Einheitliche Prüfungsanforderungen

Informatik

(Beschluss der Kultusministerkonferenz vom 01.12.1989 i.d.F. vom 05.02.2004)

Die Länder werden gebeten, die neugefassten Einheitlichen Prüfungsanforderungen in der Abiturprüfung (EPA) für die Fächer Biologie, Physik, Chemie, Informatik, Französisch, Italienisch, Spanisch, Russisch, Türkisch und Dänisch spätestens zur Abiturprüfung im Jahre 2007 umzusetzen. (Beschluss der Kultusministerkonferenz vom 12.03.2004)

Inhaltsverzeichnis

I Festlegung für die Gestaltung der Abiturprüfung

Fachpräambel

Die Vereinbarung zur Gestaltung der gymnasialen Oberstufe in der Sekundarstufe II (Beschluss der Kultusministerkonferenz vom 07.07.1972 i. d. F. vom 16.06.2000) beschreibt die grundlegenden Anforderungen an den Unterricht im mathematisch-naturwissenschaftlich-technischen Aufgabenfeld:

„Im mathematisch-naturwissenschaftlich-technischen Aufgabenfeld sollen Verständnis für den Vorgang der Abstraktion, die Fähigkeit zu logischem Schließen, Sicherheit in einfachen Kalkülen, Einsicht in die Mathematisierung von Sachverhalten, in die Besonderheiten naturwissenschaftlicher Methoden, in die Entwicklung von Modellvorstellungen und deren Anwendung auf die belebte und unbelebte Natur und in die Funktion naturwissenschaftlicher Theorien vermittelt werden."

Informatik ist die Wissenschaft, die sich mit der systematischen Darstellung, Speicherung und Übertragung von Information sowie der automatisierten Verarbeitung mit Computern befasst. Sie beschäftigt sich mit der Entwicklung formaler, maschinell durchführbarer Verfahren zur Lösung von Informationsverarbeitungsproblemen und der Bewertung des Einsatzes dieser Verfahren hinsichtlich Chancen und Risiken. Die Informatik spielt eine Schlüsselrolle bei der Entwicklung der Informations- und Kommunikationstechniken. Überall im beruflichen und privaten Leben, in Wissenschaft und Wirtschaft kommen komplexe Systeme zur Informationsverarbeitung zum Einsatz, deren Leistungsfähigkeit und Weiterentwicklung oft den Fortschritt bestimmen. Durch den Einsatz weltweiter Netze wird Information in einem bisher nicht bekannten Maße erschlossen.

Der Informatikunterricht in der gymnasialen Oberstufe leistet einen spezifischen Beitrag zur Allgemeinbildung, indem er den Erwerb eines systematischen, zeitbeständigen und über bloße Bedienerfertigkeiten hinausgehenden Basiswissens über die Funktionsweise, die innere Struktur sowie die Möglichkeiten und Grenzen von Informatiksystemen ermöglicht. Dadurch wird deren sinnvolle, kompetente und verantwortungsbewusste Nutzung und Beurteilung ermöglicht. Die Schülerinnen und Schüler machen sich mit den Denkweisen vertraut, die den Informations- und Kommunikationstechniken zugrunde liegen, und lernen dadurch auch deren prinzipielle Chancen und Risiken einzuschätzen.

Neben der Vermittlung von grundlegenden Konzepten, die sich durch Allgemeingültigkeit und Zeitbeständigkeit auszeichnen, entwickelt der Informatikunterricht übergeordnete Kompetenzen weiter und stellt Arbeitsweisen und Methoden bereit, die im Alltag, in Studium und Beruf sowie in Wissenschaft und Wirtschaft erforderlich und von Nutzen sind. So lernen die Schülerinnen und Schüler Ordnungsprinzipien kennen, die zur Orientierung in einer hoch komplexen, vernetzten Welt beitragen und die bei der Erschließung der rasch fortschreitenden Entwicklungen auf dem Sektor der Informationstechnologie, aber auch in vielen anderen Bereichen helfen.

Informatische Methoden wie das Strukturieren, das systematische Zerlegen komplexer Systeme in überschaubare Teile, das Formalisieren und Interpretieren fördern und fordern die Abstraktionsfähigkeit und das Erfassen logischer Zusammenhänge. Bei der Modellbildung, die bei der Konstruktion und Analyse von Informatiksystemen eine entscheidende Rolle spielt, üben die Schüler in besonderem Maße, eine Situation von verschiedenen Standpunkten aus zu beurteilen; die systematische Überprüfung und kritische Beurteilung der Ergebnisse sowie des gewählten Modells fördern die Fähigkeit zu konstruktiver Kritik. Gleichzeitig werden die für den erfolgreichen Einsatz des Computers nötige Sorgfalt, Genauigkeit und Ausdauer gefördert.

Diese Ziele bedingen einen Informatikunterricht, bei dem problem-, handlungs- und projektorientiertes Arbeiten im Vordergrund stehen. Die enge Verzahnung der Informatik mit Anwendungen aus Wissenschaft, Wirtschaft, Technik oder auch Verwaltung spiegelt sich insbesondere in der Wahl der Beispiele wider. Da je nach Art des Anwendungszusammenhangs unterschiedliche Betrachtungsweisen notwendig sind, beschäftigen sich die Schüler dementsprechend im Lauf des Unterrichts mit verschiedenen Modellierungstechniken und Betrachtungsweisen, die sie bei zunehmend komplexen, umfangreichen Aufgabenstellungen anwenden. Es wird deutlich, dass die Informatik Erkenntnisse anderer Disziplinen nutzt, dass umgekehrt aber Denkweisen und Verfahren in anderen Gebieten vielfältig zur Anwendung kommen und so auch andere Disziplinen beeinflusst werden.

Die Lösung eines komplexen informatischen Problems impliziert außerdem Teamorientierung und arbeitsteiliges Vorgehen, wodurch soziale Kompetenzen wie Teamfähigkeit, Zuverlässigkeit, Verantwortungsbereitschaft gefördert werden.

Die spezifischen fachlichen und methodischen Kompetenzen, die im Informatikunterricht erworben werden und für die Abiturprüfung zur Verfügung stehen müssen, sind im Einzelnen in Abschnitt 1.1 beschrieben.

Zur Sicherung eines einheitlichen und angemessenen Anforderungsniveaus in den Prüfungsaufgaben enthalten die Einheitlichen Prüfungsanforderungen für das Fach Informatik

- eine Beschreibung der Prüfungsgegenstände, d. h. der nachzuweisenden Kompetenzen sowie der fachlichen Inhalte, an denen diese Kompetenzen eingefordert werden sollen,
- Kriterien, mit deren Hilfe überprüft werden kann, ob eine Prüfungsaufgabe das anzustrebende Anspruchsniveau erreicht,
- Hinweise und Aufgabenbeispiele für die Gestaltung der schriftlichen und mündlichen Prüfung sowie zu alternativen Prüfungsformen.

Die im Folgenden aufgeführten nachzuweisenden fachlichen Kompetenzen gelten sowohl für die Prüfungen im Grundkurs- als auch im Leistungskursfach.

Als Hilfsmittel für die Konstruktion von Prüfungsaufgaben sowie für die Gestaltung der mündlichen Prüfung und alternativer Prüfungsformen dient die Beschreibung der drei Anforderungsbereiche. Mit ihrer Hilfe und nach Maßgabe des vorangegangenen Unterrichts, dem die Lehrpläne der Länder zugrunde liegen, werden Prüfungsinhalte ausgewählt und Prüfungsaufgaben erstellt.

1 Fachliche Inhalte und Qualifikationen

1.1 Fachliche und methodische Kompetenzen

Im Grund- und Leistungskursfach Informatik werden in enger Kopplung mit den Inhalten fachspezifische und allgemeine methodische Kompetenzen erworben. Die Anforderungen für die schriftliche und mündliche Prüfung sowie für alternative Prüfungskomponenten sind so zu gestalten, dass nach Möglichkeit ein breites Spektrum von Kompetenzen an geeigneten Inhalten überprüft werden kann. Hierzu werden die erforderlichen Kompetenzen in folgende Kompetenzbereiche gegliedert:

Erwerb und Strukturierung informatischer Kenntnisse

Die Prüflinge

- verfügen über strukturiertes informatisches Basiswissen,
- haben gefestigte Kenntnisse über Grundprinzipien und Basiskonzepte der Informatik und verfügen über Methoden und Strategien des selbstständigen Wissenserwerbs und der Strukturierung informatischer Kenntnisse.

Kennen und Anwenden informatischer Methoden

Die Prüflinge

- können Informatiksysteme zur Lösung einer Aufgabenstellung konfigurieren und anpassen,
- können verschiedene Problemlösungsstrategien und Techniken wie Iteration, Rekursion und Klassenbildung einsetzen,
- sind insbesondere mit dem Modellbildungszyklus vertraut; dazu gehören in problemadäquater Auswahl und Reihenfolge: Problemanalyse und Problemspezifikation, Abgrenzen des Problems, Abstraktion,

Idealisierung, Strukturieren und Zerlegen in Teilprobleme (Modularisieren), Formalisieren, Umsetzen unter Berücksichtigung der zur Verfügung stehenden Werkzeuge und Hilfsmittel, Testen der Lösung, kritisches Reflektieren der Ergebnisse und der Lösung allgemein, Überarbeitung des Modells, Optimierung der Lösung.

Kommunizieren und Kooperieren

Die Prüflinge

- können im Team arbeiten,
- organisieren und koordinieren die Arbeit in Projektgruppen,
- verwenden die Fachsprache angemessen,
- veranschaulichen und beschreiben Sachverhalte u. a. mit Hilfe von Texten und Diagrammen,
- können den Arbeitsablauf und die Arbeitsergebnisse dokumentieren und
- können Lern- und Arbeitsergebnisse adressatengerecht präsentieren.

Anwenden informatischer Kenntnisse, Bewerten von Sachverhalten und Reflexion von Zusammenhängen

Die Prüflinge

- können Informations- und Kommunikationssysteme zum Erschließen, Austauschen und Verarbeiten von Information nutzen,
- können zur Lösung eines anwendungsbezogenen Problems adäquate Verfahren und Werkzeuge selbstständig auswählen und diese sicher und reflektiert einsetzen,
- können ihre vielfältigen Erfahrungen bei der Bearbeitung von Problemen aus verschiedenen Anwendungsfeldern auf die Lösung ähnlicher Fragestellungen übertragen,
- sind in der Lage, die eigene Arbeit und die Arbeit Anderer kritisch zu reflektieren und
- können typische Einsatzbereiche, Möglichkeiten, Grenzen, Chancen und Risiken von Informations- und Kommunikationssystemen untersuchen und einschätzen.

1.2 Fachliche Inhalte

Die im Folgenden genannten drei Lern- und Prüfungsbereiche sind für das Grundkursfach und das Leistungskursfach verbindlich. Im Leistungskursfach erfolgt eine tiefergehende, erweiterte und systematischere Behandlung von Inhalten, die auf vertieftes Verständnis und Reflexion abzielt. Entsprechend unterscheiden sich die Anforderungen im Schwierigkeits- und Komplexitätsgrad sowie in Bezug auf die Selbstständigkeit bei der Bearbeitung.

Grundlegende Modellierungstechniken

- Grundprinzip des Modellierens als zielgerichtetes Vereinfachen und strukturiertes Darstellen von Ausschnitten der Wirklichkeit, Erstellen eines Modells auf der Grundlage der Problemanalyse
- Einsatz verschiedener grundlegender Betrachtungsweisen im Rahmen von Problemlösungen, Kenntnisse der folgenden Modellierungstechniken, mindestens zwei im Grundkursfach und mindestens drei im Leistungskursfach:

- Objektorientierte Modellierung
  insbesondere: Objekt, Klasse, Beziehungen zwischen Klassen, Interaktion von Objekten, Klassendiagramm (z. B. mit UML)
- Datenmodellierung
  insbesondere: semantisches Datenmodell (Beschreibung der relevanten Objekte und ihrer Beziehungen, ER-Modell), logisches Datenmodell (z. B. relationales Datenmodell)
- Zustandsorientierte Modellierung
  insbesondere: Variablenkonzept, Automaten (Zustände und Zustandsübergänge), Zustandsdiagramme
- Modellierung von Abläufen mit Algorithmen
  insbesondere: Algorithmusbegriff, Ablaufstrukturen, einfache und höhere Datenstrukturen, Zerlegen in Teilalgorithmen; Struktogramme; spezielle Verfahren (z. B. Rekursion, Sortier- und Suchverfahren, Mustererkennung, Heuristiken)
- Funktionale Modellierung
  insbesondere: Beschreibung funktionaler Zusammenhänge, Kombination von Funktionen, funktionale Abstraktion
- Regelbasierte Modellierung
  insbesondere: Fakten und Regeln, Klauseln, Anfragen.

Interaktion mit und von Informatiksystemen

- Repräsentation von Information
- Gestalten von Benutzungsoberflächen, Aspekte von Benutzungsfreundlichkeit
- Sprache als Werkzeug der Kommunikation: Aspekte formaler Sprachen, Syntax und Semantik
- Kommunikation zwischen Computern, Netze (z. B. einfaches Kommunikationsprotokoll, einfaches Schichtenmodell)
- Datenschutz und Datensicherheit (z. B. Kryptologie, Zugriffskontrolle)
- Anwendung verschiedener Werkzeuge zur Umsetzung von Modellen (z. B. Datenbankmanagementsystem, Programmierumgebung, Simulationssoftware)

Möglichkeiten und Grenzen informatischer Verfahren

- Grundsätzliche Funktionsweisen von Computersystemen (z. B. von-Neumann-Rechnermodell)
- Beurteilung von Verfahren hinsichtlich Effizienz und Bedeutung aufgrund der Einsatzmöglichkeiten
- prinzipielle und praktische Grenzen der Berechenbarkeit
- gesellschaftliche, ethische und rechtliche Aspekte (z. B. Auswirkungen des Computereinsatzes in der Arbeitswelt und im Freizeitbereich, gesetzliche Rahmenbedingungen)

1.3 Differenzierung zwischen Grund- und Leistungskursfach

1.3.1 Allgemeine Charakterisierung

Die Vereinbarung zur Gestaltung der gymnasialen Oberstufe in der Sekundarstufe II (Beschluss der Kultusministerkonferenz vom 07.07.1972 i. d. F. vom 16.06.2000) weist Grund- und Leistungskursen unterschiedlich akzentuierte Aufgaben zu: den Grundkursen die Vermittlung einer wissenschaftspropädeutisch orientierten Grundbildung, den Leistungskursen die systematische, vertiefte und reflektierte wissenschaftspropädeutische Arbeit.

Sowohl im Grund- als auch im Leistungskursfach steht das problemorientierte Arbeiten im Vordergrund.

Grundkurse führen in grundlegende Sachverhalte, Probleme, Zusammenhänge, Strukturen und Fragestellungen des Faches ein. In ihnen werden wesentliche Arbeitsmethoden und grundlegenden Zusammenhänge erarbeitet. Schülerinnen und Schüler können wesentliche informatische Arbeitsmethoden nutzen und fachübergreifende bzw. fächerverbindende Zusammenhänge exemplarisch erkennen.

Leistungskurse befassen sich methodisch ausgewiesener und systematischer als die Grundkurse mit wesentlichen, die Breite, die Komplexität und den Aspektreichtum des Faches Informatik verdeutlichenden Inhalten, Theorien und Modellen. Sie orientieren sich stärker an der Systematik der Fachwissenschaft, sind auf sichere und selbstständige Anwendung informatischer Methoden und ihre Übertragung und theoretische Reflektion gerichtet. Die Schülerinnen und Schüler lernen im Leistungskursfach fachübergreifende bzw. fächerverbindende Zusammenhänge zu erkennen.

Die Anforderungen im Grundkursfach sollen sich nicht nur quantitativ sondern vor allem auch qualitativ von denen im Leistungskursfach unterscheiden.

In den Abituraufgaben unterscheiden sich Grundkurs- und Leistungskursfach insbesondere durch

- den Grad der Vorstrukturierung bei der Problembearbeitung,
- die Offenheit der Aufgabenstellung,
- die Anforderungen an Selbstständigkeit bei der Bearbeitung der Aufgaben,
- den Umfang und die Art der bereitgestellten Hilfsmitteln und Informationen,
- den Grad der Abstraktion der zu behandelnden Inhalte und Begriffe,
- den Grad der Formalisierung von Sachverhalten und Darstellungen,
- den Grad der Komplexität der Problemstellungen,
- die Vielfältigkeit der verwendeten Methoden und
- die Vielfalt an Untersuchungs- und Lösungsstrategien.

1.3.2 Aufgabenbeispiele für die Differenzierung

Beispiel 1 ADT Dictionary

Fassung für das Grundkursfach
Die Variante für das Leistungskursfach steht bei den Aufgabenbeispielen unter 1.1.7.

Zielsetzung:
Die Aufgabe fordert die Modellierung des abstrakten Datentyps Dictionary. Die teilweise Implementierung des abstrakten Datentyps auf der Basis der Klasse Vector erfolgen. Es werden analytische und konstruktive Anforderungen gestellt sowie Kenntnis und Anwendung vorgegebenen Klassen durch Ableitung gefordert.

Unterrichtliche Voraussetzungen:
Im Unterricht wurden die abstrakten Datentypen lineare Liste, Keller und Vector behandelt. Am Beispiel der linearen Liste und Keller haben die Schüler die Vererbung und Klassenhierarchie kennen gelernt und angewendet. Mit dynamischen Datenstrukturen, Standardalgorithmen und dem Umgang mit Operationen haben die Schüler mehrfach im Unterricht Erfahrung gesammelt.

Aufgabe:
In verschiedenen Programmiersprachen gibt es den abstrakten Datentyp (ADT) *Dictionary*, bei dem man Zeichenketten als Index benutzen kann. Da ein Index normalerweise eine natürliche Zahl ist, spricht man beim Dictionary vom *Schlüssel* statt vom Index. Als Beispiel betrachten wir ein Dictionary *vokabeln*. Mit `vokabeln["lesson"] = "Unterrichtsstunde"` könnte unter dem Schlüssel `"lesson"` der Wert

`Unterrichtsstunde` eingetragen werden und mit der Prüfung `eingabe.equals(vokabeln["teacher"])` könnte geprüft werden, ob `eingabe` die richtige Übersetzung von `teacher` ist. Eine Liste mit Telefonnummer könnte auch ganz einfach mit einem Dictionary verwaltet werden, wobei die Namen die Schlüssel und die Telefonnummern die Werte wären. In einem Dictionary werden also Paare aus Schlüssel und Wert gespeichert. Um auch in Java diesen praktischen Datentyp nutzen zu können, entwickeln wir den ADT *Dictionary*.

a) Nennen Sie vier Operationen für den ADT Dictionary und geben Sie jeweils an, was die Operation bewirkt, welche Daten zur Ausführung der Operation benötigt und welche geliefert werden.

b) Der ADT Dictionary soll Elemente aufnehmen, die aus den zwei Attributen *Schlüssel* und *Wert* mit dem Datentyp String bestehen. Schreiben Sie eine Java-Klasse *DictionaryElement*, die einen Konstruktor zur Erzeugung eines Dictionary-Elements enthält.

c) Zur Implementierung des ADT Dictionary leiten wir *Dictionary* von der Java-Klasse *Vector* ab, deren Klassenbeschreibung als Anlage beigefügt ist. Entwickeln Sie eine Klassenbeschreibung für *Dictionary* unter Beachtung des Vererbungskonzepts und der unter a) angegebenen Operationen. Die Operationen müssen hierbei nicht ausprogrammiert werden, es reichen die Methodenköpfe.

d) Beschreiben Sie die Funktionsweise folgender Dictionary-Methode und erläutern Sie deren Nutzen.

```
01 private int gibIndex(String schluessel) {
02   DictionaryElement element;
03   int i = 0;
04   while ( i < size() ) {
05     element = (DictionaryElement) get(i);
06     if ( element.schluessel.equals(schluessel) )
07       return i;
08     i++;
09    }
10  return -1;
11 }
```

e) Implementieren Sie einen Konstruktor für die Klasse Dictionary. Das Dictionary soll alle Schlüssel-Wert-Paare zu einem als Parameter angegebenen Wert enthalten.

Kommentar
Im Unterschied zur Leistungskursfassung sind die Teile c) bis e) in Bezug auf Grad der Vorstrukturierung und Offenheit der Problemstellung, der Anforderungen an die Selbstständigkeit bei der Bearbeitung und den Grad der Komplexität der Aufgabe anders gefasst. Durch das Weglassen der binären Suchbäume wird den Lernvoraussetzungen des Grundkursfaches Rechnung getragen und der Anforderungsbereich III in der Grundkursfassung im Teil e) erreicht.

Beispiel 2 Kryptographie

Fassung für das Leistungskursfach
Die Variante für das Grundkursfach steht bei den Aufgabenbeispielen unter 1.2.2.

Unterrichtliche Voraussetzungen:
Im Unterricht wurden symmetrische und asymmetrische Verschlüsselungsverfahren behandelt. Bezüglich des RSA-Verfahrens sind die Algorithmen zur Schlüsselerzeugung, zum Ver- und Entschlüsseln von Nachrichten und zur Authentifizierung bekannt, geübt und implementiert worden. Die Prüflinge sind mit der Langzahlarithmetik und dem Rechnen modulo n vertraut, hingegen ist der Fermat-Test nicht bekannt. Sie kennen Verfahren zur Kryptoanalyse und die Komplexität des Faktorisierungsproblems. Aktuelle Einsatzbereiche von Kryptographie sind im Unterricht behandelt und diskutiert worden.

Aufgabe:

Lesen Sie den Auszug aus der Pressemitteilung des Landesbeauftragten für den Datenschutz Schleswig-Holstein (siehe Anlage 1.2.2).

a) Beschreiben Sie die Anwendung der asymmetrischen Verschlüsselung am Beispiel der E-Mail-Kommunikation mit dem Datenschutzbeauftragten.

b) Beschreiben Sie das RSA-Verfahren und erläutern Sie an Hand eines Beispiels die Schlüsselerzeugung sowie die Ver- und Entschlüsselung.

c) Zur Gewinnung sehr großer Zahlen, die mit hoher Wahrscheinlichkeit Primzahlen sind, erzeugt man zunächst Zufallszahlen und testet diese auf Primzahleigenschaft. Für den Test kann der kleine Satz von Fermat genutzt werden:

Ist p eine Primzahl, so gilt für jede natürliche Zahl b < p: $b^{p-1} \bmod p = 1 \bmod p$

p ist somit sicher zerlegbar, wenn es eine Zahl b < p gibt, so dass $b^{p-1} \bmod p \neq 1 \bmod p$. Je mehr Zahlen b < p gefunden werden können, welche die obige Gleichung erfüllen, umso größer ist die Wahrscheinlichkeit, dass p prim ist.

Entwickeln Sie einen Primzahltest auf der Basis des kleinen Satzes von Fermat.

c1) Geben Sie zunächst einen Algorithmus an, der eine große Zufallszahl mit Hilfe Ihres Tests auf Primzahleigenschaft testet.

c2) Implementieren Sie diesen Algorithmus.

d) Vergleichen Sie symmetrische und asymmetrische Verschlüsselungsverfahren und deren Einsatzbereiche. Bewerten Sie beide hinsichtlich Anwendbarkeit und Sicherheit.

e) Diskutieren Sie eine gesetzliche Beschränkung von Kryptographie und beziehen Sie begründet Stellung.

Kommentar
Diese Aufgabe lässt den Unterschied zwischen Grund- und Leistungskurs im Sinne der Merkmale von 1.3.1 an den folgenden Punkten deutlich werden: Grad der Vorstrukturierung, Offenheit der Problemstellung, Anforderungen an die Selbstständigkeit bei der Bearbeitung, Grad der Abstraktion und Komplexität der Aufgabe.

Die Lernvoraussetzungen dieser Leistungskursaufgabe und der Grundkursaufgabe in 1.2.2 stimmen in weiten Teilen überein. Eine Abstufung besteht hinsichtlich der Tiefe der Behandlung des RSA-Verfahrens.

2 Anforderungsbereiche

2.1 Allgemeine Hinweise

Die Abiturprüfung soll das Leistungsvermögen der Prüflinge möglichst differenziert erfassen. Die Aufgaben der Abiturprüfungen sollen Qualifikationen in möglichst großer Breite überprüfen. Dazu werden im Folgenden drei Anforderungsbereiche unterschieden.

