Erste Staatsprüfung für ein Lehramt an öffentlichen Schulen

Fach Informatik

Frühjahr 2019

66116

Datenbanksysteme / Softwaretechnologie (vertieft)

Aufgabenstellungen mit Lösungsvorschlägen

Aufgabenübersicht

Thema Nr. 1

Teilaufgabe Nr. 1

Aufgabe 2 [Medikamente]

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  ]}
```

```sql
CREATE TABLE Person (
  ID INTEGER PRIMARY KEY,
  Name VARCHAR(255),
  Wohnort VARCHAR(255)
);

CREATE TABLE Hersteller (
  ID INTEGER PRIMARY KEY,
  Name VARCHAR(255),
  Standort VARCHAR(255),
  AnzahlMitarbeiter INTEGER
);

CREATE TABLE Medikament (
  ID INTEGER PRIMARY KEY,
  Name VARCHAR(255),
  Kosten INTEGER,
  Wirkstoff VARCHAR(255),
  produziert_von INTEGER REFERENCES Hersteller(ID)
);

CREATE TABLE nimmt (
  Person INTEGER,
  Medikament INTEGER,
  von DATE,
  bis DATE,
  PRIMARY KEY (Person, Medikament, von, bis)
);

CREATE TABLE hat_Unverträglichkeit_gegen (
  Person INTEGER REFERENCES Person(ID),
  Medikament INTEGER REFERENCES Medikament(ID)
);

INSERT INTO Person VALUES
  (1, 'Walter Müller', 'Holzapfelkreuth'),
  (2, 'Dilara Yildiz', 'Fürth');

INSERT INTO Hersteller VALUES
  (1, 'Hexal', 'Holzkirchen', 3700),
  (2, 'Ratiopharm', 'Ulm', 563);

INSERT INTO Medikament VALUES
  (1, 'IbuHexal', 3, 'Ibuprofen', 1),
  (2, 'Ratio-Paracetamol', 2, 'Paracetamol', 2),
  (3, 'BudeHexal', 4, 'Budesonid', 1),
  (4, 'Ratio-Budesonid', 5, 'Budesonid', 2);

```

```sql
INSERT INTO nimmt VALUES
  (1, 1, '2021-07-12', '2021-07-23'),
  (1, 3, '2021-07-12', '2021-07-23'),
  (2, 4, '2021-02-13', '2021-03-24');

INSERT INTO hat_Unverträglichkeit_gegen VALUES
  (1, 1),
  (1, 3),
  (2, 2);
```

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.

```sql
ALTER TABLE nimmt
ADD CONSTRAINT FK_Person
  FOREIGN KEY (Person) REFERENCES Person(ID),
ADD CONSTRAINT FK_Medikament
  FOREIGN KEY (Medikament) REFERENCES Medikament(ID);
```

(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.

```sql
SELECT DISTINCT
  p.ID as Personen_ID,
  p.Name as Personen_Name,
  m.ID as Medikamenten_ID,
  m.Name as Medikamenten_Name
FROM Person p, Medikament m, nimmt n
WHERE
  n.Person = p.ID AND
  n.Medikament = m.ID;
```

(c) Schreiben Sie eine SQL-Anweisung, welche die ID und den Namen der Medikamente mit den niedrigsten Kosten je Hersteller bestimmt.

```sql
SELECT m.ID, m.Name
FROM Medikament m, Medikament n
WHERE
  m.produziert_von = n.produziert_von AND
  m.Kosten >= n.Kosten
GROUP BY m.ID, m.Name
HAVING COUNT(*) <= 1;
```

(d) Schreiben Sie eine SQL-Anweisung, die die Anzahl aller Personen ermittelt, die ein Medikament genommen haben, gegen welches sie eine Unverträglichkeit entwickelt haben.

```sql
SELECT COUNT(DISTINCT p.ID)
FROM
  Person p,
  nimmt n,
  hat_Unverträglichkeit_gegen u
WHERE
  p.ID = n.Person AND
```

```
    u.Medikament = n.Medikament;
```

(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.

```
SELECT ID, Name FROM Person
EXCEPT
SELECT p.ID, p.Name
FROM Person p, nimmt n, Medikament m
WHERE
  p.ID = n.Person AND
  n.Medikament = m.ID AND
  m.Wirkstoff IN ('Ibuprofen', 'Paracetamol');
```

(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.

