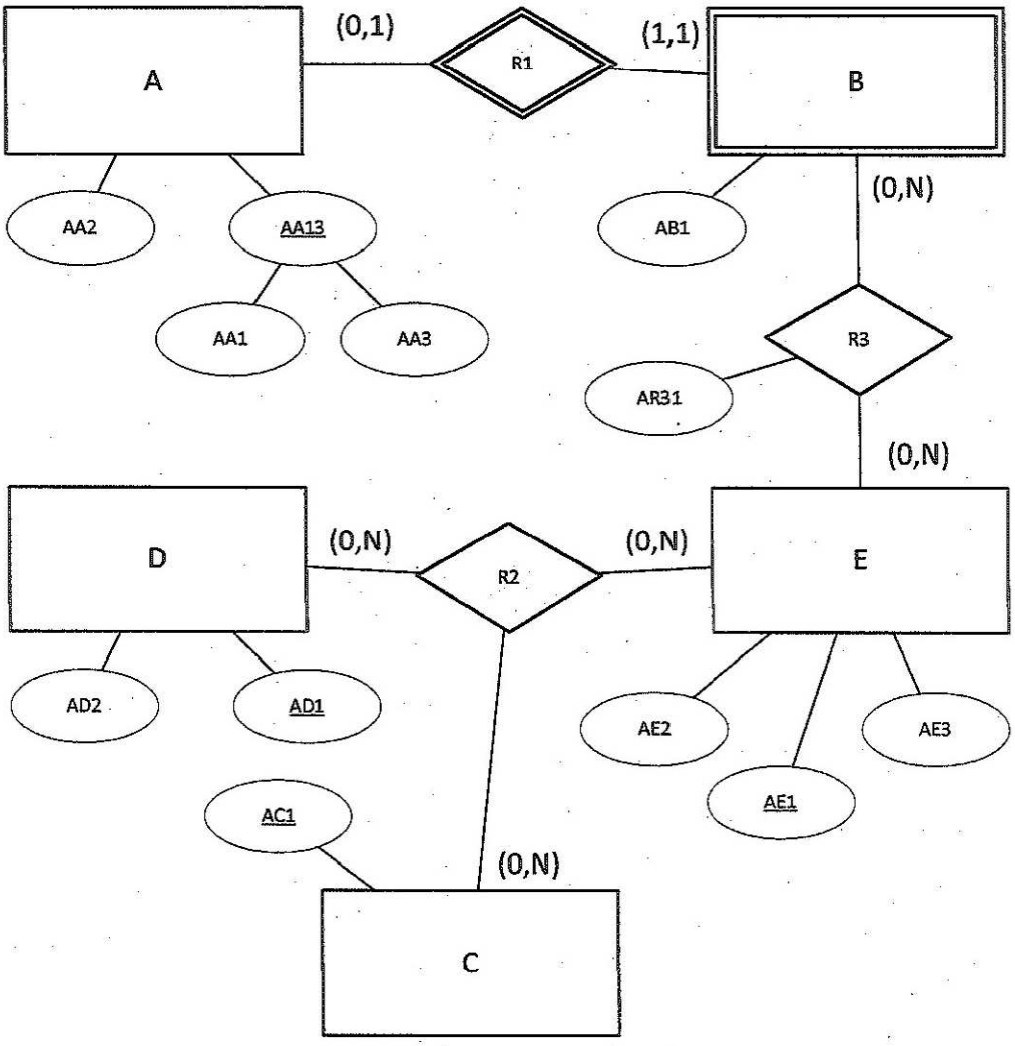
PersNr	Name	Land	Landesvorwahl	Vorwahl	Rufnummer
1	Maier	Frankreich	33	556	56881
2	Maier	Deutschland	49	821	4563
3	Pascal	Frankreich	33	556	56881
4	Konrad	Deutschland	49	841	923476
5	Schmidt	USA	1	0212	928357
6	Berner	Frankreich	33	556	183258

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!

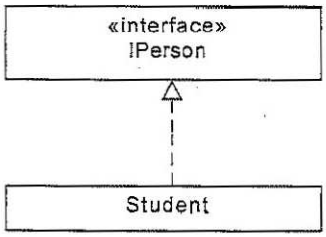
4. Objektorientierte Programmierung

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

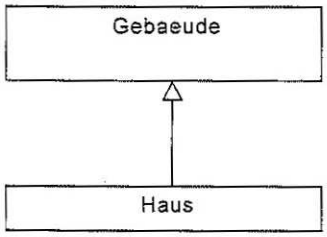
Was versteht man im Kontext der objektorientierten Programmierung unter dem Begriff "Kapselung". Nennen Sie zwei Vorteile die sich aus der konsequenten Einhaltung des Prinzips der Kapselung ergeben.

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

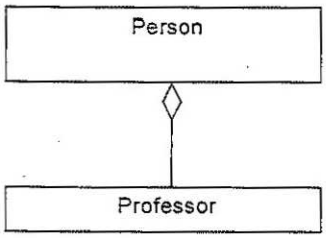
a)



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.

Frühjahr 2014

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer
Kennzahl: _____	**Frühjahr 2014**	**66116**
Kennwort: _____		
Arbeitsplatz-Nr.: _____		

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: 14

Bitte wenden!

Thema Nr. 1

Teilaufgabe 1:

1. Modellierung

Der VDS (Verein deutscher Schwimmbäder) möchte Informationen über seine bestehenden Einrichtungen in Verbindung mit dessen Besucherzahlen festhalten und digitalisieren. Dazu soll eine Datenbank in zwei Schritten erstellt werden. Der erste soll folgende Informationen enthalten:

- Personen werden spezifiziert durch ihren Vor- sowie Nachnamen und einer Ausweisnummer.
- Ein Schwimmbad besitzt einen Namen und eine Stadt, in der es sich befindet.
- Personen besuchen Schwimmbäder.
- Für Schwimmbecken wird ein Name, die Länge, die Breite sowie die Tiefe gespeichert. Ein Schwimmbecken befindet sich in genau einem Schwimmbad und kann nur durch dieses eindeutig identifiziert werden.
- Personen können in einem Schwimmbecken schwimmen, wobei dazu ein Schwimmstil festgehalten werden soll.

Im zweiten Schritt der Datenbank wird spezieller auf die Personen im Schwimmbad eingegangen:

- Im Bezug auf die Domäne unterteilen sich Personen in Personal des Schwimmbads und Besucher. Angestellte des Schwimmbads können dieses aber auch außerhalb Ihrer Arbeitszeiten besuchen. Für das Personal wird eine Personalnummer, für die Besucher eine Besuchernummer gespeichert.
- Es werden Schwimmkurse angeboten. Zu diesen wird eine Kursnummer sowie dessen Gebiet festgehalten.
- Ein Angestellter kann eine beliebige Anzahl an Kursen leiten, wobei jeder Kurs eine beliebige Anzahl an Besuchern als Teilnehmer fassen kann.

Fortsetzung nächste Seite!

a) Entwerfen Sie für das beschriebene Szenario des **ersten** Schritts der Datenbank ein ER-Modell! Bestimmen Sie hierzu:

- die Entity-Typen, die Relationship-Typen und jeweils deren Attribute,
- die Primärschlüssel der Entity-Typen, welche Sie anschließend in das ER-Diagramm eintragen, und
- die Kardinalitäten der Relationship-Typen.

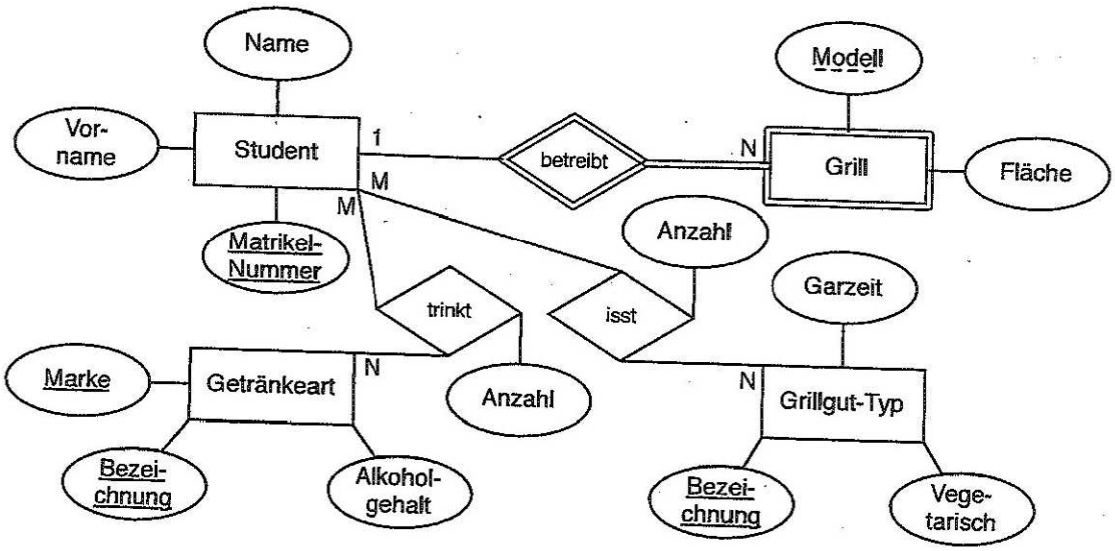
b) Übernehmen Sie den Entity-Typ für eine Person aus Aufgabenteil a) und entwerfen Sie ausgehend davon ein weiteres ER-Modell für den **zweiten** Teil der Datenbank! Bestimmten Sie dazu ebenfalls dieselben Kriterien wie in Aufgabenteil a)!

c) Überführen Sie das ER-Modell aus Aufgabe a) in ein verfeinertes relationales Modell! Geben Sie hierfür die verallgemeinerten Relationenschemata an! Achten Sie dabei insbesondere darauf, dass die Relationenschemata keine redundanten Attribute enthalten!

Fortsetzung nächste Seite!

2. SQL-Anfragen

Gegeben sei folgendes ER-Modell, welches eine Datenbank für Statistiken über Grillabende beschreibt. Darin sind Studenten, die Grills betreiben, enthalten. Zu jedem Studenten soll die Anzahl an Speisen und Getränken erfasst werden, welche sie zu sich genommen haben:



Gehen Sie von folgendem dazugehörigen relationalem Schema aus:

Student{[Matrikel_Nummer, Name, Vorname]}
Grill{[Modell, Matrikel_Nummer, Flaeche]}
Getraenkeart{[Marke, Bezeichnung, Alkoholgehalt]}.
Grillgut-Typ{[Bezeichnung, Vegetarisch, Garzeit]}
Trinkt{[Getraenkeart_Marke, Getraenkeart-Bezeichnung, Matrikel-Nummer, Anzahl]}
Isst{[Grillgut-Typ-Bezeichnung, Matrikel-Nummer, Anzahl]}

Formulieren Sie folgende Anfragen in SQL. Achten Sie, falls nötig, auf duplikatfreie Ausgaben:

a) Geben Sie die Garzeit des Grillgut-Typs mit der Bezeichnung „Schweinenacken" aus!

b) Geben Sie die Namen und Vornamen aller Grillmeister aus! *Hinweis:* Als Grillmeister gelten Studierende, die mindestens einen Grill betreiben.

c) Geben Sie alle Attribute von Studierenden aus, die mindestens ein Getränk mit mehr als 5 % Alkohol getrunken haben!

d) Geben Sie alle Attribute von Studierenden aus, die mindestens zwei alkoholhaltige Getränke konsumiert haben!

Fortsetzung nächste Seite!

3. Normalformen

Gegeben sei folgendes relationales Schema R, dessen Attribute nur atomare Attributwerte besitzen.

$$R : \{[A, B, C, D, E, F, G, H]\}$$

Es gelte die folgende Menge FD von funkionalen Abhängigkeiten:
$FD = \{$

$$A \rightarrow B,$$
$$CD \rightarrow EB,$$
$$C \rightarrow DA,$$
$$CF \rightarrow FE,$$
$$F \rightarrow GHCA$$

$\}$

a) Nennen Sie die drei Änderungsanomalien, die auftreten können, wenn ein Datenbank-Schema nicht der Boyce-Codd-Normalform entspricht, und beschreiben Sie eine näher!

b) Definieren Sie die Begriffe Superschlüssel, Schlüsselkandidat und Primarschlüssel. Wie unterscheiden sich diese voneinander?

c) Finden Sie den/die Schlüsselkandidaten!

d) Prüfen Sie, in welcher Normalform R vorliegt!

e) Berechnen Sie die kanonische Überdeckung FD_c von FD!

Hinweis: Es genügt, wenn Sie für jeden der vier Einzelschritte die Menge der funktionalen Abhängigkeiten als Zwischenergebnis angeben.

f) Bestimmen Sie eine Zerlegung der Relation R in dritter Normalform. Verwenden Sie dafür den Synthesealgorithmus!

Hinweis: Sie können dazu die kanonische Überdeckung FD_c verwenden, die Sie in der vorherigen Teilaufgabe berechnet haben.

Fortsetzung nächste Seite!

Teilaufgabe 2:

Es soll ein Framework für Zweipersonenspiele, wie z.B. Mühle, Schach, Reversi, Tic-TacToe etc. erstellt werden, welches es erleichtert, solche Spiele als Handyanwendung schnell und kostengünstig zu lancieren.

In all diesen Spielen sind zwei Spieler beteiligt, die abwechselnd ziehen.Das Framework soll aufgrund vorher festgelegter Regeln überprüfen, ob ein Zug erlaubt ist und das Ergebnis des Zuges auf dem Bildschirm anzeigen. Es muss die Möglichkeit geben, Spielregeln, die Art der Züge, die Geometrie und das konkrete Aussehen des Spielfeldes festzulegen. Außerdem sollen bisher stattgefundene Partien abgespeichert werden können und eine Bestenliste verwaltet werden.

1. Skizzieren Sie einen objektorientierten Entwurf für diese Aufgabenstellung in Form eines UML Klassendiagramms. Ihr Diagramm sollte nicht weniger als sieben Klassen haben und neben den Beziehungen der Klassen untereinander auch deren wichtigste Methoden und Attribute beinhalten. Sie können sich zunächst an einem bestimmten Spiel orientieren, müssen aber die allgemeine Einsetzbarkeit sicherstellen.
2. Erstellen Sie ein Ablaufdiagramm, aus dem hervorgeht, welcher Spieler am Zug ist, ob ein illegaler Zug durchgeführt wurde und ob das Spiel zuende ist.
3. Erläutern Sie detailliert zwei Entwurfsmuster Ihrer Wahl. Geben Sie insbesondere entsprechende Klassendiagramme an und skizzieren Sie die jeweiligen Einsatzmöglichkeiten. Es wird empfohlen, solche Muster zu wählen, die in Ihrem Entwurf auch eingesetzt wurden.

Thema Nr. 2

Teilaufgabe 1:

Aufgabe 1: E/R-Modellierung, SQL-DDL

Für die Verwaltung von Museen soll folgendes System entworfen werden:

- Die zentrale Komponente ist das Museum mit eindeutigem Namen.
- Ein Museum besteht aus mindestens einem Themenbereich, welcher ebenfalls einen eindeutigen Namen besitzt.
- Ein Themenbereich erstreckt sich über mindestens einen Raum, der einen Namen hat und über eine eindeutige Raumnummer identifiziert werden kann. Ein Raum gehört zu genau einem Themenbereich.
- In einem Raum befindet sich mindestens ein Kunstwerk. Ein Kunstwerk gehört zum Eigentum genau eines Museums, welches aber (im Fall einer Leihgabe) nicht zwingend das Museum sein muss, in dem es ausgestellt ist. Ein Museum kann Eigentümer vieler Kunstwerke sein.
- Ein Kunstwerk wurde von genau einem Künstler erstellt und gehört einer speziellen Kunstepoche an. Jedes Kunstwerk hat einen eindeutigen Titel sowie einen Untertitel und ein Entstehungsjahr.
- Eine Kunstepoche wird durch einen eindeutigen Namen und einen Zeitraum (Start- und Endjahr) beschrieben.
- Ein Künstler wird durch seinen Namen und Vornamen identifiziert und hat außerdem ein Geburtsdatum und gegebenenfalls ein Sterbedatum. Von Künstlern kann man sagen, dass sie in ihrem Leben viele Kunstwerke erschaffen und in mindestens einer Kunstepoche gewirkt haben.
- Manche Künstler sind Mitglied in einer Künstlervereinigung mit eindeutigem Namen und einem Gründungsdatum.

1. Erstellen Sie ein Entity-Relationship-Diagramm für obige Datenbank!
2. Setzen Sie das gegebene E/R-Diagramm in ein entsprechendes relationales Datenbankschema um. Geben Sie die resultierenden Relationenschemata in folgender Schreibweise an:
   `Relation (Attribut1, Attribut2, ..., AttributN)`
   Identifizieren Sie für jede Relation einen Primärschlüssel und unterstreichen Sie diesen! Achten Sie auf eine möglichst kompakte Modellierung der Relationships! Geben Sie außerdem erforderliche Fremdschlüsselbeziehungen an!
3. Geben Sie die Anweisungen in SQL-DDL an, die notwendig sind, um die Relationen für ein **Kunstwerk** und eine **Kunstepoche** aus Teilaufgabe (2) in einer relationalen Datenbank zu erzeugen! Kennzeichnen Sie dabei die Primär- und Fremdschlüssel der Relationen. Geben Sie sinnvolle Integritätsbedingungen für Zahlen und Daten an und achten Sie auf die sinnvolle Behandlung möglicher leerer Ausprägungen!

Fortsetzung nächste Seite!

Aufgabe 2: Relationale Algebra und SQL

Gegeben sei das folgende relationale Schema mitsamt Beispieldaten für eine Datenbank für Bierbrauereien und ihre Produkte. Die Primärschlüssel-Attribute sind jeweils unterstrichen, Fremdschlüssel sind überstrichen.

"Brauerei":

ID	Name	Sitz	Gründungsjahr
B1	Augustiner	S4	1328
B2	Beck's	S3	1873
B3	Paulaner	S4	1634
B4	Krombacher	S5	1803
B5	Früh	S1	1904
B6	Gaffel	S1	1908
B7	Hoepfner	S2	1798
B8	Spaten	S4	1397

"Sitz":

ID	Stadt	Bundesland
S1	Köln	Nordrhein-Westfalen
S2	Karlsruhe	Baden-Württemberg
S3	Bremen	Bremen
S4	München	Bayern
S5	Krombach	Nordrhein-Westfalen

"Biersorte":

ID	Bezeichnung	Hefetyp
T1	Pils	untergärig
T2	Lager	untergärig
T3	Weizenbier	obergärig
T4	Helles	untergärig
T5	Starkbier	ober- und untergärig
T6	Export	untergärig
T7	Kölsch	obergärig
T8	Altbier	obergärig

"Bier":

ID	Brauerei	Biersorte	Alkoholgehalt
A01	B1	T4	5,2
A02	B1	T6	5,6
A03	B2	T1	4,9
A04	B3	T3	5,5
A05	B3	T4	4,9
A06	B4	T1	4,8
A07	B5	T7	4,8
A08	B6	T7	4,8
A09	B7	T1	4,7
A10	B8	T4	5,2

1. Formulieren Sie die folgenden Anfragen auf das gegebene Schema in relationaler Algebra:

   a) Finden Sie die Namen aller Brauereien, die vor dem Jahr 1800 gegründet wurden.

   b) Finden Sie die Bezeichnungen aller Biersorten mit einem Alkoholgehalt von 4,8% oder 4,9%.

   c) Finden Sie die Städte und zugehörigen Bundesländer, in denen Brauereien ihren Sitz haben, die im 19. Jahrhundert gegründet wurden.

   d) Finden Sie die Bezeichnungen aller Biersorten, die von der Brauerei 'Paulaner' hergestellt werden.

2. Geben Sie das Ergebnis (bezüglich der Beispieldaten) der folgenden Ausdrücke der relationalen Algebra als Tabellen an:

   a) $\pi_{Name,Bundesland}(Brauerei \bowtie_{Brauerei.Sitz=Sitz.ID} Sitz)$

   b) $\pi_{Gründungsjahr,\ Hefetyp}$
   $(Brauerei \bowtie_{Brauerei.ID=Bier.Brauerei} Bier \bowtie_{Bier.Biersorte=Biersorte.ID} \sigma_{Hefetyp\ =\ 'obergärig'}(Biersorte))$

Fortsetzung nächste Seite!

3. Formulieren Sie die folgenden Anfragen in SQL:

a) Finden Sie die Namen aller Brauereien, die mehr als eine Biersorte herstellen!

b) Geben Sie die Bezeichnung einer jeden Biersorte zusammen mit ihrem durchschnittlichen Alkoholgehalt an!

c) Geben Sie die Stadt aus, in der die älteste Brauerei ihren Sitz hat!

d) Finden Sie alle Bundesländer mit Brauereien, die nur untergärige Biersorten herstellen!

4. Wie sieht die Ergebnisrelation zu folgenden Anfragen auf den Beispieldaten aus?

a)
```
select Hefetyp, count(Bezeichnung) as Anzahl
from Biersorte
group by Hefetyp
having count(Bezeichnung) > 1;
```

b)
```
Select Bezeichnung
from Biersorte
where ID not in
 (select Biersorte
  from Bier join Brauerei on Bier.Brauerei = Brauerei.ID
            join Sitz on Brauerei.Sitz = Sitz.ID
  where Bundesland = 'Bayern');
```

Fortsetzung nächste Seite!

Aufgabe 3: Entwurfstheorie

Gegeben sei das folgende relationale Schema mitsamt Beispieldaten für eine Flugdatenbank. Die Primärschlüsselattribute sind unterstrichen.

Relation "Flüge":

FNr	Datum	VonS	VonL	NachS	NachL	Gesellschaft	Sitz	Flugzeug	Typ	Plätze
LH1432	03.04.11	München	GER	Frankfurt	GER	Lufthansa	Köln	Boeing	747	452
UA732	05.04.12	Chicago	USA	Brisbane	AUS	United	Chicago	Embraer	ERJ-195	108
UA735	05.04.12	Frankfurt	GER	Chicago	USA	United	Chicago	Airbus	320	152
EK935	01.02.11	Dubai	UAE	Montreal	CAN	Emirates	Dubai	Airbus	380	526
UA735	05.04.09	Frankfurt	GER	Chicago	USA	United	Chicago	Airbus	320	152

Gegeben seien ebenso die folgenden funktionalen Abhängigkeiten:

FD1: FNr → VonS, VonL, NachS, NachL, Gesellschaft, Sitz, Flugzeug, Typ, Plätze
FD2: VonS → VonL
FD3: NachS → NachL
FD4: Gesellschaft → Sitz
FD5: Flugzeug, Typ → Plätze

1. Beschreiben Sie kurz, welche Redundanzen in der Datenbank vorhanden sind und welche Anomalien auftreten können. Geben Sie ein Beispiel für jede Anomalie an.
2. Welchen Normalformen genügt das angegebene Relationenschema? Begründen Sie Ihre Entscheidung.
3. Überführen Sie das obige Relationenschema in die dritte Normalform und geben Sie die mit obigen Daten gefüllten Relationen an.
4. Erläutern Sie kurz, welchen Nachteil Normalisierung allgemein für die Anfragebearbeitung haben kann.

Fortsetzung nächste Seite!

Teilaufgabe 2:

Aufgabe 1 (Objektorientierung)

In dieser Aufgabe werden Sie Datentypen für die Verwaltung eines Parkhauses mit Hilfe objektorientierter Methoden definieren. Bearbeiten Sie die folgenden Teilaufgaben in einer objektorientierten Programmiersprache Ihrer Wahl (geben Sie diese an)! Solange nicht anders definiert, sollen Eigenschaften und Methoden **uneingeschränkt sichtbar** sein.

(a) Erzeugen Sie eine **Klasse** Fahrzeug, deren Instanzen folgende Eigenschaften besitzen (wählen Sie geeignete Typen):

- Ein amtliches Kennzeichen (Buchstaben- und Zahlenkombination).
- Die Dimensionen des Fahrzeugs (Länge, Breite, Höhe) in Metern.
- Das Datum der Erstzulassung. Definieren Sie hierfür entweder einen eigenen Datentyp oder machen Sie Gebrauch von der Standardbibliothek Ihrer gewählten Programmiersprache.

Die Eigenschaften sollen **für Unterklassen nicht sichtbar** sein. Schreiben Sie außerdem einen **Konstruktor**, der eine Instanz erzeugt und die Eigenschaften setzt!

(b) Schreiben Sie eine **Klasse** Parkplatz, in der ebenfalls Eigenschaften für die Dimension (Länge, Breite, Höhe) in Metern vorgesehen sind! Die Eigenschaften sollen ebenfalls im **Konstruktor** initialisiert werden können. Außerdem soll eine **Objektmethode** hinzugefügt werden, die prüft, ob ein gegebenes Fahrzeug in den Parkplatz passt.

(c) Ein **Interface** Parkhaus soll **Objektmethoden** für folgende Anwendungsfälle **deklarieren**:

(i) Alle freien Parkplätze sollen (z.B. als **Array** oder als Instanz einer in der Standardbibliothek Ihrer verwendeten Sprache definierten **Kollektionsklasse**) zurückgegeben werden.
(ii) Der erste freie Parkplatz, der zu einem gegebenen Fahrzeug passt, soll zurückgegeben werden.
(iii) Ein gegebener Parkplatz soll für ein gegebenes Fahrzeug reserviert werden.

Deklarieren Sie das Interface und geben Sie geeignete Signaturen für die **Objektmethoden** an!

(d) Schreiben Sie eine **abstrakte Klasse**, die das Interface Parkhaus **partiell** implementiert! Geben Sie außerdem eine geeignete **Implementierung** für die Objektmethode (c.ii) unter Verwendung der existierenden Objektmethoden aus (b) und (c.i) an!

Fortsetzung nächste Seite!

Aufgabe 2 (Klassendiagramm)

Erstellen Sie zu der folgenden Beschreibung eines Systems zur Organisation eines Supermarktes ein UML-Klassendiagramm, das Attribute und Assoziationen mit Kardinalitäten, Rollennamen und Namen, sowie Methoden enthält! Setzen Sie dabei das Konzept der Vererbung sinnvoll ein!

- Ein Supermarkt hat einen Namen.
- Er wirbt und kann eine Inventur durchführen. Bei der Werbung werden bestimmte Waren beworben, die im Vorfeld festgelegt werden. Bei einer Inventur wird zurückgegeben, ob diese erfolgreich war.
- Der Supermarkt besteht aus beliebig vielen Regalen. Ein Regal gehört zu genau einem Supermarkt.
- Ein Regal hat einen Namen und eine eindeutige Nummer.
- Innerhalb eines Supermarktes befinden sich beliebig viele Waren. Jede Ware befindet sich in genau einem Supermarkt.
- Eine Ware hat einen Namen und einen Preis.
- Eine Ware ist einem Regal zugeordnet. Ein Regal beinhaltet beliebig viele, jedoch mindestens eine Ware.
- Im Kaufhaus arbeiten Verkäufer. Diese haben einen Namen und eine Personalnummer.
- Verkäufer werden zudem in Regalbetreuer und Kassierer unterschieden.
- Ein Verkäufer ist in mindestens einem Kaufhaus beschäftigt. Ein Kaufhaus beschäftigt beliebig viele Verkäufer.
- Ein Kassierer kennt den von ihm gemachten Umsatz.
- Ein Regalbetreuer hat einen Zuständigkeitsbereich, der als Zeichenkette repräsentiert wird.
- Beliebig viele Kunden kaufen in beliebig vielen Supermärkten ein und werden von Verkäufern beraten.
- Ein Verkäufer kann Auskunft über den Preis einer Ware geben, wenn man ihm die Ware übergibt.
- Ein Kunde kann einen Supermarkt mit gut oder schlecht bewerten. (Methode!)

Fortsetzung nächste Seite!

Aufgabe 3 (Projektmanagement)

Die Abbildung 1 zeigt ein CPM-Netzwerk mit topologisch sortierten Knoten. Die Vorgänge tragen keine Namen. Gestrichelte Linien zwischen Ereignissen stellen Scheinvorgänge (auch Pseudoaktivitäten) mit einer Dauer von 0 dar.

(a) Welche Scheinvorgänge könnten aus dem Netzwerk entfernt werden, ohne dass Informationen verloren gehen? (mit Begründung)

(b) Berechnen Sie für jedes Ereignis den frühesten Termin, wobei angenommen wird, dass das Projekt zum Zeitpunkt 0 startet! Geben Sie den frühestmöglichen Endtermin des Projekts an!

(c) Berechnen Sie für jedes Ereignis auch die spätesten Zeiten, indem Sie für das letzte Ereignis den frühesten Termin als spätesten Termin ansetzen!

(d) Geben Sie nun die Pufferzeiten der Ereignisse an!

(e) Wie verläuft der kritische Pfad durch das Netzwerk?

(f) Was bedeutet es für die Gesamtprojektdauer, wenn sich ein Vorgang auf dem kritischen Pfad um zwei Zeiteinheiten verzögert?

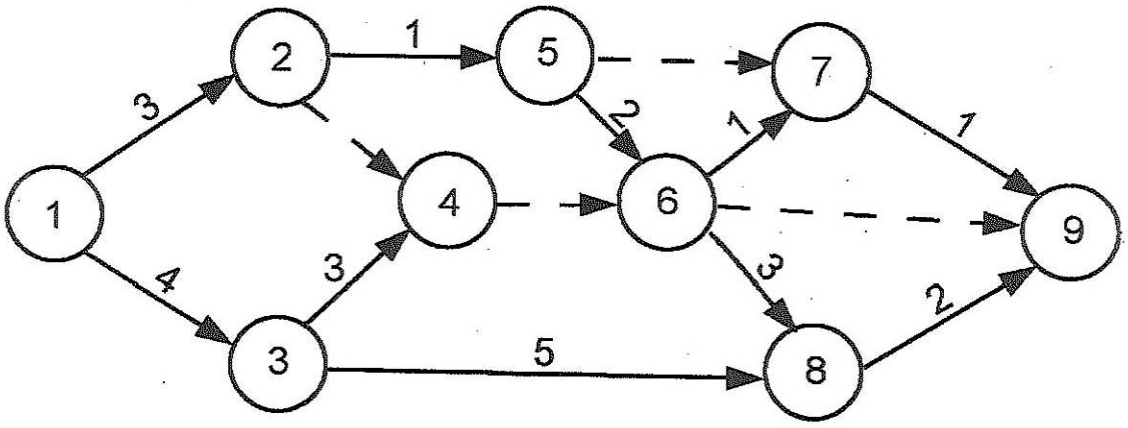


Abbildung 1: CPM-Netzwerk

Fortsetzung nächste Seite!

Aufgabe 4

a) Anwendungsfalldiagramm

Gegeben ist die folgende textuelle Beschreibung für einen **Anwendungsfall** aus dem Szenario *App-Entwicklung für eine Spedition.*

Name Übergabe der Ware an den Kunden

Akteure Speditionsfahrer, Kunde

Vorbedingung Die Tagesplanung der Spedition ist abgeschlossen.

Ablauf Dieser Anwendungsfall beschreibt einen normalen Ablauf für eine Auslieferung. Der Fahrer holt sich vorab die Information, welche Aufgaben anstehen und lässt seine Route planen. Er kündigt telefonisch die Lieferung an und stimmt sich mit dem Kunden ab. Bei Anlieferung übergibt der Fahrer die Fracht und lässt sich den Erhalt quittieren. Die erfolgreiche Auslieferung wird vom Fahrer an den zentralen Server zurückgemeldet.

Nachbedingung Der Auftrag ist erledigt.

Priorität hoch

Modellieren Sie den Anwendungsfall in einem **UML-Anwendungsfalldiagramm**!

b) Aktivitätsdiagramm

Erstellen Sie ein **UML-Aktivitätsdiagramm**, welches den Ablauf des Anwendungsfalls „Übergabe der Ware an den Kunden" aus der vorangegangenen Aufgabe modelliert! Sehen Sie für jeden Akteur sowie für die Speditionszentrale eine **Partition** vor! Simulieren Sie dabei Telefonat, Frachtübergabe und Quittierung als **Aktionsknoten** und überlegen Sie sich, welche zusätzlichen **Objektknoten** und **-flüsse** notwendig sind!

Herbst 2014

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer
Kennzahl: _____ Kennwort: _____ Arbeitsplatz-Nr.: _____	**Herbst 2014**	**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: **15**

Bitte wenden!

Thema Nr. 1

Teilaufgabe 1:

Aufgabe 1:

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

- Ein *Zugtyp* hat eine Bezeichnung, eine Höchstgeschwindigkeit und eine Kapazität.
- Es gibt *Strecken*, die einen Start- und ein Zielort haben.
- Ein *Zug* hat einen Namen und ist von einem bestimmten Zugtyp. Ein Zug kann auf einer Strecke verkehren und jede Strecke wird von mindestens einem Zug bedient. Jeder Zug, der auf einer Strecke fährt, bekommt für diese Fahrt eine eigene Bezeichnung.
- *Bahnunternehmen* besitzen Züge und bieten Verbindungen auf bestimmten Strecken an.
- *Reisende* nutzen auf Strecken angebotene Züge. Der Preis für die Strecke mit dem Zug soll protokolliert werden. Reisende haben Namen. Ein Zug verkehrt nur auf einer Strecke, wenn mindestens 10 Reisende mitfahren.

Aufgabe 2:

Wir betrachten die Schüler-Datenbank mit den folgenden Tabellen:

- Schüler(<u>Snr</u>, Vorname, Nachname, Geburtsdatum, Klasse): *Persönliche Angaben zu den Schülern.*
- Abschlussnote(<u>Snr, Fach</u>, Note): *Notenspiegel der Schüler.*
- Raum(<u>Rnr</u>, Typ, Sitzplätze): *Typ und Anzahl der Sitzplätze der Räume.*
- Raumbelegung(<u>Klasse, Fach</u>, Rnr): *Raumbelegung nach Klassen und Fächern.*

Erstellen Sie in SQL folgende Anfragen:

a) Bestimmen Sie alle Räume und die Anzahl ihrer Sitzplätze, in denen die Klasse '7b' Unterricht hat.

b) Bestimmen Sie alle Paare von Schülern (jeweils gegeben durch die beiden Schülernummern), die in dieselbe Klasse gehen und am selben Tag Geburtstag haben.

c) Bestimmen Sie die Durchschnittsnote des Schülers mit der Nummer '1'.

d) Bestimmen Sie die Fächer, in denen der Schüler mit der Nummer '1' seine schlechtesten Noten hat.

Fortsetzung nächste Seite!

Augabe 3:

Wir erweitern die Datenbank aus Aufgabe 2 um die folgenden beiden Tabellen:

- Fach(Fach, Fachbetreuer).
- Klasse(Klasse, Klassleiter)

a) Geben Sie ein ER-Diagramm für die erweiterte Datenbank an, in dem möglichst viele Relationen als Relationships modelliert sind.

b) Geben Sie geeignete Create Table-Statements mit Primär- und Fremdschlüsselbedingungen zur Erzeugung der Tabellen Schüler, Abschlussnote und Fach an.

c) Fügen Sie mit Hilfe eines SQL-Statements ein Attribut Ausbildungsrichtung mit einem geeigneten Wertebereich zur Tabelle Schüler hinzu.

d) Löschen Sie alle Abschlussnoten von Schülern der Klasse '9a'.

Aufgabe 4:

Gegeben seien die funktionalen Abhängigkeiten

$$
\begin{aligned}
F = \{ \quad & B \rightarrow AC, && \text{(i)} \\
& BE \rightarrow C, && \text{(ii)} \\
& F \rightarrow AE, && \text{(iii)} \\
& D \rightarrow EF \quad \} && \text{(iv)}
\end{aligned}
$$

der Relation $R(A,B,C,D,E,F)$.

a) Berechnen Sie eine kanonische Überdeckung F_C von F. Geben Sie alle Zwischenschritte und die angewendeten Transformationen an.

b) Geben Sie alle Kandidatenschlüssel für das Relationenschema R (jeweils mit Begründung) sowie die Nichtschlüsselattribute an.

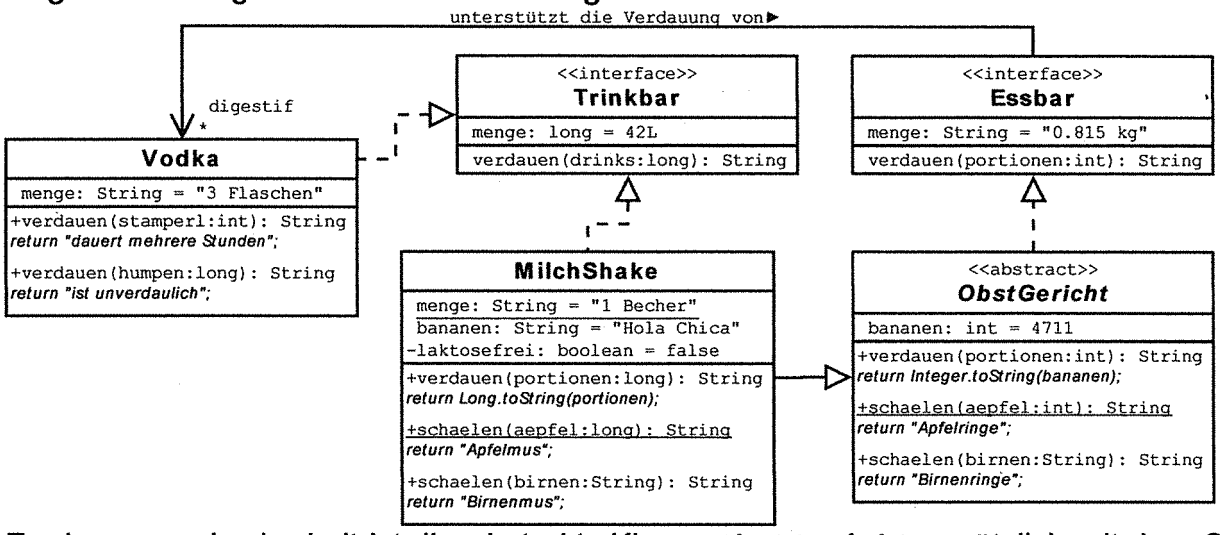
c) In welcher Normalform ist R? Geben Sie eine Begründung an.

Fortsetzung nächste Seite!

Teilaufgabe 2:

Aufgabe 1:

Gegeben sei folgendes UML-Klassendiagramm:



Zur besseren Lesbarkeit ist die *abstrakte* Klasse **`ObstGericht`** zusätzlich mit dem Stereotyp **`<<abstract>>`** gekennzeichnet. Die **`return`**-Zeile der jeweiligen Methode ist als Kommentar unmittelbar unter der Methodensignatur *kursiv* angegeben.

a) Angenommen, die Klasse **`Mahlzeit`** mit der folgenden Methode liegt mit den Klassen aus dem obigen UML-Diagramm im gleichen **package**. Geben Sie zu jeder **`System.out.println`**-Zeile die jeweils zu erwartende Ausgabe des Programms während der Ausführung an:

```
public static void main(String[] args) {
        /******************************/
        Trinkbar tv = new Vodka();
        System.out.println(tv.verdauen(1));
        /******************************/
        Trinkbar tm = new MilchShake();
        System.out.println(tm.menge);
        System.out.println(tm.verdauen(2));
        /******************************/
        ObstGericht om = new MilchShake();
        System.out.println(om.menge);
        System.out.println(om.bananen);
        System.out.println(om.verdauen(3));
        System.out.println(om.schaelen(Integer.MAX_VALUE + 123));
        System.out.println(om.schaelen("Kiwi"));
}
```

b) Geben Sie eine Implementierung der Schnittstelle **`Essbar`** in einer geeigneten objektorientierten Programmiersprache Ihrer Wahl an.

c) Geben Sie nun eine Implementierung der Klasse **`MilchShake`** in der vorangehend gewählten Programmiersprache an.

d) Jemand möchte „Island Breeze" als Unterklasse von Vodka und MilchShake modellieren, schließlich enthält der genannte Cocktail sowohl Vodka als auch Milch und Banane. Wieso ist das keine gute Idee, wenn Java die Zielsprache ist? In welcher gängigen Programmiersprache wäre das kein Problem?

Fortsetzung nächste Seite!

Aufgabe 2:

In dieser Aufgabe sollen Sie einen abstrakten Datentypen (kurz: ADT) **`Stechuhr`** zur Arbeitszeiterfassung mit folgenden Vereinfachungen spezifizieren und umsetzen:

- Die Stechuhr wird jeweils um Mitternacht ausgelesen und zurückgesetzt – zu einer Zeit, zu der sich kein Mitarbeiter mehr im Gebäude aufhalten darf. Daher muss beim Ermitteln der **`arbeitszeit`** (des heutigen Tages) nur das letzte Paar an Ein- und Ausbuchungsvorgängen berücksichtigt werden.
- Ein- und Ausbuchungszeiten sowie Arbeitszeiten werden in der Einheit „Minuten seit 0:00 Uhr des laufenden Tages" vom Typ **`Long`** verarbeitet und gespeichert.
- Mitarbeiter werden über ihren Personalstammdatensatz identifiziert, der garantiert ein-eindeutig ist und dessen ADT (vereinfacht) wie folgt aussieht:

adt *Mitarbeiter*
sorts *Mitarbeiter, String, Long*
ops

einstellen:	*String* × *Long*	→ *Mitarbeiter*	// Konstruktor
name:	*Mitarbeiter*	→ *String*	
id:	*Mitarbeiter*	→ *Long*	// eindeutige Kennung

axs

name(einstellen(n, i)) = n
id(einstellen(n, i)) = i
id(einstellen(n, i)) = id(einstellen(m, j)) ↔ n = m

end *Mitarbeiter*

adt *Stechuhr*
sorts *Stechuhr, Mitarbeiter, String, Long*
ops

init:	→ *Stechuhr*	// Konstruktor

...

axs

...

end *Stechuhr*

Für die ADTs String und Long stehen alle bekannten Operationen zur Verfügung.

a) Ergänzen Sie den obigen Ausschnitt der algebraischen Spezifikation des ADTs **`Stechuhr`** um die notwendigen Signaturen und Axiome folgender Operationen:
   i) **`einbuchen`**: vermerkt die Ankunftszeit eines **`Mitarbeiters`** (beide als Parameter übergebenen) in „Minuten seit 0:00 Uhr des laufenden Tages"
   ii) **`ausbuchen`**: wie **`einbuchen`**, nur für das Verlassen des Gebäudes
   iii) **`leeren`**: setzt die **`Stechuhr`** zurück (nach dem Auslesen um Mitternacht)
   iv) **`arbeitszeit`**: bestimmt die heutige Arbeitszeit (Datentyp: Long) eines (als Parameter übergebenen) **`Mitarbeiters`**

   Hinweis: Sie dürfen bei Bedarf zusätzlich auch Hilfsoperationen ergänzen!

b) Geben Sie eine Implementierung des ADT **`Mitarbeiter`** in einer geeigneten objektorientierten Programmiersprache Ihrer Wahl an. Der Konstruktor **`einstellen`** des ADTs soll in der implementierten Klasse sowohl durch den Konstruktor selbst als auch durch eine gleichnamige Klassenmethode umgesetzt werden. Stellen Sie in der Klasse **`Mitarbeiter`** außerdem eine Methode **`Long getNewID()`** bereit, die bei jedem Aufruf eine neue eindeutige ID generiert.

Fortsetzung nächste Seite!

Aufgabe 3:

Gegeben seien folgende Java-Methode sowie ihre Vor- und Nachbedingung:

```
public static long funI(final int n) {
        /* P */long a = 1, b = 0;
        for (int i = 0; i < n - 1; i++) {
                b += a;
                a *= 3;
        }/* R */
        long ret = 3 * a - 4 * b;/* Q */
        return ret;
}
```

$P := n \geq 1$

$Q := ret = 3^n - 4 \cdot \sum_{j=0}^{n-2} 3^j$

Zur Vereinfachung nehmen Sie bitte im Folgenden an, dass die verwendeten Datentypen unbeschränkt sind und daher keine Überläufe auftreten können.

a) Welche der folgenden Bedingungen ist eine zum Beweisen der Korrektheit der Methode mittels *wp*-Kalkül (Floyd-Hoare-Kalkül) sinnvolle Schleifeninvariante?

i) $i = 0 \wedge a = 1 \wedge b = 0$

ii) $a = 3^i \wedge b = \sum_{j=0}^{i-1} 3^j \wedge i \leq n-1$

iii) $a = 3^{n-1} \wedge b = \sum_{j=0}^{n-2} 3^j$

iv) $true$

v) $ret = 3^n - 4 \cdot \sum_{j=0}^{n-2} 3^j$

b) Zeigen Sie *formal* mittels *wp*-Kalkül, dass die von Ihnen gewählte Bedingung unmittelbar vor Beginn der Schleife gilt, wenn zu Beginn der Methode die Anfangsbedingung P gilt.

c) Zeigen Sie *formal* mittels *wp*-Kalkül, dass die von Ihnen gewählte Bedingung tatsächlich eine Invariante der Schleife ist.

d) Bestimmen Sie *formal* mittels *wp*-Kalkül die schwächste Zusicherung *R*, die unmittelbar nach dem Ende der Schleife gelten muss, so dass am Ende der Methode die Nachbedingung Q erfüllt wird. Zeigen Sie *formal* mit Hilfe der von Ihnen gewählten Invariante, dass die Zusicherung *R* auch tatsächlich unmittelbar nach der Ausführung der Schleife gilt.

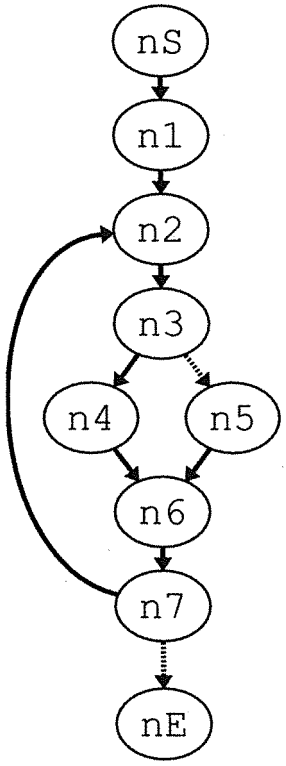
e) Beweisen Sie, dass die Methode immer terminiert. Geben Sie dazu eine Terminierungsfunktion an und begründen Sie kurz Ihre Wahl.

Fortsetzung nächste Seite!

Aufgabe 4:

Gegeben sei folgende Methode **double** p(**double** e) und ihr Kontrollflussgraph:

```
public static double p(double e) {
      double r = 0.0d;
      boolean v = false;
      double n = 1.0d;
      do {
            if (v) {
                  r -= 4 / n;
            } else {
                  r += 4 / n;
            }
            v = !v;
            n += 2;
      } while (e < 4 / n);
      return r;
}
```



a) Geben Sie je einen Repräsentanten aller Pfadklassen **im Kontrollflussgraphen** an, die zum Erzielen einer vollständigen
   i) Verzweigungsüberdeckung (Branch-Coverage, C_1)
   ii) Schleife-Inneres-Überdeckung (Boundary-Interior-Coverage, $C_{\infty,2}$)

   mit **minimaler** Testfallanzahl und möglichst kurzen Pfaden genügen würden.

b) Bestimmen Sie anhand des Kontrollflussgraphen des obigen Code-Fragments die maximale Anzahl linear unabhängiger Programmpfade, also die zyklomatische Komplexität nach McCabe.

c) Kann für dieses Modul eine 100%-ige Pfadüberdeckung erzielt werden? Begründen Sie kurz Ihre Antwort.

d) Geben Sie zu jedem Knoten die jeweilige Datenflussannotation (defs bzw. c/p-uses, sofern überhaupt vorhanden) für jede betroffene Variable in der zeitlichen Reihenfolge ihres Auftretens zur Laufzeit an.

Thema Nr. 2

Teilaufgabe 1:

Aufgabe 1:

1.1 Beantworten Sie folgende Fragen.

a) Welche Auswirkungen hat eine Transaktion, wenn sie mit Abort abgebrochen wird? Begründen Sie Ihre Antwort.
b) Grenzen Sie die Begriffe Datenbankverwaltungssystem und Datenbank voneinander ab.
c) Was ist das Ergebnis einer Projektion und was einer Selektion (= Restriktion) in der relationalen Algebra?
d) Was bedeutet die Abkürzung SQL ausgeschrieben? Worin unterscheidet sich diese Sprache von Programmiersprachen wie Java/C/Basic/... und welche Vorteile ergeben sich daraus?
e) Wozu dient der Primärschlüssel und welche Eigenschaften muss er erfüllen?

1.2 Geben Sie für jede der folgenden Aussagen an, ob sie richtig oder falsch ist und begründen Sie Ihre Antwort.

a) Das Anlegen von Views führt zu Redundanzen in der Datenbank.
b) Bei einer 1:1-Beziehung braucht nur eine der beiden Relationen einen Primärschlüssel.
c) In einer Relation sind die Tupel nicht zwingend in der Einfügereihenfolge abgelegt.
d) Eine Datenbankoperation muss die Datenbank in einem konsistenten Zustand hinterlassen.
e) Die SQL-Anweisung *select* gehört zur Data-Definition-Language (DDL).

Aufgabe 2:

a) E/R-Modell

Im Folgenden finden Sie die Beschreibung der Inventarverwaltungssoftware eines Unternehmens. Erstellen Sie zu dieser ein ER-Diagramm. Geben Sie dabei auch die Kardinalitäten an. Verwenden Sie nur die im Folgenden genannten Attribute als Schlüsselkandidaten:

Mitarbeiter, denen eine eindeutige Personalnummer zugeordnet ist, arbeiten in genau einer Abteilung, die firmenweit durch eine Abteilungsnummer identifiziert wird und an ein oder mehreren Standorten untergebracht ist. Jeder Standort hat eine Straße, eine Hausnummer, eine Postleitzahl sowie einen Ort und wird durch diese Adresse eindeutig gekennzeichnet. An den einzelnen Standorten kann es Räume geben, die durch eine Raumnummer bezeichnet werden, die am jeweiligen Standort eindeutig ist.

Jeder Mitarbeiter kann Gegenstände besitzen, für die er alleine verantwortlich ist. Alle Gegenstände haben eine eindeutige Inventarnummer und sind von einem bestimmten Inventartyp, der eine eindeutige Bezeichnung hat (z.B. Laptop Modell 500). Es gibt zwei Arten von Inventartypen: Bewegliches Inventar und unbewegliches Inventar. Zu beweglichen Inventartypen wird das Gewicht gespeichert. Unbewegliche Inventartypen haben eine Position, die aus der geographischen Breite und der geographischen Länge besteht.

Gegenstände können von einem Raum in einen anderen Raum umgezogen werden. Für Umzugswünsche kann eine Priorität vergeben werden, um festzulegen, wie dringend der Gegenstand am neuen Ort gebraucht wird. Es kann nur einen Umzugswunsch pro Gegenstand geben.

Fortsetzung nächste Seite!

