TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik Institut für Software- und Multimediatechnik, Professur Softwaretechnologie

Technische Universität Dresden, 01062 Dresden

Prof. Dr..
Uwe Aßmann

Klausur
Softwaretechnologie SS 2018

Name:	
Vorname:	
Immatrikulationsnummer:	

Aufgabe	Maximale Punktanzahl	Erreichte Punktanzahl
1	**18**	
2	**14**	
3	**32**	
4	**26**	
Gesamt	**90**	

Hinweise:

- In der Klausur ist als Hilfsmittel lediglich ein **A4-Blatt, beidseitig beschrieben,** zugelassen.
- Die Klammerung der Aufgabenblätter darf **nicht** entfernt werden.
- Tragen Sie bitte die Lösungen auf den Aufgabenblättern ein!
- Verwenden Sie keine roten, grünen Stifte oder Bleistifte!
- Es ist kein eigenes Papier zu verwenden! Bei Bedarf ist zusätzliches Papier bei der Aufsicht erhältlich. Bitte jedes zusätzliche Blatt mit Name, Vorname und Immatrikulationsnummer beschriften.
- Es sind alle Aufgabenblätter abzugeben!
- Ergänzen Sie das Deckblatt mit Name, Vorname und Immatrikulationsnummer!
- Halten Sie Ihren Studentenausweis und einen Lichtbildausweis zur Identitätsprüfung bereit.
- **Achtung!** Das Zeichen ✍ heißt: **Hier ist Java-Text einzufügen**!

Aufgabe 1: Metascience (18 Punkte)

Die wichtigsten Konzepte sind Forscher (*Researcher*), wissenschaftliche Artikel (*Paper*), Konferenzreihen (*Conference*) und die einzelnen Konferenzen, d.h. die Ausgabe einer Konferenz in Form von Proceedings (*ConferenceEdition*).

Für eine Konferenzreihe (*Conference*) gibt es mindestens eine *ConferenceEdition*, für jedes Jahr eine Proceedings (*proceedings*).

Oft schließt sich eine *Conference* als „Satellit“ (*isSatellit*) einer anderen, meist größeren, *Conference* an. Die andere Konferenz stellt sich als Veranstalter (*host*) zur Verfügung.

Researcher können Autor bzw. Mitautor (*autor*) eines *Paper*s sein. Diese Autorenschaft (*Authorship*) ist durch die Position (*position*) in der Autorenliste eines *Paper*s beschrieben. Für jedes *Paper* ist mindestens ein a*utor* gelistet.

Zusätzlich ist im *Authorship* die aktuelle (genau eine) Institution (*Institution*) eines jeden Autors registriert (*myInstitution*). Für die nationale Einordnung einer *Institution* wird das jeweilige Land (*Country*) angegeben. Jede *ConferenceEdition* wird durch genau ein Land (*Country*) organisiert (*organizedBy*).

Researcher können für die Erstellung von *ConferenceEdition*s in verschiedenen Komitees mitarbeiten:

- Ein *Researcher* kann im Programmkomitee an der Erstellung einer *ConferenceEdition* mitarbeiten (*pcMember*). Das Programmkomitee besteht aus mindestens drei Programmkomitee-Mitgliedern (*pc*).
- Ein *Researcher* kann ebenso im Steering Komitee einer *ConferenceEdition* mitarbeiten (*scMember*). Das Steering Komitee besteht aus mindestens drei Steering Komitee-Mitgliedern (*sc*).
- Ein *Researcher* kann auch in der Leitung des Programmkomitees (*pcChairCommittee*) einer *ConferenceEdition* mitarbeiten. Für die Leitung des Programmkomitees sind ein bis drei *Researcher* (*pcChair*) zuständig.

Ein *Researcher* hat häufig mehrere persönliche Profile im Web (*Profil*), wie zum Beispiel Google Scholar oder Research Gate.

