abitur.nrw
BERUFSKOLLEG
Berufliches Gymnasium

Zentrale Abiturprüfung 2022
Haupttermin
26.04.2022

Profil bildendes Leistungskursfach
Informatik
Fachbereich Informatik

Bearbeitungszeit: 270 Minuten
zusätzliche Rüstzeit: 30 Minuten

Unterlagen für die Schülerinnen und Schüler

Aufgabenstellung

Beschreibung der Ausgangssituation:

Die in Düsseldorf tätige Gemeinschaftspraxis Frau Dr. Spahn und Partnerinnen für hausmedizinische Versorgung möchte mit Ihrer Hilfe ihre EDV-Verwaltung optimieren. Neben der effizienten Verwaltung der Patientendaten sollen auch sicherheitsrelevante Aspekte berücksichtigt werden.

Aufgabe 1 – Datenbanksysteme

1.1 (6 Punkte)

Beschreiben Sie drei Anforderungen an ein DBMS bezogen auf die konkrete Anwendung in einer Arztpraxis.

Es soll ein ER-Modell für die Abläufe in der Praxis entwickelt werden. Hierzu haben Sie bereits mit den Ärztinnen der Praxis gesprochen und folgende Informationen gesammelt:

In der Praxis arbeiten mehrere Ärztinnen. Jeder Patientin und jedem Patienten wird eine Ärztin als hauptverantwortliche Betreuerin zugewiesen. Einzelne Behandlungen der Patientinnen und Patienten können jeweils auch von einer anderen Ärztin der Praxis durchgeführt werden. Es gibt Ärztinnen und Ärzte (z. B. Vertretungen), die keine Patientinnen oder Patienten betreuen, aber Behandlungen durchführen. Eine Behandlung findet zu einem bestimmten Datum statt und besteht aus einer oder mehreren Positionen. Eine Position hat einen Grundpreis, der bei einer konkreten Behandlung mit einem Faktor multipliziert wird, der die Schwierigkeit dieser Behandlung beschreibt. Eine oder mehrere Behandlungen werden in einer Rechnung zusammengefasst und abgerechnet. Nicht jede Behandlung wird abgerechnet.
Jede Patientin und jeder Patient ist bei genau einer Krankenkasse versichert. In der Datenbank sollen auch Krankenkassen gespeichert werden, denen keine Patientin und kein Patient der Praxis angehört.

1.2 (21 Punkte)

Entwerfen Sie aus den obigen Informationen ein ER-Diagramm in MC-Notation. Als Attribute sollen lediglich die im Text genannten Attribute angegeben werden.

Der Einsatz von DIA ist zwingend vorgeschrieben.

In einem ersten Entwurf wurden folgende Relationen zur Erfassung von Krankenkassendaten (R1) und Patientendaten (R2) angelegt:

R1 (KKID, Kassenname, Hauptsitz, TID, Erstattungssatz)

R2 (PID, PNachname, PVorname, KKID, TID, Kassenname, Hauptsitz)

Für R1 wurden als zusammengesetzter Primärschlüssel die Krankenkassen-ID (KKID) und die Tarif-ID (TID) gewählt.

Dabei gelten folgende Abhängigkeiten:

Aus der Krankenkassen-ID folgen der Name der Krankenkasse und der Ort des Hauptsitzes.

Der Erstattungssatz ist abhängig von der Kombination aus KKID und TID.

1.3 (8 Punkte)

Untersuchen Sie, ob die Relationen R1 bzw. R2 in der zweiten bzw. dritten Normalform sind.

1.4 (8 Punkte)

Überführen Sie das Relationenschema (R1, R2) in die dritte Normalform.
Kennzeichnen Sie Primärschlüssel durch Unterstreichen und Fremdschlüssel durch Voranstellen eines Pfeils (↑).

Im Folgenden ist ein Auszug aus den modellierten Tabellen des DBMS der Praxis gegeben:

Arzt:

ArztID	Name	Vorname	Fachgebiet	GebDatum	Durchwahl
2117	Dr. Spahn	Jelena	Innere Medizin	1972-04-26	-03
1492	Dr. Laumann	Karla	Onkologie	1952-06-13	-01
…	…	…	…	…	…

Patient:

PatientID	Name	Vorname	GebDatum	Geschlecht	Wohnort	HausarztID	KKID
P21461	Müller	Lieschen	1982-04-16	W	Düsseldorf	2117	BAR1621
G51332	Meyer	Otto	1966-12-29	M	Ratingen	1492	AOK1442
G54119	Sari	Julien	1985-09-19	D	Düsseldorf	2117	AOK1442
…	…	…	…	…	…	…	…

Krankenkasse:

KKID	Name	Versicherungsart	Mitglieder
BAR1621	Barmenia	Privat	1.500.000
AOK1442	AOK Rheinland	Gesetzlich	3.000.000
…	…	…	…

1.5 (11 Punkte)

Ermitteln Sie die SQL-Befehle, um die Tabelle Patient zu erzeugen und den ersten Datensatz (Frau Müller) hinzuzufügen.