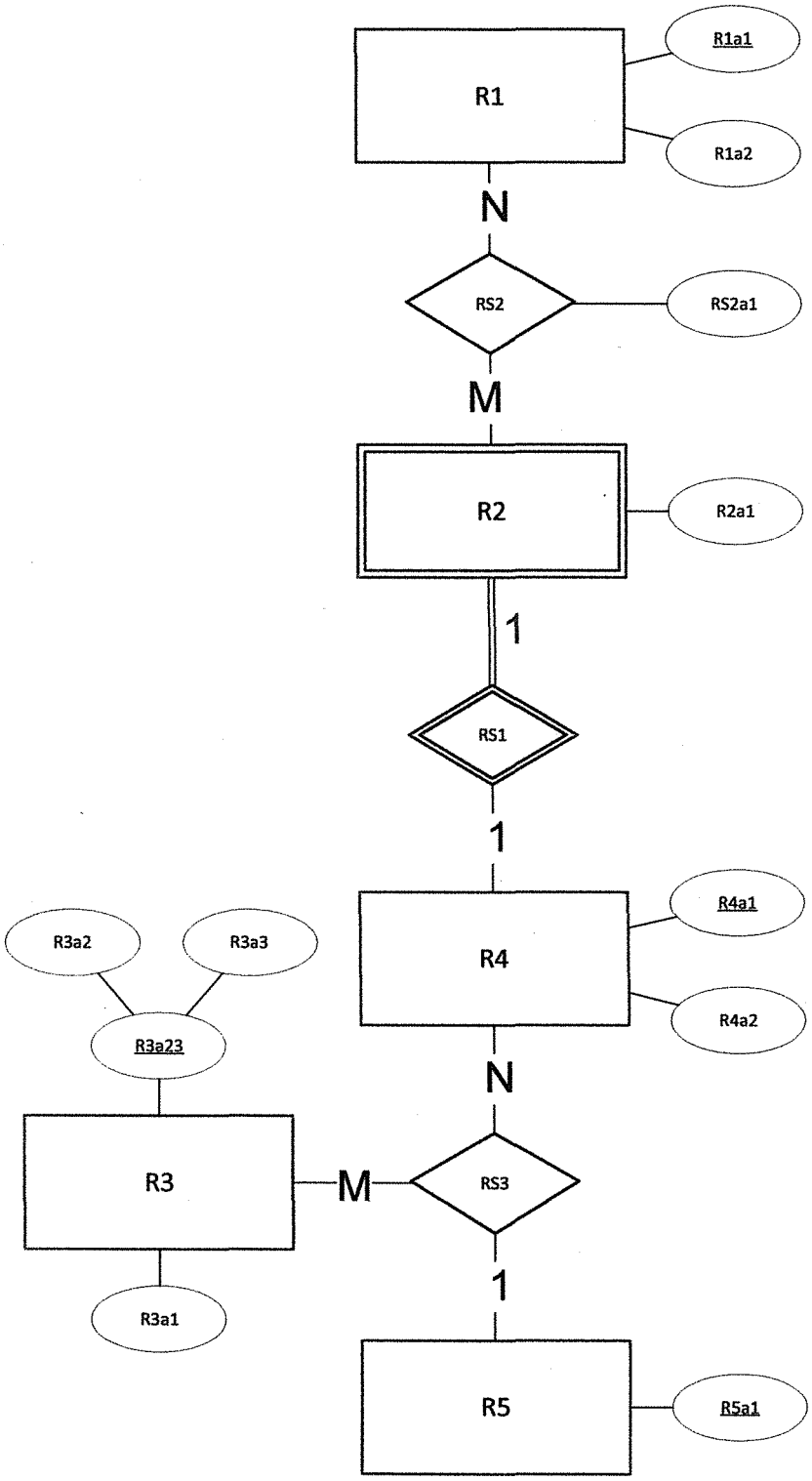
b) Abbildung auf Relationen

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, 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, Fremdschlüsselattr1[Relation2], (Fremdschlüsselattr2, Fremdschlüsselattr3)[Relation3])



Fortsetzung nächste Seite!

Aufgabe 3:

Gegeben ist folgender Auszug aus einer Datenbank zur Verwaltung von Kochrezepten:

```
Autor(ID, Vorname, Nachname)
Rezept(Rezeptnummer, Name, Verfasser[Autor], Beschreibung)
Zutat(Bezeichnung, Einkaufspreis)
Rezept_Zutat(Rezept[Rezept], Zutat[Zutat], Menge)
```

Primärschlüssel sind unterstrichen, Fremdschlüssel dadurch gekennzeichnet, dass hinter ihnen in eckigen Klammern der Name der Relation angegeben ist, auf die sie sich beziehen.
Verwenden Sie bei der Formulierung der Anfragen nur standardkonformes SQL. Die Verwendung von Views ist erlaubt und empfohlen.

a) Schreiben Sie SQL-Statements, die die Tabellen zu den oben angegebenen Relationen anlegen. Sie können beliebige sinnvolle Datentypen wählen. Denken Sie auch an die Schlüssel und referentiellen Beziehungen.
b) Schreiben Sie ein SQL-Statement, das Vor- und Nachname des Autors mit der ID 42 ausgibt.
c) Schreiben Sie ein SQL-Statement, das alle Autoren, die weniger als 10 Rezepte verfasst haben, ausgibt, sortiert nach der Anzahl der von ihnen verfassten Rezepte. Ausgegeben werden soll die Anzahl der verfassten Rezepte, sowie Vor- und Nachname des Autors. Auch Autoren, die keine Rezepte verfasst haben, sollen in der Ausgabe enthalten sein.
d) Schreiben Sie eine SQL-Anfrage, die die Namen aller Rezepte zusammen mit der Anzahl verschiedener für sie benötigter Zutaten ausgibt, absteigend sortiert nach Anzahl der Zutaten.
e) Schreiben Sie eine SQL-Anfrage, die die Top-Ten der Rezepte mit den höchsten Zutatenkosten ausgibt. Die Kosten eines Rezepts berechnen sich als Summe über das Produkt aus Menge und Einkaufspreis aller Zutaten. Ausgegeben werden soll der Name des Rezepts, die Kosten und der Rang (1 = teuerstes Rezept bis 10), aufsteigend sortiert nach Rang. Gehen Sie davon aus, dass keine zwei Rezepte gleiche Kosten aufweisen.
   Verwenden Sie keine nicht-standardisierten Erweiterungen, wie `rownum` und ebenfalls nicht das Schlüsselwort `fetch`!

Aufgabe 4:

a) Von wann bis wann werden von einer Transaktion verwendete Sperren gehalten (dynamisches Sperren)? Begründen Sie, warum genau diese Zeitpunkte sinnvoll sind.
b) Bei der Transaktionsverarbeitung kann es zu Verklemmungen (Deadlocks) kommen. Erklären Sie anhand eines Beispiels, wie diese entstehen können. Erläutern Sie, wie ein Datenbankverwaltungssystem diese Situation erkennen und was es tun kann, um eine Verklemmung aufzulösen. Gehen Sie auch auf die entstehenden Konsequenzen ein.

Fortsetzung nächste Seite!

Teilaufgabe 2:

Aufgabe 1:

In Abbildung 1 ist eine Version des Algorithmus von Euklid zur Berechnung des größten gemeinsamen Teilers zweier ganzer, positiver Zahlen angegeben. Der Algorithmus nimmt als Eingabe eine ganze Zahl $A > 0$ und eine ganze Zahl $B > 0$. Nach Ausführen des Algorithmus soll der größte gemeinsame Teiler von A und B in der Variable a stehen. Die Funktion $ggt(A,B)$ bezeichnet dabei eine Funktion die den größten gemeinsamen Teiler zweier ganzer, positiver Zahlen A und B berechnet.

```
{A > 0 ∧ B > 0}
a := A;
b := B;
while (a ≠ b) {
   if (a > b) {
      a := a – b;
   } else {
      b := b – a;
   }
}
{a = ggt(A,B)}
```

Abbildung 1

Zeigen Sie die korrekte Funktionsweise des Algorithmus. Führen Sie dabei folgende Schritte aus[1]:

1. Geben Sie die Schleifeninvariante an.
2. Annotieren Sie das Programm unter Verwendung von Hoare Logik.
3. Begründen Sie für jede Anwendung der Konsequenzregel, warum die Implikation gilt.

Fortsetzung nächste Seite!

[1] Sie dürfen dabei die Funktion ggt verwenden.

Aufgabe 2:

Sie sollen das Design für ein einfaches Wahlsystem entwerfen. Das System soll dabei die Verteilung der Stimmen auf die einzelnen Parteien ermöglichen. Zusätzlich soll es verschiedene Darstellungen dieser Daten erlauben: Eine Tabelle, in der die Daten gelesen und auch eingegeben werden können, und ein Diagramm als alternative Darstellung der Informationen. Das System soll mit dem Model-View-Controller Muster modelliert werden.

1. Beschreiben Sie das Model-View-Controller Muster:
    a. Beschreiben Sie das Problem, welches das Muster adressiert.
    b. Beschreiben Sie die Aufgaben der Komponenten, die im Muster verwendet werden.
2. Modellieren Sie das System unter Anwendung des Musters:
    a. Entwerfen Sie ein UML Klassendiagramm.
    b. Implementieren Sie eine setup-Methode, die das Objektmodell erstellt.

Fortsetzung nächste Seite!

Aufgabe 3:

Im Folgenden ist ein Algorithmus angegeben, der für eine positive Zahl *until* die Summe aller Zahlen bildet, die kleiner als *until* und Vielfache von vier oder sechs sind. Für nicht positive Zahlen soll 0 zurückgegeben werden. Der Algorithmus soll also folgender Spezifikation genügen:

$$until > 0 \Rightarrow specialSums(until) = \sum \{y \mid 0 < y < until \wedge (y\%4 = 0 \vee y\%6 = 0)\}$$

$$until \leq 0 \Rightarrow specialSums(until) = 0$$

wobei % den Modulo-Operator bezeichnet.

```
public static long specialSums (int until)
   1. long sum = 0;
   2. if (until > 0)  {
   3.     for (int i = 1; i <= until; i++)  {
   4.         if (i % 4 == 0 || i % 6 == 0)  {
   5.             sum += i;
   6.         }
   7.     }
   8. }
   9. return sum;
```

Abbildung 2: Algorithmus specialSums

Beachten Sie, dass der Algorithmus nicht der Spezifikation genügt. Der Fehler liegt in der Bedingung der for-Schleife (Zeile 3). Der Fehler kann jedoch einfach korrigiert werden indem die Bedingung

$$i \leq until \quad \text{in} \quad i < until$$

geändert würde.

1) Zeichnen Sie das zum Programm in Abbildung 2 gehörige Ablaufdiagramm.
2) Schreiben Sie einen Testfall, der das Kriterium „100% Anweisungsüberdeckung“ erfüllt, aber den Fehler trotzdem nicht aufdeckt.
3) Schreiben Sie einen Testfall, der das Kriterium „100% Zweigüberdeckung“ erfüllt, aber den Fehler trotzdem nicht aufdeckt.
4) Schreiben Sie einen Testfall, der den Fehler aufdeckt. Berechnen Sie Anweisungsüberdeckung und Zweigüberdeckung ihres Testfalles.
5) Es ist nicht immer möglich vollständige Pfadüberdeckung zu erreichen. Geben Sie einen gültigen Pfad des Programmes an der nicht erreichbar ist.

Ein Testfall kann als Menge von Paaren dargestellt werden, wobei jedes Paar *(I,O)* die Eingabe *I* und die zu dieser Eingabe erwartete Ausgabe *O* darstellt.

Fortsetzung nächste Seite!

Aufgabe 4:

a) Erläutern Sie den Unterschied zwischen der partiellen Korrektheit und der totalen Korrektheit eines Programms.

b) Geben Sie die allgemeinen Regeln zum Nachweis der partiellen Korrektheit für die zweiseitig bedingte Anweisung und für die Schleife an.

c) Führen Sie für das nachfolgende Programm den Nachweis der partiellen Korrektheit. Verwenden Sie dazu

$$I = \left\{(x \geq c^2 \wedge (a = (c+1)^2) \wedge (c \geq 0)\right\}$$

als Schleifeninvariante. Alle Variablen repräsentieren ganze Zahlen.

// P = {*true*}

s_1: c = 0;

s_2: a = c + 1;

if (y < 0) // B_1 = (y < 0)

s_3: x = -y;

else

s_4: x = y;

while (a < x + 1) { // B_2 = {a < x + 1}

s_5: c = c + 1;

s_6: a = a + 2c + 1;

}

s_7: r = 2c;

// $Q = \left\{(r^2 \leq 4x) \wedge (x < (0{,}5r+1)^2)\right\}$

d) Geben Sie eine geeignete Funktion an, mit der ein Terminierungsbeweis für die Schleife durchgeführt werden kann. Welche Anforderungen muss die Funktion erfüllen?

e) Geben Sie für die obige Schleife jeweils eine zu starke und eine zu schwache Invariante an. Erklären Sie jeweils kurz, welche Beweisteile sich mit den ungeeigneten Invarianten nicht durchführen lassen.

Fortsetzung nächste Seite!

Aufgabe 5:

Lösen Sie folgende Aufgabe in einer objektorientierten Programmiersprache Ihrer Wahl. Ein Regal habe 5 Fächer, in die je 30 Disks passen. Das Regal beinhaltet DVDs, CDs und BDs der Genres Musik, Action, Komödie, Thriller und Fantasy. Jede Disk ist mit 1 bis 10 bewertet, wobei 10 für sehr gut steht.

- Deklarieren Sie einen Aufzählungsdatentyp für den Typ der Disk. Deklarieren Sie einen weiteren Aufzählungsdatentyp für das Genre der Disk.
- Deklarieren Sie eine Klasse Disk, die den Typ, das Genre, die Bewertung, sowie den Titel der Disk speichert.
- Deklarieren Sie ein Array Regal, mit den Abmessungen des oben genannten Regals, das Disks enthält. Initialisieren Sie das Array als leeres Regal.
- Initialisieren Sie die Bewertung einer Disk. Schreiben sie dazu eine rekursive Methode erstelleStdBewertung, der ein Genre einer Disk übergeben wird und die die Bewertungen für alle Disks nach folgenden Regeln bezüglich des Genres vergibt:
  - Eine Disk mit Genre Musik wird mit einer 3 bewertet.
  - Komödien werden mit 2 bewertet.
  - Thriller werden mit dem zweifachen einer Komödie bewertet
  - Ein Fantasy-Film wird mit dem 1,5 fachen eines Thrillers bewertet.
  - Ein Actionfilm wird wie ein Thriller bewertet.
- Schreiben Sie eine Methode mittlereBewertung, die die mittlere Bewertung der Disks im Regal berechnet.

Frühjahr 2015

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer

Kennzahl: _____

Kennwort: _____

Arbeitsplatz-Nr.: _____

Frühjahr 2015

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: **12**

Bitte wenden!

Thema Nr. 1

Teilaufgabe 1

DB.1 (Datenmodellierung: ER-Modell, SQL)

In einer *Musik-Datenbank* sollen folgende Informationen zu Interpreten und deren CDs modelliert werden:

- Zu einem Interpreten soll eine eindeutige ID, der Name, das Jahr seines Bühnenstarts, seine Geschäftsadresse sowie sein Musikgenre angegeben sein. Das Musikgenre kann mehrere Werte umfassen (mehrwertiges Attribut).

- Eine CD hat eine eindeutige ID, einen Namen (Titel), einen Interpreten, ein Erscheinungsdatum und bis zu 20 Positionen (Musikstücke). An jeder Position steht ein Musikstück. Für dieses ist der Titel und die Länge in Sekunden angegeben.

- Eine CD kann Auszeichnungen – z. B. vom Typ goldene Schallplatte oder Emmy bekommen. Ebenso kann auch ein einzelnes Musikstück Auszeichnungen bekommen.

a) Modellieren Sie das oben dargestellte Szenario möglichst vollständig in einem ER–Modell. Verwenden Sie, wann immer möglich, (binäre oder auch höherstellige) Relationships. Modellieren Sie Musikstücke in einem schwachen Entity-Typen.

b) Übertragen Sie Ihr ER–Modell – bis auf die Typen zu den Auszeichnungen – ins relationale Datenmodell. Erstellen Sie dazu Tabellen mit Hilfe von CREATE TABLE–Statements in *SQL*. Berücksichtigen Sie die Fremdschlüsselbeziehungen.

c) Es soll die Integritätsbedingung eingehalten werden, so dass die Anzahl der Positionen auf einer CD höchstens 20 ist. Schreiben Sie ein SELECT–Statement, das diese Integritätsbedingung überprüft, indem es die verletzenden CDs ausgibt.

d) Geben Sie geeignete INSERT–Statements an, die in alle beteiligten Tabellen jeweils mindestens ein Tupel einfügen, so dass alle Integritätsbedingungen erfüllt sind, nachdem alle Einfügungen ausgeführt wurden. Lediglich zu den Auszeichnungen müssen keine Tupel eingefügt werden.

Fortsetzung nächste Seite!

DB.2 (Datenbankanfragen und – änderungen in SQL)

Formulieren Sie in SQL die folgenden Anfragen, Views bzw. Datenmanipulations–Statements an Teile der *Musik–Datenbank* aus Teilaufgabe DB.1:

Interpret (Interpreten_ID, Name, Bühnenstart, Geschäftsadresse),
CD (CD_ID, Name, Interpreten_ID, Erscheinungsdatum),
Musikstück (CD_ID, Position, Titel, Länge),
Auszeichnung_CD (CD_ID, Typ),
Auszeichnung_Stück (CD_ID, Position, Typ).

a) Welche CDs hat 'Adele' herausgebracht? Geben Sie die Namen der CDs aus.

b) Geben Sie für alle Interpreten – gegeben durch die ID und den Namen – die Anzahl ihrer veröffentlichten CDs an.

c) Geben Sie die Länge des längsten Musikstücks auf der CD mit dem Namen 'Thriller' des Interpreten 'Michael Jackson' an.

d) Geben Sie die Namen aller Interpreten aus, die eine Auszeichnung für eine CD oder eines ihrer Musikstücke bekommen haben.

e) Fügen Sie ein, dass 'Adele' einen 'Emmy' für ihre CD mit dem Namen 'Adele 21' bekommen hat.

f) Ändern Sie die Geschäftsadresse von 'Genesis' auf 'Hollywood Boulevard 13, Los Angeles'.

DB.3 (Funktionale Abhängigkeiten und Zerlegungen)

Gegeben sei das Relationenschema $R=(U,F)$ mit der Attributmenge

$$U=\{A,B,C,D,E\}$$

und der folgenden Menge F von funktionalen Abhängigkeiten:

$$F=\{A\rightarrow B, ABC\rightarrow D, D\rightarrow BC\}.$$

a) Geben Sie alle Schlüssel für das Relationenschema R (jeweils mit Begründung) sowie die Nichtschlüsselattribute an.

b) Ist R in 3NF bzw. in BCNF? Geben Sie jeweils eine Begründung an.

c) Geben Sie eine Basis G von F an. Zerlegen Sie R mittels des Synthesealgorithmus in ein 3NF–Datenbankschema. Es genügt, die resultierenden Attributmengen anzugeben.

Fortsetzung nächste Seite!

Teilaufgabe 2

Aufgabe 1 (Klassendiagramm)

Erstellen Sie zu der folgenden Beschreibung eines automatisierten Lagers ein UML-Klassendiagramm, das Attribute, Assoziationen mit Kardinalitäten, Rollennamen und Namen sowie Methoden enthält. Setzen Sie dabei insbesondere die Konzepte der Vererbung, der abstrakten Klassen und der Kompositionsbeziehungen sinnvoll ein.

Tipp: Zeichnen Sie sie Kompositionsbeziehungen zuletzt ein.

Klassen und Assoziationen:

- Ein Lager beinhaltet ein Transportsystem, Bearbeiter, Orte und Artikel sowie Jobs, die ausgeführt werden sollen.
- Bearbeiter sind entweder Roboter oder Arbeiter.
- Orte sind entweder Ausgaben oder Lagerplätze.
- Das Transportsystem kontrolliert alle vorhandenen Roboter, kennt die Lagerplätze und verwaltet die Jobs.
- Ein Bearbeiter ist an genau einem Ort, transportiert maximal einen Artikel und ihm ist maximal ein Job zugewiesen.
- Ein Artikel ist an maximal einem Ort zu finden und hat einen Namen.
- Einem Job ist ein Artikel, der geholt werden soll, der Ort, an dem sich der Artikel befindet, und der Ort, zu dem er gebracht werden soll, bekannt.

Öffentliche Methoden:

- Jobs werden vom Transportsystem erzeugt. Zur Erzeugung eines Jobs werden dem Transportsystem der Artikel, der geholt werden soll, der Ort, an dem sich der Artikel befindet, und der Ort, zu dem er gebracht werden soll, übergeben.
- Bearbeiter können sich zu einem Ort bewegen, der ihnen genannt wird (Parameter der Methode).
- Bearbeiter können einen Job ausführen, der ihnen genannt wird.
- An der Ausgabe kann man Artikel bestellen; dazu nennt man der Ausgabe den Namen des Artikels.
- Die Ausgabe meldet wenn ein Artikel nicht verfügbar ist. Dabei wird zurückgegeben, ob der Artikel verfügbar ist oder nicht.
- Im Lagerplatz wird ein Artikel geladen; dazu muss der Artikel angegeben werden.

Fortsetzung nächste Seite!

Aufgabe 2 (Zustandsdiagramm)

Erstellen Sie ein UML-Zustandsdiagramm für einen Radiotuner. Mit dem Radiotuner können Sie ein Radioprogramm auf der Frequenz f empfangen. Beim Einschalten wird f auf 87,5 MHz gesetzt und der Tuner empfängt. Sie können nun die Frequenz in 0,5 MHz Schritten erhöhen oder senken. Bitte beachten Sie, dass das Frequenzband für den Radioempfang von 87,5 MHz bis 108 MHz reicht. Sobald f 87,5 MHz unter- bzw. 108 MHz überschreitet, soll f auf 108 MHz bzw. 87,5 MHz gesetzt werden. Weiterhin kann ein Suchmodus gestartet werden, der automatisch die Frequenz erhöht, bis ein Sender empfangen wird. Wird während des Suchmodus die Frequenz verändert oder erneut Suchen ausgeführt, dann wird die Suche beendet (und, je nach Knopf, ggf. noch die Frequenz um 0,5 MHz verändert).
Die Klasse RadioTuner besitzt folgende Methoden:

- tunerOn()
- tunerOff()
- increaseFrequency()
- decreaseFrequency()
- seek()
- hasReception()

Hinweis: Die Hilfsmethode hasReception() liefert true zurück, genau dann wenn ein Sender empfangen wird.

Aufgabe 3 (Objektsorientierte Programmierung und Entwurfsmuster)

In dieser Aufgabe implementieren Sie ein konzeptionelles Datenmodell für eine Firma, die Personendaten von Angestellten und Kunden verwalten möchte. Gegeben seien dazu folgende Aussagen:

- Eine Person hat einen Namen und ein Geschlecht (männlich oder weiblich).
- Ein Angestellter ist eine Person, zu der zusätzlich das monatliche Gehalt gespeichert wird.
- Ein Kunde ist eine Person, zu der zusätzlich eine Kundennummer hinterlegt wird.

a) Geben Sie in einer objektorientierten Programmiersprache Ihrer Wahl (geben Sie diese an) eine Implementierung des aus den obigen Aussagen resultierenden konzeptionellen Datenmodells in Form von **Klassen** und **Interfaces** an. Gehen Sie dabei wie folgt vor:

   - Schreiben Sie ein Interface *Person* sowie zwei davon erbende Interfaces *Angestellter* und *Kunde*. Die Interfaces sollen jeweils lesende Zugriffsmethoden (Getter) für die entsprechenden Attribute (Name, Geschlecht, Gehalt, Kundennummer) deklarieren.
   - Schreiben Sie eine abstrakte Klasse *PersonImpl*, die das Interface *Person* implementiert. Für jedes Attribut soll ein Objektfeld angelegt werden. Außerdem soll ein Konstruktor definiert werden, der alle Objektfelder initialisiert.

Fortsetzung nächste Seite!

- Schreiben Sie zwei Klassen *AngestellterImpl* und *KundeImpl*, die von *PersonImpl* erben und die jeweils dazugehörigen Interfaces implementieren. Es sollen wiederum Konstruktoren definiert werden, die alle Objektfelder initialisieren und dabei auf den Konstruktor der Basisklasse *PersonImpl* Bezug nehmen.

b) Verwenden Sie das Entwurfsmuster **Adapter**, um zu ermöglichen, dass vorhandene Angestellte die Rolle eines Kunden einnehmen können. Der Adapter soll eine zusätzliche Klasse sein, die das Kunden-Interface implementiert. Wenn möglich, sollen Methodenaufrufe an den adaptierten Angestellten delegiert werden. Möglicherweise müssen Sie neue Objektfelder einführen.

c) Verwenden Sie das Entwurfsmuster **Simple Factory**, um die Erzeugung von Angestellten, Kunden, sowie Adapter-Instanzen aus Aufgabe (b) zu vereinheitlichen. Die entsprechende Erzeugungs-Methode soll neben einem Typ-Diskriminator (z.B. einem Aufzählungstypen oder mehreren booleschen Werten) alle Parameter übergeben bekommen, die für den Konstruktor irgendeiner Implementierungsklasse des Interface Person notwendig sind.

Hinweis: Um eine Adapter-Instanz zu erzeugen, müssen Sie möglicherweise zwei Konstruktoren aufrufen.

Thema Nr. 2

Teilaufgabe 1

Aufgabe 1 Modellierung

Ein junges Start-Up möchte die Welt der Sozialen Netzwerke revolutionieren. Dazu entwerfen sie zunächst ein Modell für die Datenbank. Folgendes Szenario soll modelliert werden:

- Zu einer Person werden Vor- und Nachname gespeichert, außerdem das Datum der Registrierung.
- Personen können miteinander befreundet sein. Ein Attribut Status beschreibt die Freundschaftsbeziehung näher.
- Eine Person kann beliebig viele öffentliche Status-Updates erstellen, die ihr eindeutig zugeordnet werden. Ein Status-Update umfasst einen Text und den Erstellungszeitpunkt.
- Ein Status-Update kann optional genau einen Anhang besitzen. Jeder Anhang ist dabei eindeutig einem Status-Update zugeordnet. Bei jedem Anhang handelt es sich entweder um ein Bild oder um eine Online-Ressource. Zu einem Bild werden ein lokaler Pfad und die Dateigröße gespeichert. Zur Online-Ressource werden die URL und ein Titel-Text hinterlegt.
- Zu einem Bild kann angegeben werden, welche Personen darauf abgebildet sind. Zusätzlich zu dieser Information sollen keine weiteren Details gespeichert werden.
- Personen können sich gegenseitig Nachrichten schreiben. Zu jeder Nachricht soll der Sender, der Empfänger, der Text der Nachricht und ein Zeitstempel gespeichert werden.
- Verwenden Sie Attribute ID als Primärschlüssel, um die Entitäten zu identifizieren.

a) Entwerfen Sie für das beschriebene Szenario ein ER-Modell in Chen-Notation. Bestimmen Sie hierzu:

- Die Entity-Typen, die Relationship-Typen und jeweils deren Attribute.
- Die Funktionalitäten der Relationship-Typen. Achten Sie darauf alle Totalitäten einzutragen.

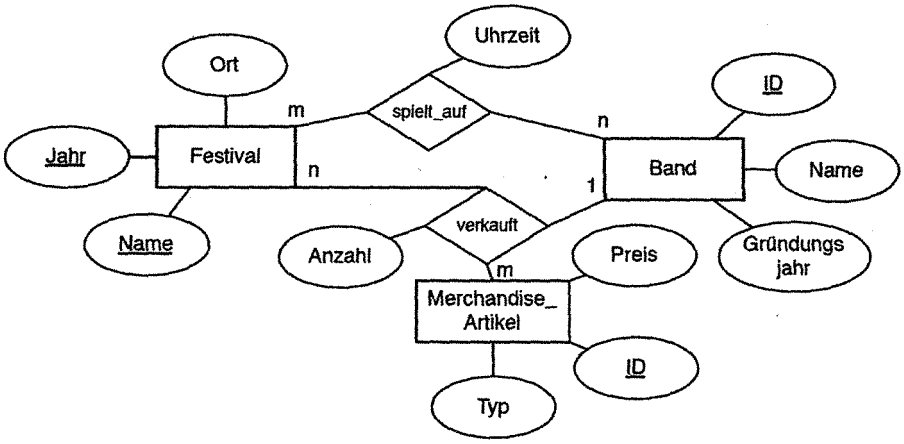
b) Konvertierung einer Generalisierung:

- Beschreiben Sie kurz die 4 Optionen, wie eine Generalisierung im ER-Diagramm in ein relationales Schema überführt werden kann.
- Gehen Sie dazu kurz darauf ein, unter welchen Bedingungen diese Optionen anwendbar sind.

Fortsetzung nächste Seite!

Aufgabe 2 SQL-Anfragen

Gegeben sei folgendes ER-Modell, welches eine Festival-Datenbank beschreibt. Es beinhaltet Bands, die auf Festivals spielen und dort Merchandise-Artikel verkaufen. Festivals werden durch ihren Namen, den Veranstaltungsort und das Jahr, in dem sie stattfinden, charakterisiert. Auf Festivals geben verschiedene Bands zu bestimmten Uhrzeiten Konzerte. Bands werden durch eine ID identifiziert und besitzen zusätzlich einen Namen und ihr Gründungsjahr. Die Merchandise-Artikel besitzen ebenfalls eine eindeutige ID, einen Typ (z.B. T-Shirt oder CD) und einen Preis (in Euro):



Gehen Sie von folgendem dazugehörigen relationalem Schema aus:

Festival {*[Name, Jahr, Ort]*}
Band {*[ID, Name, Gründungsjahr]*}
Merchandise_Artikel {*[ID, Preis, Typ]*}
Verkauft {*[Merchandise_Artikel_ID, Festival_Name, Festival_Jahr, Band_ID, Anzahl]*}
Spielt_Auf {*[Band_ID, Festival_Name, Festival_Jahr, Uhrzeit]*}

Formulieren Sie folgende Anfragen in SQL. Achten Sie, falls nötig, auf duplikatfreie Ausgaben:

a) Geben Sie das Gründungsjahr der Band „Metallica" aus.

b) Formulieren Sie eine Update-Anfrage in SQL, die den Verkauf eines Merchandise-Artikels auf dem Festival mit Namen „Wacken Open Air" im Jahr 2015 darstellt. Erhöhen Sie dazu den Wert des Attributs „Anzahl" der Relation „Verkauft" für den Merchandise-Artikel mit ID = 54.

Fortsetzung nächste Seite!

c) Geben Sie die Uhrzeit des Konzerts von „AC/DC“ aus, welches die Band auf dem Festival „Rock am Ring“ im Jahr 2006 gegeben hat.

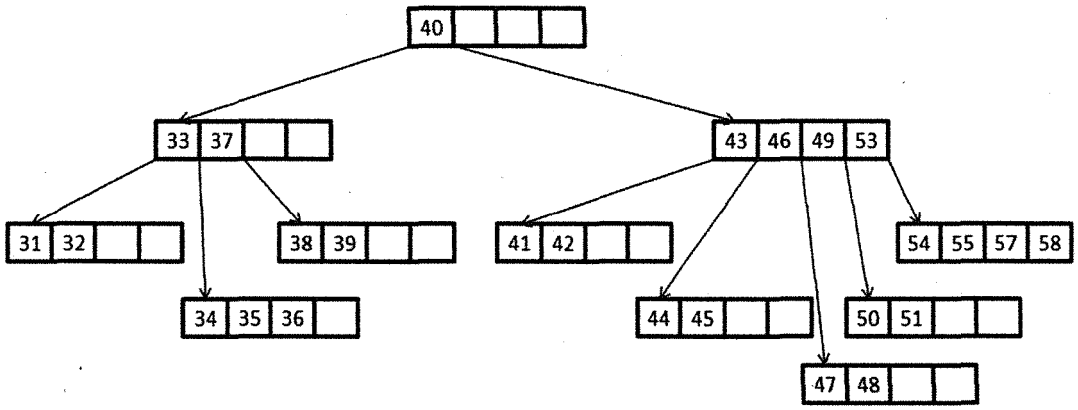
d) Formulieren Sie eine Create-Anfrage in SQL, welche eine Tabelle für die Relation „Spielt_Auf“ erzeugt.

Hinweise:
- Wählen Sie passende SQL-Datentypen.
- Achten Sie auf eventuelle Primär- und Fremdschlüsseldeklarationen.
- Achten Sie auf eventuelle NULL-Werte.

e) Geben Sie die Namen der Bands aus, die durch den Verkauf von Merchandise-Artikeln einen höheren Umsatz als 150 Euro erzielt haben.

Aufgabe 3 Indexstrukturen

Als Indexstruktur einer Datenbank sei folgender B-Baum (k = 2) gegeben:



Führen Sie nacheinander die folgenden Operationen aus. Geben Sie die auftretenden Zwischenergebnisse an. Teilbäume, die sich in einem Schritt nicht verändern, müssen nicht erneut gezeichnet werden. Sollten Wahlmöglichkeiten auftreten, so sind größere Schlüsselwerte bzw. weiter rechts liegende Knoten zu bevorzugen.

a) Einfügen des Wertes 56

b) Löschen des Wertes 37

Fortsetzung nächste Seite!

Teilaufgabe 2

Aufgabe 1 „Floyd-Hoare-Kalkül (wp-Kalkül)"

Vereinfachen Sie die folgenden Ausdrücke ***formal*** mittels ***wp-Kalkül*** und zwar so weit wie möglich, d.h. ermitteln Sie zu jedem Ausdruck der Form *wp(<Code>, <Nachbedingung>)* die schwächste Vorbedingung des *<Code>*s bzgl. der angegebenen *<Nachbedingung>*. Geben Sie Ihre Umformungen möglichst ausführlich an, d.h. einzelne Zwischenschritte müssen erkennbar und nachvollziehbar sein.

a) $wp(\text{"a -= u-- - - - - - --d;"}, a+u+d>0)$

b) $wp(\text{" if (a\&\&u>d) d=u+1; else if (d>u) a=!a; else u=d-1;"}, d>u)$

Aufgabe 2 „OOP/OOD - Reverse Engineering"

Leider ist das Klassendiagramm der folgenden Klassen verloren gegangen. Führen Sie ein Reverse Engineering durch und erstellen Sie aus dem Quellcode ein vollständiges UML-Klassendiagramm inklusive aller Klassen, Schnittstellen, Attribute, Methoden, Konstruktoren, Sichtbarkeiten, Assoziationen, Rollennamen, Multiplizitäten, Navigationspfeilen und evtl. Stereotypen. Der Quellcode innerhalb von Methoden und Konstruktoren soll nicht übertragen werden, wohl aber die Methodensignaturen. Assoziationsnamen und deren Leserichtung lassen sich aus dem Quellcode nur schwer erahnen und sollen deshalb ebenfalls weggelassen werden.

```
public abstract class Display implements PixelPainter {
	protected HardwareMatrix hardwareMatrix;
	protected int lastPaintedX;
	protected int lastPaintedY;

	public Display(HardwareMatrix hardwareMatrix) {
		this.hardwareMatrix = hardwareMatrix;
	}

	public int getWidth() {
		return hardwareMatrix.getWidth() / getWidthFactor();
	}

	public int getHeight() {
		return hardwareMatrix.getHeight() / getHeightFac-
tor();
	}

	public void clear() {
		// some longer code
	}

	protected abstract int getWidthFactor();

	protected abstract int getHeightFactor();
}
```

```
import java.awt.Color;

public interface PixelPainter {
	void set(int x, int y, Color color);

	int getHeight();

	int getWidth();
}
```

```
public interface HardwareMatrix {
	void set(int x, int y, int v);

	int getWidth();

	int getHeight();
}
```

Fortsetzung nächste Seite!

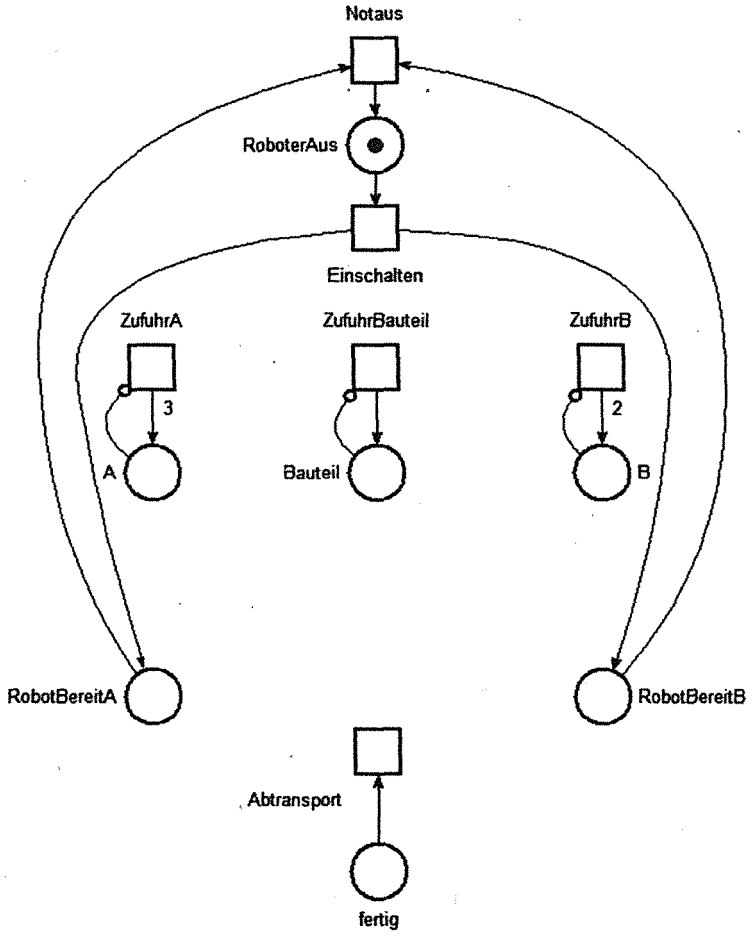
```java
import java.awt.Color;

public class DisplayUnion extends RGBDisplay {
	public static final int MAX_DISPLAY_COUNT = 50;
	private int currentDisplayCount;

	private Display[] displays;

	public DisplayUnion(Display[] displays) {
		super(null);
	}

	public int getDisplayCount() {
		return 0;
	}

	protected int getWidthFactor() {
		return 1;
	}

	protected int getHeightFactor() {
		return 1;
	}

	public void set(int x, int y, Color color) {
	}
}
```

```java
import java.awt.Color;

public class RGBDisplay extends Display {
	public RGBDisplay(HardwareMatrix matrix) {
		super(matrix);
	}

	public void set(int x, int y, Color color){

	}

	protected int getWidthFactor() {
		return 3;
	}

	protected int getHeightFactor() {
		return 1;
	}
}
```

Fortsetzung nächste Seite!

Aufgabe 3 „Petri-Netze"

Das folgende Grundgerüst stammt aus dem Petri-Netz-Modell einer Automatisierungsanlage mit zwei Robotern, das Sie auf Ihr Blatt übernehmen und geeignet um weitere Plätze (Stellen), Transitionen, Kapazitäten, Gewichte und Markierungen so ergänzen sollen, dass die darunter angegebenen Anforderungen eingehalten werden:



In der Anlage arbeiten zwei Roboter A und B, die über einen Schalter „Einschalten" aktiviert und ***jederzeit*** über einen „Notaus"-Schalter deaktiviert werden können müssen. Aufgabe der Roboter ist es, jeweils abwechselnd an Bauteilen zu arbeiten, die einzeln über ein Förderband „ZufuhrBauteil" ins System eingefahren und nach der Fertigstellung mittels „Abtransport" aus dem System abgeführt werden. Roboter A bringt genau 3 Anbauten vom Typ „A" an jedes Bauteil an und Roboter B macht das entsprechend mit genau 2 Anbauten vom Typ „B". Die Anbauten werden jeweils passend über „ZufuhrA" auf „A" bzw. mittels „ZufuhrB" auf „B" bereitgestellt. Die beiden Roboter ***dürfen niemals*** gleichzeitig an einem Bauteil arbeiten – die Reihenfolge, in der sie darauf zugreifen, ist jedoch beliebig und darf von Bauteil zu Bauteil frei variieren. Sobald einer der Roboter mit der Arbeit an einem neuen Bauteil begonnen hat, darf kein weiteres Bauteil angenommen werden, ehe der jeweils andere Roboter nicht ebenfalls mit seiner Arbeit am gleichen Bauteil fertig geworden ist (und das fertige Bauteil „auf den Platz ‚fertig' legt"). Aus Platzgründen darf höchstens ein Bauteil auf dem Ausgangsplatz „fertig" abgestellt werden, ehe es abtransportiert wird.

Herbst 2015

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer
Kennzahl: _____	**Herbst 2015**	**66116**
Kennwort: _____		
Arbeitsplatz-Nr.: _____		

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: **12**

Bitte wenden!

Thema Nr. 1

Teilaufgabe 1

1. Modellierung

Wir wollen eine relationale Datenbankstruktur für ein Online-Auktionshaus modellieren.

Das Auktionshaus hat Mitglieder. Diese Mitglieder haben Kundennummern, Namen und Adressen. Sie können Verkäufer und/oder Käufer sein. Verkäufer können eine Laden-URL (eine Subdomain) erhalten, die zu einer Seite mit ihren aktuellen Auktionen führt. Verkäufer können neue Auktionen starten. Die Auktionen eines Verkäufers werden durchnummeriert. Diese Auktionsnummer ist nur je Verkäufer eindeutig. Jede Auktion hat ein Mindestgebot und eine Ablaufzeit.

Auf Auktionen können Gebote abgegeben werden. Die Gebote auf eine Auktion werden nach ihrem Eintreffen nummeriert. Diese Nummer ist nur innerhalb einer Auktion eindeutig. Zu einem Gebot werden noch die Zeit des Gebots sowie der gebotene Geldbetrag angegeben. Die Gebote werden von Käufern abgegeben. Jedes Gebot muss einem Käufer zugeordnet sein.

Jede Auktion besteht aus einer Menge von Artikeln, aber aus mindestens einem. Artikel haben eine Beschreibung und können und über ihre Artikel-ID identifiziert werden. Zur Katalogisierung der Artikel gibt es Kategorien. Kategorien haben eine eindeutige ID und einen Namen. Jede Kategorie kann Subkategorien besitzen. Auch Subkategorien sind Kategorien (und können damit weitere Subkategorien besitzen). Jeder Artikel kann beliebig vielen Kategorien zugeordnet werden.

Entwerfen Sie für das beschriebene Szenario ein ER-Diagramm. Bestimmen Sie hierzu:
- die Entity-Typen, die Relationship-Typen und jeweils deren Attribute,
- ein passendes ER-Diagramm,
- die Primärschlüssel der Entity-Typen, welche Sie anschließend in das ER-Diagramm eintragen,
- die Funktionalitäten der Relationship-Typen, welche Sie ebenfalls in das ER-Diagramm eintragen.

2. Normalformen

Gegeben sei folgendes verallgemeinerte Relationenschema in 1. Normalform:

$$R:\{[A,B,C,D,E,F,G,H]\}$$

Für R soll die folgende Menge FD von funktionalen Abhängigkeiten gelten:

- $F \rightarrow E$
- $A \rightarrow BD$
- $AE \rightarrow D$
- $A \rightarrow EF$
- $AG \rightarrow H$

Fortsetzung nächste Seite!

Bearbeiten Sie mit diesen Informationen folgende Teilaufgaben. Vergessen Sie dabei nicht Ihr Vorgehen stichpunktartig zu dokumentieren und zu begründen.

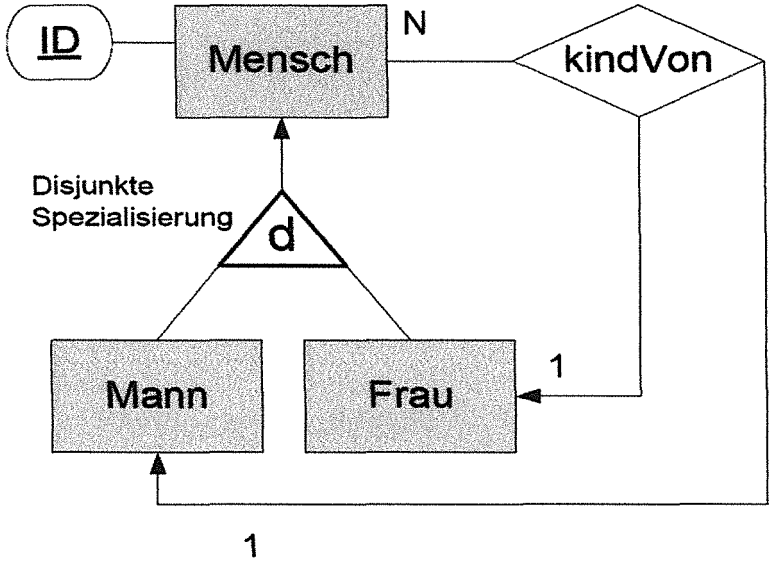
a) Bestimmen Sie alle Schlüsselkandidaten von R. Begründen Sie stichpunktartig, warum es außer den von Ihnen gefundenen Schlüsselkandidaten keine weiteren geben kann.
b) Ist R in 2NF, 3NF?
c) Berechnen Sie eine kanonische Überdeckung von FD. Es genügt, wenn Sie für jeden der vier Einzelschritte die Menge der funktionalen Abhängigkeiten als Zwischenergebnis angeben.
d) Bestimmen Sie eine Zerlegung von R in 3NF. Wenden Sie hierfür den Synthesealgorithmus an.

3. SQL

Gegeben seien folgende Relationen:

Mensch(ID, MutterID, VaterID)
Mann(ID)
Frau(ID)

Das zugehörige ER-Modell für dieses relationale Datenbankschema sieht folgendermaßen aus:



Fortsetzung nächste Seite!

Bearbeiten Sie folgende Teilaufgaben:

a) Finden Sie die Töchter der Frau mit *ID* 42.
b) Gibt es Männer, die ihre eigenen Großväter sind? Formulieren Sie eine geeignete SQL-Anfrage.
c) Definieren Sie eine View *VaterKind (VaterID; KindID)*, die allen Vätern *(VaterID)* ihre Kinder *(KinderID)* zuordnet. Diese View darf keine NULL-Werte enthalten.
d) Verwenden Sie die View aus c), um alle Väter zurückzugeben, absteigend geordnet nach der Anzahl ihrer Kinder.
e) Hugo möchte mit folgender Anfrage auf Basis der View aus c) alle kinderlosen Männer erhalten:

SELECT VaterID
FROM VaterKind
GROUP BY VaterID
HAVING COUNT(KindID) = 0

i. Was ist das Ergebnis von Hugos Anfrage und warum?
ii. Formulieren Sie eine Anfrage, die tatsächlich alle kinderlosen Männer zurückliefert.

Hinweis: Denken Sie daran, dass SQL auch Mengenoperationen kennt.

Teilaufgabe 2

1. Qualitätsmanagement

Gegeben ist die folgende Methode zur Berechnung des größten gemeinsamen Teilers zweier Zahlen m und n:

```
1   public static int ggt(int m, int n)
2   {
3       if (m <= 0 || n <= 0)
4           return -1;
5       else {
6           while (m != n) {
7               if (m < n) {
8                   int h = m;
9                   m = n;
10                  n = h;
11              }
12              int h = m - n;
13              m = n;
14              n = h;
15          }
16          return m;
17      }
18  }
```

Fortsetzung nächste Seite!

a) Geben Sie einen zur Methode ggt passenden Kontrollflussgraphen an. Benennen Sie die Knoten mit der zur jeweiligen Anweisung gehörigen Zeilennummer.
b) Erläutern Sie die Begriffe Knotenüberdeckung und Boundary-Interior-Pfadüberdeckung.
c) Geben Sie eine minimale Testmenge an, die das Kriterium der Knotenüberdeckung erfüllt. Geben Sie für jeden Testfall den durchlaufenen Pfad als Sequenz von Knotennummern an.
d) Verfahren Sie analog zu c) für das Kriterium der Kantenüberdeckung.
e) Wie viele Testfälle werden für die Methode ggt benötigt, um die Boundary-Interior-Pfadüberdeckung zu erreichen? Begründen Sie Ihre Antwort.

2. Klassendiagramm

Erstellen Sie zu der folgenden Beschreibung ein UML-Klassendiagramm. Setzen Sie insbesondere Sichtbarkeiten von Elementen, abstrakten Klassen, Vererbungsbeziehungen, Multiplizitäten, Assoziationen, Rollennamen, Assoziationsnamen, Kompositionsbeziehungen, Leserichtungen und Eigenschaften ein, um die Beschreibung möglichst vollständig umzusetzen. Wenn Sichtbarkeiten nicht explizit erwähnt sind, gehen Sie von öffentlicher Sichtbarkeit aus. Wählen Sie geeignete Datentypen für Attribute.

Eine Organisationseinheit hat einen Namen und enthält beliebig viele Organisationsobjekte. Jedes Organisationsobjekt ist in höchstens einer Organisationseinheit enthalten. Ein Organisationsobjekt ist entweder eine Organisationseinheit oder eine Person. Eine Person hat genau einen Nachnamen, beliebig viele Vornamen, einen Geburtsort und ein Geburtsdatum. Ein Datum wird durch einen Datentyp modelliert, der Tag, Monat und Jahr gruppiert. Zu einer Person kann das Alter in Jahren mittels einer Operation bestimmt werden, die das aktuelle Datum als Parameter erhält.

Jede Person ist entweder ein Student oder ein Dozent. Ein Dozent hat ein Personalkennzeichen (eine beliebige Zeichenkette). Einem Studenten ist eine (ganzzahlige) Matrikelnummer zugeordnet, die nur gelesen werden darf. Ferner soll ein Zähler für die Matrikelnummer verwaltet werden, für den als Anfangswert 1 gewählt wird.

Eine Lehrveranstaltung hat einen Namen. Lehrveranstaltungen können voneinander abhängen (uneingeschränkte Multiplizitäten). Jeder Dozent hält mindestens eine Lehrveranstaltung, die ihrerseits von mindestens einem Dozenten angeboten wird. Jeder Student hört mindestens eine Lehrveranstaltung, die ihrerseits von mindestens einem Studenten gehört wird. Jede Lehrveranstaltung ist entweder eine Vorlesung oder eine Übung. Eine Übung kann höchstens eine Vorlesung unterstützen, und zu jeder Vorlesung gibt es höchstens eine Übung.

Fortsetzung nächste Seite!

3. Zustandsdiagramm

Erstellen Sie ein UML-Zustandsdiagramm, das die möglichen Zustände und Transitionen von Benutzern eines Stock-Photo-Systems (Vorrat-Foto-System) beschreibt. Vor- und Nachbedingungen sowie Aktionen dürfen Sie in Java-ähnlicher Syntax angeben.

Orientieren Sie sich an der folgenden Verhaltensbeschreibung:

- Nach der Registrierung kann ein Benutzer entscheiden, ob er zunächst aktiv (als Fotograf) oder passiv (nicht als Fotograf) teilnimmt.
- Ein (aktiver oder passiver) Benutzer kann sich jederzeit vom System abmelden. Abgemeldete Benutzer werden automatisch aus dem System gelöscht, wenn sie ihre Entscheidung nicht innerhalb von 100 Tagen revidieren.
- Ein passiver Benutzer kann jederzeit zu einem aktiven Benutzer werden und seine Karriere als Fotograf starten.
- Ein aktiver Benutzer wird überprüft, sobald eines seiner Fotos von einem anderen Benutzer als unangemessen gemeldet wird. Die Überprüfung erfolgt durch einen Administrator. Dieser kann den betroffenen Benutzer jederzeit sperren.
- Ein Administrator kann einen gesperrten Fotografen nach einiger Zeit wieder entsperren.
- Ein Administrator kann einen beliebigen aktiven (nicht gesperrten) Benutzer jederzeit zum Administrator ernennen.
- Ein Administrator kann sein Amt niederlegen und somit zum Aktiven Benutzer werden. Dies ist jedoch nur möglich, falls davor noch mindestens drei Administratoren im System registriert sind.
- Administratoren und gesperrte Benutzer, sowie Benutzer die sich in Prüfung befinden, können sich nicht vom System abmelden.

Hinweis: Überlegen Sie sich auch, welche Parameter Sie Ihren auf den Transitionen stehenden Ereignissen mitgeben.

Thema Nr. 2

Teilaufgabe 1

1. Modellierung

Das Fitnessstudio „Bavariogym" plant zur Unterstützung des Trainings eine App für ihre Mitglieder. Betrachten Sie dazu folgendes Szenario:

- Zu einem Mitglied werden Vorname, Nachname, Eintrittsdatum und der Tarif gespeichert.
- Im Fitnessstudio arbeiten speziell ausgebildete Fitnesstrainer. Deren Aufgabe ist es, Trainingspläne zu erstellen. Ein Trainer besitzt immer eine eindeutige Personal-Nummer. Gehen Sie davon aus, dass ein Trainer kein Mitglied ist.
- Ein Trainingsplan ist nur in einem bestimmten Zeitraum gültig. Sehen Sie hierzu zwei Daten für die zeitlichen Grenzen vor. Um einen Trainingsplan zu identifizieren soll eine fortlaufende Nummer verwendet werden.
- Jeder Trainingsplan schreibt das Training an unterschiedlichen Geräten vor. Dabei sollen jeweils die Anzahl der Wiederholungen pro Satz, die Anzahl der Sätze und die Gewichtseinstellung berücksichtigt werden.
- Ein Mitglied kann nach mehreren Trainingsplänen gleichzeitig trainieren. Ebenso findet ein Trainingsplan bei mehreren verschiedenen Mitgliedern Anwendung.
- Ein Gerät besitzt einen eindeutigen Namen, sowie eine Information über die Muskelgruppe, die trainiert wird.

a) Entwerfen Sie für das beschriebene Szenario ein ER-Modell in Chen-Notation. Bestimmen Sie hierzu:
   - Die Entity-Typen, die Relationship-Typen und jeweils deren Attribute.
   - Die Funktionalitäten der Relationship-Typen.

b) Wie würde sich ihr Modell aus Teilaufgabe a) ändern, wenn die Trainer den Mitgliedern individuelle, jedoch nicht exklusive Trainingspläne anfertigen? Zeichnen Sie ein aktualisiertes ER-Modell in Chen-Notation. Bestimmen Sie hierzu:
   - Die Entity-Typen, die Relationship-Typen, jeweils **ohne** deren Attribute.
   - Die Funktionalitäten der Relationship-Typen.

c) Wie würden Sie vorgehen, um ihr Modell aus Teilaufgabe a) so zu ändern, dass Sie für jede Person im Fitnessstudio den Vornamen, den Nachnamen, die Straße, die Hausnummer, die PLZ und den Ort speichern können. Beschränken Sie ihre Antwort auf maximal drei Sätze.