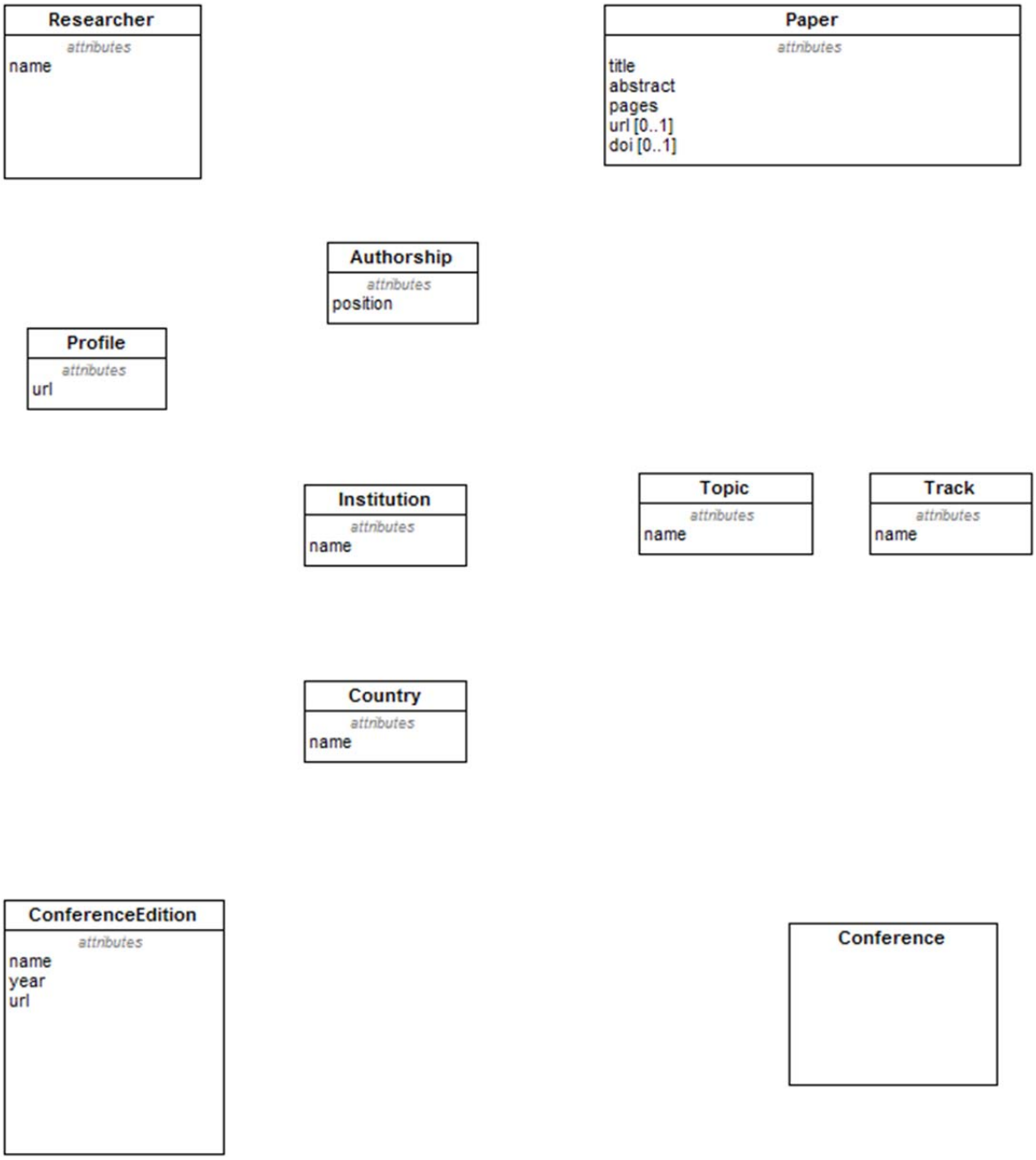
Für jedes *Paper* werden die referenzierten *Paper*s (*cites*) vermerkt. Das *Paper*, welches referenziert worden ist, wird als *isCited* bezeichnet.

