

## Erstklausur

# Softwaretechnik II

Wintersemester 2019/20

Prof. Ralf Reussner

13.03.2020

**Name:** \_\_\_\_\_

**Matrikelnummer:** \_\_\_\_\_

Zur Klausur sind keine Hilfsmittel und kein eigenes Papier zugelassen. Die Bearbeitungszeit beträgt 60 Minuten. Die Klausur ist vollständig und geheftet abzugeben. Verwenden Sie ausschließlich dokumentenechte Stifte. Mit Bleistift oder roter Farbe geschriebene Angaben werden nicht bewertet. Lösen Sie die Aufgaben in leserlicher Schrift. Unlesbare Lösungen können naturgemäß nicht positiv bewertet werden.

<b>Aufgabe</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b><math>\Sigma</math></b>
Maximal	8	16	10	8	18	60
K1						
K2						
K3						

**Aufgabe 1: Wissensfragen (8 Punkte)**

- a) Sollten Sie die Kohäsion (cohesion) einer Klasse erhöhen oder reduzieren? Begründen Sie Ihre Antwort kurz. (1 Punkt)

- b) Benennen Sie die vier Facetten der Klassifikation von Anforderungen nach Glinz? (2 Punkte)

- c) Nennen Sie zwei der drei Architekturprinzipien von Cloud-Architekturen. (1 Punkte)

Name:

Matrikelnummer:

---


d) Erläutern Sie in Stichpunkten die Bedeutung von Conway's Gesetz (Conway's Law). (1 Punkte)

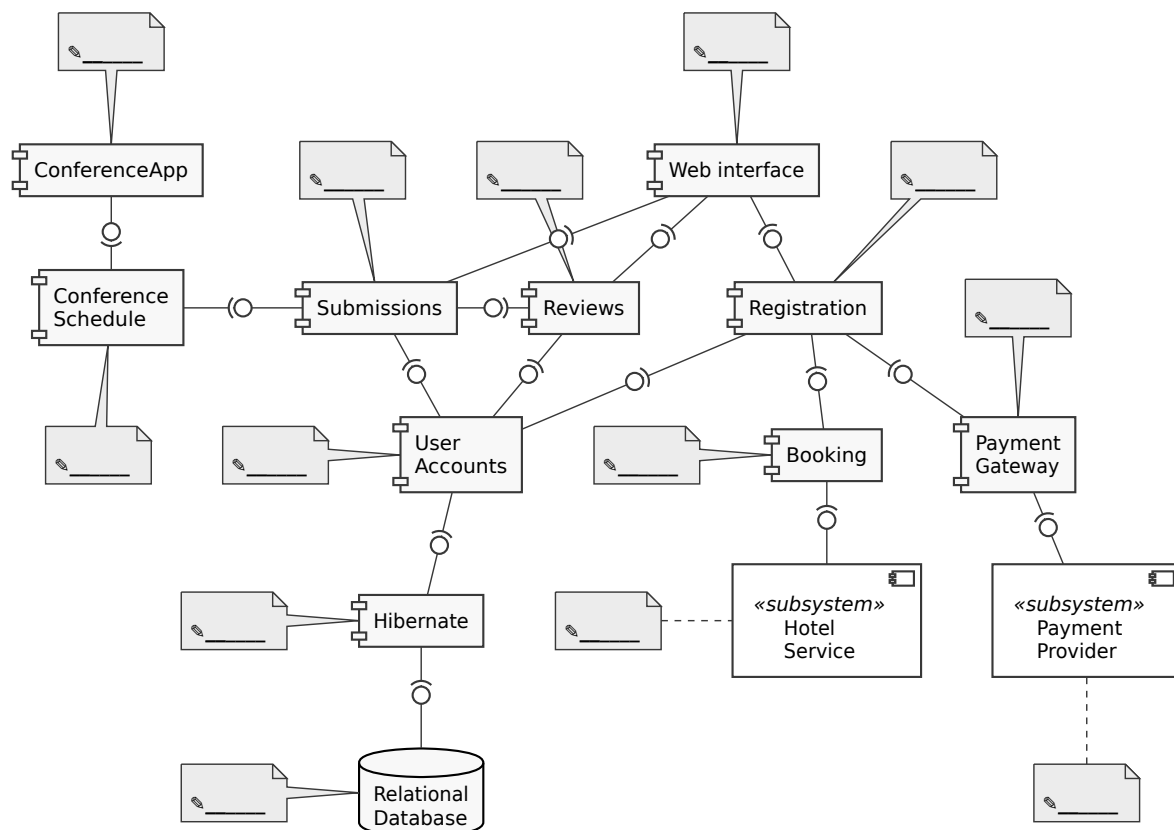
e) Welche Dimensionen der Verlässlichkeit von Softwaresystemen haben Sie kennen gelernt? Benennen und beschreiben Sie drei. (3 Punkte)