1.6 (7 Punkte)

Bestimmen Sie eine SQL-Abfrage, die die Namen aller Patientinnen und Patienten ausgibt, die bei einer großen Versicherung (mindestens 2.000.000 Mitglieder) versichert sind.

Auszugeben sind der Name, Vorname und Geburtsdatum der Patientin bzw. des Patienten alphabetisch sortiert nach Namen und bei gleichem Namen nach Vornamen.

Für eine Studie sollen die Patienten-IDs aller weiblichen Patientinnen aus Düsseldorf und Ratingen ausgegeben werden, die am 01.05.2022 mindestens 65 Jahre alt sind.

1.7 (7 Punkte)

Ermitteln Sie eine VIEW mit der Bezeichnung „Studie“ für diese Abfrage.

1.8 (7 Punkte)

Erstellen Sie eine SQL-Abfrage, die alle Krankenkassen mit insgesamt mehr als 10 Patientinnen oder Patienten aus Düsseldorf auflistet.

Auszugeben sind die KKID und die Anzahl der Patientinnen bzw. Patienten.

Aufgabe 2 – Dynamische Datenstrukturen

Für das Wartezimmer soll eine Verwaltungssoftware eingesetzt werden, die den Eintritt der Patientin oder des Patienten bei der Anmeldung am Tresen der Sprechstundenhilfen erfasst und eine Warteliste anlegt.

Grundsätzlich ist es möglich, eine solche Warteliste mit einer Queue zu realisieren.

2.1 (8 Punkte)

Erklären Sie den Datentyp Queue mit seinen Operationen enqueue, dequeue und front.

Alternativ zur Queue kann zur Realisierung der Warteliste der Datentyp List verwendet werden.

2.2 (6 Punkte)

Erklären Sie, welche zusätzlichen Möglichkeiten dieser Datentyp für die Abbildung von Situationen im Wartezimmer bietet.

Bei der Analyse der Anforderungen an die Software wird folgendes Szenario untersucht.

Um 12:00 Uhr befinden sich noch sechs Patientinnen oder Patienten auf der Warteliste:

Patienten-ID	P21416	G51332	G62162	P29841	G15423	G11678
Ankunft Praxis	11:15 Uhr	11:00 Uhr	11:20 Uhr	11:45 Uhr	11:20 Uhr	11:20 Uhr
Termin	11:30 Uhr	ohne	11:45 Uhr	ohne	11:30 Uhr	11:45 Uhr
Notfall	nein	nein	nein	ja	nein	nein
Erwartete Behandlungsdauer	10 Min	20 Min	5 Min	15 Min	20 Min	20 Min

Das System wählt die nächste Person nach den folgenden Regeln aus:
(Regel 1 hat Vorrang vor Regel 2 usw.)

1. Notfälle werden vorrangig vor allen anderen behandelt.
2. Personen mit Termin werden vorrangig vor denen ohne Termin behandelt.
3. Personen mit längerer Wartezeit werden vorrangig behandelt.
   Die Wartezeit wird bei Personen mit Termin ab der Uhrzeit des Termins berechnet, bei Personen ohne Termin ab der Ankunft in der Praxis.
4. Personen mit erwarteter kürzerer Behandlungsdauer werden vorrangig behandelt.

2.3 (8 Punkte)

Ermitteln Sie, in welcher Reihenfolge die sechs Personen aufgerufen werden.
Nehmen Sie dabei an, dass keine weiteren Personen ins Wartezimmer kommen.