Fortsetzung nächste Seite!

2. SQL-Anfragen

Gegeben ist das folgende relationale Schema:

EQUIPE {[*id*, *name*]}
FAHRER {[*id*, *name*, *equipe_id*]}
ETAPE {[*id*, *date*, *NbKm*]}
TEILNAHME {[*fahrer_id*, *etape_id*, *zeit*]}

Zur Modellierung der Tour de France im letzten Jahr (2014) wird folgendes relationales Schema verwendet. EQUIPE enthält Informationen zur Mannschaft (der EQUIPE), zu denen jeder Fahrer eindeutig zugeordnet ist. FAHRER enthält die Informationen zum Fahrer mit dem Fremdschlüssel equipe_id. ETAPE enthält die Information zu den Etapen, die gefahren werden, das Attribut NbKm ist die Länge der Etape in km. TEILNAHME stellt die Verbindung von Etapen und Fahrern her, das Attribut zeit ist die individuelle Zeit eines Fahrers fahrer_id in der Etape etape_id. Fällt der Fahrer für die Etape id aus, so erscheint er in der Relation TEILNAHME für dieses etape_id nicht.

Formulieren Sie folgende Anfragen in SQL. Achten Sie falls nötig auf duplikatfreie Ausgaben:

a) Wie lang war die Tour de France letztes Jahr?

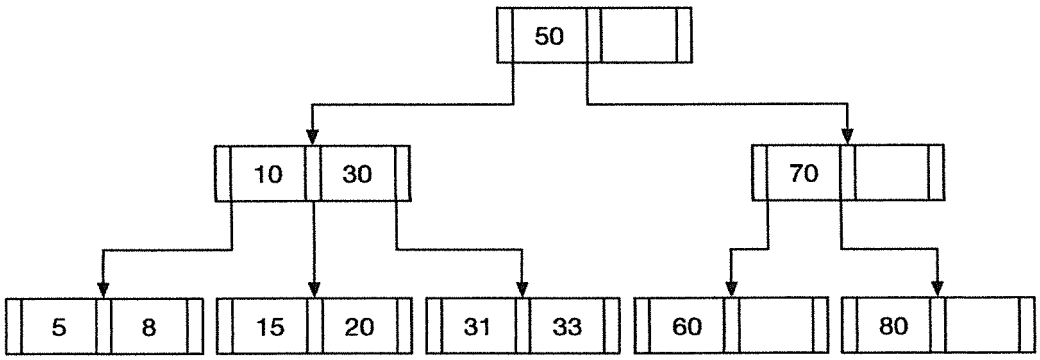
b) Geben Sie die Namen der Fahrer der Equipe 'RENNSTALL' aus.

c) Geben Sie die Namen der Fahrer an, die an allen Etapen teilgenommen haben.

d) Geben Sie die Gesamtlaufzeit aller Läufer der Equipe 'RENNSTALL' in der 1. Etape (ETAPE.id=1) aus.

3. Indexstrukturen

Gegeben ist der folgende B-Baum der Ordnung 3 (max. drei Kindknoten, max. zwei Schlüssel pro Knoten):



Fortsetzung nächste Seite!

a) Fügen Sie die Werte 9 und 45 ein. Löschen Sie anschließend die Werte 30 und 70. Zeichnen Sie den Baum nach jeder Einfüge- bzw. Lösch-Operation.

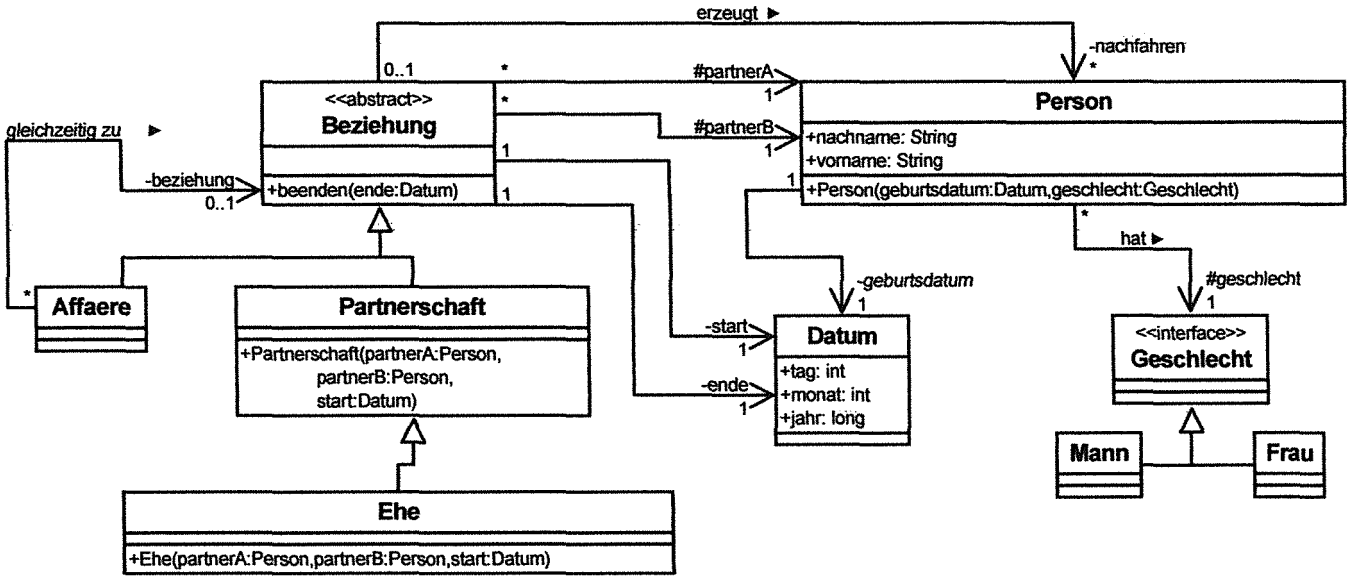
b) Geben Sie das Vorgehen für exakte Suche und Bereichssuche im B+-Baum an.

c) Geben Sie einen Vorteil und einen Nachteil von B+-Bäume im Vergleich zu Hash-Tabellen an.

Teilaufgabe 2

Aufgabe 1: „OOD/OOP"

Gegeben sei folgendes UML-Klassendiagramm:

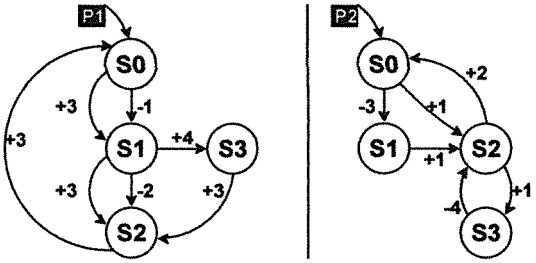


Geben Sie eine Implementierung dieses (vereinfachten) Klassendiagramms in einer gängigen und geeigneten objekt-orientierten Programmiersprache Ihrer Wahl an. Achten Sie dabei insbesondere auf die richtige Wiedergabe der Sichtbarkeiten und Multiplizitäten. Die Methoden- und Konstruktor-Rümpfe dürfen Sie leer lassen, sofern das Programm dann trotzdem noch übersetzbar ist (ggf. muss bei Konstruktoren ein gültiger **super**-Aufruf eingefügt werden).

Fortsetzung nächste Seite!

Aufgabe 2: „FSM und PN“

Betrachten Sie folgendes Modell eines einfachen Kommunikationsprotokolls:

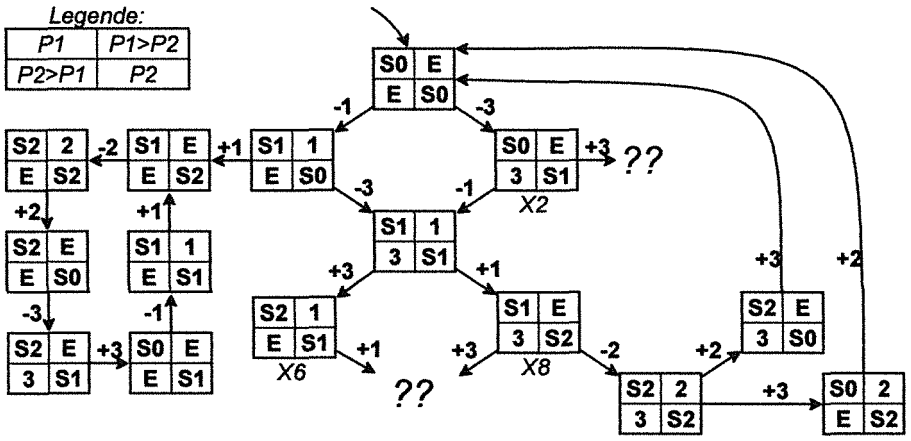


Die Prozesse P1 und P2 arbeiten asynchron und senden sich gegenseitig Nachrichten über einen Kanal mit genau einer Speicherstelle (Puffer) pro Übertragungsrichtung. Die Kantenbeschriftung ist wie folgt zu lesen:

- „**–1**“:= P1 sendet Nachricht „**1**“ an P2
- „**+2**“:= P2 empfängt Nachricht „**2**“ von P1

Von außen werden keine Nachrichten an die beiden Automaten gesendet.

a) Ergänzen Sie alle fehlenden Zustände im folgenden Erreichbarkeitsgraphen des sich ergebenden Produktautomaten. Übernehmen Sie dafür nur die benannten Zustände „X2“, „X6“ bzw. „X8“ in Ihr Arbeitsblatt und vervollständigen Sie die davon ausgehenden Teilgraphen.



b) Beantworten Sie folgende Fragestellungen und begründen Sie jeweils kurz Ihre Antwort:

i) Ist der Startzustand jederzeit direkt oder indirekt wiederherstellbar?

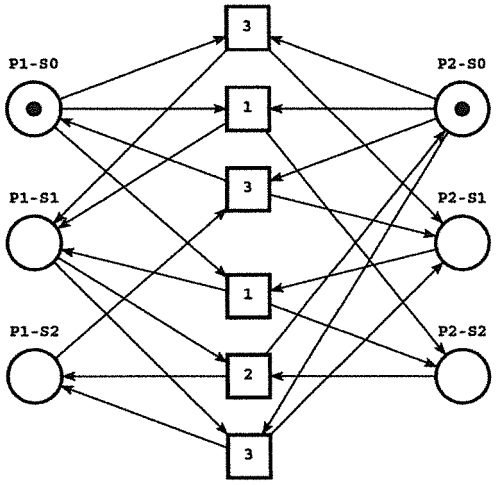
ii) Können beide Prozesse ausgehend vom Startzustand alle ihre individuellen Zustände einnehmen?

iii) Enthält das System Verklemmungssituationen (Deadlocks)?

iv) Wenn Deadlocks auftreten können, beschreiben Sie kurz die Situation, die jeweils vorliegt.

Fortsetzung nächste Seite!

c) Nach Analyse des Produktautomaten modelliert jemand das nebenstehende Petri-Netz und behauptet, es zeigt genau das gleiche Verhalten der beiden Prozesse, wenn man die Platzmarkierung als Zustandszeiger und das Schalten einer Transition als Übertragung der gleichnamigen Nachricht interpretiert. Nehmen Sie kurz aber nachvollziehbar Stellung zu dieser Behauptung.



Aufgabe 3 „Formale Verifikation"

```
long doubleFac(long n) {
      /* P */ long df = 1;
      for (long x = n; x > 1; x -= 2) {
            df *= x;
      } /* Q */
      return df;
}
```

$P := n \geq 0$

$$Q := df = n!! := \begin{cases} 2^k \cdot k! & \text{für gerade } n, \text{dabei sei } k := \frac{n}{2} \\ \frac{(2k)!}{2^k \cdot k!} & \text{für ungerade } n, \text{dabei sei } k := \frac{n+1}{2} \end{cases}$$

Zur Vereinfachung nehmen Sie im Folgenden an, dass die verwendeten Datentypen unbeschränkt sind und daher keine Überläufe auftreten können!

a) Welche der folgenden Bedingungen ist eine zum Beweisen der Korrektheit der Methode mittels *wp*-Kalkül (Floyd-Hoare-Kalkül) sinnvolle Schleifeninvariante?

- $df = n!! - x!! \wedge x \geq 1$
- $df = (n - x)!! \wedge x \geq 1$
- $df \cdot x!! = n!! \wedge x \geq 0$
- $(df + x)!! = n!! \wedge x \geq 0$

Fortsetzung nächste Seite!

b) Zeigen Sie *formal* mittels *wp*-Kalkül, dass die von Ihnen gewählte Bedingung unmittelbar vor Beginn der Schleife gilt, wenn zu Beginn der Methode die Anfangsbedingung P gilt.

c) Zeigen Sie *formal* mittels *wp*-Kalkül, dass die von Ihnen gewählte Bedingung tatsächlich eine Invariante der Schleife ist.

d) Zeigen Sie *formal* mittels *wp*-Kalkül, dass am Ende der Methode die Nachbedingung Q erfüllt wird.

e) Beweisen Sie, dass die Methode immer terminiert. Geben Sie dazu eine Terminierungsfunktion an und begründen Sie kurz Ihre Wahl.

Frühjahr 2016

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer
Kennzahl: _____	**Frühjahr 2016**	**66116**
Kennwort: _____		
Arbeitsplatz-Nr.: _____		

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: **14**

Bitte wenden!

Thema Nr. 1

Teilaufgabe 1

1. Modellierung

Für die bayerische Forstverwaltung wird eine Datenbank zur Erschließung einer Jagd-Statistik benötigt. Gehen Sie dabei von folgendem Szenario aus:

- Die Administration von Jagdgebieten obliegt den Landkreisen. Jeder Landkreis besitzt, neben seinem Namen und der Einwohnerzahl, ein eindeutiges KFZ-Kennzeichen.
- Die Jagd findet in Jagdgebieten statt. Ein Jagdgebiet soll dem Landkreis zugeteilt werden, indem es liegt. Gehen Sie davon aus, dass Jagdgebiete nicht in mehreren Landkreisen liegen können. Zusätzlich ist für jedes Jagdgebiet der Name und die Gesamtfläche zu speichern. Dabei ist zu beachten, dass die Namen nur innerhalb eines einzelnen Landkreises eindeutig sind.
- Die Erlaubnis zum Jagen wird durch einen Jagdschein erteilt. Dieser kann nur von einem Landkreis ausgestellt werden und beschränkt sich auf ein oder mehrere Jagdgebiete. Er wird durch eine Jagdschein-Nummer (JSNR) identifiziert und ist in einem bestimmtem Zeitintervall gültig. Dieses soll über zwei Zeitpunkte festgelegt werden (gültig von, gültig bis).
- Ein Jäger besitzt genau einen Jagdschein. Zu einem Jäger sollen Name, Stadt, Straße und Hausnummer, gespeichert werden. Da die Jagdtradition innerhalb einer Familie häufig von einer zur nächsten Generation weitergegeben wird, kann es vorkommen, dass Name und Adresse von zwei unterschiedlichen Jägern gleich ist (z.B. Vater und Sohn). Aus diesem Grund ist eine eindeutige Identifikationsnummer notwendig.
- Um Statistiken erheben zu können, muss berücksichtigt werden, welches Wild von welchen Jägern zu welchem Zeitpunkt in welchem Jagdgebiet erlegt worden ist. Gehen Sie davon aus, dass es mehrere Jäger geben kann, die gemeinsam ein Wild erlegen (z.B. in einer Jagdgesellschaft). Zu einem Wild gehört die Art (z.B. Reh), die Größe, das Gewicht, sowie eine eindeutige Identifikationsnummer. Zusätzlich unterscheidet man zwischen Haarwild und Federwild, wobei beim Haarwild der Typ des Gehörns (z.B. Hirschgeweih) und beim Federwild die Flügelspannweite betrachtet werden soll.

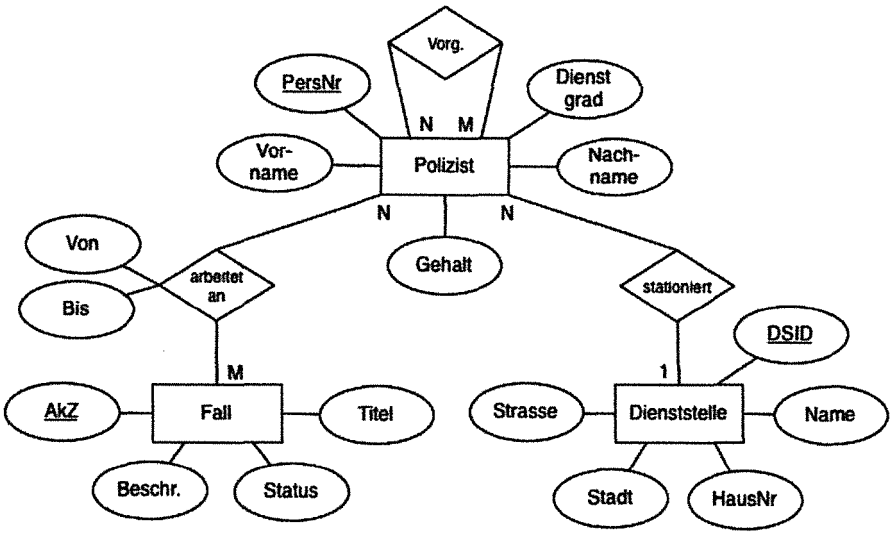
a) Entwerfen Sie für das beschriebene Szenario ein ER-Modell in Chen-Notation. Bestimmen Sie hierzu:
   - die Entity-Typen, die Relationship-Typen und jeweils deren Attribute,
   - die Primärschlüssel der Entity-Typen, welche Sie anschließend in das ER-Diagramm eintragen, und
   - die Funktionalitäten der Relationship-Typen.

b) Überführen Sie das ER-Modell aus Aufgabe a) in ein verfeinertes relationales Modell. Geben Sie hierfür die verallgemeinerten Relationenschemata an. Achten Sie dabei insbesondere darauf, dass die Relationenschemata keine redundanten Attribute enthalten.

Fortsetzung nächste Seite!

2. Relationale Algebra, SQL

Gegeben sei folgendes ER-Modell, welches Polizisten, deren Dienststelle und Fälle, an denen sie arbeiten, speichert:



Gehen Sie dabei von dem dazugehörigen relationalen Schema aus:

Polizist: {[*PersNr, DSID, Vorname, Nachname, Dienstgrad, Gehalt*]}
Dienststelle: {[*DSID, Name, Strasse, HausNr, Stadt*]}
Fall: {[{*AkZ, Titel, Beschreibung, Status*]}
Arbeitet_An: {[*PersNr, AkZ, Von, Bis*]}
Vorgesetzte: {[*PersNr, PersNr_Vorgesetzter*]}

a) Formulieren Sie eine Anfrage in relationaler Algebra, welche den Vornamen und Nachnamen von Polizisten zurückgibt, deren Dienstgrad 'Polizeikommissar' ist und mehr als 1500 Euro verdienen.

b) Formulieren Sie eine Anfrage in relationaler Algebra, welche die Titel der Fälle ausgibt, die von Polizisten mit dem Nachnamen 'Mayer' bearbeitet wurden.

c) Formulieren Sie eine SQL-Anfrage, welche die Anzahl der Polizisten ausgibt, die in der Stadt 'München' arbeiten und mit Nachnamen 'Schmidt' heißen.

d) Formulieren Sie eine SQL-Anfrage, welche die Namen der Dienststellen ausgibt, die am 14.02.2012 an dem Fall mit dem Aktenzeichen XZ1508 beteiligt waren. Ordnen Sie die Ergebnismenge alphabetisch (aufsteigend) und achten Sie darauf, dass keine Duplikate enthalten sind.

e) Definieren Sie die View 'Erstrebenswerte_Dienstgrade', welche Dienstgrade enthalten soll, die in München mit durchschnittlich mehr als 2500 Euro besoldet werden.

Fortsetzung nächste Seite!

f) Formulieren Sie eine SQL-Anfrage, welche Vorname, Nachname und Dienstgrad von Polizisten mit Vorname, Nachname und Dienstgrad ihrer Vorgesetzten als ein Ergebnis-Tupel ausgibt (siehe Beispiel-Tabelle). Dabei sind nur Polizisten zu selektieren, die an Fällen gearbeitet haben, deren Titel den Ausdruck 'Fussball' beinhalten. An Vorgesetzte sind keine Bedingungen gebunden. Achten Sie darauf, dass Sie nicht nur direkte Vorgesetzte, sondern alle Vorgesetzte innerhalb der Vorgesetzten-Hierarchie betrachten. Ordnen Sie ihre Ergebnismenge alphabetisch (absteigend) nach Nachnamen des Polizisten.

VN	NN	DG	VN_VG	NN_VG	DG_VG
Hans	Müller	Polizeimeister	Andreas	Schmidt	Polizeikommissar
Hans	Müller	Polizeimeister	Stefan	Hoffmann	Polizeidirektor
Josef	Fischer	Polizeihauptmeister	Sebastian	Wagner	Polizeioberkommissar
Josef	Fischer	Polizeihauptmeister	Stefan	Hoffmann	Polizeidirektor

Abbildung 1: Ergebnistabelle

Hinweis: Sie dürfen Views verwenden, um Teilergebnisse auszudrücken.

Fortsetzung nächste Seite!

3. Normalformen

Gegeben sei ein Auszug der Relation S einer relationalen Datenbank in Abbildung 2. Darin sind Informationen zu Schrauben eines Baumarkts enthalten. Jeder Schraubentyp besitzt eine eindeutige Artikelnummer, sowie eine Kopfform, ein Formelement, einen Gewindetyp, das Material und die jeweilige Länge. Zusätzlich ist zu jeder Schraube gespeichert, welche DIN Norm sie erfüllt. Diese Normen standardisieren Kopfform, Formelement, Material und Gewinde. Zum Beispiel erfüllt eine Senk-Holzschraube mit Schlitz die DIN Norm 97.

ArtNr	DIN	Kopfform	Formelement	Gewinde	Material	Länge
SR2141	DIN571	Sechskant	Kreuz	Holzschraube	Holz	34mm
SR8923	DIN478	Vierkant	Trox-TR	Rechts	Stahl	41mm
SR0984	DIN7996	Rund	Schlitz	Links	Messing	80mm
SR7688	DIN571	Sechskant	Kreuz	Holzschraube	Holz	52mm
SR3579	DIN965	Senk	Kreuz	Rohr	Keramik	29mm
SR4139	DIN571	Sechskant	Kreuz	Holzschraube	Holz	78mm
SR6760	DIN97	Senk	Schlitz	Holzschraube	Holz	29mm

Abbildung 2: Relation S

Bearbeiten Sie mit diesen Informationen folgende Teilaufgaben. Vergessen Sie dabei nicht, Ihr Vorgehen stichpunktartig zu dokumentieren und zu begründen.

a) Bestimmen Sie alle funktionalen Abhängigkeiten von S.

b) Bestimmen Sie alle Kandidatenschlüssel von S.

c) Zeigen Sie an einer geeigneten Instanz, dass in der Relation S eine Update-Anomalie auftreten kann.
Hinweis: Beachten Sie, dass Ihre Instanz alle Funktionalen Abhängigkeiten aus der Aufgabenstellung erfüllt.

d) Welcher Normalform genügt die Relation S (1NF, 2NF, 3NF, BCNF)? Begründen Sie Ihre Aussagen ausführlich.

Fortsetzung nächste Seite!

Teilaufgabe 2

Aufgabe 1.1: UML-Klassendiagramm

Aufgabenstellung:

Modellieren Sie das wie folgend spezifizierte soziale Netzwerk mit Hilfe eines UML-Klassendiagramms. Auf Sichtbarkeiten kann verzichtet werden.
In dieser Aufgabe soll eine Internetplattform für ein einfaches soziales Netzwerk mit Hilfe von UML modelliert werden. Das soziale Netzwerk besteht aus Teilnehmern. Für jeden Teilnehmer sind Benutzername, Passwort und E-Mail-Adresse zu speichern. Zwei Teilnehmer können miteinander befreundet sein, wobei jeder Teilnehmer beliebig viele Freunde besitzen kann. Jeder Teilnehmer kann ferner eine beliebige Anzahl von Gruppen gründen. Für jede Gruppe ist ein Titel, eine Kurzbeschreibung, der Gründer sowie ein Gründungsdatum zu erfassen. Jede Gruppe wiederum besitzt eine beliebige Anzahl von Teilnehmern als Mitglieder. Um Mitglied einer Gruppe werden zu können, muss sich ein Teilnehmer bei dieser anmelden. Damit der entsprechende Teilnehmer tatsächlich Mitglied wird, muss er nachfolgend vom Gründer der Gruppe zugelassen werden. Für jedes Mitglied einer Gruppe ist das Datum des Beitritts zu dieser Gruppe zu speichern. Zu jeder Gruppe gehört zusätzlich ein Forum, in dem sich die Mitglieder der Gruppe austauschen können. Die Forenbeiträge können nur von Gruppenmitgliedern eingestellt werden. Ein Forenbeitrag ist dabei durch einen Betreff, einen Beitragstext, den Autor, ein Einstellungsdatum sowie ggf. einen „Ursprungs"-Beitrag charakterisiert, auf den sich der entsprechende Eintrag bezieht.

Aufgabe 1.2: UML-Aktivitätsdiagramm

Aufgabenstellung:

Modellieren Sie das folgende Verhalten mit Hilfe eines Aktivitätsdiagramms.

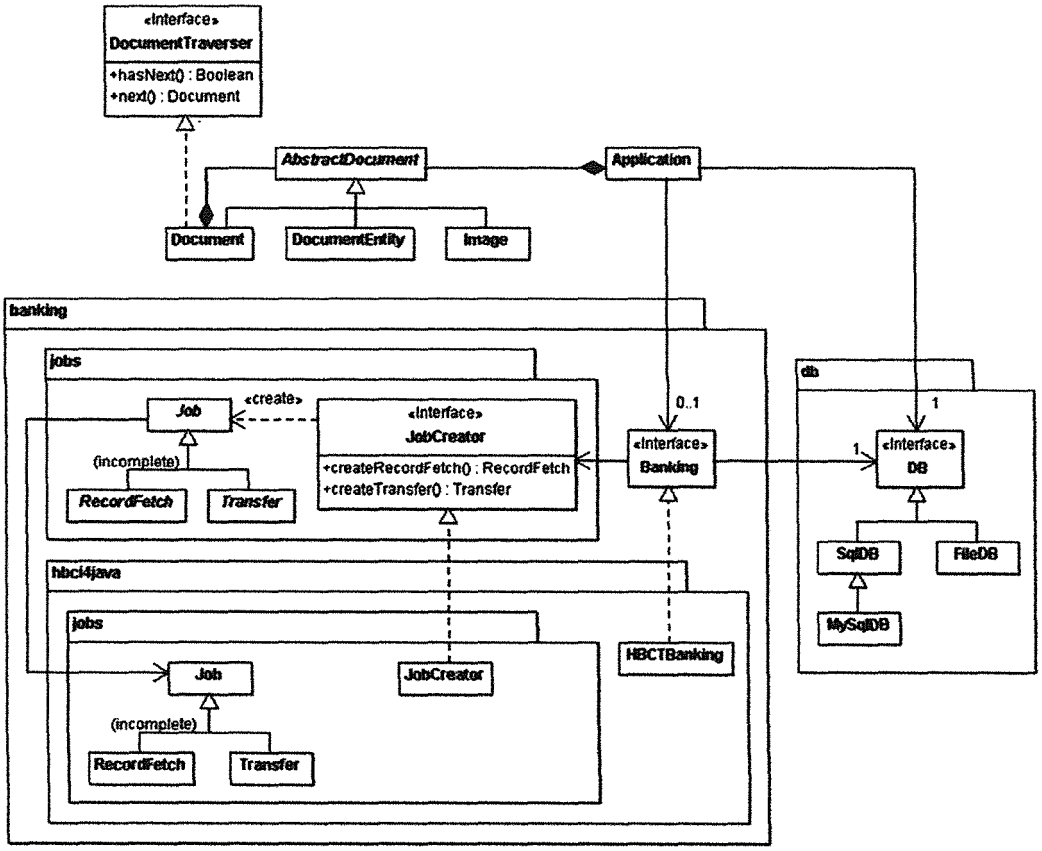
Versucht ein Besucher (nicht notwendigerweise ein Teilnehmer) der Internetplattform einen Eintrag in ein Forum zu schreiben, reagiert die Plattform wie folgt: Zunächst wird der Besucher aufgefordert sich mit Benutzername und Kennwort zu authentifizieren. Solange dies misslingt, erfolgt ein erneuter Versuch. Gelingt die Anmeldung, wird überprüft, ob der (nun authentifizierte) Teilnehmer Mitglied der Gruppe ist. Wenn dies zutrifft kann der Benutzer seinen Eintrag verfassen. Andernfalls bekommt der Teilnehmer die Möglichkeit sich für die Gruppe anzumelden.

Fortsetzung nächste Seite!

Aufgabe 2.1: Entwurfsmuster erkennen

Aufgabenstellung:

Benennen Sie im folgenden Klassendiagramm fünf verschiedene Entwurfsmuster. Geben Sie die jeweils beteiligten Klassen und deren Zuständigkeit im entsprechenden Muster an.



Fortsetzung nächste Seite!

Aufgabe 2.2: Entwurfsmuster beschreiben

Aufgabenstellung:

a) Beschreiben Sie die Funktionsweise zweier der folgenden Entwurfsmuster und geben Sie ein passendes UML-Klassendiagramm an
   - Kompositum
   - Dekorator
   - Klassenadapter
   - Objektadapter

b) Erklären Sie unabhängig zu den in a) gewählten Mustern mit maximal zwei Sätzen den Unterschied zwischen Klassenadapter und Objektadapter.

Aufgabe 2.3: Entwurfsmuster Implementierung

Aufgabenstellung:

Implementieren Sie einen Stapel in einer beliebigen Sprache. Nutzen Sie dazu ein Array mit fixer Größe. Auf eine Überlaufprüfung darf verzichtet werden. Implementieren Sie in der Klasse das Iterator Entwurfsmuster, um auf die Inhalte zuzugreifen, sowie eine Funktion zum hinzufügen von Elementen. Als Typ für den Stapel kann zur Vereinfachung ein Integertyp verwendet werden.

Thema Nr. 2

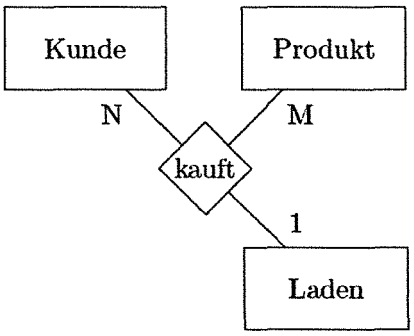
Teilaufgabe 1

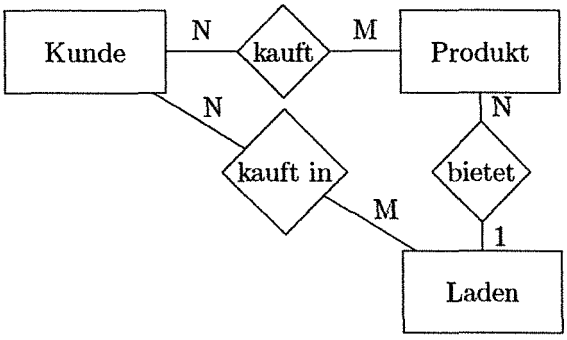
1. ER-Diagramme

1.1 Modellieren Sie den Aufbau eines Sportklubs mit einem ER-Diagramm. Kardinalitäten sind hierbei in der (min, max)-Notation anzugeben.

Personen unterteilen sich in Sportler und Betreuer. Personen haben einen Namen, eine Adresse, eine Telefonnummer sowie ein Geburtsdatum. Identifiziert werden sie über eine eindeutige Kennung. Zu den Sportlern ist ihr Gewicht sowie ihr Geburtsland bekannt, zu den Betreuern ihr Gehalt sowie eine Liste ihrer Führerscheinklassen.
Sportler können Mitglied eines oder mehrerer Teams sein, die über ihren Namen identifiziert werden. Jedes Team wird von mindestens einem Betreuer in mindestens einer Sportart betreut. Sportarten haben eine Kennung, einen Namen und eine Angabe zu den durchschnittlichen monatlichen Kosten. Sportler können mehrere Sportarten betreiben, wobei sie maximal eine Lieblingssportart haben.

1.2 Welche Aussagen lassen sich für folgende Modelle unter einer üblichen Interpretation treffen? Inwieweit unterscheiden bzw. gleichen sie sich? Es kommt hier vor allem auf die Gesamtaussage an. Formulieren Sie die Aussagen auch mit Hilfe partieller Funktionen.



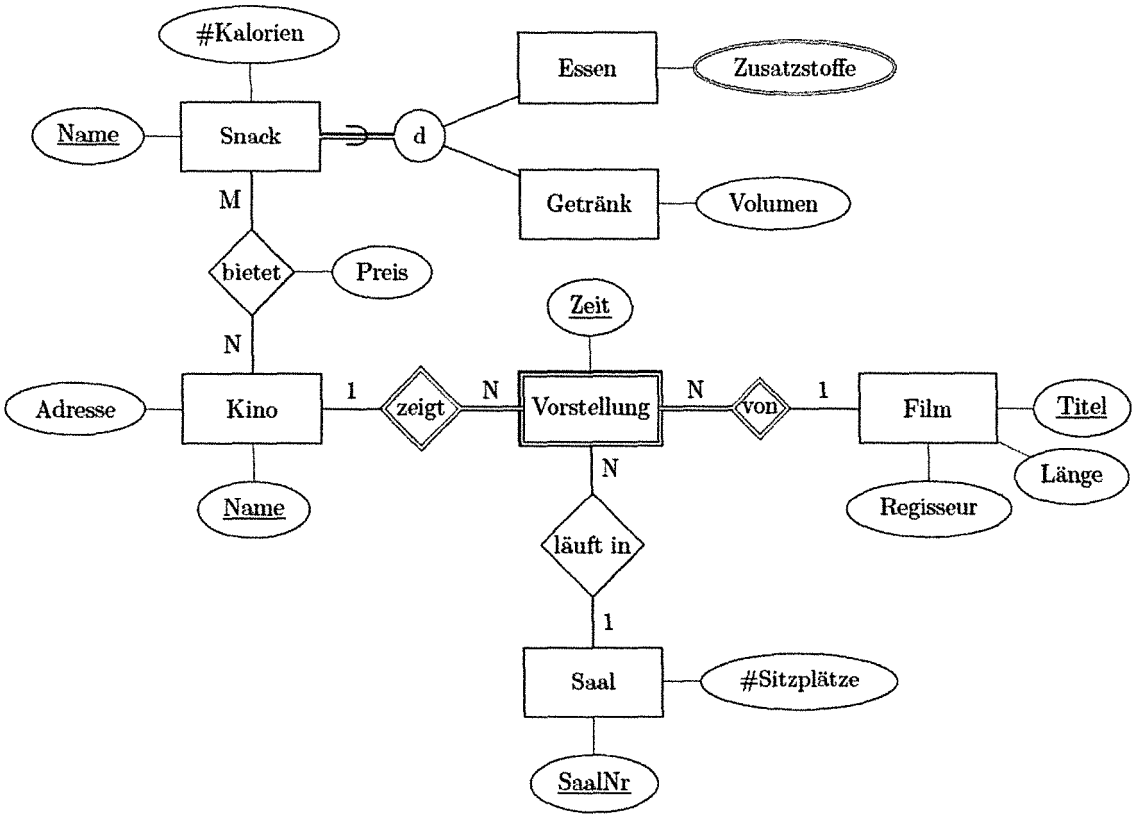


Bemerkung: Die Attribute wurden der Übersichtlichkeit wegen weggelassen.

Fortsetzung nächste Seite!

2. Relationales Modell

2.1 Überführen Sie das folgende ER-Modell in ein relationales Modell.



2.2 Macht es im ER-Modell von Aufgabe 2.1 einen Unterschied, zwischen Entitätstyp „Kino“ und Entitätstyp „Saal“ noch eine zusätzliche Beziehung einzuführen? Wenn ja, wo liegt der Unterschied? Wenn nein, warum nicht?

Fortsetzung nächste Seite!

3. Normalisierung

Gegeben ist eine Relation „Mitarbeiter" sowie alle für die Attribute der Relation geltenden funktionalen Abhängigkeiten.

3.1 Ist die Relation in erster Normalform? Ist sie in zweiter Normalform? Ist sie in dritter Normalform? Geben Sie jeweils eine Begründung für Ihre Antwort mit an.

Mitarbeiter(MID, Vorname, Nachname, Geburtsdatum, Straße, Hausnummer, Postleitzahl, Ort, PrivatVorwahl, PrivatTelefonnummer, Abteilungsnummer, Abteilungsname, Bereichsnummer, Bereichsname, Eintrittsdatum, Dienstalter, Gehaltsklasse, Gehalt)

Zusätzlich zur Schlüsseleigenschaft des Attributs MID (MID → Mitarbeiter) gelten noch weitere funktionalen Abhängigkeiten. Alle Abhängigkeiten sind in der Menge FD zusammengefasst:

FD={
MID → Vorname Nachname Geburtsdatum Straße Hausnummer Postleitzahl Ort PrivatVorwahl PrivatTelefonnummer Abteilungsnummer Abteilungsname Bereichsnummer Bereichsname Eintrittsdatum Dienstalter Gehaltsklasse Gehalt,
Postleitzahl → Ort,
Postleitzahl → PrivatVorwahl,
PrivatVorwahl → Ort,
Ort → Privatvorwahl,
Abteilungsnummer → Abteilungsname,
Abteilungsname → Abteilungsnummer,
Bereichsnummer → Bereichsname,
Bereichsname → Bereichsnummer,
Bereichsnummer → Abteilungsnummer,
Eintrittsdatum → Dienstalter,
Dienstalter → Eintrittsdatum,
Eintrittsdatum Gehaltsklasse → Gehalt
}

Bei der Bestimmung der kanonischen Überdeckung für die Menge FD muss eine Links- und eine Rechtsreduktion durchgeführt werden.

3.2 Warum führt eine Linksreduktion von FD zu keiner Veränderung?

Fortsetzung nächste Seite!

3.3 Welche Attribute auf der rechten Seite der funktionalen Abhängigkeit
`MID → Vorname Nachname Geburtsdatum Straße Hausnummer Postleitzahl Ort PrivatVorwahl PrivatTelefonnummer Abteilungsnummer Abteilungsname Bereichsnummer Bereichsname Eintrittsdatum Dienstalter Gehaltsklasse Gehalt`
sind redundant, wenn noch keine der anderen Abhängigkeiten entfernt wurden?

4. Relationale Algebra und SQL

Nachfolgend ist das Datenbankschema einer Schule gegeben. Verschiedene Anfragen in SQL sollen auf diesem Schema durchgeführt werden.

```
Lehrkraft (Personalnummer, Name, Geschlecht, Wohnort,  Geburtsjahr)
Schüler (Schülernummer, Name, Konfession, Klassennummer)
Fach (Fachname, Beschreibung)
unterrichtet (Personalnummer, Schülernummer, Fachname, Stundenzahl)
```

Bemerkung: Das Geschlecht einer Lehrkraft ist entweder 'm' oder 'w'.
In unterrichtet: Personalnummer ist Fremdschlüssel und referenziert den Primärschlüssel von Lehrkraft; Schülernummer ist Fremdschlüssel und referenziert den Primärschlüssel von Schüler; Fachname ist Fremdschlüssel und referenziert den Primärschlüssel von Fach.

4.1 Es soll eine Liste mit Schülernamen und den zugehörigen Klassen ausgegeben werden, wobei die Ausgabe absteigend nach Klassennummer und innerhalb der Klassen aufsteigend nach Name erfolgen soll.

4.2 Aus welchen Wohnorten stammen die Lehrerinnen und wie viele kommen jeweils aus einem Ort?

4.3 Wie lautet das durchschnittliche Geburtsjahr aller Lehrkräfte?

4.4 Zu welchem Fach gibt es momentan nur eine unterrichtende Lehrkraft?

4.5 Welche Lehrkräfte unterrichten mindestens zwei verschiedene Klassen im Fach Mathematik? Geben Sie Personalnummer und Name der entsprechenden Lehrkräfte aus.

Nachfolgende Anfragen soll in relationaler Algebra formuliert werden.

4.6 Geben Sie alle Schüler mit Nummer aus, die von Lehrkraft Maier unterrichtet werden.

5. Transaktionen

5.1 Welche vier grundlegenden Eigenschaften soll eine Datenbanktransaktion erfüllen? Nennen Sie diese vier Eigenschaften und beschreiben Sie sie kurz.

5.2 Geben Sie zwei Fehler an, die im unkontrollierten (nicht synchronisierten) Mehrbenutzerbetrieb auftreten können und beschreiben Sie diese jeweils anhand eines Beispiels.

Fortsetzung nächste Seite!

Teilaufgabe 2

1. Definieren Sie die folgenden UML-Begriffe und gehen Sie besonders auf die jeweiligen Unterschiede ein.

   - Assoziation, Aggregation und Komposition
   - Invariante, Vor- und Nachbedingungen

2. In der Tagespresse war zu lesen, dass die deutschen Banken an der Entwicklung eines Online Bezahlsystems „BV" arbeiten, welches eine Alternative zu „Paypal" bilden soll. Hier sollen Sie ein einfaches UML-Modell für solch ein System entwickeln.

   Benutzer können durch Email-Adresse identifizierte BV-Konten bei einer der organisierenden Banken einrichten und dieses an ein existierendes Giro- oder Kreditkartenkonto (reales Konto) binden. Von einem BV-Konto kann man an ein anderes BV-Konto Geld senden und von einem BV-Konto Geld empfangen. Möchte man Geld empfangen, so wird die zahlende Partei per Email aufgefordert, die Zahlung zu tätigen.

   Geld auf einem BV-Konto kann auf ein verbundenes reales Konto transferiert werden. Reicht das Geld auf dem BV-Konto nicht aus für die Zahlungen, dann wird automatisch vom realen Konto nachgeladen.

   Benutzer können die bisher stattgefundenen Transaktionen einsehen. Transaktionen können nach Zustimmungen der organisierenden Bank widerrufen werden.

3. Erläutern Sie die Begriffe Refactoring, Pair Programming, testgetriebene Entwicklung. Gehen Sie insbesondere darauf ein, warum diese Konzepte im Rahmen der Agilen Softwareentwicklung empfohlen werden.

4. Projektmanagement

   (a) Erklären Sie in maximal zwei Sätzen den Unterschied zwischen **Knoten**- und **Kantennetzwerken** im Kontext des Projektmanagements.

Fortsetzung nächste Seite!

Gegeben ist die folgende Tabelle zur Grobplanung eines hypothetischen Softwareprojekts:

Aktivität	**Minimale Dauer**	**Einschränkungen**
Anforderungs-analyse	2 Monate	Endet frühestens einen Monat nach dem Start der Entwurfsphase.
Entwurf	4 Monate	Startet frühestens zwei Monate nach dem Start der Anforderungsanalyse.
Implemen-tierung	5 Monate	Endet frühestens drei Monate nach dem Ende der Entwurfsphase. Darf erst starten, nachdem die Anforderungsanalyse abgeschlossen ist.

(b) Geben Sie ein **CPM-Netzwerk** an, das die Aktivitäten und Abhängigkeit des obigen Projektplans beschreibt. Gehen Sie von der Zeiteinheit „Monate" aus.
- Das Projekt hat einen Start- und einen Endknoten.
- Jede Aktivität wird auf einen Start- und einen Endknoten abgebildet.
- Die Dauer der Aktivitäten sowie Abhängigkeiten sollen durch Kanten dargestellt werden.
- Der Start jeder Aktivität hängt vom Projektstart ab, das Projektende hängt vom Ende aller Aktivitäten ab. Modellieren Sie diese Abhängigkeiten durch Pseudoaktivitäten mit Dauer null.

(c) Berechnen Sie für jedes Ereignis (d.h. für jeden Knoten) die früheste Zeit sowie die späteste Zeit. Beachten Sie, dass die Berechnungsreihenfolge einer topologischen Sortierung des Netzwerks entsprechen sollte.

(d) Geben Sie einen kritischen Pfad durch das CPM-Netzwerk an. Möglicherweise sind hierfür weitere Vorberechnungen vonnöten.
Welche Aktivität sollte sich demnach auf keinen Fall verzögern?

(e) Geben Sie ein Gantt-Diagramm an, das den Projektplan visualisiert. Gehen Sie davon aus, dass jede Aktivität zur frühesten Zeit ihres Startknotens beginnt und zur spätesten Zeit ihres Endknotens endet (s. jeweils Teilaufgabe (c)). Geben Sie die minimale Dauer jeder Aktivität, sowie die Pufferzeit (in Klammern) an. Beispiel: *4 (+2)*. Notieren Sie alle Einschränkungen mit Hilfe geeigneter Abhängigkeitsbeziehungen. Geben Sie eine absolute Zeitskala in Monaten an.

(f) Nennen Sie zwei weitere Aktivitäten, die in der obigen Tabelle fehlen, jedoch typischerweise in Softwareentwicklungs-Prozessmodellen wie etwa dem Wasserfallmodell vorkommen.

Herbst 2016

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer
Kennzahl: _____	**Herbst 2016**	**66116**
Kennwort: _____		
Arbeitsplatz-Nr.: _____		

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: **19**

Bitte wenden!

Thema Nr. 1

Teilaufgabe 1

1. Konzeptioneller Entwurf

Im Folgenden ist die Beschreibung einer Fotoverwaltung gegeben. Erstellen Sie zu dieser ein ER-Diagramm. Kennzeichnen Sie die Primärschlüssel durch passendes Unterstreichen und geben Sie die Kardinalitäten in Min-Max-Notation an. Modellieren Sie keine Attribute oder Entitytypen, die nicht aus dem Text hervorgehen.
Es werden Fotos verwaltet, die über eine eindeutige ID identifiziert werden und einen Aufnahmezeitpunkt sowie eine Beschreibung besitzen. Den Fotos können Stichworte zugeordnet werden, die einen eindeutigen Namen haben. Stichworte sind hierarchisch organisiert, d.h. ein Stichwort kann maximal einem anderen Stichwort untergeordnet sein. Jedes Foto wurde von genau einem Fotografen aufgenommen und es können Models auf ihm abgebildet sein. Zu Fotografen und Models werden Name, Vorname, Postleitzahl und Wohnort gespeichert. Zu Fotografen wird zusätzlich ihre Mitgliedsnummer im Verband deutscher Fotografen abgelegt, durch die sie eindeutig identifiziert werden können. Models gehören genau einer Agentur an und haben eine Mitgliedsnummer, die innerhalb ihrer Agentur eindeutig ist. Agenturen haben einen eindeutigen Namen sowie eine E-Mail-Adresse. Ein Fotograf kann auch gleichzeitig Model sein. Models können der Veröffentlichung von bestimmten Fotos in bestimmten Medien zustimmen. Zu jedem Medium werden ein eindeutiger Name sowie eine Beschreibung abgelegt. Zu einer Zustimmung wird das Datum gespeichert, an dem sie erfolgt ist.

Fortsetzung nächste Seite!

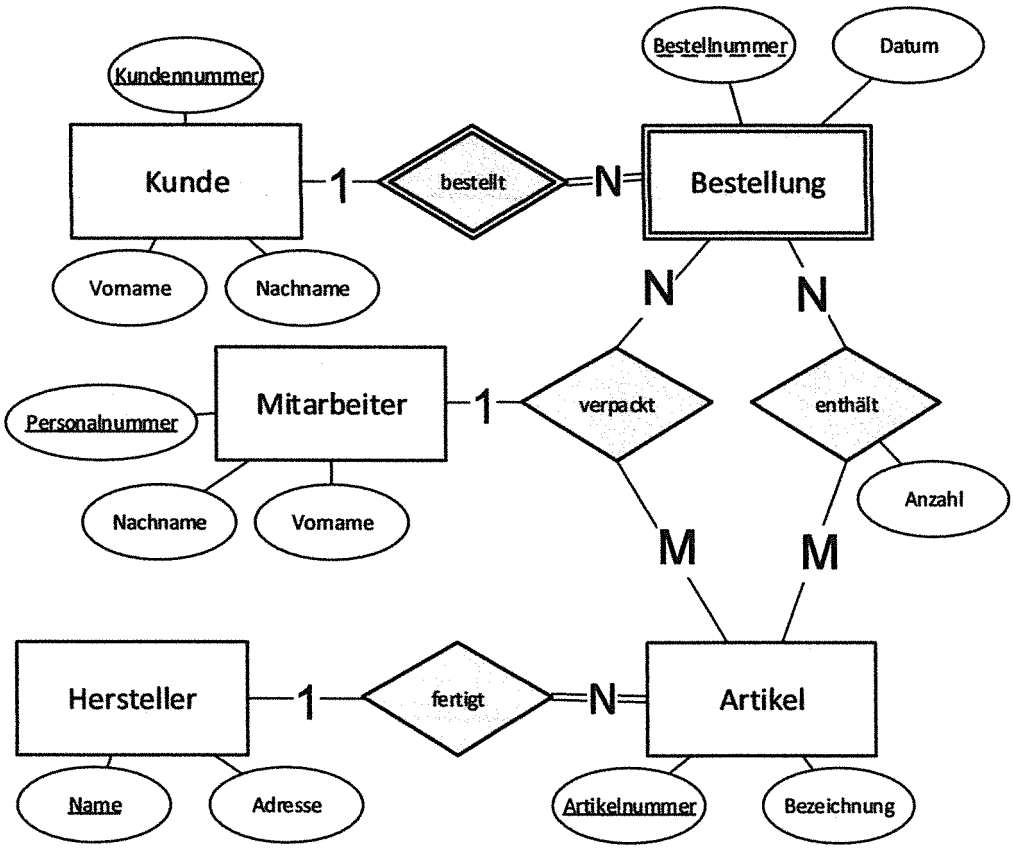
2. Relationenschema

Entwerfen Sie zum untenstehenden ER-Diagramm ein Relationenschema in dritter Normalform (3 NF) mit möglichst wenigen Relationen.

Verwenden Sie dabei folgende Notation:

Primärschlüssel werden durch Unterstreichen gekennzeichnet, 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!

3. Normalformen

a) Gegeben ist eine vollständige Extension der Relation Bestellungen mit dem zusammengesetzten Primärschlüssel {Kundennummer, Bestellnummer}:

Kundennummer	**Bestellnummer**	Kundennachname	Bestelldatum	Lieferdatum
1	1657	Frank	13.02.2015	15.02.2015
2	1306	Weizenbaum	14.01.2015	18.01.2015
3	2369	Weizenbaum	09.08.2014	10.01.2015
2	9847	Weizenbaum	21.11.2013	25.11.2013
5	2569	Schmitt	05.09.2015	05.09.2015
7	3457	Nürnberger	07.10.2012	12.12.2012
8	4468	Haberkorn	31.08.2015	01.10.2015

Geben Sie an, in welcher Normalform sich die Relation Bestellungen befindet. Begründen Sie weiterhin in zwei bis drei Sätzen, dass alle Bedingungen dieser Normalform erfüllt sind und dass nicht alle Bedingungen der nächsthöheren Normalform erfüllt sind.

b) Nennen Sie die Bezeichnungen der drei Anomalien, zu denen nicht normalisierte Relationenschemata führen können, und erläutern Sie die Anomalien in jeweils ein bis zwei Sätzen. Sie können dazu Beispiele aus der in der vorhergehenden Teilaufgabe gegebenen Relation nutzen.

