5 ② 13 27

Beuth Hochschule für Technik Berlin, FB VI – Informatik und Medien
Software Engineering (SE2), Hauptklausur SoSe 2018
Termin: 13. Juli 2018 von 12^{15} bis 13^{45}

A

SOFTWARE ENGINEERING 2 — KLAUSUR SoSe 2018

Name, Vorname _____

Matrikelnummer _____

Pseudonym _____
freiwillig, wenn Sie möchten, dass Ihr Klausurergebnis im Internet veröffentlicht wird

Platznummer
wird vom Betreuer (**evtl.**) zu Beginn der Klausur vergeben

Erster (X) **Zweiter** ( ) **Letzter** ( ) Versuch

Note für Punkte	0 bis 55,	ab 60,	ab 70,	ab 75,	ab 80,	ab 85,	ab 90,	ab 95,	ab 100,	ab 105,	ab 110
1,0 für 120 P.	**5,0**	**4,0**	**3,7**	**3,3**	**3,0**	**2,7**	**2,3**	**2,0**	**1,7**	**1,3**	**1,0**

Lesen Sie zunächst alle Aufgaben sorgfältig durch. Sollten Sie Fragen haben, **können Sie diese in den ersten zehn Minuten laut stellen**. Spätere Fragen sind nicht mehr zulässig, denn laute Fragen stören, und leise Fragen widersprechen dem Gleichbehandlungsprinzip. Es sind keine Hilfsmittel zugelassen. Schreiben Sie Ihre Lösungen auf dieses Aufgabenblatt (beachten Sie auch die Rückseite!) sowie **auf höchstens zwei einseitig beschriebene Zusatzblätter** mit Ihrem Namen; kennzeichnen Sie die Aufgabennummer eindeutig. Schreiben Sie am besten mit Kugelschreiber (Bleistift ist nicht zulässig!). Für falsche oder unverständliche Lösungen bekommen Sie grundsätzlich keine Punkte. Wenn aber aus Ihren Notizen oder Bemerkungen ersichtlich ist, dass Ihr Gedankengang korrekt war, können Sie Teilpunkte erreichen. Sie verlieren diese Möglichkeit jedoch, wenn abschreiben oder Kommunikation während der Klausur nachgewiesen werden kann. Der Kern der Fragen wurde *kursiv* gesetzt. **Die Aufgaben sind ungefähr gleich aufwändig und jeweils 40 Punkte wert.**

Bearbeiten Sie bitte *unbedingt* die Aufgaben 2 und 4 (siehe Rückseite) und nur *eine* der beiden Aufgaben 1 und 3. Kennzeichnen Sie *deutlich*, welche Aufgabe Sie abgewählt haben. Geben Sie zum Aufgabenblatt höchstens *ZWEI EINSEITIG BESCHRIEBENE* Zusatzblätter ab. Weitere Blätter werden nicht bewertet.

1. AUFGABE

Zu dem Ihnen bekannten Java-Projekt *swXercise* erläutern Sie in Stichpunkten (kein Fließtext!):

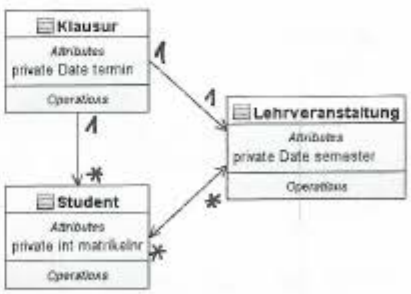
1. Welche Bedeutung haben die Annotationen `@Path`, `@POST`, `@Consumes`, `@Produces` und `@RolesAllowed` über einer REST-Methode des ViewControllers?
2. Erklären Sie folgendes *HQL-Statement* (inkl. des Doppelpunktes). Sie kennen das Statement aus einer *Named Query*.
   "`SELECT u FROM User u WHERE lower(u.profile.username) = lower(:username)`".

Lösung auf dem Extrablatt Nr. _____ **Punkte** (0 5 10 15 20 25 30 35 40)

2. AUFGABE — UNBEDINGT BEARBEITEN!

Alle Klassen des abgebildeten Klassendiagramms sollen persistent sein.