```
SELECT p.Name, (
SELECT COUNT(h.ID)
  FROM Hersteller h, Medikament m, hat_Unverträglichkeit_gegen u
  WHERE
    m.produziert_von = h.ID AND
    u.Medikament = m.ID AND
    h.ID = p.ID
) AS Unverträglichkeiten
FROM Hersteller p
ORDER BY Unverträglichkeiten ASC;
```

(g) Formulieren Sie eine Anfrage in relationaler Algebra, welche die Wohnorte aller Personen bestimmt, welche ein Medikament mit dem Wirkstoff Paracetamol nehmen oder genommen haben. Die Lösung kann als Baum- oder als Term-Schreibweise angegeben werden.

$$\pi_{\text{Person.Wohnort}}\left(\sigma_{\text{Medikament.Wirkstoff}='\text{Paracetamol}'}(\text{Person} \bowtie_{\text{Person.ID}=\text{nimmt.Person}} \text{nimmt} \bowtie_{\text{nimmt.Medikament}=\text{Medikament.ID}}$$

(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.

$$\pi_{\text{Person.Name}}\left(\sigma_{\text{Hersteller.Standort}='\text{München}'}((\text{Person} \bowtie_{\text{Person.ID}=\text{nimmt.Person}} \text{nimmt}) \bowtie_{\text{nimmt.Medikament}=\text{Medikament.ID}}$$

Aufgabe 3 [Relation A-F]

Gegeben sei folgendes relationales Schema R in erster Normalform:

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

Für R gelte folgende Menge FD funktionaler Abhängigkeiten:

$$\text{FA} = \Big\{ \begin{aligned} \{A, D, F\} &\rightarrow \{E\}, \\ \{B, C\} &\rightarrow \{A, E\}, \\ \{D\} &\rightarrow \{B\}, \\ \{D, E\} &\rightarrow \{C, B\}, \\ \{A\} &\rightarrow \{F\}, \end{aligned} \Big\}$$

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

- $\{ D, A \}$
- $\{ D, C \}$
- $\{ D, E \}$

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

R ist in 1NF, da $\{ d \} \rightarrow \{ b \}$

(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).

Linksreduktion

— Führe für jede funktionale Anhängigkeit $\alpha \rightarrow \beta \in F$ die Linksreduktion durch, überprüfe also für alle $A \in \alpha$, ob A überflüssig ist, d. h. ob $\beta \subseteq AttrHülle(F, \alpha - A)$.

$\{A, D, F\} \rightarrow \{E\}$

$E \notin \text{AttrHülle}(F, \{A, D, F \setminus A\}) = \{D, F, B\}$
$E \notin \text{AttrHülle}(F, \{A, D, F \setminus D\}) = \{A, F\}$
$E \in \text{AttrHülle}(F, \{A, D, F \setminus F\}) = \{A, B, D, \boldsymbol{F}\}$

$\{B, C\} \rightarrow \{A, E\}$

$\{A, E\} \notin \text{AttrHülle}(F, \{B, C \setminus B\}) = \{C\}$
$\{A, E\} \notin \text{AttrHülle}(F, \{B, C \setminus C\}) = \{B\}$

$\{D, E\} \rightarrow \{C, B\}$

$\{C, B\} \notin \text{AttrHülle}(F, \{D, E \setminus D\}) = \{E\}$
$\{C, B\} \notin \text{AttrHülle}(F, \{D, E \setminus E\}) = \{B, D\}$

$$\text{FA} = \Big\{ \begin{array}{r} \{ A, D \} \rightarrow \{ E \}, \\ \{ B, C \} \rightarrow \{ A, E \}, \\ \{ D \} \rightarrow \{ B \}, \\ \{ D, E \} \rightarrow \{ C, B \}, \\ \{ A \} \rightarrow \{ F \}, \end{array} \Big\}$$

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

Rechtsreduktion

— Führe für jede (verbliebene) funktionale Abhängigkeit $\alpha \rightarrow \beta$ die Rechtsreduktion durch, überprüfe also für alle $B \in \beta$, ob $B \in AttrHülle(F - (\alpha \rightarrow \beta) \cup (\alpha \rightarrow (\beta - B)), \alpha)$ gilt. In diesem Fall ist B auf der rechten Seite überflüssig und kann eleminiert werden, ðα → β wird durch $\alpha \rightarrow (\beta - B)$ ersetzt.

E

$E \notin \text{AttrHülle}(F \setminus \{ A, D \} \rightarrow \{ E \}, \{A, D\}) = \{A, B, D, F\}$
$E \notin \text{AttrHülle}(F \setminus \{ B, C \} \rightarrow \{ A, E \} \cup \{ B, C \} \rightarrow \{ A \}, \{B, C\}) = \{A, B, C, F\}$

B

$B \notin \text{AttrHülle}(F \setminus \{ D \} \rightarrow \{ B \}, \{D\}) = \{D\}$

$B \in \text{AttrHülle}(F \setminus \{ D, E \} \to \{ C, B \} \cup \{ D, E \} \to \{ C \}, \{D, E\}) = \{\boldsymbol{B}, D, E\}$

$$\text{FA} = \Big\{ \begin{array}{r} \{ A, D \} \to \{ E \}, \\ \{ B, C \} \to \{ A, E \}, \\ \{ D \} \to \{ B \}, \\ \{ D, E \} \to \{ C \}, \\ \{ A \} \to \{ F \}, \end{array} \Big\}$$