## Aufgabe 2: Saubere Unternehmensarchitekturen (16 Punkte)

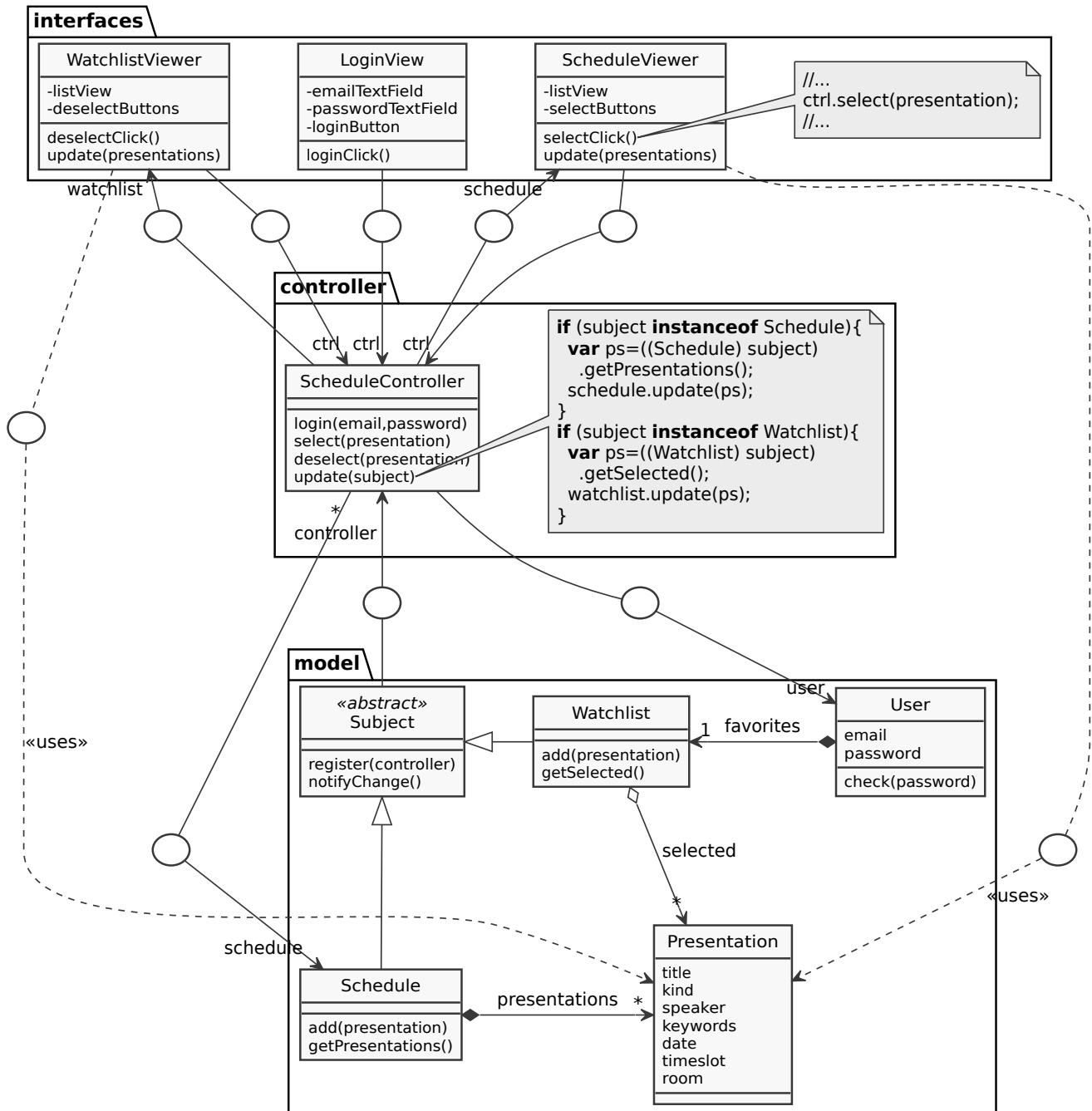
- a) Tragen Sie in die Tabelle die vier Schichten der *sauberen* Architektur (clean architecture) sowie die erlaubte Richtung der Abhängigkeiten (dependency rule) ein. (2,5 Punkte)