*Topic*s sind Themengebiete für *ConferenceEdition*s (*conferences*). Für eine *ConferenceEdition* werden meistens bestimmte *Topic*s als relevant definiert (*topicsOfInterests*). Ein *Paper* kann zu mehreren dieser *Topic*s gehören (*belongsTo*).

Jedes *Paper* wird in genau einer *ConferenceEdition* veröffentlicht (*publishedIn*), und jede *ConferenceEdition* beinhaltet mindestens ein *Paper* (*confPapers*). Ein *Paper* ist genau einem Track (*Track*) einer *ConferenceEdition* zugeordnet (*isAssignedTo*). Ein *Track* fasst thematisch mindestens drei *Paper*s zusammen. Jede *ConferenceEdition* enthält mindestens einen *Track* (*program*).

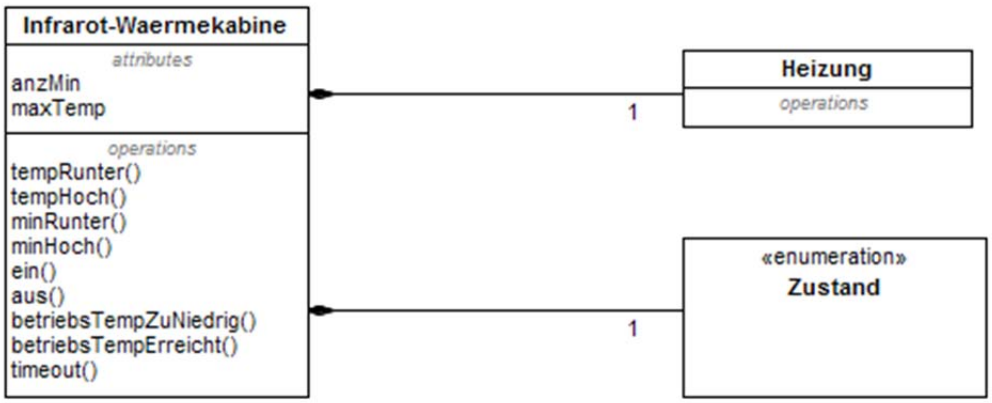
Ergänzen Sie das unten stehende Domänenmodell (UML-Analyse-Klassendiagramm)!
Berücksichtigen Sie dabei folgende Hinweise:

- **Verbinden Sie die gegebenen Klassen mit UML-Klassenbeziehungen, die die Semantik der Beziehungen zwischen den Klassen beschreiben! Berücksichtigen Sie dabei die Modellierungskonventionen, die in den Übungen verwendet wurden!**
- **Beschreiben Sie die Klassenbeziehungen möglichst genau mit Multiplizitäten (beidseitig), Assoziationsnamen mit Leserichtung oder Rollennamen!**
- **Alle domänenspezifischen Begriffe, die im obigen Text *kursiv* geschrieben sind, sollen dabei im Modell wiederzufinden sein! Es sind keine weiteren Namen notwendig!**



Aufgabe 2: Zustandsmodell der Infrarot-Wärmekabine (14 Punkte)

Eine Infrarot-Wärmekabine dient ähnlich einer Sauna zur Erhaltung der Gesundheit. Der Nutzer setzt sich entsprechend seinen Bedürfnissen bei einer voreingestellten Maximaltemperatur (*maxTemp*) eine bestimmte Zeit in die Kabine. Die Kabine wird ausgehend von der aktuellen Temperatur (*aktTemp*) auf die eingestellte Maximaltemperatur aufgeheizt. Das Klassendiagramm zeigt das statische Analysemodell einer Infrarot-Wärmekabine.



Die Wärmekabine wird durch die Betätigung des EIN-Schalters (*ein()*) in Betrieb genommen. Die Maximaltemperatur wird dabei zunächst auf 50°C gesetzt.

In die Wärmekabine ist eine Zeitschaltuhr integriert, die beim Einschalten der Kabine auf Null initialisiert wird (*anzMin=0*). Die Wärmekabine geht damit in den *Grundzustand*.