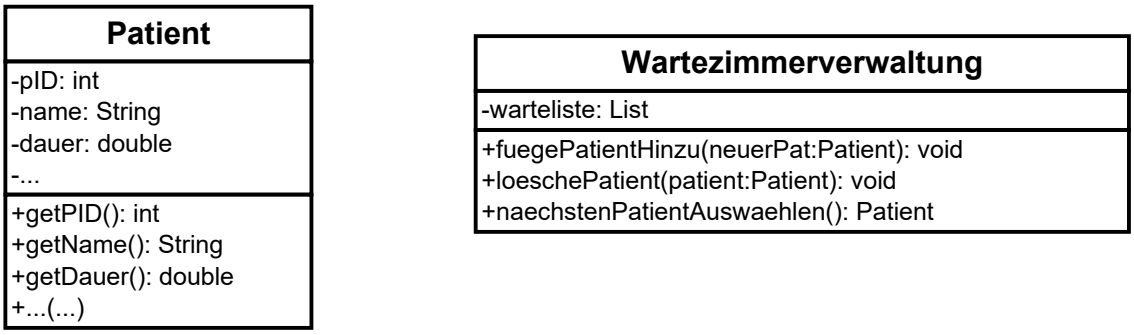
Wenn man von Notfällen absieht, können Wartelisten auch nach dem Shortest-Job-First-Prinzip abgearbeitet werden. Dabei wird immer der Job als nächstes ausgewählt, der die kürzeste Zeit benötigt.

Im Anwendungskontext ist ein Job die Behandlung einer Patientin bzw. eines Patienten.

2.4 (6 Punkte)

Erläutern Sie einen Vorteil und einen Nachteil des Shortest-Job-First-Prinzips. Nehmen Sie dabei Bezug auf den Anwendungskontext des Wartezimmers.

Für die Implementierung wird folgender Auszug des Klassendiagramms verwendet. Die Implementierung der Klasse List entspricht der Dokumentation im Anhang 1.



Patientinnen oder Patienten werden zunächst in der Reihenfolge des Eintreffens in der Praxis am Ende der Warteliste eingefügt.

2.5 (14 Punkte)

Implementieren Sie die Methode `naechstenPatientAuswaehlen`, wenn nach dem Shortest-Job-First-Prinzip in Bezug auf die Behandlungsdauer (Attribut `dauer`) verfahren werden soll.

Dabei soll die Patientin bzw. der Patient nicht aus der Liste entfernt werden.

2.6 (6 Punkte)

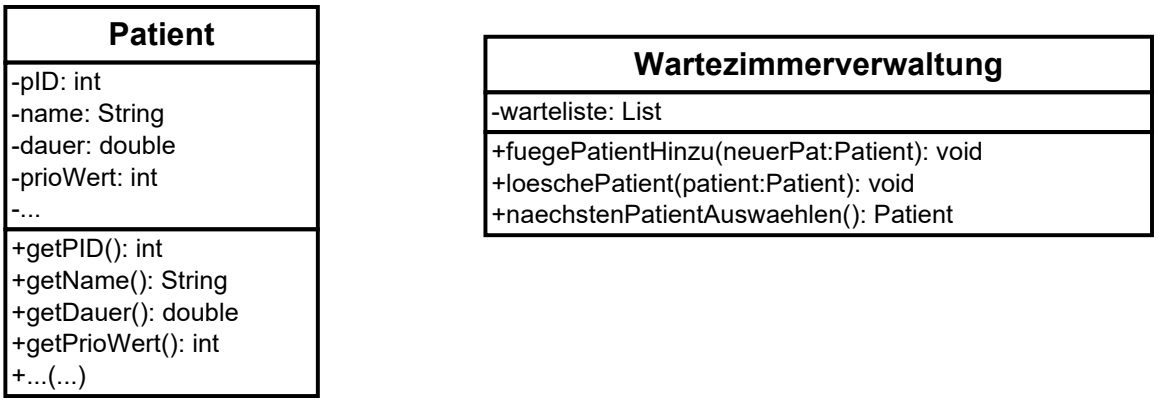
Beschreiben Sie die Funktionsweise einer Warteliste, die die Vorteile des Shortest-Job-First-Prinzips grundsätzlich beibehält, aber verhindert, dass einzelne Patientinnen oder Patienten mit längerer Behandlungsdauer ganz bis zum Schluss warten müssen.

2.7 (8 Punkte)

Implementieren Sie die Methode `loeschePatient` der Klasse `Wartezimmerverwaltung` des obigen Klassendiagramms, die das übergebene Objekt vom Typ `Patient` aus der Warteliste entfernt.

