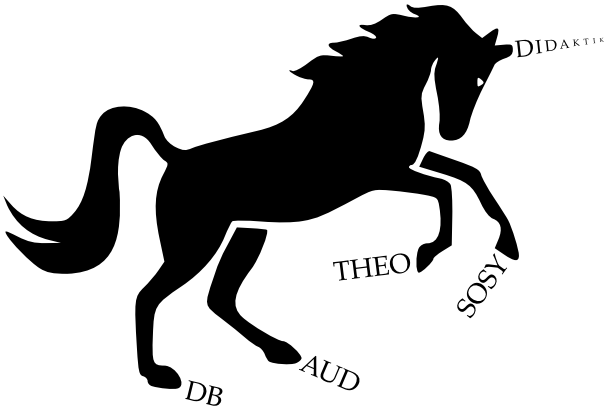
66116
Herbst
2018

Datenbanksysteme / Softwaretechnologie (vertieft)

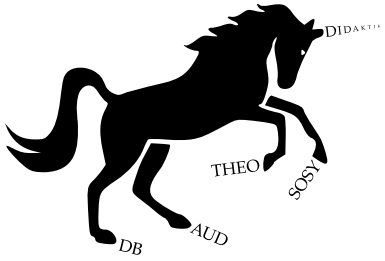
Aufgabenstellungen mit Lösungsvorschlägen



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 2 [Beatles]

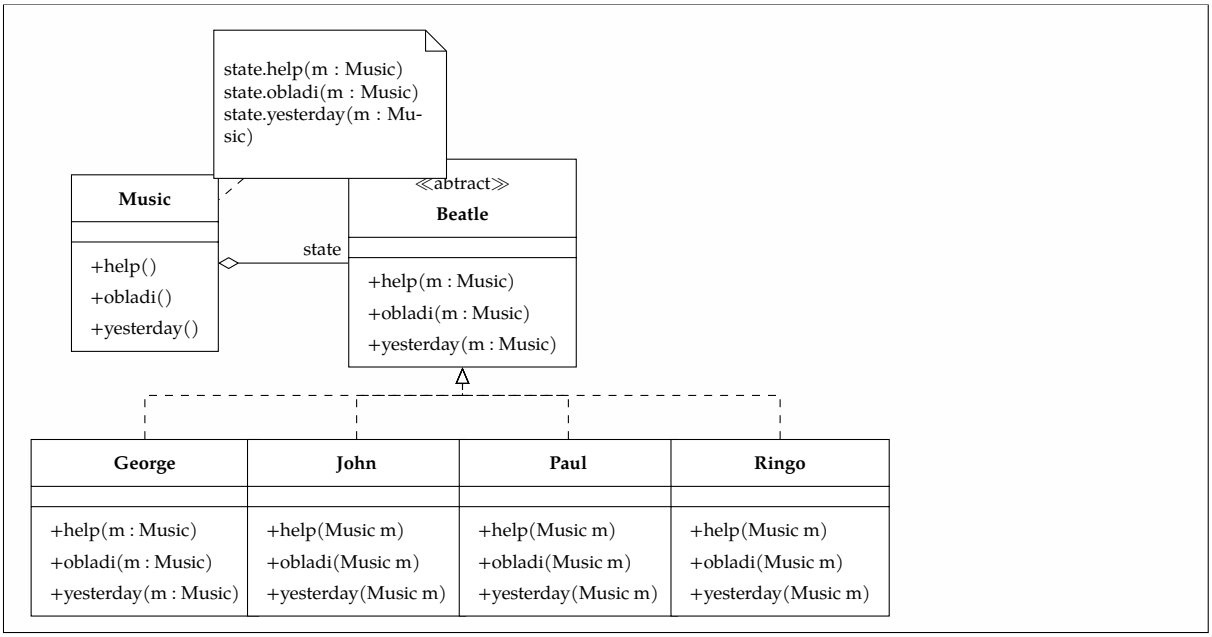
Gegeben sei das Java-Programm:

```
public 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());
  }
}
```

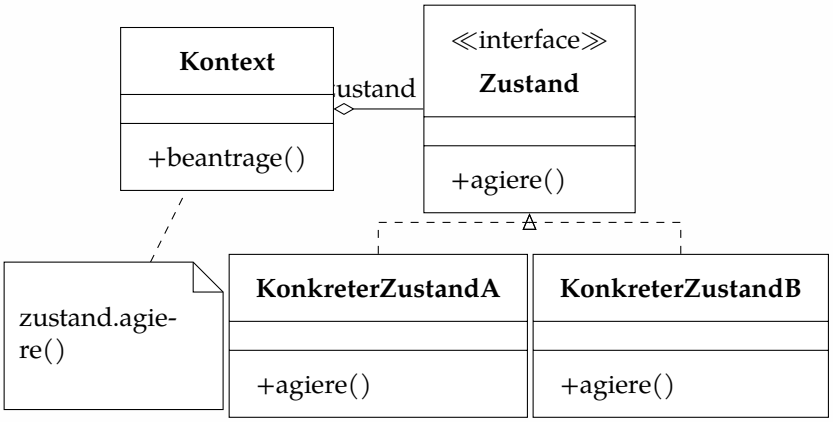
Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66116/jahr_2018/herbst/beatles/Music.java`

(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 implementiert ein Zustandsdiagramm, das das Verhalten von Objekten der Klasse `Music` beschreibt. Für die Implementierung wurde das Design-Pattern `STATE` angewendet.