Schicht	Schichtenname
1.	
2.	
3.	
4.	

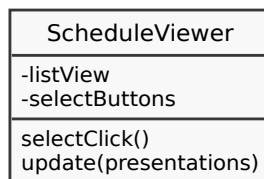
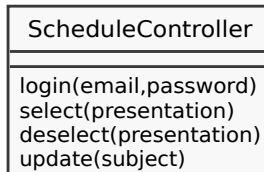
- b) Das folgende Komponentendiagramm stellt die Softwarearchitektur einer Konferenzverwaltung dar. Ordnen Sie nun jede Komponente einer der Schichten *sauberer* Architekturen zu, indem Sie die Nummer der Schicht in die mit „“ markierten Bereiche eintragen. (6,5 Punkte)



- c) Das folgende Klassendiagramm stellt den Entwurf einer Anwendung für das digitale Konferenzprogrammheft dar. Prüfen Sie für jede Assoziation und Nutzen-Beziehung (<<uses>>), ob diese gegen die Abhängigkeitsregel *sauberer* Architekturen verstößt. Markieren Sie ausschließlich die Assoziationen und Nutzen-Beziehungen, die diese Regel *verletzen*, mit einem X im zugehörigen Kreis ○. (3 Punkte)



- d) Der vorhergehende Entwurf (Teilaufgabe c) enthält unzulässige Abhängigkeiten im Kontext der sauberen Architektur (clean architecture) zwischen den Klassen Subject, ScheduleController und ScheduleViewer. Verbessern Sie nun den Entwurf, indem Sie das Prinzip der Abhängigkeitsumkehrung (dependency inversion) anwenden. Zeichnen Sie die hierfür notwendigen Klassen und Schnittstellen (interfaces) mit Ihren Methoden sowie Assoziationen und Vererbungsbeziehungen in das folgende Klassendiagramm ein. (6 Punkte)

**interfaces****controller****model**


Name:

