Technische Universität Berlin
Institut für Softwaretechnik und Theoretische Informatik
FG Softwaretechnik
Franklinstr. 28/29
10587 Berlin

Helke
Mertgen
Dobrev

MPGI 3

Muster-Klausur A

Wintersemester 2008/2009
19. Februar 2009

Prüfen Sie zunächst, ob dieses Exemplar vollständig ist (*8* beidseitig bedruckte Blätter).

Tragen Sie auf diesem Titelblatt und darüber hinaus auf allen Blättern, die Sie für Ihre Niederschrift verwenden, Ihren Namen und Ihre Matrikelnummer ein. Zum Bestehen der Klausur sind mindestens **25 Punkte** notwendig. Insgesamt sind 50 Punkte möglich. Die Bearbeitungszeit beträgt 75 Minuten. In der Klausur sind außer einem beschriebenen DIN-A4-Blatt **keine Hilfsmittel** zugelassen. Verwenden Sie ausschließlich das ausgeteilte Klausur-Papier. Es dürfen nur **permanent-schwarze oder -blaue Stifte** zum Lösen der Aufgaben verwendet werden.

Viel Erfolg!

Name, Vorname: _____

Matrikelnummer: _____

Studienrichtung: _____

MPGI3-Übungen habe ich im _____ besucht.
(z.B. WS 2008/09)

Aufgabe	1	2	3	4	5	6	Gesamt
Maximal:	7	5	8	12	12	6	50
Erreicht:							

Aufgabenstellung (Übersicht)

Das auf Seite 2 beschriebene System soll nach den in der Veranstaltung vermittelten Methoden entwickelt werden:

1. Fügen Sie im gegebenen Klassenmodell für den Gegenstandsbereich auf Seite 3 die Klassen *Modeprodukt* und *Elektronikprodukt* ein und ergänzen Sie alle fehlenden Multiplizitäten. *7 Punkte*
2. Vervollständigen Sie das Use-Case-Modell auf Seite 4, indem Sie die Akteure des Systems bestimmen und sie den Funktionsgruppen zuordnen. Fügen Sie der Vorgabe bitte keine neuen Use-Cases hinzu. Beziehungen zwischen den Funktionsgruppen dürfen hinzugefügt werden. *5 Punkte*
3. Vervollständigen Sie das vorgegebene Systemklassenmodell auf Seite 5, indem Sie Akteure, Klassen und Assoziationen hinzufügen. Kennzeichnen Sie die Klassen als *Boundary*, *Control* oder *Entity*. Multiplizitäten und Attribute müssen **nicht** angegeben werden. *8 Punkte*
4. Vervollständigen Sie das Operationsschema für die auf Seite 6 dargestellte Systemoperation *angebot_kommentieren*. *12 Punkte*
5. Vervollständigen Sie das vorgegebene Kommunikationsdiagramm für die Systemoperation *angebot_kommentieren* auf Seite 8 an. *12 Punkte*
6. Implementieren Sie auf Seite 9 die Methode *zaehleKommentare* für die Klasse *Produkt*. Beachten Sie die dort angegebenen Vorgaben. *6 Punkte*

Achten Sie auf Konsistenz zwischen allen erzeugten Modellen.

Webportal zur Bewertung von Produktangeboten

Das Webportal *OfferComments.de* beauftragt Sie mit der Entwicklung eines Softwaresystems zur Verwaltung und Bewertung von Produktangeboten verschiedener Firmen.

Zentrale Rolle in diesem Softwaresystem spielen die *Produktmanager*, die in einem *Büro* in Berlin-Mitte arbeiten. Vertreter der *Firmen* kommunizieren direkt mit den Produktmanagern, wenn ihre *Produktangebote* im Webportal veröffentlicht, aktualisiert oder gelöscht werden sollen. Die Kommunikation geschieht hauptsächlich per E-Mail und Fax, manchmal kommen aber die Firmenvertreter auch persönlich ins Büro des Webportals. Zwecks Aufwandsreduzierung, dürfen pro Firma höchstens **30** Produktangebote veröffentlicht werden. Zu einem bestimmten *Produkt* hat eine Firma höchstens ein Produktangebot. Wenn eine Firma ein ganz neues Produkt auf den Markt bringen möchte, muss einer der Produktmanager zuerst dieses Produkt ins System eintragen. Unter Anderem hat ein Produkt eine eindeutige Identifikationsnummer (*id*), einen kurzen Namen und eine Produktbeschreibung. Betrifft das zu registrierende Produktangebot ein Produkt, das bereits im System bekannt ist, so entfällt dieser Schritt. Da das Webportal *OfferComments.de* an jüngere Leute gerichtet ist, werden ausschließlich Produktangebote der Marktbereiche *Elektronik* und *Mode* angenommen und veröffentlicht.

