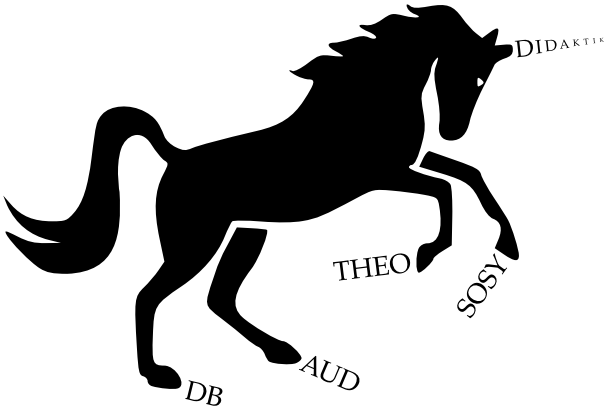
66116
Frühjahr
2015

Datenbanksysteme / Softwaretechnologie (vertieft)

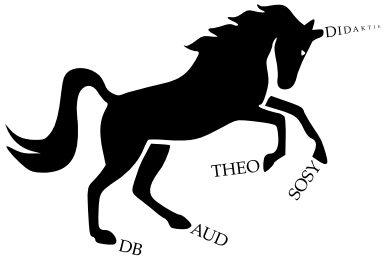
Aufgabenstellungen mit Lösungsvorschlägen



Die Bschlangaul-Sammlung

Hermine Bschlangaul and Friends

Aufgabenübersicht



Die Bschlangaul-Sammlung
Hermine Bschlangaul and Friends

Eine freie Aufgabensammlung mit Lösungen von Studierenden für Studierende zur Vorbereitung auf die 1. Staatsexamensprüfungen des Lehramts Informatik in Bayern.

Thema Nr. 1

Teilaufgabe Nr. 1

Aufgabe [Musik-Datenbank]

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 Sov. 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 Integritätsbedin. gung 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 Einfü- gungen ausgeführt wurden. Lediglich zu den Auszeichnungen müssen keine Tupel eingefügt werden.

Aufgabe 2 [Musik-CDs]

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.

Lösungsvorschlag

```
SELECT c.Name DISTINICT
FROM CD c, Interpret i
WHERE
  i.Interpreten_ID = c.Interpreten_ID AND
  i.name = 'Adele';
```

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

Lösungsvorschlag

```
SELECT i.Interpreten_ID, i.Name, COUNT(*)
FROM Interpret i, CD c
WHERE
  i.Interpreten_ID = c.Interpreten_ID
GROUP BY i.Interpreten_ID, i.Name;
```

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

Lösungsvorschlag

```
SELECT MAX(m.Länge)
FROM Interpret i, CD c, Musikstück m
WHERE
  m.CD_ID = c.CD_ID AND
  i.Name = 'Michael Jackson' AND
  c.Name = 'Thriller';
```

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

Lösungsvorschlag

```
SELECT i.Name
FROM Auszeichnung_CD acd, Auszeichnung_Stück ast, CD c, Interpret i
WHERE
  (acd.CD_ID = c.CD_ID OR ast.CD_ID = c.CD_ID) AND
  c.Interpreten_ID = i.Interpreten_ID;
```

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

Lösungsvorschlag

```
INSERT INTO Auszeichnung_CD VALUES
  (
    (
      SELECT c.CD_ID
      FROM CD c, Interpret i
      WHERE
        c.Name = 'Adele 21' AND
        i.Interpreten_ID = c.Interpreten_ID AND
        i.Name = 'Adele'
    ),
    'Emmy'
```

```
);

-- Test
SELECT * from Auszeichnung_CD;
```

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

Lösungsvorschlag

```
-- Test
SELECT * FROM Interpret;

UPDATE Interpret
SET Geschäftsadresse = 'Hollywood Boulevard 13, Los Angeles'
WHERE Name = 'Gensis';

-- Test
SELECT * FROM Interpret;
```

Aufgabe 3 [Relation A-F]

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

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

und der folgenden Menge F von funktionalen Abhängigkeiten:

$$\text{FA} = \left\{ \begin{array}{r} \{ A \} \rightarrow \{ B \}, \\ \{ A, B, C \} \rightarrow \{ D \}, \\ \{ D \} \rightarrow \{ B, C \}, \end{array} \right\}$$

(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.