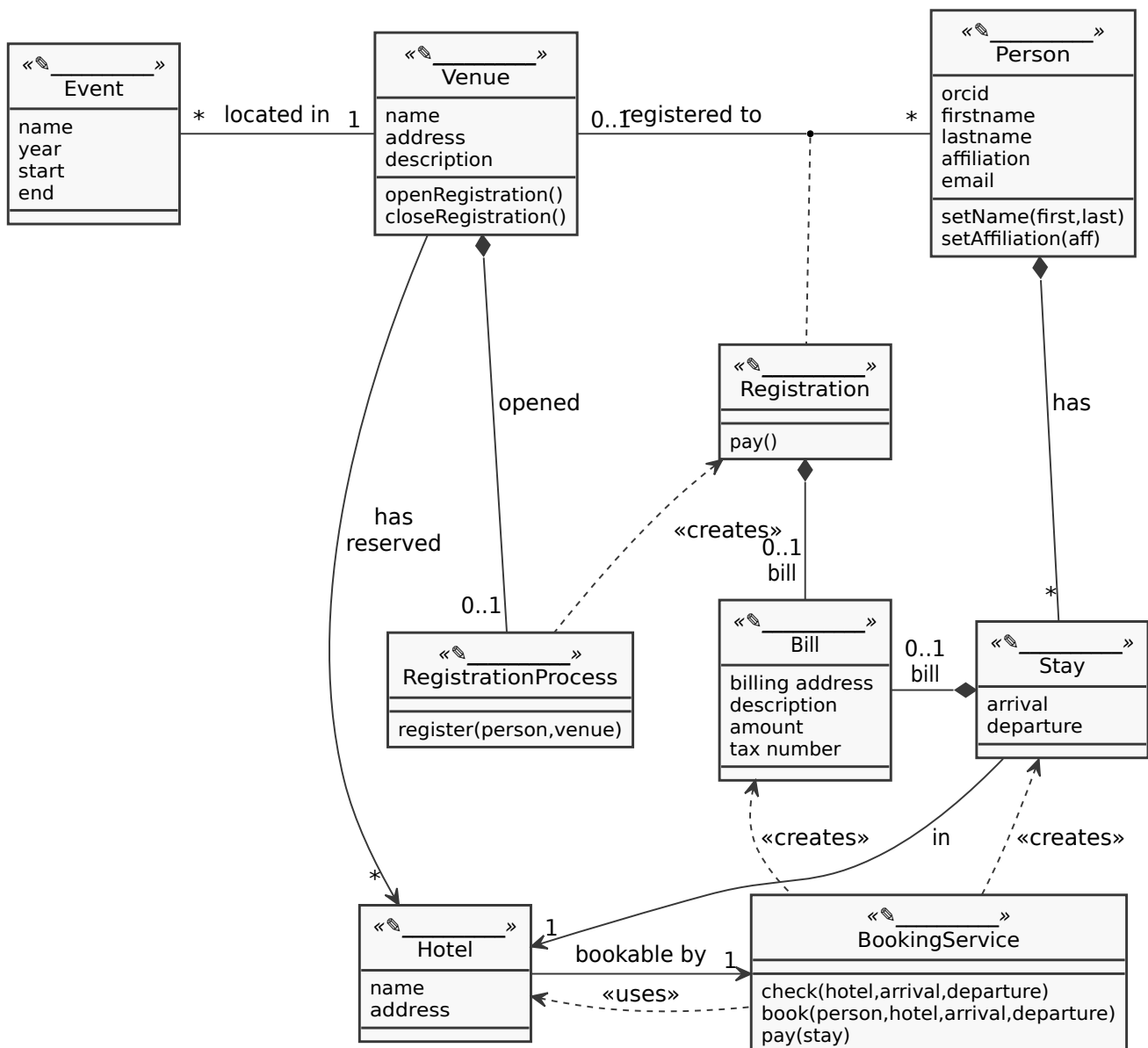
Matrikelnummer:

---

## Aufgabe 3: Domänen-getriebener Entwurf (10 Punkte)

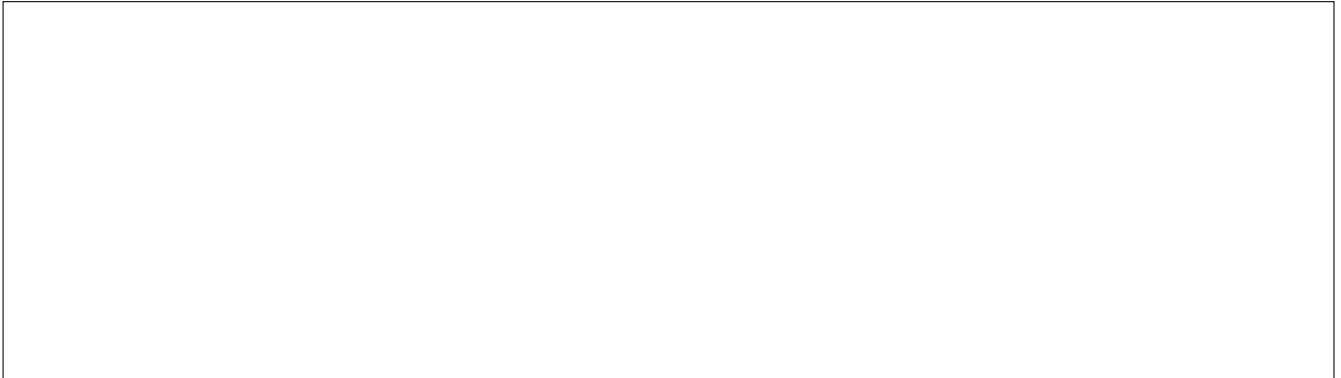
Der folgende Aufgabenkomplex widmet sich der Anwendung des Domänen-getriebenen Entwurfs des Registrierungssystems einer Verwaltungsanwendung für Konferenzen. Dieses erlaubt es Personen, sich für eine Veranstaltungen (Event) an einem Veranstaltungsort (venue) zu registrieren und ein passendes Hotel zu buchen sowie jeweils eine Rechnungen (bill) zu erhalten.

- a) Durch die Domänenanalyse des Registrierungssystems ist das folgende Klassendiagramm entstanden. Klassifizieren Sie jedes Domänenkonzept entsprechend der Bausteine (building blocks) des Domänen-getriebenen Entwurfs entweder als ValueObject, Entity oder Service. Tragen Sie hierzu die Klassifikation in die mit „“ markierten Bereiche der Domänenklassen ein. (4,5 Punkte)