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

- **Löschen leerer Klauseln**
  — *Entferne die funktionalen Abhängigkeiten der Form $\alpha \to \varnothing$, die im 2. Schritt möglicherweise entstanden sind.* ———

  $\varnothing$ Nichts zu tun
- **Vereinigung**
  — *Fasse mittels der Vereinigungsregel funktionale Abhängigkeiten der Form $\alpha \to \beta_1, \ldots, \alpha \to \beta_n$, so dass $\alpha \to \beta_1 \cup \cdots \cup \beta_n$ verbleibt.* ———

  $\varnothing$ Nichts zu tun

(d) Zerlegen Sie R mit FD_C mithilfe des Synthesealgorithmus in 3NF. Geben Sie zudem alle funktionalen Abhängigkeiten der erzeugten Relationenschemata an.

- **Relationsschemata formen**
  — *Erzeuge für jede funktionale Abhängigkeit $\alpha \to \beta \in F_c$ ein Relationenschema $\mathcal{R}_\alpha := \alpha \cup \beta$.* ———

  $R_1(\underline{A, D}, E)$
  $R_2(\underline{B, C}, A, E)$
  $R_3(\underline{D}, B)$
  $R_4(\underline{D, E}, C)$
  $R_5(\underline{A}, F)$
- **Schlüssel hinzufügen**
  — *Falls eines der in Schritt 2. erzeugten Schemata R_α einen Schlüsselkandidaten von $\mathcal{R}$ bezüglich F_c enthält, sind wir fertig, sonst wähle einen Schlüsselkandidaten $\mathcal{K} \subseteq \mathcal{R}$ aus und definiere folgendes zusätzliche Schema: $\mathcal{R}_\mathcal{K} := \mathcal{K}$ und $\mathcal{F}_\mathcal{K} := \varnothing$* ———

  $\varnothing$ Nichts zu tun
- **Entfernung überflüssiger Teilschemata**
  — *Eliminiere diejenigen Schemata R_α, die in einem anderen Relationenschema $R_{\alpha'}$ enthalten sind, d. h. $R_\alpha \subseteq R_{\alpha'}$.* ———

  $\varnothing$ Nichts zu tun

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

R1 und R4 sind in BCNF, weil ihre Determinanten Schlüsselkandidaten sind.

Teilaufgabe Nr. 2

Aufgabe 1 [Grafik: Kreis, Quadrat, Dreieck]

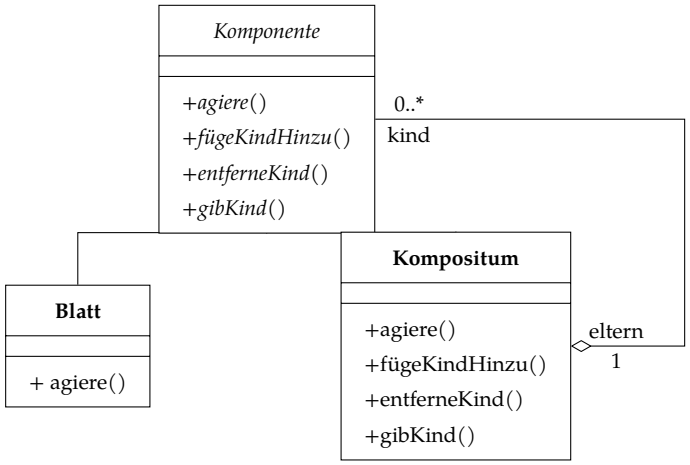
Gegeben sei folgender Sachverhalt: Eine Grafik ist entweder ein Kreis, ein Quadrat oder ein Dreieck. Eine Grafik kann zudem auch eine Kombination aus diesen Elementen sein. Des Weiteren können Sie aus mehreren Grafiken auch neue Grafiken zusammenbauen. Sie denken sich: Ich möchte eine Menge von Grafiken genauso wie eine einzelne Grafik behandeln können.

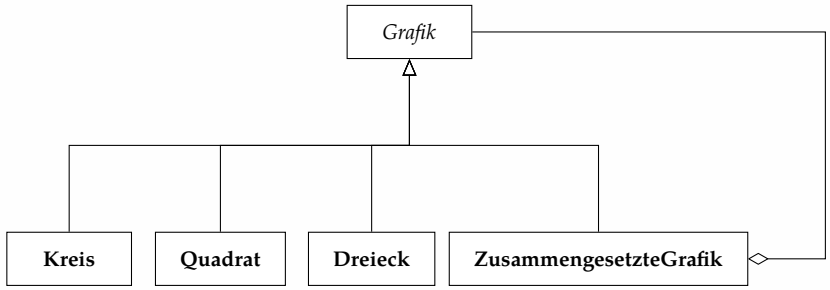
(a) Welches Entwurfsmuster sollten Sie zur Modellierung verwenden?

Kompositum

(b) Zeichnen Sie das entsprechende Klassendiagramm. Es reicht, nur die Klassennamen mit ihren Assoziationen und Vererbungsbeziehungen anzugeben; ðohne Methoden und Attribute.

Exkurs: Kompositum





Aufgabe 2 [Roboter in einer Montagehalle]

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):

(a) Er kann sich zu einer angegebenen Position bewegen.

(b) Er hat eine Methode, um einen Auftrag abzuarbeiten, d.h. alle Auftragspositionen zu holen.

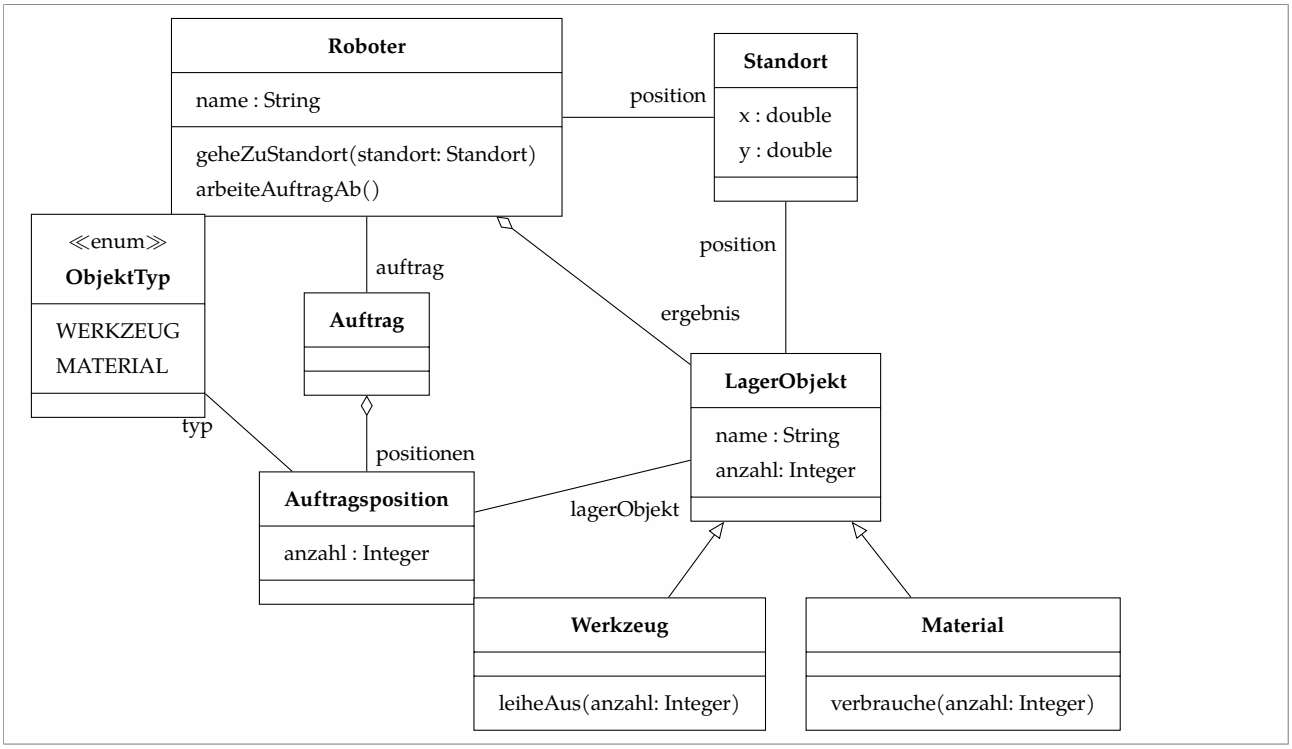
Alle Objekte im Lager haben jeweils einen Namen, einen Standort mit Angabe einer Position (s. oben) und speichern außerdem die jeweils noch vorhandene Stückzahl. Es gibt zwei Typen von Objekten: Werkzeuge mit einer Methode, mit der ein einzelnes Werkzeug ausgeliehen werden kann, und Materialien mit einer 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 × Schrauben M6, 10 × Schrauben M10 und 2 × 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 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).

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 [Roboter in einer Montagehalle]

(a) Erläutern Sie kurz, was man unter der Methode der testgetriebenen Entwicklung versteht.

> Bei der testgetriebenen Entwicklung erstellt der/die ProgrammiererIn Softwaretests konsequent vor den zu testenden Komponenten.

(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 [Roboter in einer Montagehalle]

(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 [Roboter in einer Montagehalle]

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.