Fortsetzung nächste Seite!

4. SQL

Gegeben sind folgende Relationen aus einer Personalverwaltung:

```
Mitarbeiter (MitarbeiterID, Vorname, Nachname, Vorgesetzter[Mitarbeiter],
  AbteilungsID[Abteilung], Telefonnummer, Gehalt)
Abteilung (AbteilungsID, Bezeichnung)
```

a) Schreiben Sie eine SQL-Anfrage, die Vor- und Nachnamen der Mitarbeiter aller Abteilungen mit Bezeichnung „Buchhaltung" ausgibt, absteigend sortiert nach MitarbeiterID.

b) Schreiben Sie eine SQL-Anfrage, die die Nachnamen aller Mitarbeiter zusammen mit dem Nachnamen ihres jeweiligen direkten Vorgesetzten ausgibt. Mitarbeiter ohne Vorgesetzten sollen in der Ausgabe ebenfalls enthalten sein. In diesem Fall soll der Nachname des Vorgesetzten NULL sein.

c) Schreiben Sie eine SQL-Anfrage, die die 10 Abteilungen ausgibt, deren Mitarbeiter das höchste Durchschnittsgehalt haben. Ausgegeben werden sollen der Rang (1 = höchstes Durchschnittsgehalt bis 10 = niedrigstes Durchschnittsgehalt), die Bezeichnung sowie das Durchschnittsgehalt der Abteilung.
Gehen Sie davon aus, dass es keine zwei Abteilungen mit gleichem Durchschnittsgehalt gibt. Sie können der Übersichtlichkeit halber Views oder With-Anweisungen verwenden. Verwenden Sie jedoch **keine** datenbanksystemspezifischen Erweiterungen wie limit oder rownum.

d) Schreiben Sie eine SQL-Anfrage, die das Gehalt aller Mitarbeiter aus der Abteilung mit AbteilungsID 42 um 5% erhöht.

e) Alle Abteilungen mit der Bezeichnung „Qualitätskontrolle" sollen zusammen mit den Datensätzen ihrer Mitarbeiter gelöscht werden. ON DELETE CASCADE ist für keine der Tabellen gesetzt. Schreiben Sie die zum Löschen notwendigen SQL-Anfragen.

f) Alle Mitarbeiter sollen mit SQL-Anfragen nach den Telefonnummern anderer Mitarbeiter suchen können. Sie dürfen jedoch das Gehalt der Mitarbeiter nicht sehen können. Erläutern Sie in zwei bis drei Sätzen eine Möglichkeit, wie dies in einem Datenbanksystem realisiert werden kann, ohne die gegebenen Relationen, die als Tabellen abgelegt sind, zu verändern. Sie brauchen hierzu keinen SQL-Code schreiben.

Fortsetzung nächste Seite!

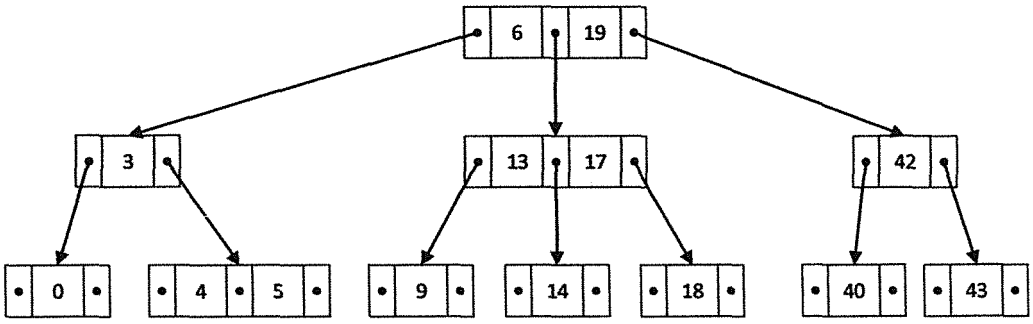
5. Physische Datenorganisation

a) Erläutern Sie die wesentliche Eigenschaft eines Tupel-Identifikators (TID) in ein bis zwei Sätzen.

b) Fügen Sie in einen anfangs leeren B-Baum mit k = 1 (maximal 2 Schlüsselwerte pro Knoten) die im Folgenden gegebenen Schlüsselwerte der Reihe nach ein. Zeichnen Sie den Endzustand des Baums nach jedem Einfügevorgang. Falls Sie Zwischenschritte zeichnen, kennzeichnen Sie die sieben Endzustände deutlich.

3, 7, 13, 11, 9, 10, 8

c) Gegeben ist der folgende B-Baum:



Die folgenden Teilaufgaben sind voneinander unabhängig.

i. Löschen Sie aus dem gegebenen B-Baum den Schlüssel 3 und zeichnen Sie den Endzustand des Baums nach dem Löschvorgang. Falls Sie Zwischenschritte zeichnen, kennzeichnen Sie den Endzustand deutlich.

ii. Löschen Sie aus dem (originalen) gegebenen B-Baum den Schlüssel 17 und zeichnen Sie den Endzustand des Baums nach dem Löschvorgang. Falls Sie Zwischenschritte zeichnen, kennzeichnen Sie den Endzustand deutlich.

iii. Löschen Sie aus dem (originalen) gegebenen B-Baum den Schlüssel 43 und zeichnen Sie den Endzustand des Baums nach dem Löschvorgang. Falls Sie Zwischenschritte zeichnen, kennzeichnen Sie den Endzustand deutlich.

6. Transaktionen

a) Nennen Sie in einem Satz den Unterschied zwischen dem strengen Zwei-Phasen-Sperrprotokoll und dem Zwei-Phasen-Sperrprotokoll.

b) Beim unkontrollierten Mehrbenutzerbetrieb kann es zum Problem der verlorengegangenen Änderungen (lost update) kommen. Erläutern Sie anhand eines Beispiels in drei bis vier Sätzen, wie dieses Problem auftreten kann und welche Folgen sich daraus ergeben.

Fortsetzung nächste Seite!

Teilaufgabe 2

Aufgabe 1: Begriffe und Konzepte

a) Welche Kriterien werden bei der Zielbestimmung im Pflichtenheft unterschieden?

b) Nennen Sie drei Einsatzziele von Softwaremaßen.

c) Welche Arten von Softwaremaßen werden unterschieden?

d) Nennen Sie drei Merkmale evolutionärer Softwareentwicklung.

e) Nennen Sie drei Vorteile des Einsatzes von Versionsverwaltungssoftware.

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

g) Was verbirgt sich hinter dem Begriff „*Continuous Integration*"?

Fortsetzung nächste Seite!

Aufgabe 2: UML

a) Gegeben sei folgende natürlichsprachliche Spezifikation eines PKI-Systems:

Damit der Schüler seinem Lehrer die Hausaufgaben verschlüsselt per E-Mail übermitteln kann, bedarf es einer entsprechenden Infrastruktur. Nachdem beide Teilnehmer die notwendige Software installiert haben, erstellt der Lehrer zunächst ein sogenanntes Schlüsselpaar, bestehend aus einem „öffentlichen" (ÖS) und einem zugehörigen „privaten" Schlüssel (PS). Anschließend veröffentlicht der Lehrer seinen ÖS durch Hochladen auf einen sogenannten Keyserver (Schlüsselverzeichnisdienst). Damit steht er jedem Schüler zur Verfügung, so dass dieser den ÖS jederzeit vom Keyserver herunterladen kann. Alternativ kann der Lehrer seinen ÖS auch direkt (z.B. per E-Mail oder USB-Stick) an den Schüler übermitteln.
Der Schüler kann nun seine Nachricht (z.B. seine Lösung) mit dem ÖS des Lehrers verschlüsseln und mit einer E-Mail versenden. Empfängt der Lehrer eine solche E-Mail, so kann er (und nur er) mit seinem PS die Nachricht wieder entschlüsseln. Umgekehrt kann der Lehrer mit seinem PS beliebige Informationen (z.B. die Note) „digital signieren". Diese unterschriebenen Daten übermittelt der Lehrer dann zusammen mit der digitalen Signatur per E-Mail an den Schüler. Der Schüler kann mit dem ÖS des Lehrers prüfen, ob die übermittelte Nachricht unverändert und tatsächlich vom unterschreibenden Lehrer stammt.

Modellieren Sie die in der Spezifikation beschriebenen Anwendungsfälle zusammen mit den jeweils beteiligten Aktoren in einem Use-Case-Diagramm. Betrachten Sie den Keyserver zunächst ebenfalls als Aktor, welcher gegenüber PKI als „externer Vermittler" auftritt.

b) Erstellen Sie ein geeignetes Klassendiagramm für das obige PKI. Berücksichtigen Sie zusätzlich zur verbalen Spezifikation noch folgende Präzisierungen:

i. Die Schlüssel werden mit einer E-Mail-Adresse „benannt", damit der Schüler den richtigen Schlüssel abrufen kann.

ii. Es gibt genau einen Keyserver; dieser verwaltet aber beliebig viele Schlüssel. Er bietet die entsprechenden Dienste zum Veröffentlichen bzw. Abfragen von Schlüsseln an.

iii. Jeder Lehrer hat höchstens ein Schlüsselpaar, aber jeder Schlüssel gehört genau einem Lehrer. Der Schüler hingegen kommuniziert mit mehreren Lehrern und kennt daher mehrere E-Mail-Adressen.

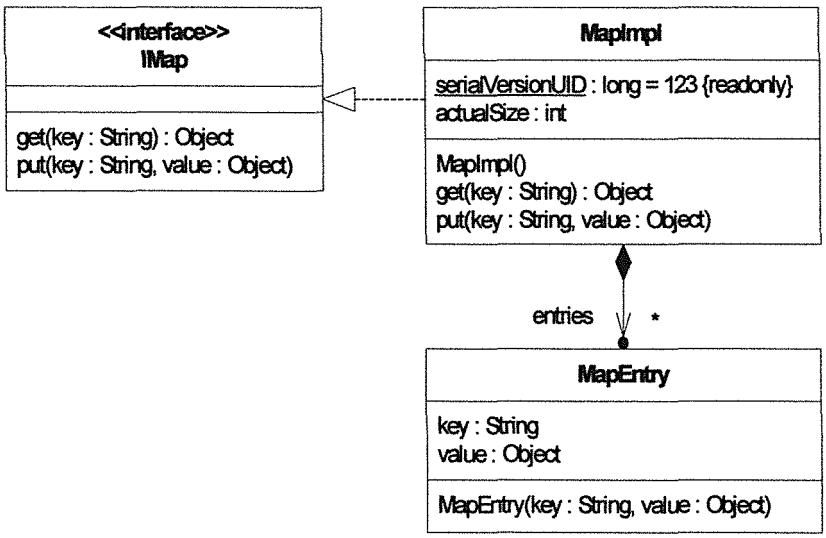
iv. Eine Nachricht kann (muss aber nicht) eine Signatur oder einen ÖS als „Anhang" zusätzlich zum eigentlichen Inhalt (zur Vereinfachung: String) mit sich führen.

v. Für das Signieren bzw. Entschlüsseln ist der PS (das zugehörige Objekt selbst) zuständig. Dafür bekommt er den Inhalt der Nachricht und gibt entsprechend eine Signatur bzw. den entschlüsselten Inhalt zurück.

vi. Für das Prüfen der Signatur und das Verschlüsseln ist der ÖS zuständig. Dazu bekommen die Methoden je nach Bedarf den Inhalt der Nachricht und die Signatur und liefern einen Wahrheitswert bzw. den verschlüsselten Inhalt zurück.

Fortsetzung nächste Seite!

c) Übertragen Sie folgendes UML-Klassendiagramm in Programm-Code einer gängigen und geeigneten objektorientierten Sprache Ihrer Wahl. Die Methodenrümpfe dürfen Sie dabei leer lassen (auch wenn das Programm dann nicht übersetzbar bzw. ausführbar wäre).



Fortsetzung nächste Seite!

Aufgabe 3: Testen

Gegeben sei folgende Java-Methode sort zum Sortieren eines Feldes ganzer Zahlen:

```
public static int[] sort(int[] array) {
  boolean swapped;
  int swapTmp;
  int[] newArray = (int[])array.clone();
  do {
    swapped = false;
    for (int index = 0; index < newArray.length - 1; index++) {
      if (newArray[index] > newArray[index + 1]) {
        swapTmp = newArray[index];
        newArray[index] = newArray[index + 1];
        newArray[index + 1] = swapTmp;
        swapped = true;
      }
    }
  } while(swapped);
  return newArray;
}
```

a) Konstruieren Sie den Kontrollflussgraphen des obigen Code-Fragments und annotieren Sie an den Knoten und Kanten die zugehörigen Datenflussinformationen (Definitionen bzw. berechnende oder prädikative Verwendungen von Variablen).

b) Nennen Sie die maximale Anzahl linear unabhängiger Programmpfade, also die zyklomatische Komplexität nach McCabe.

c) Geben Sie einen möglichst kleinen Testdatensatz an, der eine 100%-ige Verzweigungsüberdeckung dieses Moduls erzielt.

d) Beschreiben Sie kurz, welche Eigenschaften eine Testfallmenge allgemein haben muss, damit sie das datenflussorientierte Überdeckungskriterium „all-uses“ erfüllt.

e) Skizzieren Sie kurz die drei klassischen Integrationstest-Strategien „bottom-up“, „top-down“ und „sandwich“ und nennen Sie je einen typischen Vor- bzw. Nachteil.

Fortsetzung nächste Seite!

Aufgabe 4: Formale Verifikation

Gegeben seien die beiden Java-Methoden zur Berechnung der Chebyshev-Reihe:

$$C_n := (n+2) \cdot 2^{n-1}$$

```
long cpRec(long n) {
      if (n == 0) {
            return 1;
      } else if (n == 1) {
            return 3;
      } else {
            return 4 * cpRec(n-1)
                   - 4 * cpRec(n-2);
      }
}
```

```
long cpIter(long n) {
      long a = 1, k = n-1, p = 1;
      if (n > 0) {
            while (k > 0) {
                  p *= 2;
                  k--;
            }
            a = (n+2)*p;
      }
      return a;
}
```

a) Beweisen Sie formal mittels vollständiger Induktion:

$$\forall n \geq 0\text{: } cpRec(n) = C_n$$

Geben Sie dabei jeweils beide Induktionsanfänge (Basisfälle) und Induktionsvoraussetzungen (Induktionsannahmen) explizit an.

b) Welches der folgenden Prädikate stellt eine gültige Schleifeninvariante für die Schleife in **`cpIter(n)`** dar?

a. $a = (n+2) \cdot 2^{n-1}$

b. $p = 2^{n-k}$

c. $p = 2^{n-k-1} \wedge k \geq 0$

d. $C_k = (k+2) \cdot p$

c) Weisen Sie *formal* (z.B. mit der Methode der schwächsten Vorbedingung) nach, dass die von Ihnen gewählte Invariante unmittelbar vor dem ersten Betreten des Schleifenrumpfs in **`cpIter(n)`** gilt.

d) Weisen Sie nun *formal* nach, dass die von Ihnen gewählte Invariante nach jedem Durchlaufen des Schleifenrumpfs in **`cpIter(n)`** gilt.

e) Beweisen Sie schließlich *formal*, dass folgende Nachbedingung erfüllt ist:

$\forall n \geq 0$: **`cpIter(n)`** $= C_n$

f) Geben Sie eine geeignete Schleifenvariante (Terminierungsfunktion) für die Schleife in der Methode **`cpIter(n)`** an und skizzieren Sie kurz, warum damit die Terminierung bewiesen werden kann.

Thema Nr. 2

Teilaufgabe 1

1. ER-Modellierung

Gegeben seien folgende Informationen:

- Krankenhäuser bestehen aus Stationen. Zu jedem Krankenhaus ist dessen Adresse (wodurch es identifiziert werden kann) und die Anzahl an Betten bekannt. Stationen besitzen einen Namen, der nur innerhalb eines Krankenhauses eindeutig ist.
- Stationen bestehen wiederum aus Zimmern, welche nummeriert sind. Eine solche Nummer ist innerhalb einer Station eindeutig.
- Jedes Krankenhaus beschäftigt Personal. Dabei wird festgehalten, welches Gehalt bezahlt wird. Personal kann in verschiedenen Krankenhäusern beschäftigt sein.
- Personal ist durch eine Personal-Nummer gekennzeichnet und kann unter anderem in Ärzte und Krankenpfleger unterteilt werden. Zu einem Arzt ist sein Fachbereich bekannt und von welchen Krankenpflegern er Vorgesetzter ist. Kein Krankenpfleger kann zugleich Arzt sein, aber durchaus mehrere vorgesetzte Ärzte haben.
- Eine Station kann von mehreren Ärzten gleitet werden. Ein Arzt kann ebenso mehrere Stationen leiten. Außerdem ist bekannt, ob und in welchem Zimmer ein Arzt sein Büro hat. Kein Arzt muss sich sein Büro mit einem anderen Arzt teilen.
- Personal arbeitet auf Stationen in Schichten. Eine Schicht kann über Datum und Zeitraum eindeutig identifiziert werden. Eine Person kann in einer Schicht auf nur einer Station arbeiten.

a) Erstellen Sie für das oben gegebene Szenario ein geeignetes ER-Diagramm.
Verwenden Sie dabei – wenn angebracht – das Prinzip der Spezialisierung. Kennzeichnen Sie die Primärschlüssel der Entity-Typen, totale Teilnahmen und schwache Entity-Typen.
Zeichnen Sie die Funktionalitäten der Relationship-Typen in das Diagramm ein.

b) Überführen Sie Ihr in Aufgabe a) erstelltes Modell in ein verfeinertes relationales Schema.
Kennzeichnen Sie die Schlüssel durch Unterstreichen. Datentypen müssen nicht angegeben werden.

Fortsetzung nächste Seite!

2. SQL und relationale Algebra

Gegeben sei der folgende Ausschnitt aus dem Schema einer Schulverwaltung:

Person : {[
  ID : INTEGER,
  Name : VARCHAR(255),
  Wohnort : VARCHAR(255),
  Typ : CHAR(1)
]}

Unterricht : {[
  Klassenbezeichnung : VARCHAR(20),
  Schuljahr : INTEGER,
  Lehrer : INTEGER,
  Fach : VARCHAR(100)
]}

Klasse : {[
  Klassenbezeichnung : VARCHAR(20),
  Schuljahr : INTEGER,
  Klassenlehrer : INTEGER
]}

Klassenverband : {[
  Schüler : INTEGER,
  Klassenbezeichnung : VARCHAR(20),
  Schuljahr : INTEGER
]}

Hierbei enthält die Tabelle *Person* Informationen über Lehrer (Typ 'L') und Schüler (Typ 'S'); andere Werte für Typ sind nicht zulässig. *Klasse* beschreibt die Klassen, die in jedem Schuljahr gebildet wurden, zusammen mit ihrem Klassenlehrer. In *Unterricht* wird abgelegt, welcher Lehrer welches Fach in welcher Klasse unterrichtet; es ist möglich, dass derselbe Lehrer mehr als ein Fach in einer Klasse unterrichtet. *Klassenverband* beschreibt die Zuordnung der Schüler zu den Klassen.

a) Schreiben Sie eine SQL-Anweisung, die die Tabelle *Unterricht* mit allen ihren Constraints (einschließlich Fremdschlüsselconstraints) anlegt.

b) Definieren Sie ein geeignetes Constraint, das sicherstellt, dass nur zulässige Werte im Attribut Typ der (bereits angelegten) Tabelle *Person* eingefügt werden können.

c) Schreiben Sie eine SQL-Anweisung, die die Bezeichnung der Klassen bestimmt, die im Schuljahr 2015 die meisten Schüler haben.

d) Schreiben Sie eine SQL-Anweisung, die die Namen aller Lehrer bestimmt, die nur Schüler aus ihrem Wohnort unterrichtet haben.

e) Schreiben Sie eine SQL-Anweisung, die die Namen aller Schüler bestimmt, die immer den gleichen Klassenlehrer hatten.

f) Schreiben Sie eine SQL-Anweisung, die alle Paare von Schülern bestimmt, die mindestens einmal in der gleichen Klasse waren. Es genügt dabei, wenn Sie die ID der Schüler bestimmen.

g) Formulieren Sie eine Anfrage in der relationalen Algebra, die die ID aller Schüler bestimmt, die mindestens einmal von 'Ludwig Lehrer' unterrichtet wurden.

h) Formulieren Sie eine Anfrage in der relationalen Algebra, die Namen und ID der Schüler bestimmt, die von allen Lehrern unterrichtet wurden.

Beachten Sie bei der Formulierung der SQL-Anfragen, dass die Ergebnisrelationen keine Duplikate enthalten dürfen. Sie dürfen geeignete Views definieren.

Fortsetzung nächste Seite!

3. Entwurfstheorie

Gegeben sei folgendes relationale Schema R in erster Normalform:

$$R : \{[A,B,C,D,E,F]\}$$

Für R gelte folgende Menge FD funktionaler Abhängigkeiten:
$FD = \{$

$$\begin{aligned} AC &\rightarrow B, \\ DEF &\rightarrow BC, \\ F &\rightarrow AB, \\ D &\rightarrow F, \\ BC &\rightarrow E \end{aligned}$$

$\}$

a) Bestimmen Sie alle Kandidatenschlüssel von R mit FD.
Hinweis: Die Angabe von Attributmengen, die keine Kandidatenschlüssel sind, führt zu Abzügen.

b) Prüfen Sie, ob R mit FD in 2NF bzw. 3NF ist.

c) Bestimmen Sie mit folgenden Schritten eine kanonische Überdeckung FD_C von FD:
   i. Führen Sie eine Linksreduktion von FD durch. Geben Sie die Menge funktionaler Abhängigkeiten nach der Linksreduktion an (FD_L).

   ii. Führen Sie eine Rechtsreduktion des Ergebnisses der Linksreduktion (FD_L) durch. Geben Sie die Menge funktionaler Abhängigkeiten nach der Rechtsreduktion an (FD_R).

   iii. Bestimmen Sie eine kanonische Überdeckung FD_C von FD auf Basis des Ergebnisses der Rechtsreduktion (FD_R).

d) Zerlegen Sie R mit FD_C mithilfe des *Synthesealgorithmus* in 3NF.

e) Prüfen Sie *für alle Relationen* der Zerlegung aus d), ob sie jeweils in BCNF sind.

Fortsetzung nächste Seite!

4. Transaktionen

a) Erläutern Sie kurz die Konzepte Atomarität und Isolation.
b) Betrachten Sie den folgenden Schedule S:

T_1	T_2	T_3
$r_1(x)$		
	$r_2(x)$	
	$r_2(y)$	
	$w_2(x)$	
	c_2	
$r_1(y)$		
		$r_3(y)$
		$w_3(y)$
		c_3
$w_1(y)$		
c_1		

Geben Sie den Ausgabeschedule (einschließlich der Operationen zur Sperranforderung und -freigabe) im strikten Zweiphasen-Sperrprotokoll für den obigen Eingabeschedule S an.
Hinweis: Sollte während der Ausführung ein Deadlock auftreten, führen Sie ein Rollback einer der beteiligten Transaktionen durch.

Fortsetzung nächste Seite!

Teilaufgabe 2

Aufgabe 1 (Wissen)

a) Ordnen Sie die unten stehenden Aussagen entsprechend ihres Wahrheitsgehalts in einer Tabelle der folgenden Form ein:

Kategorie	*WAHR*	*FALSCH*
X	X1, X3	X2
Y	Y2	Y1
...	...	...

A-	Allgemein
A1	Im Software Engineering geht es vor allem darum qualitativ hochwertige Software zu entwickeln.
A2	Software Engineering ist gleichbedeutend mit Programmieren.
B-	Vorgehensmodelle
B1	Die Erhebung und Analyse von Anforderungen sind nicht Teil des Software Engineerings.
B2	Agile Methoden eignen sich besonders gut für die Entwicklung komplexer und sicherer Systeme in verteilten Entwicklerteams.
B3	Das Spiralmodell ist ein Vorläufer sogenannter Agiler Methoden.
C-	Anforderungserhebung
C1	Bei der Anforderungserhebung dürfen in keinem Fall mehrere Erhebungstechniken (z.B. Workshops, Modellierung) angewendet werden, weil sonst Widersprüche in Anforderungen zum Vorschein kommen könnten.
C2	Ein Szenario beinhaltet eine Menge von Anwendungsfällen.
C3	Nicht-funktionale Anforderungen sollten, wenn möglich, immer quantitativ spezifiziert werden.
D-	Architekturmuster
D1	Schichtenarchitekturen sind besonders für Anwendungen geeignet, in denen Performance eine wichtige Rolle spielt.
D2	Das Blackboard Muster ist besonders für Anwendungen geeignet, in denen Performance eine wichtige Rolle spielt.
D3	„Dependency Injection" bezeichnet das Konzept, welches Abhängigkeiten zur Laufzeit reglementiert.
E-	UML
E1	Sequenzdiagramme beschreiben Teile des Verhaltens eines Systems.
E2	Zustandsübergangsdiagramme beschreiben das Verhalten eines Systems.
E3	Komponentendiagramme beschreiben die Struktur eines Systems.
F-	Entwurfsmuster
F1	Das MVC Pattern verursacht eine starke Abhängigkeit zwischen Datenmodell und Benutzeroberfläche.
F2	Das Singleton Pattern stellt sicher, dass es zur Laufzeit von einer bestimmten Klasse höchstens ein Objekt gibt.
F3	Im Kommando Entwurfsmuster (eng.: „Command Pattern") werden Befehle in einem sog. Kommando-Objekt gekapselt, um sie bei Bedarf rückgängig zu machen.
G-	Testen
G1	Validation dient der Überprüfung von Laufzeitfehlern.
G2	Testen ermöglicht sicherzustellen, dass ein Programm absolut fehlerfrei ist.
G3	Verifikation dient der Überprüfung, ob ein System einer Spezifikation entspricht.

b) Nennen Sie jeweils zwei unterschiedliche Verfahren mit welchen in der Praxis die Prozesse der „Validierung" bzw. der „Verifikation" durchgeführt werden.

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

d) Nennen Sie fünf Herausstellungsmerkmale des „eXtreme Programming" Ansatzes.

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

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

Fortsetzung nächste Seite!

Aufgabe 2 (UML: Klassendiagramme)
Betrachten Sie folgende natürlich-sprachliche Spezifikation eines Dateisystems:

Ein Directory hat einen Namen und kann

- *Dateien,*
- *harte Verweise („hard links") auf eine Datei,*
- *symbolische Verweise („soft links") auf eine Datei oder ein Directory,*
- *sowie selbst wieder Directories*

enthalten. Die Wurzel dieses Directory-Baums wird mit „/" bezeichnet.

Modellieren Sie obigen Sachverhalt als UML-Klassendiagramm und berücksichtigen Sie dabei folgende Anforderungen daran:

- Ordnen Sie die Operationen `rename` (umbenennen), `delete` (löschen) und `dir` (Inhaltsverzeichnis zurückgeben) den richtigen Klassen zu und führen Sie sie mitsamt ihren Signaturen an. Dabei gibt `delete` zurück, ob das Löschen erfolgreich war oder nicht (z.B. weil der zu löschenden Ordner nicht leer war).
- Verwenden Sie für Ihr Klassendiagramm die in diesem Zusammenhang sinnvollen Abstraktionen der Generalisierung/Spezialisierung sowie der Aggregation/Komposition.
- Geben Sie sinnvolle Multiplizitäten bei Assoziationen an und erläutern Ihre Wahl kurz.

Fortsetzung nächste Seite!

Aufgabe 3 (Testen)

Beim strukturellen Testen wird versucht, möglichst viele Pfade im Kontrollflussgraphen eines Programms abzudecken. Die Adäquatheit der Tests misst man dabei häufig mittels Überdeckungskriterien wie etwa Anweisungsüberdeckung, Zweigüberdeckung, Bedingungsüberdeckung und Pfadüberdeckung.

Gegeben sei folgendes Problem (Spezifikation):
Für eine Zahl $x > 0$ soll die Summe aller Zahlen gebildet werden, die kleiner als x und Vielfache > 0 von 3 oder 5 sind. Für $x \leq 0$ soll 0 zurückgegeben werden.

Beispiel: Angenommen x wäre 7, dann sind 3, 5 und 6 alle Vielfachen von 3 und 5, die gleichzeitig kleiner als 7 sind. Die Summe dieser Zahlen ist 14.

Gegeben sei folgendes Programmstück zur Ermittlung dieser speziellen Summe:

```
public static long specialSums(int until) {
    long sum = 0;
    if (until > 0) {
        for (int i = 1; i <= until; i++) {
            if (i % 3 == 0 || i % 5 == 0) {
                sum += i;
            }
        }
    }
    return sum;
}
```

a) Zeichnen Sie den zum Programm gehörigen Kontrollflussgraphen.

b) Skizzieren Sie anhand der Bedingung (`i % 3 == 0 || i % 5 == 0`) den Unterschied zwischen einfacher und mehrfacher Bedingungsüberdeckung.

c) Nennen Sie Testfälle, die mindestens benötigt werden, um vollständige Anweisungsüberdeckung, Zweigüberdeckung und mehrfache Bedingungsüberdeckung bei dem oben beschriebenen Code zu erreichen. Geben Sie für jeden Testfall die laut Spezifikation erwartete Ausgabe an.

d) Welcher Fehler hat sich in obigem Code eingeschlichen und welches ist der Testfall mit dem kleinsten Eingabewert, der diesen Fehler findet?

e) Warum kann man für ein realistisches Projekt im Regelfall keine vollständige Pfadüberdeckung herstellen?

Fortsetzung nächste Seite!

Aufgabe 4 (UML: Anwendungsfall- und Sequenzdiagramme)

Gegeben ist die folgende Situation: Ihre Firma hat eine Ausschreibung gewonnen, welche die Implementierung eines neuen IT-Systems zur Vereinfachung der Ausführung von Geschäftsprozessen umfasst, einschließlich der Aufnahme von Kundeninformationen im Außendienst. Es wurden bereits Anforderungen in Form von Anwendungsfällen aufgenommen.

Sie betrachten nun den folgenden Anwendungsfall im Detail:

Anwendungsfall: **Kundendaten Online erfassen**	
Kurzbeschreibung	Die Stammdaten des Kunden werden Online im Web-Frontend erfasst.
Akteur(e)	Kunde, System
Vorbedingung	Kunde ist noch nicht im System erfasst.
Nachbedingung	Kunde ist im System erfasst.
Normalverlauf	
1. Der Kunde klickt auf der Webseite auf den Link „Als neuer Kunde anmelden“ 2. Das System zeigt die erste Webseite für die Anmeldung. 3. Der Kunde gibt seine persönlichen Daten (Name, Vorname, Titel, Anrede, *(optional) Geburtsname*, Geschlecht, Geburtsdatum, Geburtsort, Wohnanschrift, Telefonnummer, *(optional) Faxnummer*, E-Mail-Adresse) ein. 4. Das System zeigt die zweite Webseite für die Anmeldung. 5. Der Kunde wählt ein Zahlungsverfahren aus und gibt danach die Zahlungs- bzw. Rechnungsdaten ein. 6. Das System prüft die Zahlungs- bzw. Rechnungsdaten. 7. Sind die Daten gültig, zeigt das System die dritte Webseite der Anmeldung. 8. Der Kunde gibt einen Benutzernamen und ein Passwort ein, womit der Anmeldeprozess abgeschlossen wird.	
Alternativverlauf	
Beginn wie Normalverlauf 1-6 7a. Sind die Daten ungültig, zeigt das System eine Aufforderung zur Korrektur. 7b. Der Kunde ändert seine angegebenen Zahlungs- bzw. Rechnungsdaten. 7c. Das System prüft die Zahlungs- bzw. Rechnungsdaten. 7d. Sind die Daten gültig, übernimmt das System die Korrektur und zeigt die dritte Webseite der Anmeldung. Andernfalls wird die Anwendung beendet. Ende gemäß Normalverlauf 8	

Erstellen Sie ein UML-Sequenzdiagramm, das Normal- und Alternativverlauf für den oben beschriebenen Anwendungsfall darstellt.

Frühjahr 2017

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer
Kennzahl: _____	**Frühjahr 2017**	**66116**
Kennwort: _____		
Arbeitsplatz-Nr.: _____		

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: **16**

Bitte wenden!

Thema Nr. 1

Teilaufgabe 1

1. Modellierung

Modellieren Sie mit Hilfe des Entity-Relationship-Modells die Formel 1. Jedes Team hat pro Saison 2 Fahrer, die jeweils ein eigenes Auto fahren. Für die Siegerehrung ist wichtig zu wissen, welcher Nationalität die Fahrer angehören.
Es gibt mehrere Rennen (8 bis 20), die man über den Ort und das Jahr identifiziert. Die Platzierungen der Fahrer müssen für jedes Rennen gespeichert werden. Pro Rennen werden Punkte gemäß folgender Tabelle vergeben:

Platzierung	Punkte
1	25
2	18
3	15
4	12
5	10
6	8
7	6
8	4
9	2
10	1

Der Fahrer mit der höchsten Gesamtpunktzahl wird Weltmeister. Das Team, dessen beide Fahrer zusammen die meisten Punkte erzielt haben, wird Konstrukteursweltmeister. Überführen Sie Ihren ER-Entwurf in ein relationales Schema. Markieren Sie Schlüssel und Fremdschlüssel.

- Formulieren Sie diese beiden Abfragen, die den Weltmeister bzw. den Konstrukteursweltmeister ermitteln, jeweils in SQL.
- Formulieren Sie in SQL die Abfrage: Welcher Fahrer hat die meisten Siege in der Saison 2015 erzielt?
- Formulieren Sie in SQL die Abfrage: Welcher Fahrer war in keinem Rennen besser als Hamilton?

Fortsetzung nächste Seite!

2. Datenbanksysteme: Aufbau eines B-Baums

Konstruieren Sie einen B-Baum, dessen Knoten maximal 4 Einträge enthalten können, indem Sie der Reihe nach diese Suchschlüsssel einfügen:

- 8, 10, 12, 20, 5, 30, 25, 11
- Anschließend löschen Sie den Eintrag mit dem Suchschlüssel 8.

Zeigen Sie jeweils graphisch den entstehenden Baum nach relevanten Zwischenschritten; insbesondere nach Einfügen der 5 sowie nach dem Einfügen der 11 und nach dem Löschen der 8.

3. Datenbanksysteme: Normalisierung

Gegeben sei dieses Relationenschema:

Allerlei: {[MatrikelNr, SName, VNr, VTitel, PersNr, PName]}

Hier werden also Studenten (mit den Attributen MatrikelNr, SName), Vorlesungen (mit den Attributen VNr, VTitel) sowie Professoren (mit den Attributen PersNr und PName) modelliert und in Beziehung gesetzt. Es sollen natürlich viele Studenten dieselbe Vorlesung hören können, aber nur ein/e Professor/in kann die Vorlesung lesen:

- Geben Sie die nicht-trivialen funktionalen Beziehungen der Relation Allerlei an.
- Geben Sie alle Kandidatenschlüssel der Relation an.
- Normalisieren Sie die Relation systematisch mit Erläuterung.

Fortsetzung nächste Seite!

4. Normalisierung und SQL

Gegeben sei eine Relation Zehnkampf mit dem Schema

{SportlerName, Disziplin, ZeitHöheWeite, Punkte}:

SportlerName	**Disziplin**	**ZeitHöheWeite**	**Punkte**
"Jim Knopf"	"100 Meter"	10,89	986
"Usain Bolt"	"100 Meter"	9,81	1723
"Jim Knopf"	"StabHoch"	4,56	976
"Usain Bolt"	"StabHoch"	2,80	430
.....	.....	.....	.....

Für die Bearbeitung dürfen folgende Abkürzungen verwendet werden:

SN für SportlerName

D für Disziplin

ZHW für ZeitHöheWeite

P für Punkte

- Welche funktionalen Abhängigkeiten gelten?
- Welches sind die (Kandidaten-) Schlüssel der Relation?
- Normalisieren Sie die Relation.
- Formulieren Sie eine Abfrage in SQL, die die Sportler sortiert nach der Gesamtzahl ihrer Punkte ausgibt. Verwenden Sie die Original-Relation Zehnkampf <u>ohne</u> der Normalisierung.
- Formulieren Sie folgende Abfrage in SQL <u>und</u> in der Relationenalgebra. Verwenden Sie die Original-Relation Zehnkampf <u>ohne</u> der Normalisierung:
  Wer ist in der Disziplin „StabHoch“ besser als „Usain Bolt“?
- Formulieren Sie folgende Abfragen in SQL. Verwenden Sie die Original-Relation Zehnkampf <u>ohne</u> der Normalisierung:
  - Wer hat <u>insgesamt</u> mehr Punkte erzielt als „Jim Knopf“?
  - Wer hat Medaillen gewonnen? (Nur Standard SQL 92 verwenden und keine Limit-Klausel!)
  - Wer hat in mindestens einer Disziplin mehr Punkte erzielt als „Usain Bolt“ (egal in welcher)?

Fortsetzung nächste Seite!

Teilaufgabe 2

Aufgabe 1 (Qualitätskriterien von Software)

Nennen Sie vier Qualitätskriterien von Software. Geben Sie davon mindestens ein Qualitätskriterium an, das durch objektorientierte Softwareentwicklung besonders unterstützt wird und begründen Sie dies.

Aufgabe 2 (Vorgehensmodelle)

Ein wichtiges Vorgehensmodell in der Softwareentwicklung ist das V-Modell. Zeichnen Sie das V-Modell, so dass deutlich wird, welche Testphasen welchen Entwicklungsphasen gegenüberstehen.

Aufgabe 3 (Modellierung von Interaktionen)

Am Eingang eines Hallenbads befindet sich ein Drehkreuz, das mit einer Lichtanzeige verbunden ist. Das Drehkreuz und die Lichtanzeige werden durch das Klassendiagramm in Abb. 1 modelliert.

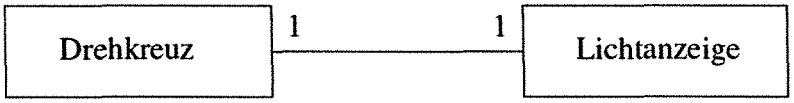


Abbildung 1: Klassendiagramm für Drehkreuz mit Lichtanzeige

Durch das Drehkreuz können Badegäste, die im Besitz einer gültigen Eintrittskarte sind, in das Hallenbad eintreten. Bevor das Drehkreuz passiert werden kann, muss ein Badegast die Eintrittskarte in einen Schlitz an der Seite des Drehkreuzes eingeben. Es gibt einen Anwendungsfall *Am Drehkreuz eintreten*, der durch folgendes Primär- und Sekundärszenario beschrieben wird. Als Vorbedingung für den Anwendungsfall gilt, dass das Drehkreuz gesperrt ist.

Primärszenario (Drehkreuz passieren):

1. Der Badegast gibt seine Eintrittskarte am Drehkreuz ein.
2. Das Drehkreuz veranlasst, dass an der Lichtanzeige das grüne Licht eingeschaltet wird, und gibt dann die Eintrittskarte wieder aus.
3. Der Badegast entnimmt die Karte, woraufhin sich das Drehkreuz entsperrt.
4. Der Badegast passiert das Drehkreuz, woraufhin das Drehkreuz veranlasst, dass an der Lichtanzeige das grüne Licht wieder ausgeschaltet wird.
5. Danach sperrt sich das Drehkreuz wieder.

Fortsetzung nächste Seite!

Sekundärszenario (Karte ungültig):

1. Der Badegast gibt seine Eintrittskarte am Drehkreuz ein.
2. Das Drehkreuz veranlasst, dass an der Lichtanzeige das rote Licht eingeschaltet wird, und gibt dann die Eintrittskarte wieder aus.
3. Der Badegast entnimmt die Karte, woraufhin das Drehkreuz veranlasst, dass an der Lichtanzeige das rote Licht wieder ausgeschaltet wird.

In den folgenden Teilen a) und b) sind Sequenzdiagramme zu erstellen, in denen der Badegast als Aktor und Interaktionen durch Nachrichten zu modellieren sind. Beachten Sie, dass sich ein Objekt innerhalb des Systems auch selbst eine Nachricht schicken kann.

a) Geben Sie ein Sequenzdiagramm an, das die Interaktionen zwischen einem Badegast, dem Drehkreuz und der Lichtanzeige gemäß des oben beschriebenen Primärszenarios zeigt.

b) Geben Sie ein Sequenzdiagramm an, das die Interaktionen zwischen einem Badegast, dem Drehkreuz und der Lichtanzeige gemäß des oben beschriebenen Sekundärszenarios zeigt.

c) Welche Operationen müssen zu der Klasse *Drehkreuz* hinzugefügt werden, damit die Nachrichten aus den Sequenzdiagrammen in den beiden Aufgabenteilen a) und b) an das Drehkreuz durch Operationsaufrufe realisiert werden können?

d) Welche Operationen müssen zu der Klasse *Lichtanzeige* hinzugefügt werden, damit die Nachrichten aus den Sequenzdiagrammen in den beiden Aufgabenteilen a) und b) an die Lichtanzeige durch Operationsaufrufe realisiert werden können?

Fortsetzung nächste Seite!

<u>Aufgabe 4</u> (Zustandsdiagramme und deren Implementierung)

Ein einfacher Taschenrechner wird durch die Klasse *Calculator* in Abb. 2 modelliert. Das Attribut *ac* repräsentiert den Accumulator, das Attribut *m* das Memory des Taschenrechners und *Calculator*() den Konstruktor. Das Verhalten des Taschenrechners wird durch das Zustandsdiagramm in Abb. 3 beschrieben. Alle Ereignisse an den Transitionen repräsentieren Operationsaufrufe (*call events*) für die in der Klasse angegebenen Operationen.

Der Taschenrechner unterstützt die Addition nicht-negativer ganzer Zahlen. Die Eingabe einer Zahl erfolgt ziffernweise durch Eingabe von Ziffern zwischen 0 und 9. Die Eingabe einer Ziffer *k* wird durch ein Ereignis der Form *digit*(*k*) modelliert. Das Drücken der Tasten +, = und AC wird durch die Ereignisse *plus*(), *result()* und *AC()* modelliert. Ereignisse, für die es in einem Zustand keine ausgehende Transition gibt, können zwar eintreten, haben in diesem Zustand aber keine Wirkung.

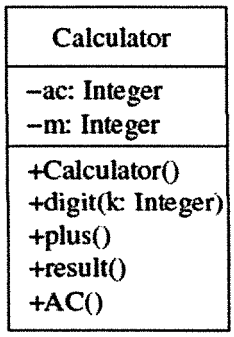


Abbildung 2: Klasse Calculator

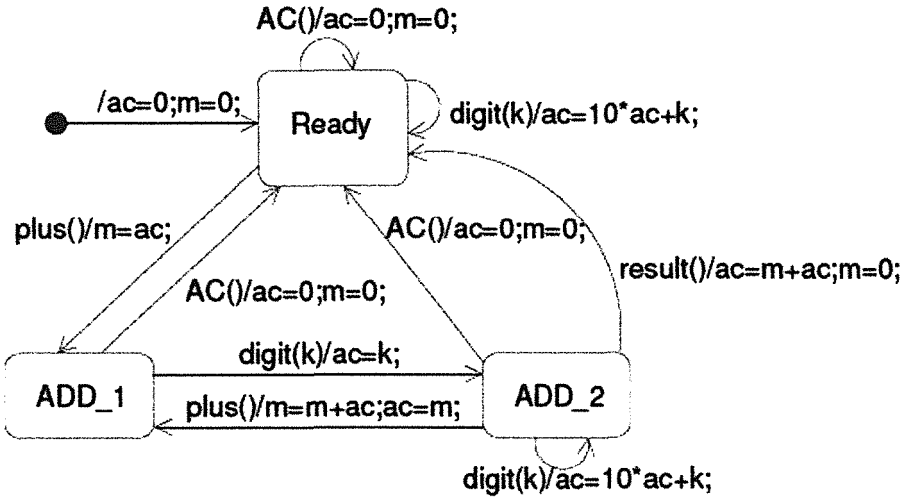


Abbildung 3: Zustandsdiagramm für Calculator

Fortsetzung nächste Seite!

a) Für ein neu erzeugtes Calculator-Objekt sollen der Reihe nach folgende Ereignisse eintreten: *digit(5), plus(), digit(2), plus(), digit(3), result(), plus(), digit(6), digit(1), result()*. In welchem Zustand ist das Calculator-Objekt zu Beginn und wie sind dann seine Attributwerte? Geben Sie an, in welchem Zustand das Calculator-Objekt nach jedem einzelnen Ereignis ist und wie dann seine Attributwerte jeweils sind.

b) Implementieren Sie die Klasse *Calculator* in einer objektorientierten Programmiersprache Ihrer Wahl (verwendete Sprache bitte angeben), so dass das in dem Zustandsdiagramm beschriebene Verhalten realisiert wird. Insbesondere sind der Konstruktor und alle angegebenen Operationen zu implementieren. Die Implementierung soll nach einer der beiden folgenden Strategien erfolgen:

(A) Einführung eines Enumerationstyps zur Darstellung der Zustände und Implementierung der Operationen (Methoden) durch Fallunterscheidung nach dem jeweiligen Zustand.

(B) Anwendung des Design Patterns namens *State*, in dem Zustände durch Objekte geeigneter zusätzlicher Klassen repräsentiert werden.

Fortsetzung nächste Seite!

Aufgabe 5 (Systemarchitektur mit Interfaces)

Gegeben sind die in Abb. 4 dargestellten Klassen zur Modellierung eines objektorientierten Systems.

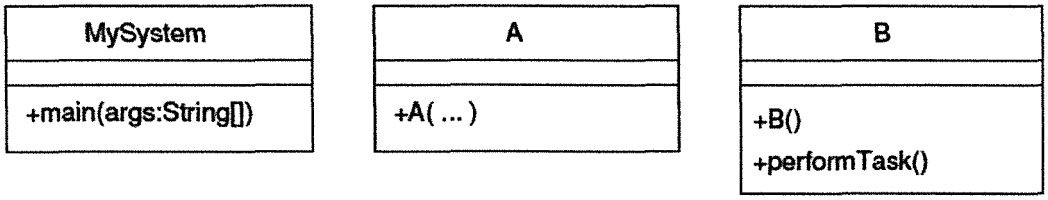


Abbildung 4: Klassen eines objektorientierten Systems

Zur Laufzeit soll

i) genau ein Objekt der Klasse *A* mit genau einem Objekt der Klasse *B* fest verbunden werden,
ii) das Objekt der Klasse *A* die Operation *performTask()* des *B*-Objekts aufrufen.

Zur Programmierzeit sollen sich die Klassen *A* und *B* nicht kennen (d.h. es gibt keine Abhängigkeiten zwischen den beiden Klassen). Zur Lösung ist ein Interface zu verwenden. Erstellen Sie ein Klassendiagramm, das die drei Klassen *MySystem*, *A* und *B* beinhaltet und zusätzlich Folgendes zeigt:

a) das benötigte Interface und welche Operationen es anbietet,

b) eine gerichtete Assoziation (mit Rollennamen und Multiplizität am gerichteten Assoziationsende), die zeigt, welche Klasse das Interface benutzt,

c) eine Realisierungsbeziehung, die zeigt, welche Klasse das Interface implementiert,

d) einen geeigneten formalen Parameter für den Konstruktor von *A*.

Des Weiteren sind in einer objektorientierten Programmiersprache Ihrer Wahl anzugeben:

e) die Implementierung des Konstruktors der Klasse *A*,

f) die Implementierung der main-Methode der Klasse *MySystem*.

Thema Nr. 2

Teilaufgabe 1

Aufgabe 1: ER-Modellierung

Stadt, Land, Fluss mal anders: Die relationale Datenbank eines geographischen Systems beinhaltet Städte, Länder und Flüsse mit folgenden Einzelheiten:

- Eine Stadt wird durch ihren eindeutigen Namen und das Land gekennzeichnet, in dem sie liegt. Außerdem hat jede Stadt eine Einwohnerzahl.
- Ein Land hat einen eindeutigen Namen, eine Fläche und ebenfalls eine Einwohnerzahl. Ein Land kann an mehrere Länder angrenzen. In einem Land liegt mindestens eine Stadt, wobei jede Stadt nur in genau einem Land liegt. Außerdem hat jedes Land eine Hauptstadt.
- Ein Fluss wird durch seinen Namen bestimmt und hat eine gewisse Länge. Jeder Fluss kann durch mehrere Städte und auch durch mehrere Länder fließen und kann am Ende in einen anderen Fluss münden. Eine Stadt kann an mehreren Flüssen liegen und durch ein Land können mehrere Flüsse fließen.

Erstellen Sie ein Entity-Relationship-Diagramm für obige Datenbank. Markieren Sie die Funktionalitäten der Relationships und unterstreichen Sie den Primärschlüssel jeder Entität.

Aufgabe 2: Tupelkalkül

Gegeben sei das folgende Datenbankschema. Alle Attribute seien vom Datentyp String.

```
Wirt(WNr, WName, WStadt)
Speise(SNr, Bezeichnung, Preis)
Gast(GNr, GName, GStadt)
Bestellung(WNr, SNr, GNr)
```

Geben Sie für die folgende, verbal formulierte Anfrage einen äquivalenten Ausdruck im **Tupelkalkül** an.
Gesucht sind Nummer, Name und Stadt der Gäste, die niemals ein Schnitzel bestellt haben.

Fortsetzung nächste Seite!

Aufgabe 3: Relationale Algebra

Gegeben seien folgende Relationen:

X

A	B	C	D
1	4	6	3
6	2	5	1
4	5	6	3
1	4	5	2
1	4	6	2
3	3	5	2
4	3	1	2
4	5	6	2

Y

C	D	E
5	2	3
6	2	1
6	3	1

Z

D	F
6	2
5	6
1	3

Geben Sie die Ergebnisrelationen folgender Ausdrücke der relationalen Algebra als Tabellen an.

1. $X \underset{D=F}{\bowtie} \sigma_{F>2}(\pi_F(Z))$

2. $(\pi_{C,D}(X) - \pi_{C,D}(Y)) \times \pi_D(Z)$

3. $\pi_{A,D}(X) \div \pi_D(Y)$

Aufgabe 4: SQL-Abfragen

Die Spieler in einer Fußballmannschaft in der Saison 2015 werden in folgender Relation verwaltet:

```
Saison2015(PersonalID, OK1, OK2, AnzahlSpiele, AnzahlTore).
```

Alle Attribute sind Integer. Für jeden Spieler wird ein Tupel mit der entsprechenden PersonalID angelegt, die ihn über die gesamte Vereinshistorie hinweg eindeutig identifiziert. Die Werte für die Attribute OK1 und OK2 haben den Defaultwert 0. Besteht ein Spieler den 1. Fitnesstest des Jahres, dann wird OK1 auf 1 aktualisiert. Besteht ein Spieler den 2. Fitnesstest des Jahres, dann wird OK2 auf 1 aktualisiert. Die Werte der Attribute AnzahlSpiele und AnzahlTore werden nach jedem Spiel, in dem der Spieler angetreten ist, aktualisiert.
Für die Saison 2014 liegt eine weitere Relation vor, die die Spieler des Jahres 2014 und ihre Ergebnisse in den Fitnesstests des Jahres 2014 in gleicher Weise abspeichert:

```
Saison2014(PersonalID, OK1, OK2, AnzahlSpiele, AnzahlTore).
```

Fortsetzung nächste Seite!

Bitte beachten Sie: Ein Spieler, der in der Saison 2015 in dieser Mannschaft spielt, muss nicht schon 2014 in dieser Mannschaft gespielt haben. Die PersonalID eines Spielers ist eindeutig über alle Saisonen hinweg.
Wegen finanzieller Engpässe muss sich der Verein von einigen Spielern trennen und möchte daher:

1. in **einer einzigen SQL-Abfrage** alle Spieler ermitteln,
   - die im Jahr 2015 beide Fitnesstests bestanden haben **oder**
   - die in den Jahren 2014 und 2015 zusammen mindestens 3 von 4 Fitnesstests bestanden haben.

   Geben Sie ein SQL-Statement an, das das gewünschte Ergebnis liefert.

2. in **einer einzigen SQL-Abfrage** die Spieler ermitteln, die in mindestens einer der beiden Saisonen in weniger als fünf Spielen angetreten sind.
   Geben Sie ein SQL-Statement an, das das gewünschte Ergebnis liefert.