Die eingestellte Zeit kann zu jedem Zeitpunkt mit Hilfe der Minutentaste der Zeitschaltuhr (repräsentiert durch die Variable *anzMin*) um eine Minute erhöht/erniedrigt werden (Methoden *minHoch()/minRunter()*). Die Zeitschaltuhr läuft dann automatisch bis zu einem Timeout (*timeout()*), d.h. bis *anzMin* wieder Null ist.

Im *Grundzustand* wird nicht geheizt. Die Heizung wird eingeschaltet, sobald vom Sensor der Wärmekabine festgestellt wurde, dass die aktuelle Temperatur niedriger als die maximale Temperatur (*betriebsTempZuNiedrig()*) und die eingestellte Anzahl von Minuten (*anzMin*>0) größer als Null ist.

Falls durch die Heizung der Wärmekabine die Maximaltemperatur erreicht ist (*betriebsTempErreicht()*), wird die Heizung ausgeschaltet. Ebenso wird die Heizung bei *timeout()* ausgeschaltet. Der Nutzer hat zu jedem Zeitpunkt die Möglichkeit, die Maximaltemperatur in einem Bereich von 30°C bis 50°C um jeweils ein Grad Celcius zu erhöhen bzw. zu erniedrigen (*tempHoch()*/*tempRunter()*). Die Wärmekabine kann zu jedem Zeitpunkt durch Betätigen des AUS-Schalters (*aus()*) außer Betrieb genommen werden.

Ergänzen Sie das Analyse-Zustandsmodell (Objektlebenszyklus) für diese Infrarot-Wärmekabine um die fehlenden Transistionen (S. 5)! Modellieren Sie dazu die Infrarot-Wärmekabine als Protokollzustandsmaschine ohne Oberzustände!


Grundzustand
Heizzustand

Aufgabe 3: Konfiguration eines Autos (32 Punkte)

Das Entwurfsmodell für die Konfiguration eines Autos stellt das **Interface `IAuto`** bereit. In einer Anwendung sollen Autos wie Cabrios (**Klasse `Cabrio`**) und Limousinen (**Klasse `Limousine`**) mit zusätzlichen Ausstattungsdetails (**abstrakte Klasse `Ausstattung`**) konfiguriert werden können.

Ausstattungsdetails sind ein Seitenairbag (Klasse `AirBag`), eine Klimaanlage (Klasse `AirCon`) und/oder ein Navigationssystem (Klasse `GPS`).

Das Interface stellt zwei Methoden bereit:

- `berechnePreis()` berechnet den Preis des Autos einschließlich der zusätzlichen Ausstattung. Der Einfachheit halber gelten die folgenden festen Preise (in Euro):

Cabrio:	50000
Limousine:	35000
AirBag:	1000
AirCon:	1500
GPS:	500

- `zeigeDetails()` gibt auf der Konsole die Ausstattung des konfigurierten Autos aus.

Beispiel für die Konfiguration eines voll ausgestatteten Cabrios:

```
public static void main(String[] args) {
    IAuto auto1 = new Cabrio();
    IAuto auto2 = new AirCon(auto1);
    IAuto auto3 = new AirBag(auto2);
    IAuto auto4 = new GPS(auto3);
    auto4.zeigeDetails();
    System.out.println("\n fuer " + auto4.berechnePreis()   + " Euro\n");
 }
```

