13 34 (5)

Berliner Hochschule für Technik, FB VI – Informatik und Medien
***Software Engineering (SE2)**, Hauptklausur SoSe 2024*
Termin: 19. Juli 2024 von 12[15] bis 13[45]

SOFTWARE ENGINEERING 2 KLAUSUR SoSe 2024

B

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 29,	ab 30,	ab 35,	ab 40,	ab 45,	ab 50,	ab 55,	ab 60,	ab 65,	ab 70,	ab 75
1,7 für 68	**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 höchstens *zwei einseitig* beschriebene DIN A4-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 Aufgabe 2 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 DIN A4-Zusatzblätter ab. Weitere Blätter und andere Formate werden nicht bewertet.

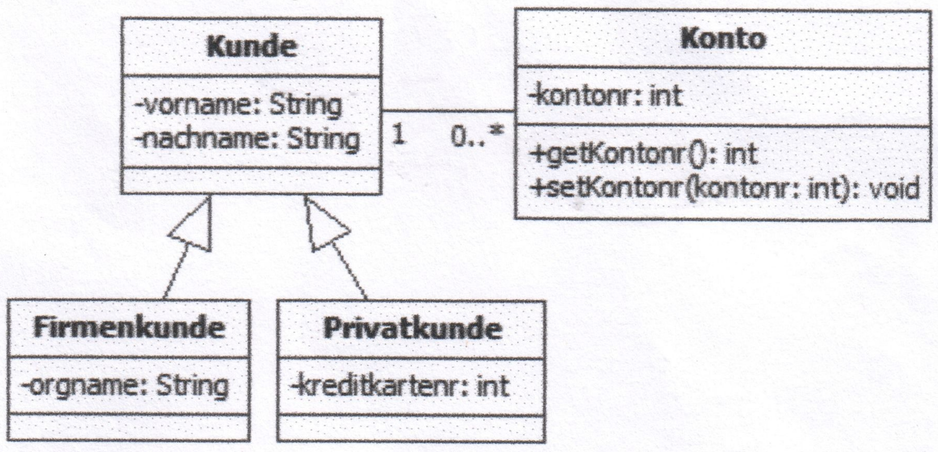
1. AUFGABE

1. *Beschreiben* (nicht nur nennen!) Sie drei übliche *Anpassungen oder Erweiterungen am Fachklassendiagramm* eines Analysemodells, um dieses in *ein Entwurfsmodell zu transformieren*. Wählen Sie hierzu bitte nicht die „Anpassung der Bezeichner an die ausgewählte Programmiersprache", denn das wäre zu einfach. ;)
2. Was ist ein *abgeleitetes Attribut* und was eine *abgeleitete Assoziation*? Und was haben beide gemeinsam?
3. Warum sollten *bidirektionale Assoziationen* im Klassendiagramm des Entwurfsmodells vermieden werden?
4. Beschreiben Sie mindestens zwei Vorteile, die sich aus der Verwendung der Drei-Schichten-Architektur ergeben.

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

2. AUFGABE UNBEDINGT BEARBEITEN!

Die abgebildeten Klassen sollen *persistent werden*:



1. *Notieren* Sie die Entitätenklassen in Java *und annotieren* Sie sie gemäß der bereits vorgegebenen Multiplizitäten und Navigationsrichtungen. Überlegen Sie genau, an welchem Objektattribut (oder welcher Getter-Methode) das *mappedBy*-Attribut am sinnvollsten ist.
2. Verwenden Sie die Vererbungsstrategie *JOINED*.
3. Begründen Sie stichpunktartig, weshalb das *mappedBy-Attribut* in der von Ihnen hierfür gewählten Klasse *besonders sinnvoll* ist.
4. Modellieren Sie das *Entity-Relationship-Modell*, welches ein Persistenz-Framework gemäß objekt-relationaler Abbildung aus Ihren Entitätenklassen generieren würde. Machen Sie darin besonders deutlich, wo sich die *Primärschlüssel* (primary keys), *Fremdschlüssel* (foreign keys) und weitere *Attribute befinden*.
5. Beschreiben Sie stichpunktartig, worin sich der Vorgang des Materialisierens und Dematerialisierens der Objekte unterscheiden würde, wenn *statt der Getter-Methoden* die *Objektattribute* annotiert würden. Denken Sie in diesem Zusammenhang auch an den Mechanismus *Java Reflections*.