3. in **einer einzigen SQL-Abfrage** die Spieler ermitteln, die in jeder der beiden Saisonen in mindestens einem Spiel angetreten sind und bei denen das Verhältnis der geschossenen Tore pro Spiel im Durchschnitt in mindestens einer der beiden Saisonen unter 10 % liegt.
   Geben Sie ein SQL-Statement an, das das gewünschte Ergebnis liefert.

4. in **einer einzigen SQL-Abfrage** die Spieler ermitteln, die 2015 mehr Tore geschossen haben als 2014.
   Geben Sie ein SQL-Statement an, das das gewünschte Ergebnis liefert.

5. in **einer einzigen SQL-Abfrage** ermitteln, welche Spieler 2015 die meisten Tore geschossen haben (mehrere Spieler können gleich viele Tore geschossen haben).
   Geben Sie ein SQL-Statement an, das das gewünschte Ergebnis liefert.

Aufgabe 5: Entwurfstheorie

In der folgenden Datenbank sind die Ausleihvorgänge einer Bibliothek gespeichert:

Ausleihe	LNr	Name	Adresse	BNr	Titel	Kategorie	ExemplarNr
	1	Müller	Winklerstr.	1	Datenbanksysteme	Informatik	1
	1	Müller	Winklerstr.	1	Datenbanksysteme	Informatik	2
	2	Huber	Friedrichstr.	2	Anatomie I	Medizin	5
	2	Huber	Friedrichstr.	3	Harry Potter	Literatur	20
	3	Meier	Bismarkstr.	4	OODBS	Informatik	1
	4	Meier	Marktpl.	5	Pippi Langstrumpf	Literatur	1

Für die Datenbank gilt:
Jeder Leser hat eine eindeutige Lesernummer (LNr), einen Namen und eine Adresse. Ein Buch hat eine Buchnummer (BNr), einen Titel und eine Kategorie. Es kann mehrere Exemplare eines Buches geben, welche durch eine, innerhalb einer Buchnummer eindeutigen, Exemplarnummer unterschieden werden.

Fortsetzung nächste Seite!

1. Beschreiben Sie kurz, welche Redundanzen in der Datenbank vorhanden sind und welche Anomalien auftreten können.

2. Nachfolgend sind alle nicht-trivialen funktionalen Abhängigkeiten, welche in der obigen Datenbank gelten, angegeben:

(1) *LNr* → *Name*
(2) *LNr* → *Adresse*
(3) *BNr* → *Titel*
(4) *BNr* → *Kategorie*
(5) *LNr, BNr, ExemplarNr* → *Name, Adresse, Titel, Kategorie*

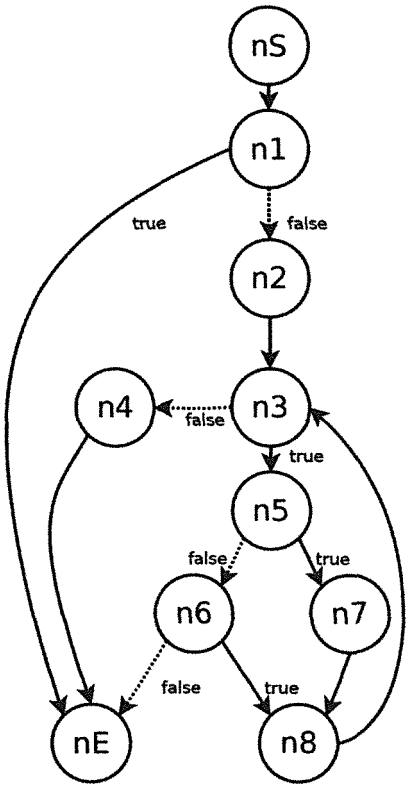
Einziger Schlüsselkandidat ist {*LNr, BNr, ExemplarNr*}.
Überführen Sie das Schema mit Hilfe des Synthesealgorithmus für 3NF in die dritte Normalform.

Teilaufgabe 2

Aufgabe 1: „Testen"

Gegeben sei folgende Methode und ihr Kontrollflussgraph:

```
int binToInt(String bin) {
    if (bin.isEmpty())
        return -1;
    int place = 1, value = 0;
    int length = bin.length() - 1;
    for (int i = length; i >= 0; --i) {
        char ch = bin.charAt(i);
        if (ch == '1') {
            value += place;
        } else if (ch == '0') {
            // do nothing
        } else {
            return -1;
        }
        place *= 2;
    }
    return value;
}
```



a) Geben Sie je einen Repräsentanten aller Pfadklassen ***im Kontrollflussgraphen*** an, die zum Erzielen einer vollständigen

   i. Verzweigungsüberdeckung (Branch-Coverage, C_1)

   ii. Schleife-Inneres-Überdeckung (Boundary-Interior-Coverage, $C_{\infty,2}$)

mit ***minimaler*** Testfallanzahl genügen würden.

Fortsetzung nächste Seite!

b) Welche der vorangehend ermittelten Pfade für die C_1-Überdeckung sind mittels Testfälle tatsächlich überdeckbar („feasible")? Falls der Pfad ausführbar ist, geben Sie bitte den Testfall (also den Wert des Parameters bin) an – andernfalls begründen Sie kurz, weshalb der Pfad nicht überdeckbar ist.

c) Bestimmen Sie anhand des Kontrollflussgraphen des obigen Code-Fragments die maximale Anzahl linear unabhängiger Programmpfade, also die zyklomatische Komplexität nach McCabe.

d) Kann für dieses Modul eine 100%-ige Pfadüberdeckung erzielt werden? Begründen Sie kurz Ihre Antwort.

e) Geben Sie zu jedem Knoten die jeweilige Datenfluss-Annotation (defs bzw. uses, sofern überhaupt vorhanden) für jede betroffene Variable in der zeitlichen Reihenfolge ihres Auftretens zur Laufzeit an.

Aufgabe 2: „UML-Klassendiagramm"

Das Warenhaussystem des Online-Händlers Rhein soll modelliert werden. Erstellen Sie dazu ein UML-Klassendiagramm basierend auf den folgenden Angaben. Dabei sind Attribute nur dann einzutragen, wenn es sich um Klassenattribute handelt.

Den Kern des Systems bildet die Verwaltung des Rhein-Systems. Von dieser gibt es immer nur eine Objekt-Instanz. Zum System gehören ein Verkaufsservice und ein Inventar. Verkaufsservice, Inventar und die Objekt-Instanz sollen über öffentliche Getter-Methoden zur Verfügung gestellt werden.
Ein Inventar setzt sich aus einer Liste von Artikeln zusammen. Diese Liste soll über eine Getter-Methode zur Verfügung gestellt werden. Über eine Methode sollen der Liste neue Artikel hinzugefügt werden können.
Artikel stellen die Obermenge der im Webshop angebotenen Waren dar. Eine angebotene Ware muss dabei immer eine weiter spezifizierte Variante von „Artikel" sein. Artikel verfügen über einen Lagervorrat, einen Namen, eine Beschreibung und eine unverbindliche Preisempfehlung. Diese sollen über öffentliche Methoden auslesbar sein. Des Weiteren soll es möglich sein, den Lagerbestand um eine bestimmte Menge zu erhöhen.
Als Varianten des Artikels sind Bücher und DVDs zu spezifizieren. Diese verfügen über alle Eigenschaften von Artikeln, Bücher bieten aber zusätzlich ein Inhaltsverzeichnis und DVDs einen Regionalcode, die jeweils öffentlich per Methode zur Verfügung gestellt werden.
Ein Verkaufsservice verwaltet Angebote und Regeln, die diese Angebote Artikeln zuordnen. Neue Angebote und Regeln sollen hinzugefügt werden können. Für einen Artikel kann das günstigste Angebot abgefragt werden.
Zur Preisbestimmung ist ein Interface „Pricer" zu definieren, welches für einen Artikel einen Preis zurückliefern kann. Für dieses gibt es mehrere Implementierungen: den Standard-Pricer, der die unverbindliche Kaufempfehlung zurückliefert und eine andere Implementierung für die Angebote. Letztere kann zusätzlich noch den ursprünglichen Preis (diese Methode wird erst durch die konkreten, spezialisierten Angebote implementiert) und den prozentualen Preisunterschied zurückgeben.
Als Spezialisierungen der Angebote werden das wahre Angebot und das falsche Angebot bereitgestellt. Das wahre Angebot liefert die unverbindliche Preisempfehlung als ursprünglichen Preis und einen um 20 % reduzierten Preis als den aktuellen Preis des Artikels zurück. Das falsche Angebot liefert die unverbindliche Preisempfehlung als aktuellen Preis zurück und liefert als ursprünglichen Preis eine um 20 % verteuerte unverbindliche Preisempfehlung zurück.

Fortsetzung nächste Seite!

Die Regeln zur Zuordnung zwischen Angeboten und Artikeln werden über ein Interface dargestellt. Dieses bietet eine boolesche Funktion an, die zurückliefert, ob ein Angebot zu einem Artikel passt. Dieses Interface wird von einer Regel für hohen Lagerbestand und einer Regel für niedrigen Preis implementiert.

Aufgabe 3: „Zustandsdiagramme"

a) Erstellen Sie ein Zustandsdiagramm für den im Folgenden beschriebenen Parkautomaten. Modellieren Sie dazu soweit nötig sowohl Zustände und Transitionsbedingungen als auch die Eingangsaktionen der Zustände.

Der Parkautomat startet im bereiten Grundzustand und wartet auf Interaktionen. Wird eine Karte eingelegt, wird diese eingezogen und der Automat wechselt in die Prüfung. In dieser wird die Karte überprüft. Ist sie ungültig, so gibt der Automat die Karte wieder aus, der Vorgang wird abgebrochen und der Parkautomat wechselt wieder in die Bereitschaft. Ist die Karte gültig, so zeigt der Automat den zu zahlenden Preis an und wartet auf die Geldeingabe. Wird Geld eingeworfen, wird, solange der eingeworfene Betrag kleiner als der Preis ist, lediglich der angezeigte Preis aktualisiert und auf weiteres Geld gewartet. Ist der gewünschte Betrag eingeworfen worden, so gibt der Automat eine Parkmünze aus und wartet auf einen eventuellen Quittungswunsch. Erfolgt dieser wird die Quittung ausgegeben und in den Ursprungszustand gewechselt. Erfolgt dieser nicht, wird nach 5 Sekunden in den Ausgangszustand gewechselt.

b) Erweitern Sie das obige Diagramm sinnvoll so, dass wenn ein zu großer Betrag eingeworfen wird, entsprechend Wechselgeld ausgegeben wird.

Fortsetzung nächste Seite!

Aufgabe 4: „Formale Verifikation"

Sie dürfen im Folgenden davon ausgehen, dass keinerlei Under-/Overflows auftreten.
Gegeben sei folgende Methode mit Vorbedingung (pre-condition) $P := x \geq 0 \wedge y \geq 0$ und Nachbedingung (post-condition) $Q := x \cdot y = z$:

```
int mul(int x, int y) {
        // P
        int z = 0, i = 0;
        while (i++ != x) z += y;
        // Q
        return z;
}
```

Betrachten Sie dazu die folgenden drei Prädikate:

1. $I_1 := z + i \cdot y = x \cdot y$
2. $I_2 := \mathit{false}$
3. $I_3 := z + (x - i) \cdot y = x \cdot y$

a) Beweisen Sie *formal* (z. B. mit der Methode der schwächsten Vorbedingung) für jedes der drei Prädikate, ob es unmittelbar vor Betreten der Schleife in `mul` gilt oder nicht.

b) Weisen Sie *formal* nach, welche der drei Prädikate Invarianten des Schleifenrumpfs in `mul` sind und welche nicht.

c) Beweisen Sie *formal*, aus welchem der drei Prädikate die Nachbedingung gefolgert werden darf bzw. nicht gefolgert werden kann.

d) Skizzieren Sie den Beweis der totalen Korrektheit der Methode `mul`. Zeigen Sie dazu auch die Terminierung der Methode.

Herbst 2017

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer
Kennzahl: _____	**Herbst 2017**	**66116**
Kennwort: _____		
Arbeitsplatz-Nr.: _____		

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: **18**

Bitte wenden!

Thema Nr. 1

Teilaufgabe 1

1. Grundwissen Datenbanksysteme

1. Benennen Sie die vier ACID-Eigenschaften.

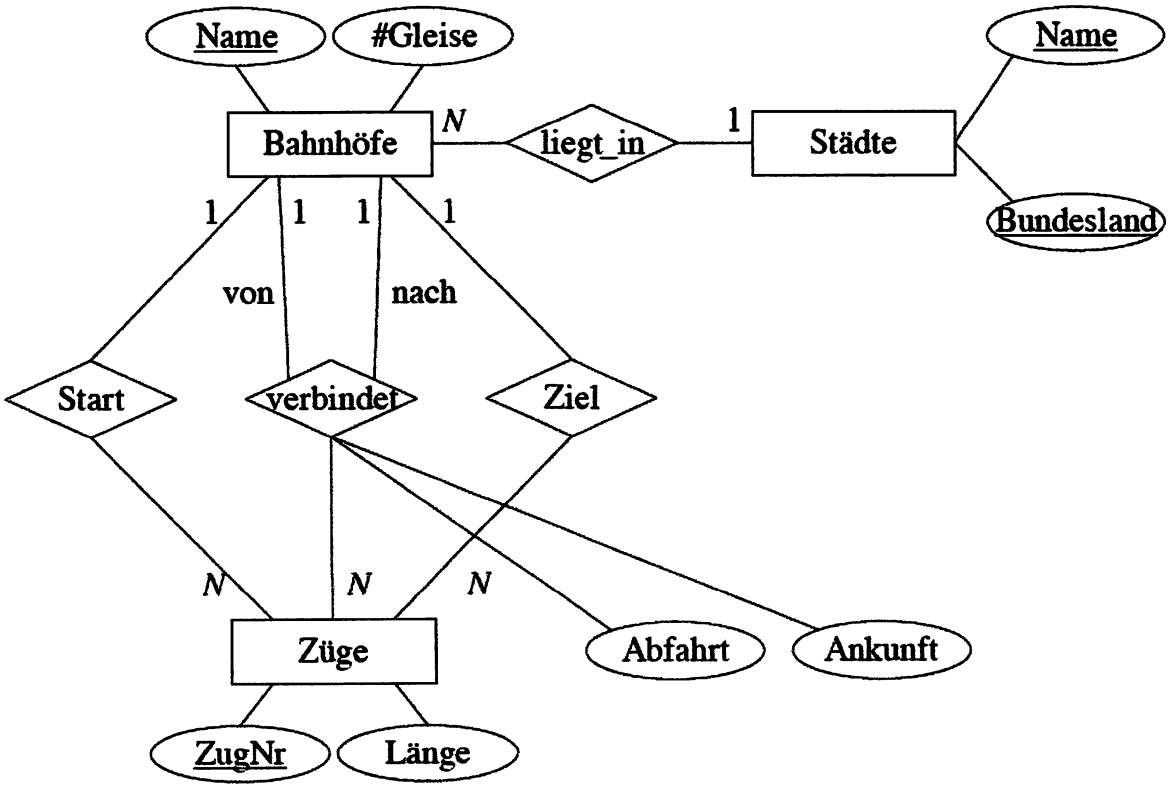
2. Grenzen Sie die Begriffe *Superschlüssel* und *Kandidatenschlüssel* mit einer kurzen Erklärung voneinander ab.

3. Nennen Sie zwei mögliche Vorteile, die durch den Zusammenschluss von Festplatten zu einem RAID-System entstehen.

4. Eignet sich eine Hash-Tabelle zur Bearbeitung von Bereichsanfragen? Begründen Sie Ihre Antwort und nennen Sie gegebenenfalls eine Alternative.

2. Relationale Modellierung

Das folgende *ER* -Diagramm modelliert ein Zugauskunftssystem.



Fortsetzung nächste Seite!

1. Setzen Sie das *ER*-Diagramm in ein relationales Schema um.

   - Fassen Sie, wenn möglich, Relationen zusammen.
   - Markieren Sie Primärschlüssel.
   - Markieren Sie Fremdschlüssel.
   - Wählen Sie sinnvolle Typen für die Attribute.

2. Geben Sie die Schemadefinitionen für die Tabellen *verbindet* und *Bahnhöfe* inklusive Primär- und Fremdschlüsseldefinitionen in SQL an.

3. Ist es in dem gegebenen Modell möglich einen Zug einzutragen, welcher von München nach Berlin fährt (Start & Ziel) und als Zwischenstation in Nürnberg hält? Begründen Sie Ihre Antwort und formulieren Sie die entsprechenden Befehle in SQL.
   Sie können davon ausgehen, dass es in der Datenbank bereits einen Bahnhof für Berlin, München und Nürnberg (mit selbigen Namen) gibt. Das Datum für Ankunft und Abfahrt ist dabei jeweils frei wählbar und muss nicht realistisch sein.

3. Konsistenzbedingungen

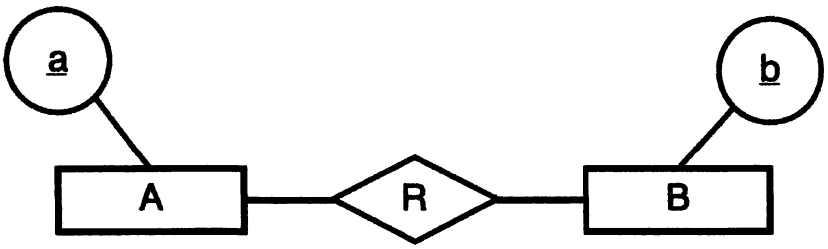
Betrachten Sie das folgende *ER*-Diagramm sowie die gegebenen Ausprägungen der Relationen A und B.

A:

a
1
2
3

B:

b
1
2
3



R_1:

a	b
1	1
1	1
2	1
2	2
3	2

R_2:

a	b
1	1
1	1
2	2
2	2

R_3:

a	b
1	1
1	1
1	1
2	1
3	2
3	2

Zeichnen Sie das angegebene *ER*-Diagramm ab und:

1. Annotieren Sie das Diagramm mit Min/Max-Angaben so, dass Ausprägung R_1 eine gültige Ausprägung der Beziehung R ist, nicht jedoch Ausprägungen R_2 bzw. R_3.

2. Fügen Sie dem Diagramm nun auch Funktionalitätsangaben hinzu, so dass die Ausprägung R_1 erlaubt ist.

Fortsetzung nächste Seite!

4. Normalformen

a) Geben Sie für die vier Relationen (R_0, R_1, R_2 und R_3) jeweils die höchste Normalform an (1. Normalform, 2. Normalform, 3. Normalform oder Boyce-Codd Normalform), welche die jeweilige Relation mit gegebenen FDs noch erfüllt.

$R_0(A,B,C,D)$

- $AB \rightarrow C$
- $C \rightarrow D$

$R_1(A,B,C,D)$

- $A,B \rightarrow C,D$
- $C \rightarrow A$

$R_2(A,B,C,D,E,F)$

Keine nicht-trivialen FDs.

$R_3(A,B,C,D,E,F)$

- $A,B \rightarrow C,D,E,F$
- $A \rightarrow E,F$

b) Gegeben sei die Relation $R(A,B,C,D,E,F,G)$ mit den FDs

1. $A \rightarrow BCDE$
2. $E \rightarrow FB$
3. $F \rightarrow A$

i) Geben Sie alle Kandidatenschlüssel an.

ii) Überführen Sie die Relation – wenn nötig mittels Synthesealgorithmus – in die 3. NF. Geben Sie ***alle Relationen in der 3. NF*** an und ***unterstreichen*** Sie in ***jeder*** einen ***Kandidatenschlüssel.***

Fortsetzung nächste Seite!

5. Schemadefinition

Gegeben ist die folgende Definition zweier Tabellen:

```
CREATE TABLE R2 (
        b integer not null,
        c integer unique,
        primary key (b)
);

CREATE TABLE R1 (
        a integer not null,
        b integer references R2,
        primary key (a)
);
```

Geben Sie jeweils an, ob das Statement syntaktisch korrekt ist und ob es von der gegebenen Datenbank ausgeführt werden kann.
Beantworten Sie jede der folgenden Fragen unabhängig von allen anderen, d.h. es liegt immer das hier gezeigte Schema vor und alle Relationen sind leer.

1. `DELETE FROM R1;`

2.
```
INSERT INTO R2 VALUES (1,1);
INSERT INTO R1 VALUES (1,1);
INSERT INTO R1 VALUES (2,1);
INSERT INTO R1 VALUES (3,1);
```

3.
```
INSERT INTO R2 VALUES (1,1);
INSERT INTO R2 VALUES (2,2);
INSERT INTO R1 VALUES (1,1);
DELETE FROM R2 WHERE b=a;
```

4. `INSERT INTO R1 SELECT * FROM R1;`

5. `DROP TABLE R2 FROM DATABASE;`

Fortsetzung nächste Seite!

6. Relationale Anfragen in SQL

Folgende Tabellen veranschaulichen eine Ausprägung eines Fluginformationssystems:

Flughäfen

Code	Stadt	Transferzeit (min)
LHR	London	30
LGW	London	20
JFK	New York City	60
EWR	New York City	35
MUC	München	30
FRA	Frankfurt	45
⋮	⋮	⋮

Verbindungen

ID	Von	Nach	Linie	Abflug (MEZ)	Ankunft (MEZ)
410	MUC	FRA	LH	2016-02-24 07:00:00	2016-02-24 08:10:00
411	MUC	FRA	LH	2016-02-24 08:00:00	2016-02-24 09:10:00
412	FRA	JFK	LH	2016-02-24 10:50:00	2016-02-24 19:50:00
⋮	⋮	⋮	⋮	⋮	⋮

Hinweise

- Formulieren Sie alle Abfragen in *SQL* – 92 (insbesondere sind LIMIT, TOP, FETCH FIRST, ROWNUM und dergleichen *nicht* erlaubt).
- Alle Datum/Zeit-Angaben erlauben arithmetische Operationen, beispielsweise wird bei der Operation *ankunft+transferzeit* die *transferzeit* auf den Zeitstempel *ankunft* addiert.
- Es müssen *keine* Zeitverschiebungen berücksichtigt werden. Alle Zeitstempel sind in MEZ.

1. Ermitteln Sie die Städte, in denen es mehr als einen Flughafen gibt.
2. Ermitteln Sie die Städte, in denen man mit der Linie „LH“ an mind. zwei verschiedenen Flughäfen landen kann.
3. Ermitteln Sie die Flugzeit des kürzesten Direktflugs von München nach London.
4. Ermitteln Sie die kürzeste Roundtrip-Zeit (nur Direktflüge) zwischen den Flughäfen FRA und JFK (Transferzeit am Flughafen JFK beachten).

Fortsetzung nächste Seite!

Teilaufgabe 2

1. Prozessmodelle

Das V-Modell 97 und das Spiralmodell sind Prozessmodelle für die Durchführung von Softwareprojekten.

1. Nennen und erläutern Sie für jedes der beiden Prozessmodelle drei wichtige Merkmale.

2. Geben Sie je zwei Beispiele für Projekteigenschaften, bei denen zum einen das V-Modell 97 und zum anderen das Spiralmodell Vorteile hat.

2. Fragen zur Softwaretechnik

1. Erläutern Sie, was sich hinter dem Prinzip DRY – Don't Repeat Yourself – verbirgt und nennen sowie erläutern Sie zwei Arten von Dopplung.

2. Erläutern Sie jeweils an einem konkreten Beispiel die Begriffe Fehlerwirkung, Fehlerzustand und Fehlerhandlung.

3. Nennen Sie drei konkrete Software-Metriken und diskutieren Sie die Aussagekraft einer dieser Metriken.

3. Systementwicklung

Sie wurden mit der Entwicklung eines Online-Shops zur Vermarktung elektronischer Artikel beauftragt. Um Waren kaufen zu können, müssen sich die Käufer am System registrieren. Das System soll einen virtuellen Warenkorb bieten, zu dem die Käufer die gewünschten Artikel hinzufügen und anschließend den Kauf durchführen können.

1. Identifizieren Sie vier Stakeholder und nennen Sie dazu je zwei unterschiedliche Anwendungsfälle (Funktionen) des Online-Shops, in die diese involviert sind.

2. Beschreiben Sie textuell die Schritte des Szenarios beim Kauf eines DVD-Recorders. Berücksichtigen Sie dabei die Nutzeridentifikation und -authentifizierung, das Aussuchen der Waren mittels eines Warenkorbs und die Bezahlung.

3. Modellieren Sie das Szenario von 2 in Form eines Sequenzdiagramms.

4. Modellieren Sie in Form eines Zustandsübergangsdiagramms die Zustände eines Warengegenstandes im Laufe des Bestell- und Liefervorgangs beginnend vom Zustand „angeboten“ bis hin zu seiner Auslieferung. Berücksichtigen Sie auch Reklamationen durch Rückgabe und die Stornierung einer noch nicht versandten Bestellung.

Fortsetzung nächste Seite!

4. Qualitätssicherung

Die folgende Seite enthält Software-Quellcode, der einen Algorithmus zur binären Suche implementiert. Dieser ist durch Inspektion zu überprüfen. Im Folgenden sind die Regeln der Inspektion angegeben.

RM1	(Dokumentation)	Jede Quellcode-Datei beginnt mit einem Kommentar, der den Klassennamen, Versionsinformationen, Datum und Urheberrechtsangaben enthält.
RM2	(Dokumentation)	Jede Methode wird kommentiert. Der Kommentar enthält eine vollständige Beschreibung der Signatur sowie eine Design-by-Contract-Spezifikation.
RM3	(Dokumentation)	Deklarationen von Variablen werden kommentiert.
RM4	(Dokumentation)	Jede Kontrollstruktur wird kommentiert.
RM5	(Formatierung)	Zwischen einem Schlüsselwort und einer Klammer steht ein Leerzeichen.
RM6	(Formatierung)	Zwischen binären Operatoren und den Operanden stehen Leerzeichen.
RM7	(Programmierung)	Variablen werden in der Anweisung initialisiert, in der sie auch deklariert werden.
RM8	(Bezeichner)	Klassennamen werden groß geschrieben, Variablennamen klein.

1. Überprüfen Sie durch Inspektion, ob die obigen Regeln für den Quellcode eingehalten wurden. Erstellen Sie eine Liste mit allen Verletzungen der Regeln. Geben Sie für jede Verletzung einer Regel die Zeilennummer, Regelnummer und Kommentar an, z.B. (07, RM4, while nicht kommentiert). Schreiben Sie nicht in den Quellcode.

2. Entspricht die Methode 'binarySearch' ihrer Spezifikation, die durch Vor- und Nachbedingungen angeben ist? Geben Sie gegebenenfalls Korrekturen der Methode an.

3. Beschreiben alle Kommentare ab Zeile 24 die Semantik des Codes korrekt? Geben Sie zu jedem falschen Kommentar einen korrigierten Kommentar mit Zeilennummer an.

4. Geben Sie den Kontrollflussgraphen für die Methode 'binarySearch' an.

5. Geben Sie maximal drei Testfälle für die Methode 'binarySearch' an, die insgesamt eine vollständige Anweisungsüberdeckung leisten.

Fortsetzung nächste Seite!

```
00 /*
01 * BinarySearch.java
02 *
03 * Eine Implementierung der "Binaere Suche"
04 * mit einem iterativen Algorithmus
05 */
06 class BinarySearch {
07
08  /**
09   * Binaere Suche
10   * a:       Eingabefeld
11   * item:    zu suchendes Element
12   * returnValue: der Index des zu suchenden Elements oder -1
13   *
14   * Vorbedingung:
15   * a.length > 0
16   * a ist ein linear geordnetes Feld:
17   * For all k: ( 1 <= k < a.length ) ==> ( a[k-1] <= a[k] )
18   *
19   * Nachbedingung:
20   * Wenn item in a, dann gibt es ein k mit a[k] == item und returnValue == k
21   * Genau dann wenn returnValue == -1 gibt es kein k mit 0 <= k < a.length
22   * und a[k] == item.
23   */
24   public static int binarySearch(float a[], float item) {
25
26     int End; // exklusiver Index fuer das Ende des
27              // zu durchsuchenden Teils des Arrays
28
29     int start = 1; // inklusiver Index fuer den Anfang der Suche
30     End = a.length;
31
32     // Die Schleife wird verlassen, wenn keine der beiden Haelften das
33     // Element enthaelt.
34     while(start < End) {
35
36       // Teilung des Arrays in zwei Haelften
37       // untere Haelfte: [0, mid[
38       // obere Haelfte:  ]mid, End[
39       int mid = (start + End) / 2;
40
41       if (item > a[mid]) {
42         // Ausschluss der oberen Haelfte
43         start = mid + 1;
44       } else if(item < a[mid]) {
45         // Ausschluss der unteren Haelfte
46         End = mid-1;
47       } else {
48         // Das gesuchte Element wird zurueckgegeben
49         return (mid);
50       }
51     } // end of while
52
53     // Bei Misserfolg der Suche wird -1 zurueckgegeben
54     return (-1);
55   }
56 }
```

Thema Nr. 2

Teilaufgabe 1

1. *ER*-Modellierung

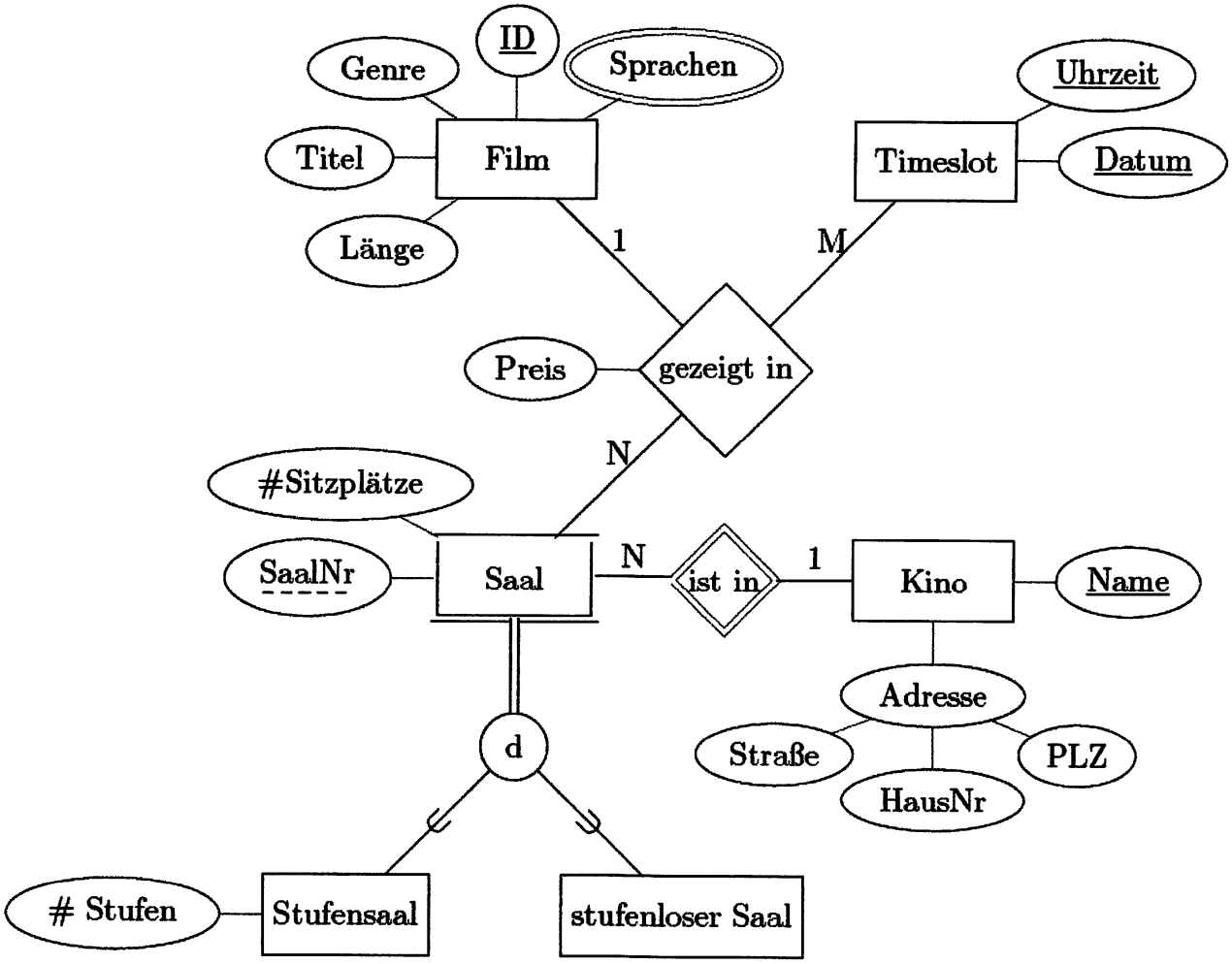
Für nachfolgend gegebene Miniwelt soll ein *ER*-Modell erstellt werden. Geben Sie Kardinalitäten in Chen-Notation an. Vergessen Sie nicht, dass zur Chen-Notation nicht nur Funktionalitäten, sondern auch Partizipitäten, also die Angabe von Existenzabhängigkeit bzw. totaler Teilnahme, zählen.

Das örtliche Tierheim möchte ein Datenbanksystem zur einfacheren Verwaltung seiner Tiere und Mitarbeiter anlegen. Hierfür speichert es zum einen alle Informationen über die beherbergten Tiere. Tiere lassen sich unterteilen in Hunde, Katzen, Vögel sowie Kleintiere. Im Tierheim bekommt jedes Tier einen Namen; dieser wird abgespeichert genau wie das Geschlecht und, soweit bekannt, das Geburtsdatum des jeweiligen Tieres. Außerdem soll festgehalten werden, ob das jeweilige Tier kastriert ist. Zu den Hunden wird ihre Rasse sowie ihre Schulterhöhe angegeben. Zu den Katzen wird ebenfalls die Rasse angegeben. Bei den Kleintieren wird angegeben, um was für ein Tier es sich handelt, z. B. Maus, Hamster, Kaninchen. Ab und zu kann es vorkommen, dass auch noch andere Tiere im Tierheim aufgenommen werden.
Jedes Tier bekommt eine bestimmte Zusammenstellung von mehreren Futtersorten. Jede Futtersorte hat einen Namen, einen Hersteller sowie eine Beschreibung. Für jedes Tier wird die Menge der jeweiligen Sorte festgehalten.
Die Mitarbeiter des Tierheims, von denen Name, Geburtsdatum und Anschrift, bestehend aus Straße, Hausnummer, Postleitzahl und Wohnort, gespeichert werden, lassen sich in feste Mitarbeiter mit Bezahlung sowie ehrenamtliche Mitarbeiter ohne Bezahlung unterteilen. Bei den festen Mitarbeitern wird die Höhe der Bezahlung in der Datenbank gespeichert. Für jeden ehrenamtlichen Mitarbeiter wird mindestens ein Tier symbolisch als „Patentier" eingetragen. Sind zu wenig ehrenamtliche Mitarbeiter verfügbar, kann es vorkommen, dass nicht jedes Tier einen Paten bekommt. Die festen Mitarbeiter kümmern sich aber natürlich um alle Tiere.

Fortsetzung nächste Seite!

2. Relationenmodell

Überführen Sie das folgende *ER*-Modell in ein Relationenschema, das die Bedingungen der 3. Normalform erfüllt. Geben Sie bei jedem Fremdschlüssel an, worauf er verweist. Überlegen Sie sich bei der Transformation von Generalisierung bzw. Spezialisierung, welche der zur Verfügung stehenden Möglichkeiten die beste ist. Beantworten Sie anschließend noch kurz die Frage, warum Sie sich für die von Ihnen gewählte Methode entschieden haben bzw. was gegen die anderen Möglichkeiten spricht.



Fortsetzung nächste Seite!

3. Normalisierung

Gegeben ist eine Relation R in 1. Normalform mit den Attributen {A, B, C, D, E, F, G, H}. In diesem Schema gelten die folgenden funktionalen Abhängigkeiten:

$AB \rightarrow D$

$ABC \rightarrow E$

$B \rightarrow C$

$FG \rightarrow HA$

$GA \rightarrow C$

3.1 Zweite Normalform

In einem ersten Schritt wurde diese Relation in zwei Relationen gespalten:

Relation1 (A, B, C, D, E)

Relation2 (F, G, H, A, C)

Überprüfen Sie für das gegebene Relationenschema aus den zwei Relationen, ob es in 2. Normalform ist. Begründen Sie sämtliche Ihrer Schlussfolgerungen und Entscheidungen.

Hinweis: Wie gehen Sie vor, wenn Sie eine Menge an Relationen gegeben haben? Was sagen die erfüllten Normalformen der einzelnen Relationen über die Normalform des Schemas aus?

3.2 Synthesealgorithmus

Wenden Sie den Synthesealgorithmus an, um das Schema in ein Schema in 3. Normalform zu überführen. Denken Sie daran, welche Voraussetzung dabei für die Menge der gegebenen funktionalen Abhängigkeiten gelten muss. Führen Sie auch die Schritte des Algorithmus kurz an, die keine Auswirkungen auf das Ergebnis haben, aber grundsätzlich zu beachten sind.

Fortsetzung nächste Seite!

4. SQL und Relationale Algebra

Gegeben ist folgendes Relationenschema für die Angebotsauswahl eines Wirtshauses:

Produkt(PNr, Name)
Getränk(PNr, Verkaufsvolumen, VPreis, EPreis, Alkoholgehalt)
FK(PNr) referenziert Produkt(PNr)
Speise(PNr, Beschreibung, VPreisP, EPreisP)
FK(PNr) referenziert Produkt (PNr)
Zusatzstoff(ZID, Name)
enthaelt(PNr,ZID)
FK(PNr) referenziert Produkt(PNr)
FK(ZID) referenziert Zusatzstoff(ZID)

VPreis und EPreis bezeichnen den Verkaufs- bzw. Einkaufspreis für 100 ml des jeweiligen Getränks. VPreisP und EPreisP bezeichnen den Verkaufs- bzw. Einkaufspreis für eine Portion der jeweiligen Speise.
Verwenden Sie für die folgenden Abfragen stets nur Standard-*SQL*-Befehle.

4.1
Wie viele verschiedene Speisen werden in dem Wirtshaus angeboten? Schreiben Sie eine *SQL*-Abfrage, welche das Ergebnis in einer Spalte mit dem Namen „Anzahl-Speisen" ausgibt.

4.2
Mit welchem Getränk bzw. welchen Getränken macht der Wirtshausbetreiber den größten Gewinn (auf den Preis bezogen)? Schreiben Sie eine *SQL*-Abfrage, welche die Nummer und den Namen des gefragten Getränks bzw. der gefragten Getränke ausgibt.

4.3
Welche angebotenen Produkte enthalten Zusatzstoffe mit dem Namensteil „phosphat"?
Schreiben Sie eine *SQL*-Abfrage, welche die Nummer und den Namen der Produkte ausgibt.

4.4
Welches sind die fünf teuersten Speisen im Angebot des Wirtshauses? Der *LIMIT*-Operator ist für diese Aufgabe nicht zulässig, da die Ausgabe sportlich fair sein soll. Sportlich fair bedeutet, dass falls beispielsweise ein Platz doppelt belegt ist, der nachfolgende Platz ausgelassen wird. Es können somit auch mehr als fünf Datensätze ausgegeben werden. Schreiben Sie eine *SQL*-Abfrage, welche die Nummer, den Namen, den Preis und die Platzierung aufsteigend, beim teuersten Gericht mit 1 beginnend, ausgibt.

4.5
Nennen Sie drei verschiedene Möglichkeiten, wie Löschregeln (*ON DELETE* ...) beim Anlegen von Tabellen definiert sein können. Wie lautet der jeweilige *SQL*-Befehl und welche Auswirkung hat er, wenn ein Eintrag einer referenzierten Tabelle gelöscht wird?

4.6
Formulieren Sie die Abfrage von Aufgabe 4.3 in relationaler Algebra.

Fortsetzung nächste Seite!

5. Grundlagen

5.1
Welche zwei Alternativen gibt es grundsätzlich, eine Datenbankanfrage zu optimieren? Nennen Sie diese beiden Arten der Optimierung und erklären Sie in ein bis zwei Sätzen, unter Zuhilfenahme welcher Methoden jeweils optimiert wird. Geben Sie für beide Alternativen je ein Beispiel an.

5.2
Für die Einhaltung der referentiellen Integrität gibt es in *SQL* für jeden der drei Begriffe Schlüsselkandidat, Primärschlüssel und Fremdschlüssel mindestens eine Beschreibungsmöglichkeit. Wie lautet diese jeweils bzw. falls es mehrere Möglichkeiten gibt, wie lauten diese?

5.3
Mit welchen zwei Möglichkeiten kann eine Transaktion abgeschlossen werden? Nennen Sie diese beiden Begriffe und die Auswirkungen auf die Datenbank.

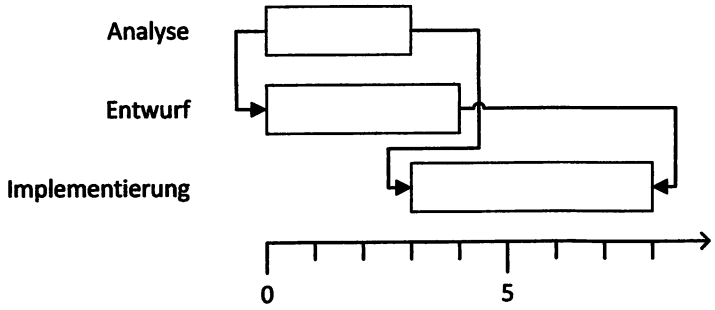
5.4
Erklären Sie anhand des *ANSI – SPRARC* – Schemas den Begriff der Datenunabhängigkeit in zwei bis drei Sätzen.

Fortsetzung nächste Seite!

Teilaufgabe 2

1. Projektmanagement

Gegeben ist das folgende **Gantt-Diagramm** zur Planung eines hypothetischen Softwareprojekts:



a) Konvertieren Sie das Gantt-Diagramm in ein **CPM-Netzwerk**, das die Aktivitäten und Abhängigkeiten äquivalent beschreibt. Gehen Sie von der Zeiteinheit „Monate" aus. Definieren Sie im CPM-Netzwerk je einen globalen Start- und Endknoten. Der Start jeder Aktivität hängt dabei vom Projektstart ab, das Projektende hängt vom Ende aller Aktivitäten ab.

b) Berechnen Sie für jedes Ereignis (d.h. für jeden Knoten Ihres CPM-Netzwerks) die **früheste Zeit**, die **späteste Zeit** sowie die **Pufferzeit**. Beachten Sie, dass die Berechnungsreihenfolge einer topologischen Sortierung des Netzwerks entsprechen sollte.

c) Geben Sie einen **kritischen Pfad** durch das CPM-Netzwerk an.
Welche **Aktivität** darf sich demnach wie lange verzögern?

Fortsetzung nächste Seite!

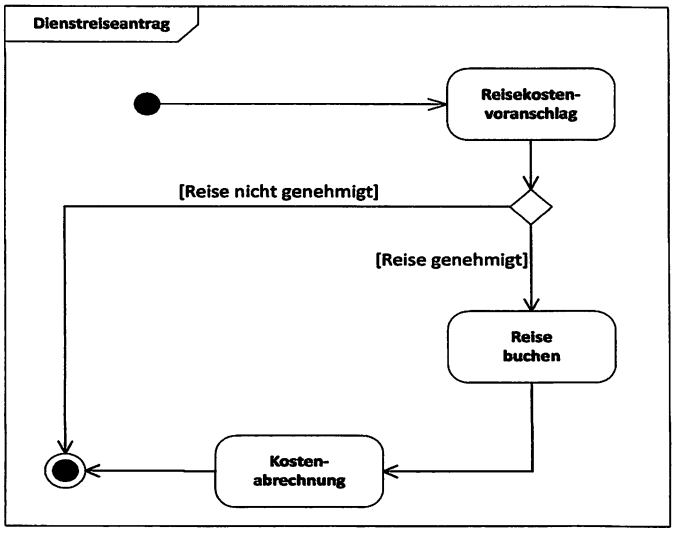
2. Objektorientierte Modellierung mit UML

Gegeben sei das folgende **Glossar**, welches die statische Struktur von einfachen **Aktivitätsdiagrammen** in natürlicher Sprache beschreibt:

> ***Aktivitätsdiagramm:*** *Benannter Container für Aktivitäten und Datenflüsse. Eine der definierten Aktivitäten ist als Start-Aktivität ausgezeichnet.*
>
> ***Aktivität:*** *Teil des beschriebenen Verhaltens. Man unterscheidet Start-, End-, echte Aktivitäten sowie Entscheidungen. Aktivitäten können generell mehrere ein- und auslaufende Kontrollflüsse haben.*
>
> ***Startaktivität****: Ist im Aktivitätsdiagramm eindeutig und dient als Einstiegspunkt des beschriebenen Ablaufs.*
>
> ***Endaktivität****: Wird eine solche Aktivität erreicht, ist der beschriebene Ablauf zu Ende.*
>
> ***Echte Aktivität****: Benannte Aktion, die nach Ausführung zu einer definierten nächsten Aktivität führt.*
>
> ***Entscheidung****: Aktivität, die mehrere Nachfolger hat. Welche davon als nächstes ausgeführt wird, wird durch entsprechende Bedingungen (s. Kontrollfluss) gesteuert.*
>
> ***Kontrollfluss:*** *Verbindet je eine Quell- mit einer Zielaktivität. Kann eine Bedingung enthalten, die erfüllt sein muss, damit die Zielaktivität im Falle einer Entscheidung ausgeführt wird.*

a) Geben Sie ein **UML-Klassendiagramm** an, welches die im Glossar definierten Konzepte und Beziehungen formal beschreibt. Geben Sie bei allen Attributen und Assoziationsenden deren Sichtbarkeit, Multiplizität und Typ an. Benennen Sie alle Assoziationen.

b) Nachfolgend ist ein Beispiel eines Aktivitätsdiagramms in der gängigen grafischen Notation abgebildet. Stellen Sie den beschriebenen Kontrollfluss als **UML-Objektdiagramm** *konform zum in Teilaufgabe a erstellten UML-Klassendiagramm* dar. Referenzieren Sie die dort definierten Klassen und Assoziationen; auf Objektbezeichner dürfen Sie verzichten.



Fortsetzung nächste Seite!

3. Entwurfsmuster

Verwenden Sie geeignete **Entwurfsmuster**, um die folgenden Sachverhalte mit Hilfe von **UML-Klassendiagrammen** zu beschreiben. Nennen Sie das zu verwendende Entwurfsmuster namentlich, wenden Sie es zur Lösung der jeweiligen *Fragestellung* an und erstellen Sie damit das problemspezifische UML-Klassendiagramm. Beschränken Sie sich dabei auf die statische Sicht, d.h. definieren Sie keinerlei Verhalten mit Ausnahme der Definition geeigneter Operationen.

a) Ein Unternehmen ist *baumartig* strukturiert. Abteilungen können wiederum selbst Unterabteilungen und/oder Personen beinhalten. Eine Person/Abteilung kann dabei nicht mehreren Abteilungen untergeordnet sein. Jede Abteilung wird außerdem von einer Person geleitet; umgekehrt können Personen beliebig viele Abteilungen leiten. Personen wie Abteilungen haben einen zahlenwertigen Identifikationsschlüssel. Bei Abteilungen wird eine Bezeichnung, bei Personen Name und Geburtsdatum hinterlegt.

b) Es gibt unterschiedliche Arten universitärer Veranstaltungen: Vorlesungen, Übungen und Seminare. Alle Veranstaltungen haben einen Namen und eine ID (Beispiel: „INF301"). Übungen beziehen sich immer auf eine Vorlesung. Seminare können wöchentlich, zweiwöchentlich oder im Block stattfinden. Definieren Sie eine Klasse „Hochschule", die die unterschiedlichen Veranstaltungen zusammenfasst und verwaltet. Es soll sichergestellt werden, dass es von dieser Klasse *nur eine einzige Instanz* gibt.

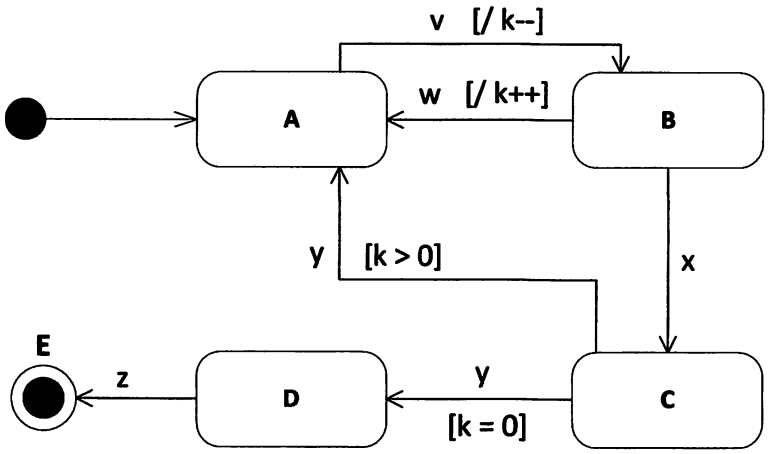
c) Eine Autowaschanlage besteht aus mehreren Reinigungsbürsten, vier Felgenreinigern und einem Trockengebläse. Alle Komponenten können unabhängig voneinander ein- und ausgeschaltet werden. Die Waschanlage bietet eine Operation an, die den Waschvorgang durchführt indem sie die einzelnen Komponenten in geeigneter Reihenfolge ein- und ausschaltet. Die Waschanlage bietet somit eine *vereinheitlichte Schnittstelle* für die Untersysteme (Bürsten, Felgenreiniger, ...) an.

Fortsetzung nächste Seite!

4. Zustandsbasiertes Testen

a) Geben Sie den allgemeinen Unterschied zwischen **Black-** und **White-Box-Tests** an. Zu welcher Kategorie gehört das **Zustandsbasierte Testen**?

Gegeben ist folgendes **UML-Zustandsdiagramm**. Die Variable *k* sei von ganzzahligem Typ.



b) Geben Sie **einen möglichst kurzen Testfall** an, der das Kriterium der **Zustandsüberdeckung** erfüllt.
Stellen Sie den Testfall als eine Sequenz von Operationsaufrufen dar. Notieren Sie bei jedem Zustandsübergang Vorzustand, Transition, Nachzustand, sowie den Wert der Variablen *k* vor und nach dem Operationsaufruf. Der Nachzustand bzw. Wert „k nachher" muss dem Vorzustand bzw. „k vorher" des nächsten Operationsaufrufs entsprechen. Den Initialwert von *k*, also das „k vorher" der ersten Zeile, sollen Sie dabei selbst bestimmen.

Beispiel:

Vorzustand	*k vorher*	*Transition*	*Nachzustand*	*k Nachher*
A	*3*	*v*	*B*	*2*
B	*2*	*w*	*A*	*3*

c) Geben Sie nun **einen möglichst kurzen Testfall** an, der das Kriterium der **Transitionsüberdeckung** erfüllt.
Verwenden Sie für den Testfall eine Darstellung mit denselben Eigenschaften wie in Teilaufgabe b.

Frühjahr 2018

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer
Kennzahl: _____		
Kennwort: _____	**Frühjahr**	**66116**
Arbeitsplatz-Nr.: _____	**2018**	

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: **21**

Bitte wenden!

Thema Nr. 1

Teilaufgabe 1

1. Grundlagen

Beantworten Sie die folgenden Fragen und begründen oder erläutern Sie Ihre Antwort in jeweils ein bis zwei Sätzen.

a) Was versteht man unter Selektivität?

b) Worin unterscheidet sich SQL von Programmiersprachen wie Java, C, C++, . . . und welche Vorteile ergeben sich daraus?

c) Wieso ist es wichtig, dass eine Transaktion atomar ausgeführt wird?

d) Welches Problem könnte auftreten, wenn Transaktionen nicht isoliert ablaufen würden?

e) Ist es richtig, dass die Ausführung von Anfragen durch die Verwendung von Sichten beschleunigt wird, weil dadurch weniger Daten von der Festplatte gelesen werden müssen?

f) Kann auf jeder Sicht eine SQL-Insert-Anweisung ausgeführt werden?

g) Kann die Performance von Anfragen durch Redundanzen im Datenbestand verbessert werden?