```java
import java.util.ArrayList;

public class Auftrag {
  ArrayList<AuftragsPosition> positionen;

  public Auftrag() {
    this.positionen = new ArrayList<AuftragsPosition>();
  }

  public void fügePositionHinzu(LagerObjekt lagerObjekt, int anzahl) {
    positionen.add(new AuftragsPosition(lagerObjekt, anzahl));
  }
}
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/examen/examen_66116/jahr_2019/fruehjahr/roboter/Auftrag.java

```java
public class AuftragsPosition {
  ObjektTyp typ;
  LagerObjekt lagerObjekt;
  int anzahl;

  public AuftragsPosition(LagerObjekt lagerObjekt, int anzahl) {
    if (lagerObjekt instanceof Werkzeug) {
      typ = ObjektTyp.WERKZEUG;
    } else {
      typ = ObjektTyp.MATERIAL;
    }
    this.lagerObjekt = lagerObjekt;
    this.anzahl = anzahl;
  }
}
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/examen/examen_66116/jahr_2019/fruehjahr/roboter/AuftragsPosition.java

```java
public class LagerObjekt {
  String name;
  Standort position;
  int anzahl;

  public LagerObjekt(String name, int x, int y, int anzahl) {
    this.name = name;
    this.position = new Standort(x, y);
    this.anzahl = anzahl;
  }
}
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/examen/examen_66116/jahr_2019/fruehjahr/roboter/LagerObjekt.java

```java
public class Material extends LagerObjekt {
  public Material(String name, int x, int y, int anzahl) {
    super(name, x, y, anzahl);
  }

  void verbrauche(int anzahl) {
  }

}
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/examen/examen_66116/jahr_2019/fruehjahr/roboter/Material.java