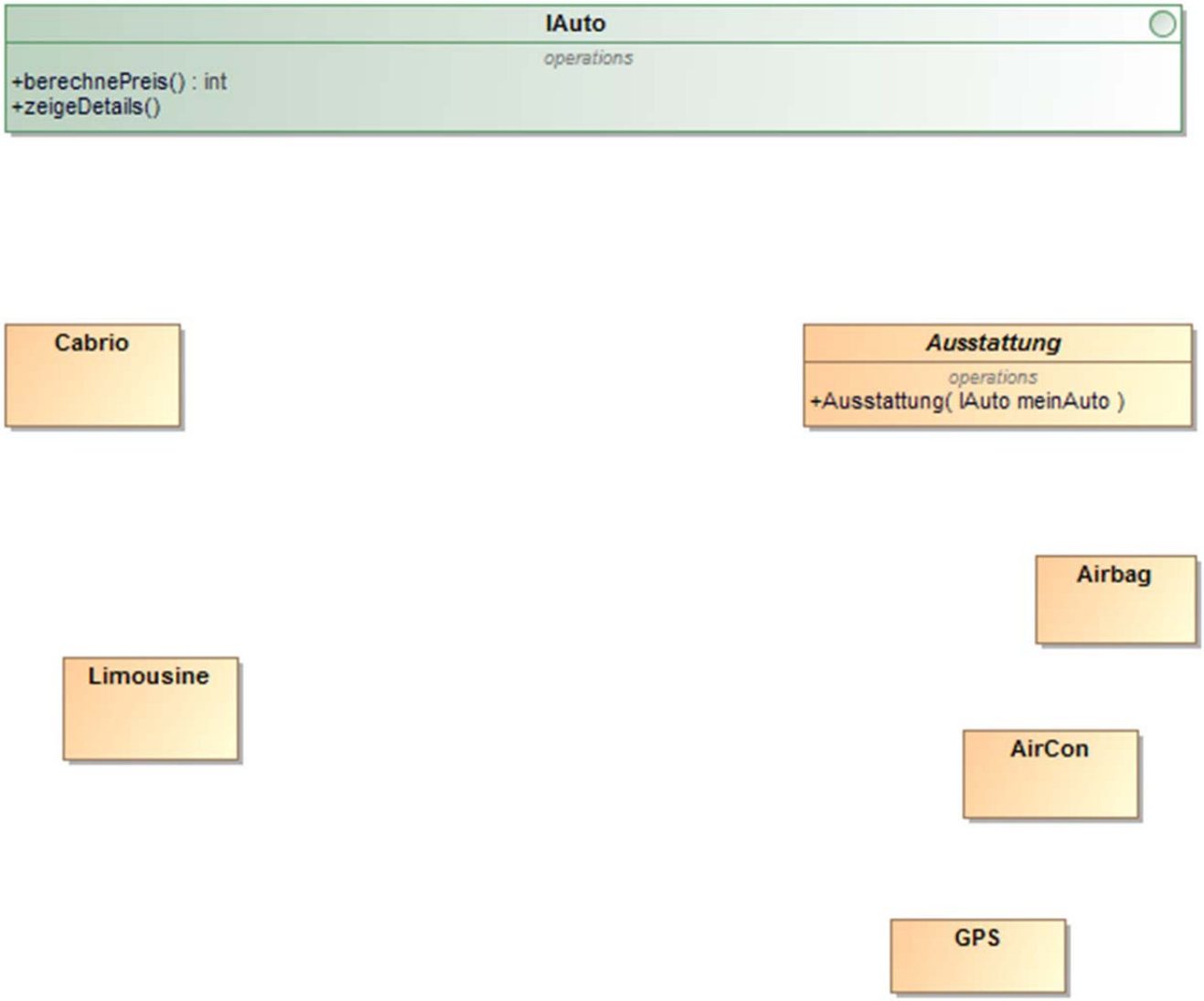
Diese `main()`-Methode gibt auf der Konsole aus:

```
Cabrio, AirCon, AirBag, GPS
 fuer 53000 Euro
```

Lösen Sie folgende Teilaufgaben:

I. Ergänzen Sie entsprechend der `main()`-Methode das folgende Entwurfsklassendiagramm um die notwendigen Beziehungen zwischen dem Interface und den Klassen! Beachten Sie, dass Sie im Fall einer Assoziation bzw. Navigation Richtung, Rollenname und Multiplizität nicht vergessen.

II. Um welches Entwurfsmuster handelt es sich? Tragen Sie dazu das Entwurfsmuster in das Entwurfsklassendiagramm mit den beteiligten Rollen als UML-Kollaboration ein!