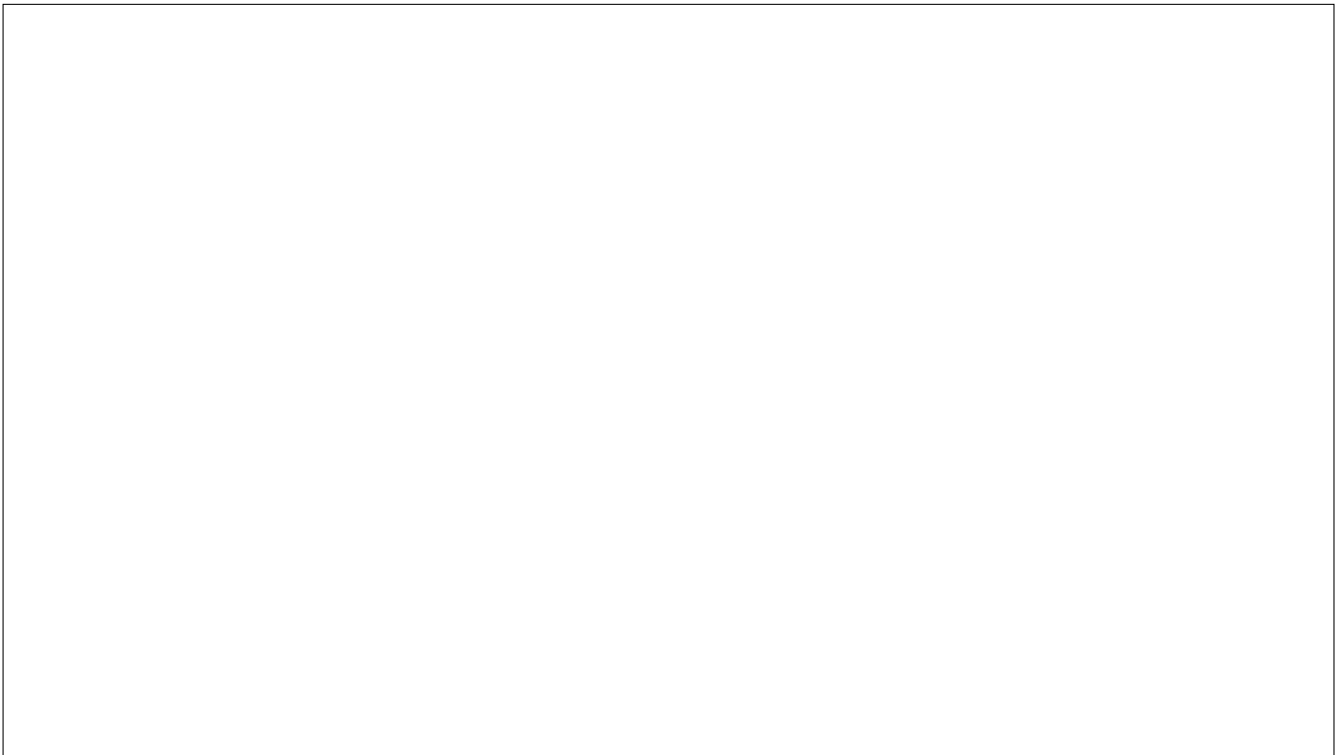


- b) Begründen Sie Ihre Entscheidung für die Klassifikation der Domänenklasse `Hotel` in Stichpunkten. (1 Punkt)



- c) Entwerfen Sie nun die Unterstützungsmuster *Repository* und *Factory* für das Domänenkonzept `Hotel`. Zeichnen Sie hierfür ein Klassendiagramm, welches ein *Repository*, eine *Factory* sowie ihre Methoden mit Parametern beinhaltet. Spezifizieren Sie für die *Factory* mindestens eine Methode und für das *Repository* mindestens drei Methoden. Zeichnen Sie zusätzlich die Domänenklasse `Hotel` sowie ihre Beziehungen zum *Repository* beziehungsweise zur *Factory* ein. (4,5 Punkte)

**Hinweis:** Datentypen (wie zum Beispiel `int`, `long`, `Date`, `String`, ...) müssen nicht angegeben werden. Beschriften Sie jede Assoziation mit mindestens einem Rollenname oder Assoziationsname sowie mindestens einer Kardinalität.



## Aufgabe 4: Quelltextqualität (8 Punkte)

Der Quelltext in Listing 1 stellt den Controller für die Verwaltung von Nutzerkonten (User) der Konferenzverwaltung dar. Die hierfür entwickelte Anwendung enthält jedoch einige Verstöße gegen Praktiken des sauberen Programmierens (Clean-Code-Prinzipien und Coding Conventions aus der Vorlesung).

- a) Identifizieren Sie fünf Zeilen im Quelltextbeispiel (nächste Seite), die unterschiedliche Arten von Verstößen gegen die Praktiken des sauberen Programmierens (Clean-Code-Prinzipien und Coding Conventions aus der Vorlesung) enthalten. Benennen Sie jeweils die Art des Verstoßes mit der zugehörigen Zeilennummer. (5 Punkte)

- b) Die gezeigte Implementierung schützt die Nutzerdaten nur unzureichend. Nennen und erläutern Sie kurz zwei Sicherheitsmechanismen, um die Nutzerdaten zu schützen. (2 Punkt)

```
1 public class User {
2     private final String email;
3     String password;
4     private boolean enabled = true;
5     User(String email, String password) {
6         this.email = email;
7         this.password = password;
8     }
9     public String getEmail() { return email; }
10    public boolean isEnabled() { return enabled; }
11    public void setEnabled(boolean enabled) { this.enabled = enabled; }
12 }
13 public class Admin extends User {
14     Admin(String email, String password) { super(email, password); }
15     public void setEnabled(boolean enabled) { /* Can't disable Admins */ }
16 }
17 public class UserManager {
18     private final List<User> repository = new ArrayList<>();
19     public User create(String email, String password) {
20         if (findByEmail(email) == null)
21             throw new IllegalArgumentException("User already exists!");
22         User account = new User(email, password);
23         save(account);
24         return account;
25     }
26     public void save(User user) {
27         // if (user!=null && !repository.contains(user))
28         repository.add(user);
29     }
30     public boolean authenticate(User user, String password) {
31         return (repository.contains(user) && user.password.equals(password));
32     }
33     public ArrayList<User> findAll() {
34         return repository;
35     }
36     public User findByEmail(String email) {
37         // iterate over each user and return user if emails are equal
38         for (User user : repository) if (user.getEmail().equals(email)) return user;
39         return null;
40     }
41     public List<User> collectDisabled() {
42         List<User> disabled = new ArrayList<>();
43         for (User user : repository) {
44             if (!user.isEnabled())
45                 disabled.add(user);
46         }
47         repository.removeAll(disabled);
48         return disabled;
49     }
50 }
```

