14 (3) 16 50

Beuth Hochschule für Technik Berlin, FB VI – Informatik und Medien
Software Engineering (SE2), Nachklausur SoSe 2014
Termin: 29. September 2014 von 16^{00} bis 17^{30}

SOFTWARE ENGINEERING 2 – NACHKLAUSUR SoSe 2014

Name, Vorname _____

Matrikelnummer _____

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

Platznummer _____ **Erster** (X) **Zweiter** ( ) **Letzter** ( ) **Versuch**
wird vom Betreuer (evtl.) zu Beginn der Klausur vergeben

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 110	**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 Blatt (beachten Sie auch die Rückseite!), bzw. auf nummerierte leere Blätter 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 (auf der Rückseite) und nur *eine* der beiden Aufgaben 1 und 3. Kennzeichnen Sie *deutlich*, welche Aufgabe Sie abgewählt haben.

1. AUFGABE

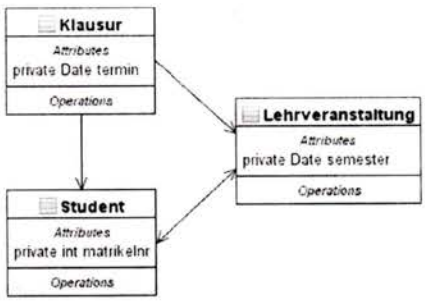
1. Beschreiben Sie tabellarisch in Stichpunkten den Begriff *Objektpersistenz* und grenzen Sie ihn vom *direkten Datenbankzugriff* mittels SQL-Anweisungen ab.
2. Erläutern Sie im Kontext JPA die Begriffe *Entity Manager*, *Persistence Context* und *Cache*. Erläutern Sie zwei *Transaktionszustände* im Cache.
3. Skizzieren Sie grob die Klassenstruktur eines *Virtual Proxy-Musters* und erläutern Sie daran, wie mit dessen Hilfe ein *„teurer" Datenbankzugriff* vorübergehend *vermieden* werden kann.

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

2. AUFGABE – UNBEDINGT BEARBEITEN!

Alle drei Klassen des abgebildeten Fachklassendiagramms aus einem Entwurfsmodell sollen persistent sein.

1. Ergänzen Sie das Klassendiagramm zunächst um *fachlich sinnvolle Multiplizitäten*.
2. Notieren Sie die Entitätenklassen in Java und annotieren Sie sie gemäß der von Ihnen gewählten Multiplizitäten. „Student" ist hierbei die führende Klasse („owning side").
3. Modellieren Sie das Entity-Relationship-Modell (ER-Modell), welches ein Persistence Framework gemäß objekt-relationaler Abbildung (OR-Mapping) aus Ihren Entitätenklassen erzeugen würde. Machen Sie deutlich, wo sich darin die *Fremdschlüssel befinden*.



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

Notieren Sie stichpunktartig (kein Fließtext!):

1. Beschreiben Sie mit vier Beispielen, in welchen Situationen es sich *nicht empfiehlt*, das einzusetzende Datenbanksystem *selbst zu entwickeln*, sondern es sinnvoller ist, ein *Standardprodukt* zu wählen.
2. Erläutern Sie das Konzept der *Lazy Creation* im *Singleton*-Entwurfsmuster, und vergleichen Sie es mit der *Lazy Materialization* mithilfe des *Proxy*-Musters in einem *Persistenz-Framework*.

Lösung auf dem Extrablatt Nr. 1/3

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.

(X) Richtig
( ) Falsch

4.1 Durch die Verwendung des Entwurfsmusters *Strategie* erhöht sich *die Anzahl der Objekte*.

Grund Für jeden Algorithmus wird eine neue Klasse erstellt und darin neue Objekte erzeugt. (5)

(X) Richtig
( ) Falsch

4.2 *Verwaltungsmethoden* wie `get()` und `set()` sind im *Klassendiagramm eines Entwurfsmodells* zu modellieren.

Grund ebenso wie insert() und update() (5)

( ) Richtig
(X) Falsch

4.3 Die *Kann-Kriterien* aus dem Pflichtenheft werden im Entwurfsmodell *nicht länger berücksichtigt*.

Grund Sie werden ~~berücksichtigt~~, es sollte aber baldmöglichst entschieden werden, ob sie realisiert werden. (5)

( ) Richtig
(X) Falsch

4.4 Objektorientierte Vererbung lässt sich *nicht auf relationale Datenmodelle* abbilden.

Grund Es gibt 3 Arten: JOINED, TABLE-PER-CLASS und SINGLE-TABLE (5)

( ) Richtig
(X) Falsch

4.5 Ein Web-Server funktioniert nach dem *HTTP-Request-Response-Paradigma*. Das klassische *Model-View-Controller-Konzept* lässt sich hier nicht anwenden.