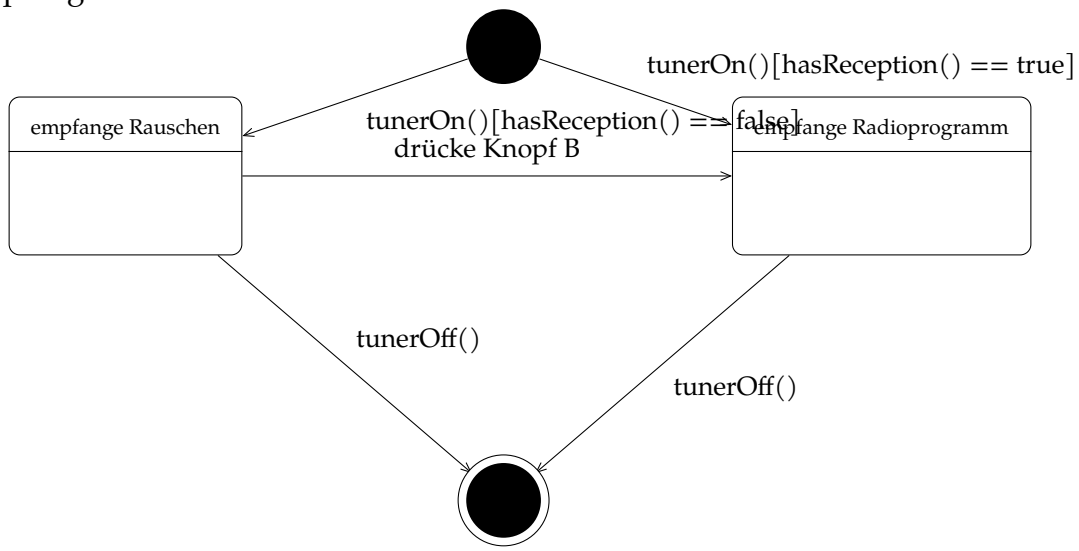
Teilaufgabe Nr. 2

Aufgabe 2 [Radiotuner]

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 [Kunden und Angestellte einer Firma]

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) die entsprechenden Attribute (Name, Geschlecht, Gehalt, Kundennummer) deklarieren.

Lösungsvorschlag

```java
public interface Person {
  String getName();

  char getGeschlecht();
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2015/fruehjahr/Person.java`

```java
public interface Angestellter extends Person {
  int getGehalt();
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2015/fruehjahr/Angestellter.java`

```java
public interface Kunde extends Person {
  int getKundennummer();
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2015/fruehjahr/Kunde.java`

- 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.

```
public abstract class PersonImpl implements Person {
  protected String name;
  protected char geschlecht;

  public PersonImpl(String name, char geschlecht) {
    this.name = name;
    this.geschlecht = geschlecht;
  }

  public String getName() {
    return name;
  }

  public char getGeschlecht() {
```

```
    return geschlecht;
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2015/fruehjahr/PersonImpl.java`

- 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.

Lösungsvorschlag

```
public class AngestellterImpl extends PersonImpl implements Angestellter {
  protected int gehalt;

  public AngestellterImpl(String name, char geschlecht, int gehalt) {
    super(name, geschlecht);
    this.gehalt = gehalt;
  }

  public int getGehalt() {
    return gehalt;
  }

}
```

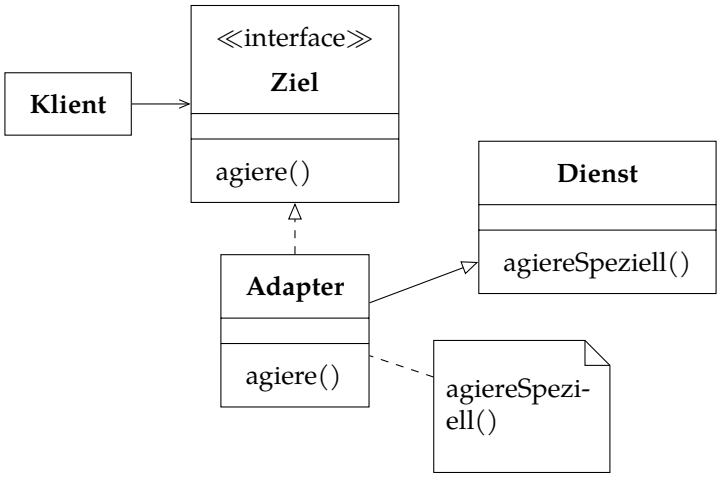
Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2015/fruehjahr/AngestellterImpl.java`

```
public class KundeImpl extends PersonImpl implements Kunde {
  protected int kundennummer;

  public KundeImpl(String name, char geschlecht, int kundennummer) {
    super(name, geschlecht);
    this.kundennummer = kundennummer;
  }

  public int getKundennummer() {
    return kundennummer;
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2015/fruehjahr/KundeImpl.java`

(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.

> **Exkurs: Entwurfsmuster „Adapter“**



Ziel (Target) Das Ziel definiert die Schnittstelle, die der Klient nutzen kann.

Klient (Client) Der Klient nutzt Dienste über inkompatible Schnittstellen und greift dabei auf adaptierte Schnittstellen zurück.

Dienst (Adaptee) Der Dienst bietet wiederzuverwendende Dienstleistungen mit fest definierter Schnittstelle an.

Adapter Der Adapter adaptiert die Schnittstelle des Dienstes auf die Schnittstelle zum Klienten.

