Karlsruher Institut für Technologie

Software Design and Quality

Zweitklausur

Softwaretechnik II

Wintersemester 2020/21

Prof. Dr. Ralf H. Reussner

07.05.2021

Musterlösung

Zur Klausur sind keine Hilfsmittel und kein eigenes Papier zugelassen. Die Bearbeitungszeit beträgt 90 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.

Aufgabe	**1**	**2**	**3**	**4**	**5**	**6**	Σ
Maximal	10	16	21	13	16	14	90
K1							
K2							
K3							

Aufgabe 1: Wissensfragen *(10 Punkte)*

a) Nennen Sie die fünf Architekturmuster für Datenquellen *(data source architectural patterns)* aus der Vorlesung Patterns of Enterprise Application Architectures. (2,5 Punkte)

Musterlösung:

- Record set (ResultSet)
- Table Data Gateway
- Active Record
- Row Data Gateway
- Data Mapper

b) Nennen Sie vier der sieben internen Bestandteile eines Microservices nach Toby Clemson aus der Vorlesung Microservices. (2 Punkte)

Musterlösung:

- Resources
- Service Layer
- Domain
- Repositories
- Gateways
- HTTP Client
- Data Mappers / ORM

c) Nennen Sie vier der sechs *Interaktionen zwischen Domänen* des strategischen Entwurfs aus der Vorlesung Domain-driven Design. (2 Punkte)

Musterlösung:

- Shared Kernel
- Consumer/Supplier Development Team
- Conformist
- Anticorruption Layer
- Open Host Service
- Published Language

d) Nennen Sie die vier *Service-Deployment*-Modelle des Cloud Computings. (2 Punkte)

Musterlösung: [redacted]

- Private cloud
- Public cloud
- Hybrid cloud
- Community cloud

e) Nennen Sie die drei Hauptrollen im Scrum-Prozess. (1,5 Punkte)

Musterlösung [redacted]

- Product owner
- Team (member)
- Scrum master

Aufgabe 2: Anforderungserhebung *(16 Punkte)*

Im Folgenden betrachten wir die Entwicklung eines Praxisverwaltungssystems *(medical practice management software)*, abgekürzt **PVS**:

> *Ein Praxisverwaltungssystem ist ein komplexes Softwaresystem, welches alle Vorgänge in einer Arztpraxis – von der Termin- und Wartezimmerverwaltung über die Führung einer elektronischen Patientenakte bis zur Abrechnung am Quartalsende – unterstützt. Das System erlaubt weiterhin das Anbinden mehrerer Laborgeräte, wie zum Beispiel Ultraschallgeräte, Lungenfunktionsmessgeräte, oder Elektrokardiogramm-Messgeräte. Zusätzlich muss das Praxisverwaltungssystem Abrechnungsinformationen direkt an die Kassenärztliche Vereinigung* (KV) *übermitteln können, die diese wiederum an die Krankenkassen* (health insurances) *weiterleitet.*

Während der Anforderungserhebung wurden für das Praxisverwaltungssystem unter anderem die folgenden Anforderungen identifiziert:

Nr.	Anforderung
/R1/	*Das PVS legt für jeden Patienten eine elektronische Patientenakte an, die die Allergien, Diagnosen, Behandlungen und Medikation enthält.*
/R2/	*Das PVS benötigt für die Ausführung nicht mehr als 2 GB RAM.*
/R3/	*Das PVS verschlüsselt alle Patienteninformationen mit einem sicheren Verfahren.*
/R4/	*Das PVS erlaubt es der Sprechstundenhilfe, Patienten in die Warteliste aufzunehmen und Patienten einen Termin zu einem späteren Zeitpunkt zu geben.*
/R5/	*Beim Durchführen der Abrechnung sammelt das PVS alle relevanten Informationen des vergangenen Quartals, prüft diese und sendet sie an die zuständige KV.*
/R6/	*Beim Einlesen einer Krankenkassenkarte wird automatisch der zugehörige Patient ausgewählt.*
/R7/	*Das PVS implementiert den Gerätedaten-Transfer-Standard* (GDT)*, um Laborgeräte anzubinden.*
/R8/	*Das PVS wurde von der zuständigen KV zertifiziert.*

a) Nennen Sie vier weitere Stakeholder, die neben *Patienten* und der *Ärztin* für die Entwicklung eines Praxisverwaltungssystems relevant sind. (2 Punkte)

Musterlösung: [redacted]

- Arzthelferin, Sprechstundenhilfe, Verwaltungspersonal
- Krankenkassen
- Kassenärztliche Vereinigung
- Gesetzgeber (Bundes-/Landesregierung, Bundesbeauftragter für Datenschutz)
- Laborgerätehersteller
- Techniker (z.B.: Wartungstechniker, Admin, Software-Entwickler, Software-Tester)
- Finanzabteilung, Steuerberater, Wirtschaftsprüfer

Die folgenden **keine** Stakeholder: *Arztpraxis, Gesundheitsamt, Angehörige von Patienten, Apotheker, Putzkräfte, Pharmaunternehmen!*

b) Klassifizieren Sie die oben stehenden Anforderungen nach der in der Vorlesung behandelten Klassifikation nach Glinz. Ordnen Sie hierzu die acht Anforderungen des Praxisverwaltungssystems den in der Vorlesung genannten Anforderungsarten *(kind: functional, quality, constraint)*, deren Unterkategorie *(subkind: functions, data, performance, security, availability, legal, ...)* sowie deren Repräsentation *(representation: operational, quantitative, qualitative, declarative)* zu. (12 Punkte)