Für das Webportal arbeiten zusätzlich mehrere studentische Hilfskräfte als *Reviewer* von zu Hause. Sie loggen sich unregelmäßig auf eine speziell eingerichtete Webseite (hier *ReviewerPortal* genannt) ein. Ihre Arbeit besteht darin, Produktangebote zu kommentieren. Hierzu müssen sie zuerst das zu kommentierende Produktangebot aus einer nach Produkten gruppierten Liste auswählen. Zu jedem Produktangebot können mehrere *Kommentare* abgegeben werden. Ein Kommentar besteht aus einem längeren Text, der die Vor- und Nachteile von diesem Angebot beschreibt. Im System muss immer nachvollziehbar sein, welche Kommentare ein Reviewer geschrieben hat.

Die *Benutzer* des Systems sind Personen mit Zugang zum Internet, die die *Webseite* von *OfferComments.de* in ihrem Webbrowser öffnen. Ein Benutzer kann die Produktangebote der verschiedenen Firmen recherchieren und die abgegebenen Kommentare lesen. Hierzu muss er zuerst das gewünschte Produkt aus einer Liste auswählen. Anschließend bekommt er eine Liste mit allen Produktangeboten zu diesem Produkt, die nach Preis geordnet sind. Erst dann kann er die Kommentare zu den einzelnen Angeboten sehen. Für ihre Recherche müssen sich die Benutzer nicht gesondert registrieren oder einloggen.

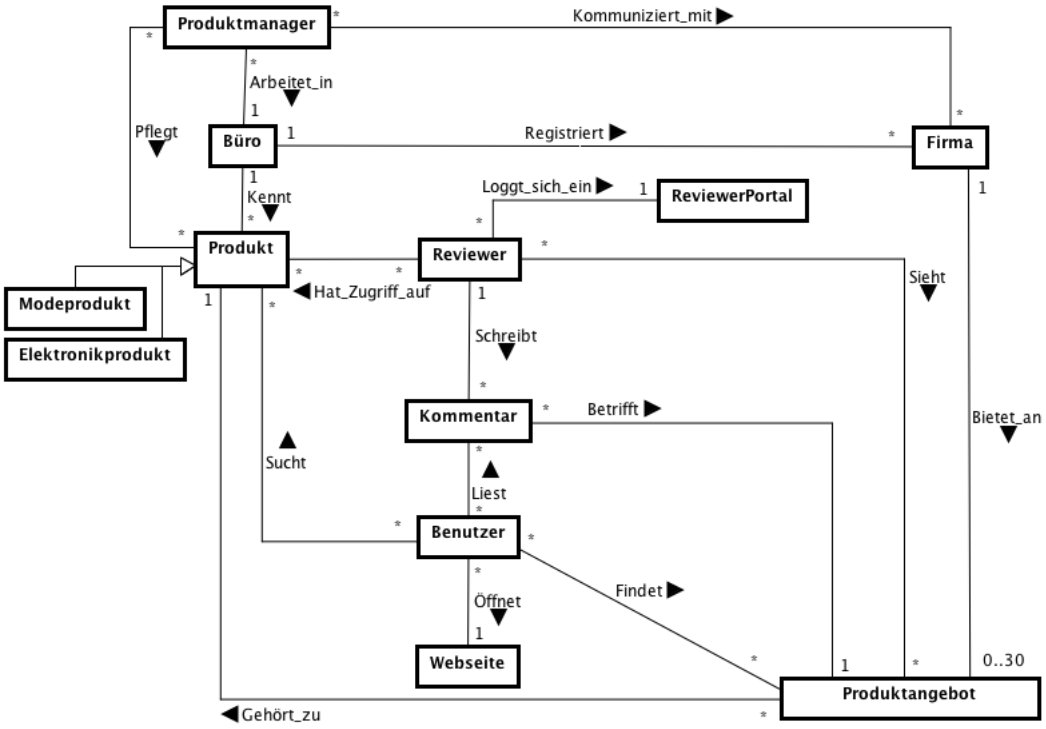
1. Klassenmodell für den Gegenstandsbereich

7 Punkte

- Fügen Sie im unten gegebenen Klassenmodell für den Gegenstandsbereich die Klassen *Modeprodukt* und *Elektronikprodukt* ein.
- Ergänzen Sie alle fehlenden Multiplizitäten.