```java
public enum ObjektTyp {
  WERKZEUG,
  MATERIAL
}
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/examen/examen_66116/jahr_2019/fruehjahr/roboter/ObjektTyp.java

```java
import java.util.ArrayList;

public class Roboter {
  String name;
  Standort position;
  ArrayList<LagerObjekt> ergebnis;

  void geheZuStandort(Standort standort) {

  }

  void arbeiteAuftragAb() {

  }

  public static void main(String[] args) {
    Werkzeug bohrerB1 = new Werkzeug("Bohrer Typ B1", 1, 0, 1);
    Material schraubenM6 = new Material("Schrauben M6", 1, 1, 100);
    Material schraubenM10 = new Material("Schrauben M10", 1, 2, 10);
    Material blechB72 = new Material("Blech B72", 1, 3, 2);

    Auftrag auftrag = new Auftrag();
    auftrag.fügePositionHinzu(bohrerB1, 1);
    auftrag.fügePositionHinzu(schraubenM6, 100);
    auftrag.fügePositionHinzu(schraubenM10, 10);
    auftrag.fügePositionHinzu(blechB72, 2);
  }
}
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/examen/examen_66116/jahr_2019/fruehjahr/roboter/Roboter.java

```java
public class Standort {
  double x;
  double y;

  public Standort(int x, int y) {
    this.x = x;
    this.y = y;
  }
}
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/examen/examen_66116/jahr_2019/fruehjahr/roboter/Standort.java

```java
public class Werkzeug extends LagerObjekt {

  public Werkzeug(String name, int x, int y, int anzahl) {
    super(name, x, y, anzahl);
  }

  public void leiheAus(int anzahl) {

  }

}
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/examen/examen_66116/jahr_2019/fruehjahr/roboter/Werkzeug.java

Thema Nr. 2

Teilaufgabe Nr. 1

Aufgabe 1 [Allgemeine Fragen]

(a) Das ACID-Prinzip fordert unter anderem die Atomaritat und die Isolation einer Transaktion. Beschreiben Sie kurz die Bedeutung dieser Begriffe.

Atomicity Eine Transaktion ist atomar, ðvon den vorgesehenen Änderungsoperationen auf die Datenbank haben entweder alle oder keine eine Wirkung auf die Datenbank.

Isolation Eine Transaktion bemerkt das Vorhandensein anderer (parallel ablaufender) Transaktionen nicht und beeinflusst auch andere Transaktionen nicht.

(b) Was versteht man unter physischer Datenunabhängigkeit?

Änderungen an der physischen Speicher- oder der Zugriffsstruktur (beispielsweise durch das Anlegen einer Indexstruktur) haben keine Auswirkungen auf die logische Struktur der Datenbasis, also auf das Datenbankschema.

(c) In welche drei Ebenen unterteilt die 3-Ebenen-Architektur Datenbanksysteme?

Externe Ebene / (Benutzer)sichten (Views) Die externe Ebene stellt den Benutzern und Anwendungen individuelle Benutzersichten bereit, wie beispielsweise Formulare, Masken-Layouts, Schnittstellen.

Konzeptionelle / logische Ebene Die konzeptionelle Ebene beschriebt, welche Daten in der Datenbank gespeichert sind, sowie deren Beziehungen. Ziel des Datenbankdesigns ist eine vollständige und redundanzfreie Darstellung aller zu speichernden Informationen. Hier findet die Normalisierung des relationalen Datenbankschemas statt.

Interne / physische Ebene Die interne Ebene stellt die physische Sicht der Datenbank im Computer dar. Sie beschreibt, wie und wo die Daten in der Datenbank gespeichert werden. Oberstes Designziel ist ein effizienter Zugriff auf die gespeicherten Informationen, der meist durch bewusst in Kauf genommene Redundanz (z. B. Index) erreicht wird.

(d) Definieren Sie, was ein Schlüssel ist.

Ein Schlüssel ist ein Attribut oder eine Attributkombination, von der alle Attribute einer Relation funktional abhängen.

(e) Gegeben ist eine Relation $R(A, B, C)$, deren einziger Schlüsselkandidat A ist. Nennen Sie zwei Superschlüssel.

$\{A, B\}$ oder $\{A, C\}$ oder $\{A, B, C\}$

(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;
```

(i) Wieviele Ergebnistupel liefert die Anfrage Q_1 mindestens?

0

(ii) Wieviele Ergebnistupel liefert die Anfrage Q_1 höchstens?

7

(iii) Wieviele Ergebnistupel liefert die Anfrage Q_2 mindestens?

9

(iv) Wieviele Ergebnistupel liefert die Anfrage Q_2 höchstens?

9

(g) Erläutern Sie, was es bedeutet, wenn ein Attribut prim ist.

Sei R ein Relationenschema. Ein Attribut $A \in R$ heißt *prim*, falls A Teil eines Schlüsselkandidaten von R ist. Andernfalls heißt A *nichtprim*.

(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.