2. ER-Modellierung

Im Folgenden finden Sie die Beschreibung der Verwaltung eines Verkehrsunternehmens. Erstellen Sie zu dieser Beschreibung ein erweitertes ER-Diagramm. Kennzeichnen Sie die Primärschlüssel durch passendes Unterstreichen und geben Sie die Kardinalitäten in Chen-Notation (= Funktionalitäten) an. Kennzeichnen Sie auch die totale Teilnahme (= Existenzabhängigkeit, Partizipität) von Entitytypen.

Das Unternehmen betreibt Fahrzeuge. Diese sind durch eine Fahrgestellnummer eindeutig identifizierbar. Weiterhin haben Sie eine maximale Fahrgastzahl. Jedes Fahrzeug ist entweder ein Bus, eine Straßenbahn oder eine U-Bahn. Nur Busse haben eine Treibstoffart.

Weiterhin werden Mitarbeiter verwaltet. Diese verfügen über eine eindeutige Personalnummer, einen Nachnamen und einen Vornamen.

Zu Fahrzeugen werden Inspektionen verwaltet. Eine Inspektion hat ein für das jeweilige Fahrzeug eindeutiges Inspektionsdatum und wird von ein oder mehreren Mitarbeitern durchgeführt.

Das Verkehrsunternehmen bedient verschiedene Haltestellen, die alle einen eindeutigen Namen haben. Außerdem wird zu ihnen gespeichert, ob sie ein Wartehäuschen besitzen.

Fortsetzung nächste Seite!

Es gibt verschiedene Linien. Eine Linie hat eine eindeutige Nummer und beinhaltet ein oder mehrere Haltestellen. Es werden auch die Position einer Haltestelle auf einer Linie gespeichert sowie die Fahrzeit von der vorhergehenden Haltestelle auf der Linie. Pro Linie wird jede Haltestelle maximal einmal angefahren. Jede Linie wird entweder von genau einem Bus, einer Straßenbahn oder einer U-Bahn bedient.

Mitarbeiter können in einer Schicht mit einem bestimmten Fahrzeug auf einer bestimmten Linie fahren. Eine Schicht ist eindeutig gekennzeichnet durch ihren Startzeitpunkt und hat außerdem einen Endzeitpunkt. In jeder Schicht führt genau ein Mitarbeiter die Verkehrsaufsicht.

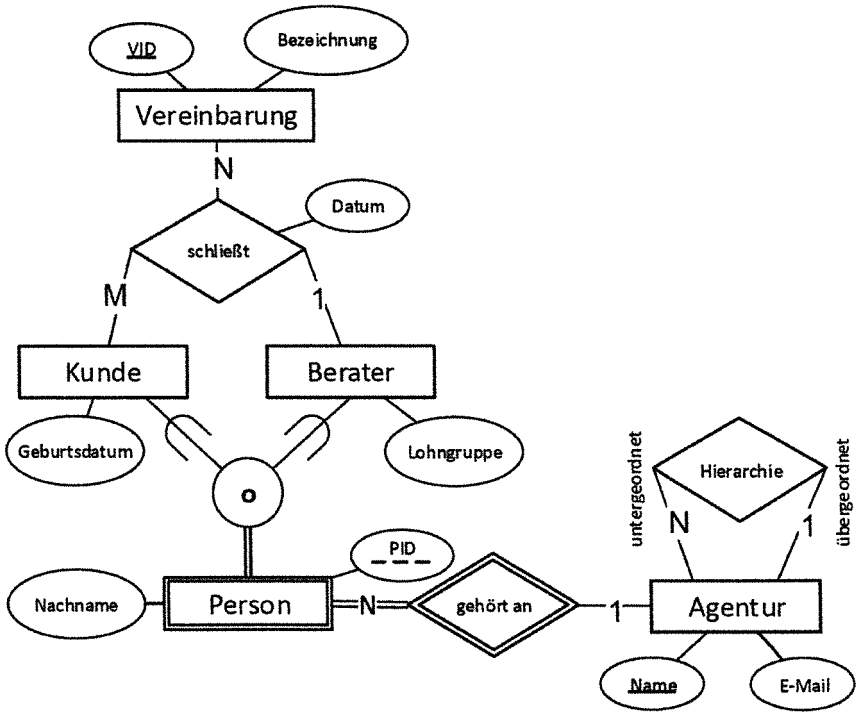
3. Relationaler Entwurf

Entwerfen Sie zum untenstehenden ER-Diagramm ein Relationenschema in dritter Normalform (3 NF) mit möglichst wenig Relationen.

Verwenden Sie dabei folgende Notation: Primärschlüssel werden durch Unterstreichen gekennzeichnet, 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,
Fremdschlüsselattribut1[Relation1],
(Fremdschlüssel2_Attribut1, Fremdschlüssel2_Attribut2)[Relation2])
```



Fortsetzung nächste Seite!

4. SQL

Gegeben sind folgende Relationen aus einer Bibliotheksverwaltung:

```
Buch(ISBN, Titel, ID[Autor], Erscheinungsjahr)
Autor(ID, Nachname, Vorname, Geburtsdatum)
Referenz(Referenzierendes_Buch[Buch], Referenziertes_Buch[Buch])
```

Verwenden Sie im Folgenden nur Standard-SQL und keine produktspezifischen Erweiterungen. Sie dürfen bei Bedarf Views anlegen. Geben Sie einen Datensatz nicht mehrfach aus.

a) Schreiben Sie eine SQL-Anweisung, die die Tabelle Buch (inklusive Integritätsbedingungen) anlegt. Beim Löschen eines Autors sollen auch all dessen Bücher gelöscht werden. Verwenden Sie sinnvolle Datentypen.

b) Schreiben Sie einen Trigger, der beim Einfügen neuer Autoren den Nachnamen auf die Zeichenkette „NN“ setzt, falls dieser NULL ist.

c) Schreiben Sie eine SQL-Anfrage, die die Titel aller Bücher von Autoren listet, die vor dem 1.1.1990 geboren wurden.

d) Schreiben Sie eine SQL-Anfrage, die zu einem Buch die Anzahl der Bücher auflistet, in denen es referenziert wird. Geben Sie nur Bücher aus, die in weniger als fünf Büchern referenziert wurden. Ausgegeben werden sollen der Titel des Buchs und die Anzahl der Referenzen, absteigend sortiert nach der Anzahl an Referenzen. Achten Sie darauf, dass auch Bücher, die nicht referenziert werden, ausgegeben werden.

e) Ermitteln Sie alle Autoren, die sich hauptsächlich selbst referenzieren. Schreiben Sie dazu eine SQL-Anfrage, die ID und Nachname aller Autoren liefert, in deren Büchern die Referenzen im Durchschnitt zu mindestens 90% auf eigene Bücher des Autors verweisen.

f) Schreiben Sie eine SQL-Anweisung, die alle Bücher löscht, die weder andere Bücher referenzieren noch selbst referenziert werden.

g) Nennen Sie die beiden SQL-Befehle zur Vergabe und zum Entzug von Rechten.

5. Relationale Algebra

Geben Sie an ob der folgende Zusammenhang in der relationalen Algebra gilt und beweisen Sie Ihre Aussage.

$$\pi_A(R - S) = \pi_A(R) - \pi_A(S)$$

mit:
R, S: Relationen mit gleichen Attributnamen und Attributtypen (= Domänen)
A: Teilmenge der Menge der Attribute von R oder S

Umgangssprachlich: Dürfen Projektion und Mengendifferenz vertauscht werden?

Fortsetzung nächste Seite!

6. Normalformen

a) Erläutern Sie in zwei bis drei Sätzen die beiden grundlegenden Eigenschaften, die bei der Aufspaltung eines Relationenschemas gelten müssen.

b) Gegeben ist die Relation Verbräuche (Gebäudenummer, Straße, Hausnummer, PLZ, Jahr, Wasserverbrauch, Stromverbrauch) mit den beiden Schlüsselkandidaten (Gebäudenummer, Jahr) und (Straße, Hausnummer, PLZ, Jahr). Die Attribute Wasserverbrauch und Stromverbrauch enthalten die entsprechenden Verbrauchswerte für ein bestimmtes Gebäude in einem bestimmten Jahr. Alle Attributwerte sind atomar. Geben Sie die höchste Normalform an, die die Relation Verbräuche erfüllt. Zeigen Sie, dass alle Bedingungen für diese Normalform erfüllt sind und dass mindestens eine Bedingung der nächsthöheren Normalform verletzt ist. Beziehen Sie sich bei der Begründung auf die gegebene Relation und nennen Sie nicht nur die allgemeinen Definitionen der Normalformen.

7. Optimierung

a) Erläutern Sie den Unterschied zwischen logischer (= algebraischer) und physischer Optimierung in drei bis vier Sätzen.

b) Übertragen Sie folgendes SQL-Statement in einen *nicht optimierten* algebraischen Ausdruck (= Anfragegraph, Operatorgraph).

```
select Kund.Vname, Kund.Nname
from Kund, Rechng
where Kund.ID = Rechng.Kund
and Rechng.Sum > 500;
```

c) Nennen Sie zwei Möglichkeiten, den algebraischen Ausdruck aus der vorhergehenden Teilaufgabe logisch (= algebraisch) zu optimieren. Beziehen Sie sich auf konkrete Stellen und Operatoren des von Ihnen aufgestellten algebraischen Ausdrucks.

Fortsetzung nächste Seite!

Teilaufgabe 2

Aufgabe 1: „Formale Spezifikation Abstrakter Datentypen (ADT)"

Hinweis: Sie dürfen in dieser Aufgabe alternativ auch andere *formale* Schreibweisen wie zum Beispiel UML/OCL oder die Z-Notation verwenden.

WICHTIG: In dieser Aufgabe stehen **keine** anderen Datentypen (also **kein int**, **String** usw.), **keine** anderen Konstanten (auch **nicht** 0, 1 usw.) und **keine** anderen Operationen (also **kein** <, +, ==, ≠ usw.) zur Verfügung: Sie dürfen ausschließlich die vorgegebenen oder zu ergänzenden **adt**s mit deren **ops** verwenden.

Gegeben seien die folgenden Gerüste der ADTs *Nat* (Repräsentation natürlicher Zahlen $\mathbb{N}_0^+$) und *NatStack* (Stapel mit *Nat*-Elementen):

adt *Nat*
sorts *Nat*
ops

NaN:		→ *Nat*	// „Not a Nat" – vergleichbar mit Double.NaN
zero:		→ *Nat*	// entspricht der natürlichen Zahl „0"
succ:	*Nat*	→ *Nat*	// Nachfolger „(x + 1)" der Nat-Zahl x
add:	*Nat* × *Nat*	→ *Nat*	// Addition im Nat-Raum
sub:	*Nat* × *Nat*	→ *Nat*	// Subtraktion im Nat-Raum (z.B. „42 − 666 = 0")
mul:	*Nat* × *Nat*	→ *Nat*	// Multiplikation im Nat-Raum

axs

succ(NaN) = NaN // alle Operationen mit NaN ergeben stets wieder NaN
// weitere Axiome aus Platzgründen weggelassen

end *Nat*

---

adt *NatStack*
sorts *NatStack, Nat*
ops

empty:		→ *NatStack*	// erzeugt einen neuen leeren Stapel
push:	*NatStack* × *Nat*	→ *NatStack*	// legt ein Nat ***auf*** den Stapel
peek:	*NatStack*	→ *Nat*	// gibt bei leerem Stapel NaN und bei
			// nicht-leerem Stapel das „***oberste***" Nat zurück
pop:	*NatStack*	→ *NatStack*	// gibt bei leerem Stapel empty und bei nicht-leerem
			// Stapel den Stapel ohne das „***oberste***" Nat zurück

axs

// Axiome aus Platzgründen weggelassen

end *NatStack*

a) Ergänzen Sie den ADT *Nat* um die Axiome des Vergleichsoperators *eq*, der genau dann *succ(zero)* zurück gibt, wenn die Operanden die gleiche *Nat*-Zahl darstellen – sonst *zero*, d.h. der Rückgabewert ist boolesch codiert: false ≡ 0 ≡ *zero* bzw. true ≡ 1 ≡ *succ(zero)*. Beachten Sie, dass alle Operationen *NaN* zurückgeben müssen, falls mind. einer der Operanden *NaN* ist. *Nicht vergessen:* Sie dürfen **keine** Vergleichsoperatoren wie ==, ≠ usw. verwenden – auch **nicht** auf der „rechten Seite" der Axiome (in „falls/sonst"-Bedingungen).

ops

eq: *Nat* × *Nat* → *Nat* // *succ(zero), falls die Operanden gleich sind; zero sonst*

Fortsetzung nächste Seite!

b) Ergänzen Sie den ADT *NatStack* um die Axiome der Operation *stackMul* gemäß Kommentar. Beachten Sie auch hier alle bisher genannten Einschränkungen.

ops
stackMul: NatStack → *Nat* // *multipliziert alle auf dem Stapel liegenden Nat-Zahlen*

c) Ergänzen Sie den ADT *NatStack* um die Axiome der Operation *stackPairOp* gemäß Kommentar bzw. Beispiel. Die Zahlen im Stack werden paarweise von „oben nach unten" betrachtet und genau dann durch ihr Produkt ersetzt, wenn sie gleich sind (→ Quadratzahl) – andernfalls durch ihre Summe. Sie dürfen annehmen, dass die Anzahl der Elemente im Stack beim Aufruf der Operation geradzahlig ist. Alle Einschränkungen gelten weiterhin.

ops
stackPairOp: NatStack → *NatStack* // *multipliziert gleiche bzw. addiert ungleiche*
// *benachbarte Zahlen auf dem Stack und*
// *ersetzt sie mit dem Ergebnis der Operation*

$$Beispiel: \begin{Bmatrix} \oplus & 47_{(Nat)} \\ \oplus & 11_{(Nat)} \\ \otimes & 11_{(Nat)} \\ \otimes & 11_{(Nat)} \\ \oplus & 11_{(Nat)} \\ \oplus & 655_{(Nat)} \end{Bmatrix} \xrightarrow{stackPairOp} \begin{Bmatrix} \oplus & 58_{(Nat)} \\ \otimes & 121_{(Nat)} \\ \oplus & 666_{(Nat)} \end{Bmatrix}$$

Zur Vereinfachung wurden die *Nat*-Zahlen hier mit Ziffern ausgedrückt,
d.h. $3_{(Nat)}$ entspräche *succ(succ(succ(zero)))*

d) Ergänzen Sie den ADT *NatStack* um die Axiome der folgenden Operationen, so dass der *NatStack* wie eine *Deque* von beiden Seiten bearbeitet werden kann. Führen Sie dazu die Operationen *put/get/poll* rekursiv und <u>ausschließlich</u> auf *empty/push/peek/pop* zurück:

ops
put: Nat × NatStack → *NatStack* // *fügt ein Nat ganz* ***unten*** *in den Stapel ein*
get: NatStack → *Nat* // *gibt bei leerem Stapel NaN und bei*
// *nicht-leerem Stapel das „**unterste**" Nat zurück*
poll: NatStack → *NatStack* // *gibt bei leerem Stapel empty und bei nicht-leerem*
// *Stapel den Stapel ohne das „**unterste**" Nat zurück*

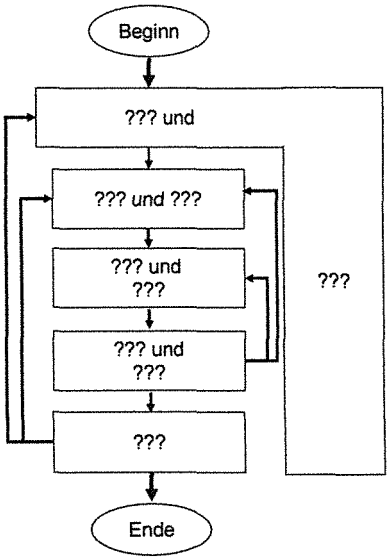
Fortsetzung nächste Seite!

Aufgabe 2: „Prozesse und Rollen"

a) Die Zertifizierungsstelle für Softwaretester ISTQB (International Software Testing Qualifications Board) definiert den Prozess des formalen Reviews (Inspektion) mit sechs Hauptphasen, während der Standard IEEE-1028 etwas genauer unterteilt und acht (auf den ISTQB-Prozess leicht abbildbare) Schritte vorsieht. Nennen und beschreiben Sie kurz jede einzelne Phase bei einem der beiden Standards.

b) Der Standard IEEE-1028-2008 definiert für ein Technisches Review vier Hauptrollen. Nennen Sie den jeweiligen Personenkreis zusammen mit deren jeweils wichtigsten Aufgaben.

c) Das ISTQB definiert den sogenannten „fundamentalen Testprozess" und stellt seinen Ablauf wie folgt dar:



Übernehmen Sie die Skizze, ergänzen Sie die fehlenden Hauptaktivitäten („???") und beschreiben Sie jede dieser Aktivitäten kurz.

Fortsetzung nächste Seite!

Aufgabe 3: „Entscheidungstabellentest"

Zu den funktionalen Testverfahren gehört der sog. Entscheidungstabellentest. Dabei werden die Informationen aus der Spezifikation systematisch in einer Tabelle erfasst, woraus sich die einzelnen Testfälle direkt ablesen lassen. Die Tabelle hat dabei folgende Grundstruktur:

Bedingungen	Regeln
Aktionen	Aktionsanzeiger

a) Skizzieren Sie kurz die vier Quadranten der Entscheidungstabelle.

b) Extrahieren Sie aus folgender Spezifikation die zugehörige Entscheidungstabelle für den Test.

„Das Warenwirtschaftssystem BitShop ist für kleine Online-Shops mit genau einem Produkt gedacht. Es verwaltet automatisch den aktuellen Lagerbestand des Artikels, bietet eine Web-Oberfläche für Bestellungen durch Kunden des Shops und generiert bei Bedarf Berichte für den Betreiber oder Bestellungen an Lieferanten. Dabei gelten folgende Geschäftsregeln:

- *Die über die Web-Oberfläche eingegebene Bestellmenge muss > 0 sein, andernfalls wird eine Fehlermeldung ausgegeben und die Bestellung storniert.*
- *Es sind keine Teillieferungen zugelassen. Falls der Lagerbestand für die aktuelle Bestellung zu niedrig ist, wird die Bestellung ebenfalls zurückgewiesen.*
- *Sinkt der Lagerbestand bei einer Bestellung unter ein vorgegebenes Minimum, dann wird automatisch eine Nachbestellung an den Lieferanten erstellt."*

c) Geben Sie exemplarisch vier logische Testfälle an, die Sie aus Ihrer Tabelle abgeleitet haben.

Thema Nr. 2

Teilaufgabe 1

Aufgabe 1: Allgemeine Fragen

a) Nennen Sie die 4 Bestandteile des ACID-Prinzips und erläutern Sie diese kurz.

b) Was versteht man unter einer Relation? (Es genügt eine Definition als Antwort.)

c) Gegeben seien zwei Relationen *R(A,B,C)* und *S(C,D,E)*. Die Relation *R* enthalte 20 Tupel und die Relation S enthalte 10 Tupel. Sei p ein beliebiges Selektionsprädikat.
Gegeben seien außerdem folgende Anfragen:

Q_1:

```
SELECT *
FROM R,S
WHERE p;
```

Q_2:

```
SELECT DISTINCT A
FROM R,S;
```

- Wieviele Ergebnistupel liefert die Anfrage Q_1 mindestens?
- Wieviele Ergebnistupel liefert die Anfrage Q_1 höchstens?
- Wieviele Ergebnistupel liefert die Anfrage Q_2 mindestens?
- Wieviele Ergebnistupel liefert die Anfrage Q_2 höchstens?

d) Nennen Sie drei mögliche Anomalien, die bei einem unkontrollierten Mehrbenutzerbetrieb auftreten können.

e) Welche Anomalie tritt bzgl. welcher Variable in dem folgenden Schedule der Transaktionen T_1 und T_2 auf?

$$S_1 = (r_1(x), r_2(y), w_2(x), r_1(z), r_1(x), w_2(y), w_1(z))$$

f) Gegeben seien die Transaktionen T_1, T_2 und T_3. Wie viele mögliche verschiedene **serielle** Schedules gibt es?

g) Was unterscheidet eine materialisierte View von einer gewöhnlichen View?

h) Geben Sie ein Beispiel für eine transitive Abhängigkeit.

i) Geben Sie ein Beispiel für eine reflexive Abhängigkeit.

Fortsetzung nächste Seite!

Aufgabe 2: Relationales Modell

Das Fremdenverkehrsamt will sich einen besseren Überblick über Zirkusse verschaffen. In einer Datenbank sollen dazu die Zirkusse, die angebotenen Vorstellungen, die einzelnen Darbietungen in einer Vorstellung sowie die zugehörigen Dompteure und Tiere verwaltet werden.

Ein Zirkus wird eindeutig durch seinen Namen gekennzeichnet und hat einen Besitzer. Vorstellungen haben eine eindeutige VorstellungsID und ein Datum. Darbietungen haben neben der eindeutigen ProgrammNr eine Uhrzeit. Ein Dompteur hat eine eindeutige AngestelltenNr sowie einen Künstlernamen. Tiere sind eindeutig durch eine TierNr bestimmt und haben außerdem eine Bezeichnung der Tierart.

Vorstellungen werden von genau einem Zirkus angeboten. Ein Zirkus bietet mehrere Vorstellungen an und stellt mehrere Dompteure an. Ein Dompteur ist genau bei einem bestimmten Zirkus angestellt. Eine Darbietung findet in einer bestimmten Vorstellung statt, während sich eine Vorstellung aus mehreren Darbietungen zusammensetzt. Des Weiteren trainiert ein Dompteur mehrere Tiere, ein Tier kann allerdings auch von mehreren Dompteuren trainiert werden. In einer Darbietung tritt genau ein Dompteur mit mindestens einem Tier auf. Ein Dompteur kann in mehreren Darbietungen auftreten.

Erstellen Sie ein Entity-Relationship-Diagramm für obige Datenbank.

Aufgabe 3: Relationale Algebra

Gegeben seien folgende Relationen:

X

A	B	C	D
1	4	6	3
6	2	5	1
4	5	6	3
1	4	5	2
1	4	6	2
3	3	5	2
4	3	1	2
4	5	6	2

Y

C	D	E
5	2	3
6	2	1
6	3	1

Z

D	F
6	2
5	6
1	3

Fortsetzung nächste Seite!

Geben Sie die Ergebnisrelationen folgender Ausdrücke der relationalen Algebra als Tabellen an.

$$\sigma_{B>3}(X) \underset{C=F}{\bowtie} \sigma_{F>4}(\pi_F(Z))$$

$$\pi_{C,D}(X) - (\pi_C(Y) \times \pi_D(Z))$$

$$\pi_{A,C}(X) \div \pi_C(Y)$$

Aufgabe 4: Tupelkalkül

Gegeben sei das folgende Datenbank-Schema, das für die Speicherung der Daten einer Universität entworfen wurde, zusammen mit einem Teil seiner Ausprägung. Die Primärschlüssel-Attribute sind jeweils unterstrichen.
Die Relation *Dozent* enthält allgemeine Daten zu den Dozentinnen und Dozenten.
Dozentinnen und Dozenten halten Vorlesungen, die in der Relation *Vorlesung* abgespeichert sind.
Wir gehen davon aus, dass es zu jeder Vorlesung genau einen Dozenten (und nicht mehrere) gibt.
Zusätzlich wird in der Relation *Vorlesung* das Datum gespeichert, an dem die Klausur stattfindet.
In der Relation Student werden die Daten der teilnehmenden Studierenden verwaltet,
während die Relation *besucht* Auskunft darüber gibt, welche Vorlesung von welchen Studierenden besucht wird.

Dozent

DNR	DVorname	DNachname	DTitel
1	Christina	Müller	Prof. Dr.
2	Sabine	Schulz	Prof. Dr.
3	Herbert	Schmidt	PD Dr.
4	Adelheit	Berger	PD Dr.
...	...	...	...

Vorlesung

VNR	VTitel	Klausurtermin	Dozent
1	Datenbanksysteme I	29.01.2018	3
2	Einführung in die Programmierung	30.01.2018	2
3	Index- und Speicherstrukturen	29.01.2018	1
4	Knowledge Discovery in Databases I	05.02.2018	4
5	Softwareentwicklungspraktikum	22.02.2018	3
6	Datenbankpraktikum	11.02.2018	4
...	...	...	...

Student

Matrikelnummer	SVorname	SNachname	Semesterzahl
10000000	Frederik	Frühling	3
10000001	Valerie	Winter	12
...	...	...	...

besucht

Student	Vorlesung
10000000	3
...	...

Fortsetzung nächste Seite!

Dozent (DNR, DVorname, DNachname, DTitel)
Vorlesung (VNR, VTitel, Klausurtermin, Dozent)
Student (Matrikelnummer, SVorname, SNachname, Semesterzahl)
besucht (Student, Vorlesung)

Formulieren Sie die folgenden Anfragen im Tupelkalkül.
Datumsvergleiche können Sie mit >, ≥, <, ≤ oder = angeben:

a) Geben Sie die Vornamen aller Studierenden aus, die die Vorlesung „Datenbanksysteme I" besuchen oder besucht haben.

b) Geben Sie die Matrikelnummern der Studierenden an, die keine Vorlesung mit einem Klausurtermin nach dem 31.12.2017 besuchen oder besucht haben.

c) Geben Sie die Matrikelnummern der Studierenden aus, die alle Vorlesungen von Prof. Dr. Schulz hören oder gehört haben.

Aufgabe 5: Anfragen

Gegeben sei das folgende Datenbankschema. Die Attribute Preis und Menge seien ganzzahlig, alle weiteren Attribute seien Zeichenketten. Im folgenden Schema sind Primärschlüssel unterstrichen und Fremdschlüssel überstrichen, wobei die von Fremdschlüsseln referenzierte Relation in eckigen Klammern nach dem Fremdschlüsselattribut angegeben ist.

`Lieferant(LNr, LName, LStadt)`

`Ware(WNr, Bezeichnung, Preis)`

`Kunde(KNr, KName, KStadt)`

`Auftrag(LNr[Lieferant], WNr[Ware], KNr[Kunde], Menge)`

a) Geben Sie die Anweisung in **SQL-DDL** an, die notwendig ist, um die Relation 'Auftrag' zu erzeugen. Achten Sie dabei auf Fremdschlüsselbeziehungen.

b) Formulieren Sie die folgende Anfrage in **SQL**:
Gesucht sind die Nummern aller Waren, die Lieferant Huber nach Hamburg liefert.

c) Formulieren Sie die folgende Anfrage in **SQL**: Gesucht ist eine Liste aller belieferten Städte mit ihrem jeweiligen gesamten Lieferungsvolumen.

Fortsetzung nächste Seite!

Aufgabe 6 Normalisierung

Gegeben sei das Relationenschema $R(A,B,C,D,E,F)$, sowie die Menge der zugehörigen funktionalen Abhängigkeiten F:

- $C \rightarrow B$
- $B \rightarrow A$
- $CE \rightarrow D$
- $E \rightarrow F$
- $CE \rightarrow F$
- $C \rightarrow A$

a) Bestimmen Sie den Schlüsselkandidaten der Relation R und begründen Sie, warum es keine weiteren Schlüsselkandidaten gibt.

b) Überführen Sie das Relationenschema R mit Hilfe des Synthesealgorithmus in die dritte Normalform. Führen Sie hierfür jeden der vier Schritte durch und kennzeichnen Sie Stellen, bei denen nichts zu tun ist.

Fortsetzung nächste Seite!

Teilaufgabe 2

Aufgabe 1: (Objektorientierte Programme und Reverse Engineering)

Gegeben sei das folgende Java-Programm:

```
class M {
   private boolean b;
   private F f;
   private A a;
   public void m() {
      f = new F();
      a = new A(f);
      b = true;
   }
}


class A {
   private R r;
   public A(I i) {
      r = i.createX();
   }
}


interface I {
   public X createX();
}


class F implements I {
   public X createX() {
      return new X(0,0);
   }
}


abstract class R {
   protected int v;
}


class X extends R {
   private int v, w;
   public X(int v, int w) {
      this.v = v;
      this.w = w;
   }
}
```

Fortsetzung nächste Seite!

a) Das Subtypprinzip der objektorientierten Programmierung wird in obigem Programmcode zweimal ausgenutzt. Erläutern Sie wo und wie dies geschieht.

b) Zeichnen Sie ein UML-Klassendiagramm, das die statische Struktur des obigen Programms modelliert. Instanzvariablen mit einem Klassentyp sollen durch gerichtete Assoziationen mit Rollennamen und Multiplizität am gerichteten Assoziationsende modelliert werden. Alle aus dem Programmcode ersichtlichen statischen Informationen (insbesondere Interfaces, abstrakte Klassen, Zugriffsrechte, benutzerdefinierte Konstruktoren und Methoden) sollen in dem Klassendiagramm abgebildet werden.

c) Es wird angenommen, dass ein Objekt der Klasse `M` existiert, für das die Methode `m()` aufgerufen wird. Geben Sie ein Instanzendiagramm (Objektdiagramm) an, das alle **nach** der Ausführung der Methode `m` existierenden Objekte und deren Verbindungen (Links) zeigt.

d) Wie in Teil c) werde angenommen, dass ein Objekt der Klasse `M` existiert, für das die Methode `m()` aufgerufen wird. Diese Situation wird in Abb. 1 dargestellt. Zeichnen Sie ein Sequenzdiagramm, das Abb. 1 so ergänzt, dass alle auf den Aufruf der Methode `m()` folgenden Objekterzeugungen und Interaktionen gemäß der im Programmcode angegebenen Konstruktor- und Methodenrümpfe dargestellt werden. Aktivierungsphasen von Objekten sind durch längliche Rechtecke deutlich zu machen.

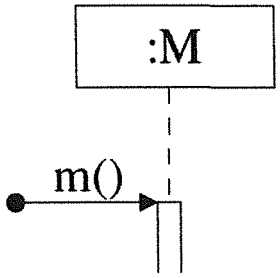


Abbildung 1: Beginn eines Sequenzdiagramms

Aufgabe 2: (Anwendung des Observer-Patterns)

Es soll eine (kleine) Anwendung entwickelt werden, in der ein Zähler in 1-er Schritten von 5000 bis 0 herunterzählt. Der Zähler soll als Objekt der Klasse `Countdown` realisiert werden, die in UML-Notation in Abb. 2 dargestellt ist. Das Attribut `value` soll den aktuellen Zählerstand speichern, der mit dem Konstruktor zu initialisieren ist. Die Methode `getValue` soll den aktuellen Zählerstand liefern und die Methode `countdown` soll den Zähler von 5000 bis 0 herunterzählen.

Der jeweilige Zählerstand soll von einem Objekt der in Abb. 2 angegebenen Klasse `Display` am Bildschirm ausgegeben werden. Bei der Konstruktion eines `Display`-Objekts soll es mit einem `Countdown`-Objekt verbunden werden, indem dessen Referenz unter `myCountdown` abgespeichert wird. Die Methode `update` soll den aktuellen Zählerstand vom `Countdown`- Objekt holen und mit `System.out.println` am Bildschirm ausgeben. Dies soll zu Beginn des Zählprozesses und nach jeder Änderung des Zählerstands erfolgen.

Fortsetzung nächste Seite!

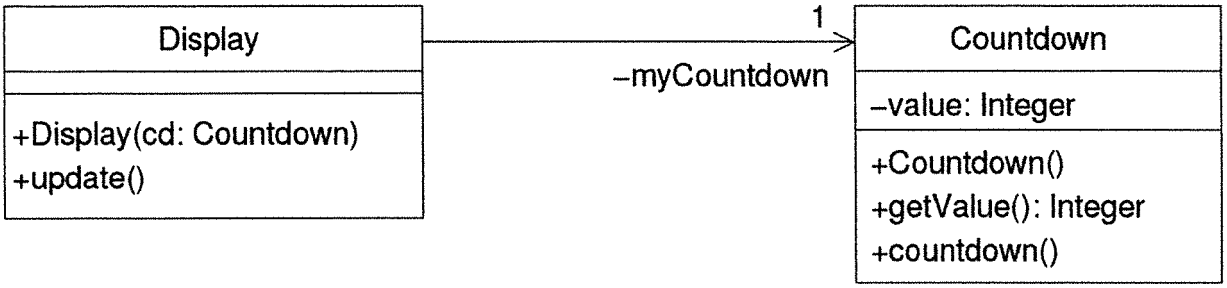


Abbildung 2: `Display` und `Countdown`

Damit das `Display`-Objekt über Zählerstände des `Countdown`-Objekts informiert wird, soll das Observer-Pattern angewendet werden. Abb. 3 zeigt die im Observer-Pattern vorkommenden abstrakten Klassen. (Kursivschreibweise bedeutet abstrakte Klasse bzw. abstrakte Methode.)

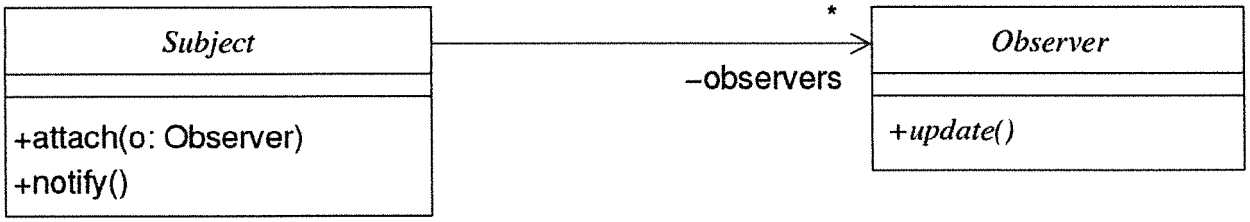


Abbildung 3: *Subject* und *Observer*

a) Welche Wirkung haben die Methoden `attach` und `notify` gemäß der Idee des Observer-Patterns?

b) Welche der beiden Klassen `Display` und `Countdown` aus Abb. 2 spielt die Rolle eines *Subject* und welche die Rolle eines *Observer*?

c) Erstellen Sie ein Klassendiagramm, das die in Abb. 2 und Abb. 3 gegebenen Diagramme in geeigneter Weise, d.h. entsprechend der Idee des Observer-Patterns, zusammenfügt. Es reicht die Klassen und deren Beziehungen anzugeben. Eine nochmalige Nennung der Attribute und Methoden ist nicht notwendig.

Fortsetzung nächste Seite!

d) Unsere Anwendung soll nun in einer objektorientierten Programmiersprache Ihrer Wahl (z.B. Java oder C++) implementiert werden. Dabei soll von folgenden Annahmen ausgegangen werden:

- Das Programm wird mit einer `main`-Methode gestartet, die folgenden Rumpf hat:

```
{Countdown cd = new Countdown();
new Display(cd);
cd.countdown();}
```

- Die beiden Klassen *Subject* und *Observer* sind bereits gemäß der Idee des Observer-Patterns implementiert.

Geben Sie auf dieser Grundlage eine Implementierung der beiden Klassen `Display` und `Countdown` an, so dass das gewünschte Verhalten, d.h. Anzeige der Zählerstände und Herunterzählen des Zählers, realisiert wird. Die Methoden der Klassen *Subject* und *Observer* sind dabei auf geeignete Weise zu verwenden bzw. zu implementieren. Geben Sie die verwendete Programmierprache an.

Aufgabe 3: (Entwicklung von Zustandsdiagrammen)

Ein Benutzer einer Online-Plattform für Flugreservierungen möchte für eine gegebene Flugnummer einen Flug buchen. Das Sequenzdiagramm in Abb. 4 spezifiziert, welche Interaktionen dazu in einem Hauptablauf durchgeführt werden. An der Interaktion sind Objekte der Klassen `Flugreservierung`, `Flug` und `Printer` beteiligt. Zunächst wird geprüft, ob für die gegebene Flugnummer noch Plätze frei sind. Ist dies der Fall, wird die Buchung getätigt und das Flugticket ausgedruckt. Das Sequenzdiagramm in Abb. 5 spezifiziert einen Ausnahmeablauf, in dem eine Fugbuchung mangels freier Plätze nicht möglich ist. Das Sequenzdiagramm in Abb. 6 spezifiziert einen zweiten Ausnahmeablauf, in dem zwar zunächst freie Plätze vorhanden sind, jedoch in der Zwischenzeit, bis der Benutzer die Buchung vornehmen will, der Flug ausgebucht ist.

Es ist **ein** Zustandsdiagramm für die Klasse `Flugreservierung` zu entwickeln. Das Zustandsdiagramm soll das mögliche Verhalten eines Objekts der Klasse Flugreservierung zeigen, indem es die Lebenslinien des `Flugreservierung`-Objekts aus allen drei Abläufen geeignet integriert. Auf folgende Punkte ist zu achten:

- Alle Abläufe sollen beliebig wiederholt werden können.

- Ereignisse sind von Aktionen zu unterscheiden und es muss klar sein, welche Aktionen als Reaktion auf welche Ereignisse durchgeführt werden.

- Aktionen sind hier immer Meldungen an den Benutzer oder Sendeaktionen oder (sequentiell) zusammengesetzte Aktionen. Das Senden einer Nachricht `n()` an ein Objekt `o:K` einer Klasse `K` ist im Zustandsdiagramm mit `o.n()` zu notieren. Die sequentielle Zusammensetzung zweier Aktionen `a1` und `a2` ist mit `a1;a2` zu notieren.

- Das Zustandsdiagramm hat einen Anfangszustand.

Fortsetzung nächste Seite!

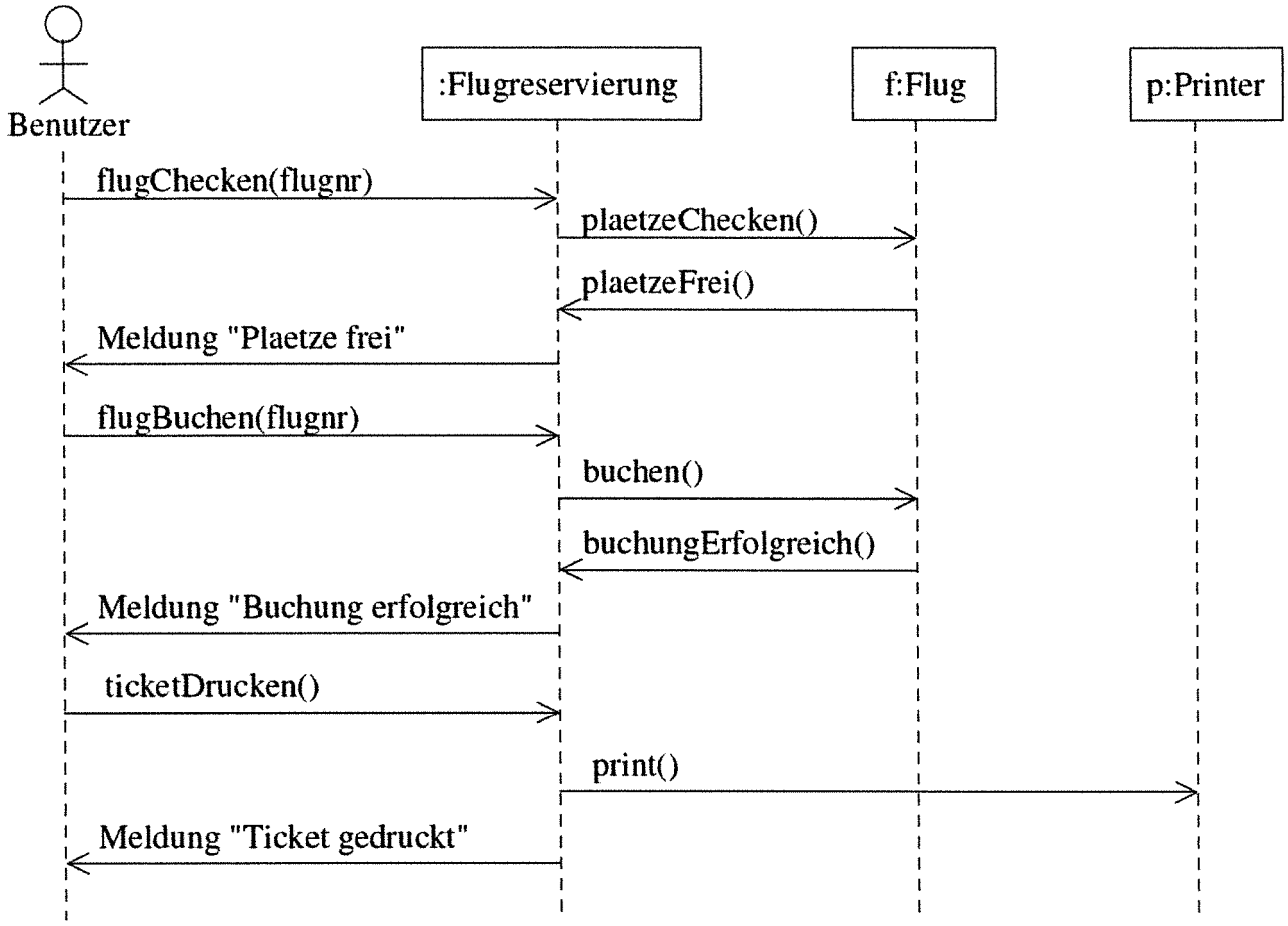


Abbildung 4: Hauptablauf

Fortsetzung nächste Seite!

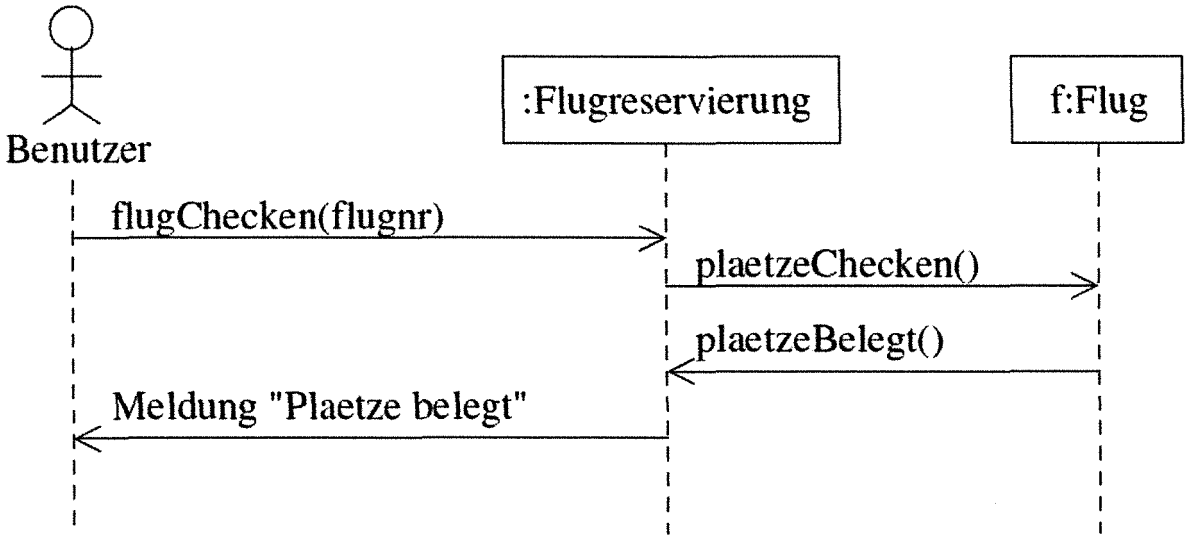


Abbildung 5: Ausnahmeablauf 1

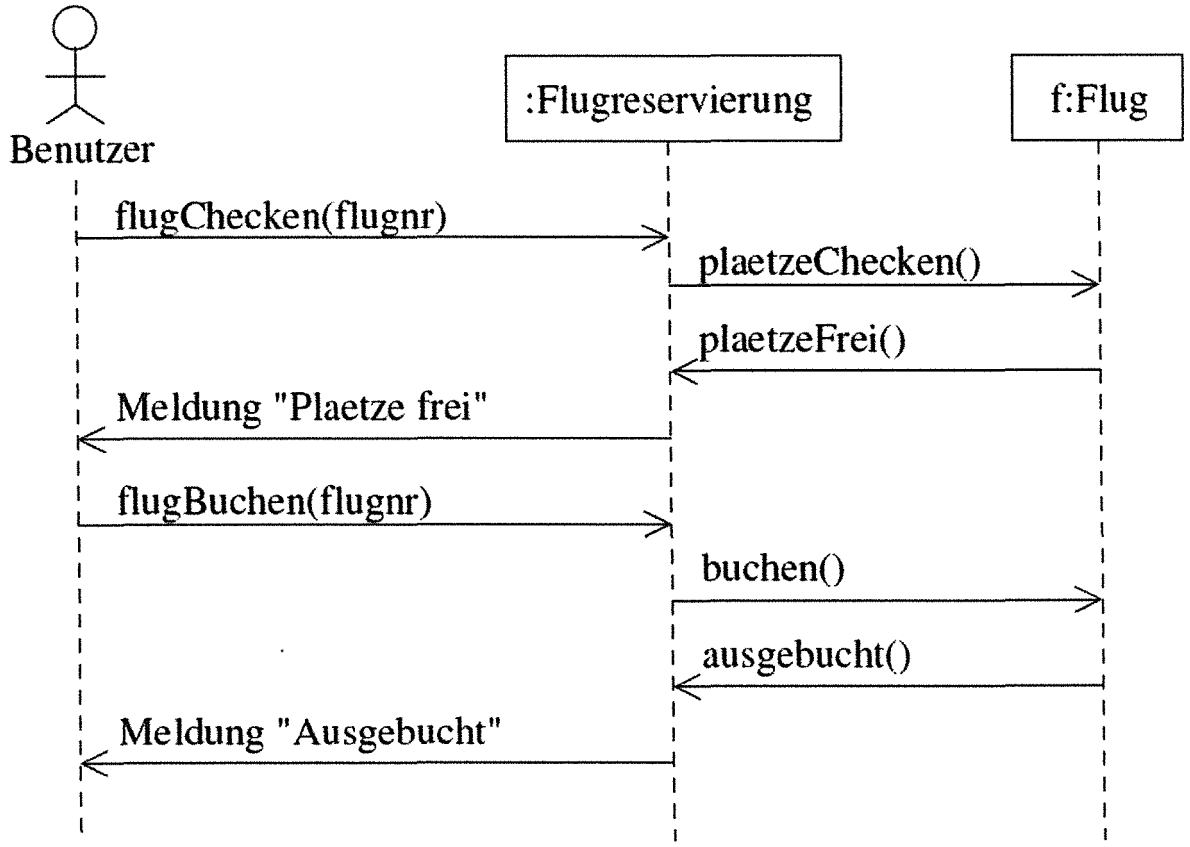


Abbildung 6: Ausnahmeablauf 2

Fortsetzung nächste Seite!

Aufgabe 4: (Verifikation funktionaler Programme)

Gegeben sei der folgende polymorphe Datentyp *'a boolexp* zur Darstellung Boolescher Ausdrücke über einer beliebigen Menge *'a* von Booleschen Variablen. Boolesche Ausdrücke werden hier gebildet aus Booleschen Variablen sowie aus Negation und Konjunktion.

datatype *'a boolexp* = *Var* **of** *'a* | *Not* **of** *'a boolexp* | *And* **of** *'a boolexp* * *'a boolexp*

Zum Beispiel wird der Boolesche Ausdruck : $\neg(x \wedge \neg y) \wedge \neg(\neg x \wedge y)$ dargestellt durch:

And(Not(And(x, Not(y))), Not(And(Not(x), y)))

Die Tiefe eines Ausdrucks wird durch die folgende Funktion *depth* (durch Pattern-Matching gemäß des strukturellen Aufbaus eines Ausdrucks) definiert. Sie berechnet die Anzahl der *Not*- und *And*-Konstruktoren auf dem längsten Ast des Ausdrucks.

```
fun   depth(Var(x))        = 0
  |   depth(Not(B))        = depth(B) + 1
  |   depth(And(B1,B2))    = max(depth(B1), depth(B2)) + 1
```

Die Breite eines Ausdrucks wird durch die folgende Funktion *width* definiert. Sie berechnet die Anzahl der in einem Ausdruck vorkommenden Variablen.

```
fun   width (Var(x))       = 1
  |   width (Not(B))       = width(B)
  |   width (And(B1,B2))   = width(B1) + width(B2)
```

Es wird behauptet, dass für alle Ausdrücke *B* des Typs *'a boolexp* Folgendes gilt:

$$width(B) \leq 2^{depth(B)}$$

Beweisen Sie diese Behauptung durch strukturelle Induktion gemäß des Aufbaus des Ausdrucks *B*. Geben Sie zunächst an, welche Fälle für die Gestalt von *B* unterschieden werden müssen und welcher Fall der Basisfall der Induktion ist. Für jeden Nicht-Basisfall ist im Beweis die Induktionsvoraussetzung zu nennen und genau anzugeben, wo diese verwendet wird.

Herbst 2018

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer
Kennzahl: _____		
Kennwort: _____	**Herbst**	**66116**
Arbeitsplatz-Nr.: _____	**2018**	

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: **20**

Bitte wenden!

Thema Nr. 1
(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

Teilaufgabe 1

Aufgabe 1: Objektdiagramme (14 Punkte)

Das folgende Klassendiagramm beschreibt Modellelemente der Modellierungssprache UML und Beziehungen zwischen diesen. Kursiv geschriebene Klassennamen bezeichnen abstrakte Klassen.

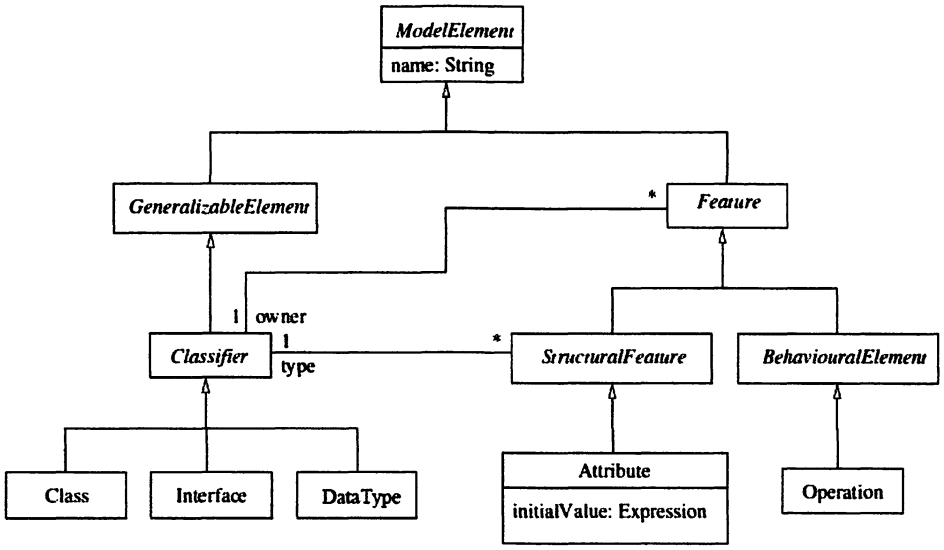
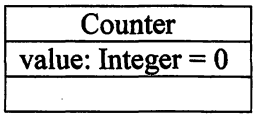


Abbildung 1: Modellelemente der UML

Ein Beispiel einer Klasse in der üblichen, grafischen UML-Notation ist die folgende Klasse **Counter**, die ein Attribut **value** vom Typ **Integer** und mit initialem Wert **0** hat.



Stellen Sie diese Klasse als UML-Objektdiagramm dar, das konform zum Klassendiagramm in Abbildung 1 gebildet ist. Für jedes Objekt ist - in der üblichen UML-Notation für Objekte - anzugeben, zu welcher Klasse es gehört, welche Werte die Attribute haben und mit welchen anderen Objekten es verbunden ist und dabei ggf. eine bestimmte Rolle spielt. Auf Objektbezeichner kann verzichtet werden.

Hinweis: **Integer** ist ein Datentyp, also als ein Objekt der Klasse **DataType** darzustellen, das den String **"Integer"** als Wert des von der abstrakten Klasse **ModelElement** geerbten Attributs **name** hat.

Fortsetzung nächste Seite!

Aufgabe 2: Klassendiagramme, STATE-Pattern, Zustandsdiagramme (12+12+19=43 Punkte)

Gegeben sei das Java-Programm in Listing 1.

a) Zeichnen Sie ein UML-Klassendiagramm, das die statische Struktur des Programms modelliert. Instanzvariablen mit einem Klassentyp sollen durch gerichtete Assoziationen mit Rollennamen und passender Multiplizität am gerichteten Assoziationsende modelliert werden. Alle aus dem Programmcode ersichtlichen statischen Informationen sollen in dem Klassendiagramm dargestellt werden.

Das Programm in Listing 1 implementiert ein Zustandsdiagramm, das das Verhalten von Objekten der Klasse `Music` beschreibt. Für die Implementierung wurde das Design-Pattern STATE angewendet.

b) Geben Sie die statische Struktur des STATE-Patterns an und erläutern Sie, welche Rollen aus dem Entwurfsmuster den Klassen des gegebenen Programms dabei zufallen und welche Operationen aus dem Entwurfsmuster durch (ggf. mehrere) Methoden in unserem Beispielprogramm implementiert werden. Es ist von den z. B. im Design-Pattern-Katalog von Gamma et al. verwendeten Namen auszugehen, das heißt von Klassen mit Namen `Context`, `State`, `ConcreteStateA`, `ConcreteStateB` und von Operationen mit Namen `request` und `handle`.

c) Zeichnen Sie das UML-Zustandsdiagramm (mit Anfangszustand), das von dem Programm in Listing 1 implementiert wird. Dabei muss - gemäß der UML-Notation - unterscheidbar sein, was Ereignisse und was Aktionen sind. In dem Diagramm kann zur Vereinfachung statt `System.out.println("x")` einfach `"x"` geschrieben werden.

Fortsetzung nächste Seite!