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

3. AUFGABE

Beschreiben Sie stichpunktartig (kein Fließtext!):

1. „GoF"-Muster werden gemäß ihrer *Aufgabe kategorisiert*. Welche Aufgabenkategorien kennen Sie? Beschreiben Sie sie.
2. Welche drei Merkmale machen eine Klasse zu einem *Singleton-Muster*?
3. Erklären Sie die *Aufgaben* der im Observer-Pattern verwendeten Operationen.
4. Was sind die *Gemeinsamkeiten* und die Unterschiede der beiden Entwurfsmuster *Fabrikmethode* und *Strategie*?

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

Die folgende Fläche dürfen Sie gern für Ihre eigenen Notizen verwenden. Sie geht NICHT in die Bewertung ein.

Berliner Hochschule für Technik, FB VI – Informatik und Medien
***Software Engineering (SE2)**, Hauptklausur SoSe 2024*
Termin: 19. Juli 2024 von 12^{15} bis 13^{45}

Bewertungsbogen der SE2-Klausur im Sommersemester 2024

Name, Vorname

Persistenzaufgabe „Kunde/Konto“ (unbedingt zu bearbeiten)

(3) Annotation `@OneToMany` in Klasse *Kunde*, sowie `@ManyToOne` in Klasse *Konto*
(5) Annotation `@OneToMany(mappedBy="kunde")` in Klasse *Kunde*, Attribut in *Konto*
(5) Annotation `@Inheritance(strategy=...JOINED)` in Basisklasse *Kunde*
(5) Begründung, warum *mappedBy* in Klasse *Kunde* besonders sinnvoll ist (o.ä.)
(5) Alle Tabellen *Konto*, *Kunde*, *Firmenkunde* und *Privatkunde* (ohne Join Tables) gefunden
(3) Attribute in den richtigen Tabellen eingetragen (*vorname*, *nachname*, *orgname*, ...)
(5) Fremdschlüssel *kunde* in Tabelle *Konto* eingetragen, Fremdschlüssel auf die Basisklasse
(5) Sinnvolle Erklärung für *Objektattribut vs. Getter-Methoden* gegeben

---

36 Punkte von 40 (max. 8 x 5)

Entwurfsmuster

( ) *Strukturmuster* beschrieben (*Strukturen* sind Assoziationen oder Vererbungen)
( ) *Verhaltensmuster* beschrieben (Verhalten wird durch *Methoden* definiert)
( ) *Erzeugungsmuster* beschrieben (Objekte werden durch das Muster *instanziiert*)
( ) Merkmale des Singletons gut erläutert (Klassenattr., stat. `getInstance()`, priv. Konstrukt.)
( ) `attach()` und `detach()` im Model skizziert (Liste zum Eintragen der Views)
( ) `notify()` im Model und Empfang von `update()` im View bzw. Controller skizziert
( ) Fabrikmethode *und* Strategie produzieren/liefern ein Objekt *einer ihrer* Unterklassen (o.ä)
( ) Fabrikmethode bestimmt *allein*, welcher Objekttyp tatsächlich produziert wird (o.ä.)

_____ Punkte von 40 (max. 8 x 5)

Erläuterungen „Entwurfsmodell“

(4) Erste Anpassung oder Erweiterung am Fachklassendiagramm in Ordnung
(4) Zweite Anpassung oder Erweiterung am Fachklassendiagramm in Ordnung
(4) Dritte Anpassung oder Erweiterung am Fachklassendiagramm in Ordnung
(3) Bedeutung von *abgeleitetem Attribut* und *abgeleiteter Assoziation* gut erläutert
(2) Gemeinsamkeit(en) von *abgeleitetem Attribut* und *abgeleiteter Assoziation* gut erläutert
(5) *Bidirektionale Assoziationen* vermeiden (z.B. Join Tables, Fehlerpotenzial, Synchronisation)
(5) Erster Vorteil aus der Verwendung der Drei-Schichten-Architektur (z.B. Austauschbarkeit)
(5) Zweiter Vorteil aus der Verwendung der Drei-Schichten-Architektur (z.B. Wartbarkeit)