```java
/**
 * GoF: Adaptee
 */
public class Dienst {

  public void agiereSpeziell() {
      System.out.println("Agiere speziell!");
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/entwurfsmuster/adapter/allgemein/Dienst.java`

```java
public interface Ziel {
  public void agiere();
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/entwurfsmuster/adapter/allgemein/Ziel.java`

```java
public class Adapter extends Dienst implements Ziel {

  @Override
  public void agiere() {
      System.out.print("agiere: ");
      agiereSpeziell();
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/entwurfsmuster/adapter/allgemein/Adapter.java`

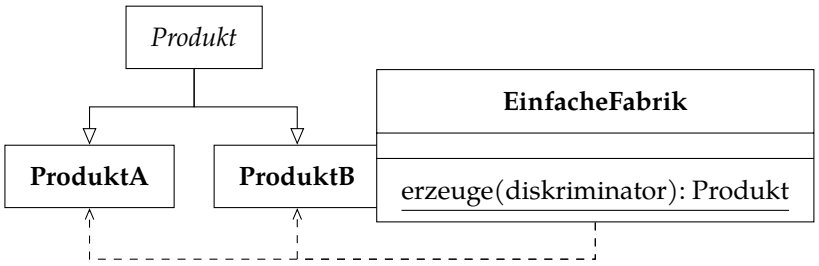
```java
public class Klient {

  public static void main(String[] args) {
    new Adapter().agiere();
    // agiere: Agiere speziell!
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/entwurfsmuster/adapter/allgemein/Klient.java`

(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.

Exkurs: Entwurfsmuster „Einfache Fabrik“



EinfacheFabrik Eine Klasse mit einer Erzeugunsmethode, die über eine größere Bedingung verschiedene Objekt instanziert.

Produkt Eine abstrakte Klasse, die von den konkreten Produkten geerbt wird.

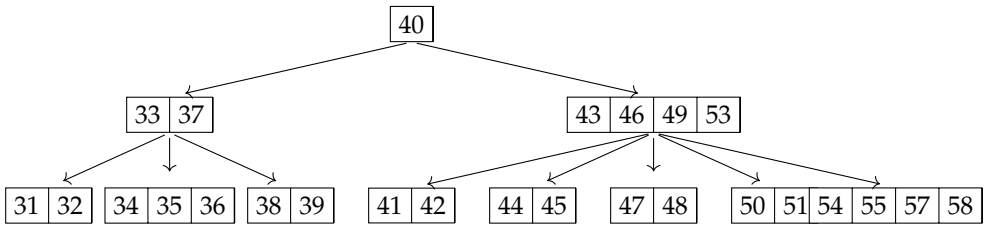
KonkretesProdukt Ein konkretes Produkt, das von der einfachen Fabrik erzeugt wird.

Thema Nr. 2

Teilaufgabe Nr. 1

Aufgabe 3 [Indexstrukturen]

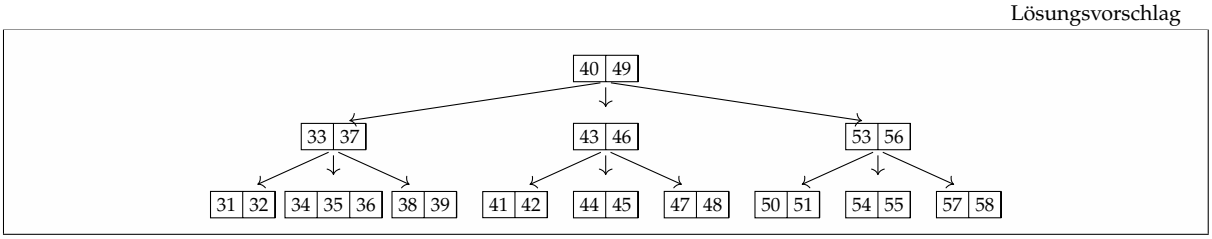
Als Indexstruktur einer Datenbank sei folgender B-Baum ($k = 2$) gegeben:



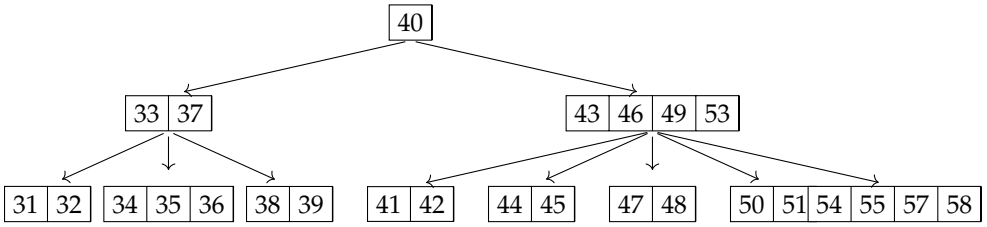
Führen Sie nacheinander die folgenden Operationen aus. Geben Sie die auftretenden Zwischenergebnisse an. Teilbäume, die sich in einem Schritt nicht verändern, müssen nicht erneut gezeichnet werden. Sollten Wahlmöglichkeiten auftreten, so sind größere Schlüsselwerte bzw. weiter rechts liegende Knoten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

(a) Einfügen des Wertes 56

Lösungsvorschlag



(b) Löschen des Wertes 37



Teilaufgabe Nr. 2

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() / getHeightFactor();
  }

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

  protected abstract int getWidthFactor();

  protected abstract int getHeightFactor();
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2015/fruehjahr/reverse/Display.java`