Musterlösung: [redacted]

Nr.	Anforderung	Art	Unterkategorie	Repräsentation
/R1/	*elektronische Patientenakte*	functional	data/functions	operational
/R2/	*2 GB RAM*	constraint	environment/physical design&impl.	quantitative
		(quality)	(performance)	
/R3/	*verschlüsselt Patientenakte*	quality	security	operational/qualitative
/R4/	*Warteliste und Termine*	functional	functions	operational
/R5/	*Durchführen der Abrechnung*	functional	functions	operational
/R6/	*Krankenkassenkarte einlesen*	functional	functions	operational
		(quality)	(usability)	operational
/R7/	*GDT-Standard*	constraint	environment/interface/ design&impl.	declarative
/R8/	*Zertifizierung durch KV*	constraint	legal	declarative

c) Welche der oben stehenden Anforderungen verstoßen gegen die Empfehlungen für gut geschriebene Anforderungen *(basic writing recommendations)* aus der Vorlesung? Finden Sie **zwei** Anforderungen die gegen **unterschiedliche** Richtlinien verstoßen. Nennen Sie jeweils die Nummer und Richtlinien, gegen die verstoßen wurden. (2 Punkte)

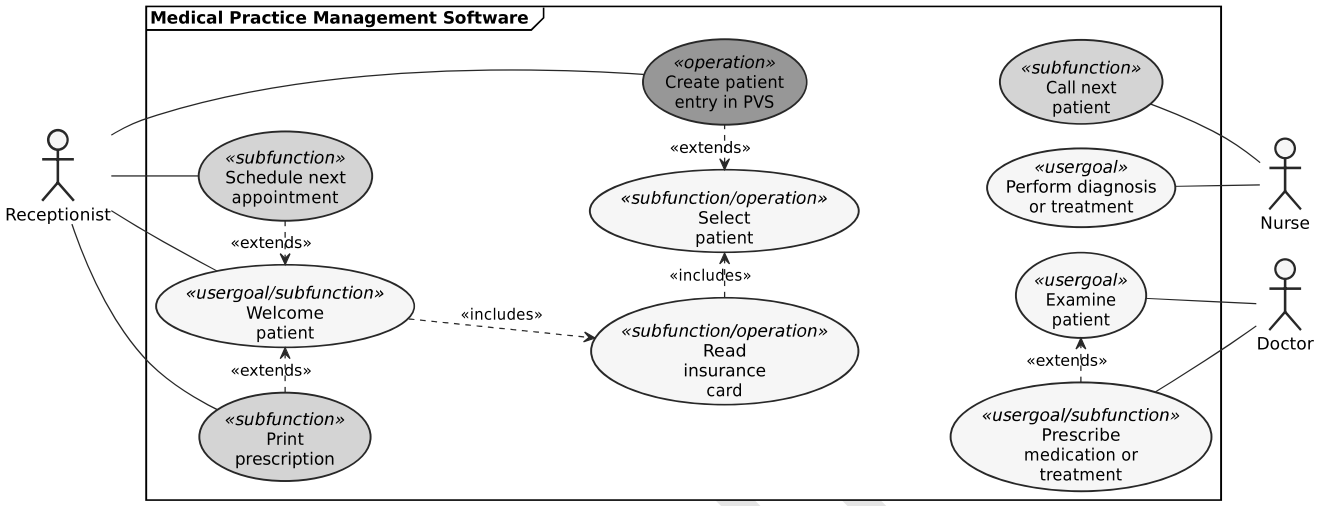
Musterlösung: [redacted]

- /R3/: Schwache Sprache, *sicher* und *alle Patienteninformationen* nicht eindeutig definiert
- /R4/: Mehrere Belange in einer Anforderung, *Warteliste verwalten* und *Termine anlegen*
- /R5/: Mehrere Belange in einer Anforderung oder schwache Sprache
- /R6/: Unklarer Akteur
- /R6,R8/: Passive Sprache

[redacted]

Aufgabe 3: Anwendungsfallbeschreibung *(21 Punkte)*

Diese Aufgabe betrachtet wesentliche Anwendungsfälle der Arztin *(doctor)*, der Arzthelferin *(nurse)* und der Sprechstundenhilfe *(receptionist)* im Praxisverwaltungssystem (PVS). Diese sind im folgenden Anwendungsfalldiagramm zusammengefasst und in der nachfolgenden Tabelle kurz erläutert.



Anwendungsfall	Beschreibung
Examine patient	Untersuchung eines Patienten.
Prescribe medication or treatment	Verschreiben von Medikamenten oder einer Behandlungen.
Schedule next appointment	Festlegen des nächsten Termins für einen Patienten.
Welcome patient	Empfang eines Patienten an der Anmeldung.
Read insurance card	Einlesen der Krankenkassenkarte.
Print prescription	Ausdrucken eines Rezeptes für einen Patienten.
Select patient	Auswählen eines Patienten aus Datenbestand des PVS.
Create patient entry in PVS	Erstellen eines neuen Patienteneintrags im PVS.
Add patient to waiting list	Hinzufügen eines Patienten zur Warteliste.
Perform diagnosis or treatment	Durchführen einer Laboruntersuchung (z.B.: Blutabnahme) oder Behandlung (z.B.: Verband anlegen).
Call next patient	Aufrufen des nächsten Patienten in das Behandlungszimmer.

a) Klassifizieren Sie jeden Anwendungsfall im obigen Anwendungsfalldiagram (use case diagram) entweder als Nutzerziel *(user goal)* mit U, als Teilfunktion *(subfunction)* mit S oder als Operation *(operation)* mit O. Tragen Sie hierzu die entsprechende Klassifikation in die mit „✎“ markierten Bereiche der Anwendungsfälle ein. (5 Punkte)