Obwohl sich weder die Anforderungsbereiche scharf gegeneinander abgrenzen noch die zur Lösung einer Prüfungsaufgabe erforderlichen Teilleistungen in jedem Einzelfall eindeutig einem bestimmten Anforderungsbereich zuordnen lassen, kann die Berücksichtigung der Anforderungsbereiche wesentlich dazu beitragen, Einseitigkeiten zu vermeiden und die Durchschaubarkeit und Vergleichbarkeit der Prüfungsaufgaben sowie der Bewertung der Prüfungsleistungen zu erhöhen.

Beim Entwurf einer Prüfungsaufgabe wird jede von den Prüflingen erwartete Teilleistung mindestens einem der drei Anforderungsbereiche zugeordnet. Offenere Fragestellungen führen in der Regel über formales Anwenden von Begriffen und Verfahren hinaus und damit zu einer Zuordnung zu den Anforderungsbereichen II oder III. Die tatsächliche Zuordnung der Teilleistungen hängt davon ab, ob die jeweils aufgeworfene Problematik eine

selbstständige Auswahl unter Bearbeitungsansätzen in einem durch Übung bekannten Zusammenhang erfordert oder ob kreatives Erarbeiten, Anwenden und Bewerten in komplexeren und neuartigen Zusammenhängen erwartet wird.

In jedem Fall ist die Zuordnung zu den Anforderungsbereichen abhängig von im Lehrplan verbindlich vorgeschriebenen Zielen und Inhalten bzw. vom vorangegangenen Unterricht sowie von der Leistungsfähigkeit zugelassener Hilfsmittel (z. B. Hand- und Fachbuch, spezifiziertes Informatiksystem).

2.2 Fachspezifische Beschreibung der Anforderungsbereiche

2.2.1 Anforderungsbereich I

Der Anforderungsbereich I umfasst

- die Wiedergabe von bekannten Sachverhalten aus einem abgegrenzten Gebiet im gelernten Zusammenhang,
- die Beschreibung und Darstellung bekannter Verfahren, Methoden und Prinzipien der Informatik,
- die Beschreibung und Verwendung gelernter und geübter Arbeitstechniken und Verfahrensweisen in einem begrenzten Gebiet und in einem wiederholenden Zusammenhang.

Dazu kann u. a. gehören:

- Wiedergeben von Begriffsdefinitionen, Regeln, Zusammenhängen, bekannten Verfahren, einfachen Algorithmen, einfachen Modellierungen und Strukturen in einer im Unterricht behandelten Darstellungsform
- Wiedergeben eines bekannten Modells in geübter Darstellung
- Beschreiben der Funktionsweise und des Aufbaus bekannter Informatiksysteme
- Identifizieren von Objekten und ihren Beziehungen in einem bekannten Sachzusammenhang
- Beschreiben und Darstellen bekannter Automaten und Prozesse
- Beschreiben von Daten- und Kontrollstrukturen
- Beschreiben von Anwendungsmöglichkeiten der Informations- und Kommunikationstechniken und deren Wechselwirkungen mit Individuen und Gesellschaft
- Beschreiben grundlegender Anliegen des Datenschutzes und des Urheberrechts
- Verwenden einfacher vorgegebener grafischer Modellierungen
- Verwenden einfacher Modellierungen und bekannter einfacher Algorithmen
- Übersetzen in eine andere Darstellungsform in einem wiederholenden Zusammenhang
- Einfaches Erweitern einer vorgegebenen Problemlösung in geübtem Zusammenhang

2.2.2 Anforderungsbereich II

Der Anforderungsbereich II umfasst

- die selbstständige Verwendung (Auswählen, Anordnen, Verarbeiten und Darstellen) bekannter Sachverhalte zur Bearbeitung neuer Frage- oder Problemstellungen unter vorgegebenen Gesichtspunkten in einem durch Übung bekannten Zusammenhang,
- die selbstständige Übertragung des Gelernten auf vergleichbare neue Situationen, wobei es entweder um veränderte Fragestellungen oder um veränderte Sachzusammenhänge oder um abgewandelte Verfahrensweisen gehen kann,

- die Anwendung bekannter Verfahren, Methoden und Prinzipien der Informatik zur Lösung eines neuen Problems aus einem bekannten Problemkreis.

Dazu kann u. a. gehören:

- Verwenden bekannter Fakten, Definitionen, Begriffe, Regeln, Begründungen und Schlussfolgerungen bei der Bewältigung neuer Fragestellungen aus im Unterricht behandelten Gebieten
- planvolles Einsetzen bekannter Informatiksysteme zur Lösung einer neuen Problemstellung aus einem bekannten Bereich
- Überprüfen der Eignung eines bekannten informatischen Modells für die Lösung einer neuen Problemstellung
- Erstellen eines Modells zu einem Problem mit bekannten Verfahren
- Durchführen einer objektorientierten Analyse und Entwickeln eines objektorientierten Designs für eine vergleichbare neue Problemstellung
- Erstellen eines ER-Diagramms für eine vergleichbare neue Problemstellung
- Umsetzen eines ER-Diagramms in ein Relationenmodell
- Nutzen vorhandener Programmbibliotheken für die eigene Problemlösung
- Implementieren von Prozeduren, Funktionen und Methoden im vorgegebenen Kontext
- Analysieren eines gegebenen Algorithmus
- Begründen von bestimmten Eigenschaften (z. B. Terminierung, Zeit- und Speicheraufwand) eines gegebenen Algorithmus durch nicht formale Überlegungen
- Übertragen von Aufwandsbetrachtungen auf einen vergleichbaren aber nicht bekannten Algorithmus
- Dokumentieren einer Problemlösung mit angemessenen Darstellungsmitteln
- Entwerfen einer Datenstruktur, Ersetzen einer gegebenen Datenstruktur durch eine geeignete andere
- Entwickeln eines einfachen Automaten
- Analysieren eines Fallbeispiels (z. B. Datenschutz, Auswirkungen der neuen Informations- und Kommunikationstechniken)

2.2.3 Anforderungsbereich III

Der Anforderungsbereich III umfasst

- das planmäßige Verarbeitung komplexer Gegebenheiten mit dem Ziel, zu selbstständigen Gestaltungen bzw. Deutungen, Folgerungen, Begründungen, Wertungen zu gelangen,
- die bewusste und selbstständige Auswahl und Anpassung geeigneter gelernter Methoden und Verfahren in neuartigen Situationen. Dabei werden aus gelernten Denkmethoden bzw. Lösungsverfahren die zur Bewältigung der Aufgabe geeigneten selbstständig ausgewählt und einer neuen Problemstellung angepasst.

Dazu kann u. a. gehören:

- Durchführen einer komplexen Problemanalyse
- Zerlegen eines gegebenen anspruchsvollen Problems in geeignete Teilprobleme
- Entwerfen und Beurteilen von Schnittstellen
- Entwickeln eines Verfahrens bzw. Algorithmus zur Lösung eines neuen Problems

- Formulieren einer begründeten Stellungnahme zu einem authentischen Text in Bezug auf Möglichkeiten, Angemessenheit und Grenzen des Einsatzes von Informatiksystemen
- Beurteilen der eigenen Modellierung und Problemlösung im Anwendungskontext
- Entwickeln einer Sprache (z. B. Angabe der Syntax und Semantik einer einfachen Steuersprache für einen Roboter)

3 Schriftliche Prüfung

3.1 Allgemeine Hinweise

Eine Prüfungsaufgabe für die schriftliche Abiturprüfung im Fach Informatik besteht aus zwei oder drei Aufgaben. Die Prüfungsaufgabe muss sich auf verschiedene in Abschnitt 1.2 genannten Bereiche mit ihren Vernetzungen und in jedem Fall auf den Bereich Grundlegende Modellierungstechniken beziehen. Sie darf sich nicht auf die Inhalte nur eines Kurshalbjahres beschränken, siehe Vereinbarung über die Abiturprüfung der gymnasialen Oberstufe in der Sekundarstufe II (Beschluss der Kultusministerkonferenz vom 13.12.1973 i. d. F. vom 16.06.2000, § 5 Abs. 4). Sofern in der Prüfungsaufgabe andere als die unter 1.2 genannten Bereiche berücksichtigt werden, dürfen sich die Anforderungen höchstens zu einem Drittel auf diese anderen Bereiche beziehen. Das zugehörige Anforderungsniveau muss dem der anderen Aufgaben entsprechen.

Jede Aufgabe kann in Teilaufgaben gegliedert sein, die jedoch nicht beziehungslos nebeneinander stehen sollen. Durch die Gliederung in Teilaufgaben können

- verschiedene Blickrichtungen eröffnet,
- mögliche Vernetzungen gefördert und
- unterschiedliche Anforderungsbereiche gezielt angesprochen werden.

Durch Modifikation von Teilaufgaben können Leistungskursaufgaben den Anforderungen des Grundkursfaches angepasst werden. Die Teilaufgaben einer Aufgabe sollen so unabhängig voneinander sein, dass eine Fehlleistung – insbesondere am Anfang – nicht die weitere Bearbeitung der Aufgabe unmöglich macht. Falls erforderlich, können Zwischenergebnisse in der Aufgabenstellung enthalten sein. Die Aufgliederung darf nicht so detailliert sein, dass dadurch ein Lösungsweg zwingend vorgezeichnet wird.

3.2 Aufgabenarten

Folgende Arten von Aufgaben oder Teilaufgaben können u. a. vorkommen, wobei teilweise Überschneidungen möglich sind:

- Modellierung einer konkreten Problemstellung
- Implementierung einer konkreten bereits modellierten Problemstellung
- Darstellung, Erläuterung und sachgerechte Anwendung von informatischen Begriffen und Verfahren
- Untersuchung und Beschreibung vorgegebener informatischer Konstrukte
- Visualisierung von Sachverhalten und informatischen Zusammenhängen
- Interpretation, Vergleich und Bewertung von Daten, Ergebnissen, Lösungswegen oder Verfahren
- Übertragung von Ergebnissen auf einen anderen Sachverhalt

Unterscheidungsmerkmale für die Aufgabenstellung in Grund- und Leistungskursfach sind unter 1.3 benannt.

3.3 Hinweise zum Erstellen einer Prüfungsaufgabe

Die Prüfungsaufgabe für die schriftliche Abiturprüfung soll sowohl fachliche und methodische Kompetenzen als auch Kenntnisse fachlicher Inhalte in möglichst großer Breite überprüfen. Eine Prüfungsaufgabe muss sich auf alle drei in Abschnitt 2.2 beschriebenen Anforderungsbereiche erstrecken, so dass eine Beurteilung ermöglicht wird, die das gesamte Notenspektrum umfasst. Die Prüfungsaufgabe sowohl für das Grundkursfach als auch für das Leistungskursfach erreicht dann ein angemessenes Niveau, wenn das Schwergewicht der zu erbringenden Prüfungsleistungen im Anforderungsbereich II liegt und daneben die Anforderungsbereiche I und III berücksichtigt werden, und zwar Anforderungsbereich I in höherem Maße als Anforderungsbereich III.

Entsprechende Anteile der Anforderungsbereiche können insbesondere durch geeignete Wahl der nachzuweisenden fachlichen und methodischen Kompetenzen, durch die Struktur der Prüfungsaufgabe sowie durch entsprechende Formulierung des Textes erreicht werden (vgl. 2.1). Diese Wahl sollte so erfolgen, dass eine prüfungsdidaktisch sinnvolle, selbstständige Leistung gefordert wird, ohne dass der Zusammenhang zur bisherigen Unterrichts- und Klausurpraxis verloren geht.

Das Erstellen einer Prüfungsaufgabe einschließlich des Abschätzens ihrer Angemessenheit lässt sich in folgender Weise vornehmen:

- Nach Auswahl der Problemfelder und der darin möglichen Fragestellungen werden die Aufgaben bzw. Teilaufgaben unter Berücksichtigung der in 3.1 beschriebenen Bedingungen formuliert.
- Zu jeder Teilaufgabe werden erwartete Lösungsschritte beschrieben (siehe 3.4 und Teil II, 1).
- Aufgrund des vorangegangenen im Rahmen der geltenden Bestimmungen erteilten Unterrichts werden die erwarteten Lösungsschritte nach pädagogischem Ermessen den Anforderungsbereichen I bis III zugeordnet.
- Zum Abschätzen des Anteils der einzelnen Anforderungsbereiche ist zu beachten, dass die erwarteten Lösungsschritte jeweils Teilleistungen darstellen, die im Rahmen der gesamten Prüfungsaufgabe von unterschiedlicher Bedeutung sein können. Deshalb kann es hilfreich sein, den Anteil dieser einzelnen zu erbringenden Teilleistungen an der erwarteten Gesamtleistung zu kennzeichnen. Diese Kennzeichnung berücksichtigt vorwiegend die zur Lösung erforderlichen gedanklichen Einzelschritte und die für die Bearbeitung und Darstellung geschätzte Zeit; sie beruht vornehmlich auf der pädagogischen Erfahrung.

3.4 Beschreibung der erwarteten Prüfungsleistungen (Erwartungshorizont)

„Den Aufgaben der schriftlichen Prüfung werden von der Aufgabenstellerin bzw. dem Aufgabensteller eine Beschreibung der von den Schülerinnen und Schülern erwarteten Leistungen einschließlich der Angabe von Bewertungskriterien beigegeben. Dabei sind von der Schulaufsichtsbehörde gegebene Hinweise für die Bewertung zu beachten und auf die gestellten Aufgaben anzuwenden.“, siehe § 5 Absatz 3 der „Vereinbarung über die Abiturprüfung der gymnasialen Oberstufe in der Sekundarstufe II“ (Beschluss der Kultusministerkonferenz vom 13.12.1973 i. d. F. vom 16.06.2000). Die erwarteten Prüfungsleistungen sind darzustellen. Werden Prüfungsaufgaben nicht zentral gestellt, so ist der vorangegangene Unterricht, aus dem die vorgeschlagene Prüfungsaufgabe erwachsen ist, so weit kurz zu erläutern, wie dies zum Verständnis der Aufgabe notwendig ist. Damit soll zugleich der Bezug zu den Anforderungsbereichen einsichtig gemacht werden. Zugelassene Hilfsmittel sind anzugeben. Beim Einsatz der Hilfsmittel muss der Grundsatz der Gleichbehandlung gewahrt bleiben.

3.5 Bewertung von Prüfungsleistungen

Nach § 6 Absatz 5 der „Vereinbarung über die Abiturprüfung der gymnasialen Oberstufe in der Sekundarstufe II“ (Beschluss der Kultusministerkonferenz vom 13.12.1973 i. d. F. vom 16.06.2000) soll aus der Korrektur und Beurteilung der schriftlichen Arbeit (Gutachten) hervorgehen, „welcher Wert den von der Schülerin bzw. dem Schüler vorgebrachten Lösungen, Untersuchungsergebnissen oder Argumenten beigemessen wird und wieweit die Schülerin bzw. der Schüler die Lösung der gestellten Aufgaben durch gelungene Beiträge gefördert oder durch sachliche oder logische Fehler beeinträchtigt hat. Die zusammenfassende Beurteilung schließt mit einer Bewertung gemäß Ziffer 9.1 und 9.2 der Vereinbarung vom 07.07.1972 i. d. F. vom 16.06.2000.“ Das Beurteilen

der von den Prüflingen erbrachten Prüfungsleistung erfolgt unter Bezug auf die beschriebene erwartete Gesamtleistung. Den Beurteilenden steht dabei ein Beurteilungsspielraum zur Verfügung. Liefern Prüflinge zu einer gestellten Aufgabe oder Teilaufgabe Lösungen, die in der Beschreibung der erwarteten Prüfungsleistungen nicht erfasst waren, so sind die erbrachten Leistungen angemessen zu berücksichtigen. Dabei kann der vorgesehene Bewertungsrahmen für die Teilaufgabe nicht überschritten werden.

Für die Bewertung der Prüfungsleistungen sind sowohl die rein formale Lösung als auch das zum Ausdruck gebrachte informatische Verständnis maßgebend. Daher sind erläuternde, kommentierende und begründende Texte unverzichtbare Bestandteile der Prüfungsleistung. Mangelhafte Gliederung, Fehler in der Fachsprache, Ungenauigkeiten in Darstellungen oder unzureichende oder falsche Bezüge zwischen Darstellungen und Text sind als fachliche Fehler zu werten. Darüber hinaus sind schwerwiegende und gehäufte Verstöße gegen die sprachliche Richtigkeit in der Muttersprache (Unterrichtssprache) oder gegen die äußere Form gemäß § 6 Abs. 5 der „Vereinbarung über die Abiturprüfung der gymnasialen Oberstufe in der Sekundarstufe II“ (Beschluss der Kultusministerkonferenz vom 13. 12. 1973 i. d. F. vom 16.06.2000) zu bewerten. Da jede Prüfungsaufgabe in mehrere voneinander unabhängige Aufgaben gegliedert ist, ist es notwendig, für diese Teile den jeweiligen Anteil an der erwarteten Gesamtleistung anzugeben.

Die Festlegung der Schwelle zur Note „ausreichend“ (05 Punkte) und die Vergabe der weiteren Noten sind Setzungen, die in besonderem Maße der pädagogischen Erfahrung und Verantwortung der Beurteilenden unterliegen. Die Note „ausreichend“ (05 Punkte) soll erteilt werden, wenn annähernd die Hälfte (mindestens 45 Prozent) der erwarteten Gesamtleistung erbracht worden ist. Dazu reichen Leistungen allein im Anforderungsbereich I nicht aus. Oberhalb und unterhalb dieser Schwelle sollen die Anteile der erwarteten Gesamtleistung den einzelnen Notenstufen jeweils ungefähr linear zugeordnet werden, um zu sichern, dass mit der Bewertung die gesamte Breite der Skala ausgeschöpft werden kann. Die Note „gut“ (11 Punkte) soll erteilt werden, wenn annähernd vier Fünftel (mindestens 75 Prozent) der erwarteten Gesamtleistung erbracht worden ist.

4 Mündliche Prüfung

4.1 Aufgabenstellung

Die Aufgabenstellung in der mündlichen Prüfung bezieht sich schwerpunktmäßig auf die in Abschnitt 1.2 genannten Bereiche. Dabei sollen die Prüflinge zeigen, dass sie über informatische Sachverhalte in freiem Vortrag berichten und im Gespräch zu informatischen Fragen Stellung nehmen sowie fachlich argumentieren können. Sie sollen insbesondere nachweisen, in welchem Umfang sie

- einen Überblick über wesentliche Begriffe und Verfahren der Informatik besitzen,
- Verständnis für informatische Denk- und Arbeitsweisen haben,
- einen Einblick in informatische Problemstellungen, Ergebnisse und Möglichkeiten besitzen.

Die Aufgabenstellung für die mündliche Prüfung unterscheidet sich grundsätzlich von der für die schriftliche Prüfung. Stärker berücksichtigt wird die Darstellung und Begründung von Sachverhalten und Verfahren. In der Prüfung ist der Nachweis verschiedener fachlicher und methodischer Kompetenzen zu fordern. Umfangreiche Detaildarstellungen sind zu vermeiden.

Besonders geeignet sind Fragestellungen, die

- Teilaufgaben enthalten, die eine Erläuterung der Grundgedanken der Modellierung in den Mittelpunkt stellen,
- analytische Elemente der Lösungsfindung enthalten, Diagramme, Ergebnisse, Resultate usw. vorgeben, an denen wesentliche Gedankengänge zu erläutern sind,
- Aussagen enthalten, zu denen der Prüfling bewertend Stellung nehmen kann.

Die Art und Anzahl der Teilaufgaben einer Aufgabe sollte so gestaltet sein, dass der Prüfling die Chance hat, den Umfang seiner Fähigkeiten und die Tiefe seines informatischen Verständnisses darzustellen. Für den Prüfungsausschuss ermöglichen sie die differenzierte Beurteilung der Leistungsfähigkeit des Prüflings. Die Aufgabe muss so angelegt sein, dass in der Prüfung unter Beachtung der Anforderungsbereiche (vgl. 2), die auf der Grundlage eines Erwartungshorizontes zugeordnet werden, grundsätzlich jede Note erreichbar ist.

4.2 Kriterien für die Bewertung

Bei der Bewertung der mündlichen Prüfungsleistung sollen neben den in Abschnitt 1.1 beschriebenen fachlichen und methodischen Kompetenzen vor allem folgende Kriterien berücksichtigt werden:

- Umfang und Qualität der nachgewiesenen informatischen Kenntnisse und Fertigkeiten,
- sachgerechte Gliederung und folgerichtiger Aufbau der Darstellung, Beherrschung der Fachsprache, Verständlichkeit der Darlegungen, adäquater Einsatz der Präsentationsmittel und die Fähigkeit, das Wesentliche herauszustellen,
- Verständnis für informatische Probleme sowie die Fähigkeit, Zusammenhänge zu erkennen und darzustellen, informatische Sachverhalte zu beurteilen, auf Fragen und Einwände einzugehen und gegebene Hilfen aufzugreifen; speziell im Prüfungsgespräch: gekonntes Zuhören und Reagieren,
- Kreativität und Selbstständigkeit im Prüfungsverlauf.

4.3 Fünfte Prüfungskomponente

„Die Abiturprüfung umfasst mindestens 4, höchstens 5 Komponenten. Fünfte Komponente ist entweder eine schriftliche oder eine mündliche Prüfung in einem weiteren Fach oder eine besondere Lernleistung.“, siehe Vereinbarung zur Gestaltung der gymnasialen Oberstufe in der Sekundarstufe II (Beschluss der Kultusministerkonferenz vom 07.07.1972 i. d. F. vom 16.06.2000). Im Rahmen der fünften Prüfungskomponente können die Länder neue Prüfungsformen entwickeln. Für alle Formen der fünften Prüfungskomponente gelten die Abschnitte 1 bis 4.2 sinngemäß.

Im Folgenden werden für die fünfte Prüfungskomponente als „mündliche Prüfung in neuer Form“ für das Fach bzw. Referenzfach Informatik Festlegungen getroffen, die über die Bestimmungen der Abschnitte 1 bis 4.2 hinausgehen.

Die Themenstellung soll durch Reichhaltigkeit der informatischen oder fachübergreifenden Bezüge gekennzeichnet sein. Sie soll in hohem Maße Originalität und Kreativität bei der Bearbeitung ermöglichen.

Die fünfte Prüfungskomponente als „mündliche Prüfung in neuer Form“ zielt insbesondere auf die Einbeziehung größerer fachlicher Zusammenhänge und fachübergreifender Aspekte in die Abiturprüfung. Sie sollte deshalb vor allem gekennzeichnet sein durch einen längeren zeitlichen Vorlauf und einen besonderen Stellenwert der vorbereiteten Präsentation.

Hinzu kommt die Möglichkeit Gruppenprüfungen durchzuführen. Dabei ist durch Begrenzung der Gruppengröße, durch die Art der Aufgabenstellung und die Gestaltung des Prüfungsgesprächs dafür Sorge zu tragen, dass die individuelle Leistung eindeutig erkennbar und bewertbar ist. Für Gruppenprüfungen eignen sich im Fach Informatik insbesondere Prüfungsaufgaben, bei denen unterschiedliche Aspekte eines Problems behandelt und adäquate Modellierungstechniken benutzt werden.

Die Gewährung eines längeren zeitlichen Vorlaufs kann insbesondere nötig sein bei Prüfungsaufgaben mit komplexerer Fragestellung, aufwändigerer Erschließung (z. B. durch Literatur-, Internet-Recherche) oder bei Projektarbeit.

Die Präsentation geht aus von einer vorgelegten Dokumentation und wird bestimmt durch die verfügbaren technischen Möglichkeiten, z. B. Folien, Präsentationssoftware und Informatiksysteme.

Bei der Bewertung der fünften Prüfungskomponente als „mündliche Prüfung in neuer Form“ kommen neben der nachgewiesenen Fach- und Methodenkompetenz

- der Klarheit, Vollständigkeit und Angemessenheit von Dokumentation und Präsentation,
- der Selbstständigkeit und dem Einfallsreichtum bei der Ausführung der Arbeitsanteile und Arbeitsschritte,
- dem Grad der Durchdringung und den aufgezeigten Vernetzungen sowie
- der Souveränität im Prüfungsgespräch

besondere Bedeutung zu.