1. Ergänzen Sie zunächst das Diagramm um die *fachlich sinnvollen Multiplizitäten*. Beachten Sie hierbei: Student hört *mehrere* Lehrveranstaltungen und Lehrveranstaltung hat *mehrere* Studenten.
2. *Notieren* Sie die Entitätenklassen in Java *und annotieren* Sie sie gemäß der von Ihnen gewählten Multiplizitäten.
3. Begründen Sie stichpunktartig, warum das `mappedBy` in der von Ihnen gewählten Klasse besonders gut passt.
4. Modellieren Sie das *Entity-Relationship-Modell* (ER-Modell), welches ein Persistence Framework gemäß objekt-relationaler Abbildung (OR-Mapping) aus Ihren Entitätsklassen erzeugen würde.
5. Beschreiben Sie stichpunktartig, inwiefern sich das Materialisieren und Dematerialisieren verändern würde, wenn *statt der Getter-Methoden* die *Instanzattribute* annotiert würden (denken Sie in diesem Zusammenhang an den Mechanismus *Java Reflections*).



Lösung auf dem Extrablatt Nr. ①, ② **Punkte** (0 5 10 15 20 25 30 35 40)

Achtung, beachten Sie auch die Rückseite!

3. AUFGABE

Erläutern Sie tabellarisch in Stichpunkten (kein Fließtext!)

1. vier übliche *Anpassungen* am Klassendiagramm eines Analysemodells, um es in *ein Entwurfsmodell zu transformieren*,
2. den Grund dafür, dass abgeleitete Attribute im Entwurfsmodell kritisch hinterfragt werden müssen,
3. den Grund dafür, dass bidirektionale Assoziationen das Fehlerpotenzial in der Softwareentwicklung erhöhen.

Lösung auf dem Extrablatt Nr. (2)

Punkte (0 5 10 15 20 25 30 35 (40))

4. AUFGABE — UNBEDINGT BEARBEITEN!

Kreuzen Sie die korrekte Antwort an und geben Sie Ihre *Begründung in Stichworten* dazu. Jede Einzelfrage ist fünf Punkte wert. Ohne Begründung wird Ihre Antwort nicht bewertet.

( ) Richtig
(X) Falsch

4.1 Die *Kann-Kriterien* aus dem Pflichtenheft werden im Entwurfsmodell *nicht länger berücksichtigt*, da sie im fertigen Softwareprodukt ohnehin nicht realisiert sein werden. (5)

Grund sie werden wie Muss-Kriterien auch beachtet, man muss sich nur entscheiden, wann man sie umsetzt ✓

( ) Richtig
(X) Falsch

4.2 Ein *fachlich relevantes, eindeutiges* Attribut wie `kundenNr` oder `artikelId` ist als *Schlüsselattribut* (primary key) der Datenbanktabelle einer Entitätenklasse *besonders gut geeignet*. (5)

Grund diese können sich ändern, man sollte ~~nicht~~ generische Ids ✓ verwenden

( ) Richtig
(X) Falsch

4.3 Das Entwurfsmuster *Singleton* ist ein *klassenbasiertes Strukturmuster*. (5)

Grund es ist ein objektbasiertes Erzeugungsmuster

(X) Richtig
( ) Falsch

4.4 Im *Entwurfsmodell* ist für jede Operation die vollständige Signatur *anzugeben*. (5)

Grund es ist immerhin das "Spiegelbild des späteren Programms" :)

( ) Richtig
(X) Falsch

4.5 Die *Transformationstätigkeiten des OR-Mappings* bei Verwendung einer *objektorientierten* Datenbank sind die selben wie bei einer *relationalen* Datenbank. (5)

Grund bei objektorientierten Datenbanken gibt es logischerweise kein OR-Mapping

(X) Richtig
( ) Falsch

4.6 Die *Fachlogikschicht* in der *Drei Schichten Architektur* darf auf die *darunter liegenden Schichten* zugreifen, jedoch nicht auf die *darüber liegenden*. (5)

Grund man will es gekapselt halten, sie greift auf die Datenhaltungsschicht zu, nicht aber auf die Oberflächenschicht, die darüber liegt

(X) Richtig
( ) Falsch

4.7 Das *objektorientierte Entwurfsmodell* ist das *„Spiegelbild des späteren Programms"*. (5)

Grund man legt sich auf eine Programmiersprache fest und passt Klassen, Operationen dahingehend an

( ) Richtig
(X) Falsch

4.8 Bereits *zu Beginn der Softwareentwicklung* wird die Entscheidung darüber getroffen, welche *Programmiersprache* später zur Implementierung verwendet wird. (5)

Grund in der Analysephase noch nicht, erst in der Entwurfsphase

Punkte (0 5 10 15 20 25 30 35 (40)) Perfekt!