Musterlösung: [redacted]

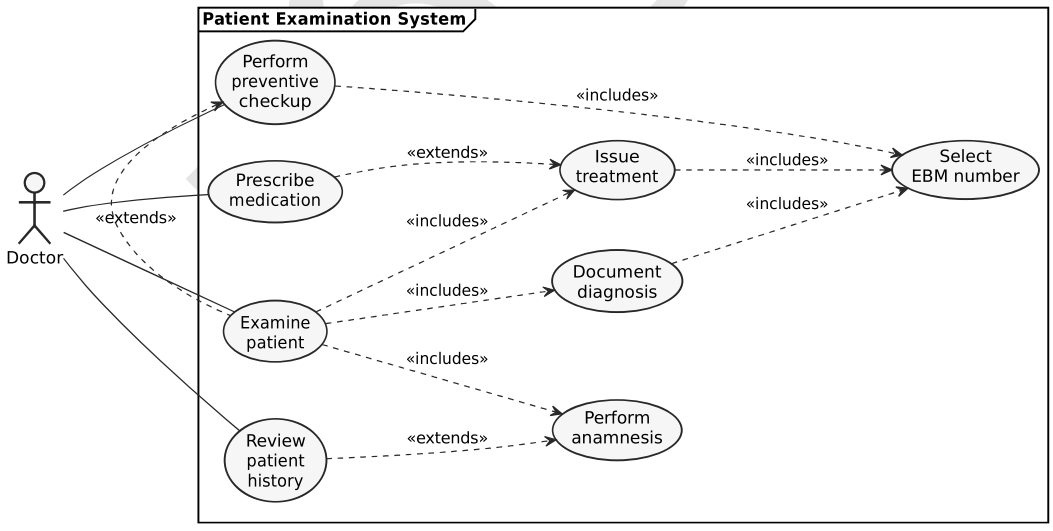
Die Musterlösung ist im obigen Anwendungsfalldiagramm dargestellt.

Da diese Anwendungsfälle nicht detailliert genug sind, ist es Ihre Aufgabe, weitere Anwendungsfalldiagramme zu erstellen. Hierzu charakterisiert die folgende Beschreibung die zugrundeliegenden Aufgaben der Ärztin im Praxisverwaltungssystem *(PVS)*:

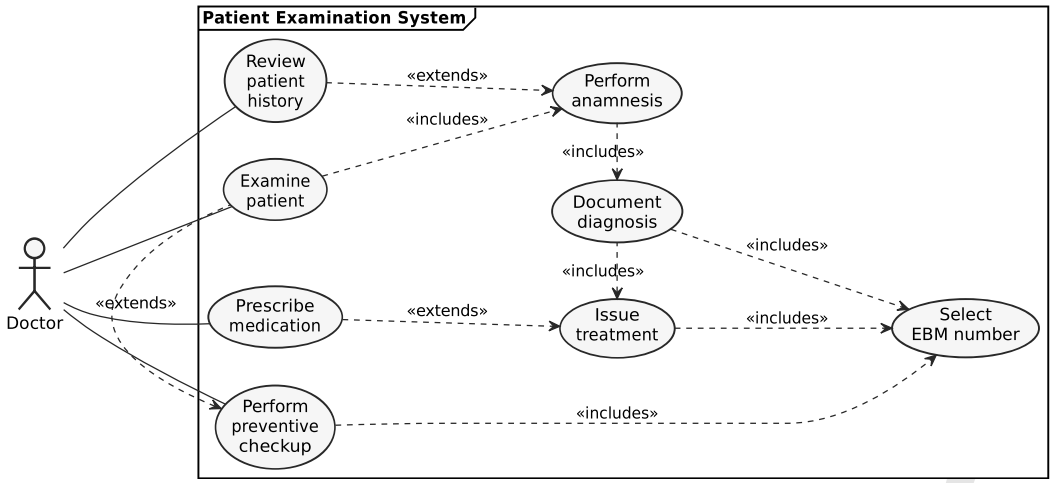
> *Das PVS erlaubt der Ärztin* (doctor) *einen vorab ausgewählten Patienten zu untersuchen* (examine patient) *oder bei ihm eine Vorsorgeuntersuchung durchzuführen* (preventive medical checkup). *Bei der Untersuchung eines Patienten* (examine patient) *führt die Ärztin zuerst eine Anamnese durch* (perform anamnesis), *danach dokumentiert sie ihre Diagnose* (document diagnosis) *und verschreibt anschließend eine Behandlung* (issue treatment). *Wenn bei der Anamnese die Befragung des Patienten nicht ausreicht, kann die Ärztin die zugehörige Patientenakte einsehen* (review patient history). *Falls die Behandlung des Patienten es erfordert, kann die Ärztin die nötigen Medikamente verschreiben* (prescribe medication). *Bei einer Vorsorgeuntersuchung kann die Ärztin bei Verdacht sofort eine ausführliche Untersuchung des Patienten veranlassen* (examine patient). *Für die leichtere Dokumentation und Abrechnung muss die Ärztin bei Vorsorgeuntersuchungen, bei Diagnosen sowie beim Verschreiben von Behandlungen jeweils die zugehörige Nummer des einheitlichen Bewertungsmaßstabs (EBM) auswählen* (select EBM number), *die daraufhin der Patientenakte hinzugefügt wird.*

b) Erstellen Sie nun ein Anwendungsfalldiagramm (use case diagram) genau so, dass es die Anforderung im obigen Text beschreibt. Zeichnen Sie nur Anwendungsfälle und Aktoren ein, die im Text referenziert werden. Achten Sie darauf, an den geeigneten Stellen Beziehungen zwischen den Anwendungsfällen (use cases) zu modellieren. (9 Punkte)