II Aufgabenbeispiele

Mit Rücksicht auf die unterschiedliche Praxis in den Ländern bilden die aufgeführten Beispiele für sich keine geschlossenen Prüfungsaufgaben; sie ergeben vielmehr erst durch Hinzufügen weiterer Aufgaben auch unterschiedlichen Umfangs eine vollständige Prüfungsaufgabe. Dabei muss sichergestellt werden, dass in der vollständigen Prüfungsaufgabe alle Bedingungen entsprechend den Festlegungen in Teil I, 3.1, 3.3 und 4.1, berücksichtigt werden.

Durch die ausgewählten Beispiele sollen weder besondere thematische Schwerpunkte gesetzt noch thematische Festlegungen getroffen werden. Vielmehr soll die Vielfalt der Möglichkeiten bei der Themenauswahl, bei der Aufgabenkonstruktion sowie bei den verwendeten Ausdrucks- und Schreibweisen verdeutlicht werden. Die Beispiele betonen neuere fachdidaktische Entwicklungen, ohne auf bewährte Aufgabenstellungen zu verzichten. Sie sind jedoch nicht repräsentativ hinsichtlich formaler und anwendungsbezogener Anteile der Prüfungsaufgabe.

Besondere Aspekte wie z. B.

- inhaltliche Reflexion und Interpretation von Begriffen und Verfahren,
- Art der Modellierung,
- Öffnung von Fragestellungen,
- Vernetzung von Sachgebieten und Themenbereichen und
- Verwendung von PCs

werden bei den einzelnen Aufgabenbeispielen unter Anmerkungen benannt.

1 Aufgabenbeispiele für die schriftliche Prüfung

Die Aufgabenbeispiele enthalten Angaben über die Zielsetzung der Aufgabe, die unterrichtlichen Voraussetzungen, die zugelassenen Hilfsmittel und über die vorgesehene Bearbeitungszeit. Die Beispiele in den Abschnitten 1.1 und 1.2 sind ausführlicher dargestellt. Sie enthalten zusätzlich die Lösungsskizzen, die Zuordnungen zu den Anforderungsbereichen und die vorgesehenen Bewertungseinheiten.

Die in den Anmerkungen beschriebenen unterrichtlichen Voraussetzungen dienen dazu, die Angemessenheit der jeweiligen Aufgabenstellung zu beurteilen. Bei allen Zeitangaben handelt es sich um Richtwerte.

1.1 Ausführlich kommentierte Beispiele für das Leistungskursfach

1.1.1 Fuhrparkverwaltung *LF*

Diese Aufgabe ist mit dem PC zu lösen.

Die Firma Spectral-Lacke erteilt den Auftrag, für die Verwaltung der Firmenfahrzeuge eine Datenbank mit den definierten Funktionalitäten zu entwerfen. Die Anforderungen an die zu entwerfende Softwarelösung sind in einem Schreiben der Firma (Anlage 1) festgehalten. Daraus ergeben sich folgende Funktionen, welche die Software realisieren soll:

- Der Benutzer soll für ein bestimmtes Datum ein Auto vorbestellen können (Zuweisung Benutzer – Auto).
- Die Autos sollen nach Kfz-Kennzeichen, Anschaffungsdatum und nach Autotyp geordnet werden können.
- Werkstattaufenthalte sollen protokolliert werden.
- Es soll möglich sein, ein Auto dauerhaft für eine bestimmte Person zu reservieren. Dieses Fahrzeug steht anderen Personen nicht mehr zur Verfügung.
- Bei den Betankungen soll auch erkennbar sein, welcher Mitarbeiter diese vornahm.
- Bei jeder Wartung wird u. a. der Kilometerstand des Fahrzeugs festgehalten.

Die EDV-Abteilung, die das Projekt betreut, hat einen ersten Grobentwurf des Entity-Relationship-Modells erstellt, der aber noch nicht vollständig ist. Er ist im Anhang 2 abgebildet. In diesem Entwurf wurden unter anderem Betankungen und Wartungen noch nicht berücksichtigt.

a) Ergänzen Sie das Entity-Relationship-Modell in der Anlage, so dass Betankungen und Wartungen erfasst werden können. Definieren Sie auch die Attribute und Schlüssel.

b) Ergänzen Sie die vorliegende Datenbank Auto mit den Tabellen und Beziehungen, die Sie benötigen, um Betankungen und Wartungen zu verwalten.

c) Entwerfen Sie ein Bildschirmfenster, mit dem Sie Autos erfassen können. Den Autotyp soll man auswählen können.

d) Es ist abzusehen, dass folgende Informationen von der Datenbank immer wieder gefordert werden. Formulieren Sie deshalb SQL-Anweisungen für die aufgeführten Problemstellungen:

   1. Die Firma Spectral-Lacke bietet ihren Mitarbeitern die Autos, die über zwei Jahre alt sind, zum Kauf an. Entwerfen Sie eine Abfrage, welche alle Autos auflistet, die von heute aus betrachtet, älter als zwei Jahre sind.
   2. Die Firmenleitung wünscht eine Liste aus der hervorgeht, welche Mitarbeiter welche Autotypen fahren. Aufzuführen sind: Kurzzeichen, Nachname, Typbezeichnung, Herstellername.
   3. Welche Autos aus dem Fuhrpark sind im Nominalverbrauch sparsamer als der durchschnittliche Nominalverbrauch des Autobestands der Firma?
   4. Die Firmenleitung will wissen, an wie vielen Tagen Mitarbeiter ein Auto reservierten. Aufzuführen sind alle Mitarbeiter, die an 20 oder mehr Tagen ein Auto reservierten. Anzugeben sind Kurzzeichen, Nachname und die Summe der Reservierungstage.

e) Die Tabelle Autotypen entspricht nicht den Normalisierungsregeln. Entwerfen Sie ein Relationenmodell, das diesen Mangel beseitigt.

Anlage 1:

Projekt Firmen-Fahrzeuge

Spectral Lacke

Die Firma Spectral-Lacke besitzt momentan ca. 35 Firmen-PKWs. Es soll ein zentrales Programm erstellt werden, das es ermöglicht, diese Fahrzeuge intern zu verwalten und bei Bedarf für Mitarbeiter zu reservieren.

Von der Thematik her ist es so, dass sich viele einzelne Abteilungen mit den Fahrzeugen beschäftigen.

Beispiel:	Personalbüro	Fahrzeugvergabe bzw. -belegung
	Kostenrechnung	Anschaffungswerte, Kraftstoffverbrauch,
	Einkauf	Versicherungsabschluss, Neuanschaffung
	Geschäftsführung	Allgemeine Informationen über Kosten und Anzahl der Fahrzeuge

Momentan gibt es keine einheitliche Verwaltung der Fahrzeuge, es existiert eine redundante Datenhaltung, da mehrere Abteilungen die gleichen Daten pflegen.

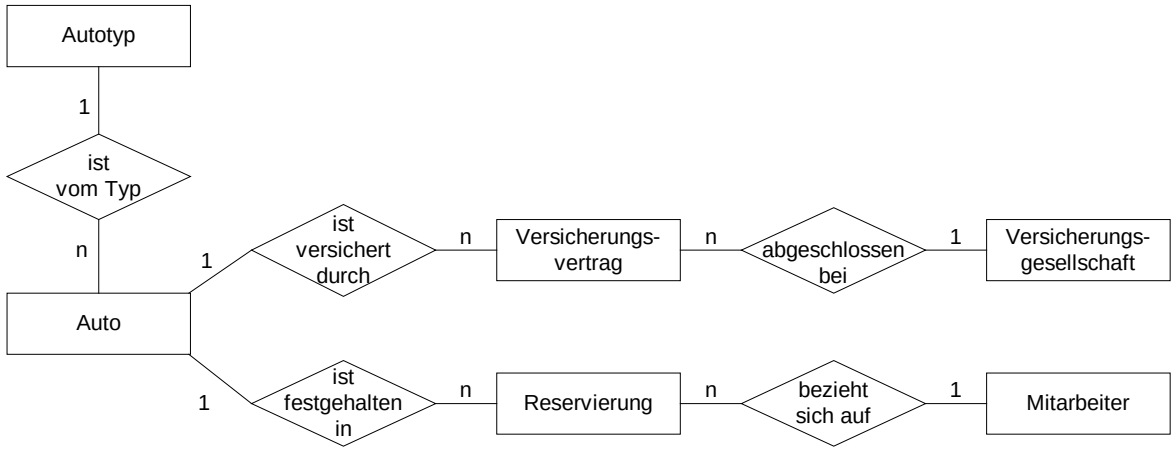
Die Stammdaten der Fahrzeuge können über eine Schnittstelle aus der Betriebsabrechnung (Kostenstelle) gezogen werden.

Folgende Punkte sollten in diesem Programm berücksichtigt werden:

- Anschaffungswerte
- Anschaffungsdatum
- Abschreibungswerte
- Möglichkeit zur Übernahme von Kraftstoffverbrauch aus Tankstelle/Tankrechnung (Schnittstelle)
- Kilometerstände (bei Wartung)

- Versicherung
- Wartung
- Belegungszeiten der Fahrzeuge

Anlage 2: vorläufiges ER-Modell



Anmerkungen

Zusätzliche Hilfsmittel:
Für die Lösung dieser Aufgabe steht jedem Schüler ein Computer mit einer der Aufgabe entsprechenden Softwareausstattung zur Verfügung. Die Datenbank Auto liegt in einem gängigen Format vor und kann somit in jedes relationales Datenbankmanagementsystem konvertiert werden. Außerdem stehen für die Erzeugung der Datenbank die im Folgenden angegebenen DDL-Befehle in SQL zur Verfügung. Die Ausführung dieser SQL-Befehle erzeugt die Datenbank in dem Managementsystem der jeweiligen Schule. Die Datenbank ist für jeden Schüler bei der Prüfung am Rechner verfügbar.

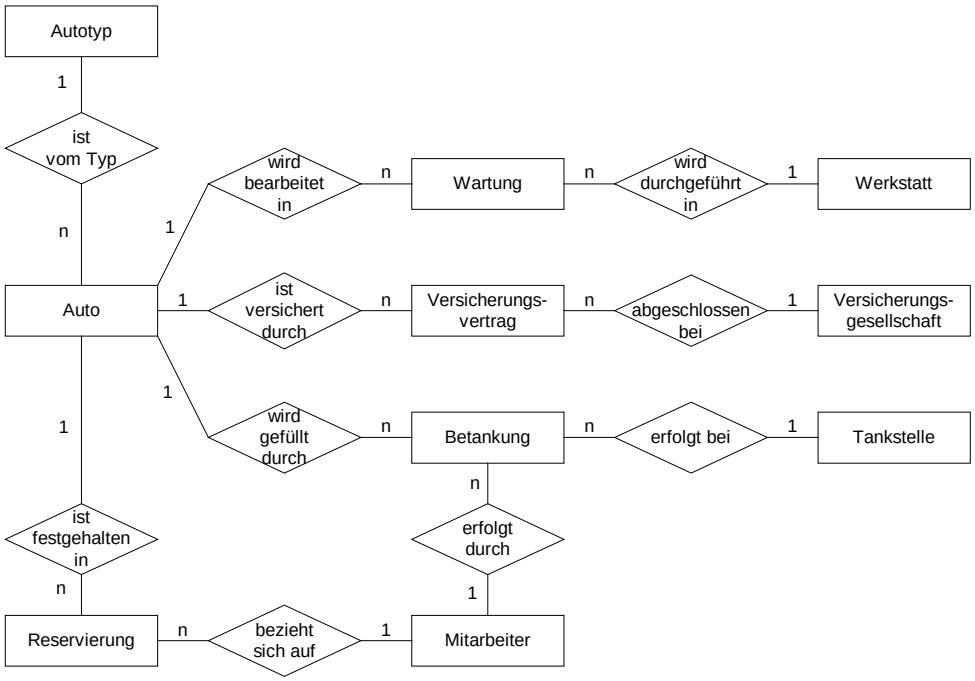
Die Ausgangsdatenbank Auto kann mit folgenden SQL-Befehlen erzeugt werden:

```
CREATE TABLE Versicherungen(Versicherungsnummer  INTEGER, Versicherungsname
CHAR(50), PRIMARY KEY(Versicherungsnummer));
CREATE TABLE Mitarbeiter(Nachname  CHAR(50),Abteilungsnummer  INTEGER, Vorname_
CHAR(50), Kurzzeichen  CHAR(50), PRIMARY KEY(Kurzzeichen));
CREATE TABLE Reservierungen(Von  DATETIME, Reservierungsnummer  INTEGER, Bis
DATETIME, feste_Reservierung  CHAR(50), Kurzzeichen  CHAR(50), Kennzeichen
CHAR(50), PRIMARY KEY(Reservierungsnummer));
CREATE TABLE Versicherungsvertraege(Praemie  DOUBLE, Vertragsnummer  INTEGER,
Versicherungsart  CHAR(50), Kennzeichen  CHAR(50), Versicherungsnummer  INTEGER,
PRIMARY KEY(Vertragsnummer));
CREATE TABLE Autotypen(Typnummer  INTEGER, Typbezeichnung  CHAR(50),
Herstellernummer  INTEGER, Herstellername  CHAR(50), Nominalverbrauch  DOUBLE,
PRIMARY KEY(Typnummer));
CREATE TABLE Autos(Anschaffungswert  DOUBLE, Anschaffungsdatum  DATETIME, Kilometer
DOUBLE, Kennzeichen  CHAR(50), Abschreibungswerte  DOUBLE, Typnummer  INTEGER,
PRIMARY KEY(Kennzeichen));

ALTER TABLE Reservierungen ADD CONSTRAINT REF5 FOREIGN KEY(Kurzzeichen) REFERENCES
Mitarbeiter
ALTER TABLE Reservierungen ADD CONSTRAINT REF6 FOREIGN KEY(Kennzeichen) REFERENCES
Autos
```

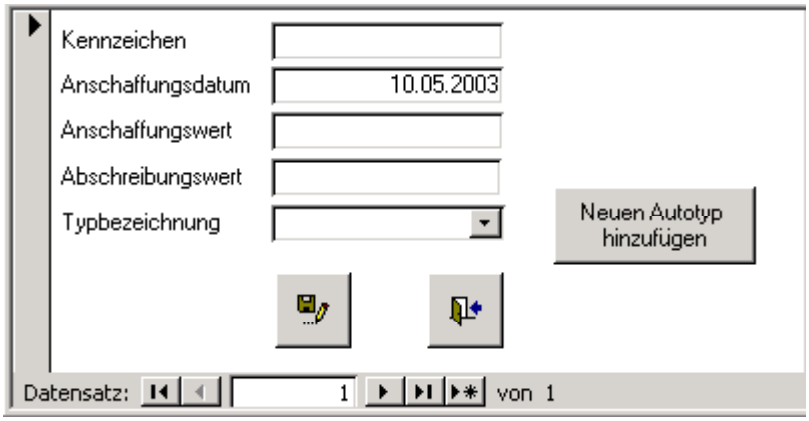
```
ALTER TABLE Versicherungsvertraege ADD CONSTRAINT REF7 FOREIGN KEY(Kennzeichen)
REFERENCES Autos
ALTER TABLE Versicherungsvertraege ADD CONSTRAINT REF8 FOREIGN
KEY(Versicherungsnummer) REFERENCES Versicherungen
ALTER TABLE Autos ADD CONSTRAINT REF10 FOREIGN KEY(Typnummer) REFERENCES Autotypen
```

Vorgesehene Bearbeitungszeit: 120 min

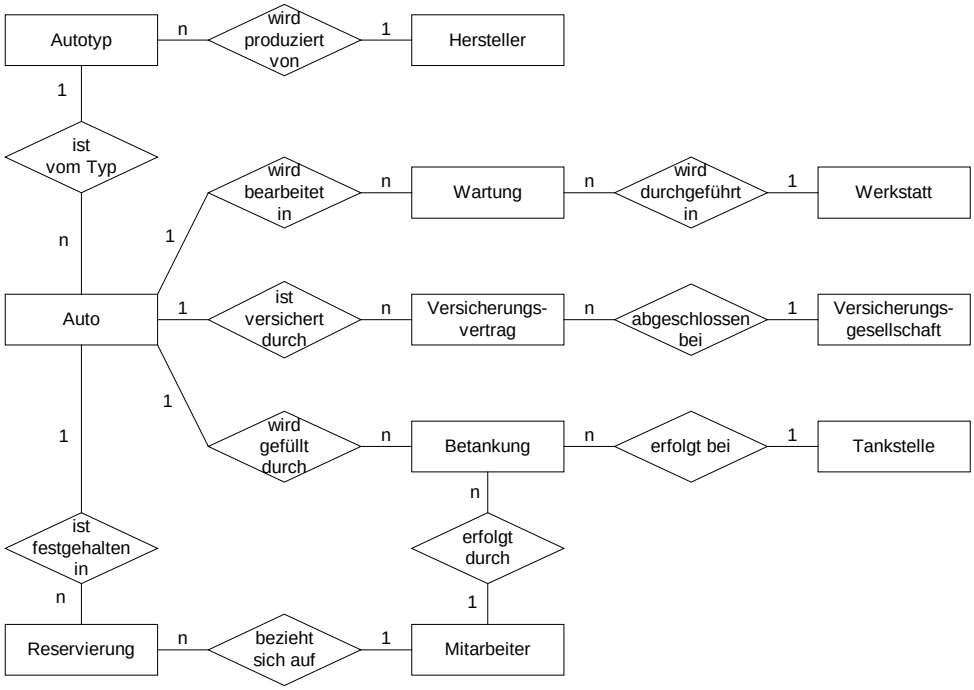
Lösungsskizze, vorgesehene Bewertungseinheiten und ihre Zuordnung zu den Anforderungsbereichen I, II, III:

<table>
<tr><th></th><th>Lösungsskizze</th><th>I</th><th>II</th><th>III</th></tr>
<tr><td>a)</td><td>


Relationen mit Attributen und Primärschlüsseln:
<code>Betankungen (Tankungsnummer, Tag, ↑Kennzeichen, Kraftstoffart,
Preis_der_Tankung, Fuellmenge, ↑Tankstellennummer, ↑Kurzzeichen)
Tankstellen (Tankstellennummer, Beschreibung)
Wartungen (Wartungsnummer, ↑Kennzeichen, von, bis, Wartungskosten,
↑Werkstattnummer, Kilometerstand)
Werkstätten (Werkstattnummer, Werkstattname)</code>
(Primärschlüssel sind unterstrichen, Fremdschlüsseln ist ein ↑ vorangestellt.)
</td><td></td><td>12</td><td>6</td></tr>
<tr><td>b)</td><td>Die Schüler sollen die folgenden Tabellen und Beziehungen, die im Folgenden durch SQL-Anweisungen ausgedrückt sind, der Datenbank hinzufügen. Jeder Lösungsweg ist dabei zulässig und auch vom eingesetzten RDBMS abhängig.
<code>CREATE TABLE Betankungen(Kraftstoffart CHAR(50), Preis_der_Tankung DOUBLE, Tankungsnummer DOUBLE, Fuellmenge DOUBLE, Datum DATETIME, Tankstellennummer DOUBLE, Kurzzeichen CHAR(50), Kennzeichen CHAR(50), PRIMARY KEY(Tankungsnummer));</code>
<code>CREATE TABLE Wartungen(Wartungskosten DOUBLE, Wartungsnummer DOUBLE, von DATETIME, bis DATETIME, Werkstattnummer DOUBLE, Kennzeichen CHAR(50), PRIMARY KEY(Wartungsnummer));</code>
<code>CREATE TABLE Tankstellen(Tankstellennummer DOUBLE, Beschreibung CHAR(50), PRIMARY KEY(Tankstellennummer));</code>
<code>CREATE TABLE Werkstaetten(Werkstattnummer DOUBLE, Werkstattname CHAR(50), PRIMARY KEY(Werkstattnummer));</code></td><td>6</td><td>2</td><td></td></tr>
</table>

	Lösungsskizze	I	II	III
	ALTER TABLE Betankungen ADD CONSTRAINT REF0 FOREIGN KEY(Tankstellennummer) REFERENCES Tankstellen **ALTER TABLE Betankungen** ADD CONSTRAINT REF1 FOREIGN KEY(Kurzzeichen) REFERENCES Mitarbeiters **ALTER TABLE Betankungen** ADD CONSTRAINT REF2 FOREIGN KEY(Kennzeichen) REFERENCES Autos **ALTER TABLE Wartungen** ADD CONSTRAINT REF3 FOREIGN KEY(Werkstattnummer) REFERENCES Werkstaette **ALTER TABLE Wartungen** ADD CONSTRAINT REF4 FOREIGN KEY(Kennzeichen) REFERENCES Autos			
c)		6	6	
d)	1. `SELECT Kennzeichen, Anschaffungsdatum, Anschaffungswert FROM Autos WHERE Anschaffungsdatum < (DATE() -2*365) ORDER BY Anschaffungsdatum;`	4	3	
	2. `SELECT Mitarbeiter.Kurzzeichen, Mitarbeiter.Nachname, Autotypen.Typnummer, Autotypen.Typbezeichnung, Autotypen.Herstellername FROM Autotypen, Autos ,Mitarbeiter, Reservierungen WHERE Mitarbeiter.Kurzzeichen = Reservierungen.Kurzzeichen AND Reservierungen.Kennzeichen = Autos.Kennzeichen AND Autos.Typnummer = Autotypen.Typnummer ORDER By 1;`	2	5	1
	3. `SELECT Kennzeichen, Autotypen.Typnummer, Typbezeichnung, Herstellername, Nominalverbrauch FROM Autotypen, Autos WHERE Autotypen.Typnummer = Autos.Typnummer AND Nominalverbrauch<(Select AVG(Nominalverbrauch) FROM Autotypen, Autos WHERE Autotypen.Typnummer = Autos.Typnummer);` Beachten Sie, dass die Unterabfrage *nicht* lauten darf: `(SELECT AVG(Nominalverbrauch) FROM Autotypen)` Begründung: Wenn von einem Typ mehrere Autos im Fuhrpark existieren, so wirkt sich dies gewichtend auf das arithmetische Mittel aus.		5	4
	4. `SELECT Mitarbeiter.Kurzzeichen, Nachname, Sum([bis]-[von]+1) AS [Anzahl Tage] FROM Mitarbeiter, Reservierungen WHERE Mitarbeiter.Kurzzeichen = Reservierungen.Kurzzeichen GROUP BY Mitarbeiter.Kurzzeichen, Nachname HAVING Sum([bis]-[von]+1) AS [Anzahl Tage] >=20 ;` Bitte beachten: Gesucht ist die Summe der Reservierungstage, nicht die Anzahl der Reservierungen!	1	2	6

	Lösungsskizze	I	II	III

e)	
	Der Objekttyp `Autotypen` ist nicht in 3. Normalform, da der Herstellername von der Herstellernummer funktional abhängig ist. Der Hersteller muss in eine eigene Entität ausgelagert werden:
	`Autotypen (Typnummer, Typbezeichnung, Nominalverbrauch,`
	`↑Herstellernummer)`
	`Hersteller (Herstellernummer, Herstellername)`
		3	6	

	Insgesamt 80 BWE	22	41	17

1.1.2 Immobilien *LF*

Diese Aufgabe ist mit dem PC zu lösen.

Ein Immobilienmakler möchte sich selbstständig machen und dafür ein Immobilienbüro eröffnen. Der Makler hat verschiedene Immobilien im Angebot. Für die Verwaltung dieser Immobilien soll ein Programm objektorientiert entwickelt werden. Sie sollen als ersten Schritt der Softwareentwicklung eine objektorientierte Analyse (OOA) für den Problembereich durchführen und Ihre Ergebnisse mit Hilfe eines Klassendiagramms modellieren und darstellen. Verwenden Sie dafür die Unified Modeling Language (UML).

In einem Gespräch schildert Ihnen der Makler die zentralen Aspekte der einzelnen Immobilien, die er im Angebot hat:

„Also, ich habe Immobilien im Angebot, die ich gerne mit dem neuen Programm verwalten möchte. Dabei habe ich Eigentumswohnungen, Eigentumshäuser und Mietimmobilien. Für alle diese Immobilien möchte ich gerne Ort, Strasse, Wohnfläche und Anzahl der Stockwerke festhalten. Es gibt, wenn auch selten, zweistöckige Wohnungen. Diese Daten möchte ich jederzeit abfragen können.