---

32 Punkte von 40 (max. 8 x 5)

19.07.24

2. Aufgabe

1. @Entity ✓
   @Inheritance (strategy = InheritanceStrategy.JOINED) ✓
   ~~public class Kunde~~
   public class Kunde {
   ✓ @Id @GeneratedValue
   private long id;
   private String nachname;
   @NotNull
   private List<Konto> konten;
   @NotNull
   ✓ @OneToMany(mappedBy = "kunde")
   public List<Konten> getKonten() {
   return konten;
   }
   }

✓ @Entity
public class Konto {
✓ @Id @GeneratedValue
private long id;
private int kontonr;
@NotNull private Kunde kunde;
public int getKontonr() {
return kontonr;

public void setKontonr (int) {
int kontonr

public Kunde getKunde() {
return kunde;

@ManyToOne

Wo geht's hier weiter?

Kunde ✓
- str1: vorname ✓
- str2: nachname ✓
- oid_Kunde_PK

Konto ✓
- int: kontonr ✓
- oid_kunden_FK ✓
+ getKonto()
+ getKontonr()

Firmenkunde ✓
= str1: organame
- oid_FKunde_PK
~~- str1: vorname~~
~~- str2: nachname~~
~~- kunde_FK~~ ✓

Privatkunde ✓
- int: kreditkarte
- oid_PKunde_PK
~~- str1: vorname~~
~~- str2: nachname~~
- kunde_FK ✓

@Entity
public class Privatkunde extends Kunde
@Id @GeneratedValue
private long id;
private int kreditkarte;

@Entity ~~Privatkunde~~
public class Privatkunde extends Kunde {
✓ @Id @GeneratedValue
private long id;
private int kreditkarte;
}

2. Aufgabe
3. Mapped by Attribut rollte da verwendet werden, wo es @keinen Fremdschlüssel gibt. ✓
- bei der To Many Seite ~~schreiben~~ schreiben, wo Join-Table nicht verwendet wird ✓

5a Java Reflektion hat Zugriff auf die Attribute, damit es die Attribute dematerialisieren kann. und materialisieren
- die Attribute der get-Methoden müssen erst mit getter geholt werden, bevor sie in die Datenbank geschrieben werden können und oder gelesen (Instanzattribute (Objektattribute)). Objekte werden direkt in Datenbank geschrieben

SE2-Klausur 19.07.24

Aufgabe 1)

1. -Benutzen von Pattern: -man sollte je nachdem wie die Funktionalität ist, passende Entwurfsmuster benutzen

-Benutzen von Packages (Packetisierung): -man sollte Klassen sinnvoll in packages verteilen das heißt?

-Benutzen von Container-Klassen: man sollte sinnvolle Container Klassen erstellen, wenn die auch notwendig sind ?

2. -abgeleitetes Attribut: -wird aus einem anderen Attribut abgeleitet ✓
z.B. Gesamtsumme einer Lieferung kann als abgeleitetes Attribut aus den Einzelpreisen berechnet werden ✓

-abgeleitete Assoziation: ist eine Beziehung zwischen mindestens zwei Klassen, z.B. ist Klasse "Ausleihe" eine abgeleitete Assoziation von Klasse "Leser" und Klasse "Buch" das ist eine assoziative Klasse", keine abgeleitete Assoziation!

Gemeinsamkeit: beide werden auf Beziehung/Bezug zum anderen Attribute/Assoziation berechnet

3. bidirektionale Assoziationen erhöht nur die Komplexität des Programms ✓

-durch engere Kopplung der Klassen, müssen meist auch Änderungen in anderen Klassen vorgenommen werden, somit gelt ✓

höherer Aufwand einher

-kann zu Inkonsistenz führen, → heißt die Daten stimmen nicht in beiden Richtungen überein ✓

-höherer Aufwand bei der Programmierung (Implementierung) ✓

4. -dadurch das die Schichten voneinander getrennt sind und jede Schicht eine gewisse Aufgabe hat, gibt es eine gute Strukturierung ✓

-wenn es eine Änderung der einen Schicht gibt. z.B. ~~Anwendungsschicht hat es~~ Datenzugriffsschicht hat es keine Auswirkung auf die Anwendungsschicht, da sie klar voneinander getrennt sind. ✓

-klarer Schutz der Sicherheit: Wichtige und sensible Daten können in Datenzugriffschicht gesichert werden, da die Anwendungsschicht nur auf für sie bereitgestellte Daten zurückgreifen kann. ✓

2/2