Reflexivität: Eine Menge von Attributen bestimmt eindeutig die Werte einer Teilmenge dieser Attribute (triviale Abhängigkeit), das heißt, $\beta \subseteq \alpha \Rightarrow \alpha \rightarrow \beta$.

Verstärkung: Gilt $\alpha \rightarrow \beta$, so gilt auch $\alpha\gamma \rightarrow \beta\gamma$ für jede Menge von Attributen γ der Relation.

Transitivität: Gilt $\alpha \rightarrow \beta$ und $\beta \rightarrow \gamma$, so gilt auch $\alpha \rightarrow \gamma$.

(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.

1NF. $\{A\} \rightarrow \{C\}$ verletzt die 2NF, da das Nichtprimärattribut C funktional von einer echten Teilmenge (A) des Schlüsselkandidaten $(\{A, B\} \rightarrow \{D\})$ abhängt.

(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.)

Vergleiche Kemper Seite 341 Kapitel „Theorie der Serialisierbarkeit“.

Aufgabe 3 [Relation X und Y]

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

Geben Sie die Ergebnisrelationen folgender Ausdrücke der relationalen Algebra als Tabellen an; machen Sie Ihren Rechenweg kenntlich.

(a) $\sigma_{A=2}(X) \bowtie Y$

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

(b) $(\pi_{B,C}(X) - \pi_{B,C}(Y)) \bowtie X$

(c)

A
1
2
3

Aufgabe 5 [App für konfizierte Schüler-Smartphones]

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, Fachl, 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 SOL-DDL an, die notwendig ist, um die Relation 'App' zu erzeugen.

```
CREATE TABLE App(
Name VARCHAR(50) PRIMARY KEY,
Bewertung VARCHAR(255),
```

```
Preis INTEGER);
```

(b) Geben Sie die Anweisung in SOL-DDL an, die notwendig ist, um die Relation 'Installiert' zu erzeugen.

```
CREATE TABLE Installiert(
ID INTEGER REFERENCES Smartphone(ID),
Name VARCHAR(50) App(Name),
PRIMARY KEY(ID,Name));
```

(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.

```
SELECT DISTINCT a.Name, a.Bewertung
FROM Eingesammelt e, Installiert i, App a
WHERE e.ID =i.ID AND i.Name = a.Name AND e.Name = "Keating"
ORDER BY a.Bewertung DESC;
```

(d) Formulieren Sie die folgende Anfrage in SQL:

Gesucht ist der durchschnittliche Notenschnitt der Schüler, denen ein iPhone 65 abgenommen wurde. Ein iPhone 6s kann A1633 als Modellnummer haben oder A1688.

```
SELECT DISTINCT AVG(s.Notenschnitt)
FROM Schüler s, Eingesammelt e, Smartphone sm
WHERE s.Vorname = e.Vorname AND s.Nachname = e.Nachname
AND e.ID=sm.ID AND(sm.Modellnr = 'A1633' OR sm.Modellnr = '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.