```
import java.awt.Color;

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

  int getHeight();

  int getWidth();
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2015/fruehjahr/reverse/PixelPainter.java`

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

  int getWidth();

  int getHeight();
}
```

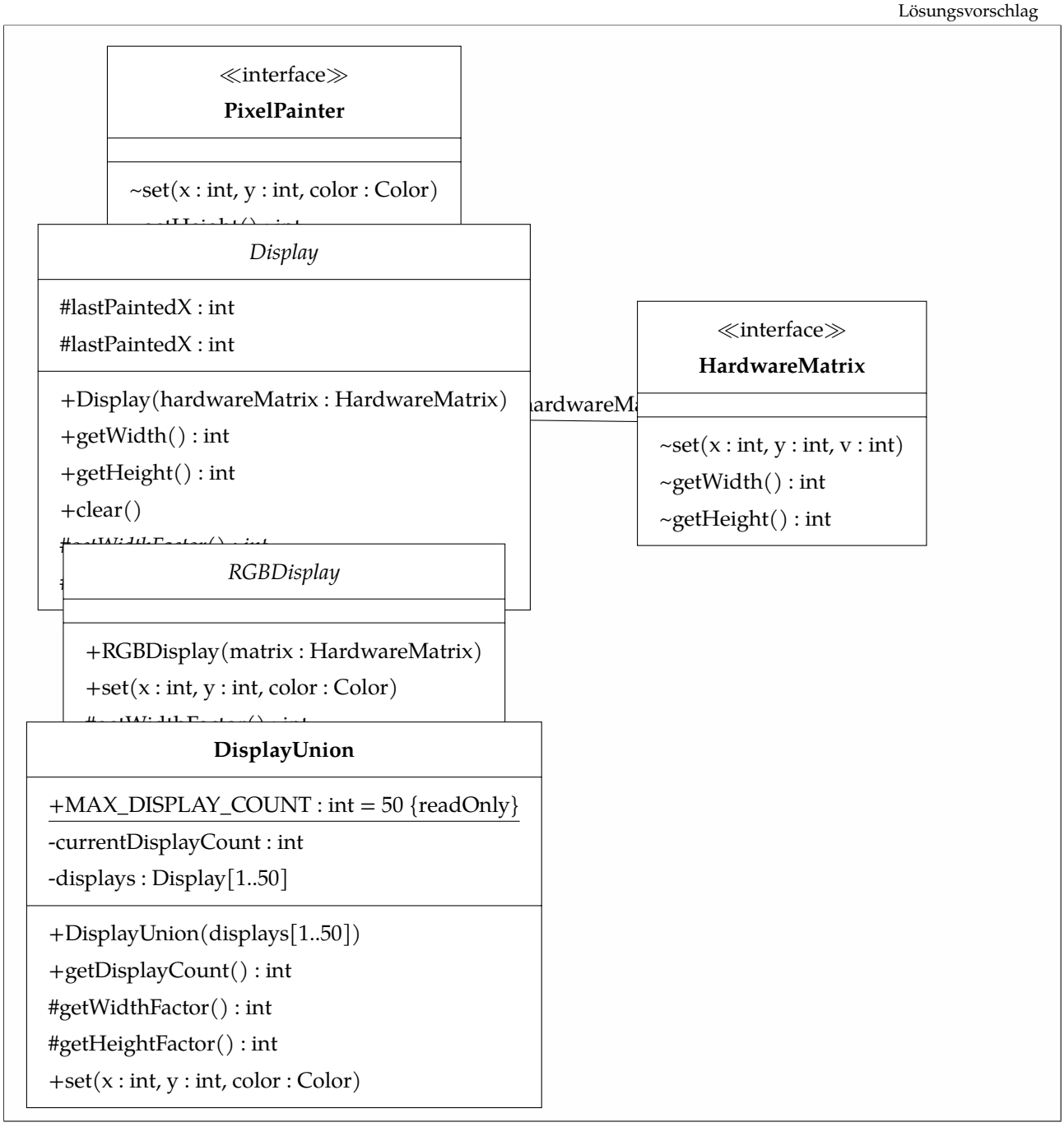
Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2015/fruehjahr/reverse/HardwareMatrix.java`