Die auf dem Diagramm gezeichneten Dreiecke (▶) kennzeichnen die Leserichtung der Assoziationen.

Lösung:



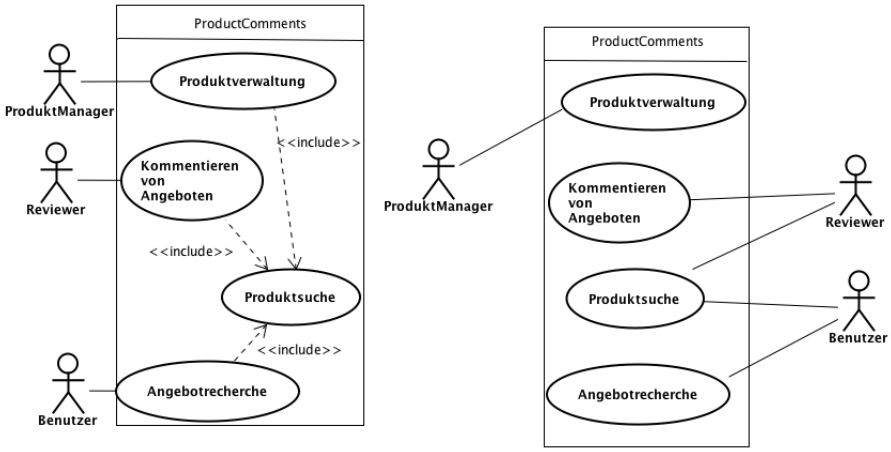
2. Use-Case-Modell

5 Punkte

Vervollständigen Sie das vorgegebene Use-Case-Model:

- Bestimmen Sie die Akteure des Systems und ordnen Sie diese den Funktionsgruppen zu.
- Fügen Sie der Vorgabe keine neuen Funktionsgruppen hinzu.
- Sie dürfen Beziehungen zwischen den Funktionsgruppen hinzufügen.

Beispiellösungen:



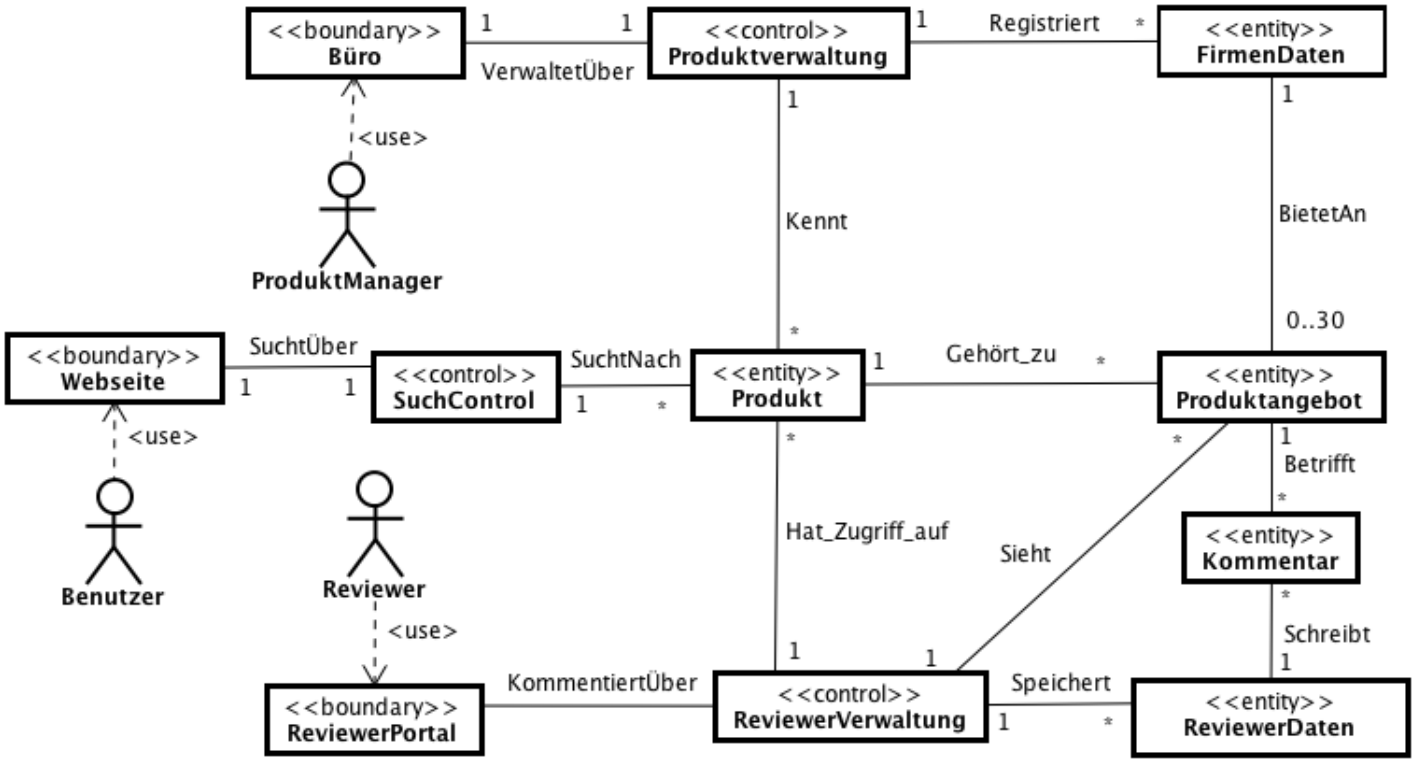
3. Systemklassenmodell

8 Punkte

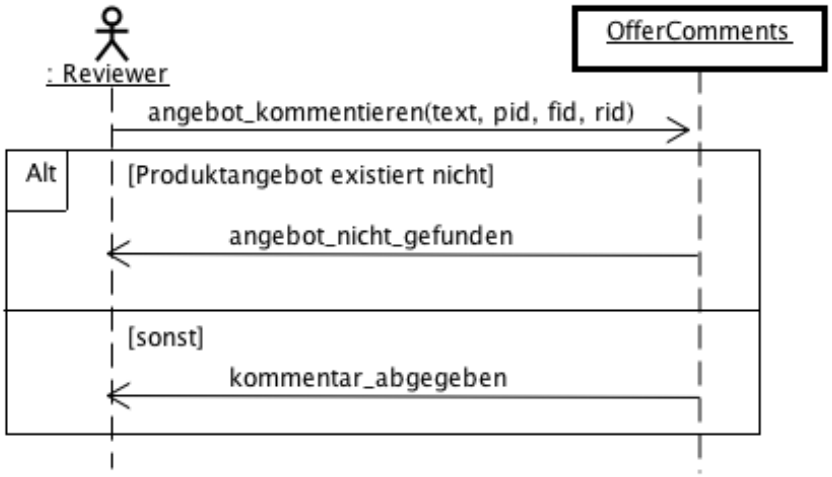
Vervollständigen Sie das vorgegebene Systemklassenmodell, indem Sie Akteure, Klassen und Assoziationen hinzufügen. Kennzeichnen Sie die Klassen als *Boundary*, *Control* oder *Entity*.

Multiplizitäten und Attribute müssen **nicht** eingetragen werden!

Beispiellösung:



Sequenzdiagramm *angebot_kommentieren*



Die Eingabeparameter der Systemoperation *angebot_kommentieren* haben folgende Bedeutungen:

text	der Text des zu erstellenden Kommentars
pid	die eindeutige Identifikationsnummer des Produktes
fid	die eindeutige Identifikationsnummer der Firma
rid	die eindeutige Identifikationsnummer des Reviewers

Die Bedingung *Produktangebot existiert nicht* drückt aus, dass im System kein Angebot gefunden wurde, das zum Produkt mit der Identifikationsnummer *pid* gehört und von einer Firma mit der Identifikationsnummer *fid* angeboten wird.

Es wird in dieser Systemoperation davon ausgegangen, dass ein Reviewer mit der Identifikationsnummer *rid* im System bekannt ist.

4. Operationsschema *angebot_kommentieren* *12 Punkte*

Vervollständigen Sie auf Basis Ihres Systemklassenmodells das vorgegebene Operationsschema für die Systemoperation *angebot_kommentieren* auf der nächsten Seite. Es sollen nur die im Sequenzdiagramm dargestellten Fälle abgedeckt werden.