Für die Eigentumswohnungen, die ich als Makler für den Verkauf vermittle, muss ich den Verkaufspreis, den Provisionssatz und die Nebenkosten erfassen. Diese Daten möchte ich natürlich auch jederzeit abfragen können. Außerdem möchte ich aus dem Verkaufspreis und dem Provisionssatz meine Maklergebühr ermitteln können.

Ähnlich sieht es bei den Eigentumshäusern aus, die ich für den Verkauf vermittle. Hier will ich den Verkaufspreis, die Grundstücksgröße und den Provisionssatz abspeichern und abfragen, sowie aus dem Verkaufspreis und dem Provisionssatz meine Maklergebühr ermitteln können.

Ach ja, der Provisionssatz ist für alle Verkaufsobjekte gleich, das heißt für alle Eigentumswohnungen und Eigentumshäuser gilt immer der gleiche Provisionssatz.

Dann habe ich da noch die Mietimmobilien, für die ich Vermietungen vermittle. Hier möchte ich die Monatsmiete, die Nebenkosten sowie die Kaution und deren Zinssatz abspeichern und abfragen können. Meine Provision entspricht hier zwei Monatsmieten. Vermietet wird immer bis zum Monatsende.

Bei Beendigung des Mietverhältnisses wird die Kaution, sofern keine Beanstandungen vorliegen, verzinst zurückgezahlt. Die Höhe der Rückzahlung möchte ich abfragen können, wenn ich dem Programm Beginn und Ende des Mietverhältnisses übergebe. Auf alle Kautionen wird der gleiche Zinssatz angewandt. Die Kautionen werden nur für volle Monate verzinst, denn ein Mietverhältnis beginnt immer am 1. eines Monats und endet zu einem Monatsende.“

Auf Nachfrage wurde die Zinsberechnung der Kaution näher erklärt:

„Die Zinsen werden am Ende des Jahres dem Kautionsguthaben hinzugefügt, im ersten und letzten Jahr der Vermietung muss monatsgenau gerechnet werden, der Zinssatz beträgt 3 %.“

a) Erstellen Sie aufgrund der Aussagen des Maklers ein UML-Klassendiagramm, in welchem Sie unter anderem zwischen Mietimmobilien und Eigentumsimmobilien unterscheiden.

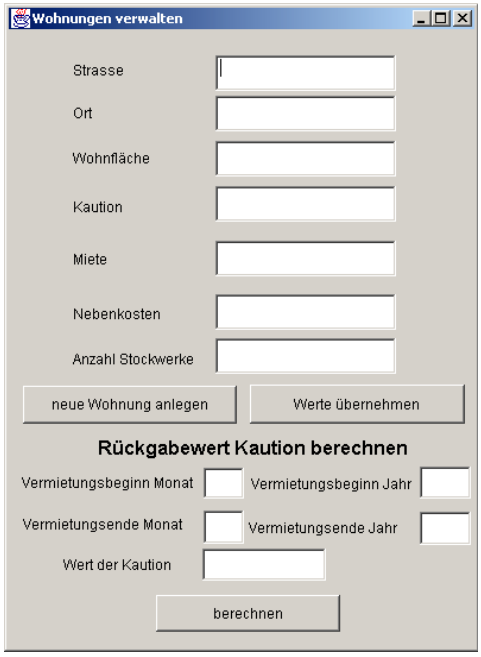
b) Implementieren Sie eine grafische Oberfläche (siehe Anlage) zur Verwaltung einer Mietwohnung.

c) Entwickeln Sie für die Funktion der Berechnung des Rückgabewertes der Kaution ein Struktogramm.

d) Belegen Sie die in der grafischen Oberfläche vorgegebenen Schaltflächen mit Funktionalität.

Anlage:

Grafische Oberfläche zur Verwaltung der Mietimmobilien:



Anmerkungen:

Zusätzliche Hilfsmittel:

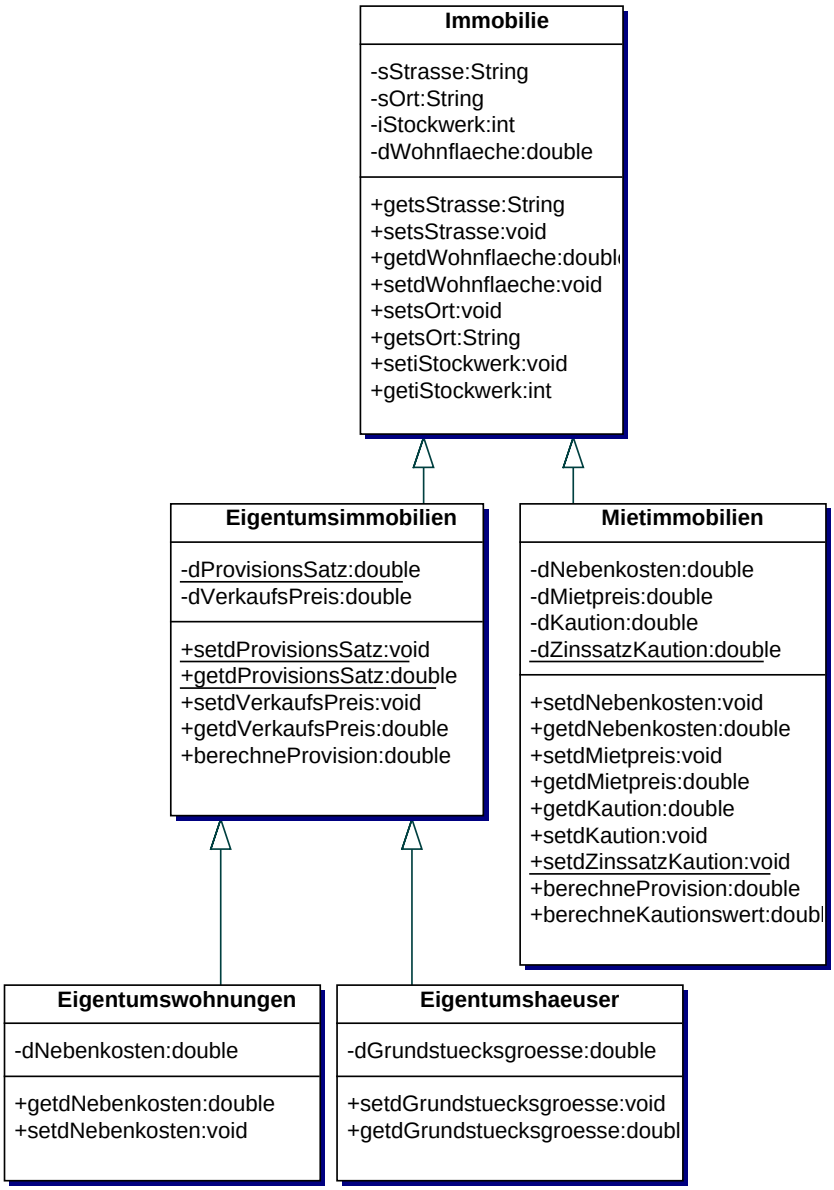
Den Schülern steht zur Lösung der Aufgabe ein PC mit objektorientierter Softwareentwicklungsumgebung zur Verfügung.

Vorgesehene Bearbeitungszeit:

120 min

Lösungsskizze, vorgesehene Bewertungseinheiten und ihre Zuordnung zu den Anforderungsbereichen I, II, III:

	Lösungsskizze	I	II	III
a)	Das folgende Klassendiagramm gibt eine von zahlreichen denkbaren Lösungen wieder. So könnte beispielsweise die Klasse Immobilie auch als abstrakte Klasse gesehen werden. Auch wäre es denkbar bei den Mietimmobilien in Mietshaus und Mietswohnung zu unterscheiden, was aber für den Makler von geringem Interesse ist.	10	20	10

	Lösungsskizze	I	II	III
	 Erläuterungen: Der Provisionssatz ist eine statische Variable (Klassenvariable), da dieser Satz für alle Objekte der Klasse Eigentumsimmobilien gleich ist. Die Methode *berechneProvision* ist polymorph, d. h. es wird erst zur Laufzeit kontextbezogen entschieden, welche Methode zum Tragen kommt.			
b)	siehe Aufgabenstellung	5	10	

	Lösungsskizze	I	II	III
c)	berechneKautionswert(int Monat1, int Jahr1, int Monat2, int Jahr2) Deklaration der Variablen: Zins: double, AnzahlJahr: int, AnzahlMonat: double Eingabe Monat1, Monat2, Jahr1, Jahr2 Jahr1<Jahr2 J: AnzahlMonat = 12 - Monat1 +1 Zins = AnzahlMonat/12*dZinssatzKaution*dKaution dKaution = dKaution+Zins AnzahlJahr=Jahr2-Jahr1+1 Wiederhole AnzahlJahr mal dKaution = dKaution* (1 + dZinssatzkaution) AnzahlMonat = Monat2 Zins = AnzahlMonat/12*dZinssatzKaution*dKaution dKaution = dKaution+Zins N: AnzahlMonat=Monat2-Monat1+1 Zins = AnzahlMonat/12*dZinssatzKaution*dKaution dKaution = dKaution+Zins Rückgabe dKaution	5	15	10

d)	Fachklassen
	`package makler;`
	`public class Immobilie {`
	`private String sStrasse;`
	`private String sOrt;`
	`private double dWohnflaeche;`
	`private int iStockwerk;`
	`public String getsStrasse() { return sStrasse; }`
	`public void setsStrasse(String Strasse) { sStrasse = Strasse; }`
	`public double getdWohnflaeche() { return dWohnflaeche; }`
	`public void setdWohnflaeche(double qm) { dWohnflaeche = qm; }`
	`public String getsOrt() { return sOrt; }`
	`public void setsOrt(String Ortsname) { sOrt = Ortsname; }`
	`public void setiStockwerk(int Etagen) { iStockwerk = Etagen; }`
	`public int getiStockwerk() { return iStockwerk; }`
	`}`
	`package makler;`
	`public class Mietimmobilien extends Immobilie {`
	`private double dNebenkosten;`
	`private double dMietpreis;`
	`private double dKaution;`
	`private static double dZinssatzKaution;`
	`public void setdNebenkosten(double nK) { dNebenkosten = nK; }`
	`public double getdNebenkosten() { return dNebenkosten; }`
	`public void setdMietpreis(double miete) { dMietpreis = miete; }`
	`public double getdMietpreis() { return dMietpreis; }`
	`public double getdKaution() { return dKaution; }`
	`public void setdKaution(double Kaution) { dKaution = Kaution; }`
		5	20	10

	Lösungsskizze	I	II	III

```
    public static void setdZinssatzKaution(double dZK)    {
      dZinssatzKaution = dZK;
    }
    public double berechneProvision()    {
      return dMietpreis* 2;
    }
    public double berechneKautionswert(int Monat1, int Jahr1,
      int Monat2, int Jahr2)
    {
      double Zins, anzahlMonat;
      int anzahlJahr;
      if (Jahr1 < Jahr2)
      {
        anzahlMonat = 12-Monat1+1;
        Zins = anzahlMonat/12*dZinssatzKaution*dKaution;
        dKaution = dKaution + Zins;
        anzahlJahr= Jahr2-Jahr1-1;
        for (int i=1; i<=anzahlJahr; ++i)
        {
          dKaution = dKaution*(1+dZinssatzKaution);
        }
        anzahlMonat = Monat2;
        Zins = anzahlMonat/12*dZinssatzKaution*dKaution;
        dKaution = dKaution + Zins;
      }
      else
      {
        anzahlMonat = Monat2 - Monat1+1 ;
        Zins = anzahlMonat/12*dZinssatzKaution*dKaution;
        dKaution = dKaution + Zins;
      }
      return dKaution;
   }
}
```

Funktionalitäten in der Oberfläche (drei Befehlsschaltflächen)

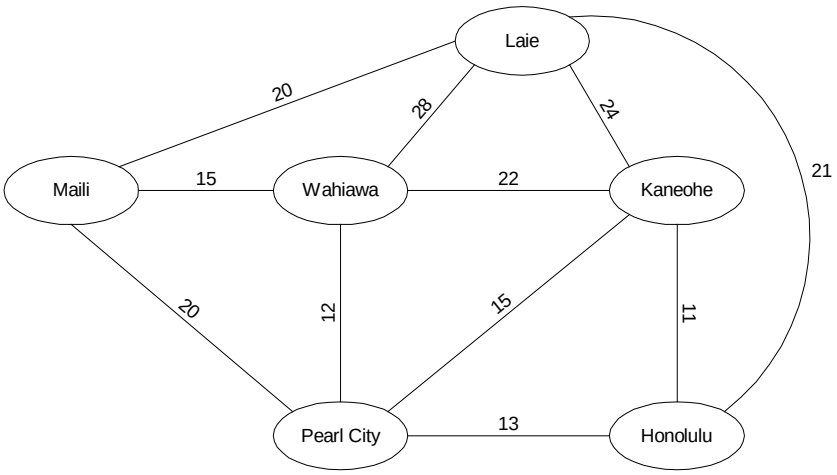
```
  void btNeueWohnung_actionPerformed(ActionEvent e)
  {
    aktuelleWohnung= new Mietimmobilien();
    tFFlaeche.setText("");
    tFJahr1.setText("");
    tFJahr2.setText("");
    tFKaution.setText("");
    tFMiete.setText("");
    tFMonat1.setText("");
    tFMonat2.setText("");
    tFNebenkosten.setText("");
    tFStrasse.setText("");
    tFOrt.setText("");
    tFAnzahlStockwerke.setText("");
  }
  void btWerteuebernehmen_actionPerformed(ActionEvent e)
  {
    aktuelleWohnung.setdWohnflaeche
      (Double.parseDouble(tFFlaeche.getText()));
    aktuelleWohnung.setsStrasse(tFStrasse.getText());
    aktuelleWohnung.setsOrt(tFOrt.getText());
    aktuelleWohnung.setdKaution
      (Double.parseDouble(tFKaution.getText()));
    aktuelleWohnung.setdMietpreis
      (Double.parseDouble(tFMiete.getText()));
    aktuelleWohnung.setdNebenkosten
      (Double.parseDouble(tFNebenkosten.getText()));
    aktuelleWohnung.setiStockwerk
      (Integer.parseInteger(tFAnzahlStockwerke.getText()));
  }
  void btBerechnen_actionPerformed(ActionEvent e)  {
    int m1, m2, j1, j2;
    m1=Integer.parseInt(tFMonat1.getText());
    m2=Integer.parseInt(tFMonat2.getText());
    j1=Integer.parseInt(tFJahr1.getText());
    j2=Integer.parseInt(tFJahr2.getText());
    tFWert.setText(Double.toString(aktuelleWohnung.
      berechneKautionswert(m1,j1,m2,j2)));
  }
}
```

	Insgesamt 120 BWE	25	65	30

1.1.3 Graphenalgorithmen *LF*

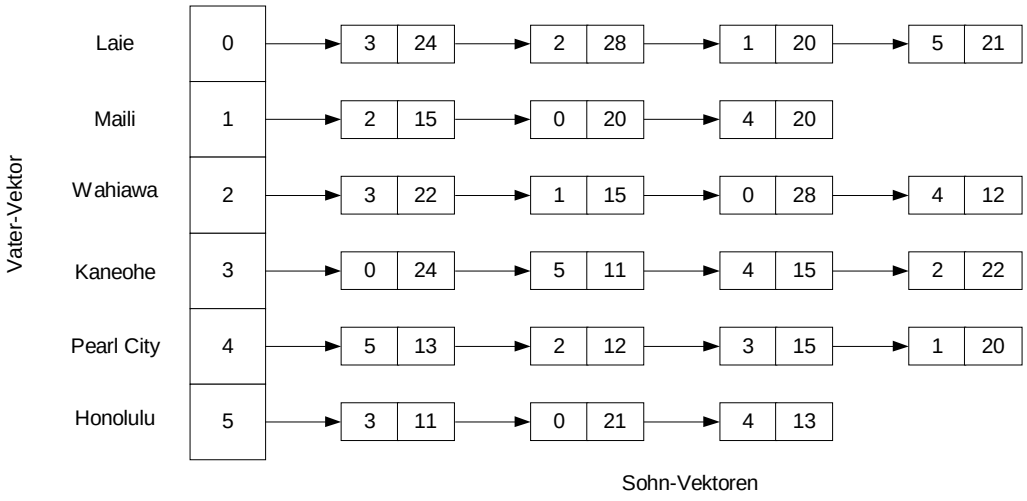
Beim Rundreiseproblem soll eine möglichst kurze Rundreise durch vorgegebene Städte bestimmt werden. In der Rundreise müssen alle Städte enthalten sein und sie muss am Ausgangspunkt enden.

a) Berechne eine Rundreise für die Städte der Hawaiiinsel Oahu nach der „Nächster-Nachbar-Strategie“, bei der man von einer Stadt aus immer zur nächstliegenden noch nicht besuchten Stadt weiter reist.



b) Zur Berechnung einer Rundreise müssen die Daten des Graphen in eine geeignete Datenstruktur eingelesen werden. Als ersten Ansatz nehmen wir dafür eine zweidimensionale Reihung `int[][] distanzen` in dem wir die Entfernungen direkt verbundener Städte i und k als `distanzen[i][k]` speichern. Die als Index ungeeigneten Städtenamen werden in einer separaten Reihung `String[] staedte` gespeichert. Eingelesen wird mit Hilfe der Methoden der Klasse *Eingabe*. Schreibe eine Methode zum Einlesen der Daten in diese Datenstruktur auf Basis der Anlage.

c) Hat man viele Städte, aber nur wenige direkte Verbindungen, so ist die sogenannte Adjazenzlisten-Darstellung günstiger.



Zur Realisierung der Adjazenzlisten-Darstellung in Java benutzen wir den ADT Vector. Der Vater-Vektor erhält so viele Elemente wie Städte vorhanden sind. Jedes Element des Vater-Vektors ist selbst ein Vektor. Der zur ersten Stadt gehörige Vektor enthält alle direkten Verbindungen zu den anderen Städten. Die Elemente der Sohn-Vektoren haben im Gegensatz zu den Verbindung-Objekten nur zwei Attribute, wofür eine eigene Klasse erforderlich ist. Schreibe eine Methode zum Erstellen der Adjazenzlisten-Darstellung.

d) Eine kürzeste Rundreise, bei der jede Stadt mit Ausnahme der Ausgangsstadt nur einmal besucht wird, soll mit einem Backtracking-Algorithmus ermittelt werden. Skizziere den Anfang eines Suchbaums zur

Organisation der systematischen Suche, erläutere dessen prinzipiellen Aufbau und die Funktionsweise von Backtracking.

e) Schreibe eine Methode zur Ermittlung der kürzesten Rundreise unter Beachtung der Vorgaben aus der Anlage.

Anlage:

```
class Verbindung {
  int von;
  int nach;
  int distanz;

  public Verbindung (int von, int nach, int distanz) {
    this.von = von;
    this.nach = nach;
    this.distanz = distanz;
  }
}

public class Eingabe {
  // diese Klasse stellt für die Eingabe des Graphen folgende Methoden zur Verfügung
  // im Beispiel haben wir 6 Städte und 11 Verbindungen
  // angegeben sind nur die Methodenköpfe
  public int gibAnzahlStaedte ();
  public String gibStadt(int i);
  public int gibAnzahlVerbindungen ();
  public Verbindung gibVerbindung(int i);
}

public class Rundreiseproblem {
  String[] staedte;        // die Städtenamen
  int anzahlStaedte;       // die Anzahl der Städte
  int[][] distanzen;       // die Entfernungstabelle
  Vector entfernungen;     // die Adjazenzliste

  int[] besucht;           // die bisher besuchten Städte
  int kuerzesteWeglaenge; // die bisher kürzeste Rundreisenlänge

  public void einlesen() {
    // Staedte einlesen
    Eingabe graph = new Eingabe();
    anzahlStaedte = graph.gibAnzahlStaedte();
    staedte = new String[anzahlStaedte];
    <...ergänzen...>
    // Distanzen einlesen
    distanzen = new int[anzahlStaedte][anzahlStaedte];
    <...ergänzen...>
    }
  }

  public void loesungAusgeben() {
    for ( int i = 0 ; i < anzahlStaedte ; i++ )
      System.out.print(besucht[i] + " ");
    System.out.println();
    int summe = 0;
    for (int i = 0; i < anzahlStaedte-1; i++) {
      int d = distanzen[besucht[i]][besucht[i+1]];
      System.out.print( d + " ");
      summe = summe + d;
    }
    int d = distanzen[besucht[anzahlStaedte-1]][0];
    summe = summe + d;
    System.out.print( d + " ");
    System.out.println(" : " + summe);
  }
```

```
  public void starteSuche() {
    besucht = new int[anzahlStaedte];
    besucht[0] = 0;  // Start in Stadt 0
    kuerzesteWeglaenge = 10000000;
    sucheRundreise(0, 1, 0);
  }

  // stadt ist aktueller Endpunkt der Rundreise
  // anzahl gib die Anzahl der schon besuchten Städte an
  // laenge ist die bisherige Rundreisenlänge
  public void sucheRundreise(int stadt, int anzahl, int laenge) {
    <... ergänzen ...>
  }
}
```

Anmerkungen

Zielsetzung:
Die Aufgabenstellung erfordert die Anwendung von Klassen in einem vorgegebenen Rahmenprogramm, die Implementierung der komplexen Adjazenzlisten-Datenstruktur, die Modellierung eines Suchbaums und Programmierung eines Backtracking-Algorithmus im Hinblick auf eine konkrete Anwendungssituation.

Unterrichtliche Voraussetzungen:
Im Unterricht wurden unter anderem Reihungen und die abstrakten Datentypen lineare Liste, Keller und Schlange behandelt. Am Beispiel von linearer Liste, Keller und Suchbaum haben die Schüler die Vererbung und Klassenhierarchie kennen gelernt und angewendet. Dabei wurde auch die Java-Klasse Vector eingesetzt. Die Adjazenzlisten-Darstellung ist den Schülern unbekannt. Das schwierige Backtracking-Verfahren wurde an mehreren Beispielen geübt. Das Rundreiseproblem ist neu.

Zusätzliche Hilfsmittel: Kurzbeschreibung der Klasse Vector

Vorgesehene Bearbeitungszeit: 90 min

Lösungsskizze, vorgesehene Bewertungseinheiten und ihre Zuordnung zu den Anforderungsbereichen I, II, III:

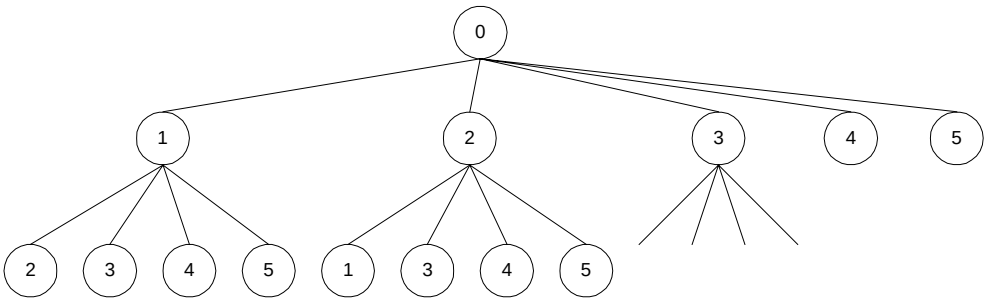
<table>
<tr><th></th><th>Lösungsskizze</th><th>I</th><th>II</th><th>III</th></tr>
<tr><td>a)</td><td>Ausgehend von Laie liefert die Nächster-Nachbar-Strategie eine Rundreise der Länge 95
Verwendung gelernter Arbeitstechniken in einem einfachen Beispiel</td><td>4</td><td></td><td></td></tr>
<tr><td>b)</td><td>Anwendung von Standardalgorithmen auf Reihungen. Verstehen und Anwenden der vorgegebenen einfachen Klasse Eingabe.
<pre>public void einlesen() {
  // Staedte einlesen
  ....
  for (int i = 0; i < anzahlStaedte ; i++)
    staedte[i]= graph.gibStadt(i);

  // Distanzen einlesen
  distanzen = new int[anzahlStaedte][anzahlStaedte];
  int kanten = graph.gibAnzahlVerbindungen();
  Verbindung kante;

  for (int i =0 ; i < kanten ; i++) {
    kante = graph.gibVerbindung(i);
    distanzen[kante.von][kante.nach]= kante.distanz;
    distanzen[kante.nach][kante.von]= kante.distanz;
  }
}</pre>Implementieren einer einfachen vorliegenden Problemlösung auf der Basis geübter Verfahrensweisen</td><td>12</td><td></td><td></td></tr>
</table>

<table>
<tr><th></th><th>Lösungsskizze</th><th>I</th><th>II</th><th>III</th></tr>
<tr><td>c)</td><td>Konstruktion einer komplexen geschachtelten Datenstruktur und einer neuen Klasse für die Elemente der Adjazenzlisten. Anwendung von Standardalgorithmen und Bibliotheksklassen.
<pre>
public void einlesen2() {
  Eingabe graph = new Eingabe();
  anzahlStaedte = graph.gibAnzahlStaedte();
  entfernungen = new Vector(anzahlStaedte);

  for (int i=0 ; i < anzahlStaedte ; i++)
    entfernungen.add(new Vector());

  int kanten = graph.gibAnzahlVerbindungen();
  Verbindung kante;
  Vector sohn;
  for (int i =0 ; i < kanten ; i++) {
    kante = graph.gibVerbindung(i);
    sohn = (Vector) entfernungen.get(kante.von);
    sohn.add(new Element(kante.nach, kante.distanz));
    sohn = (Vector) entfernungen.get(kante.nach);
    sohn.add(new Element(kante.von, kante.distanz));
  }
}
</pre>
Ersetzen einer gegebenen Datenstruktur durch eine geeignete andere
Nutzung vorhandener Programmbibliotheken für die eigene Problemlösung</td><td></td><td>16</td><td></td></tr>
<tr><td>d)</td><td>Teilweise grafische Darstellung des Suchbaums für das Backtrackingverfahren. Beschreibung der Struktur des entwickelten Suchbaums: auf jeder Stufe verringert sich die Anzahl der möglichen Verzweigungen um eins, weil die schon besuchten Knoten nicht noch einmal besucht werden dürfen. Alle Blätter des Suchbaumes enthalten den Knoten 0, weil die Rundreise sich schließen muss.