```java
import java.awt.Color;

@SuppressWarnings({ "unused" })
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) {
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2015/fruehjahr/reverse/DisplayUnion.java`

```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;
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2015/fruehjahr/reverse/RGBDisplay.java`

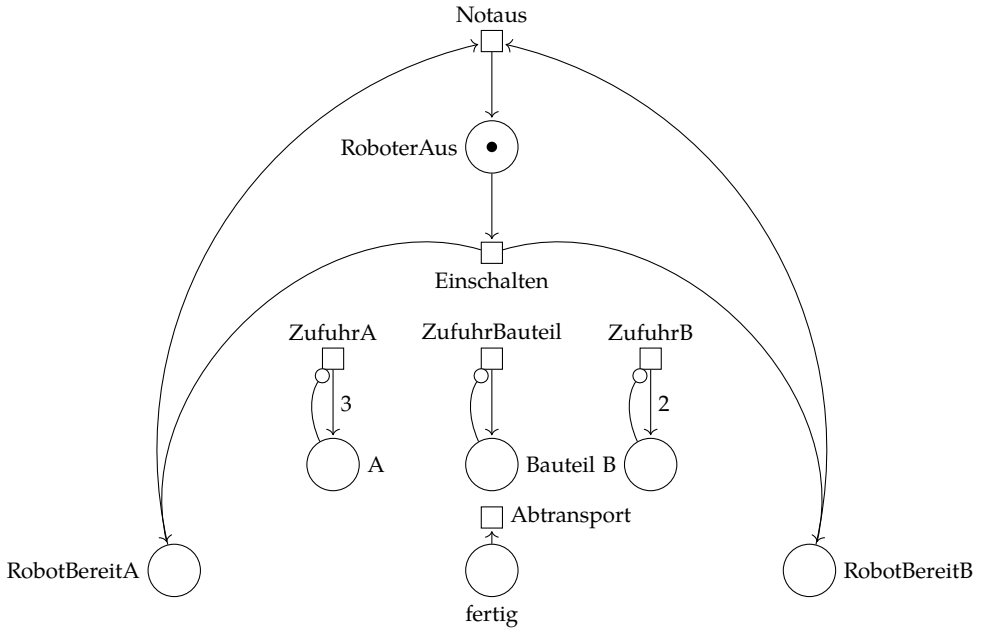


Aufgabe 3 [Automatisierungsanlage mit zwei Robotern]

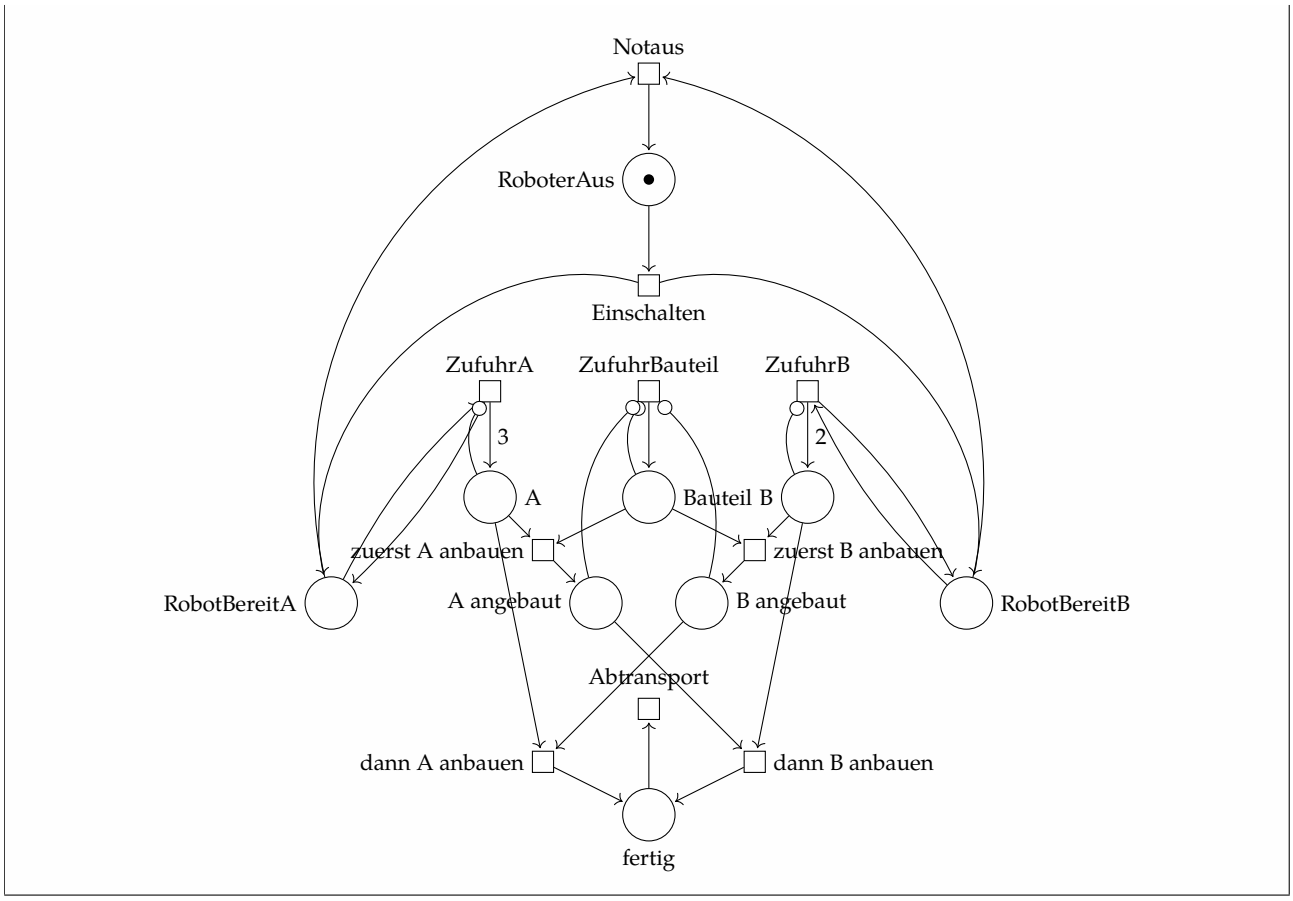
Das folgende Grundgerüst stammt aus dem Petri-Netz-Modell einer Automatisierungsanlage mit zwei Robotern, das Sie auf Ihr Blatt übernehmen und geeignet um weitere Plätze (Stellen), Transitionen, Kapazitäten, Gewichte und Markierungen so ergänzen sollen, dass die darunter angegebenen Anforderungen eingehalten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

transport“ 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 varüeren. 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.



Lösungsvorschlag


Notaus
RoboterAus
Einschalten
ZufuhrA
ZufuhrBauteil
ZufuhrB
3
2
A
Bauteil B
zuerst A anbauen
zuerst B anbauen
RobotBereitA
A angebaut
B angebaut
RobotBereitB
Abtransport
dann A anbauen
dann B anbauen
fertig