Nehmen Sie die folgenden Attribut- und Typdeklarationen an:

Produkt
id: int kurzname: String beschreibung: String ...

FirmenDaten
id: int name: String ...

Kommentar
text: String ...

ReviewerDaten
id: int name: String ...

Beispiellösung:

Op.	=	angebot_kommentieren
Desc.	=	Ein Reviewer gibt einen neuen Kommentar im System ab. Der Kommentar betrifft ein bestimmtes Produktangebot, das durch die Eingabeparameter *pid* und *fid* identifiziert wird. Es wird davon ausgegangen, dass ein Reviewer mit der Idenfikationsnummer *rid* bereits im System bekannt ist.
Input	=	$text : String, pid : int, fid : int, rid : int$
Reads	=	$rv : ReviewerVerwaltung, Speichert, Sieht,$ $r : ReviewerDaten$ **with** $r.id = rid \wedge (rv, r) \in Speichert,$ $Produktangebot, BietetAn, FirmenDaten, GehörtZu, Produkt$
Changes	=	$Schreibt, Betrifft, Kommentar$ $k : Kommentar$ **type**
Sends	=	Reviewer {kommentar_abgegeben, angebot_nicht_gefunden}
Pre	=	*implicit*
Post	=	**let** $firmenangebote == \{pa : Produktangebot \mid (rv, pa) \in Sieht$ $\wedge (\exists f : Firmendaten \bullet f.id = fid \wedge (f, pa) \in BietetAn)\}$; $firmenproduktangebote == \{fa : firmenangebote \mid$ $(\exists p : Produkt \bullet p.id = pid \wedge (p, fa) \in GehörtZu)\};$ $\bullet$ $(\#firmenproduktangebote = 0 \Rightarrow$ **is_sent** $\{angebot\_nicht\_gefunden\} \wedge$ **no_effect** $) \wedge$ $(\#firmenproduktangebote \neq 0 \Rightarrow$ **is_sent** $\{kommentar\_abgegeben\} \wedge$ k **new** $\wedge\ k.text' = text \wedge Schreibt' = Schreibt \cup (r, k) \wedge$ $\exists pa : firmenproduktangebote \bullet Betrifft' = Betrifft \cup (k, pa))$

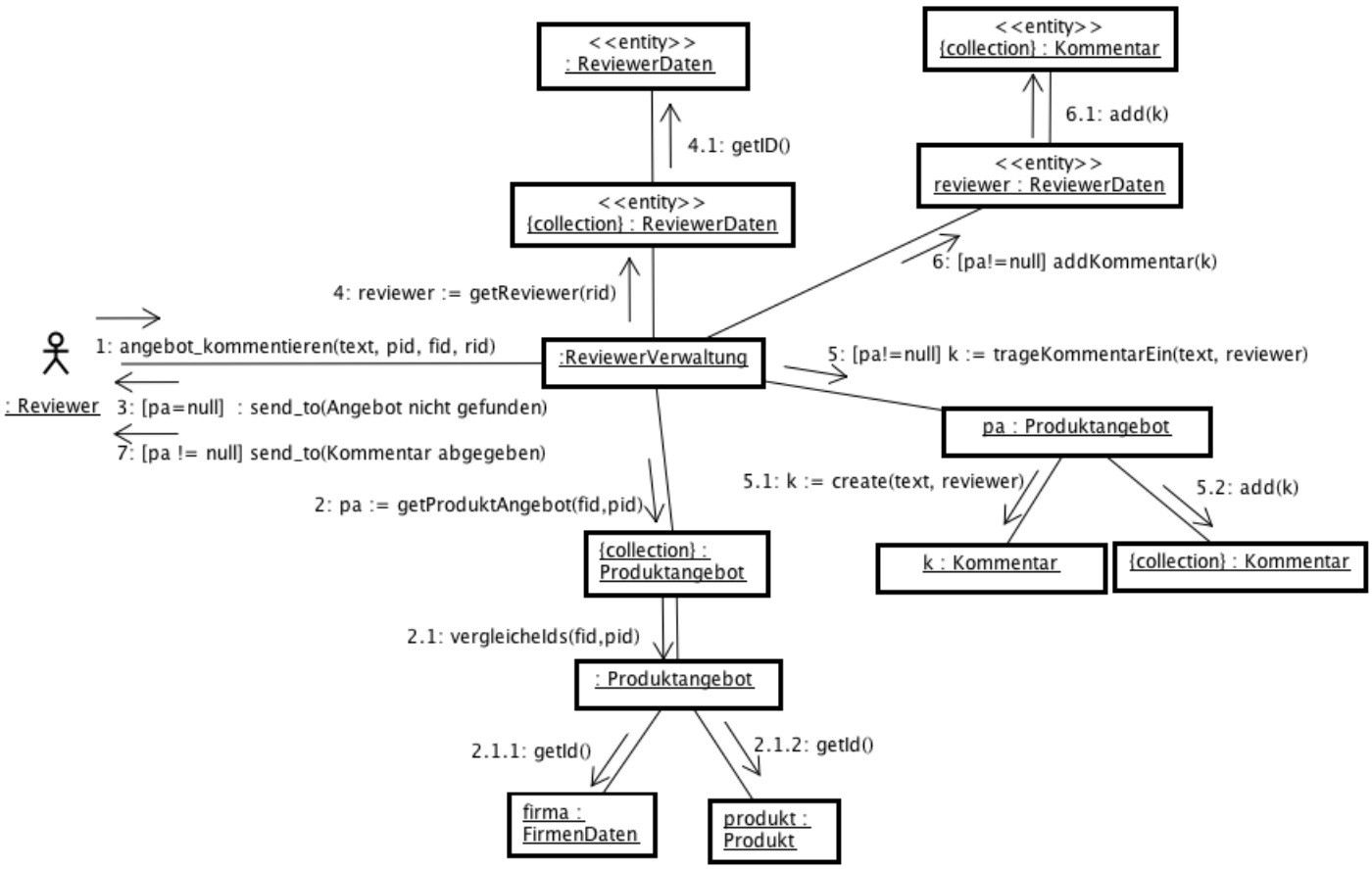
5. Kommunikationsdiagramm *angebot_kommentieren* *12 Punkte*