(c) 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`.



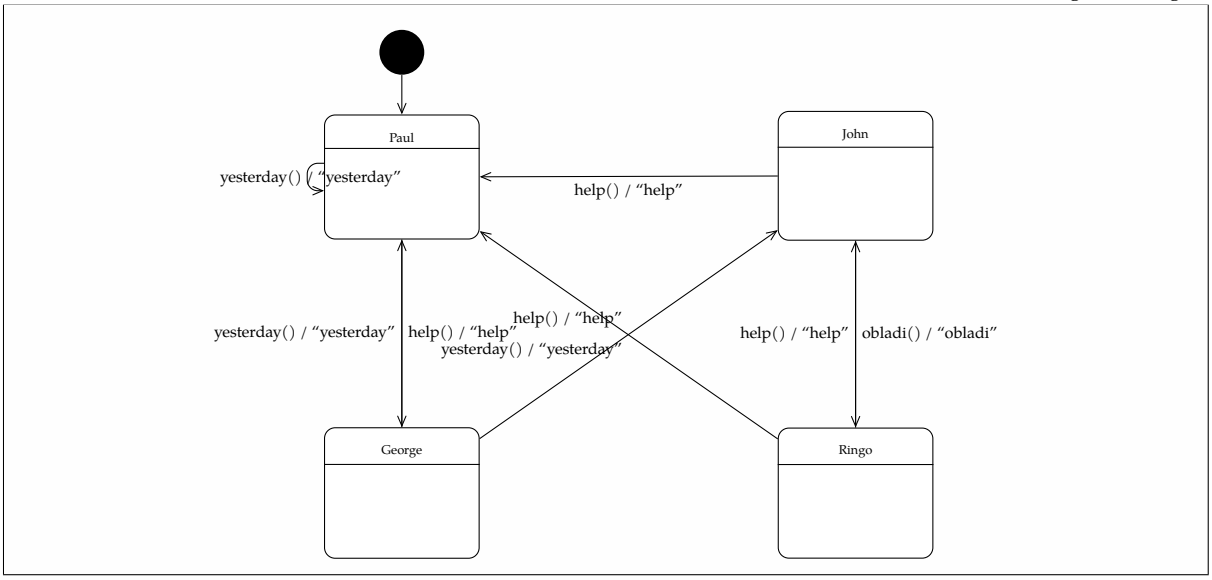
Kontext (Context) definiert die clientseitige Schnittstelle und verwaltet die separaten Zustandsklassen.

State (Zustand) definiert eine einheitliche Schnittstelle aller Zustandsobjekte und implementiert gegebenenfalls ein Standardverhalten.

KontreterZustand (ConcreteState) implementiert das Verhalten, das mit dem Zustand des Kontextobjektes verbunden ist.

(d) Zeichnen Sie das UML-Zustandsdiagramm (mit Anfangszustand), das von dem Programm 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.

Lösungsvorschlag



Teilaufgabe Nr. 2

Aufgabe 4 [Triathlon]

Gegeben sind folgende Relationen aus einer Datenbank zur Verwaltung von Triathlon-Wettbewerben.

```
Athlet : {[ ID, Vorname, Nachname ]}

Ergebnis : {[ Athlet[Athlet], Wettbewerb[Wettbewerb], Schwimmzeit, Radzeit, Laufzeit ]}

Wettbewerb : {[ Name, Jahr ]}
```

```
CREATE TABLE Athlet (
  ID INTEGER PRIMARY KEY,
  Vorname VARCHAR(20),
  Nachname VARCHAR(20)
);

CREATE TABLE Wettbewerb (
  Name VARCHAR(40) PRIMARY KEY,
  Jahr INTEGER
);

CREATE TABLE Ergebnis (
  Athlet INTEGER REFERENCES Athlet(ID),
```

```
  Wettbewerb VARCHAR(40) REFERENCES Wettbewerb(Name),
  Schwimmzeit INTEGER NOT NULL,
  Radzeit INTEGER,
  Laufzeit INTEGER,
  PRIMARY KEY (Athlet, Wettbewerb)
);

INSERT INTO Athlet VALUES
  (1, 'Boris', 'Stein'),
  (2, 'Trevor', 'Wurtele'),
  (3, 'Reichelt', 'Horst'),
  (12, 'Mitch', 'Kibby');

INSERT INTO Wettbewerb VALUES
  ('Zürichsee', 2018),
  ('Ironman Vichy', 2018),
  ('Challenge Walchsee', 2018),
  ('Triathlon Alpe d'Huez', 2017);

INSERT INTO Ergebnis VALUES
  (1, 'Zürichsee', 14, 10, 11),
  (2, 'Zürichsee', 13, 10, 11),
  (3, 'Zürichsee', 12, 10, 11),
  (12, 'Zürichsee', 11, 10, 11),
  (2, 'Challenge Walchsee', 12, 10, 11),
  (3, 'Challenge Walchsee', 11, 10, 11),
  (12, 'Triathlon Alpe d'Huez', 9, 10, 11);
```