Musterlösung:



Eine weitere Lösung ist im Folgenden Anwendungsfalldiagramm dargestellt:


Patient Examination System
Review patient history
Perform anamnesis
Examine patient
Document diagnosis
Prescribe medication
Issue treatment
Select EBM number
Perform preventive checkup
Doctor
«extends»
«includes»
«includes»
«includes»
«includes»
«extends»
«extends»
«includes»
«includes»

c) Verwenden Sie das textuelle Anwendungsfallbeschreibungsschema *Fully-Dressed* zur vollständigen Spezifikation des Erfolgsfalls der Untersuchung eines Patienten *(examine patient)*. Nutzen Sie hierfür die unten gegebene Schablone. Nennen Sie drei Stakeholder und geben Sie mindestens eine Erweiterung an. (7 Punkte)

Musterlösung [redacted]

Name d. Anwendungsfalls *(Use Case Name)*	**Untersuchung eines Patienten** *(examine patient)*
Zielebene *(Goal Level)*	Nutzerziel (User Goal)
Primäraktor(en) *(Primary Actor(s))*	Arztin, (Patient)
Stakeholder(s)	Arzthelferin, Patient, Kassenärztliche Vereinigung, Krankenkasse
Vorbedingungen *(Preconditions)*	• Der Patient ist im Patientenuntersuchungssystem ausgewählt. • (Die Ärztin ist im PVS angemeldet.) • (Der Patient wurde im PVS angelegt.)
Nachbedingungen *(Postconditions)*	• Zur Patientenakte wurde die EBM-Nummer der Behandlung und Diagnose hinzugefügt. • (Der Patient wurde untersucht.)
Primäres Erfolgsszenario *(Main Success Scenario)*	1) Anamnese durchführen 2) Diagnose dokumentieren 3) Auswahl der EBM-Nummer für die Diagnose 4) Behandlung verschreiben 5) Auswahl der EBM-Nummer für die Behandlung

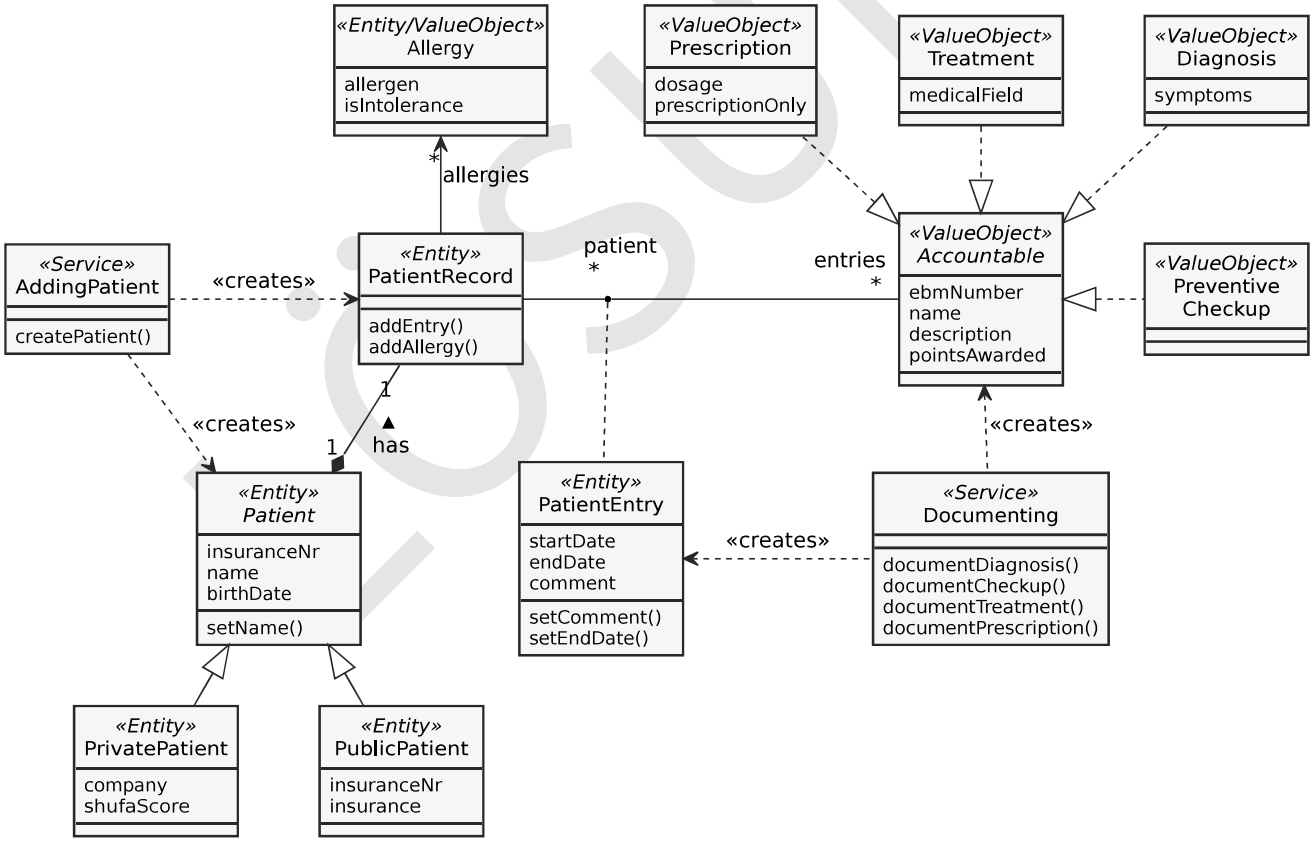
Name d. Anwendungsfalls *(Use Case Name)*	**Untersuchung eines Patienten** *(examine patient)*
Erweiterung (Weiterer Ablauf) *(Extension; Additional Flow)*	• 1a) Die Anamnese reicht nicht aus, sieht die Ärztin die Patientenakte ein. • 4a) Die Behandlung erfordert Medikamenteneinnahme, die Ärztin verschreibt die nötigen Medikamente.