In einer weiterentwickelten Version der Wartezimmerverwaltung wird jeder Person beim Empfang eine Priorität zugewiesen. Die Priorität wird durch einen Zahlenwert zwischen 1 und 5 ausgedrückt (1= höchste Priorität, 5= niedrigste Priorität).

Neue Personen sollen nun sortiert entsprechend dieser Priorität in die anfangs leere Warteliste der Klasse `Wartezimmerverwaltung` eingefügt werden. Dafür soll die Methode `fuegePatientHinzu` verwendet werden. Personen mit derselben Priorität werden in der Reihenfolge der Ankunft in der Praxis einsortiert.



2.8 (14 Punkte)

Implementieren Sie die Methode `fuegePatientHinzu`.

Die Ärztinnen legen nach Abschluss einer Behandlung die Patientenakte als oberstes in den dafür vorgesehenen Korb. Angestellte aus der Verwaltung nehmen jeweils eine Akte vom Stapel, um die Behandlungsdaten ins Abrechnungssystem einzupflegen und die Akte ins Archiv zu bringen. Dies kann durch einen Stack modelliert werden.

2.9 (5 Punkte)

Erklären Sie die Standardoperationen der dynamischen Datenstruktur Stack.

Aufgabe 3 – Kryptologie

Der Schutz der Patientendaten hat für die Arztpraxis Dr. Spahn hohe Priorität. Aus diesem Grund wendet sich die Praxis an ein externes IT-Unternehmen, um sich fachgerecht beraten zu lassen.

3.1 (6 Punkte)

Erklären Sie, was die Einhaltung der kryptographischen Schutzziele Vertraulichkeit, Authentizität und Integrität im Kontext von Patientendaten bedeutet.

Dr. Spahn erwähnt im Gespräch mit dem IT-Unternehmen die Verschlüsselungsmethode Atbasch, die bereits in der Bibel benutzt wurde:

Grundlage für die Atbasch ist, dass man alle Buchstaben in einer Doppelreihe anordnet, in der oberen Reihe die ersten 13 Buchstaben in aufsteigender Folge, darunter die letzten 13 Buchstaben in absteigender Folge:

A	B	C	D	E	F	G	H	I	J	K	L	M
Z	Y	X	W	V	U	T	S	R	Q	P	O	N

Demnach wird beispielsweise ein A durch ein Z oder ein U durch ein F ersetzt.

3.2 (5 Punkte)

Bestimmen Sie die Verschlüsselung des Wortes IMPFUNG unter Anwendung des Atbasch-Verfahrens.

3.3 (11 Punkte)

Implementieren Sie eine Methode `encrypt`, welche den als Parameter übergebenen Text mit Atbasch verschlüsselt und zurückgibt. Gehen Sie davon aus, dass der übergebene Text lediglich aus Großbuchstaben besteht.

3.4 (6 Punkte)

Analysieren Sie die Sicherheit des Atbasch-Verfahrens.

Nach dieser Analyse schlägt der Chef des IT-Unternehmens die Verwendung eines asymmetrischen Verschlüsselungsverfahrens vor.

3.5 (5 Punkte)

Erklären Sie den prinzipiellen Unterschied zwischen symmetrischen und asymmetrischen Verschlüsselungsverfahren bei der Kommunikation zweier Kommunikationspartner.

3.6 (7 Punkte)

Erklären Sie, wie und warum symmetrische und asymmetrische Verschlüsselungsverfahren zu hybriden Verschlüsselungsverfahren kombiniert werden.

Als asymmetrisches Verfahren wird RSA vorgeschlagen.

Zur Erläuterung soll das Verfahren mit den Primzahlen p=23 und q=31 demonstriert werden, wobei als öffentlicher Schlüsselbestandteil e=43 gewählt wird.

3.7 (11 Punkte)

Ermitteln Sie zu den genannten Zahlen den privaten Schlüssel d unter Angabe der Schritte des erweiterten euklidischen Algorithmus.

Außerdem soll die RSA-Verschlüsselung unter Verwendung der obigen Zahlen p=23, q=31 und e=43 dargestellt werden.

3.8 (9 Punkte)

Berechnen Sie die Chiffre zur Nachricht m=3 unter Angabe der Schritte des Square-and-Multiply-Algorithmus.

Die Praxis plant die Einführung einer digitalen Patientenakte. Auszüge aus dieser Patientenakte sollen an die Patientin oder den Patienten über das Internet übermittelt werden. Diese können auch sehr große Datenmengen wie z. B. hochaufgelöste Röntgenbilder enthalten. Zur Absicherung der Kommunikation erhalten jede Patientin und jeder Patient einmalig in der Arztpraxis eine Chipkarte ausgehändigt.