Beschreibung des Backtracking-Verfahrens für das Rundreiseproblem anhand der Zeichnung
Anwendung eines bekannten grafischen Modellierungsverfahrens auf ein neues Problem
Dokumentieren der erhaltenen Lösung mit angemessenen Mitteln</td><td></td><td>8</td><td></td></tr>
<tr><td>e)</td><td>Für den Backtracking-Algorithmus ist der Aufruf und der Methodenkopf vorgegeben. Der eigentliche komplexe Kern des Backtracking-Algorithmus ist selbstständig zu entwickeln.
<pre>
public void sucheRundreise(int stadt, int anzahl, int laenge) {
  for (int i=0 ; i < staedte.length ; i++) {
    boolean schonBesucht = false;
    for (int j=0; j < anzahl ; j++)
      if (besucht[j] == i)
        schonBesucht = true;
    if (! schonBesucht && (distanzen[stadt][i] > 0)) {
      // Stadt hinzunehmen
      besucht[anzahl] = i;
      laenge = laenge + distanzen[stadt][i];
      if (anzahl + 1 == staedte.length) {
        // Rundreise gefunden,
        // Weg zurück zur 0. ten Stadt berücksichtigen
        if ((distanzen[i][0] > 0) && (laenge + distanzen[i][0] <
             kuerzesteWeglaenge)) {
</pre></td><td></td><td></td><td>12</td></tr>
</table>

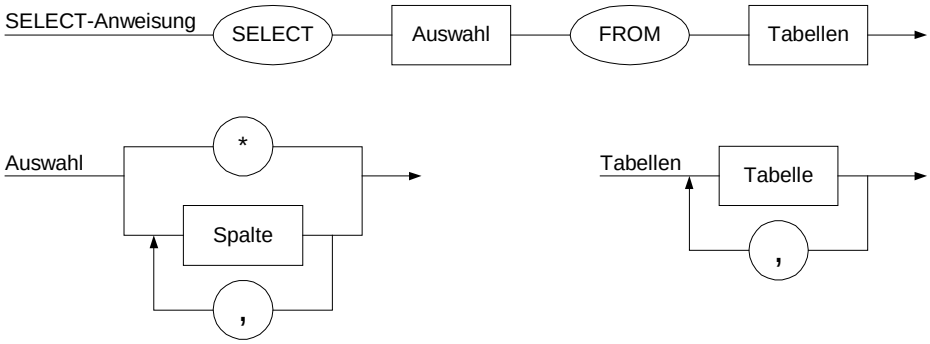
	Lösungsskizze	I	II	III

	<pre>kuerzesteWeglaenge = laenge + distanzen[i][0];
	loesungAusgeben();
	}
	}
	else {
	sucheRundreise(i, anzahl+1, laenge);
	}
	laenge = laenge – distanzen[stadt][i];
	besucht[anzahl] = 0;
	};
	}
	}</pre> Planmäßiges Verarbeiten komplexer Gegebenheiten, Durchführung einer komplexeren Problemanalyse, Zerlegung in Teilprobleme und Lösung			

	Insgesamt 52 BWE	16	24	12

1.1.4 SQL-Grammatik *LF*

In der Informatik benutzen wir zur Lösung von Problemen unterschiedliche Sprachen wie z. B. Programmiersprachen, HTML oder die Datenbanksprache SQL. Im Folgenden ist ein Ausschnitt von SQL mit Syntaxdiagrammen (in einer für diese Aufgabe vereinfachten Form) dargestellt:



a) Erläutern Sie die drei Syntaxdiagramme für SELECT-Anweisungen in SQL. Spalte und Tabelle sind Bezeichner aus Kleinbuchstaben ohne Sonderzeichen und Ziffern. Ergänzen Sie die Syntaxdiagramme für Spalte und Tabelle.

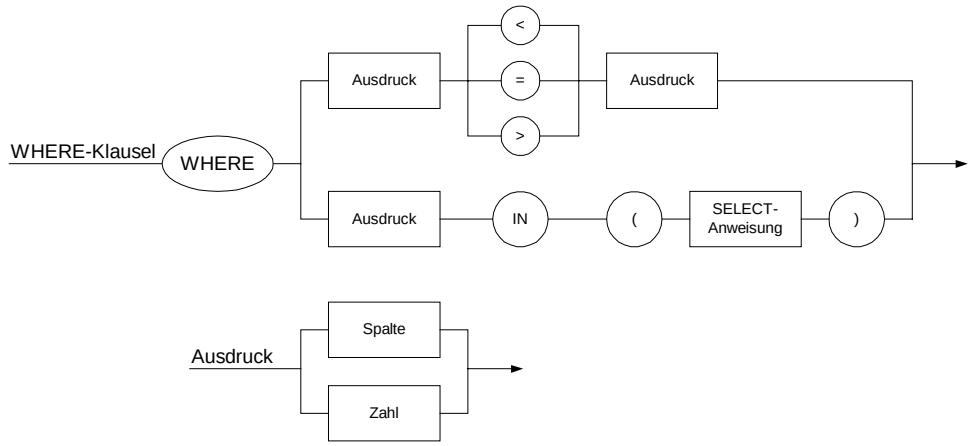
b) Übersetzen Sie die fünf Syntaxdiagramme aus Aufgabe a) in eine Grammatik und geben Sie den Typ der Grammatik an.

c) Konstruieren Sie einen endlichen Automaten als Akzeptor für die Sprache der SELECT-Anweisungen. Stellen Sie ihn durch ein Zustandsdiagramm dar.

d) Programmieren Sie in Prolog einen Akzeptor für *Auswahl*.

e) Erweitern Sie die Syntaxdiagramme so, dass auch die Aggregatfunktion MAX( ) benutzt werden kann.

f) Wir ergänzen die Syntaxdiagramme aus a) um eine optionale WHERE-Klausel hinter der FROM-Klausel gemäß dem folgenden Syntaxdiagramm:



Vergleichen Sie die Sprachen aus a), e) und f) in Bezug auf Grammatik, Sprachtyp und Akzeptor.

Anmerkungen

Zielsetzung:
Die Aufgabe vernetzt den Themenbereich Grammatiken und formale Sprachen mit der Datenbanksprache SQL. Syntaxdiagramme sind in Grammatiken und das Zustandsdiagramm eines erkennenden Automaten umzusetzen, ein Akzeptor ist in Prolog zu programmieren. Die Erweiterung der Sprache führt auf Klammerterme, die nur im Fall f) geschachtelt werden können. Im Fall e) bleibt es bei einer regulären Sprache, im Fall f) entsteht eine kontextfreie Sprache.

Unterrichtliche Voraussetzungen:
Im Unterricht wurden Syntaxdiagramme, Grammatiken, endliche Automaten, Kellerautomaten, Zustandsdiagramme, Akzeptoren samt deren Modellierung in Prolog an verschiedenen Beispielen behandelt. Der Problembereich schließt an SQL-Abfragen bei Datenbanken an, betrachtet die inhaltlichen Aspekte jedoch unter ganz anderen Fragestellungen.

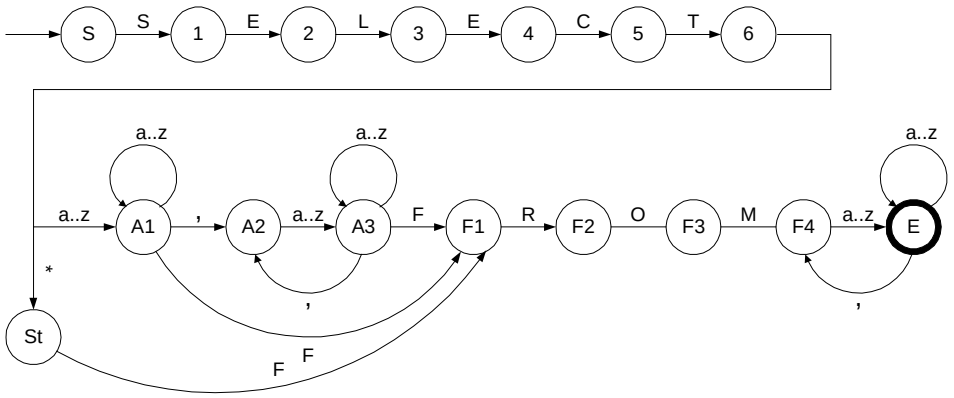
Zusätzliche Hilfsmittel: keine

Vorgesehene Bearbeitungszeit: 80 min

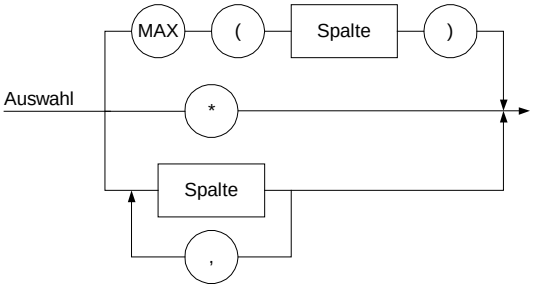
Lösungsskizze, vorgesehene Bewertungseinheiten und ihre Zuordnung zu den Anforderungsbereichen I, II, III:

	Lösungsskizze	I	II	III
a)	Die SELECT-Anweisungen enthalten keine WHERE-, GROUP BY-, ORDER BY-Klauseln. Es können alle oder ausgewählte Spalten projiziert werden und das Kreuzprodukt von Tabellen bestimmt werden. Selektionen und Joins sind nicht möglich. Die Syntaxdiagramme enthalten alle Grundstrukturen wie Sequenz, Fallunterscheidung, Schleife. Wiedergabe von Sachverhalten im gelernten Zusammenhang Beschreibung und Verwendung einfacher Syntaxdiagramme	10		

b)	Zur Übersetzung in eine Grammatik setzen wir Terminale in Anführungszeichen und Variablen in spitze Klammern. Schleifen werden in rekursive Produktionen umgesetzt.
	
	`<SELECT-Anweisung> → 'SELECT' <Auswahl> 'FROM' <Tabellen>`
	`<Auswahl> → * | <Spalten>`
	`<Spalten> → <Spalte> | <Spalte> ',' <Spalten>`
	`<Tabellen> → <Tabelle> | <Tabelle> ',' <Tabellen>`
	`<Spalte> → <Bezeichner>`
			6	

	Lösungsskizze	I	II	III
	<Tabelle> → <Bezeichner> <Bezeichner> → a|b|c|...|x|y|z|a<Bezeichner>| |b<Bezeichner>|...|z<Bezeichner> Planvolles Anwenden eines bekannten Verfahrens auf ein neues Beispiel Transformation von Schleifen in Rekursionen			
c)	 Selbstständiges Übertragen des Gelernten auf vergleichbare neue Situationen Entwickeln von Automaten		10	

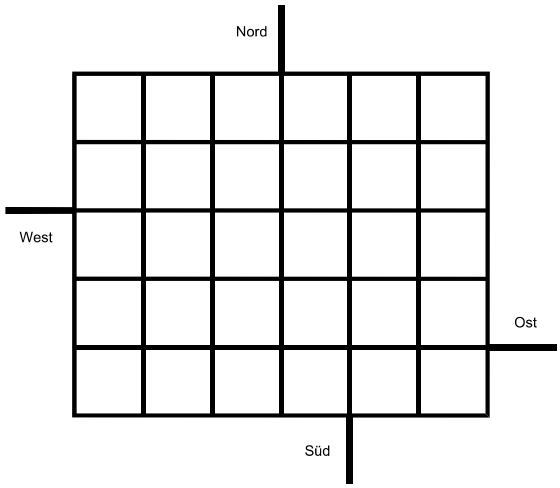
d)	`akzeptiere_auswahl(Auswahl, Rest):-`
	`  Auswahl = ['*'|Rest].`
	`akzeptiere_auswahl(Auswahl, Rest):-`
	`  akzeptiere_spalten(Auswahl, Rest).`
	
	`akzeptiere_spalten(Auswahl, Rest):-`
	`  akzeptiere_bezeichner(Auswahl, [','| Rest1]),`
	`  akzeptiere_spalten(Rest1, Rest).`
	`akzeptiere_spalten(Auswahl, Rest):-`
	`  akzeptiere_bezeichner(Auswahl, Rest).`
	
	`akzeptiere_bezeichner(Name, Rest):-`
	`  Name = [Anfang|Ende],`
	`  kleinbuchstabe(Anfang),`
	`  akzeptiere_bezeichnerendung(Ende, Rest).`
	`akzeptiere_bezeichnerendung(Name, Rest):-`
	`  Name = [Anfang|Ende],`
	`  kleinbuchstabe(Anfang),`
	`  akzeptiere_bezeichnerendung(Ende, Rest).`
	`akzeptiere_bezeichnerendung(Ende, Ende).`
	
	Planvolles Anwenden eines bekannte Verfahrens auf ein komplexes Beispiel
	Prolog-Akzeptor aus Syntaxdiagrammen entwickeln
			12	

e)	 Hier wird Wissen über die Syntax von Aggregatfunktionen gebraucht. Die Erweiterung lässt sich leicht im Syntaxdiagramm von Auswahl ergänzen.	2		

	Lösungsskizze	I	II	III
	Verwendung geübter Verfahrensweisen in wiederholendem Zusammenhang			
f)	Kurzgefasst sollten folgende Aspekte beschrieben werden: In einer IN-Bedingung kann eine weitere SELECT-Anweisung auftreten und darin natürlich wieder eine. Grundsätzlich sind daher beliebig tief geschachtelte SELECT-Anweisungen möglich. Die Sprache der SELECT-Anweisungen ist also nicht mehr regulär sondern kontextfrei. Sie kann nicht mehr von einem endlichen Automaten erkannt werden. Jetzt wird ein Kellerautomat benötigt. Die Grammatik war in beiden Fällen kontextfrei. Die Konstruktion des endlichen Automaten in Aufgabe c) zeigt aber, dass man die kontextfreie Grammatik aus b) in eine reguläre Grammatik nach c) überführen kann. Die Klammern aus e) sind nicht schachtelbar, die Erweiterung führt also nicht aus dem Bereich der regulären Sprachen hinaus. Durchführung einer komplexeren Problemanalyse Verarbeiten und Werten komplexer Gegebenheiten			8
	Insgesamt 48 BWE	12	28	8

1.1.5 Verkehrsplanung *LF*

Die Stadtverwaltung einer Stadt, bei der im Stadtkern (siehe nebenstehende Abbildung) alle Straßen schachbrettartig angeordnet sind, möchte den Verkehrsfluss durch den Stadtkern optimieren, da eine Umgehung nicht möglich ist.



Da die Straßen sehr eng sind, hat man vor, alle Straßenabschnitte im Stadtkern als Einbahnstraßen auszuweisen. Die Zufahrt in und die Ausfahrt aus dem Stadtkern ist wegen der erhaltenen Stadtmauern nur über die vier Straßen (Nord, Ost, Süd und West) möglich. Dabei muss jeder Ort von mindestens einer der vier Zufahrten aus erreichbar sein und der Stadtkern muss von jedem Ort aus wieder verlassen werden können.

Die Stadtverwaltung fordert von einer Lösung, dass der Durchgangsverkehr auf möglichst kurzem Wege den Stadtkern durchquert.

a) Schlagen Sie eine Datenmodellierung zur Bearbeitung dieses Problems vor.

b) Die folgende Scheme-Funktion sucht einen kürzesten Weg zwischen zwei Positionen:

```
01: (define (Suchverfahren Start-Ort Ziel-Ort Suchraum)
02:
03: ; (define (finde-Nachfolger Ort)
04: ;    --> Auswertungsteil: Gibt die Nachfolger als Liste zurueck
05:
06: (define (expandiere-Knoten Weg Nachfolger)
07:  (cond ((null? Nachfolger)())
08:        (else (cons (cons (car Nachfolger) Weg)
09:                    (expandiere-Knoten Weg (cdr Nachfolger))))))
10:
11: (define (suche besucht Wege)
12:  (cond ((null? Wege) #f)
13:        ((member (caar Wege) besucht)
14:         (suche besucht (cdr Wege)))
15:        ((member Ziel-Ort (finde-Nachfolger (caar Wege)))
16:         (cons (car (member Ziel-Ort (finde-Nachfolger (caar Wege))))
17:               (car Wege)))
```

```
18:         (else
19:          (suche
20:           (cons (caar Wege) besucht)
21:           (append
22:            (cdr Wege)
23:            (expandiere-Knoten (car Wege)
24:                               (finde-Nachfolger (caar Wege)))))))))
25: (suche () (list (list Start-Ort))))
```

Analysieren Sie, welches Suchverfahren hier verwendet wird und erläutern Sie anhand der vorgegebenen Funktion, wie dieses Suchverfahren arbeitet. Begründen Sie, dass damit ein kürzester Weg gefunden wird.

c) Wie viele verschiedene Belegungen des Stadtplanes mit Einbahnstraßen sind prinzipiell möglich und welche Bedeutung hat das für die Lösbarkeit des Problems?

d) Zum Vergleich verschiedener Belegungen des Stadtplanes mit Einbahnstraßen benötigt man eine Bewertungsfunktion. Welche Größen sollten bei der Bewertungsfunktion für verschiedene Lösungsvorschläge berücksichtigt werden?
Führen Sie eine funktionale Modellierung durch und implementieren Sie eine Funktion einschließlich geeigneter Hilfsfunktionen, mit der die Bewertung einer vorgegebenen Lösung durchführt werden kann.

e) Beschreiben Sie zur Lösung des Verkehrsproblems ein Verfahren, das die Prinzipien von evolutionären Algorithmen verwendet. Bewerten Sie die Angemessenheit einer solchen Lösung für das vorliegende Problem.

Anmerkungen

Unterrichtliche Voraussetzungen:
Im Unterricht wurden Suchverfahren, insbesondere Tiefen- und Breitensuche, Suchverfahren mit Verwendung einer Bewertungsfunktion sowie evolutionäre Algorithmen und ihre Einsetzbarkeit erarbeitet. Außerdem sind Abschätzungen der Größe von Suchräumen und Abschätzungen über den Zeitaufwand von Suchverfahren behandelt worden. Funktionale Modellierung und Implementation von Funktionen mit einer funktionalen Programmiersprache sind geübt. Datenstrukturen wie Listen, Assoziationslisten, Bäume und Graphen sind bekannt.

Zusätzliche Hilfsmittel: keine

Vorgesehene Bearbeitungszeit: 90 min

Lösungsskizze, vorgesehene Bewertungseinheiten und ihre Zuordnung zu den Anforderungsbereichen I, II, III:

	Lösungsskizze	I	II	III
a)	Es handelt sich um einen Graphen. Grundlegend unterschiedlich sind die Modellierungen mit Kanten oder mit Knoten. Beide sind angemessen, da es sich wegen der Einbahnstraßenregelung um einen gerichteten Graphen handelt. Im ersten Fall werden alle möglichen Kanten erzeugt und die Nachfolgekanten gespeichert. Im anderen Fall ist zu beachten, dass ein Ausgangsknoten nicht als Zielknoten vorkommt. Besonders zu beachten ist die Beschreibung der Zu- und Abfahrtsstraßen. Als Datentyp bietet sich eine Assoziationsliste an, die zu jeder Kante bzw. jedem Knoten die Kanten bzw. die Knoten angibt, zu denen man von dort aus kommen kann.	3	6	
b)	Es handelt sich um Breitensuche. Dies erkennt man in den Zeilen 21 bis 24 der Suchfunktion. In der Liste Wege werden die schon entwickelten Wege gehalten. Die durch Expansion neu gefundenen Nachfolgerknoten werden hinten in die Liste eingefügt. Damit wird beim erneuten rekursiven Aufruf der Suchfunktion zunächst untersucht, ob der Zielknoten unter den restlichen Knoten ist, bevor die neu gefundenen Nachfolgerknoten untersucht werden. Die Breitensuche liefert den kürzesten Weg, weil alle Kanten gleich bewertet werden.	6	3	

	Lösungsskizze	I	II	III
c)	Hier reicht eine einfache Beschreibung: Jede Kante kann je nach Einbahnstraßenrichtung zwei verschiedene Werte tragen. Daher gibt es $2^{\text{Kantenzahl}}$ Möglichkeiten, also hier 2^{71}. Das Problem wächst also exponentiell. Allerdings ist die Stadtverwaltung sicher nicht an der generellen Lösbarkeit interessiert, sondern an der ganz konkreten für ihre eigene Stadt.		3	
d)	Einerseits ist die Länge der optimalen Durchgangsverbindungen zu berücksichtigen. Diese erhält man, indem die Längen aller Verbindungen von einem Eingang zu einem beliebigen Ausgang addiert werden. Die Längen der Verbindungen sind die Längen der Listen, die der Aufruf der Funktion Suchverfahren für den jeweiligen Start- und Zielort liefert. Zusätzlich ist aber noch die Zulässigkeit zu untersuchen. Dazu muss von jedem Ort aus ein Ausgang erreichbar sein und von den Zufahrten aus jeder Ort. Darüber hinaus ist die Fahrtlänge für den Durchgangsverkehr einzubringen. Dazu ist von jeder Zufahrt aus der kürzeste Weg zu jeder gewünschten Ausfahrt zu bestimmen und zu summieren. Die Berücksichtigung zusätzlicher Bewertungen wie etwa unterschiedlicher Verkehrsdichten werden nicht erwartet. Erwartet wird eine verbale Beschreibung der funktionale Modellierung sowie deren Implementation mit Scheme.		9	8
e)	Die Schüler können hier verschiedene Ansätze finden. Ein möglicher Ansatz ist: man wählt willkürlich einen oder mehrere Straßenabschnitte (Kanten) aus, ändert bei ihnen die Einbahnstraßenrichtung und bewertet neu. Danach arbeitet man mit der besseren Variante weiter. Abbruchkriterium kann die Zahl der Schritte oder ein Wert der Bewertungsfunktion sein. Der Nachteil eines solchen Verfahrens ist, dass zwar eine gute, jedoch nicht immer eine optimale Lösung gefunden wird. Der Lösungsraum des Problems ist mit ca. 10^{21} recht groß. Es bieten sich aber gute Heuristiken an, die andere Ansätze ermöglichen.		1	5
	Insgesamt 44 BWE	9	22	13

1.1.6 Verkehrszentralregister *LF*

Das Kraftfahrtbundesamt in Flensburg führt das Verkehrszentralregister, in dem die im Straßenverkehr auffällig gewordenen Verkehrsteilnehmer registriert werden. Einträge werden durch Bußgeldbehörden bei Ordnungswidrigkeiten mit einer Geldbuße von mindestens 40 Euro und durch Gerichte bei Straftaten im Zusammenhang mit dem Straßenverkehr veranlasst. Jeder Eintrag wird nach Art und Schwere bepunktet und nach bestimmten Fristen gelöscht. Bei 18 oder mehr Punkten wird die Fahrerlaubnis entzogen. Die mit einem Eintrag verbundenen Punkte werden bei Ordnungswidrigkeiten nach 2 Jahren, bei Straftaten, die nicht im Zusammenhang mit Alkohol und Drogen stehen, nach 5 Jahren ansonsten nach 10 Jahren gelöscht.

a) Modellieren Sie eine Datenbank für das Verkehrszentralregister als ER-Diagramm nach folgenden Vorgaben: Von jedem erfassten Fahrer sind erforderlich der Vor- und Nachname, die Personalausweisnummer sowie die Adresse und die ihn betreffenden Einträge sowie Angaben über jeden Eintrag und die veranlassende Behörde. Bei den Führerscheinen sind die Nummer, die Führerscheinklasse, das Ausstellungsdatum, die Probezeit (2 Jahre) und der Vermerk, ob ein Fahrverbot besteht, erforderlich. Für den Austausch zwischen dem Verkehrszentralregister und einer Behörde werden deren Name, Adresse, Telefon- und Telefax-Nummer benutzt.

b) Analysieren und beschreiben Sie im Detail die im ER-Diagramm vorkommenden Beziehungen. Übertragen Sie das ER-Diagramm in ein optimiertes Relationenmodell und erläutern Sie, wie Probezeit und Führerscheinentzug auf Daten abgebildet werden können.

c) Erläutern Sie die beiden folgenden SQL-Anweisungen:

c1)
```
SELECT Nachname, Vorname, Klasse
FROM Fahrer, Führerschein
WHERE Fahrer.FührerscheinNr = Führerschein.FührerscheinNr
      AND Klasse = 'B'
```

c2)
```
SELECT Nachname, Vorname, PLZOrt
FROM Fahrer, Eintrag
WHERE Eintrag.PersonalausweisNr = Fahrer.PersonalausweisNr
      AND Punkte = (SELECT MAX(Punkte) FROM Eintrag)
```

d) Formulieren Sie SQL-Anweisungen zur Lösung folgende Fälle:

d1) Gesucht ist die Adresse der Behörde *Ordnungsamt Wiesbaden.*

d2) Erstellen Sie eine Statistik für die Anzahl der Einträge aufgeschlüsselt nach der Art (Ordnungswidrigkeit, Straftat mit bzw. ohne Alkohol und Drogen)

d3) Gesucht ist eine Liste aller Fahrer, die mehr als 20 Punkte haben.

e) Nach einem Unfall hat der Verursacher Fahrerflucht begangen. Glücklicherweise konnte sich der Geschädigte das Kfz-Kennzeichen merken. Wieso kann er bei unserer Modellierung und wieso darf er bei der tatsächlichen Modellierung beim Kraftfahrtbundesamtes (www.kba.de) den Halter nicht per Online-Recherche ermitteln? Erläutern Sie, wie der Verursacher zu seinen Punkten kommt.