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.

Lösungsvorschlag

```
CREATE TABLE IF NOT EXISTS Ergebnis (
  Athlet INTEGER REFERENCES Athlet(ID),
  Wettbewerb INTEGER REFERENCES Wettbewerb(Name),
  Schwimmzeit INTEGER NOT NULL,
  Radzeit INTEGER,
  Laufzeit INTEGER,
  PRIMARY KEY (Athlet, Wettbewerb)
);
```

(b) Schreiben Sie eine SQL-Anweisung, die die Radzeit des Teilnehmers mit der ID 12 beim Wettbewerb „Zürichsee“ um eins erhöht.

Lösungsvorschlag

```
-- Nur für Test-Zwecke
SELECT * FROM Ergebnis WHERE Athlet = 12 AND Wettbewerb = 'Zürichsee';

UPDATE Ergebnis
```

```sql
SET Radzeit = Radzeit + 1
WHERE Athlet = 12 AND Wettbewerb = 'Zürichsee';

-- Nur für Test-Zwecke
SELECT * FROM Ergebnis WHERE Athlet = 12 AND Wettbewerb = 'Zürichsee';
```

(c) Schreiben Sie eine SQL-Anweisung, die die Namen aller Wettbewerbe des Jahres 2018 ausgibt, absteigend sortiert nach Name.

Lösungsvorschlag

```sql
SELECT Name
FROM Wettbewerb
WHERE Jahr = 2018
ORDER BY Name DESC;
```

(d) Schreiben Sie eine SQL-Anweisung, die die Namen aller Wettbewerbe ausgibt, in der die durchschnittliche Schwimmzeit größer als 10 ist.

Lösungsvorschlag

```sql
SELECT Wettbewerb, AVG(Schwimmzeit)
FROM Ergebnis
GROUP BY Wettbewerb
HAVING AVG(Schwimmzeit) > 10;
```

(e) Schreiben Sie eine SQL-Anweisung, die die IDs aller Athleten ausgibt, die im Jahr 2017 an keinem Wettbewerb teilgenommen haben.

Lösungsvorschlag

```sql
(SELECT DISTINCT Athlet FROM Ergebnis)
EXCEPT
(SELECT DISTINCT Athlet FROM Ergebnis e, Wettbewerb w
WHERE e.Wettbewerb = w.name AND w.Jahr = 2017);
```

(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.

Lösungsvorschlag

vermutlich falsch

```sql
CREATE VIEW Gesamtzeiten AS
SELECT e.Athlet AS NameAthlet, e.Radzeit + e.Schwimmzeit + e.Laufzeit
AS Gesamtzeit, w.NameWettbewerb
FROM Ergebnis e, Wettbewerb w
WHERE e.Wettbewerb = w.Name
CREATE VIEW Sieger AS
SELECT g1.NameAthlet
FROM Gesamtzeiten g1, Gesamtzeiten g2
GROUP BY g1.NameWettbewerb
HAVING g1.Gesamtzeit < g2.Gesamtzeit
```

```sql
SELECT NameAthlet
FROM Sieger
GROUP BY NameAthlet
HAVING count(*) > 10;
```

(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.

Lösungsvorschlag

```sql
CREATE VIEW AthletenParis AS
SELECT a.Nachname, e.Schwimmzeit
FROM Athlet a, Ergebnis e INNER JOIN Wettbewerb W ON e.Wettbewerb = w.Name
WHERE w.Name = "Paris" AND a.ID = e.Athlet
SELECT a.Nachname COUNT(*) + 1 AS Platzierung
FROM AthletenParis a, AthletenParis b
WHERE a.Schwimmzeit < b.Schwimmzeit
GROUP BY a.Nachname
HAVING Platzierung <= 10;
```

(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.