Listing 1: Unsicherer Quelltext mit Bad Smells

- c) In einer späteren Version der Anwendung wurden die verschiedenen Methoden zum Finden von Nutzerkonten durch Datenbankabfragen ersetzt. Die `findByEmail` Methode ist im Folgenden angegeben. Für welche typische Sicherheitslücke ist diese Implementierung anfällig? Benennen Sie die Sicherheitslücke und erläutern Sie einen Lösungsansatz. (1 Punkte)

```
1 public class UserManager {
2     @PersistenceContext
3     private EntityManager em;
4     //...
5     public User findByEmail(String email) {
6         Query query = em.createQuery("select u from User u where u.email = '"+email+"'");
7         List result = query.getResultList();
8         if (! result.isEmpty())
9             return (User) result.get(0);
10        return null;
11    }
12    //...
13 }
```

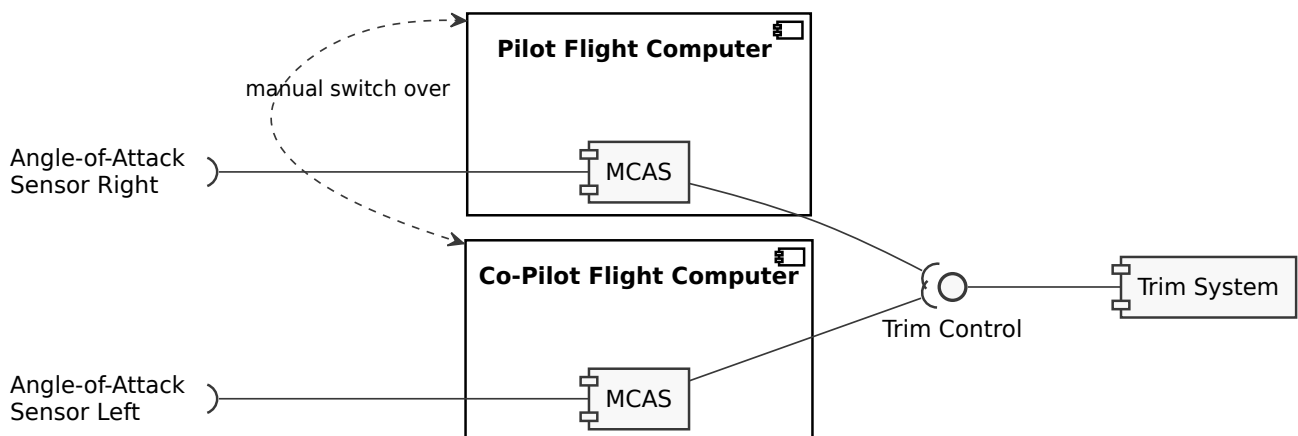
Antwort:

## Aufgabe 5: Echtzeitsysteme (18 Punkte)

Ihre Aufgabe ist es, die Zuverlässigkeit der automatische Fluglagesteuerung (MCAS) des Flugzeugs Rowing MAT 9 zu bewerten und zu verbessern.