Anmerkungen

Zielsetzung:
In der Aufgabe sind die Modellierung einer Datenbank als ER-Diagramm, die Umsetzung in Relationen, die Analyse und Konstruktion von SQL-Anweisungen sowie die Interpretation der Lösung im Hinblick auf technische und Datenschutz-Aspekte gefordert.

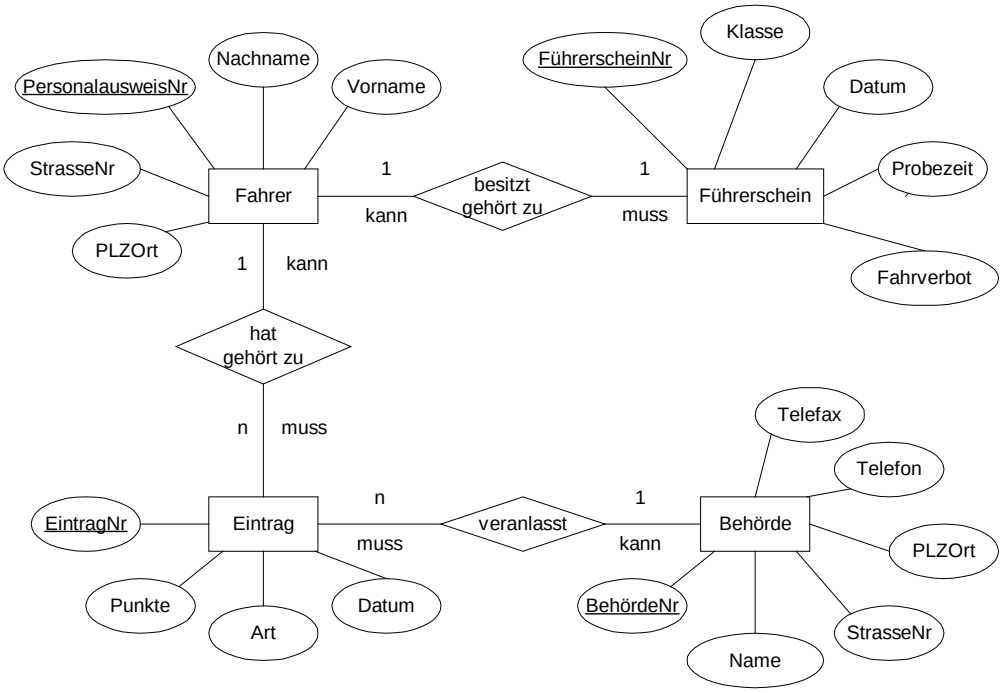
Unterrichtliche Voraussetzungen:
Im Unterricht wurden unter anderem die Modellierung von Mini-Welten mit ER-Diagrammen, die Analyse der Beziehungen und Komplexitäten, die Umsetzung in das Relationenmodell, Relationenalgebra und die Abfragesprache SQL behandelt.

Zusätzliche Hilfsmittel: keine

Vorgesehene Bearbeitungszeit: 80 min

Lösungsskizze, vorgesehene Bewertungseinheiten und ihre Zuordnung zu den Anforderungsbereichen I, II, III:

	Lösungsskizze	I	II	III
a)	Das zu entwerfende ER-Diagramm muss die vier Objekttypen *Fahrer, Führerschein, Eintrag* und *Behörde* enthalten, sowie die drei Beziehungen *besitzt/gehört zu, hat/gehört zu* und *veranlasst.* Primärschlüssel sind zu unterstreichen, bei der Gestaltung der Attribute gibt es Freiheiten. Die im Text genannten Attribute müssen vorhanden, bzw. wie bei Adresse aufgelöst sein.	8	8	

	Lösungsskizze	I	II	III
	 Selbstständiges Übertragen des Gelernten auf vergleichbare neue Situationen Beschreiben und Darstellen einfacher Objekte Modellierung einer Miniwelt mit ER-Diagramm.			
b)	Kardinalität und Optionalität sind im ER-Diagramm dargestellt. Diese sind sprachlich zu erläutern. Z. B. ein Fahrer kann mehrere Einträge haben, daher Kardinalität n, muss aber keinen Eintrag haben, daher Kann-Beziehung. Ein Eintrag gehört zu genau einem Fahrer, daher Kardinalität 1 und Muss-Beziehung. Fahrer(PersonalausweisNr, Nachname, Vorname, StrasseNr, PLZOrt, ↑FührerscheinNr) Führerschein(FührerscheinNr, Klasse, Datum, Probezeit, Fahrverbot) Eintrag(EintragNr, Punkte, Art, Datum, ↑PersonalausweisNr, ↑BehördeNr) Behörde(BehördeNr, Name, StrasseNr, PLZOrt, Telefon, Telefax) (Primärschlüssel sind unterstrichen, Fremdschlüsseln ist ein ↑ vorangestellt.) Die zweijährige Probezeit und das Fahrverbot können durch Angabe des Ende-Datums erfasst werden. Verwendung bekannter Regeln und Begründungen bei der Bewältigung neuer Fragestellungen Übersetzen eines ER-Diagramms in Relationen	6	8	
c)	c1) Es handelt sich um einen Inner-Join, dabei werden die Tabellen Fahrer und Führerschein über das Attribut FührerscheinNr zusammen geführt. Zudem erfolgen eine Selektion über die Führerscheinklasse B und eine Projektion auf Name, Vorname und Klasse, d. h. die Liste aller Personen mit Führerscheinklasse B wird erstellt. c2) Es handelt sich um eine geschachtelte Select-Abfrage. Die Unterabfrage ermittelt die maximale Anzahl der Punkte eines Eintrags, die äußere Select-Anweisung führt einen Inner-Join der Tabellen Fahrer und Eintrag über die Personalausweisnummer aus. Insgesamt wird eine Liste der Fahrer ausgegeben, die bei einem Eintrag die maximale Anzahl von Punkten haben. Semantische Analyse gegebener Algorithmen		6	
d)	d1) `SELECT Telefon FROM Behörde` `WHERE Name = "Ordnungsamt Wiesbaden"`	4	4	4

	Lösungsskizze	I	II	III

	d2) `SELECT COUNT(*) FROM Eintrag GROUP BY Art`
	d3) `SELECT Nachname, Vorname, SUM(Punkte)`
	`FROM Fahrer, Eintrag`
	`WHERE Fahrer.Personalausweis = Eintrag.Personalausweis`
	`GROUP BY Nachname, Vorname`
	`HAVING SUM(Punkte) > 20`
	planmäßiges Verarbeiten einfacher und komplexer Gegebenheiten			

e)	Die Fragestellung vermischt datentechnische mit Anwendungsfragen, setzt die modellierte Datenbank in einen soziotechnischen Zusammenhang (Verkehrszentralregister, Versicherung, Polizei, Gericht). Die Modellierung müsste erst um Fahrzeugdaten erweitert werden, um dies technisch zu ermöglichen. Aus Datenschutzgründen kann eine solche Abfrage nicht erlaubt werden, sonst könnte jeder auch ohne berechtigtes Interesse über Auto-Kennzeichen Halterdaten ermitteln. Beschreibung des Weges vom Kfz-Kennzeichen bis zum Eintrag Straftat in das Verkehrszentralregister. Analysieren eines Fallbeispiels Reflektierte Stellungnahme in Bezug auf Möglichkeiten Adäquatheit und Grenzen des Einsatzes von Informatiksystemen		4	4
	Insgesamt 56 BWE	18	30	8

1.1.7 ADT Dictionary *LF*

In verschiedenen Programmiersprachen gibt es den abstrakten Datentyp (ADT) *Dictionary*, bei dem man Zeichenketten als Index benutzen kann. Da ein Index normalerweise eine natürliche Zahl ist, spricht man beim Dictionary vom *Schlüssel* statt vom Index. Als Beispiel betrachten wir ein Dictionary *vokabeln*. Mit `vokabeln["lesson"] = "Unterrichtsstunde"` könnte unter dem Schlüssel `lesson` der Wert `Unterrichtsstunde` eingetragen werden und mit der Prüfung `eingabe.equals(vokabeln["teacher"])` könnte geprüft werden, ob Eingabe die richtige Übersetzung von `teacher` ist. Eine Liste mit Telefonnummer könnte auch ganz einfach mit einem Dictionary verwaltet werden, wobei die Namen die Schlüssel und die Telefonnummern die Werte wären. In einem Dictionary werden also Paare aus Schlüssel und Wert gespeichert. Um auch in Java diesen praktischen Datentyp nutzen zu können, entwickeln wir den ADT *Dictionary*.

a) Nennen Sie vier Operationen für den ADT Dictionary und geben Sie jeweils an, was die Operation bewirkt, welche Daten zur Ausführung der Operation benötigt und welche geliefert werden.

b) Der ADT Dictionary soll Elemente aufnehmen, die aus den zwei Attributen *Schlüssel* und *Wert* mit dem Datentyp String bestehen. Schreiben Sie eine Java-Klasse *DictionaryElement*, die einen Konstruktor zur Erzeugung eines Dictionary-Elements enthält.

c) Zur Implementierung des ADT Dictionary leiten wir *Dictionary* von der Java-Klasse *Vector* ab, deren Klassenbeschreibung als Anlage beigefügt ist. Entwickeln Sie eine Klassenbeschreibung für *Dictionary* unter Beachtung des Vererbungskonzepts und der unter a) angegebenen Operationen. Die Operationen müssen hierbei nicht ausprogrammiert werden, es reichen die Methodenköpfe. Beschreiben Sie die Funktionsweise folgender Dictionary-Methode und erläutern Sie deren Nutzen.

```
01 private int gibIndex(String schluessel) {
02   DictionaryElement element;
03   int i = 0;
04   while ( i < size() ) {
05     element = (DictionaryElement) get(i);
06     if ( element.schluessel.equals(schluessel) )
07       return i;
08     i++;
09   }
10   return -1;
11 }
```

d) Implementieren Sie
   - eine Methode zum Einfügen eines Schlüssel-Wert-Paares in ein Dictionary. Falls der Schlüssel schon existiert wird nur der zugehörige Wert geändert.
   - eine Methode, die in ein neues Dictionary erzeugt. Das neue Dictionary soll alle Schlüssel-Wert-Paare zu einem als Parameter angegebenen Wert enthalten.

e) Die Dictionary-Implementierung auf der Basis der Vector-Klasse ist nicht sehr effizient. Programmieren Sie eine Implementierung auf der Basis eines binären Suchbaums, mit mindestens den Methoden *Dictionary.gibWert* und *BinBaum.suchen*. Vergleichen Sie die beiden Implementierungen.

```
public class BinBaum {
  BinKnoten wurzel;    // die Wurzel des binären Baumes
  public BinBaum() {
    wurzel = null;
  }
}
```

Anmerkungen
Eine Variante dieser Aufgabe für das Grundkursfach ist in Abschnitt 1.3.2 beschrieben[1].

Zielsetzung:
Die Aufgabe fordert die Modellierung des abstrakten Datentyps Dictionary. Die teilweise Implementierung des abstrakten Datentyps soll auf der Basis der Klassen Vector und binärer Suchbaum erfolgen und miteinander verglichen werden. Es werden analytische und konstruktive Anforderungen gestellt sowie Kenntnis und Anwendung vorgegebenen Klassen durch Ableitung gefordert.

Unterrichtliche Voraussetzungen:
Im Unterricht wurden die abstrakten Datentypen lineare Liste, Keller, Schlange, Vector und binärer Suchbaum behandelt. Am Beispiel der linearen Liste, Keller und Suchbaum haben die Schüler die Vererbung und Klassenhierarchie kennen gelernt und angewendet. Mit dynamischen Datenstrukturen, Standardalgorithmen und dem Umgang mit Operationen haben die Schüler mehrfach im Unterricht Erfahrung gesammelt.

Zusätzliche Hilfsmittel: Kurzbeschreibung der Klassen Vector und String

Vorgesehene Bearbeitungszeit: 70 min

[1] Die Aufgabe kann unter Weglassung der Teilaufgabe e) auch im abgestuften Leistungsfach gestellt werden. Dabei ist die zweite Teilaufgabe von d) dann dem Anforderungsbereich III zuzuordnen.

Lösungsskizze, vorgesehene Bewertungseinheiten und ihre Zuordnung zu den Anforderungsbereichen I, II und II:

	Lösungsskizze	I	II	III
a)	Vier Operationen auf einem Dictionary sind durch genaue Angabe der Bedeutung anzugeben, z. B. - erzeugen – Erzeugen eines neuen Dictionary - einfügen – Einfügen eines Elements (Schlüssel, Wert) in ein Dictionary - löschen – Löschen eines durch den Schlüssel zu identifizierenden Elements - gibWert – Liefert den Wert zum Schlüssel - istEnthalten – Prüft, ob ein Schlüssel im Dictionary vorkommt Verwendung geübter Arbeitstechniken in einem einfachen Beispiel	8		
b)	Für die Elemente ist eine eigene, einfache Klasse nötig, die neben Schlüssel und Wert noch einen Konstruktor enthält. `class DictionaryElement { String schluessel; String wert; public DictionaryElement (String schluessel, String wert) { this.schluessel = schluessel; this.wert = wert; } }` Verwendung einer gelernten und geübten Arbeitstechnik in einem wiederholenden Zusammenhang	8		
c)	`public class Dictionary extends Vector { Dictionary () {...} public void einfuegen(String schluessel, String wert) {...} public boolean loeschen(String schluessel) {...} public String gibWert(String schluessel) {...} public boolean istEnthalten(String schluessel) {...} private int gibIndex(String schluessel) {...} }` Insbesondere sind die Nutzung geerbter Methoden (size und get), der Typcast und der Aufruf der String-Methode equals zu erläutern. Die interne Methode gibIndex liefert den Index, an dem das Dictionary-Element mit dem gegebenen Schlüssel im Vector gespeichert ist. Kommt der Schlüssel nicht vor, wird –1 zurückgegeben. Sie kann gut zur Implementierung der öffentlichen Methoden genutzt werden. Modellierung eines Problems auf Grundlage bekannter Verfahren Semantische Analyse eines gegebenen Algorithmus Nutzung vorhandener Programmbibliotheken für die eigene Problemlösung		14	

d)	Implementierung auf der Basis von c) und der geerbten Methoden set und add.
	`public void insert(String schluessel, String wert) { DictionaryElement element = new DictionaryElement(schluessel, wert); int i = gibIndex(schluessel); if (i >= 0) set(i, element); else add(element); }`
	Anwendung des Dictionary
	`public Dictionary wertDictionary(String wert) { DictionaryElement element; Dictionary myDictionary = new Dictionary();`
			12	

	Lösungsskizze	I	II	III

	``
	  for (int i = 0 ; i < size() ; i++) {
	    element = (DictionaryElement) get(i);
	    if (element.wert.equals(wert))
	      myDictionary.add(element);
	  }
	  return myDictionary;
	}
	``
	Standardalgorithmen auf dynamischen Datenstrukturen
	Selbständiges Übertragen des Gelernten auf vergleichbare Situationen			

e)	Die Formulierung ist bewusst knapp gehalten, um die Anforderungen an die Modellierung zu erschweren. Drei neue Klassen, Vererbung und drei neue Methoden werden gebraucht. Suchen operiert auf der Zeigerstruktur des binären Suchbaums.
	``
	public class Dictionary extends BinBaum {
	  public String gibWert(String schluessel) {
	    BinKnoten myKnoten;
	
	    myKnoten = suchen(schluessel);
	    if (myKnoten == null)
	      return "";
	    else
	      return myKnoten.wert;
	  }
	}
	class BinKnoten {
	  String schluessel;
	  String wert;
	  BinKnoten links, rechts; // linker und rechter Nachfolger
	
	  public BinKnoten (String schluessel, String wert) {
	    this.schluessel = schluessel;
	    this.wert = wert;
	    links = null;
	    rechts = null;
	  }
	}
	class BinBaum {
	  BinKnoten wurzel; // die Wurzel des binären Baumes
	
	  public BinBaum() { // Konstruktor
	    wurzel = null;
	  }
	
	  public BinKnoten suchen(String schluessel) {
	    BinKnoten myKnoten = wurzel;
	    while (myKnoten != null)
	      if (myKnoten.schluessel.equals(schluessel))
	        return myKnoten;
	      else
	        if (myKnoten.schluessel.compareTo(schluessel) > 0)
	          myKnoten = myKnoten.links;
	        else
	          myKnoten = myKnoten.rechts;
	    return null;
	  }
	}
	``
	Vergleich der Datenstrukturen hinsichtlich Effizienz: bei der Implementierung auf der Basis des Datentyps Vector haben alle Operationen lineare, beim binären Suchbaum logarithmische Zeitkomplexität.
	Durchführung einer komplexeren Problemanalyse
				12

	Lösungsskizze	I	II	III
	Zerlegung eines gegebenen anspruchsvollen Problems in geeignete Teilprobleme und deren Lösung			
	Insgesamt 54 BWE	16	26	12

1.2 Ausführlich kommentierte Beispiele für das Grundkursfach

1.2.1 Schneller Datenzugriff *GF*

Diese Aufgabe ist mit dem PC zu lösen.

Zum schnellen Bereitstellen von Datensätzen zur weiteren Bearbeitung soll ein zweistufiges Speichersystem verwendet werden, das aus einem großen langsamen Speicher (GLS) und einem kleinen schnellen Speicher (KSS) besteht. Der Speicher GLS enthält 1000 Datensätze. Die Datensätze besitzen die Nummern von 1 bis 1000. Der Speicher KSS kann fünf Datensätze speichern. Das Anfordern eines Datensatzes bewirkt in dem Speichersystem die folgenden Vorgänge: Der angeforderte Datensatz wird aus dem Speicher GLS in den Speicher KSS kopiert, falls er in dem Speicher KSS noch nicht enthalten ist. Soll ein Datensatz in den Speicher KSS kopiert werden und dieser ist voll belegt, so wird zuerst ein Datensatz im Speicher KSS gelöscht. Dann erfolgt das Kopieren. Das Bereitstellen eines Datensatzes erfolgt stets aus dem Speicher KSS. Für das Löschen eines Datensatzes sind verschiedene Strategien anwendbar. Eine Strategie ist das Löschen des Datensatzes, der sich die längste Zeit im Speicher KSS befindet.

a) Der Speicher KSS ist zu Beginn leer. Dann werden Datensätze in der folgenden Reihenfolge angefordert: 250, 645, 14, 250, 800, 170, 170, 300, 800, 250. Geben Sie für dieses Beispiel die Vorgänge an, die von dem Speichersystem bei Anwendung der dargestellten Strategie zum Löschen ausgeführt werden. Gehen Sie dabei auch auf die Verwaltung der Datensätze im Speicher KSS ein.

b) Geben Sie eine andere Strategie zum Löschen eines Datensatzes an und erläutern Sie an einem selbst gewählten Beispiel die Arbeitsweise des Speichersystems bei Anwendung dieser Strategie.

c) Entwerfen und implementieren Sie ein Programm, das das zweistufige Speichersystem unter Anwendung der Strategie von Teilaufgabe a) oder der Strategie von Teilaufgabe b) realisiert. Das Programm soll die angeforderten Datensätze einlesen und die Vorgänge, die von dem Speichersystem ausgeführt werden, protokollieren.

d) Erläutern Sie die Methoden der Softwareentwicklung, die von Ihnen in Teilaufgabe c) verwendet wurden.

Anmerkungen

Zielsetzung:
Der Prüfling soll nachweisen, dass er eine komplexe Aufgabe, die Bezüge zu technischen Aspekten der Informatik besitzt, unter praktischem Einsatz eines Informatiksystems lösen kann. Er soll die beschriebenen Vorgänge in einen Softwareentwurf und weiter in ein Programm umsetzen können. Der Prüfling soll in der Lage sein, die von ihm angewandten Methoden der Softwareentwicklung anzugeben und zu erläutern.

Unterrichtliche Voraussetzungen:
Im Unterricht wurden Programme in einer höheren Programmiersprache unter Anwendung der Methoden der Softwareentwicklung erstellt. Dabei wurden die Phasen Entwurf, Implementierung und Reflexion unterschieden.

Zusätzliche Hilfsmittel: Für die Lösung dieser Aufgabe steht jedem Schüler ein Computer mit einer der Aufgabe entsprechenden Softwareausstattung zur Verfügung.