```
CREATE VIEW NumberApps AS
SELECT s.ID, s.Modellnr, COUNT(i.Name) AS number
FROM Smartphone s, Installiert i
WHERE s.ID = i.ID
GROUP BY s.ID
SELECT ModellNr, AVG(number)
FROM NumberApps
GROUP BY ModellNr
```

Aufgabe 6 [Normalisierung]

Gegeben sei das Relationsschema $R(A, B, C, D, E, F)$, sowie die Menge der zugehörigen funktionalen Abhängigkeiten:

$$\mathrm{FA} = \Big\{ \begin{aligned} \{B\} &\to \{F\}, \\ \{C, D\} &\to \{E\}, \\ \{C\} &\to \{A\}, \\ \{C, D\} &\to \{A\}, \\ \{D\} &\to \{F\}, \\ \{D\} &\to \{B\}, \end{aligned} \Big\}$$

(a) Bestimmen Sie den Schlüsselkandidaten der Relation R und begründen Sie, warum es keine weiteren Schlüsselkandidaten gibt.

Der Schlüsselkandidat ist $\{ C, D \}$, da $\{ C, D \}$ auf keiner rechten Seiten der Funktionalen Abhängigkeiten vorkommt. Außerdem ist $\{ C, D \}$ ein Superschlüssel da gilt: AttrHülle$(F, \{C, E\}) = \{A, B, C, D, E, G\} = R$

(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.

(i) **Kanonische Überdeckung**

— *Die kanonische Überdeckung - also die kleinst mögliche noch äquivalente Menge von funktionalen Abhängigkeiten kann in vier Schritten erreicht werden.*

i. **Linksreduktion**

— *Führe für jede funktionale Anhängigkeit $\alpha \to \beta \in F$ die Linksreduktion durch, überprüfe also für alle $A \in \alpha$, ob A überflüssig ist, d. h. ob $\beta \subseteq$ AttrHülle$(F, \alpha - A)$.*

$$\text{FA} = \Big\{ \begin{aligned} \{ B \} &\to \{ F \}, \\ \{ C, D \} &\to \{ E \}, \\ \{ C \} &\to \{ A \}, \\ \{ C \} &\to \{ A \}, \\ \{ D \} &\to \{ F \}, \\ \{ D \} &\to \{ B \}, \end{aligned} \Big\}$$

ii. **Rechtsreduktion**

— *Führe für jede (verbliebene) funktionale Abhängigkeit $\alpha \to \beta$ die Rechtsreduktion durch, überprüfe also für alle $B \in \beta$, ob $B \in$ AttrHülle$(F - (\alpha \to \beta) \cup (\alpha \to (\beta - B)), \alpha)$ gilt. In diesem Fall ist B auf der rechten Seite überflüssig und kann eleminiert werden, ðα → β wird durch $\alpha \to (\beta - B)$ ersetzt.*

$$\text{FA} = \Big\{ \begin{aligned} \{ B \} &\to \{ F \}, \\ \{ C, D \} &\to \{ E \}, \\ \{ C \} &\to \{ \varnothing \}, \\ \{ C \} &\to \{ A \}, \\ \{ D \} &\to \{ \varnothing \}, \\ \{ D \} &\to \{ B \}, \end{aligned} \Big\}$$

iii. **Löschen leerer Klauseln**

— *Entferne die funktionalen Abhängigkeiten der Form $\alpha \to \varnothing$, die im 2. Schritt möglicherweise entstanden sind.*

$$\text{FA} = \Big\{ \begin{aligned} \{ B \} &\to \{ F \}, \\ \{ C, D \} &\to \{ E \}, \\ \{ C \} &\to \{ A \}, \\ \{ D \} &\to \{ B \}, \end{aligned} \Big\}$$

iv. **Vereinigung**

— *Fasse mittels der Vereinigungsregel funktionale Abhängigkeiten der Form $\alpha \to \beta_1, \ldots, \alpha \to \beta_n$, so dass $\alpha \to \beta_1 \cup \cdots \cup \beta_n$ verbleibt.*

$\varnothing$ Nichts zu tun

(ii) **Relationsschemata formen**

— *Erzeuge für jede funktionale Abhängigkeit $\alpha \to \beta \in F_c$ ein Relationenschema $\mathcal{R}_\alpha := \alpha \cup \beta$.*

- $R_1(B, F)$

- $R_2(C, D, E)$
- $R_3(C, A)$
- $R_4(D, B)$

(iii) **Schlüssel hinzufügen**

— Falls eines der in Schritt 2. erzeugten Schemata R_α einen Schlüsselkandidaten von $\mathcal{R}$ bezüglich F_c enthält, sind wir fertig, sonst wähle einen Schlüsselkandidaten $\mathcal{K} \subseteq \mathcal{R}$ aus und definiere folgendes zusätzliche Schema: $\mathcal{R}_\mathcal{K} := \mathcal{K}$ und $\mathcal{F}_\mathcal{K} := \varnothing$ ———

$\varnothing$ Nichts zu tun

(iv) **Entfernung überflüssiger Teilschemata**