```
class Music {
  private Beatle state;
  public Music() {state = new Paul();}
  public void help() {this.state.help(this);}
  public void obladi() {this.state.obladi(this);}
  public void yesterday() {this.state.yesterday(this);}
  public void setBeatle(Beatle b) {state = b;}
}
abstract class Beatle {
  public abstract void help(Music m);
  public abstract void obladi(Music m);
  public abstract void yesterday(Music m);
}
class George extends Beatle {
  public void help(Music m) {
    System.out.println("help"); m.setBeatle(new John());}
  public void obladi(Music m) {}
  public void yesterday(Music m) {
    System.out.println("yesterday"); m.setBeatle(new Paul());}
}
class John extends Beatle {
  public void help(Music m) {
    System.out.println("help"); m.setBeatle(new Paul());}
  public void obladi(Music m) {
    System.out.println("obladi"); m.setBeatle(new Ringo());}
  public void yesterday(Music m) {}
}
class Paul extends Beatle {
  public void help(Music m) {
    System.out.println("help"); m.setBeatle(new George());}
  public void obladi(Music m) {}
  public void yesterday(Music m) {
    System.out.println("yesterday");}
}
class Ringo extends Beatle {
  public void help(Music m) {
    System.out.println("help"); m.setBeatle(new John());}
  public void obladi(Music m) {}
  public void yesterday(Music m) {
    System.out.println("yesterday"); m.setBeatle(new Paul());}
}
```

Listing 1

Fortsetzung nächste Seite!

Aufgabe 3: Systemarchitektur und Interaktionen (4+16+5=25 Punkte)

Das UML-Diagramm in Abbildung 2 zeigt eine Zwei-Schichten-Architektur eines Systems mit Benutzerschnittstelle und Anwendungskern.

a) Erläutern Sie kurz, inwiefern die gegebene Architektur die beiden wichtigen Prinzipien der hohen Kohäsion *(high cohesion)* und der losen Kopplung *(loose coupling)* erfüllt.

Im Folgenden wird ein typisches Interaktionsmuster für Objekte dieser Architektur angegeben. Zu Beginn der Kommunikation sollen existieren: Ein Benutzer als Aktor außerhalb des Systems, ein `MainWindow`-Objekt und ein `ApplicationClass`-Objekt.

1. Der Benutzer startet einen Anwendungsfall *(Use-Case)*, indem er an das `MainWindow`-Objekt die Nachricht `startUseCase` schickt.

2. Das `MainWindow`-Objekt erzeugt daraufhin ein `DialogControl`-Objekt zur Dialogkontrolle für den Use-Case.

3. Das `DialogControl`-Objekt erzeugt daraufhin ein `ApplicationControl`-Objekt sowie ein `UseCaseWindow`-Objekt (für Ein- und Ausgaben, die den Use-Case betreffen).

Damit ist das Starten des Anwendungsfalls abgeschlossen. Es folgt nun die Durchführung des Anwendungsfalls, bei der folgende Nachrichten ausgetauscht werden.

4. Der Benutzer übermittelt mit der Nachricht `enterInfoUseCase` spezifische Informationen an das `UseCaseWindow`-Objekt.

5. Die Informationen werden vom `UseCaseWindow`-Objekt mit der Nachricht `enterInfo` an das `DialogControl`-Objekt übergeben.

6. Der Übergang zum Anwendungskern erfolgt nun, indem das `DialogControl`-Objekt die Nachricht `compute` an das `ApplicationControl`-Objekt schickt.

7. Daraufhin schickt das `ApplicationControl`-Objekt die Nachricht `runApp` an das `ApplicationClass`-Objekt (das dann die eigentliche Anwendungslogik für den Use-Case durchführt).

b) Zeichnen Sie ein Sequenzdiagramm, das den beschriebenen Nachrichtenaustausch modelliert. Alle Nachrichten sollen synchron übermittelt werden. Stellen Sie in dem Sequenzdiagramm die Aktivierungsphasen der Objekte klar dar (im Unterschied zu den Zeiten, in denen ein Objekt lebt, aber nicht an einer Interaktion beteiligt ist). Die Nachrichten brauchen keine Parameter enthalten.

c) Geben Sie für jede Klasse in Abbildung 2 an, welche Operationen die Klasse anbieten muss, damit der beschriebene Nachrichtenaustausch stattfinden kann. Die Namen der Operationen sollen sich aus den Namen der Nachrichten ergeben.

Fortsetzung nächste Seite!

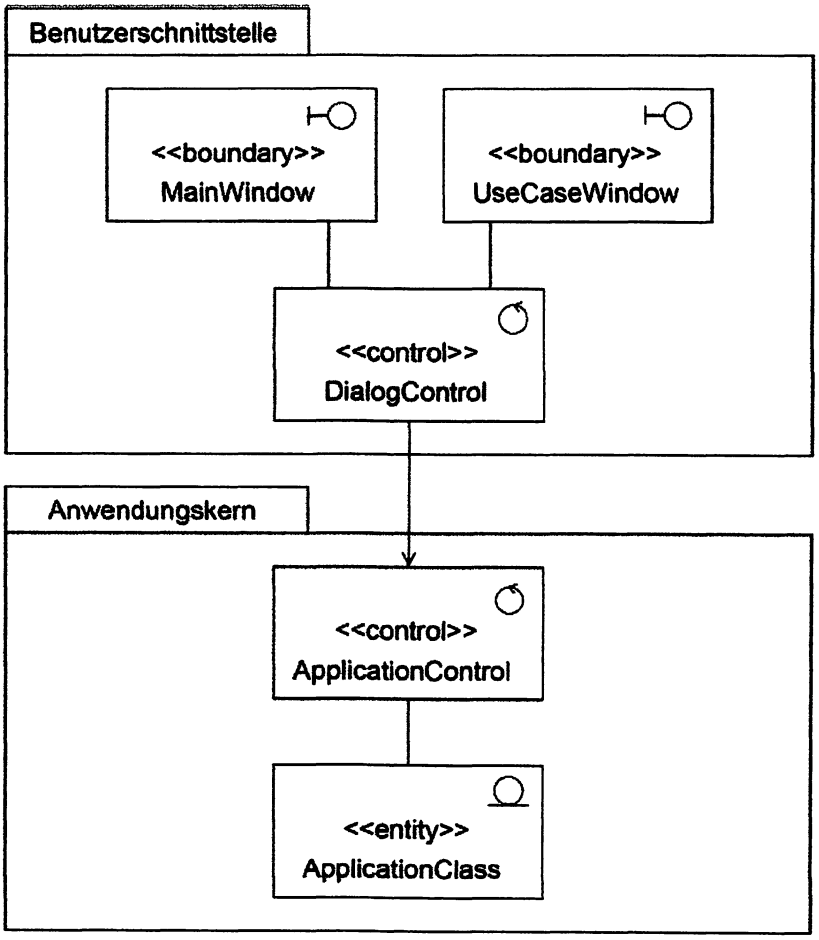


Abbildung 2: Zwei-Schichten Architektur

Fortsetzung nächste Seite!

Aufgabe 4: Sichtbarkeiten (16 Punkte)

Abbildung 3 zeigt ein statisches UML-Modell für eine Systemarchitektur mit Paketen sowie für jede der vorkommenden Klassen eine Java-Implementierung. Stellen Sie fest, welche Anweisungen in den Java-Implementierungen gemäß der angegebenen Sichtbarkeiten **unzulässig** sind und geben Sie jeweils eine kurze Begründung dafür an. Für jede falsche Antwort wird eine (vorhandene) richtige Antwort gestrichen.
Bemerkung: Für Ihre Antwort sind die im Java-Code angegebenen Sichtbarkeiten ausschlaggebend. Das UML-Modell kann jedoch hilfreich sein, um die Systemarchitektur zu überblicken. Das Symbol "~" im UML-Modell drückt Paketsichtbarkeit von Elementen einer Klasse aus. Klassen, und damit ihre nicht-privaten Elemente, sind immer im selben Paket sichtbar, auch wenn die Klasse mit "–" gekennzeichnet ist. Die <<access>>-Beziehung drückt privaten Paket-Import aus, d.h., importierte Elemente können nicht weiter exportiert werden. Die gestrichelten Linien drücken Benutzungs-Abhängigkeiten aus.

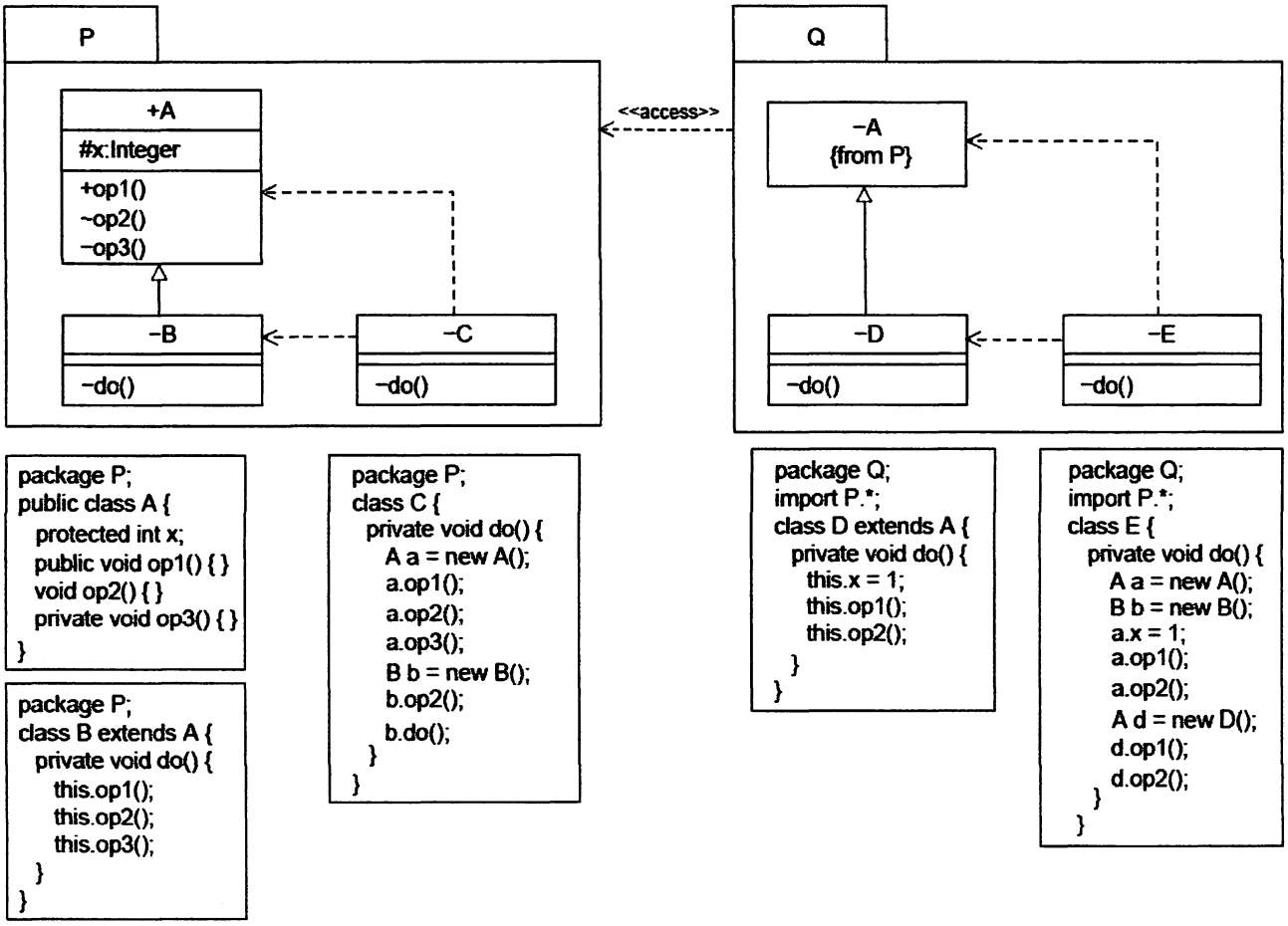


Abbildung 3: Systemarchitektur mit Paketen und Java-Implementierung

Fortsetzung nächste Seite!

Aufgabe 5: Verifikation funktionaler Programme (22 Punkte)

Gegeben sei der folgende polymorphe Datentyp *'a leafTree* zur Darstellung binärer Bäume mit Blättern. Die Blätter sind mit Elementen eines beliebigen Typs *'a* markiert.

datatype *'a leafTree = Leaf* **of** *'a | Node* **of** *'a leafTree * 'a leafTree*

Die Höhe eines Baums wird durch die folgende rekursive Funktion *height* (durch Pattern-Matching gemäß dem strukturellen Aufbaus eines Baumes) berechnet. Sie berechnet die Anzahl der maximal nötigen Schritte, um von der Wurzel des Baumes zu einem Blatt zu gelangen.

fun *height(Leaf (x))* = 0
  | *height(Node(T1, T2)) = max(height(T1), height(T2))* + 1

Die Anzahl der Knoten eines Baumes (einschließlich der Blätter) wird durch die folgende rekursive Funktion *nodes* berechnet.

fun *nodes(Leaf (x))* = 1
  | *nodes(Node(T1, T2)) = nodes(T1) + nodes(T2)* + 1

Es wird behauptet, dass für alle Ausdrücke *T* des Typs *'a leafTree* folgendes gilt:

$$nodes(T) \leq 2^{height(T)+1} - 1.$$

Beweisen Sie diese Behauptung durch strukturelle Induktion gemäß des Aufbaus des Ausdrucks *T*. Geben Sie zunächst an, welche Fälle für die Gestalt von T unterschieden werden müssen und welcher Fall der Basisfall der Induktion ist. Für einen Nicht-Basisfall ist die Induktionsvoraussetzung zu nennen und genau anzugeben, wo diese im Beweis verwendet wird.

Fortsetzung nächste Seite!

Teilaufgabe 2

Aufgabe 1: Vermischte Fragen (2+2+2+2+2+2=12 Punkte)

Beantworten Sie die folgenden Fragen und begründen oder erläutern Sie Ihre Antwort in jeweils ein bis zwei Sätzen.

a) Was versteht man unter einem abgeleiteten Attribut?

b) Was versteht man unter referentieller Integrität?

c) Dürfen Primärschlüsselwerte NULL sein?

d) Was versteht man unter embedded SQL?

e) Was ist der Effekt des Aufrufs von commit innerhalb einer Transaktion?

f) Müssen Transaktionen von einem Datenbankverwaltungssystem, das ACID garantiert, strikt hintereinander ausgeführt werden oder dürfen mehrere Transaktionen parallel ausgeführt werden?

Aufgabe 2: ER-Modellierung (22 Punkte)

Im Folgenden finden Sie die Beschreibung eines Online-Bewerbungssystems. Erstellen Sie zu dieser Beschreibung ein erweitertes ER-Diagramm. Kennzeichnen Sie die Primärschlüssel durch passendes Unterstreichen und geben Sie die Kardinalitäten in Chen-Notation (= Funktionalitäten) an. Kennzeichnen Sie auch die totale Teilnahme (= Existenzabhängigkeit, Partizipität) von Entity-Typen.

In dem System werden Stellen ausgeschrieben. Jede Stelle hat eine eindeutige ID, eine Bezeichnung und eine Beschreibung. Stellen können notwendige Kenntnisse zugeordnet werden. Diese haben einen eindeutigen Namen und eine Beschreibung. Das System verwaltet auch Personendaten. Dabei werden Bewerber und Mitarbeiter unterschieden. Daten anderer Personengruppen sind nicht zugelassen, aber eine Person kann gleichzeitig Bewerber und Mitarbeiter sein. Alle Personen haben eine eindeutige E-Mail-Adresse, ein oder mehrere Vornamen, einen Nachnamen und ein Geburtsdatum. Bewerbern werden weiterhin Kenntnisse zugeordnet; das können auch mehrere sein. Bewerber können sich für beliebig viele Stellen bewerben. Zur Bewerbung eines Bewerbers werden ein Text und das Bewerbungsdatum gespeichert. Bewerber können sich zu einem anderen Zeitpunkt erneut auf die gleiche Stelle bewerben, aber mit einer Bewerbung immer nur auf genau eine Stelle. Einer Bewerbung kann ein zuständiger Mitarbeiter zugewiesen werden, wobei ein Mitarbeiter für beliebig viele Bewerbungen zuständig sein kann.

Fortsetzung nächste Seite!

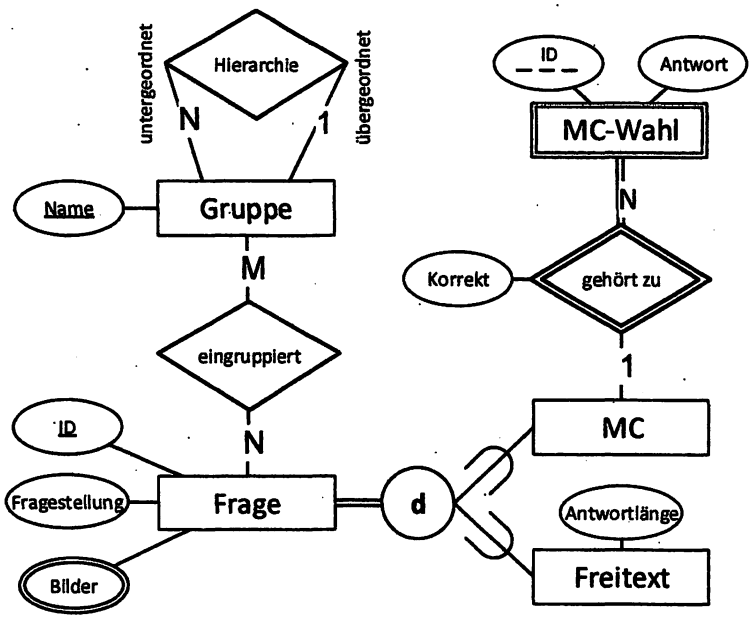
Aufgabe 3: Relationaler Entwurf (13 Punkte)

Entwerfen Sie zum untenstehenden ER-Diagramm ein Relationenschema (in dritter Normalform, 3 NF) mit möglichst wenigen Relationen.

Verwenden Sie dabei folgende Notation: Primärschlüssel werden durch Unterstreichen gekennzeichnet, 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. Wenn ein Attribut zur korrekten Abbildung des ER-Diagramms als `UNIQUE` oder `NOT NULL` ausgezeichnet werden muss, geben Sie dies an.

Beispiel:

```
Relation1 (Primärschlüssel, Attribut1, Attribut2,
   Fremdschlüsselattribut1[Relation2],
   (Fremdschlüssel2_Attribut1, Fremdschlüsel2_Attribut2)[Relation3]);
Attribut1 UNIQUE
```



Zwischen „Frage“ einerseits und „MC“ bzw. „Freitext“ andererseits besteht eine Generalisierungsbeziehung. Das „d“ im Kreis steht für „disjunkt“, also nicht überlappende Untertypen.

Fortsetzung nächste Seite!

Aufgabe 4: SQL (5+3+3+4+6+8+7+4=40 Punkte)

Gegeben sind folgende Relationen aus einer Datenbank zur Verwaltung von Triathlon-Wettbewerben.

```
Athlet(ID, Vorname, Nachname)
Ergebnis(Athlet[Athlet], Wettbewerb[Wettbewerb], Schwimmzeit, Radzeit,
  Laufzeit); Schwimmzeit NOT NULL
Wettbewerb(Name, Jahr)
```

Verwenden Sie im Folgenden nur Standard-SQL und keine produktspezifischen Erweiterungen. Sie dürfen bei Bedarf Views anlegen. Geben Sie einen Datensatz, also eine Entity, nicht mehrfach aus.

a) Schreiben Sie eine SQL-Anweisung, die die Tabelle „Ergebnis" anlegt. Gehen Sie davon aus, dass die Tabellen „Athlet" und „Wettbewerb" bereits existieren. Verwenden Sie sinnvolle Datentypen.

b) Schreiben Sie eine SQL-Anweisung, die die Radzeit des Teilnehmers mit der ID 12 beim Wettbewerb „Zürichsee" um eins erhöht.

c) Schreiben Sie eine SQL-Anweisung, die die Namen aller Wettbewerbe des Jahres 2018 ausgibt, absteigend sortiert nach Name.

d) Schreiben Sie eine SQL-Anweisung, die die Namen aller Wettbewerbe ausgibt, in der die durchschnittliche Schwimmzeit größer als 10 ist.

e) Schreiben Sie eine SQL-Anweisung, die die IDs aller Athleten ausgibt, die im Jahr 2017 an keinem Wettbewerb teilgenommen haben.

f) Schreiben Sie eine SQL-Anweisung, die die Nachnamen aller Athleten ausgibt, die mindestens 10 Wettbewerbe gewonnen haben, das heißt im jeweiligen Wettbewerb die kürzeste Gesamtzeit erreicht haben. Die Gesamtzeit ist die Summe aus Schwimmzeit, Radzeit und Laufzeit. Falls zwei Athleten in einem Wettbewerb die gleiche Gesamtzeit erreichen, sind beide Sieger.

g) Schreiben Sie eine SQL-Anweisung, die die Top-Ten der Athleten mit der schnellsten Schwimmzeit des Wettbewerbs „Paris" ausgibt. Ausgegeben werden sollen die Platzierung (1 bis 10) und der Nachname des Athleten, aufsteigend sortiert nach Platzierung. Gehen Sie davon aus, dass keine zwei Athleten die gleiche Schwimmzeit haben und verwenden Sie **keine** produktspezifischen Anweisungen wie beispielsweise `rownum`, `top` oder `limit`.

h) Schreiben Sie einen Trigger, der beim Einfügen neuer Tupel in die Tabelle „Ergebnis" die Schwimmzeit auf den Wert 0 setzt, falls diese negativ ist.

Fortsetzung nächste Seite!

Aufgabe 5: Relationale Algebra (5+5=10 Punkte)

a) Formulieren Sie basierend auf den in Aufgabe 4 gegebenen Relationen eine Anfrage **in der Relationenalgebra**, die die IDs derjenigen Athleten ausgibt, die an allen Wettbewerben des Jahres 2015 teilgenommen haben.

b) Formulieren Sie basierend auf den in Aufgabe 4 gegebenen Relationen eine Anfrage **im relationalen Domänenkalkül**, die die Nachnamen der Athleten ausgibt, die im Jahr 2016 an mindestens einem Wettbewerb teilgenommen haben.

Aufgabe 6: Normalisierung (2+2+9=13 Punkte)

a) Erläutern Sie in etwa zwei Sätzen, wann eine Zerlegung von Relationen **verlustlos** ist.

b) Erläutern Sie in etwa zwei Sätzen, wann eine Zerlegung von Relationen **abhängigkeitsbewahrend** (abhängigkeitserhaltend) ist.

c) Erläutern Sie die drei Anomalien, die bei einem nicht-normalisierten Relationenschema auftreten können anhand von jeweils einem Beispiel. Geben Sie dazu am Besten zunächst ein nicht-normalisierte Relationenschema an und dann jeweils ein Beispiel anhand dieses Schemas.

Aufgabe 7: Optimierung (6+4=10 Punkte)

a) Übertragen Sie folgendes SQL-Statement in einen **nicht optimierten** Ausdruck der relationalen Algebra (= Anfragegraph, Operatorgraph).

```
SELECT r.id , SUM(p.anzahl)
FROM r, p
WHERE r.id = p.r
GROUP BY r.id , r.plz
HAVING SUM (p.anzahl) < 5 AND r.plz = '11037';
```

Sie können für die Aggregationsfunktion SUM in einer Attributliste SUM(r.A) mit einem qualifizierten Attribut r.A schreiben.

b) Nennen Sie zwei Möglichkeiten, den Ausdruck der relationalen Algebra aus der vorhergehenden Teilaufgabe logisch (= algebraisch) zu optimieren. Beziehen Sie sich auf konkrete Stellen und Operatoren des von Ihnen aufgestellten Ausdrucks.

Thema Nr. 2
(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

Teilaufgabe 1

Aufgabe 1: Projektmanagement (10 Punkte)

Ein Team von zwei Softwareentwicklern soll ein Projekt umsetzen, das in sechs Arbeitspakete unterteilt ist. Zeichnen Sie ein Gantt-Diagramm (Balkenplan), das eine kürzestmögliche Projektabwicklung beinhaltet. Die Dauer der Arbeitspakete und ihre Abhängigkeiten können Sie aus der folgenden Tabelle entnehmen.

Name	Dauer in Wochen	Abhängig von
A1	2	-
A2	5	-
A3	2	A1
A4	5	A1, A2
A5	7	A3
A6	4	A4

Bestimmen Sie anschließend die Länge des kritischen Pfades und geben Sie an, welche Arbeitspakete an ihm beteiligt sind.

Aufgabe 2: Softwareprozesse (12 Punkte)

a) Nennen Sie vier Phasen der Softwareentwicklung in der richtigen Reihenfolge.

b) Nennen und erklären Sie kurz jeweils zwei Vorteile und zwei Nachteile beim Einsatz von Prototypen im Entwicklungsprozess.

Aufgabe 3: Softwareentwurf (19 Punkte)

Zeichnen Sie ein UML-Klassendiagramm zu folgendem Problem:
Die Verwaltung einer Universität möchte ein neues Raumverwaltungssystem entwickeln. In dem System soll für jeden Raum die Raumnummer (einfacher Integer) und die Größe in Quadratmetern gespeichert werden. Bei den Räumen soll zwischen Büros, Hörsälen und Seminarräumen unterschieden werden. Zu jedem Hörsaal und Seminarraum soll zusätzlich noch die maximale Anzahl von Sitzplätzen gespeichert werden. Außerdem soll jedes Büro mindestens einem Lehrstuhl zugeordnet sein. Es kann dabei auch vorkommen, dass ein Büro zwei oder maximal drei Lehrstühlen zugeordnet ist. Für jeden Lehrstuhl soll außerdem der Name des Lehrstuhlinhabers gespeichert werden.

Wichtig: Es müssen *keine* Methoden modelliert werden.

Fortsetzung nächste Seite!

Aufgabe 4: Design Patterns (11 Punkte)

a) Nennen Sie vier Vorteile, die die Verwendung von Design Patterns haben kann.

b) Nennen Sie drei Arten von Design Patterns und erklären Sie kurz ihren Verwendungszweck.

Aufgabe 5: Implementierung (24 Punkte)

a) Wie kann man die Lesbarkeit und Verständlichkeit des folgenden Quelltextes erhöhen? Nennen Sie vier Aspekte und geben Sie jeweils ein konkretes Beispiel anhand des gegebenen Quelltextes.

```
1  public class Calculations {
2  final static int zero=0;
3
4  /**
5   * Important computation
6   */
7  public int compl(int BASE, int para2) {
8  if (para2<=zero) return 1;
9  return BASE * this.compl(BASE,para2 - 1);
10 }
11 }
```

b) Lösen Sie die folgende Berechnung unter Zuhilfename einer while-Schleife (in der die Schleifenprüfung vor Ausführung des Rumpfes durchgeführt wird) und einer do-while-Schleife (in der die Schleifenprüfung nach der ersten Ausführung des Rumpfes durchgeführt wird):

Hinweise:

- Beachten Sie, dass das übergebene Array auch eine Länge von 0 haben kann.
- Sie können davon ausgehen, dass die Methode nicht mit *null* aufgerufen wird.

```
1 private static void forVariant(int[] myArray) {
2   int sum = 0;
3   for(int i = 0; i < myArray.length; i += 2) {
4     sum += myArray[i];
5   }
6   System.out.println(sum);
7 }
```

Fortsetzung nächste Seite!

Aufgabe 6: Entwicklung und Wartung (12 Punkte)

Ordnen Sie die gegebenen Wartungsaufgaben A – D jeweils einer der drei folgenden Wartungskategorien I – III zu. Begründen Sie Ihre Entscheidung kurz in einem Satz.

Wartungskategorien

(I) Corrective Maintenance (korrigierend)

(II) Adaptive Maintenance (adaptierend)

(III) Perfective Maintenance (perfektionierend)

Wartungsaufgaben

(A) Die Webseite eines Kunden soll eine Datenbankanbindung bekommen, da der Kunde sein Business-Modell geändert hat und in der Folge mit viel größeren Datenvolumina und konkurrierenden Datenzugriffen rechnet. Die Datenbankanbindung war ursprünglich nicht vorgesehen und die Daten wurden in Textdateien gespeichert.

(B) Nachdem ein Kunde auf neue Hardware und die nächste Version des Betriebssystems umgestellt hat, treten unerwartete Verhalten in den Prozessabläufen des Softwaresystems auf, die früher nicht aufgetreten sind. Die Systemumgebung wird so nicht vom Hersteller unterstützt und das System muss angepasst werden.

(C) Die Ausgabe von Prozessergebnissen läuft komplett zufriedenstellend, dennoch lässt sich die Performanz verbessern. Führen Sie die möglichen Änderungen durch.

(D) Bei der Eingabe bestimmter Zahlenfolgen stürzt das Programm ohne sichtbare Fehlermeldung ab.

Fortsetzung nächste Seite!

Aufgabe 7: Refactoring and Versionskontrolle (12 Punkte)

a) Nennen Sie sechs beliebige Refactorings, die zur Verbesserung der Lesbarkeit, Verständlichkeit und Erweiterbarkeit eines Softwaresystems durchgeführt werden können.

b) Diskutieren Sie anhand der folgenden Projekt-Beschreibung was für den Einsatz eines zentralen oder dezentralen Versionskontrollsystems spricht. Begründen Sie Ihre Entscheidung.

Wichtig: Es geht bei dieser Entscheidung nur um *zentral* oder *dezentral*, es sollen keine konkreten Versionskontrollsysteme (wie bspw. GIT oder SVN) betrachtet werden.

Eine Softwarefirma möchte ein neues Softwareentwicklungsprojekt umsetzen und muss sich entscheiden, ob es ein zentrales oder dezentrales Versionskontrollsystem einsetzt. Es gibt dabei folgende Dinge im Entwicklungsprozess zu beachten:

- Die Entwickler dürfen Home-Office machen.
- Einige Entwickler sind viel auf Reisen und deswegen nicht immer mit dem Intranet verbunden.
- Es gibt einen zentralen Build-Server im Intranet der Firma, der sich aus einer Kopie des Repositories speist. Es ist wegen der technischen Details nicht möglich, Tests auf den Arbeitsplatzrechnern der Entwickler laufen zu lassen.
- Features werden auf Branches entwickelt, der Build-Server testet stets nur einen bestimmten Branch, den die Entwickler in Absprache untereinander konfigurieren können.

Aufgabe 8: Verifikation, Validierung und Testing (8 Punkte)

a) Erklären Sie in wenigen Sätzen den Unterschied zwischen Fehlervermeidung und Fehlertoleranz.

b) Was versteht man unter Regression-Testing und welche Anforderungen sind an Regressionstests gestellt?

Aufgabe 9: Softwarequalität (4 Punkte)

Warum sollte Qualitätsmanagement vom Projektmanagement getrennt werden?

Aufgabe 10: Softwaremaße (8 Punkte)

Nennen Sie drei Kategorien von einfachen Softwaremaßen und beschreiben Sie kurz, wie mithilfe dieser die Komplexität von Softwareprojekten ermittelt wird.

Fortsetzung nächste Seite!

Teilaufgabe 2

Aufgabe 1: Datenmodellierung mit ER-Modellen (15+4+12+3=34 Punkte)

Wir betrachten eine Datenbank, die einem Sportverband (z.B. dem *Deutschen Fußballbund*) zur Verwaltung seiner Unterverbände, Vereine, sowie deren Mitglieder und Funktionsträger dient. Sie verfügt über die folgenden Eigenschaften:

- Für jeden (Unter-)Verband werden sein Name sowie eine eindeutige Verbandsnummer abgespeichert. So gibt es beispielsweise den *Bayerischen Fußballverband.*
- Die Verbände sind hierarchisch organisiert, d.h. zu jedem Verband kann sein Dachverband angegeben werden. Für die höchste Ebene (im Beispiel *Deutscher Fußballbund*) wird kein Dachverband hinterlegt.
- Jeder Verein gehört genau einem Verband an. Ein Verein besitzt eine Vereinsnummer, die zusammen mit der Verbandsnummer eindeutig ist. Zum Verein werden der Vereinsname (z.B. *FC Bayern München*) sowie das Gründungsdatum abgespeichert.
- Eine Person ist gekennzeichnet durch eine eindeutige Nummer sowie einen Vor- und einen Nachnamen.
- Eine Person kann Mitglied in beliebig vielen Vereinen sein. Zu jeder Mitgliedschaft sollen Beginn und Ende abgespeichert werden.
- Es gibt verschiedene Funktionen, wie z.B. *Vorsitzende/r* oder *Kassierer/in.* Ihre Bezeichnung ist eindeutig. Zu jeder Funktion wird abgespeichert, ob sie nach Vereinsrecht notwendig ist oder nicht. Für die Verbände und Vereine können Personen mit ihrer Verbands- bzw. Vereinsfunktion hinterlegt werden. So hat bspw. *Uli Hoeneß* beim *FC Bayern München* die Vereinsfunktion als Präsident inne.

Mögliche Dateneinträge sind *kursiv* hervorgehoben und dienen einzig der Veranschaulichung.

a) Modellieren Sie das oben genannte Datenbanksystem möglichst vollständig in einem erweiterten ER-Diagramm. Kennzeichnen Sie die Schlüssel der Entity-Typen und geben Sie für die Relationship-Typen jeweils Funktionalitätsbedingungen (in N:M-Notation) oder Kardinalitätsbedingungen (in (MIN, MAX)-Notation) an.

b) Was ist eine schwache Entity? Ist der Schlüssel einer schwachen Entity global eindeutig? Falls nein, wie entsteht ein global eindeutiger Schlüssel für eine solche Entity?

c) Übertragen Sie Ihr ER-Modell in das relationale Datenmodell. Erstellen Sie dazu die Tabellen `Verband`, `Verein` und `Person` mit Hilfe von CREATE TABLE Statements in SQL. Berücksichtigen Sie die Fremdschlüsselbeziehungen, und geben Sie für diese explizit die Update- und Delete-Modi über `ON UPDATE` bzw. `ON DELETE` an. Wählen Sie sinnvolle Datentypen.

Fortsetzung nächste Seite!

d) Was ist zu beachten, wenn in SQL zu einem Verein all seine Dachverbände bestimmt werden sollen? Sie müssen die SQL-Anfragen nicht angeben, Sie sollen nur die auftretende Problematik beleuchten.

Aufgabe 2: SQL (6+3+3+2,5+4,5+3+5+2+8 = 37 Punkte)

Wir betrachten eine Datenbank zur Speicherung der Fahrplandaten von Fernverkehrszügen der Deutschen Bahn. Die Datenbank enthält die Tabellen BAHNHOF, ZUG und HALT, deren Inhalt im Folgenden in Auszügen angegeben ist:

BAHNHOF		
Bhf	Name	Gleise
AH	Hamburg Hbf	8
BH	Berlin Hbf	14
FF	Frankfurt Hbf	24
HH	Hannover Hbf	12
MH	München Hbf	32
NN	Nürnberg Hbf	22
TS	Stuttgart Hbf	16

HALT			
Zug_Nr	Bhf	An	Ab
ICE 882	MH	-	08:22
ICE 882	NN	09:30	09:33
ICE 882	HH	12:32	12:36
ICE 882	AH	13:54	-
ICE 595	BH	-	07:12
ICE 595	FF	11:44	11:50
ICE 595	TS	13:46	13:50
ICE 595	MH	16:04	-
ICE 778	FF	-	16:34

ZUG	
Zug_Nr	Wagen
ICE 882	12
ICE 595	10
ICE 778	8

a) Entwerfen Sie passend zu diesen Tabellen ein erweitertes ER-Diagramm. Geben Sie für die Relationship-Typen jeweils sinnvolle Funktionalitätsbedingungen (in N:M-Notation) oder Kardinalitätsbedingungen (in (MIN, MAX)-Notation) an.

b) Geben Sie das Resultat der folgenden SQL-Anweisung an:

```
SELECT Zug.Zug_Nr, SUM(Wagen) AS Summe FROM Halt, Zug
WHERE Halt.Zug_Nr = Zug.Zug_Nr
GROUP BY Zug.Zug_Nr;
```

c) Geben Sie das Resultat der folgenden SQL-Anweisung an:

```
SELECT Bhf, COUNT(*) AS Gleise FROM Halt
GROUP BY Bhf
HAVING Gleise != 1;
```

d) Schreiben Sie eine SQL-Anweisung, die den HALT für *ICE 595* in *Hannover Hbf* einfügt (Ankunft 09:11 Uhr, Abfahrt 09:13 Uhr).

e) Schreiben Sie eine SQL-Anweisung, die einen View erzeugt, der zu jeder Zugnummer die Anzahl der angefahrenen Halte enthält.

Fortsetzung nächste Seite!

f) Schreiben Sie eine SQL-Anweisung, die alle Züge löscht, für die keine Halte gespeichert sind.

g) Schreiben Sie eine SQL-Anweisung, die die Namen aller Bahnhöfe zurückliefert, deren Gleisanzahl kleiner ist als die durchschnittliche Gleisanzahl aller Bahnhöfe. Die Bahnhöfe sollen aufsteigend nach ihrer Gleisanzahl ausgegeben werden, bei Gleichheit alphabetisch sortiert.

h) Schreiben Sie eine SQL-Anweisung, die alle Wagenanzahlen der Züge um eins erhöht.

i) Schreiben Sie eine SQL-Anweisung, die die Zugnummern all jener Züge zurückliefert, die im Laufe einer Fahrt den selben Bahnhof mehrfach anfahren. Geben Sie jedes Paar von Zugnummer und Bahnhofskürzel nur einmal aus.

Aufgabe 3: Funktionale Abhängigkeiten und Zerlegungen (5+5+5+9+4+6=34 Punkte)

Gegeben sei das Relationenschema $R = (U; F)$ mit der Attributmenge U und der Menge F von funktionalen Abhängigkeiten (fds):

$$U = \{A,B,C,D,E,G,H,I\};$$
$$F = \{\{B,C\} \rightarrow \{E\}, \{B,C,G\} \rightarrow \{H\}, \{D\} \rightarrow \{A,I\}, \{E,G\} \rightarrow \{H\}\}.$$

a) Zeigen Sie, dass die fd $\{B,C,G\} \rightarrow \{H\}$ in F redundant ist, d.h., dass sie bereits aus den drei anderen fds von F folgt.

b) Geben Sie alle Schlüssel für das Relationenschema R sowie die Nichtschlüsselattribute an (mit Begründung).

c) Ist R in 3NF, ist R in BCNF (mit Begründung)?

d) Geben Sie eine Basis (kanonische Überdeckung) G von F an und führen Sie damit die 3NF-Synthese aus.

e) Welches Ergebnis würde der Algorithmus 3NF-Synthese liefern, der fds in der Basis mit gleichen linken Seiten zusammenfasst?

f) Geben Sie eine erlaubte Instanz r zu R mit mindestens 4 Tupeln an, so dass die funktionale Abhängigkeit $\{B,C\} \rightarrow \{H\}$ nicht gilt (mit Begründung).

Hinweis: Wenn Sie die Attributwerte für B bzw. E alle konstant auf b bzw. e setzen und die Werte der Attribute $X \in \{A,D,G,H,I\}$ tupelweise durchnummerieren (d.h. X hat den Wert x_i in Tupel i), dann müssen Sie nur geeignete Werte für das Attribut C finden.

Fortsetzung nächste Seite!

Aufgabe 4: Transaktionen (8+7=15 Punkte)

Gegeben seien die folgenden beiden Schedules 1 und 2.

- Schedule 1:

T_1	T_2	T_3	T_4	T_5
				$read(A)$
$read(B)$				
$read(A)$				
				$write(A)$
			$read(C)$	
			$write(C)$	
		$read(A)$		
$write(B)$				
	$read(B)$			
	$write(B)$			
		$read(C)$		

- Schedule 2:

T_1	T_2	T_3
	$read(X)$	
$read(Z)$		
$write(Z)$		
		$read(Y)$
	$read(Y)$	
	$write(X)$	
	$write(Y)$	
		$write(Y)$

In beiden Schedules sollen alle Transaktionen am Ende erfolgreich beendet werden (commit).

a) Prüfen Sie Schedule 1 auf konfliktbasierte Serialisierbarkeit. Geben Sie eine Begründung und eine eventuelle Serialisierungsreihenfolge an.

b) Prüfen Sie Schedule 2 auf konfliktbasierte Serialisierbarkeit. Geben Sie eine Begründung und alle möglichen Serialisierungsreihenfolgen an.

Frühjahr 2019

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer
Kennzahl: _____ Kennwort: _____ Arbeitsplatz-Nr.: _____	**Frühjahr 2019**	**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: **15**

Bitte wenden!

Thema Nr. 1

Teilaufgabe I: Datenbanken

Aufgabe 1: ER-Modellierung

Gegeben seien folgende Informationen:

- Ein Land, beschrieben durch eine eindeutige Vorwahl und einen Namen, hat mehrere Anbieter für Mobilfunk, die wiederum einen Namen und eine innerhalb des Landes eindeutige Vorwahl besitzen.

- Ein Handy gehört zu einem Anbieter und besitzt neben einer Farbe eine nur bei dem jeweiligen Anbieter eindeutige Nummer. Handys können andere Handys anrufen, wobei der Zeitpunkt gespeichert wird.

- Personen werden anhand einer PersID identifiziert. Zudem werden Informationen über Name und Geburtsdatum gespeichert. Jeder Kontakt, identifiziert über eine ID und ein optionales Bild, entspricht genau einer Person. Zudem wird jeder Kontakt in mindestens einem Handy gespeichert.

- Eine Person kann mehrere Handys besitzen, aber ein Handy gehört höchstens einer Person.

- Handys lassen sich unter anderem in Smartphones und Tastenhandys unterscheiden, wobei ein Handy nicht beides sein kann. Für ein Smartphone wird der Speicherplatz und für ein Tastenhandy die Größe der Tasten gespeichert.

a) Erstellen Sie für das oben gegebene Szenario ein geeignetes ER-Diagramm.
Verwenden Sie dabei – wenn angebracht – das Prinzip der Spezialisierung. Kennzeichnen Sie die Primärschlüssel der Entity-Typen, totale Teilnahmen und schwache Entity-Typen.
Zeichnen Sie die Funktionalitäten der Relationship-Typen in das Diagramm ein.

b) Überführen Sie Ihr in Aufgabe a) erstelltes Modell in ein verfeinertes relationales Schema. Kennzeichnen Sie die Schlüssel durch Unterstreichen. Datentypen müssen nicht angegeben werden. Die einzelnen Schritte müssen angegeben werden.

Fortsetzung nächste Seite!

Aufgabe 2: (SQL und relationale Algebra)

Gegeben sei der folgende Ausschnitt des Schemas für die Verwaltung der Einnahme von Medikamenten:

Person : {[
ID : INTEGER,
Name : VARCHAR(255),
Wohnort : VARCHAR(255)
]}

Hersteller : {[
ID : INTEGER,
Name : VARCHAR(255),
Standort : VARCHAR(255),
AnzahlMitarbeiter : INTEGER
]}

Medikament : {[
ID : INTEGER,
Name : VARCHAR(255),
Kosten : INTEGER,
Wirkstoff : VARCHAR(255),
produziert_von : INTEGER
]}

nimmt : {[
Person : INTEGER,
Medikament : INTEGER,
von : DATE,
bis : DATE
]}

hat_Unverträglichkeit_gegen : {[
Person : INTEGER,
Medikament : INTEGER
]}

Die Tabelle *Person* beschreibt Personen über eine eindeutige ID, deren Namen und Wohnort. Die Tabelle *Medikament* enthält Informationen über Medikamente, unter anderem deren Namen, Kosten, Wirkstoffe und einen Verweis auf den Hersteller dieses Medikaments. Die Tabelle *Hersteller* verwaltet verschiedene Hersteller von Medikamenten. Die Tabelle *hat_Unverträglichkeit_gegen* speichert die IDs von Personen zusammen mit den IDs von Medikamenten, gegen die diese Person eine Unverträglichkeit hat. Die Tabelle *nimmt* hingegen verwaltet die Einnahme der verschiedenen Medikamente und speichert zudem in welchem Zeitraum eine Person welches Medikament genommen hat bzw. nimmt.

Beachten Sie bei der Formulierung der SQL-Anweisungen, dass die Ergebnisrelationen keine Duplikate enthalten dürfen. Sie dürfen geeignete Views definieren.

a) Schreiben Sie SQL-Anweisungen, die für die bereits existierende Tabelle `nimmt` alle benötigten Fremdschlüsselconstraints anlegt. Erläutern Sie kurz, warum die Spalten `von` und `bis` Teil des Primärschlüssels sind.

b) Schreiben Sie eine SQL-Anweisung, welche sowohl den Namen als auch die ID von Personen und Medikamenten ausgibt, bei denen die Person das jeweilige Medikament nimmt.

c) Schreiben Sie eine SQL-Anweisung, welche die ID und den Namen der Medikamente mit den niedrigsten Kosten je Hersteller bestimmt.

d) Schreiben Sie eine SQL-Anweisung, die die Anzahl aller Personen ermittelt, die ein Medikament genommen haben, gegen welches sie eine Unverträglichkeit entwickelt haben.

Fortsetzung nächste Seite!

e) Schreiben Sie eine SQL-Anweisung, die die ID und den Namen derjenigen Personen ermittelt, die weder ein Medikament mit dem Wirkstoff `Paracetamol` noch ein Medikament mit dem Wirkstoff `Ibuprofen` genommen haben.

f) Schreiben Sie eine SQL-Anweisung, welche die Herstellernamen und die Anzahl der bekannten Unverträglichkeiten gegen Medikamente dieses Herstellers ermittelt. Das Ergebnis soll aufsteigend nach der Anzahl der bekannten Unverträglichkeiten sortiert werden.

g) Formulieren Sie eine Anfrage in **relationaler Algebra**, welche die Wohnorte aller Personen bestimmt, welche ein Medikament mit demWirkstoff `Paracetamol` nehmen oder genommen haben. Die Lösung kann als Baum- oder als Term-Schreibweise angegeben werden.

h) Formulieren Sie eine Anfrage in **relationaler Algebra**, welche die Namen aller Personen bestimmt, die von allen bekannten Herstellern, deren Standort `München` ist, Medikamente nehmen oder genommen haben. Die Lösung kann als Baum- oder als Term-Schreibweise angegeben werden.

Aufgabe 3: (Entwurfstheorie)

Gegeben sei folgendes relationales Schema R in erster Normalform:

$$R : \{[A, B, C, D, E, F]\}$$

Für R gelte folgende Menge FD funktionaler Abhängigkeiten:
$FD = \{$

$$\begin{gathered} ADF \rightarrow E, \\ BC \rightarrow AE, \\ D \rightarrow B, \\ DE \rightarrow CB, \\ A \rightarrow F \end{gathered}$$

$\}$

a) Bestimmen Sie alle Kandidatenschlüssel/Schlüsselkandidaten von R mit FD.
Hinweis: Die Angabe von Attributmengen, die keine Kandidatenschlüssel sind, führt zu Abzügen.

b) Prüfen Sie, ob R mit FD in 2NF bzw. 3NF ist.

c) Bestimmen Sie mit folgenden Schritten eine kanonische Überdeckung FD_C von FD:

i. Führen Sie eine Linksreduktion von FD durch. Geben Sie die Menge funktionaler Abhängigkeiten nach der Linksreduktion an (FD_L).

ii. Führen Sie eine Rechtsreduktion des Ergebnisses der Linksreduktion (FD_L) durch. Geben Sie die Menge funktionaler Abhängigkeiten nach der Rechtsreduktion an (FD_R).

iii. Bestimmen Sie eine kanonische Überdeckung FD_C von FD auf Basis des Ergebnisses der Rechtsreduktion (FD_R).

Fortsetzung nächste Seite!

d) Zerlegen Sie R mit FD_C mithilfe des *Synthesealgorithmus* in 3NF. Geben Sie zudem alle funktionalen Abhängigkeiten der erzeugten Relationenschemata an.

e) Prüfen Sie *für alle Relationen* der Zerlegung aus d), ob sie jeweils in BCNF sind.

Aufgabe 3: (Transaktionen)

a) Nennen Sie die Phasen, die während des Ablaufs eines Zweiphasen-Sperrprotokolls auftreten und beschreiben Sie diese kurz.

b) Betrachten Sie den folgenden Schedule S:

T_1	T_2	T_3
	$r_2(x)$	
$w_1(y)$		
		$r_3(z)$
$r_1(x)$		
	$w_2(y)$	
	$r_2(z)$	
		$r_3(x)$
$w_1(z)$		
c_1		
		$w_3(x)$
		c_3
	$w_2(x)$	
	c_2	

Geben Sie den Ausgabeschedule (einschließlich der Operationen zur Sperranforderung und -freigabe) im strikten Zweiphasen-Sperrprotokoll für den obigen Eingabeschedule S an.

Fortsetzung nächste Seite!

Teilaufgabe II: Softwaretechnik

Aufgabe 1: (Entwurfsmuster)

Gegeben sei folgender Sachverhalt: Eine Grafik ist entweder ein Kreis, ein Quadrat oder ein Dreieck. Eine Grafik kann zudem auch eine Kombination aus diesen Elementen sein. Des Weiteren können Sie aus mehreren Grafiken auch neue Grafiken zusammenbauen. Sie denken sich: Ich möchte eine Menge von Grafiken genauso wie eine einzelne Grafik behandeln können.

a) Welches Entwurfsmuster sollten Sie zur Modellierung verwenden?

b) Zeichnen Sie das entsprechende Klassendiagramm. Es reicht, nur die Klassennamen mit ihren Assoziationen und Vererbungsbeziehungen anzugeben; d.h. ohne Methoden und Attribute.

Hintergrundinformationen für die Aufgaben 2-5: Modellierung und Implementierung eines Programms

Ein autonomer Roboter in einer Montagehalle bekommt einen Auftrag, eine Menge von Objekten aus dem Lager (Material, z. B. Schrauben oder Werkzeug, z. B. Bohrer) zu holen und anschließend an seinen Ausgangspunkt zu bringen. Der Roboter hat einen Namen, der ihn eindeutig identifiziert, kennt seine aktuelle Position (x- und y-Koordinate) und hat als weitere Eigenschaft den Auftrag, der aus einer Liste von Auftragspositionen besteht, die er holen soll. Der Roboter besitzt folgende Fähigkeiten (Methoden):

- Er kann sich zu einer angegebenen Position bewegen.
- Er hat eine Methode, um einen Auftrag abzuarbeiten, d.h. alle Auftragspositionen zu holen.

Alle Objekte im Lager haben jeweils einen Namen, einen Standort mit Angabe einer Position (s. oben) und speichern außerdem die jeweils noch vorhandene Stückzahl. Es gibt zwei Typen von Objekten: Werkzeuge mit einer Methode, mit der ein einzelnes Werkzeug ausgeliehen werden kann, und Materialien mit einer Methode, mit der sie verbraucht werden, wobei eine gewünschte Stückzahl angegeben wird. Diese vorgegebenen Methoden aktualisieren die Stückzahlen der Werkzeuge (Reduktion um 1) bzw. Materialien (Reduktion um verbrauchte Stückzahl), wobei hier vereinfachend angenommen wird, dass die Stückzahlen der Werkzeuge und Materialien immer ausreichend groß sind, um die geforderten Mengen bedienen zu können (d.h. Sie können diese beiden Methoden nutzen, ohne deren Implementierung angeben zu müssen).

Ein Auftrag (z. B. "Hole Bohrer Typ B1, 100 x Schrauben M6, 10 x Schrauben M10 und 2x Blech B72") besteht aus einer Menge von Auftragspositionen. Eine Auftragsposition besteht aus dem Typ des zu holenden Objekts (Werkzeug oder Material, soll als Enumeration modelliert werden), einem Verweis auf das zu holende Objekt und der zu holenden Stückzahl (Quantität; bei Werkzeugen wird diese ignoriert, da sie immer 1 ist). Der Roboter soll über die Auftragsposition außerdem die Position bestimmen, zu der er fahren muss, um das Objekt zu holen.
Der Roboter arbeitet die Auftragspositionen in der Reihenfolge ab, indem er sich von seinem aktuellen Standpunkt immer zur am nächsten liegenden Auftragsposition bewegt, um dort das nächste Objekt zu holen. Wir gehen der Einfachheit halber davon aus, dass die Montagehalle gut aufgeräumt ist und der

Fortsetzung nächste Seite!

Roboter sich quasi entlang der Luftlinie bewegen kann, d.h. die Entfernung zwischen zwei Positionen entspricht der euklidischen Distanz (Wurzel aus der Summe der Quadrate der Differenzen der x- und y-Koordinaten der Positionen). Der Roboter soll zur Kontrolle als weitere Eigenschaft "Ergebnis" die Liste der eingesammelten Objekte zu einem Auftrag in der Reihenfolge speichern, in der er sie geholt hat, z. B. (M6 M10 B72 B1).

Aufgabe 2 (UML-Modellierung: Klassendiagramm)

Geben Sie ein UML-Klassendiagramm zu der Aufgabenstellung an.
Hinweis: Bei den Aufgaben 4 und 5 wird Konsistenz des Aktivitätsdiagramms bzw. des Codes mit dem Klassendiagramm verlangt.

Aufgabe 3 (Testgetriebene Entwicklung)

a) Erläutern Sie kurz, was man unter der Methode der testgetriebenen Entwicklung versteht.

b) Geben Sie für obige Aufgabenstellung (Abarbeitung der Aufträge durch den Roboter) einen Testfall für eine typische Situation an (d. h. das Einsammeln von 4 Objekten an unterschiedlichen Positionen). Spezifizieren Sie als Input alle für die Abarbeitung eines Auftrages relevanten Eingabe- und Klassen-Parameter sowie den vollständigen und korrekten Output.

Aufgabe 4 (Pseudocode und Aktivitätsdiagramm)

a) Geben Sie für alle Methoden, die zur Abarbeitung des Auftrages für den Roboter erforderlich sind, Pseudocode an. Nutzen Sie (im Klassendiagramm definierte) Hilfsmethoden, um den Code übersichtlicher zu gestalten. (Verwenden Sie statt einer komplizierten Methode mehrere einfachere Methoden, die sich aufrufen.)

   Achten Sie auf die Konsistenz zum Klassendiagramm (Inkonsistenzen führen zu Punktabzügen).

b) Überführen Sie den Pseudocode der Hauptmethode zur Abarbeitung eines Auftrags in ein syntaktisch korrektes UML-Aktivitätsdiagramm. Schleifen und Verzweigungen müssen durch strukturierte Knoten dargestellt werden (d. h. kein graphisches "goto").

Aufgabe 5 (Objektorientierte Programmierung)

Geben Sie korrekten Code für die Abarbeitung eines Auftrages durch den Roboter in einer objektorientierten Programmiersprache Ihrer Wahl an. Sie sollen nur den Code für die Methoden angeben. Sie brauchen keinen Code für Klassendefinitionen angeben, sondern können sich auf das UML-Klassendiagramm aus Aufgabe 2 beziehen.

Thema Nr. 2

Teilaufgabe I: Datenbanken

Aufgabe 1: (Allgemeine Fragen)

a) Das ACID-Prinzip fordert unter anderem die Atomarität und die Isolation einer Transaktion. Beschreiben Sie kurz die Bedeutung dieser Begriffe.
b) Was versteht man unter physischer Datenunabhängigkeit?
c) In welche drei Ebenen unterteilt die 3-Ebenen-Architektur Datenbanksysteme?
d) Definieren Sie, was ein Schlüssel ist.
e) Gegeben ist eine Relation $R(A, B, C)$, deren einziger Schlüsselkandidat A ist. Nennen Sie zwei Superschlüssel.
f) Gegeben seien 2 Relationen $R(A, B, C)$ und $S(C, D, E)$. Die Relation R enthalte 9 Tupel und die Relation S enthalte 7 Tupel. Sei p ein beliebiges Selektionsprädikat. Gegeben seien außerdem folgende Anfragen:

Q_1:

```
SELECT *
FROM R NATURAL JOIN S
WHERE p;
```

Q_2:

```
SELECT DISTINCT A
FROM R LEFT OUTER JOIN S ON R.C=S.C;
```

- Wieviele Ergebnistupel liefert die Anfrage Q_1 **mindestens**?
- Wieviele Ergebnistupel liefert die Anfrage Q_1 **höchstens**?
- Wieviele Ergebnistupel liefert die Anfrage Q_2 **mindestens**?
- Wieviele Ergebnistupel liefert die Anfrage Q_2 **höchstens**?

g) Erläutern Sie, was es bedeutet, wenn ein Attribut prim ist.

h) Nennen Sie die drei Armstrong-Axiome, die dazu dienen aus einer Menge von funktionalen Abhängigkeiten, die auf einer Relation gelten, weitere funktionale Abhängigkeiten abzuleiten.

i) Nennen Sie die höchste Normalform, der die Relation $R(A, B, C, D)$ mit den Abhängigkeiten $A \rightarrow C$ und $A, B \rightarrow D$ entspricht. Begründen Sie Ihre Antwort.

j) Gegeben seien die Transaktionen T_1, T_2 und T_3. Wie viele mögliche verschiedene **serialisierbare** Schedules gibt es *mindestens*? (Geben Sie Ihren Rechenweg an.)

Fortsetzung nächste Seite!

Aufgabe 2: (Relationales Modell)

Das Direktorat verliert bald den Überblick über die ganzen Smartphones, die die Lehrer von ihren Schülern während des Unterrichts konfiszieren mussten. Es wurde bei jedem Smartphone aufgeschrieben, welcher Lehrer es welchem Schüler an welchem Datum abgenommen hat. (Es kann angenommen werden, dass sich jeder Schüler nur einmal pro Tag erwischen lässt.) Um die Smartphones wieder dem jeweils richtigen Schüler zurückgeben zu können, soll deshalb eine Datenbank auf der Basis der folgenden Informationen erstellt werden. Die Schüler können durch die Kombination aus ihrem Nachnamen und Vornamen eindeutig identifiziert werden. Auch der Notenschnitt jeden Schülers ist im Lehrerkollegium bekannt.

Die Lehrer können anhand ihres Nachnamens eindeutig identifiziert werden und unterrichten jeweils drei Fächer.

Da der Informatiklehrer alles über die neuesten Trends wissen möchte, werden von den Smartphones jeweils die Modellnummer, sowie der RAM und die Bildschirmdiagonale aufgezeichnet. Um die Smartphones nicht zu verwechseln, wird jedem eine schulinterne ID zugewiesen. Auch welche Apps auf welchen Smartphones installiert sind, wird vermerkt. Es wird jeweils ihr eindeutiger Name, sowie die Bewertung im Store und ihr Preis notiert.

Erstellen Sie ein vollständiges Entity-Relationship-Diagramm für obige Datenbank. Vergessen Sie nicht, Schlüssel als solche durch Unterstreichen zu markieren und alle Funktionalitäten anzugeben.

Aufgabe 3: (Relationale Algebra)

Gegeben seien folgende Relationen:

X

A	B	C
1	2	3
2	3	4
4	3	3
1	2	2
2	3	2
3	3	2
3	1	2
2	2	1
1	1	1

Y

B	C	D
2	3	1
1	1	3
2	1	3
3	2	3
2	2	3
1	3	2

Fortsetzung nächste Seite!

Geben Sie die Ergebnisrelationen folgender Ausdrücke der relationalen Algebra als Tabellen an; machen Sie Ihren Rechenweg kenntlich.

- $\sigma_{A=2}(X) \bowtie Y$
- $(\pi_{B,C}(X) - \pi_{B,C}(Y)) \bowtie X$
- $\pi_A(X) \bowtie_{A>D} \sigma_{C=1}(Y)$
- $\pi_{A,C}(X) \div \pi_C(Y)$

Aufgabe 4: (Tupelkalkül und Bereichskalkül)

Gegeben sei das folgende Datenbankschema, welches für die Problemstellung aus Aufgabe 2 entwickelt wurde. Primärschlüssel sind unterstrichen, Fremdschlüssel sind überstrichen und der Text in den darauf folgenden eckigen Klammern benennt die Relation, auf die verwiesen wird.

Lehrer (Name, Fach1, Fach2, Fach3)
Schüler (Vorname, Nachname, Notenschnitt)
Smartphone (ID, Modellnr, RAM, Bildschirmdiagonale)
App (Name, Bewertung, Preis)
Eingesammelt ($\overline{\text{Vorname}}$ [Schüler], $\overline{\text{Nachname}}$ [Schüler], $\overline{\text{Name}}$ [Lehrer], $\overline{\text{ID}}$ [Smartphone], Datum)
Installiert ($\overline{\text{ID}}$ [Smartphone], Name [App])

Formulieren Sie die folgenden Anfragen a) bis c) im Tupelkalkül. Vergessen Sie dabei nicht, das Schema anzugeben. Formulieren Sie Aufgabe d) sowohl im Tupel- als auch im Bereichskalkül. Datumsvergleiche können Sie mit >, ≥, <, ≤ oder = angeben.

a) Geben Sie die Vornamen aller Schüler an, von denen Lehrer Lämpel am 16.10.2017 ein Smartphone eingesammelt hat.

b) Geben Sie die Namen aller Apps an, die auf den Smartphones waren, die vom Schüler Max Moritz eingesammelt wurden.

c) Geben Sie die Namen und alle Fächer der Lehrer an, die Smartphones mit einer Bildschirmdiagonale von unter 5,5" eingesammelt haben.

d) Sowohl im Tupel- ALS AUCH im Bereichskalkül: Geben Sie Namen und Hauptfach (also Fach1) der Lehrer an, die Smartphones von Schülern eingesammelt haben, die einen schlechteren Notenschnitt als 3,0 haben.

Fortsetzung nächste Seite!

Aufgabe 5: (Anfragen)

Wir betrachten erneut das gegebene Schema aus Aufgabe 4. Primärschlüssel sind unterstrichen, Fremdschlüssel sind überstrichen und der Text in den darauf folgenden eckigen Klammern benennt die Relation, auf die verwiesen wird.

Der Notenschnitt, der Preis und die Bewertung werden als Kommazahl dargestellt, wobei die Bewertung die Anzahl der Sterne angibt, also maximal 5 und mindestens 0. Die Modellnummer kann sowohl aus Zahlen und Buchstaben bestehen, ist jedoch nie länger als 50 Zeichen. ID, RAM, Bildschirmdiagonale und Datum sind ganze Zahlen. Die restlichen Attribute sind Strings der Länge 15.

Lehrer (Name, Fach1, Fach2, Fach3)
Schüler (Vorname, Nachname, Notenschnitt)
Smartphone (ID, Modellnr, RAM, Bildschirmdiagonale)
App (Name, Bewertung, Preis)
Eingesammelt (Vorname [Schüler], Nachname [Schüler], Name [Lehrer], ID [Smartphone], Datum)
Installiert (ID [Smartphone], Name [App])

a) Geben Sie die Anweisung in **SQL-DDL** an, die notwendig ist, um die Relation 'App' zu erzeugen.

b) Geben Sie die Anweisung in **SQL-DDL** an, die notwendig ist, um die Relation 'Installiert' zu erzeugen.

c) Formulieren Sie die folgende Anfrage in **SQL**:

Gesucht sind die Namen der Apps zusammen mit ihrer Bewertung, die auf den Smartphones installiert sind, die Lehrer Keating eingesammelt hat. Sortieren Sie das Ergebnis nach Bewertung absteigend.

Hinweis: Achten Sie auf gleichnamige Attribute.

d) Formulieren Sie die folgende Anfrage in **SQL**:

Gesucht ist der durchschnittliche Notenschnitt der Schüler, denen ein iPhone 6s abgenommen wurde. Ein iPhone 6s kann A1633 als Modellnummer haben oder A1688.

e) Formulieren Sie die folgende Anfrage in **SQL**:
Gesucht ist die Modellnummer der Smartphones, die durchschnittlich die meisten Apps installiert haben.

Tipp: Die Verwendung von Views kann die Aufgabe vereinfachen.

Fortsetzung nächste Seite!

Aufgabe 6: (Normalisierung)

Gegeben sei das Relationsschema $R(A, B, C, D, E, F)$, sowie die Menge der zugehörigen funktionalen Abhängigkeiten FD:

- $B \rightarrow F$
- $CD \rightarrow E$
- $C \rightarrow A$
- $CD \rightarrow A$
- $D \rightarrow F$
- $D \rightarrow B$

a) Bestimmen Sie den Schlüsselkandidaten der Relation R und begründen Sie, warum es keine weiteren Schlüsselkandidaten gibt.

b) Überführen Sie das Relationsschema R mit Hilfe des Synthesealgorithmus in die dritte Normalform. Führen Sie hierfür jeden der vier Schritte durch und kennzeichnen Sie Stellen, bei denen nichts zu tun ist. Benennen Sie alle Schritte und begründen Sie eventuelle Reduktionen.

Fortsetzung nächste Seite!

Teilaufgabe II:

Aufgabe 1: (Test-getriebene Entwicklung)

Bewerten Sie die folgenden Aussagen und nehmen Sie jeweils Stellung.

a) Tests zuerst

In test-getriebener Softwareentwicklung wird der Test vor der zu testenden Funktion programmiert.

b) Komponententests

Komponententests sind immer White-Box-Tests und nie Black-Box-Tests.

c) Akzeptanztests

Akzeptanz- resp. Funktionstests sind immer Black-Box-Tests und nie White-Box-Tests.

d) Fehler

Ein fehlgeschlagener Test und ein Testausführungsfehler bezeichnen denselben Sachverhalt.

e) Test Suiten

Tests können hierarchisch strukturiert werden, so dass mit einem Befehl das gesamte zu testende System getestet werden kann.

Aufgabe 2: (Objektorientierte Modellierung)

Eine Pizzeria möchte mit dem Webdienst PizzaNow anbieten, Pizzen online bestellen zu können. Es gilt die folgende Beschreibung:

„Ein Kunde kann mehrere Pizzen in PizzaNow zu einer Bestellung zusammenstellen. Nachdem er die Bestellung abgeschickt hat, wird die Pizzeria über diese Bestellung informiert. Dort wird die Bestellung bestätigt und der Kunde erhält eine Bestätigung mit Zahlungsinformationen. Diese müssen vom jeweiligen Kreditinstitut überprüft und bestätigt werden. Danach erhält der Kunde eine Bestätigung über den erfolgreichen Bezahlvorgang und die Pizzeria beginnt nach der Bestätigung der Bezahlung die Herstellung und im Anschluss die Auslieferung der Bestellung."

Modellieren Sie den dargestellten Prozess mit Hilfe eines UML-Sequenzdiagramms.

Fortsetzung nächste Seite!

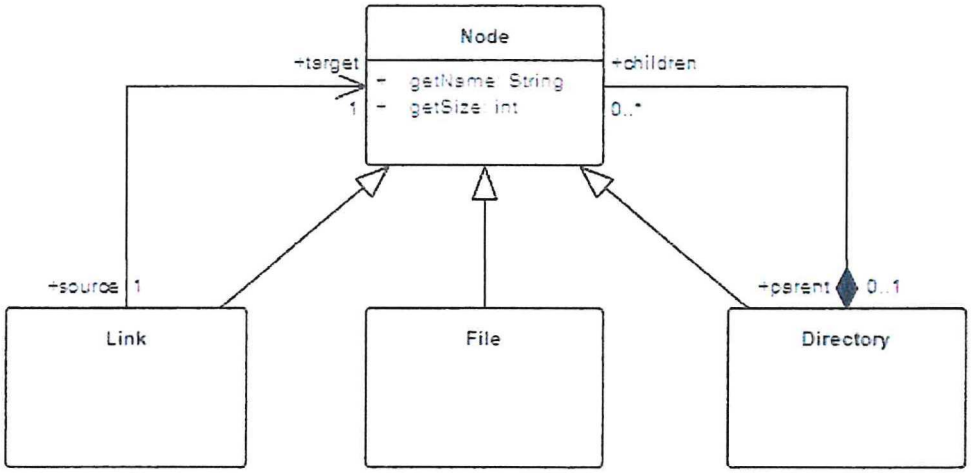
Aufgabe 3: (Objektorientierte Modellierung)

a) Nehmen Sie an, dass in einem System zur Durchführung von Umfragen die drei Anwendungsfälle „Teilnahme“, „Fragen beantworten“ und „Registrierung für Verlosung“ gegeben seien. Alle Anwendungsfälle sind einem Teilnehmer (an einer Umfrage) zugänglich. Stellen Sie die drei Anwendungsfälle in einem UML-Anwendungsfalldiagramm („use case diagram“) dar. Verwenden Sie dabei sowohl die <<extend>> - als auch die <<include>> -Beziehung und beschreiben Sie die Bedeutung der Beziehungen für die Existenz des „Teilnahme“-Anwendungsfalls.

b) Welche Informationen sollte eine typische Spezifikation eines Anwendungsfalls enthalten?

Aufgabe 4: (Objektorientierte Modellierung)

Das im Folgenden dargestellte Diagramm ist das von Aufgaben 4a)-e) referenzierte Diagramm. Das Diagramm stellt ein Klassenmodell eines Dateisystems dar.



a) Entwurfsmuster

Benennen Sie alle Anwendungen von Entwurfsmustern („design patterns“), die explizit im Diagramm erkennbar sind. (Falschmeldungen zählen negativ.)

b) Verweise

Ein Link-Objekt verweist mittels target-Referenz auf weitere Objekte. Benennen Sie die Klassen aller möglichen Objekte, die gemäß Diagramm referenziert werden können.

Fortsetzung nächste Seite!

c) Verzeichnisbaum

Benennen Sie die minimale Tiefe sowie die maximale Tiefe des durch Directory-Objekte aufgespannten Verzeichnisbaums.

d) Erweiterung

Wie würden Sie mit diesem Entwurf vereinbar eine Klasse Gerätetreiber einführen, welche externe Dateisysteme in den Verzeichnisbaum einbindet? Begründen Sie Ihre Entscheidung.

e) UML-Collaboration

Nehmen Sie an, dass Sie diesen Entwurf um eine UML-Collaboration zur Modellierung von Eltern-Kind-Beziehungen erweitern. Benennen Sie die Klasse, an welche Sie die Kind-Rolle binden würden, und begründen Sie Ihre Entscheidung.

Herbst 2019

Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer
Kennzahl: _____	**Herbst 2019**	**66116**
Kennwort: _____		
Arbeitsplatz-Nr.: _____		

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!

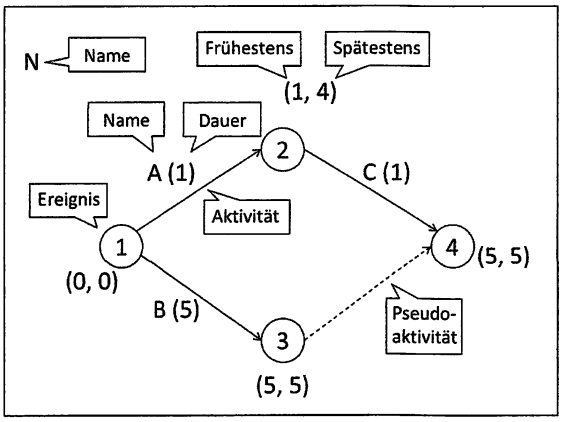
Thema Nr. 1

Teilaufgabe I: Softwaretechnologie

Aufgabe 1: (Klassen- und Objektdiagramme) (30 Punkte)

Ein CPM-Netzwerk („Critical Path Method") ist ein benannter Projektplan, der aus Ereignissen und Aktivitäten besteht. Ein Ereignis wird durch eine ganze Zahl > 0 identifiziert. Jede Aktivität führt von einem Quellereignis zu einem Zielereignis. Eine reale Aktivität hat einen Namen und eine Dauer (eine ganze Zahl > 0). Eine Pseudoaktivität ist anonym. Ereignisse und Pseudoaktivitäten verbrauchen keine Zeit. Zu jedem Ereignis gibt es einen frühesten und einen spätesten Zeitpunkt (eine ganze Zahl ≥ 0), deren Berechnung nicht Gegenstand der Aufgabe ist.

Beispiel:



a) Erstellen Sie ein UML-Klassendiagramm zur Modellierung von CPM-Netzwerken. Geben Sie für Attribute jeweils den Namen und den Typ an. Geben Sie für Assoziationen den Namen und für jedes Ende den Rollennamen und die Multiplizität an. Nutzen Sie ggf. abstrakte Klassen, Vererbung, Komposition oder Aggregation. Verzichten Sie auf Operationen und Sichtbarkeiten.

b) Erstellen Sie für das Klassendiagramm aus a) und das Beispiel aus der Aufgabenstellung ein Objektdiagramm. Geben Sie Rollennamen nur an, wenn es notwendig ist, um die Enden eines Links (Instanz einer Assoziation) zu unterscheiden.

Fortsetzung nächste Seite!

Aufgabe 2: (Anwendungsfall- und Aktivitätsdiagramme) (30 Punkte)

Eine Dienstreise an einer Universität wird folgendermaßen abgewickelt: Ein Mitarbeiter erstellt einen Dienstreiseantrag und legt ihn seinem Vorgesetzten zur Unterschrift vor. Mit seiner Unterschrift befürwortet der Vorgesetzte die Dienstreise. Verweigert er die Unterschrift, wird der Vorgang abgebrochen. Nach der Befürwortung wird der Antrag an die Reisekostenstelle weitergeleitet, die über die Annahme des Antrags entscheidet. Im Falle einer Ablehnung wird der Vorgang abgebrochen; sonst genehmigt die Reisekostenstelle den Antrag. Der Mitarbeiter kann nun einen Abschlag beantragen. Stimmt der Vorgesetzte zu, so entscheidet die Reisekostenstelle über den Abschlag und weist ggf. die Kasse der Universität an, den Abschlag auszuzahlen. Nach Ende der Dienstreise erstellt der Mitarbeiter einen Antrag auf Erstattung der Reisekosten an die Reisekostenstelle. Die Reisekostenstelle setzt den Erstattungsbetrag fest und weist die Kasse an, den Betrag auszuzahlen.

a) Erstellen Sie ein Anwendungsfalldiagramm (Use-Case-Diagramm) für Dienstreisen.

b) Erstellen Sie ein Aktivitätsdiagramm, das den oben beschriebenen Ablauf modelliert. Ordnen Sie den Aktionen die Akteure gemäß a) zu. Beschränken Sie sich auf den Kontrollfluss (keine Objektflüsse).

Fortsetzung nächste Seite!

Aufgabe 3: (White-Box-Tests) (30 Punkte)

Eine Dezimalzahl hat ein optionales Vorzeichen, dem eine nichtleere Sequenz von Dezimalziffern folgt. Der anschließende gebrochene Anteil ist optional und besteht aus einem Dezimalpunkt, gefolgt von einer nichtleeren Sequenz von Dezimalziffern.

Die folgende Java-Methode erkennt, ob eine Zeichenfolge eine Dezimalzahl ist:

```
 1  public static boolean istDezimalzahl(char[] zeichen){
 2      boolean resultat; int laenge = zeichen.length;
 3      if (laenge == 0)
 4          resultat = false;
 5      else {
 6          int i = 0;
 7          if (zeichen[i] == '+' || zeichen[i] == '-')
 8              i++;
 9          int j = i;
10          while (i < laenge && '0' <= zeichen[i] && zeichen[i] <= '9')
11              i++;
12          if (i == j)
13              resultat = false;
14          else {
15              if (i < laenge && zeichen[i] == '.')
16                  do
17                      i++;
18                  while (i < laenge && '0' <= zeichen[i] &&
                           zeichen[i] <= '9');
19              resultat = i == laenge && '0' <= zeichen[i-1] &&
                           zeichen[i-1] <= '9';
20          }
21      }
22      return resultat;
    }
```

a) Konstruieren Sie zu dieser Methode einen Kontrollflußgraphen. Markieren Sie dessen Knoten mit Zeilennummern des Quelltexts.

b) Geben Sie eine minimale Testmenge an, die das Kriterium der Knotenüberdeckung erfüllt. Geben Sie für jeden Testfall den durchlaufenen Pfad in der Notation 1 -> 2 -> ... an.

c) Verfahren Sie wie in b) für das Kriterium der Kantenüberdeckung.

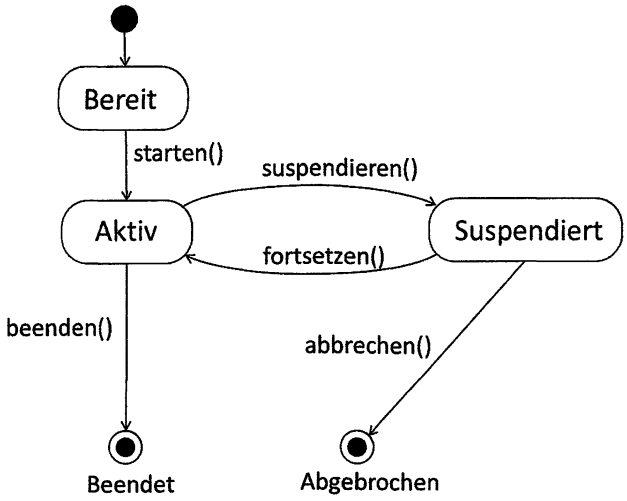
d) Wie stehen die Kriterien der Knoten- und Kantenüberdeckung zueinander in Beziehung? Begründen Sie Ihre Anwort.

Hinweis: Eine Testmenge ist minimal, wenn es keine andere Testmenge mit einer kleineren Zahl von Testfällen gibt. Die Minimalität braucht nicht bewiesen zu werden.

Fortsetzung nächste Seite!

Aufgabe 4: (Entwurfsmuster) (30 Punkte)

Zu den Aufgaben eines Betriebssystems zählt die Verwaltung von Prozessen. Jeder Prozess durchläuft verschiedene Zustände; Transitionen werden durch Operationsaufrufe ausgelöst. Folgendes Zustandsdiagramm beschreibt die Verwaltung von Prozessen:



Implementieren Sie dieses Zustandsdiagramm in einer Programmiersprache Ihrer Wahl mit Hilfe des Zustandsmusters; geben Sie die gewählte Sprache an. Die Methoden für die Transitionen sollen dabei die Funktionalität der Prozessverwaltung simulieren, indem der Methodenaufruf auf der Standardausgabe protokolliert wird. Falls Transitionen im aktuellen Zustand undefiniert sind, soll eine Fehlermeldung ausgegeben werden.

Teilaufgabe II: Datenbanken

Aufgabe 1: (Wissensfragen) (9 Punkte)

Antworten Sie kurz und prägnant.

1. [1 Punkt] Nennen Sie einen Vorteil und einen Nachteil der Schichtenarchitektur.

2. [1 Punkt] Wie ermöglicht es ein Datenbankensystem, verschiedene Sichten darzustellen?

3. [3 Punkte] Was beschreibt das Konzept der Transitiven Hülle? Erklären Sie dies kurz und nennen Sie ein Beispiel für (1) die Transitive Hülle eines Attributes bei funktionalen Abhängigkeiten und (2) die Transitive Hülle einer SQL Anfrage.

4. [4 Punkte] Nennen Sie zwei Indexstrukturen und beschreiben Sie jeweils ihren Vorteil.

Fortsetzung nächste Seite!

Aufgabe 2: (ER-Modellierung) (40 Punkte)

Entwerfen Sie ein ER-Diagramm für eine Schule aus einer imaginären Film-Reihe. Geben Sie alle Attribute an und unterstreichen Sie Schlüsselattribute. Für die Angabe der Kardinalitäten von Beziehungen soll die Min-Max-Notation verwendet werden. Führen Sie wenn nötig einen Surrogatschlüssel ein.

An der Schule werden Schüler ausgebildet. Sie haben einen Namen, ein Geschlecht und ein Alter. Da die Schule klein ist, ist der Name eindeutig. Jeder Schüler ist Teil seines Jahrgangs, bestimmt durch Jahr und Anzahl an Schüler (Jahrgänge ohne Schüler sind erlaubt), und besucht mit diesem Kurse. Dabei wird jeder Kurs von min. einem Jahrgang besucht und jeder Jahrgang hat zwischen 2 und 5 Kurse. Kurse haben einen Veranstaltungsort und einen Namen.
Außerdem wird jeder Schüler einem von vier Häusern zugeordnet. Diese Häuser sind Gryffindor, Slytherin, Hufflepuff und Ravenclaw. Jedes Haus hat eine Anzahl an Mitgliedern.

Um die Organisation an der Schule zu erleichtern, gibt es pro Haus einen Vertrauensschüler und pro Jahrgang einen Jahrgangssprecher. Außerdem können Schüler Quidditch spielen. Dabei können sie die Rollen Sucher, Treiber, Jäger und Hüter spielen. Jedes Haus der Schule hat eine Mannschaft. Diese besteht aus genau einem Sucher, einem Hüter, drei Jäger und zwei Treiber. Jedes Jahr gibt es an der Schule eine Trophäe zu gewinnen. Diese ist abhängig von der Mannschaft und weiterhin durch das Jahr identifiziert.

Fortsetzung nächste Seite!

Aufgabe 3: (SQL) (30 Punkte)

Formulieren Sie folgende Anfragen in SQL gegen die angegebene Datenbank aus einer imaginären Serie.

Figur = {Id, Name, Schwertkunst, Lebendig, Titel}
gehört_zu = {Id, Familie,
FK (Id) references Figur(Id),
FK (Familie) references Familie(Id) }
Familie = {Id, Name, Reichtum, Anführer}
Drache = {Name, Lebendig}
besitzt = {Id, Name,
FK (Id) references Figur(Id),
FK (Name) references Drache(Name) }
Festung = {Name, Ort, Ruine}
besetzt = {Familie, Festung,
FK (Familie) references Familie(Id),
FK (Festung) references Festung(Name) }
lebt = {Id, Name,
FK (Id) references Figur(Id),
FK (Name) references Festung(Name) }

1. [2 Punkte] Geben Sie für alle Figuren an, wie oft alle vorhandenen Titel vorkommen.

2. [3 Punkte] Welche Figuren (Name ist gesucht) kommen aus "Kings Landing"?

3. [5 Punkte] Geben Sie für jede Familie (Name) die Anzahl der zugehörigen Charaktere und Festungen an.

4. [11 Punkte] Gesucht sind die besten fünf Schwertkämpfer aus Festungen aus dem Ort "Westeros". Es soll der Name, die Schwertkunst und die Platzierung ausgegeben werden. Die Ausgabe soll nach der Platzierung sortiert erfolgen.

5. [2 Punkte] Schreiben Sie eine Anfrage, die alle Figuren löscht, die tot sind. Das Attribut Lebendig kann dabei die Optionen "ja" und "nein" annehmen.

6. [2 Punkte] Löschen Sie die Spalten "Lebendig" aus der Datenbank.

7. [5 Punkte] Erstellen Sie eine weitere Tabelle mit dem Namen Waffen, welche genau diese auflistet. Eine Waffe ist genau einer Figur zugeordnet, hat einen eindeutigen Namen und eine Stärke zwischen 0 und 5. Wählen Sie sinnvolle Typen für die Attribute.

Fortsetzung nächste Seite!

Aufgabe 4: (Relationale Algebra) (14 Punkte)

Übertragen Sie die folgenden Ausdrücke in die relationale Algebra. Beschreiben Sie diese Ausdrücke umgangssprachlich, bevor Sie die Umformung durchführen. Das Schema der Datenbank entspricht dem Schema aus Aufgabe 3.

1. [6 Punkte] $\{f \mid f \in \text{Figur} \wedge \neg\exists g \in \text{gehört_zu}(f.\text{Id} = g.\text{Id})\}$

2. [8 Punkte] $\{f \mid f \in \text{Figur} \wedge \exists l \in \text{lebt}(l.\text{Id} = c.\text{Id}) \wedge \exists f \in \text{Festung}(l.\text{Festung} = f.\text{Name}) \wedge$
$\exists b \in \text{besetzt}(f.\text{Name} = b.\text{Festung}) \wedge \exists f2 \in \text{Familie}(b.\text{Familie} = f2.\text{Id}) \wedge f2.\text{Name} = \text{Stark})\}$

Fortsetzung nächste Seite!

Aufgabe 5: (Entwurfstheorie) (27 Punkte)

Gegeben sind folgende funktionale Abhängigkeiten.

$$X \rightarrow YZ$$
$$Z \rightarrow WX$$
$$Q \rightarrow XYZ$$
$$V \rightarrow ZW$$
$$ZW \rightarrow YQV$$

1. [10 Punkte] Berechnen Sie die kanonische Überdeckung.

Gegeben ist folgende Tabelle.

JedID	Name	Rasse	Lichtschwert	Seite der Macht
2	Yoda	Unbekannt	Grün	Gute Seite
3	Anakin Skywalker	Mensch	Blau, Rot	Gute Seite, dunkle Seite
4	Mace Windou	Mensch	Lila	Gute Seite
5	Count Dooku	Mensch	Rot	Dunkle Seite
6	Ahsoka Tano	Togruta	Grün	Gute Seite
7	Yoda	Mensch	Rot	Dunkle Seite

2. [2 Punkte] Geben Sie zuerst die funktionalen Abhängigkeiten in der Tabelle an.

3. a) [1 Punkt] Geben Sie die zentrale Eigenschaft der 1. NF an.

   b) [3 Punkte] Nennen Sie alle Stellen, an denen das Schema die 1. NF verletzt.

   c) [3 Punkte] Überführen Sie die Tabelle in die 1. NF.

4. a) [1 Punkt] Geben Sie die Definition der 2. NF an.

   b) [2 Punkte] Arbeiten Sie bitte mit folgender, nicht korrekten Zwischenlösung weiter. Erläutern Sie, inwiefern dieses Schema die 2. NF verletzt.

JedID	Name	Rasse	Lichtschwert	Seite der Macht
2	Yoda	Unbekannt	Grün	Gute Seite
3	Skywalker	Mensch	Blau	Gute Seite
4	Windou	Mensch	Lila	Gute Seite
5	Dooku	Mensch	Rot	Dunkle Seite
6	Tano	Togruta	Grün	Gute Seite
2	Yoda	Mensch	Rot	Dunkle Seite

   c) [3 Punkte] Überführen Sie die Tabelle in die 2. NF.

5. a) [1 Punkt] Geben Sie die Definition der 3. NF an.

   b) [1 Punkt] Erläutern Sie, ob und wenn ja, wie das von Ihnen in 3c) neu erstellte Schema die 3. NF verletzt.

Thema Nr. 2

Teilaufgabe I: Softwaretechnologie

Aufgabe 1: (Modellierung und Muster) (75 Punkte)

Hierarchische Dateisysteme bestehen aus den FileSystemElements Ordner, Dateien und Verweise. Ein Ordner kann seinerseits Ordner, Dateien und Verweise beinhalten; jedem Ordner ist bekannt, welche Elemente (children) er enthält. Mit Ausnahme des Root-Ordners auf der obersten Hierarchieebene ist jeder Ordner, jede Datei und jeder Verweis Element eines Elternordners. Jedem Element ist bekannt, was sein Elternordner ist (parent). Ein Verweis verweist auf einen Verweis, eine Datei oder einen Ordner (link). Wenn ein Ordner gelöscht wird, werden alle seine Bestandteile ggf. rekursiv ebenfalls gelöscht. Sie dürfen die Lösungen für Aufgabenteil b) und c) in einem gemeinsamen Code kombinieren.

a) [23 Punkte] Modellieren Sie diesen Sachverhalt mit einem UML-Klassendiagramm. Benennen Sie die Rollen von Assoziationen und geben Sie alle Kardinalitäten an. Ihre Lösung soll mindestens eine sinnvolle Spezialisierungsbeziehung enthalten.

b) [20 Punkte] Implementieren Sie das Klassendiagramm als Java- oder C++-Programm. Jedes Element des Dateisystems soll mindestens über ein Attribut *name* verfügen. Übergeben Sie den Elternordner jedes Elements als Parameter an den Konstruktor; der Elternordner des Root-Ordners kann dabei als *null* implementiert werden. Dokumentieren Sie Ihren Code.

c) [17 Punkte] Ordnen Sie eine Methode *delete*, die Dateien, Ordner und Verweise rekursiv löscht, einer oder mehreren geeigneten Klassen zu und implementieren Sie sie. Zeigen Sie die Löschung jedes Elements durch eine Textausgabe von *name* an. *toString()* müssen Sie dabei nicht implementieren. Gehen Sie davon aus, dass Verweis- und Ordnerstrukturen azyklisch sind und dass jedes Element des Dateisystems höchstens einmal existiert. Wenn ein Verweis gelöscht wird, wird sowohl der Verweis als auch das verwiesene Element bzw. transitiv die Kette der verwiesenen Elemente gelöscht. Bedenken Sie, dass die Löschung eines Elements immer auch Konsequenzen für den dieses Element beinhaltenden Ordner hat. Es gibt keinen Punktabzug, wenn Sie die Löschung des Root-Ordners nicht zulassen.

d) [6 Punkte] Was kann im Fall von *delete* passieren, wenn die Linkstruktur zyklisch ist oder die Ordnerstruktur zyklisch ist? Kann es zu diesen Problemen auch dann führen, wenn weder die Linkstruktur zyklisch ist, noch die Ordnerstruktur zyklisch ist? Wie kann man im Programm das Problem lösen, falls man Zyklizitäten zulassen möchte?

e) [9 Punkte] Was ist ein Design Pattern? Nennen Sie drei Beispiele und erläutern Sie sie kurz. Welches Design Pattern bietet sich für die Behandlung von hierarchischen Teil-Ganzes-Beziehungen an, wie sie im Beispiel des Dateisystems vorliegen?

Fortsetzung nächste Seite!

Aufgabe 2: (Assertions) (17 Punkte)

Methoden in Programmen funktionieren nicht immer für alle möglichen Eingaben – klassische Beispiele sind die Quadratwurzel einer negativen Zahl oder die Division durch Null. Zulässige (bzw. in negierter Form unzulässige) Eingabewerte sollten dann spezifiziert werden, was mit Hilfe sog. assertions geschehen kann. Assertions prüfen zur Laufzeit den Wertebereich der Parameter einer Methode, bevor der Methodenkörper ausgeführt wird. Wenn ein oder mehrere Argumente zur Laufzeit unzulässig sind, wird eine Ausnahme geworfen.

Betrachten Sie das folgende Java-Programm:

```
public static double[][] magic(double[][] A, double[][] B, int m) {
      double[][] C = new double[m][m];
      for (int i = 0; i < m; i++)
            for (int j = 0; j < m; j++)
                  for (int k = 0; k < m; k++)
                        C[i][j] += A[i][k] * B[k][j];
      return C;
}
```

a) [2 Punkte] Beschreiben Sie kurz, was dieses Programm tut.

b) [8 Punkte] Implementieren Sie drei nützliche Assertions, die zusammen verhindern, dass das Programm abstürzt.

c) [4 Punkte] Wann und warum kann es sinnvoll sein, keine expliziten Assertions anzugeben? Erläutern Sie, warum das potentiell gefährlich ist.

d) [3 Punkte] Assertions können wie oben sowohl für Vorbedingungen am Anfang einer Methode als auch für Nachbedingungen am Ende einer Methode verwendet werden. Solche Spezifikationen bilden dann sog. Kontrakte. Kontrakte werden insbesondere auch statisch verwendet, d. h. nicht zur Laufzeit. Skizzieren Sie ein sinnvolles Anwendungsgebiet und beschreiben Sie kurz den Vorteil der Verwendung von Kontrakten in diesem Anwendungsgebiet.

Fortsetzung nächste Seite!

Aufgabe 3: (Softwarearchitektur und Agilität) (28 Punkte)

Die Komponentenarchitektur eines Softwaresystems beschreibt die unterschiedlichen Softwarebausteine, deren Schnittstellen und die Abhängigkeiten von Softwarekomponenten wie beispielsweise Klassen. Wir unterscheiden zwischen der Soll- und der Ist-Architektur eines Systems.

a) [6 Punkte] Nennen und definieren Sie drei Qualitätsattribute von Software, die durch die Architektur beeinflusst werden und charakterisieren Sie jeweils eine „schlechte" Architektur, die diese Qualitätsattribute negativ beeinflusst.

b) [5 Punkte] Erläutern Sie, was Information Hiding ist. Wie hängt Information Hiding mit Softwarearchitektur zusammen? Wie wird Information Hiding auf Klassenebene in Java implementiert? Gibt es Situationen, in denen Information Hiding auch negative Effekte haben kann?

c) [2 Punkte] Erklären Sie, was Refactoring ist.

d) [15 Punkte] Skizzieren Sie die Kernideen von Scrum inkl. der wesentlichen Prozessschritte, Artefakte und Rollen. Beschreiben Sie dann die Rolle von Ist- und Soll-Architektur in agilen Entwicklungskontexten wie Scrum.

Fortsetzung nächste Seite!

Teilaufgabe II: Datenbanken

Aufgabe 1: (ER-Modellierung) (35 Punkte)

Erstellen Sie ein möglichst einfaches ER-Schema, das alle gegebenen Informationen enthält. Attribute von Entitäten und Beziehungen sind anzugeben, Schlüsselattribute durch Unterstreichen zu kennzeichnen. Verwenden Sie für die Angabe der Kardinalitäten von Beziehungen die Min-Max-Notation. Führen Sie Surrogatschlüssel nur dann ein, wenn es nötig ist und modellieren Sie nur die im Text vorkommenden Elemente.

Ein örtlicher Sportverein möchte seine Vereinsangelegenheiten mittels einer Datenbank verwalten. Der Verein besteht aus verschiedenen Abteilungen, welche eine eindeutige Nummer und einen aussagekräftigen Namen besitzen. Für jede Abteilung soll zudem automatisch die Anzahl der Mitglieder gespeichert werden, wobei ein Mitglied zu mehreren Abteilungen gehören kann. Die Mitglieder des Vereins können keine, eine oder mehrere Rollen (auch Ämter genannt) einnehmen. So gibt es die Ämter: 1. Vorstand, 2. Vorstand, Kassier, Jugendleiter, Trainer sowie einen Abteilungsleiter für jede Abteilung. Es ist dabei auch möglich, dass ein Abteilungsleiter mehrere Abteilungen leitet oder ein Mitglied mehrere Aufgaben übernimmt, mit der Einschränkung, dass die Vorstandsposten und Kassier nicht von der gleichen Person ausgeübt werden dürfen. Zu jedem Trainer wird eine Liste von Lizenzen gespeichert. Jeder Trainer ist zudem in mindestens einer Abteilung eine bestimmte Anzahl von Stunden tätig. Zu allen Mitgliedern werden Mitgliedsnummer, Name (bestehend aus Vor- und Nachname), Geburtsdatum, E-Mail, Eintrittsdatum, Adresse (bestehend aus PLZ, Ort, Straße, Hausnummer), IBAN und die Vereinszugehörigkeit in Jahren gespeichert.

Im Verein fallen Finanztransaktionen an. Zu jeder Transaktion wird ein Zeitstempel, der Betrag und eine eindeutige Transaktionsnummer gespeichert. Die Mitglieder leisten Zahlungen an den Verein. Umgekehrt erstattet der Verein auch bestimmte Kosten. Im Verein existieren drei verschiedene Mitgliedsbeiträge. So gibt es einen Kinder-und-Jugendlichen-Tarif, einen Erwachsenentarif und einen Familientarif.

Mitgliedern entstehen des Öfteren Fahrtkosten. Für jede Fahrtkostenabrechnung werden das Datum, die gefahrenen Kilometer und Start und Ziel, der Zweck sowie das Mitglied gespeichert, welches den Antrag gestellt hat. Zu jeder Fahrtkostenabrechnung existiert genau eine Erstattung.

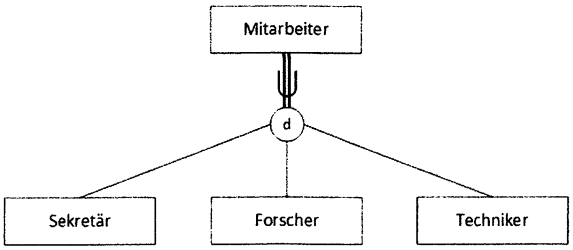
Durch die Teilnahme an verschiedenen Wettbewerben besteht die Notwendigkeit die von einem Team (also mehreren Mitgliedern zusammen) oder Mitgliedern erzielten sportlichen Erfolge, d. h. Platzierungen, zu verwalten. Jeder Wettkampf besitzt eine eindeutige ID, ein Datum und eine Kurzbeschreibung.

Das Vereinsleben besteht aus zahlreichen Terminen, die durch Datum und Uhrzeit innerhalb einer Abteilung eindeutig identifiziert werden können. Zu jedem Termin wird zusätzlich eine Kurzbeschreibung gespeichert.

Fortsetzung nächste Seite!

Aufgabe 2: (ER-Modellierung) (4 Punkte)

In einer Datenbank zur Mitarbeiterverwaltung werden die Mitarbeiter über ihr Ausscheiden aus dem Betrieb hinaus (z. B. Ruhestand oder Arbeitsplatzwechsel) gespeichert. Im Folgenden ist ein Ausschnitt aus dem ER-Diagramm dargestellt. Erweitern Sie das Diagramm um genau eine Entität, welche die derzeit aktiven Mitarbeiter aus allen Unterklassen umfasst. Benennen und erläutern Sie das von Ihnen verwendete Modellierungskonstrukt.



Aufgabe 3: (Relationale Algebra) (6 Punkte)

Gegeben seien die folgenden beiden Relationen:

R1	P	Q	S
	10	a	5
	15	b	8
	25	a	6

R2	A	B	C
	10	b	6
	25	c	3
	10	b	5

Geben Sie die Ergebnisse der folgenden relationalen Ausdrücke an:

a) [1,5 Punkte] $R_1 \bowtie_{R1.P=R2.A} R_2$ (Equi-Join)

b) [1,5 Punkte] $R_1 ⟕_{R1.Q=R2.B} R_2$ (Right-Outer-Join)

c) [3 Punkte] Es ist bekannt, dass die minimale Menge relationaler Operatoren Selektion, Projektion, Vereinigung, Differenz und kartesisches Produkt umfasst. Wie kann die Division zweier Relationen mit diesen Operatoren ausgedrückt werden? Begründen Sie kurz die einzelnen Bestandteile Ihres relationalen Ausdrucks.

Fortsetzung nächste Seite!

Aufgabe 4: (Normalisierung) (22 Punkte)

Gegeben sei das Relationenschema `R (A, B, C, D, E, F)` sowie die Menge der zugehörigen funktionalen Abhängigkeiten `FD`:

```
A B    -> C
A      -> D
F      -> B
D E    -> B
B      -> A
```

a) [5 Punkte] Bestimmen Sie sämtliche Schlüsselkandidaten der Relation `R` und begründen Sie, warum es keine weiteren Schlüsselkandidaten geben kann.

b) [8 Punkte] Ist die gegebene Menge an funktionalen Abhängigkeiten minimal? Fall sie minimal ist begründen Sie diese Eigenschaft ausführlich, anderenfalls minimieren Sie `FD` schrittweise. Vergessen Sie nicht die einzelnen Schritte entsprechend zu begründen.

c) [9 Punkte] Überführen Sie falls nötig das Schema in dritte Normalform. Ist die dritte Normalform bereits erfüllt, begründen Sie dies ausführlich.

Aufgabe 5: (Anfrageoptimierung) (10 Punkte)

Gegeben seien die beiden Relationen *Professor* und *Vorlesung* mit folgendem Umfang:

Relation Professor: 5 Spalten, 164 Datensätze
Relation Vorlesung: 10 Spalten, 333 Datensätze.

Optimieren Sie auf geeignete Weise folgende SQL-Anweisung möglichst gut und berechnen Sie wie stark sich die Datenmenge durch jede Optimierung reduziert (nehmen Sie für Ihre Berechnung an, dass Herr Mustermann genau zwei Vorlesungen hält). Geben Sie jeweils den Operatorbaum vor und nach Ihren jeweiligen Optimierungen an.

```
SELECT Titel FROM Professor, Vorlesung WHERE Name = ´Mustermann` AND PersNr =
gelesenVon;
```

Fortsetzung nächste Seite!

Aufgabe 6: (Vermischte Fragen) (25 Punkte)

Begründen oder erläutern Sie Ihre Antworten.

a) [2 Punkte] Erklären Sie kurz den Unterschied zwischen einem Natural-Join und einem Equi-Join.

b) [1 Punkt] Erläutern Sie kurz was man unter einem Theta-Join versteht.

c) [3 Punkte] Was versteht man unter Unionkompatibilität? Nennen Sie drei SQL-Operatoren welche Unionkompatibilität voraussetzen.

d) [4 Punkte] Erläutern Sie Backward und Forward Recovery und grenzen Sie diese voneinander ab.

e) [5 Punkte] Erklären Sie das Zwei-Phasen-Freigabe-Protokoll.

f) [5 Punkte] Erläutern Sie Partial Undo / Redo und Global Undo / Redo und deren Bedeutung für die Umsetzung des ACID-Prinzips. Geben Sie zu jeder dieser Konzepte an, ob System-, Programm- oder Gerätefehler damit korrigiert werden können.

g) [2 Punkte] Erklären Sie das WAL-Prinzip (Write ahead logging)!

h) [3 Punkte] Erklären Sie den Begriff „Datenbankindex“ und nennen Sie zwei häufige Arten.

Fortsetzung nächste Seite!

Aufgabe 7: (SQL) (18 Punkte)

Gegeben sind folgende Relationen aus einem Verwaltungssystem für die jährlichen Formel 1-Rennen:

```
Strecke(Strecken_ID, Streckenname, Land, Länge)
Fahrer(Fahrer_ID, Fahrername, Nation, Rennstall)
Rennen(Strecken_ID, Jahr, Wetter)
            FK (Strecken_ID) referenziert Strecke(Strecken_ID)
Rennteilnahme(Fahrer_ID, Strecken_ID, Jahr, Rundenbestzeit, Gesamtzeit,
disqualifiziert)
            FK (Fahrer_ID) referenziert Fahrer(Fahrer_ID)
            FK (Strecken_ID, Jahr) referenziert Rennen(Strecken_ID, Jahr)
```

Der Einfachheit halber wird angenommen, dass Fahrer den Rennstall nicht wechseln können. Das Attribut „disqualifiziert" kann die Ausprägungen „ja" und „nein" haben. Formulieren Sie folgende Abfragen in SQL. Vermeiden Sie nach Möglichkeit übermäßige Nutzung von Joins und Views.

a) [3 Punkte] Geben Sie für jeden Fahrer seine ID sowie die Anzahl seiner Disqualifikationen in den Jahren 2005 bis 2017 aus. Ordnen Sie die Ausgabe absteigend nach der Anzahl der Disqualifikationen.

b) [4 Punkte] Gesucht sind alle Länder, aus denen noch nie ein Fahrer disqualifiziert wurde.

c) [8 Punkte] Gesucht sind die ersten fünf Plätze des Rennens von 2011 in „Abu Dhabi" (Streckenname). Die Ausgabe soll nach der Platzierung absteigend erfolgen. Geben Sie Fahrer_ID, Fahrername, Nation und Rennstall mit aus.

d) [3 Punkte] Führen Sie eine neue Spalte Gehalt in die Tabelle Fahrer ein. Da sich die Prämien für die Fahrer nach einem Rennstallwechsel ändern, soll ein Trigger geschrieben werden, mit dem das Gehalt des betreffenden Fahrers um 10% angehoben wird.