Vorgesehene Bearbeitungszeit: 100 min

Lösungsskizze, vorgesehene Bewertungseinheiten und ihre Zuordnung zu den Anforderungsbereichen I, II, III:

	Lösungsskizze	I	II	III
a)	An dem vorgegebenen Beispiel werden die Vorgänge Anfordern, Kopieren, Löschen und Bereitstellen eines Datensatzes erläutert. Zur Verwaltung der Datensätze im Speicher KSS gibt es mehrere Möglichkeiten. So kann beim Kopieren eines Datensatzes aus dem Speicher GLS in den Speicher KSS der Zeitpunkt des Kopiervorgangs gespeichert werden oder der Speicher KSS wird als	4	4	

	Warteschlange organisiert.			
b)	Andere Strategien sind zum Beispiel das Löschen des Datensatzes, der die längste Zeit nicht angefordert wurde oder des Datensatzes, der in einer bestimmten Zeit die geringste Anzahl an Anforderungen hatte. Eine Strategie wird genannt und erläutert.		8	4
c)	Ein Programm wird entworfen und implementiert.		20	10
d)	Die verwendeten Methoden werden erläutert (z. B. strukturiertes Programmieren, modulares Programmieren, Top-down-Strategie).	6	6	

1.2.2 Kryptographie *GF*

Lesen Sie den Auszug aus der Pressemitteilung des Landesbeauftragten für den Datenschutz Schleswig-Holstein (siehe Anlage).

a) Beschreiben Sie die Funktionsweise eines Kryptosystems mit öffentlichen und privaten Schlüsseln am Beispiel einer E-Mail eines Bürgers an den Datenschutzbeauftragten und der Antwort des Datenschutzbeauftragten.

b) In der Pressemitteilung steht, dass die Verschlüsselung zu übermittelnder Daten in Deutschland gesetzlich nicht beschränkt ist. Stellen Sie Argumente für und gegen eine gesetzliche Beschränkung von Verschlüsselung gegenüber. Welche Position vertreten Sie selbst? Begründen Sie.

c) Erläutern Sie die Begriffe Vertraulichkeit und Authentizität im Rahmen einer sicheren Kommunikation. Was müssen der Bürger und der Datenschutzbeauftragte tun, um die Authentizität sicherzustellen?

d) Wie schätzen Sie die Verbreitung von Verschlüsselungsverfahren, die er mit dieser Mitteilung 1997 fördern wollte, heute im privaten, geschäftlichen und öffentlichen Bereich ein? Geben Sie Gründe für die Verwendung bzw. Nichtverwendung an.

e) Beschreiben Sie je ein historisches Substitutions- und Transpositionsverfahren. Warum stellen historische Verfahren keine Grundlage für heutige Verschlüsselungsprogramme dar?

f) Das Verschlüsselungsprogramm PGP verwendet das RSA-Verfahren. Worauf beruht die Sicherheit des RSA-Verfahrens?

g) Zeigen sie beispielhaft anhand der Zahlen p = 13 und 1 = 17 die Schlüsselerzeugung von n, e und d, wobei e = 7 sein soll. Verschlüsseln Sie die Nachricht m = 5.

Anlage

Der Landesbeauftragte für den Datenschutz Schleswig-Holstein

28. Juli 1997

PRESSEMITTEILUNG

Schleswig-Holsteinischer Datenschutzbeauftragter nimmt ab sofort E-Mails auch verschlüsselt entgegen

Wer in offenen Netzen wie z. B. dem Internet Informationen austauschen will, kann sich gegen Mithören und Mitlesen durch unbefugte Dritte wirksam nur durch die Verschlüsselung der zu übermittelnden Daten schützen. Dies ist in Deutschland gesetzlich in keiner Weise beschränkt. Deshalb empfiehlt der Schleswig-Holsteinische Datenschutzbeauftragte, hiervon Gebrauch zu machen, wann immer es möglich ist.

Da sich Bürger immer häufiger auch per E-Mail an ihn wenden, hat er ab sofort die Voraussetzungen dafür geschaffen, dass dies in verschlüsselter Form möglich ist. Dabei wird das Verschlüsselungsprogramm „Pretty Good Privacy“ (PGP) angewandt, das nach heutigem Kenntnisstand eine gute Sicherheit bietet und kostenlos aus dem Internet entnommen werden kann.

Das Verfahren funktioniert so: Der Datenschutzbeauftragte gibt jedem, der Interesse daran hat, seinen „öffentlichen Schlüssel“ bekannt. Wer mit ihm vertraulich kommunizieren will, kann seinen Text vor der Absendung mit diesem „public key“ verschlüsseln. Die Entschlüsselung ist nur dem Datenschutzbeauftragten mit seinem nur ihm bekannten „private key“ möglich.

Der Datenschutzbeauftragte will auf diesem Wege die Verbreitung von Verschlüsselungsverfahren fördern und selbst Erfahrungen beim Umgang mit der Verschlüsselungstechnik sammeln.

Anmerkungen
Eine Variante dieser Aufgabe für das Leistungsfach ist in Abschnitt 1.3.2 beschrieben.

Unterrichtliche Voraussetzungen:
Im Unterricht wurden historische Substitutions- und Transpositionsverfahren, symmetrische und asymmetrische Verschlüsselungsverfahren behandelt. Bezüglich des RSA-Verfahrens sind die Algorithmen zur Schlüsselerzeugung, zum Ver- und Entschlüsseln von Nachrichten und zur Authentifizierung bekannt und geübt. Die Prüflinge kennen Verfahren zur Kryptoanalyse der historischen Verfahren und die Komplexität des Faktorisierungsproblems. Aktuelle Einsatzbereiche von Kryptographie sind im Unterricht behandelt und diskutiert worden.

Zusätzliche Hilfsmittel: Taschenrechner

Vorgesehene Bearbeitungszeit: 80 min

Lösungsskizze, vorgesehene Bewertungseinheiten und ihre Zuordnung zu den Anforderungsbereichen I, II, III:

	Lösungsskizze	I	II	III
a)	Ein Bürger verschlüsselt seine Nachricht an den Datenschutzbeauftragten mit dessen öffentlichen Schlüssel. Die verschlüsselte Nachricht kann nur mit dem zugehörigen privaten Schlüssel, den der rechtmäßige Empfänger besitzt, entschlüsselt werden. Diese Nachricht sollte auch den öffentlichen Schlüssel des Absenders enthalten, damit der Datenschutzbeauftragte ebenfalls seine Antwort an den Bürger verschlüsseln kann und nur der gewünschte Empfänger sie entschlüsseln kann.	8		
b)	Als Argument für eine gesetzliche Beschränkung wird oft die dem Staat prinzipiell zu ermöglichende Entschlüsselung im Rahmen der Kriminalitätsbekämpfung genannt. Gegen diese Beschränkung spricht das Grundrecht des Brief-, Post- und Fernmeldegeheimnisses. Formulierung und Begründung einer eigenen Position		5	2
c)	„Vertraulichkeit“ bedeutet, dass nur die Personen, an die eine Nachricht gerichtet ist, diese Nachricht lesen können. „Authentizität“ bedeutet die Echtheit, Glaubwürdigkeit einer Nachricht. Die Authentizität einer Nachricht wird dadurch sichergestellt, dass der Absender seine Nachricht zusätzlich mit seinem privaten Schlüssel signiert. Der Empfänger kann dann durch Anwendung des öffentlichen Schlüssels des Absenders verifizieren, ob die Nachricht wirklich von ihm stammt.	4	4	
d)	Erwartet werden Aussagen zu Schutzbedürftigkeit, Verfügbarkeit und Aufwand. Im privaten Bereich werden E-Mails in der Regel aus Bequemlichkeitsgründen und geringerer Schutzbedürftigkeit nicht verschlüsselt. Im geschäftlichen Bereich werben beispielsweise Banken im Rahmen des Online-Bankings mit sicheren Verschlüsselungen. Im öffentlichen Bereich ist die Authentizität besonders wichtig.		4	2
e)	Die Caesar-Verschlüsselung ist ein Substitutionsverfahren: jeder Buchstabe des Klartextes wird durch einen im Alphabet im festen Abstand folgenden Buchstaben ersetzt. Schon bei relativ kurzen Texten kann mit Hilfe der Buchstabenhäufigkeit zunächst der Buchstabe „E“	6	6	

	Lösungsskizze	I	II	III
	und damit auch alle anderen entschlüsselt werden. Die Skytale von Sparta ist ein Transpositionsverfahren. Die Buchstaben des Klartextes bleiben erhalten und werden nur in einer anderen Reihenfolge angeordnet. Durch Aufspüren der häufigen Zweierkombinationen von Buchstaben kann die Umordnung der Buchstaben ermittelt werden.			
f)	Die Schlüssel werden als Produkt von sehr großen Primzahlen gebildet. Die Umkehrung, nämlich die Faktorisierung des Produktes, ist praktisch unmöglich.	3		
g)	Es ist $n = p \cdot q = 13 \cdot 17 = 221$ und $(p - 1)(q - 1) = 12 \cdot 16 = 192$. Wegen $1 = 55 \cdot 7 - 2 \cdot 192$ ist dann $d = 55$. Verschlüsselung: Wegen $5^7 = 78125 = 353 \cdot 221 + 112$ wird 5 zu 112 verschlüsselt.		6	
	Insgesamt 50 BWE	21	25	4

1.2.3 Möglichkeiten und Grenzen *GF*

a) Beurteilen Sie die folgende These:

Jedes Problem, das sich präzise beschreiben lässt, kann mit einem Computer gelöst werden.

b) An Ihrer Schule findet ein Forum zum Thema „Informatik und Gesellschaft“ statt. Sie haben die Aufgabe erhalten, das Forum mit einem Kurzvortrag zu eröffnen. In dem Kurzvortrag sollen Sie Chancen und Möglichkeiten, aber auch Gefahren und Risiken moderner Kommunikations- und Informationstechniken an einem konkreten Beispiel darstellen.

Geben Sie für den Kurzvortrag eine Gliederung mit geeigneten Schwerpunkten an.

c) Ein Automat bohrt Löcher in Leiterplatten. Der horizontale Gesamtweg des Bohrkopfes bei der Bearbeitung einer Leiterplatte soll möglichst kurz sein.

Entwerfen Sie einen Algorithmus, der einen möglichst kurzen Gesamtweg des Bohrkopfes findet.

d) In einer Firma sollen Leiterplatten mit 2000 Löchern hergestellt werden.
Ermitteln Sie, ob sich der von Ihnen in Teilaufgabe c) entworfene Algorithmus dafür eignet.

Anmerkungen

Zielsetzung:
Der Prüfling soll zeigen, dass er für konkrete Situationen angeben kann, ob der Einsatz eines Computers machbar und sinnvoll ist. In die Bearbeitung werden der theoretische Aspekt, der praktische Aspekt und der ökonomisch-soziale Aspekt einbezogen. Die Aufgaben ermöglichen verschiedene Lösungswege. Der Prüfling kann bei der Bearbeitung Bezüge zu anderen Unterrichtsfächern herstellen.

Unterrichtliche Voraussetzungen:
Aus dem Unterricht ist bekannt, dass es Probleme gibt, die prinzipiell mit einem Computer unlösbar sind oder bei denen bei bekanntem Lösungsalgorithmus die verfügbaren Ressourcen zur Problemlösung nicht ausreichen und dass zahlreiche dieser Probleme hohe praktische Relevanz besitzen. Im Unterricht wurden Algorithmen entworfen und die Zeitkomplexität von Algorithmen abgeschätzt. Der Entwurf wurde in Beschreibungs-, Strukturierungs- und algorithmische Phase gegliedert. Die Auswirkungen des Computereinsatzes in mehreren gesellschaftlichen Bereichen wurde diskutiert.

Zusätzliche Hilfsmittel: Taschenrechner

Vorgesehene Bearbeitungszeit: 75 min

Lösungsskizze, vorgesehene Bewertungseinheiten und ihre Zuordnung zu den Anforderungsbereichen I, II, III:

	Lösungsskizze	I	II	III
a)	Die These ist nicht gültig. Für die Berechenbarkeit gibt es prinzipielle (vgl. Halteproblem) und praktische Grenzen (Zeit oder Speicher reichen zur Problemlösung nicht aus).	10		
b)	Das Beispiel kann aus dem Bereich des Einsatzes moderner Software in einer Firma stammen (z. B. Textverarbeitung, Datenbanken, CAD-System im Maschinenbau). Für das Beispiel „Computereinsatz in der Medizin“ ist eine mögliche Gliederung, die durch entsprechende Schwerpunkte zu untersetzen ist: • Verwaltung von Patientendaten in einer Arztpraxis und in einem Krankenhaus, Einsatz eines Patientenchips, • Computer als Hilfsmittel beim Erstellen einer Diagnose und beim Unterbreiten eines Therapievorschlags (Expertensysteme, Computertomographie), • Operationen mit Computerunterstützung und • Experimente mit Hilfe des Computers anstelle von Tierversuchen (Problematik eines Modells, ethische Fragen).	8	8	
c)	Für die Problemlösung kann z. B. ein einfacher Algorithmus (es wird stets das nächste Bohrloch aufgesucht) oder ein Backtracking-Algorithmus, der die exakte Lösung ermittelt, entworfen werden. Die Algorithmen operieren auf Datenstrukturen, die anzugeben sind (z. B. Reihung oder Liste).		10	6
d)	Die Zeitkomplexität des Algorithmus wird grob abgeschätzt. Dabei zeigt sich, dass der einfache Algorithmus auch bei 2000 Bohrlöchern einsetzbar ist. Ein Backtracking-Algorithmus ist nicht nutzbar (der Suchbaum hätte die Tiefe von 2000).		4	4
	Insgesamt 50 BWE	18	22	10

1.2.4 Netzwerke *GF*

Eine Firma mit Hauptsitz in der Nähe von Freudenstadt entwickelt, produziert und vertreibt Verankerungssysteme (Dübel). Für den Vertrieb über den Großhandel wird Deutschland in 15 Gebiete aufgeteilt, für die jeweils ein Außendienstmitarbeiter zuständig ist. Diese besuchen den Großhandel und nehmen dort Bestellungen auf. Außerdem besuchen die Außendienstmitarbeiter auch Endkunden auf Baustellen und beraten über den Einsatz von meist größer dimensionierten Verankerungssystemen. Besonders für das zweite Aufgabenfeld ist eine Verbindung zur Forschungs- und Entwicklungsabteilung des Stammhauses erforderlich. Als IP-Adresskreis wurde der Firma das Klasse-C-Netz mit der Adresse 194.168.3.0 zugeteilt.

a) Hardware
Als Berater sollen Sie die Außendienstmitarbeiter der Firma hardwaretechnisch so ausstatten, dass diese von unterwegs Zugriff auf die zentrale Unternehmensdatenbank und die Wissensdatenbank der Forschungs- und Entwicklungsabteilung der Firma haben.

b) Teilnetze
Weiterhin sollen Sie das oben genannte Klasse-C-Netz so unterteilen, dass die Mitarbeiter der Abteilungen Einkauf, Arbeitsvorbereitung, Logistik, Vertrieb sowie die Außendienst-Mitarbeiter jeweils in eigenen Teilnetzen arbeiten können.

c) VPN-Protokolle
Damit die Datenübertragung nicht von der Konkurrenz abgehört werden kann, entscheidet sich die Geschäftsleitung für ein VPN-System (virtuelles privates Netzwerk). Sie sollen sich um die Verschlüsselungsmethode, die Authentifizierung und die Tunnel-Protokolle kümmern. Beschreiben Sie

eine Methode der Verschlüsselung und nennen Sie Möglichkeiten der Authentifizierung. Welche Tunnel-Protokolle könnten verwendet werden? Nennen Sie Bewertungskriterien für die Protokolle.

d) Firewall
Als weitere Sicherheitsmaßnahme werden Firewalls verwendet. Beschreiben Sie, was mit Firewalls erreicht werden soll.

Anmerkungen

Zusätzliche Hilfsmittel: keine

Vorgesehene Bearbeitungszeit: 45 min

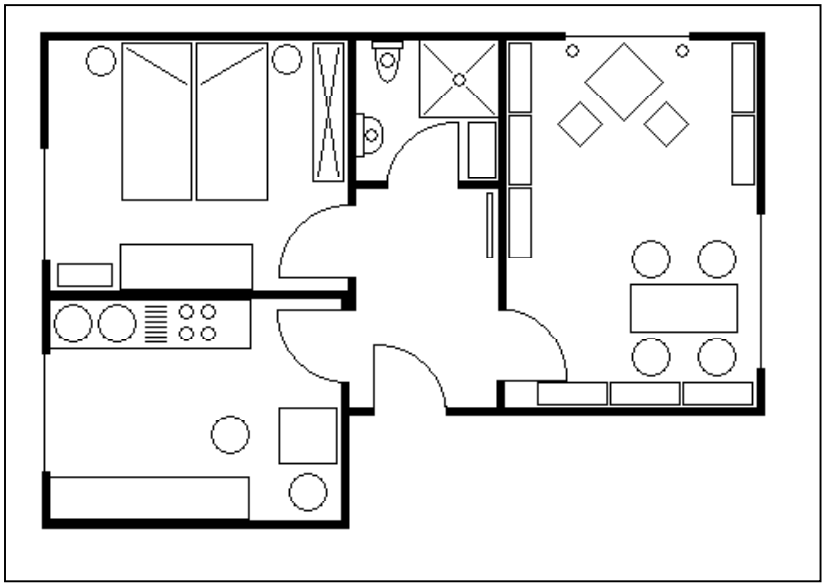
Lösungsskizze, vorgesehene Bewertungseinheiten und ihre Zuordnung zu den Anforderungsbereichen I, II, III:

	Lösungsskizze	I	II	III
a)	Hardware: Die Laptops der Mitarbeiter benötigen sowohl ein (DSL) Modem + Handy (evtl. integriert) als auch geschickterweise eine Funk-LAN-Karte nach IEEE 802.11 zur Nutzung von Funk-LANs und hot spots.	2	4	
b)	Mögliche Teilnetze sind: 194.168.3.0 – 194.168.3.31 194.168.3.32 – 194.168.3.63 194.168.3.64 – 194.168.3.95 194.168.3.96 – 194.168.3.127 194.168.3.128 – 194.168.3.159 194.168.3.160 – 194.168.3.191 194.168.3.192 – 194.168.3.223 194.168.3.224 – 194.168.3.255 Bei dieser Aufteilung sind drei Bereiche frei, die für neue oder zu vergrößernde Abteilungen benutzt werden können. Diese Lösung sieht für jeden Bereich 29 IP-Adressen vor. Sollten einzelne Abteilungen einen größeren Adressbedarf haben, kann man Teilnetze zu einem größeren Teilnetz zusammenfassen.		6	4
c)	VPN Protokolle *Verschlüsselung* z. B. asymmetrisch mit PGP (Pretty Good Privacy) Es wird ein privater und ein öffentlicher Schlüssel verwendet. *Authentifizierung* mit Kerberos, EAP (Extensible Authentication-Protokoll), Smartcard, ... Das EAP und die strengen EAP-Authentifizierungsmethoden sind eine wichtige Technologiekomponente für sichere Verbindungen über VPN (virtuelle private Netzwerke), da es mehr Sicherheit gegen Hacker- und Wörterbuchangriffe sowie gegen das Erraten von Kennwörtern bietet als andere Authentifizierungsmethoden wie beispielsweise CHAP. *Tunneling Protokolle:* IPSec (Internet Protokoll Security) PPTP (Point to Point Tunneling Protokoll) L2TP (Layer Two Tunneling Protokoll) *Bewertungskriterien:* Quellcode offen oder proprietär	2	5	3

	Lösungsskizze	I	II	III
	Verschlüsselungstiefe Welche Betriebssysteme werden unterstützt?			
d)	Firewall: Mit einer hardwaretechnisch oder softwaretechnisch ausgeführten Firewall sollen offene Ports (z. B. http, E-Mail, ...) beschränkt, die Paketarten (z. B. ICMP) kontrolliert und die Protokolle (TCP, UDP) überwacht werden.	2	2	
	Insgesamt 30 BWE	6	17	7

1.2.5 Objektorientierte Modellierung eines Grafiksystems *GF*

Eine Wohnung mit Innenausstattung soll mit einem universellen Grafiksystem entworfen werden. Als Beispiel sehen Sie einen fertigen Grundriss mit Möblierung.

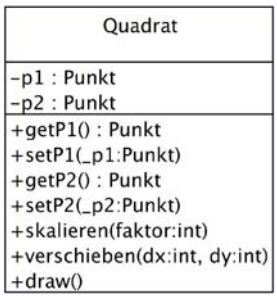


a) Erläutern Sie, welche geometrischen Elemente das Grafiksystem darstellen können muss, um diese Zeichnung zu erstellen.

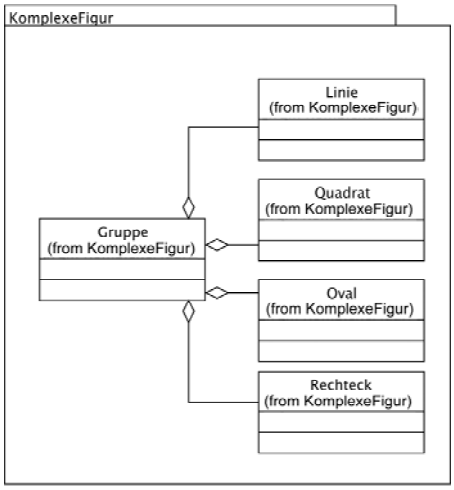
b) In der Anlage sind drei alternative Klassendiagramme angegeben, die jeweils die Beziehungen zwischen den Klassen Punkt, Linie und Rechteck darstellen. Vergleichen Sie die mit den Klassendiagrammen beschriebenen Modellierungen. Entscheiden Sie sich für ein Modell und begründen Sie Ihre Entscheidung.

c) Implementieren Sie für Modell_1 aus der Anlage die notwendigen Klassen. Dabei sollen von den Methoden nur Konstruktoren und Zugriffsmethoden auf die Attribute sowie die move-Methode implementiert werden.

d) Für die Erstellung der obigen Zeichnung sollen Quadrate auch gedreht werden können. Es liegt die dargestellte Klasse Quadrat vor. Diese Klasse Quadrat kann nicht modifiziert werden, da man keinen Zugriff auf den Quellcode hat. Es muss eine neue Klasse entwickelt werden, damit das Quadrat gedreht werden kann. Beschreiben Sie, wie Sie dieses Problem lösen würden.



e) Um zusammengesetzte Elemente, z. B. ein WC mit Wasserkasten, leichter erstellen zu können, müssen die geometrischen Formen gruppiert werden können. Zur Modellierung wird in einem ersten Versuch das nebenstehende Klassendiagramm entworfen. Beschreiben Sie die grundlegenden Probleme des Modells und schlagen Sie eine Verbesserung des Modells vor.



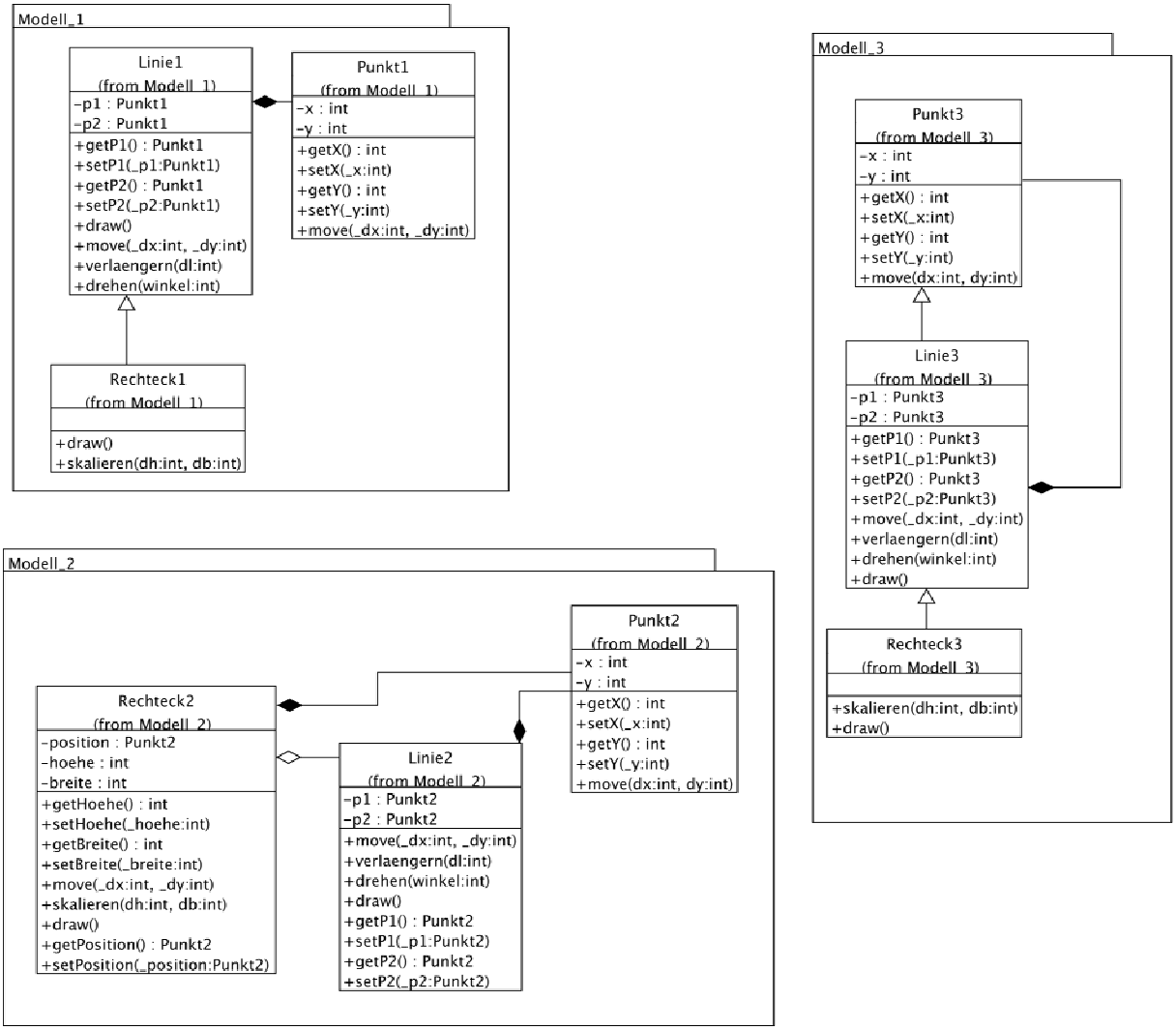
Anmerkungen

Unterrichtliche Voraussetzungen:
Im Unterricht sind Grundlagen objektorientierter Modellierung erarbeitet worden. Dazu gehörte u. a. die Erstellung von Klassen- und Sequenzdiagrammen (UML), die Eigenschaften verschiedenartiger Beziehungen zwischen Klassen sowie die Konzepte der Kapselung, der Vererbung und der Polymorphie. Modelle sind mit Hilfe einer geeigneten Programmiersprache implementiert und Klassenbibliotheken genutzt worden.