Grund es lässt sich ab HTML5 anwenden (also MVC) ist nur richtig, wenn der Webserver ohne HTML5 arbeitet (5)

( ) Richtig
(X) Falsch

4.6 Eine Klasse kann entweder *ganz oder gar nicht* persistiert sein.

Grund ~~es~~ Attribute können transient sein (5)

(X) Richtig
( ) Falsch

4.7 In der Programmiersprache *Java* gibt es *keine Mehrfachvererbung*.

Grund eine Klasse kann nur von einer anderen Klasse erben (5)

(X) Richtig
( ) Falsch

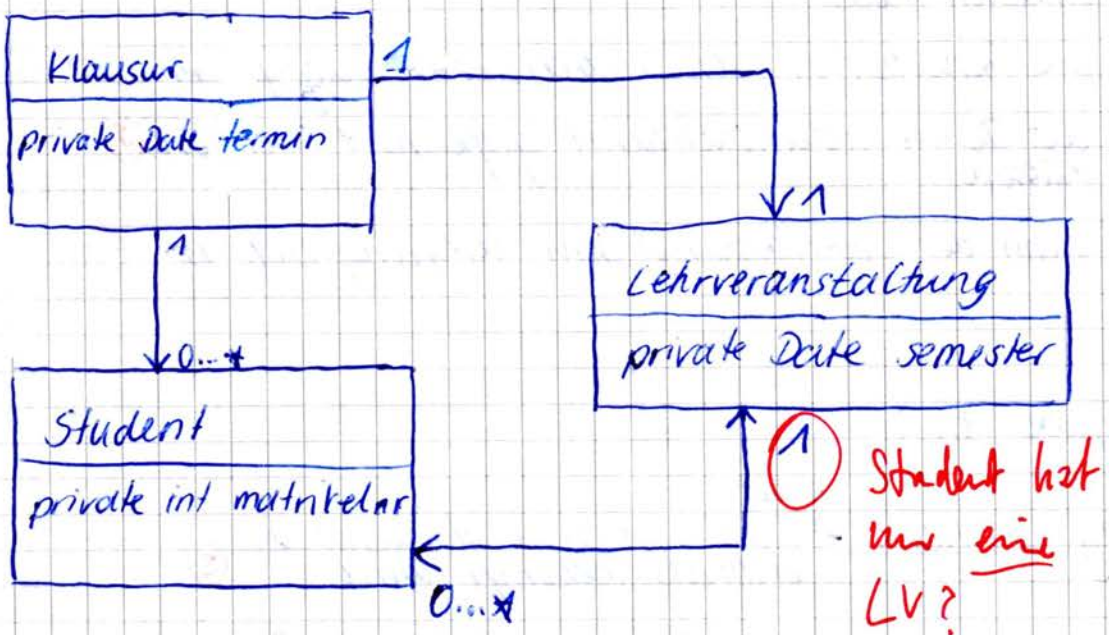
4.8 Im *Klassendiagramm des Entwurfs* sollten möglichst viele *Assoziationen unidirektional* sein.

Grund Bidirektionalität ist sehr aufwändig Gebäude 10... (5)

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

SE-II Klausur

Aufgabe 2. 1))



Aufgabe 2. 2) //imports

(S) @Entity
public class Student implements Serializable {
(S) { @Id
@GeneratedValue
private long id;
private int matrikelnr;
@ManyToOne @... gilt immer nur für ein Attribut
private Collection<Klausur> k;
private Collection<Lehrveranstaltung> l;
} @ManyToOne
}

Rest auf Seite 2

SE-II Klausur

Aufgabe 2.2 - Fortsetzung)