— Eliminiere diejenigen Schemata R_α, die in einem anderen Relationenschema $R_{\alpha'}$ enthalten sind, d. h. $R_\alpha \subseteq R_{\alpha'}$. ————

$\varnothing$ Nichts zu tun

Teilaufgabe Nr. 2

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.

Bei der testgetriebenen Entwicklung erstellt der/die ProgrammiererIn Softwaretests konsequent vor den zu testenden Komponenten.

(b) Komponententests

Komponententests sind immer White-Box-Tests und nie Black-Box-Tests.

Falsch: Komponententests (anderes Wort für Unit- oder Modul-Tests) können beides sein. [a]

[a] https://qastack.com.de/software/362746/what-is-black-box-unit-testing

(c) Akzeptanztests

Akzeptanz - resp. Funktionstests sind immer Black-Box-Tests und nie White-Box-Tests.

In systems engineering, it may involve black-box testing performed on a system

Acceptance testing in extreme programming: Acceptance tests are black-box system tests.

[a]

[a] https://en.wikipedia.org/wiki/Acceptance_testing

(d) Fehler

Ein fehlgeschlagener Test und ein Testausführungsfehler bezeichnen denselben Sachverhalt.

Falsch:

Fehler (Error)

Der Software-Test konnte durchgeführt werden, das ermittelte Ist-Ergebnis und das vorgegebene Soll-Ergebnis weichen jedoch voneinander ab. Ein derartiger Fehler liegt z. B. dann vor, wenn ein Funktionsaufruf einen abweichenden Wert berechnet.

Fehlschlag (Failure)

Während der Testdurchführung wurde ein Systemversagen festgestellt. Ein Fehlschlag liegt z. B. dann vor, wenn das zu testende System einen Programmabsturz verursacht und hierdurch erst gar kein Ist-Ergebnis ermittelt werden konnte.

Das Misslingen kann als Ursache einen Fehler (Error) oder ein falsches Ergebnis (Failure) haben, die beide per Exception signalisiert werden. Der Unterschied zwischen den beiden Begriffen liegt darin,

dass Failures erwartet werden, während Errors eher unerwartet auftreten. [a]

[a] https://de.wikipedia.org/wiki/JUnit

(e) Test Suiten

Tests können hierarchisch strukturiert werden, so dass mit einem Befehl das gesamte zu testende System getestet werden kann.

Richtig:

test suite (englisch „Testsammlung“, aus französisch suite Folge, Verkettung) bezeichnet eine organisierte Sammlung von Werkzeugen zum Testen technischer Apparate und Vorgänge. [a]

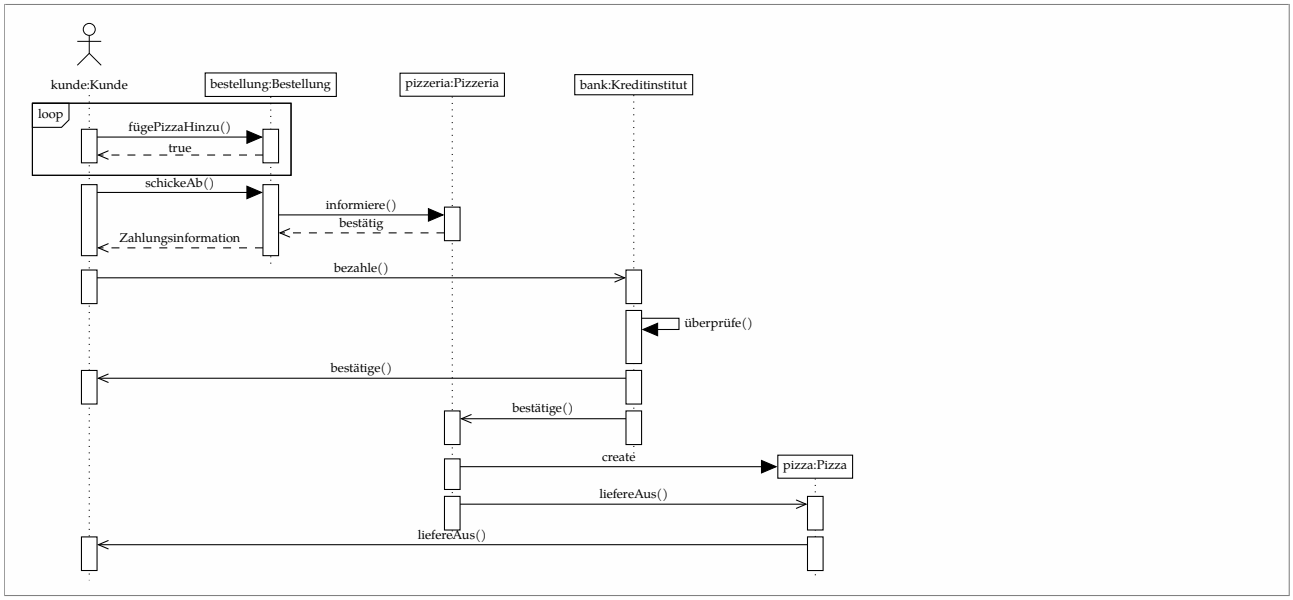
[a] https://de.wikipedia.org/wiki/Testsuite

Aufgabe 2 [Webdienst PizzaNow]

Eine Pizzeria möchte mit dem Webdienst PizzaNow anbieten, Pizzen online bestellen zu können. Es gilt die folgende Beschreibung:

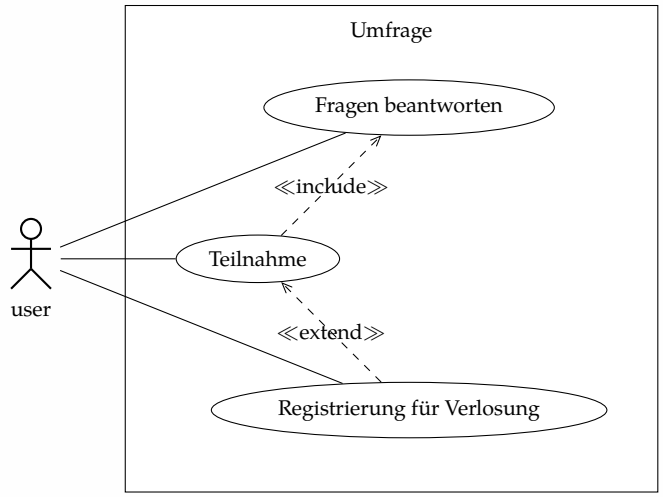
„Ein Kunde kann mehrere Pizzen in PizzaNow zu einer Bestellung zusammenstellen. Nachdem er die Bestellung abgeschickt hat, wird die Pizzeria über diese Bestellung informiert. Dort wird die Bestellung bestätigt und der Kunde erhält eine Bestätigung mit Zahlungsinformationen. Diese müssen vom jeweiligen 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.



Aufgabe 3 [Umfrage]

(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.



«include» Der/die Anwender/in kann nur an der Umfrage teilnehmen, wenn er/sie die Fragen beantwortet hat.

«extend» Der/die Anwender/in kann zusätzlich zur Teilnahme an der Umfrage sich für die Verlosung registrieren.

(b) Welche Informationen sollte eine typische Spezifikation eines Anwendungsfalls enthalten?

Ein Use-Case-Diagramm zeigt das externe Verhalten eines Systems aus der Sicht der Nutzer, indem es die Nutzer, die Use-Cases und deren Beziehungen zueinander darstellt.