Dabei sollen die Schutzziele Geheimhaltung und Authentizität erreicht werden.

3.9 (15 Punkte)

Entwerfen Sie ein auf RSA basierendes Konzept, welches die beiden genannten Schutzziele unter Nutzung der Chipkarte realisiert.

Gehen Sie davon aus, dass jede Patientin und jeder Patient im Besitz geeigneter Hard- und Software sind.

Materialgrundlage

Alle Materialien wurden selbst erstellt.

Zugelassene Hilfsmittel

- Wörterbuch der deutschen Rechtschreibung
- Graphikfähiger Taschenrechner (GTR) oder Computeralgebrasystem (CAS)
- Eine Nutzung von Computersystemen ist verpflichtend vorgesehen. Als Programme werden „Dia“ Version 0.97 oder höher und ein Texteditor eingesetzt.

Punktevergabe und Arbeitszeit

Inhaltliche Leistung	225 Punkte
Darstellungsleistung	15 Punkte
Gesamtpunktzahl	240 Punkte

Bearbeitungszeit	270 Minuten
zusätzliche Rüstzeit	30 Minuten

Anhang 1 (zu Teilaufgaben 2.5, 2.7 und 2.8)

Dokumentation der Klasse List

Konstruktor	**List()** Eine leere Liste wird erzeugt.
Anfrage	**boolean isEmpty()** Die Anfrage liefert den Wert true, wenn die Liste keine Objekte enthält, sonst liefert sie den Wert false.
Anfrage	**boolean hasAccess()** Die Anfrage liefert den Wert true, wenn es ein aktuelles Objekt gibt, sonst liefert sie den Wert false.
Auftrag	**void next()** Falls die Liste nicht leer ist, es ein aktuelles Objekt gibt und dieses nicht das letzte Objekt der Liste ist, wird das dem aktuellen Objekt in der Liste folgende Objekt zum aktuellen Objekt, andernfalls gibt es nach Ausführung des Auftrags kein aktuelles Objekt.
Auftrag	**void toFirst()** Falls die Liste nicht leer ist, wird das erste Objekt der Liste aktuelles Objekt. Ist die Liste leer, geschieht nichts.
Auftrag	**void toLast()** Falls die Liste nicht leer ist, wird das letzte Objekt der Liste aktuelles Objekt. Ist die Liste leer, geschieht nichts.
Anfrage	**Object getObject()** Falls es ein aktuelles Objekt gibt, wird das aktuelle Objekt zurückgegeben, andernfalls gibt die Anfrage den Wert null zurück.
Auftrag	**void setObject(Object pObject)** Falls es ein aktuelles Objekt gibt und pObject ungleich null ist, wird das aktuelle Objekt durch pObject ersetzt.
Auftrag	**void append(Object pObject)** Ein neues Objekt pObject wird am Ende der Liste angefügt. Das aktuelle Objekt bleibt unverändert. Wenn die Liste leer ist, wird das Objekt pObject in die Liste eingefügt und es gibt weiterhin kein aktuelles Objekt. Falls pObject gleich null ist, bleibt die Liste unverändert.
Auftrag	**void insert(Object pObject)** Falls es ein aktuelles Objekt gibt, wird ein neues Objekt vor dem aktuellen Objekt in die Liste eingefügt. Das aktuelle Objekt bleibt unverändert. Falls die Liste leer ist und es somit kein aktuelles Objekt gibt, wird pObject in die Liste eingefügt und es gibt weiterhin kein aktuelles Objekt. Falls es kein aktuelles Objekt gibt und die Liste nicht leer ist oder pObject gleich null ist, bleibt die Liste unverändert.
Auftrag	**void concat(List pList)** Die Liste pList wird an die Liste angehängt. Das aktuelle Objekt bleibt unverändert. Falls pList null oder eine leere Liste ist, bleibt die Liste unverändert.
Auftrag	**void remove()** Falls es ein aktuelles Objekt gibt, wird das aktuelle Objekt gelöscht und das Objekt hinter dem gelöschten Objekt wird zum aktuellen Objekt. Wird das Objekt, das am Ende der Liste steht, gelöscht, gibt es kein aktuelles Objekt mehr. Wenn die Liste leer ist oder es kein aktuelles Objekt gibt, bleibt die Liste unverändert.