Lösungsvorschlag

```sql
CREATE TRIGGER update_Ergebnis AFTER UPDATE ON Ergebnis AS
IF(UPDATE Schwimmzeit AND Schwimmzeit < 0) BEGIN UPDATE Ergebnis
SET Schwimmzeit = 0
WHERE (Athlet, Wettbewerb) IN (SELECT DISTINCT (Athlet, Wettbewerb) FROM inserted)
END;
```

Thema Nr. 2

Teilaufgabe Nr. 1

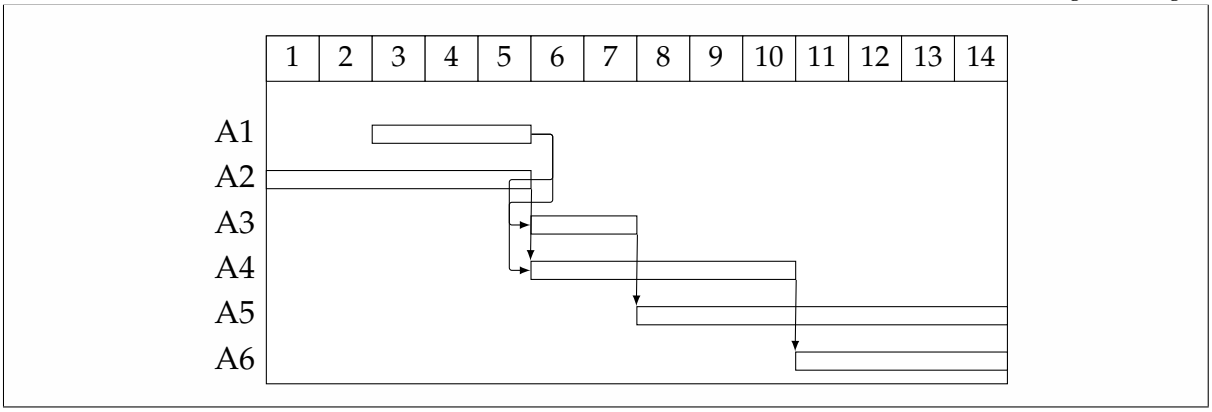
Aufgabe 3 [Gantt und zwei Softwareentwickler]

Ein Team von zwei Softwareentwicklern soll ein Projekt umsetzen, das in sechs Arbeitspakete unterteilt ist. Die Dauer der Arbeitspakete und ihre Abhängigkeiten können Sie aus folgender 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

(a) Zeichnen Sie ein Gantt-Diagramm, das eine kürzestmögliche Projektabwicklung beinhaltet.

Lösungsvorschlag



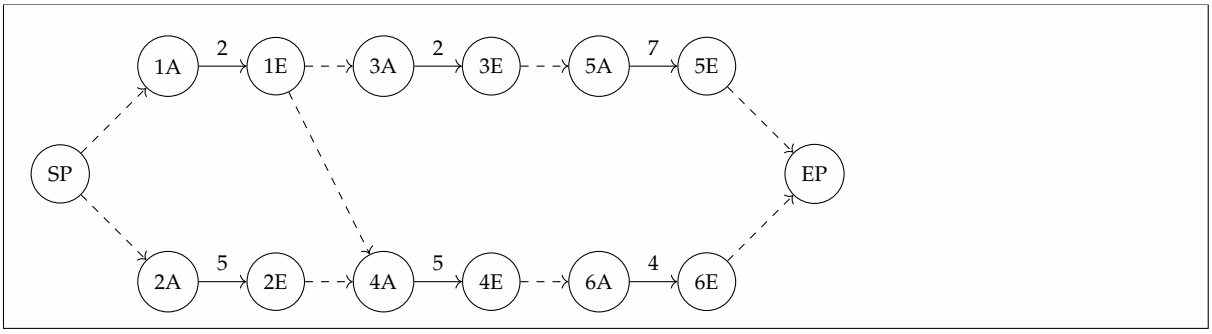
(b) Bestimmen Sie die Länge des kritischen Pfades und geben Sie an, welche Arbeitspakete an ihm beteiligt sind.

Lösungsvorschlag

Auf dem kritischen Pfad befinden die Arbeitspakete A2, A4 und A6. Die Länge des kritischen Pfades ist 14.

(c) Wandeln Sie das Gantt-Diagramm in ein CPM-Netzplan um.

Lösungsvorschlag



(d) Berechnen Sie für jedes Ereignis den *frühesten Termin* und den *spätesten Termin* sowie die *Gesamtpufferzeiten*.

Lösungsvorschlag

i	SP	1A	1E	2A	2E	3A	3E	4A	4E	5A	5E	6A	6E	EP
FZ_i	0	0	2	0	5	2	4	5	10	4	11	10	14	14
SZ_i	0	3	5	0	5	5	7	5	10	7	14	10	14	14
GP	0	3	3	0	0	3	3	0	0	3	3	0	0	0