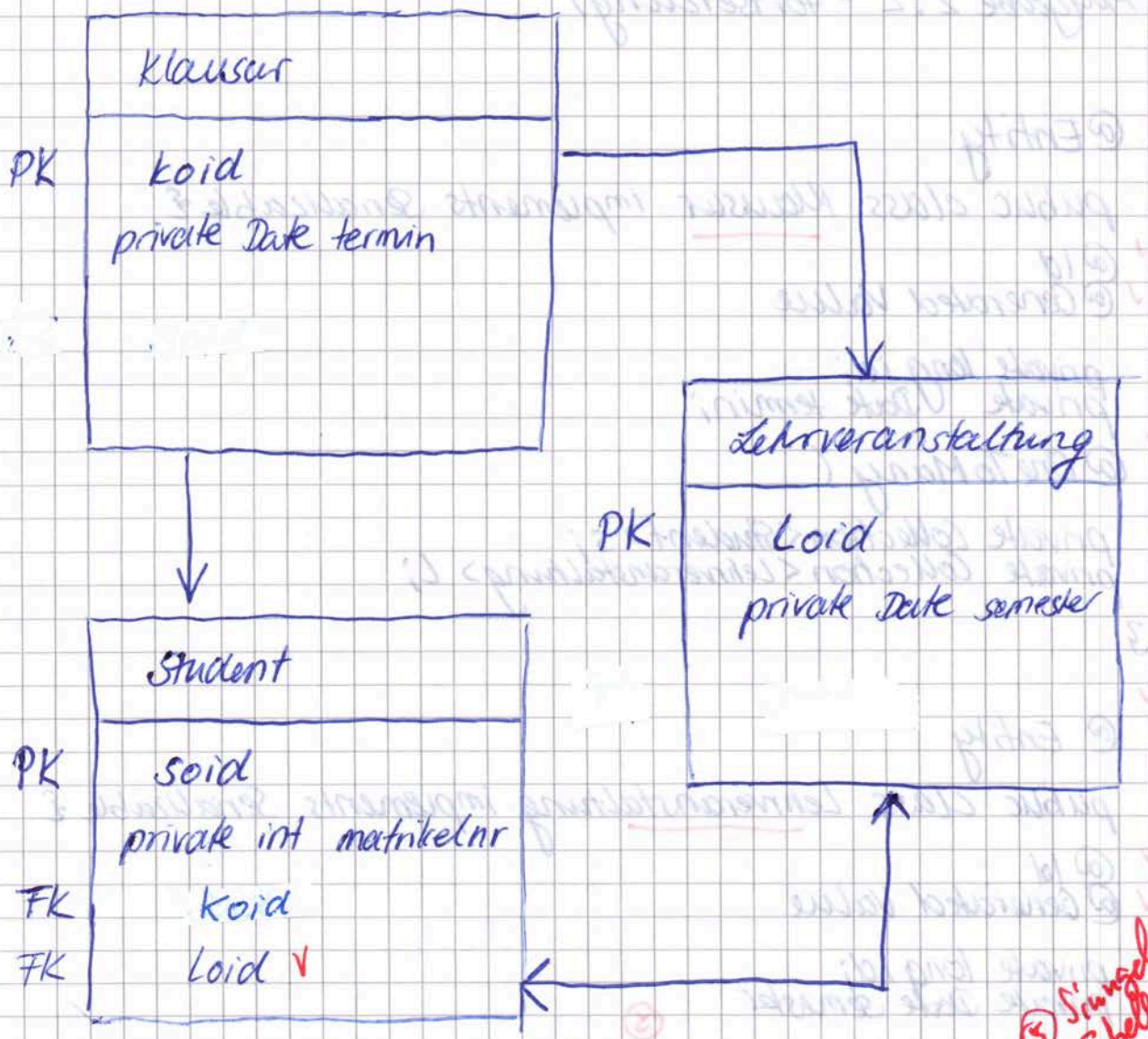
```
@Entity
public class Klausur implements Serializable {
@Id
@GeneratedValue
private long id;
private Date termin;
@OneToMany
private Collection<Student> s;
private Collection<Lehrveranstaltung> L;
}

@Entity
public class Lehrveranstaltung implements Serializable {
@Id
@GeneratedValue
private long id;
private Date semester
@OneToOne (mappedBy="Lehrveranstaltung")
private Collection<Klausur> k;
private Collection<Student> s;
}
```


Bitte wenden!

→

PK ≙ Primary Key (Primärschlüssel)
FK ≙ Fremdschlüssel (Foreign Key)

Aufgabe 2.3)



Student kennt seine Klausur
Student kennt seine Lehrveranstaltung

③ Sinnvolle Tabellen gefunden
③ FK gefunden
③

15

30

Aufgabe 3.1)

- Wenn mehrere Personen parallel mit dem Datenbestand arbeiten muss (5)
- alle Benutzer brauchen dedizierte Zugriffsberechtigungen (5)
- die Daten redundanzarm gespeichert werden müssen (5)
- wenn der Datenbestand sehr umfangreich ist (5)

Aufgabe 3.2)

Lazy Creation: Objekt
- wird erst erzeugt, wenn es erstmals benötigt wird (5)
- Rechenzeit und Speicherplatz werden erst in Anspruch genommen, wenn das Objekt erzeugt wurde
- getInstance()-Methode wird ~~erst~~ einmalig vom Konstruktor erzeugt, wenn das Objekt benötigt wird (5)
- Objekt kann öfter verwendet werden, es wird aber nur ~~ein Mal~~ erzeugt (also ein Singleton)
- es gibt dieses Objekt nur ein einziges Mal

Virtual-Proxy-Muster

Optimierung der Materialisierung ~~durch~~ ermöglicht Lazy Materialization ✓

- hier wird ein Objekt verwendet, wenn es benötigt wird, wenn mit diesem einen Objekt weitere Objekte im Zusammenhang stehen, werden diese virtuell durch den Virtual-Proxy erzeugt (5)
- wenn diese anderen Objekte dann tatsächlich benötigt werden, werden sie realisisch ~~dargestellt~~ erzeugt ✓

SE-II Klausur

Aufgabe 3.2 - Fortsetzung)

Vergleich:

- mit dem VirtualProxy können mehrere Objekte bei Bedarf realistisch erzeugt werden
- mit Lazy Creation kann nur ein einziges Objekt erzeugt werden?
- Speicherbedarf und Rechenzeit sind beim VirtualProxy höher, sofern mehr als ein Objekt erzeugt wird (bei Lazy Creation kann ja nur ein Objekt erzeugt werden) ②

(40)

Prima!