Aufgabe 4: Domänen-getriebener Entwurf *(13 Punkte)*

Der folgende Aufgabenkomplex widmet sich der Anwendung des Domänen-getriebenen Entwurfs für die Verwaltung der Patienten *(patient)* und Patientenakten *(patient record)* des Praxisverwaltungssystems. Dieses erlaubt es, einen neuen Patienten und seine zugehörige Patientenakte anzulegen *(adding patients)* sowie Diagnosen *(diagnosis)*, Vorsorgeuntersuchungen *(preventive checkups)*, verschriebene Behandlungen *(treatments)* und Medikamente *(prescriptions)* zu dokumentieren *(documenting)*. Hierbei muss die Menge der abrechenbaren Leistungen *(accountables)* nach dem einheitlichen Bewertungsmaßstabs (EBM) von der zuständigen Kassenärztlichen Vereinigung *(KV)* übernommen werden.

a) Durch die Domänenanalyse der Stationsverwaltung 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. (6,5 Punkte)

Musterlösung: [redacted]



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

Musterlösung: [redacted]

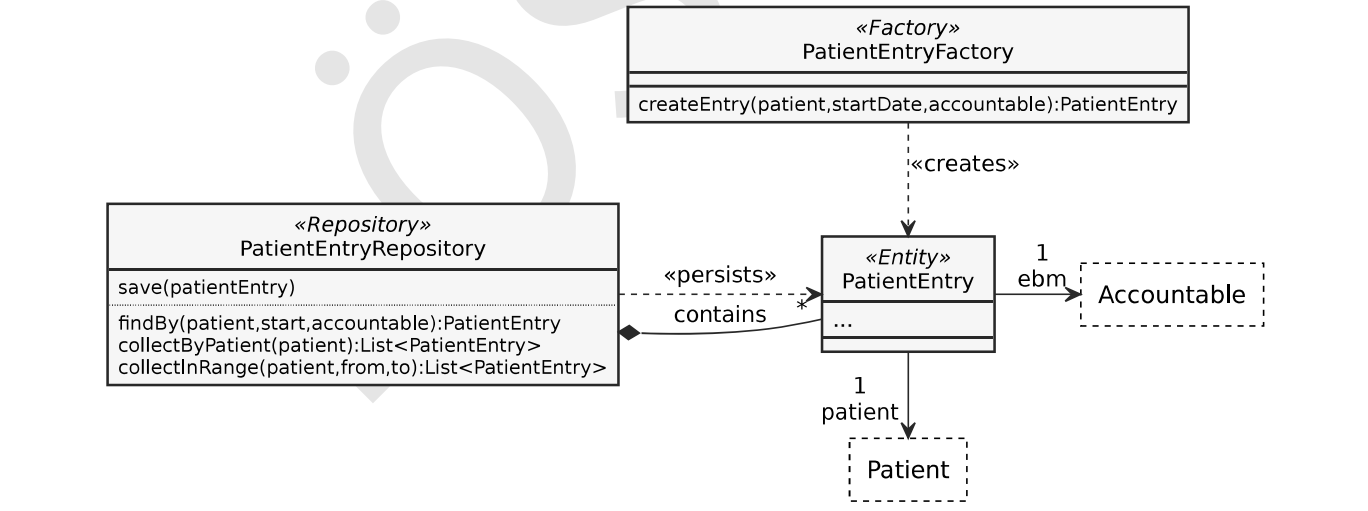
Accountable muss als `<<ValueObject>>` klassifiziert sein

- Identität von `Accountables` und alle Felder hängen direkt von `ebmNumber` ab.
- `Accountables` sind unveränderlich (und werden von der KV vorgegeben).

c) Entwerfen Sie nun die Unterstützungsmuster *Repository* und *Factory* für das Domänenkonzept `PatientEntry`. Zeichnen Sie hierfür ein UML-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 `PatientEntry` sowie ihre Beziehungen zum *Repository* beziehungsweise zur *Factory* ein. (5,5 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.