Das Maneuvering Characteristics Augmentation System (MCAS) dient dazu, die Steigung des Flugzeugs zu reduzieren, sollte der Anströmwinkel (angle of attack) einen Grenzwinkel überschreiten, bei dem ein Strömungsabriss und damit der Absturz des Flugzeugs droht. Dazu wertet es Daten eines Anströmwinkel-Sensors aus und muss im Fall einer Grenzwertüberschreitung innerhalb von 100 Millisekunden ein Steuersignal an die Trimungssteuerung (trim system) senden, um einen Strömungsabriss zu verhindern. Im aktuellen Entwurf läuft jeweils ein MCAS auf dem Bordcomputer des Piloten und des Co-Piloten und verwendet einen eigenen Anströmwinkel-Sensor (angle of attack sensor). Im Betrieb ist nur einer der beiden Systeme aktiv, welches manuell umgeschaltet werden kann.

Die Architektur des Systems und Ihre Verteilung auf Computern ist in dem Folgendem Komponentendiagramm dargestellt.



- a) Handelt es sich bei diesem System um ein hartes oder weiches Echtzeitsystem? Begründen Sie Ihre Antwort kurz. (1 Punkte)

Name:

Matrikelnummer:

---

- b) Handelt es sich bei dem beschriebenen System um ein Monitoring- oder Kontrollsystem? Begründen Sie Ihre Antwort kurz. (1 Punkte)

- c) Dieser Entwurf ist offensichtlich unzuverlässig. Verwenden Sie ein geeignetes Architekturmuster für Echtzeitsysteme, um die Zuverlässigkeit des Systems zu garantieren. Zeichnen Sie hierfür ein Komponentendiagramm, welches die MCAS Komponenten enthält sowie die dafür notwendigen (eventuell neuen) Sensoren, Komponenten und Computer. (7 Punkte)

## Scheduling in Echtzeitsystemen

Gegeben sind die folgenden vier Prozesse mit Angabe ihrer jeweiligen Startzeit (*arrival time*), Dauer (*duration*) und Frist (*deadline*):

Process	Arrival Time	Duration	Deadline
P <sub>0</sub>	2	2	6
P <sub>1</sub>	1	6	12
P <sub>2</sub>	6	3	15
P <sub>3</sub>	8	1	10

**Hinweis:** Die Prozesse kommen zu Beginn des in *Arrival Time* angegebenen Zeitslots an und sollen bis zum Ende des in *Deadline* angegebenen Zeitslots beendet sein. Führen Sie das Scheduling auch dann fort, wenn die *Deadline* nicht eingehalten werden kann.

d) Verteilen Sie die Prozesse auf die Prozessorzeit nach dem First-In-First-Out-Verfahren. (2 Punkte)

P <sub>n</sub>	t <sub>1</sub>	t <sub>2</sub>	t <sub>3</sub>	t <sub>4</sub>	t <sub>5</sub>	t <sub>6</sub>	t <sub>7</sub>	t <sub>8</sub>	t <sub>9</sub>	t <sub>10</sub>	t <sub>11</sub>	t <sub>12</sub>	t <sub>13</sub>	t <sub>14</sub>	t <sub>15</sub>
P <sub>0</sub>															
P <sub>1</sub>															
P <sub>2</sub>															
P <sub>3</sub>															

e) Teilen Sie die Prozessorzeit nach dem präemptiven Least-Laxity-Verfahren ein, um die Prozesse zu verteilen. (7 Punkte)

P <sub>n</sub>	t <sub>1</sub>	t <sub>2</sub>	t <sub>3</sub>	t <sub>4</sub>	t <sub>5</sub>	t <sub>6</sub>	t <sub>7</sub>	t <sub>8</sub>	t <sub>9</sub>	t <sub>10</sub>	t <sub>11</sub>	t <sub>12</sub>	t <sub>13</sub>	t <sub>14</sub>
P <sub>0</sub>														
lax <sub>0</sub>														
P <sub>1</sub>														
lax <sub>1</sub>														
P <sub>2</sub>														
lax <sub>2</sub>														
P <sub>3</sub>														
lax <sub>3</sub>														

Name:

Matrikelnummer:

---