Vervollständigen Sie auf Basis Ihres Systemklassenmodells und Ihres Operationsschemas das Kommunikationsdiagramm auf der nächsten Seite für die Systemoperation *angebot_kommentieren*.

Beachten Sie dabei folgende Anweisungen:

- Für diese Systemoperation gilt die Vorbedingung, dass ein Reviewer mit der Identifikationsnummer *rid* im System bekannt ist.
- Kommentare zu einem Produktangebot sollen direkt aus dem entsprechenden Objekt der Klasse *Produktangebot* erreichbar sein.
- Kommentare eines Reviewers sollten direkt aus dem entsprechenden Objekt der Klasse *ReviewerDaten* erreichbar sein.

Beispiellösung:



Auszug aus der Java-API

Methoden der Klasse **java.util.Collection<E>**

TYP	NAME UND PARAMETER
boolean	add(E e)
boolean	addAll(Collection <? extends E> c)
void	clear()
boolean	contains(Object o)
boolean	containsAll(Collection<?> c)
boolean	equals(Object o)
int	hashCode()
boolean	isEmpty()
Iterator<E>	iterator()
boolean	remove(Object o)
boolean	removeAll(Collection<?> c)
boolean	retainAll(Collection<?> c)
int	size()
Object[]	toArray()
<T>T[]	toArray(T[] a)

6. Implementierung der Methode *zaehleKommentare*

6 Punkte

Implementieren Sie die Methode *zaehleKommentare* für die Klasse *Produkt*, die angibt, wie viele Kommentare zu Produktangeboten für dieses Produkt im System gespeichert sind. Ergänzen Sie alle für diese Implementierung notwendigen Methoden in den vorgegebenen Ausschnitten der Klassen *Produkt* und *Produktangebot* auf der nächsten Seite.

Gehen Sie davon aus, dass die Objekte korrekt initialisiert werden. Sie müssen **keine** Konstruktoren implementieren. Sie müssen **keine** `import`-Anweisungen angeben.

Beispiellösung:

```
public class Produktangebot {
        private Produkt produkt;
        private java.util.List<Kommentar> kommentare;
        public int anzahlKommentare() {
                return kommentare.size();
        }
}

public class Produkt {
        private int id;
        private java.util.List<Produktangebot> angebote;

        public int zaehleKommentare () {
                int count = 0;
                for (Produktangebot pa: angebote) {
                        count = count + pa.anzahlKommentare();
                }
                return count;
        }
}
```