Musterlösung: [redacted]



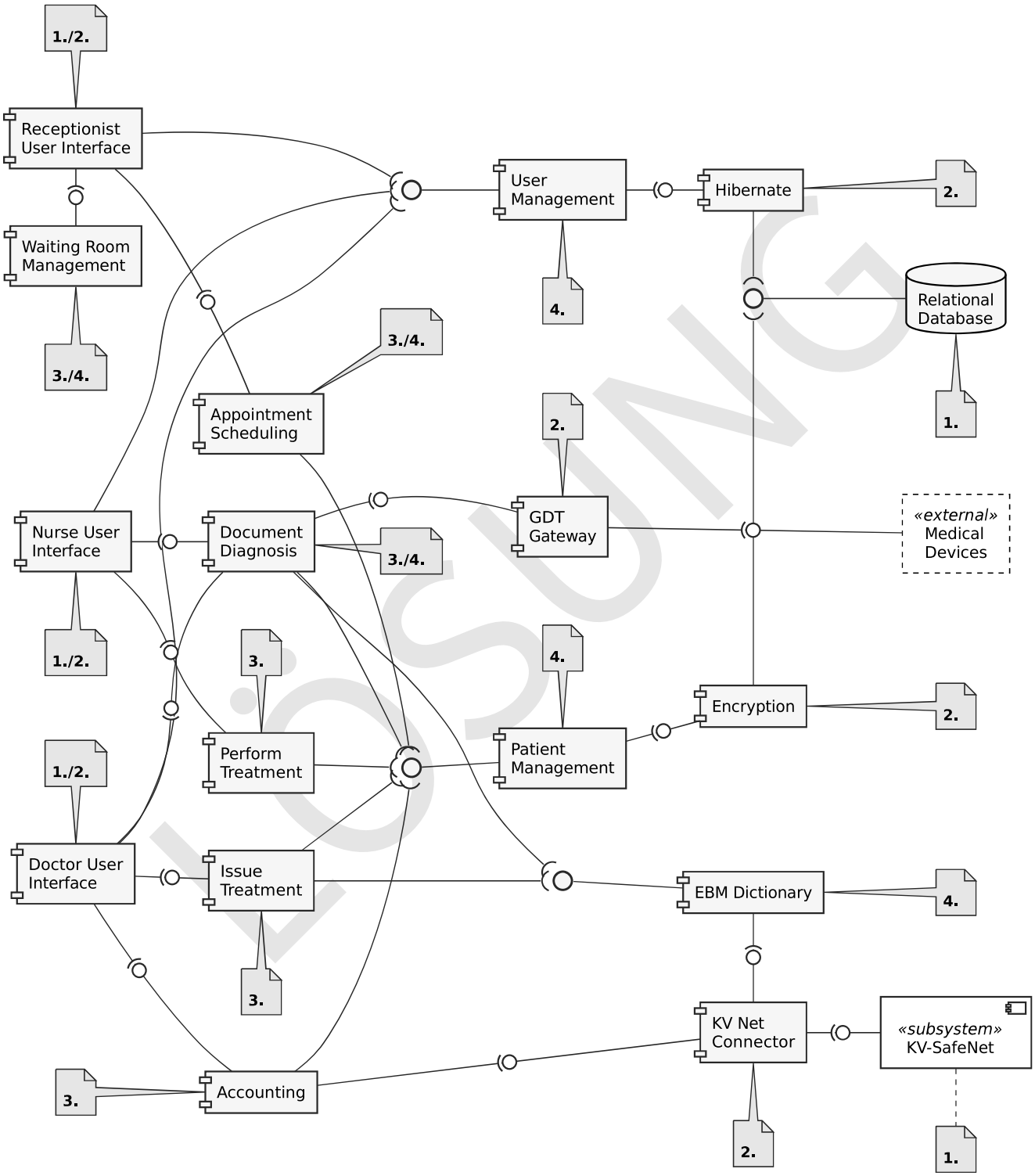
Aufgabe 5: Saubere Unternehmensarchitekturen *(16 Punkte)*

a) Tragen Sie in die Tabelle die vier Schichten der *sauberen* Architektur (clean architecture) so ein, dass sie zur angegebenen Richtung der Abhängigkeiten passen (dependency rule). (2 Punkte)

Musterlösung:

Schicht	Abhängigkeitsregel	Schichtenname
1.	↓	Framework & Drivers (External Interfaces)
2.	↓	Interface Adapter (Controllers)
3.	↓	Application Business Rules (Use Cases)
4.	↓	Enterprise Business Rules (Entities)

b) Das folgende Komponentendiagramm stellt die Softwarearchitektur eines Praxisverwaltungssystems dar. Ordnen Sie nun jede Komponente einer der Schichten *sauberer* Architekturen zu, indem Sie die Nummer der Schicht in die mit „✎“ markierten Bereiche eintragen. (9 Punkte)



Musterlösung: [redacted]

Der Quelltext in Listing 1 stellt einen Ausschnitt der Implementierung der Nutzerverwaltung (`UserManagement`) des Praxisverwaltungssystems dar. Genauer werden hier die unterschiedlichen Nutzer verwaltet und authentifiziert. Dabei sollen ausschließlich Technikerinnen (`Technicians`) und Ärztinnen (`Doctors`) als Administratoren neue Nutzer zum PVS hinzufügen können. Die hierfür entwickelte Komponente enthält jedoch einige Verstöße gegen Praktiken des sauberen Programmierens (Clean-Code-Prinzipien und Coding Conventions aus der Vorlesung).

c) 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)

Musterlösung: [redacted]

- Zeile 2: Encapsulation, Field `userName` not declared `private`
- Zeile 13: *Liskov Substitution Principle*, `changePassword` durch leere Implementierung ersetzt.
- Zeile 18: Verstoß gegen *Single Responsibility Principle*, `UserManagement` verwaltet sowohl Nutzer als auch deren Sitzungen.
- Zeile 22: Auskommentierte Eingabevalidierung.
- Zeile 26: Formatierung, schwer zu lesen da Zeilenumbrüche fehlen
- Zeile 26: *Law of Demeter*, direkter Zugriff auf Feld `userName`.
- Zeile 33: *Open Closed Principle*, `isAdmin` erlaubt keine Erweiterung um neue Administratoren.
- Zeile 48: Verstoß gegen KISS-Prinzip, Exponieren der `HashSet` Implementierung nach außen.