Aufgabe 2
2.)

(5) Multiplizitäten

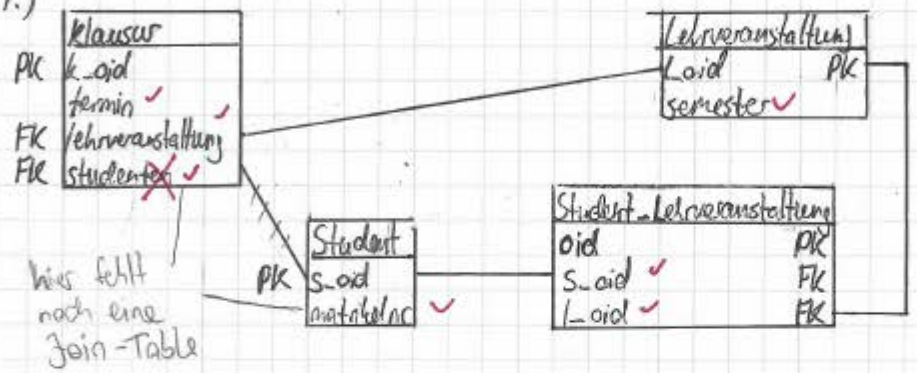
```
@Entity
public class Klausur {
  @Id @GeneratedValue
  private long k_oid;
  private Date termin;
  private Lehrveranstaltung lehrveranstaltung;
  private Collection<Student> studenten;

  @OneToOne
  public Lehrveranstaltung getLehrveranstaltung() {
    return lehrveranstaltung;
  }

  @OneToMany
  public Collection<Student> getStudenten() {
    return ~~Collection~~ studenten;
  }
  ...
}

@Entity
public class Lehrveranstaltung {
  @Id @GeneratedValue
  private long l_oid;
  private Date semester;
  private Collection<Student> studenten;
  ~~private G~~
  @ManyToMany(mappedBy = "lehrveranstaltungen")
  public Collection<Student> getStudenten() {
    return studenten;
  }
  ...
}

@Entity
public class Student {
  @Id @GeneratedValue
  private long s_oid;
  private int matrikelNr;
  private Collection<Lehrveranstaltung> lehrveranstaltungen;

  @ManyToMany
  public Collection<Lehrveranstaltung> getLehrveranstaltungen() {
    return lehrveranstaltungen;
  }
  ...
}
```

(5) mappedBy

Aufgabe 2

3.)
- mappedBy passt bei Lehrveranstaltung, denn es sagt damit, dass Student den Foreign-Key besitzt ?
- somit ist Student die "owning-side" und zeigt auf Lehrveranstaltung
- logisch überlegt ist die Lehrveranstaltung zuerst da und dann die Klasse Student, die dann auf Lehrveranstaltung zeigt (5) Begr.

4.)



PK Primary Key
FK Foreign Key

(5) Attribute
(5) Join Table
(5) FKs
15

5.)
- mit Java Reflektions kann man zur Laufzeit, die Eigenschaften von Klassen oder Methoden einsehen ✓
- damit wird geschaut, ob es Annotationen in der Klasse gibt
- werden die Attribute annotiert können diese gleich in die Datenbank geschrieben werden
- annotiert man Getter müssen diese zuerst die Attribute laden ✓

(5) Begr.

40

Aufgabe 3

1.)
(5) - die Navigation von Assoziationen bestimmen, uni- oder bidirektional
(5) - die vollständige Signatur von Operationen angeben
(5) - Klassen mit Paketierung fachlich oder technisch gruppieren
(5) - die Sichtbarkeit von Attributen public, private oder protected setzen

2.)
- diese werden nicht persistiert und sind nur sinnvoll, wenn Werte sich selten ändert (5)
- damit kann man beispielsweise teure Berechnungen seltener durchführen und das Ergebnis speichern (5)
- man kann Berechnungen auch im Konstruktor machen ✓

3.)
(5) - man muss dabei das mappedBy beachten
(5) - außerdem kann es zu JOIN-Tables kommen, die teuer sind
- man muss mehr Annotationen beachten ✓

40 :)