IAuto
operations
+berechnePreis() : int
+zeigeDetails()
Cabrio
Ausstattung
operations
+Ausstattung(IAuto meinAuto)
Airbag
Limousine
AirCon
GPS

III. Erstellen Sie für das Endergebnis der `main()`-Methode ein Objektdiagramm! Tragen Sie ausschließlich die Objekte mit Objektnamen und Klassennamen sowie die Links in das Diagramm ein!

IV. Implementieren Sie die abstrakte Klasse `Ausstattung` und die Klasse `AirBag` entsprechend dem UML-Modell! Lassen Sie den Test auf null-Objekte weg!

Zur Hilfestellung ist die Klasse `Cabrio` gegeben:

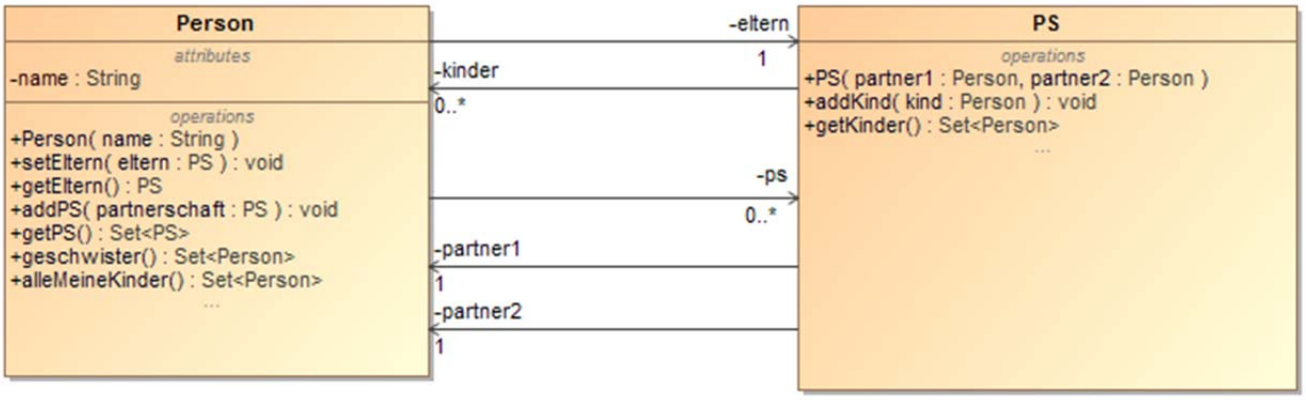
```
public class Cabrio implements IAuto
{
   public void zeigeDetails() {
      System.out.print ("Cabrio");
   }

   public int berechnePreis() {
      return 50000;
   }
}
```

// Klasse **Ausstattung**

// Klasse **AirBag**

Aufgabe 4: Stammbaum (26 Punkte)

Das unten stehende Entwurfsklassendiagramm zeigt den Entwurf für die Implementation der Struktur und des Verhaltens eines Stammbaumes.



Zur Vereinfachung wird angenommen, dass in einer Partnerschaft (Klasse `PS`) immer beide Partner (Klasse `Person`) bekannt sind. Des Weiteren sind die Kinder einer Partnerschaft immer leibliche Kinder beider Partner. Die Methode `geschwister()` soll keine Halbgeschwister zurückliefern.

Implementieren Sie beide Klassen!

- **Nutzen Sie für die Implementierung der Datenstruktur Set die Klasse java.util.HashSet!**
- **Gehen Sie bei der Implementierung der Einfachheit halber davon aus, dass alle Objekte konsistent sind.**
- **Prüfen Sie nicht auf null-Objekte und andere Fehler! Implementieren Sie keine Exceptionen!**
- **Verzichten Sie auf das Überschreiben der Methoden `hashcode()` und `equals()`!**
- **Verwenden Sie für die Attribute und Methoden der Klassen genau die Bezeichnungen wie im Entwurfsklassendiagramm!**

// Klasse `Person`

// Klasse PS