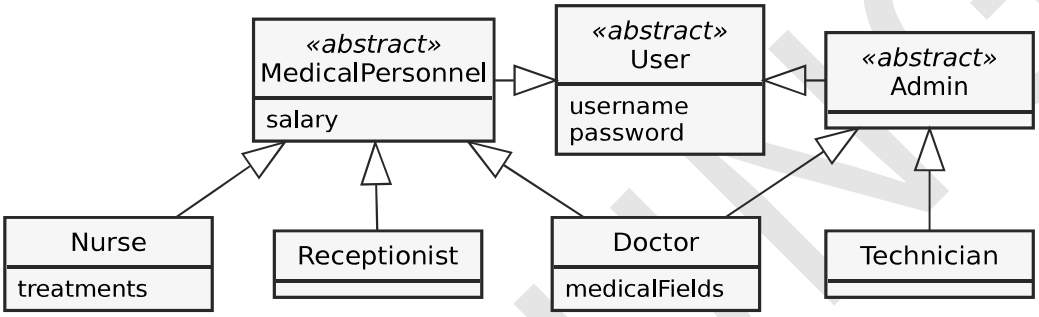
[redacted]

```java
public abstract class User {
  String userName;
  private String password;
  public User(String name, String pw) { userName = name; password = pw; }
  public void changePassword(String oldPassword, String newPassword) {
    if (verify(oldPassword)) password = newPassword;
  }
  public final boolean verify(String pw) { return password.equals(pw); }
}
public class Technician extends User {
  public Technician(String name, String pw) { super(name, pw); }
  @Override
  public void changePassword(String oldPassword, String newPassword) { /* noop */ }
}
public class Nurse extends User { /*...*/ }
public class Doctor extends User { /*...*/ }
/*...*/
public class UserManagement {
  private HashSet<User> activeSessions = new HashSet<User>();
  private HashSet<User> users = new HashSet<User>();
  public UserManagement(Technician technician) {
    //if (technician == null) throw new NullPointerException();
    users.add(technician);
  }
  private User findUserByName(String name) {
    for (User user : users) if (user.userName.equals(name)) return user;
    return null;
  }
  private boolean isActive(User user) {
    return user != null && activeSessions.contains(user);
  }
  private boolean isAdmin(User user) {
    return user!=null && (user instanceof Technician || user instanceof Doctor);
  }
  public User login(String name, String pw) {
    User user = findUserByName(name);
    if (user == null || !user.verify(pw))
      return null;
    activeSessions.add(user);
    return user;
  }
  public boolean logout(User user) {
    return isActive(activeUser) && activeSessions.remove(user);
  }
  public boolean addUser(User activeUser, User newUser) {
    return isActive(activeUser) && isAdmin(activeUser) && users.add(newUser);
  }
  public HashSet<User> getUsers() { return users; }
}
```

Listing 1: UserManagement-Klasse mit Bad Smells

Aufgabe 6: Entwurfsmuster für Unternehmenssoftware *(14 Punkte)*

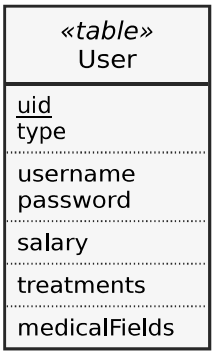
Die Nutzerverwaltung des Praxisverwaltungssystems (PVS) enthält unter anderem die folgende Klassenhierarchie, um die unterschiedlichen Arten von Nutzern des PVS abzubilden. Bei den Nutzern wird zwischen medizinischem Personal (`MedicalPersonnel`) und Administratoren (`Admin`) unterschieden. Während die Sprechstundenhilfe (`Receptionist`), Arzthelferin (`Nurse`) und die Ärztin (`Doctor`) zur ersten Gruppe gehören, zählen nur die Technikerin (`Technician`) und die Ärztin (`Doctor`) zu den Administratoren. Um diese Klassenhierarchie in einer Datenbank abzubilden, sollen Sie in dieser Aufgabe die unterschiedlichen Objekt-Relationalen Strukturmuster (*object-relational structural patterns*), welche in der Vorlesung behandelt wurden, anwenden.



a) Erstellen Sie ein Datenbankschema nach dem Strukturmuster *Single Table Inheritance*. Erstellen Sie hierzu ein UML-Klassendiagramm der nötigen Tabellen (dargestellt als UML-Klasse mit dem `<<table>>` Stereotyp) und ihrer Attribute. Führen Sie, falls nötig, neue Attribute ein, um die *Primärschlüssel* und *Vererbung* darzustellen. (3 Punkte)

Hinweis: Datentypen (wie zum Beispiel `int`, `long`, `Date`, `String`, ...) müssen nicht angegeben werden.

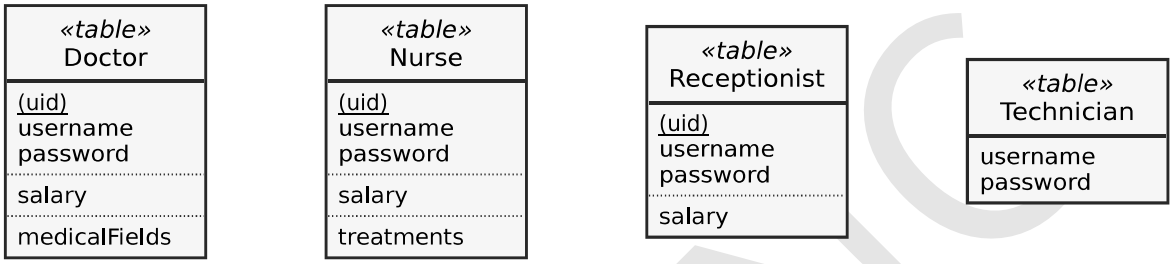
Musterlösung:



b) Erstellen Sie ein Datenbankschema nach dem Strukturmuster *Concrete Table Inheritance*. Erstellen Sie hierzu ein UML-Klassendiagramm der nötigen Tabellen (dargestellt als UML-Klasse mit dem `<<table>>` Stereotyp) und ihrer Attribute. Führen Sie, falls nötig, neue Attribute ein, um *Primärschlüssel* und *Vererbung* darzustellen. (6,5 Punkte)

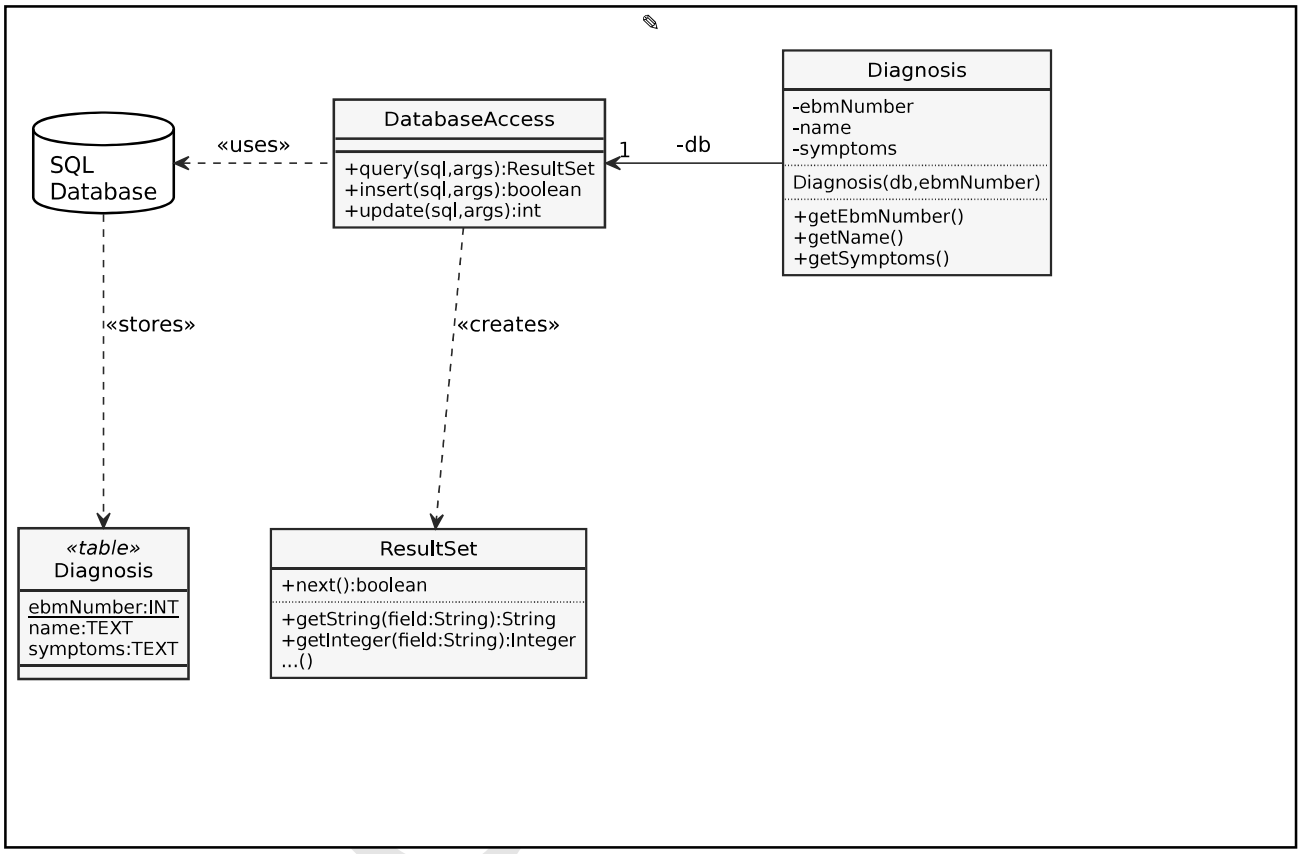
Hinweis: Datentypen (wie zum Beispiel `int`, `long`, `Date`, `String`, ...) müssen nicht angegeben werden.

Musterlösung



Abschließend betrachten wir die Anbindung einer existierenden Datenbank für einheitliche Bewertungsmaßstäbe (EBM) von Diagnosen an das Praxisverwaltungssystem. Die nachfolgende Abbildung zeigt die Diagnosetabelle (`Diagnosis`) mit der EBM-Nummer (`ebmNumber`) als Primärschlüssel sowie den Feldern `name` und `symptoms`. Um auf diese Tabelle zuzugreifen, sollen Sie die existierende Klasse `DatabaseAccess` verwenden.

Musterlösung



c) Skizzieren Sie, wie die Diagnosetabelle (`Diagnosis`) mit Hilfe des *Domain Model* Strukturmusters zur Repräsentation von Domänenlogik *(domain logic pattern)* im Praxisverwaltungssystem abgebildet werden kann. Zeichnen Sie hierzu die notwendige Klasse und die notwendigen Methoden inklusive ihrer Parameter und Assoziation in das obige UML-Klassendiagramm ein. (4,5 Punkte)

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