Zusätzliche Hilfsmittel: Dokumentationen von Klassenbibliotheken

Vorgesehene Bearbeitungszeit: 80 min

Anlage



Lösungsskizze, vorgesehene Bewertungseinheiten und ihre Zuordnung zu den Anforderungsbereichen I, II, III:

	Lösungsskizze	I	II	III
a)	Die Figuren werden z. B. erstellt, gezeichnet, verschoben, skaliert, gelöscht, gruppiert. Das Programm muss die Zeichnungen speichern und öffnen können. Objekte müssen markiert, Eigenschaften müssen geändert werden können. Elemente wie Linien unterschiedlicher Strichstärke, Quadrate, Rechtecke, Kreise, Ellipsen sind erforderlich.		5	
b)	Erwartet wird eine schlüssige Argumentation unter korrekter Verwendung der Fachsprache, beispielsweise wie folgt: Modell_1: Linie enthält Punkte, Rechteck erbt von Linie. Damit kann die Klasse `Rechteck` Verhalten von der Klasse `Linie` übernehmen. Damit kann man Quellcode sparen. Die Bedeutung der Punkte ist aber eine andere: in der Klasse `Linie` legen sie die Endpunkte fest, in der Klasse `Rechteck` wird die Diagonale aufgespannt. Die Methode `draw` muss deshalb überschrieben werden, die Methode `verlaengern` könnte man noch akzeptieren, aber das Drehen einer Linie hat für Linie und Rechteck eine andere Bedeutung. Bei dem gegebenen Modell lässt sich die Drehung des Rechtecks auch durch Überschreiben nicht realisieren. Es bliebe nur die Möglichkeit, die Methode zu blocken, indem man in der Klasse `Rechteck`		6	6

	eine Leermethode schreibt. Dies ist aber sehr schlechter Stil. Es wird hier eine Design-Regel verletzt: Vererbung ist nur dann zu wählen, wenn die Beziehung „ist ein Spezialfall von“ gilt. Modell_2: Dieses Modell arbeitet nur mit Delegation, damit werden negative Effekte, die durch die Vererbung entstehen, vermieden. Es muss zwar geringfügig mehr Programmcode geschrieben werden, dieser Nachteil fällt gegenüber dem strukturellen Fehler nicht ins Gewicht. Möchte man das Rechteck drehbar gestalten, muss man sich auf diese Klasse konzentrieren, da die Kopplung der Klassen gering ist. Dies gilt auch für Veränderungen der Klasse `Linie`. Modell_3: Für Modell_3 gilt im Wesentlichen das, was für Modell_1 gesagt wurde. Die zusätzliche Vererbung bringt keinen Vorteil.			
c)	Quellcode siehe Anhang.	10		
d)	Das Problem lässt sich mit Vererbung lösen. Es wird eine neue Klasse `Drehquadrat` abgeleitet von `Quadrat` mit einem zusätzlichen Attribut zum Speichern der Orientierung geschrieben. Man benötigt mindestens einen Konstruktor, der eine Anfangsorientierung setzt, eine Methode `getOrientierung` sowie eine Methode zum Setzen der Orientierung. Die Zeichenmethode `draw` muss überschrieben werden. Schwierig ist zu erkennen, dass es nicht sinnvoll ist, bei geänderter Orientierung die Koordinaten der Punkte p1 und p2 neu zu setzen.		5	5
e)	Bei dem vorgegebenen Modell müsste für jede Klasse grafischer Elemente eine eigene Liste erzeugt werden. Dies kann dadurch vermieden werden, dass eine übergeordnete, abstrakte Klasse `Shape` erstellt wird. Dann können alle geometrischen Figuren in eine einzige Liste vom Typ `Shape` abgelegt werden. Sind Entwurfsmuster bekannt, so ist das Iterator-Muster naheliegend.		2	5
	Insgesamt 44 BWE	10	18	16

Lösungen zu c): Anhang

```
package Modell1;

import java.util.*;

public class Punkt1 {

  private int x;
  private int y;

  public Punkt1(int _x, int _y)
  {
    x=_x;
    y=_y;
  }

  public int getX()
  {
    return x;
  }

  public void setX(int _x)
  {
    x = _x;
  }

  public int getY()
  {
    return y;
  }

  public void setY(int _y)
  {
    y = _y;
  }

  public void move(int _dx, int _dy)
  {
    x = x + _dx;
    y = y + _dy;
  }

} // end Punkt1


public class Linie1 {

  private Punkt1 p1;
  private Punkt1 p2;

  public Linie1(int x1, int y1,
                int x2, int y2)
  {
    p1 = new Punkt1(x1,y1);
    p2 = new Punkt1(x2,y2);
  }

  public Linie1(Punkt1 _p1, Punkt1 _p2)
  {
```

```
    p1 = _p1;
    p2 = _p2;
  }
  public Punkt1 getP1()
  {
    return p1;
  }

  public Punkt1 getP2()
  {
    return p2;
  }

  public void setP2(Punkt1 _p2)
  {
    p2 = _p2;
  }

  public void draw()
  {
    // Code muss ergaenzt werden
  }

  public void move(Punkt1 dist)
  {
    p1.move(dist.getX(),dist.getY());
    p2.move(dist.getX(),dist.getY());
  }
```

```
  public void verlaengern(int dl)
  {
    // Code muss ergaenzt werden
  }

  public void drehen(int winkel)
  {
    // Code muss ergaenzt werden
  }

} // end Linie1

public class Rechteck1 extends Linie1 {
  public Rechteck1(Punkt1 _p1,
                   int _h, int _b)
  {
    super(_p1,new Punkt1(_h,_b));
  }

  public Rechteck1(Punkt1 _p1,
                   Punkt1 _p2)
  {
    super(_p1,_p2);
  }

  public void draw()
  {
    // Code muss ergaenzt werden
  }

  public void skalieren(int dh, int db)
  {
    // Code muss ergaenzt werden
  }
} // end Rechteck1
```

1.2.6 Zeitsynchronisation *GF*

Nach der Einführung der Eisenbahnen und der elektrischen Telegraphen musste ein Problem gelöst werden, das vordem keine große Bedeutung hatte: das Fehlen einer einheitlichen Zeit. Bis dahin hatte jeder Ort seine Ortszeit, Zonenzeiten gab es nicht.

Mit der Vernetzung von Informatiksystemen tritt ein ähnliches Problem auch in Bezug auf das Internet auf. Bereits zu Beginn der Entwicklung des Internets wurde das Problem erkannt und mit dem RFC 868 (siehe Anlage) ein Vorschlag unterbreitet, wie dieses Problem gelöst werden könnte.

a) Wozu werden in vernetzten Umgebungen abgestimmte Zeiten benötigt? Welche Gründe sind ursächlich für die Zeitdifferenzen in Informatiksystemen verantwortlich?

b) Welche Lösungsmöglichkeit wird in dem RFC vorgeschlagen? Stellen Sie die Kommunikation zwischen Server und Klient in geeigneter Weise grafisch dar.

c) Erläutern Sie die Unterschiede zwischen dem UDP- und dem TCP-Protokoll. Geben Sie typische Anwendungsfälle mit der entsprechenden Protokollauswahl an und stellen Sie Vor- und Nachteile der jeweiligen Wahl zusammen.

d) Beschreiben Sie die Vor- und Nachteile verbindungsloser Protokolle am Beispiel der Internettelefonie.

e) Analysieren Sie das folgende undokumentierte Python-Programm und beschreiben Sie seine Funktionalität unter Bezugnahme auf RFC 868.

```
01 import socket
02 import struct, time
03 serverName = "haspe.homeip.net"
04 serverPort = 8042
```

```
05 zeit1970 = 2208988800L
06 stecker = socket.socket()
07 stecker.connect((serverName, serverPort))
08 zeit = stecker.recv(4)
09 zeit = struct.unpack("!I", zeit)[0]
10 zeit = int(zeit - zeit1970)
11 stecker.close()
12 print "Serverzeit ist", time.ctime(zeit)
13 print "lokale Uhr weicht um", int(time.time()) - zeit, "Sek. ab"
```

f) Entwerfen und implementieren Sie einen Server, mit dem das Programm aus e) kommunizieren kann.

Anmerkungen:
Es wird der RFC 868 zur Verfügung gestellt (siehe http://www.faqs.org/rfcs/rfc868.html).

Unterrichtliche Voraussetzungen:
Im Unterricht wurden Grundlagen von Rechnernetzen und Verteilten Systemen, sowie die Modellierung von Prozessen in Server-Klienten Strukturen behandelt. Insbesondere sind den Schülerinnen und Schülern Unterschiede zwischen verbindungsorientierten und verbindungslosen Protokollen sowie ihre Vor- bzw. Nachteile, die beiden Internetprotokolle TCP und UDP, deren Einsatzbereiche und Restriktionen bekannt. Sie kennen Unterschiede zwischen belegten und freien Portnummern, Sockets und ihre prinzipielle Funktionsweise als Schnittstelle zu den unteren Ebenen der Protokolle.

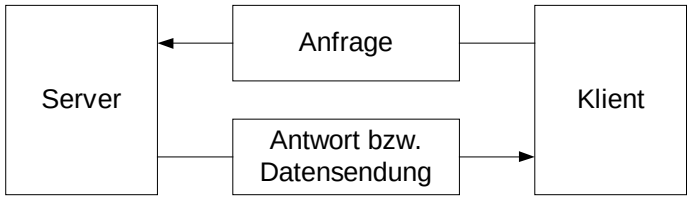
Sie kennen das Port-Modell als eine Möglichkeit zur Realisierung der Protokolle, haben die Entwicklung einer Modellierung mit Sequenzdiagrammen ausgehend von textuellen Beschreibungen geübt.

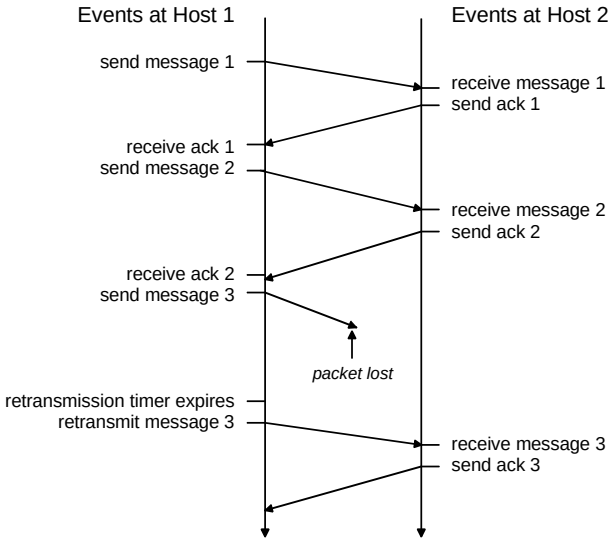
Sie haben gelernt, Modelle mit Hilfe einer geeigneten Programmiersprache zu implementieren und dabei Klassenbibliotheken zu nutzen.

Zusätzliche Hilfsmittel: Dokumentationen von Klassenbibliotheken

Vorgesehene Bearbeitungszeit: 80 min

Lösungsskizze, vorgesehene Bewertungseinheiten und ihre Zuordnung zu den Anforderungsbereichen I, II, III:

	Lösungsskizze	I	II	III
a)	In vernetzten Systemen werden aufeinander abgestimmte Zeiten benötigt, um Prozesse aufeinander abstimmen zu können. Ursächlich für Zeitdifferenzen ist das teilweise Fehlen eines Uhrenchips mit Speicherung während der Zeit, in der das System ausgeschaltet ist, dessen unzureichende Präzision, fehlende Systemzeitfunktion des Betriebssystems, sowie eine von einem Zeitnormal abweichende Zeiteinstellung durch die Nutzer.	4	2	-
b)	Um die Zeitdifferenzen zu beseitigen, werden in vernetzten Systemen Time-Server eingesetzt. Der Server liefert auf Anfrage, die über einen festgelegten Port akzeptiert wird, ein Datenpaket mit Zeitdaten an den Klienten aus. 	2	4	-
c)	Der grundlegende Unterschied zwischen dem TCP- und dem UDP-Protokoll liegt darin, dass das TCP-Protokoll ein verbindungsorientiertes Protokoll zwischen Server und Klient ist und das UDP-Protokoll ein verbindungsloses Protokoll. Zwar ist das darunter liegende IP-Protokoll verbindungslos, auf der darüber liegenden Ebene wird aber logisch unterschieden, so dass für die Anwendungsschicht der genannte Unterschied besteht. TCP baut im Gegensatz zu UDP vor der Datenübertragung zwischen den Kommunikationspartnern eine (virtuelle)	6	4	-

	Lösungsskizze	I	II	III
	Verbindung auf. Ein weiterer Unterschied besteht in der Sicherheit. Bei TCP wird jedes Paket bestätigt.  Dies ist bei UDP nicht der Fall. Da keine logische Verbindung zwischen Server und Klient besteht, bekommt der Server keine Auskunft darüber, ob die gesendeten Daten richtig und vollständig empfangen wurden. Es könnten Daten verloren gegangen oder von Dritten verändert worden sein. UDP wird z. B. bei folgenden Protokollen genutzt: dns, dhcp. TCP wird z. B. bei folgenden Protokollen genutzt: http, ftp, telnet, pop3, smtp.			
d)	Der Vorteil der Internettelefonie mit Hilfe eines verbindungslosen Protokolls liegt im günstigen Preis. Das einfache Protokoll vermeidet die für die Bestätigung von Paketen notwendigen Zeiten. Damit wird die Geschwindigkeit der Datenübertragung erhöht und die Kosten reduziert. Im Gegensatz zur traditionellen Telefonie muss keine permanente Leitung geschaltet sein. Nachteilig ist die mäßige Sprachqualität. Sie kann bei Verlust oder Verzögerung von Datenpaketen insbesondere bei Überlastungen leicht zustande kommen. Ein weiterer Nachteil liegt in der nicht abhörsicheren Übertragung über das Internet.	-	4	4

e)	Das vorgegebene Programm dient als Klient zur Abfrage der Zeit von einem Zeit-Server. In der Beschreibung der Funktionalität muss auf die Initialisierung, den Kommunikationsauf- und -abbau sowie die Zeit-Abfrage eingegangen werden.
	`01 import socket`
	Alle Elemente (in diesem Fall Klasse `socket`) des Moduls `socket` werden für den aktuellen Namensraum verfügbar gemacht.
	`02 import struct, time`
	Die Module `struct` (zur Konvertierung der Datenstruktur `struct` aus C -- in diesem Format bieten die Timeserver ihre Daten an) und `time` (zur Umrechnung von Zeitdaten) werden für den aktuellen Namensraum verfügbar gemacht.
	`03 serverName = "haspe.homeip.net"`
	Die Variable `serverName` wird mit dem Wert `"haspe.homeip.net"` (Typ Zeichenkette) belegt.
	`04 serverPort = 8042`
	Die Variable `serverPort` wird mit dem Wert 8042 (Typ integer) belegt.
	`05 zeit1970 = 2208988800L`
	Die Variable zeit1970 wird mit dem Wert 2208988800L (Typ longinteger) belegt.
		-	6	-

	Lösungsskizze	I	II	III

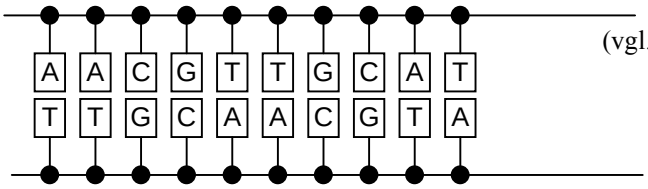
	`06 stecker = socket.socket()`
	Instanziierung der Klasse `socket` aus dem Modul `socket` mit Standardbelegungen für die Adressfamilie und die Art der Daten. Das Objekt erhält die Bezeichnung `stecker`.
	`07 stecker.connect((serverName, serverPort))`
	Die Methode `connect` wird auf das `socket`-Objekt mit dem Server und dem spezifizierten Port verbunden.
	`08 zeit = stecker.recv(4)`
	Die Zeitdaten (vier Bytes -- siehe RFC) werden vom Server angefragt und in der Variablen zeit abgelegt.
	`09 zeit = struct.unpack("!I", zeit)[0]`
	Umwandlung der Daten -- siehe RFC (Format ist der erste Parameter) in eine Pythonliste, von der das erste Element die Zeit (in Sekunden) darstellt. Formatangabe: "!I":
	`_____!____|____I_________`
	`big-endian| unsigned int`
	`10 zeit = int(zeit - zeit1970)`
	Normalisierung
	`11 stecker.close()`
	Verbindung zum Server schliessen.
	`12 print "Serverzeit ist", time.ctime(zeit)`
	Ausgabe der Zeichenkette und der ermittelten (und umgewandelten) Serverzeit als Zeichenkette (durch die Funktion ctime).
	`13 print "lokale Uhr weicht um", int(time.time()) - zeit, "Sek. ab"`
	Ermittlung der Abweichung. `time.time()` liefert die lokale Systemzeit, die durch `int()` in den Datentyp Integer umgewandelt wird.			

f)	Verbale Beschreibung des Algorithmus
	
	Implementation mit Python:
	
	`zeitServer.py`
	`==========`
	`import socket`
	`import struct, time`
	`# für normale Benutzer nutzbare Portnummer`
	`portNR = 8042`
	`# Referenzzeit`
	`zeit1970 = 2208988800L`
	`# Vorbereitung des Servers`
	`dienst = socket.socket(socket.AF_INET, socket.SOCK_STREAM)`
	`dienst.bind(("", portNR))`
	`dienst.listen(1)`
	`print "Abhören des Ports mit der Nummer", portNR`
	`while 1:`
	`  # bediene permanent ohne Unterbrechung`
	`  kanal, anfrage = dienst.accept()`
	`  print "Verbindung von ", anfrage`
	`  zeit = int(time.time()) + zeit1970`
	`  zeit = struct.pack("!I", zeit)`
	`  kanal.send(zeit) # sende Zeit[stempel]`
	`  kanal.close() # Verbindung abbauen`
		-	4	4

	Insgesamt 44 BWE	12	24	8

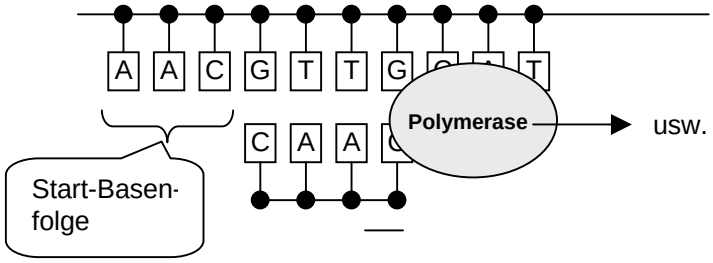
1.2.7 DNS-Replikation *GF*

In einem DNS-Molekül stehen sich jeweils die Basen Adenin und Thymin sowie Cytosin und Guanin (A, T, C, G) komplementär in einer Doppelkette gegenüber (vgl. nachfolgendes Beispiel).



usw.

Der Vorgang der DNS-Replikation kann vereinfachend wie folgt beschrieben werden: die Doppelkette trennt sich auf und beide Einzelstränge werden durch Polymerase zu vollständigen DNS-Molekülen ergänzt. Die Kettenbildung beginnt ab einer Start-Basenfolge, die durch eine bestimmte Anordnung der Basen gekennzeichnet ist.

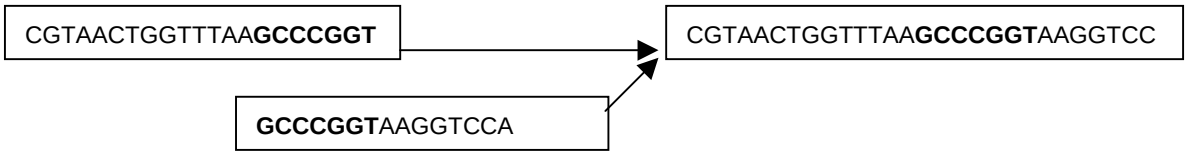


Stark vereinfacht kann die Polymerase als endlicher Automat beschrieben werden, der Basen als Eingabezeichen erhält und die zugehörigen Basen ausgibt, nachdem er die Startfolge gelesen hat. Als Startfolge wollen wir **A-A-C** festlegen. Vor dem Lesen dieser Startfolge gibt der Automat nichts aus.

a) Beschreiben Sie die Polymerase als Automaten durch Angabe der Alphabete und des Transitionsgraphen.

b) Geben Sie eine Strategie an, wie der Automat aus a) in einer geeigneten Programmiersprache realisiert werden kann und implementieren Sie dann den Automaten in der von ihnen genannten Art.

c) Bei der DNS-Analyse werden immer nur Teilstücke von ca. 500 Basen untersucht. Man fügt diese Teilstücke zusammen, indem man die überlappenden Enden findet, dann aus zwei Teilen deren Gesamtsequenz zusammenfügt, und an diese danach ebenso die nächsten Stücke. Schreiben Sie eine Funktion, der zwei Strings (die Basenketten) übergeben werden, und die diese richtig zusammensetzt zurück gibt. Die Basenfolgen sind richtig zusammengesetzt, wenn sie sich möglichst weit, aber wenigstens mit drei Basen, überlappen. Sie können davon ausgehen, dass der Überlappungsbereich kleiner als jedes der Teilstücke ist. Findet man keine richtige Überlappung, dann wird ein leerer String zurück gegeben. Beschreiben Sie Ihr Verfahren stichwortartig.



Anmerkungen:

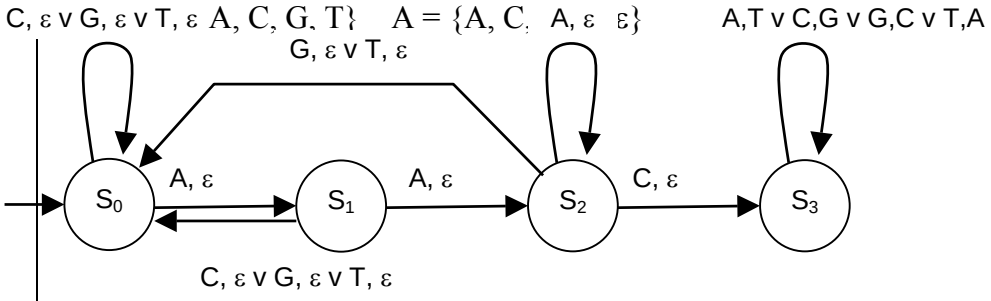
Zielsetzung:
Der Prüfling soll nachweisen, dass er bekannte Sachverhalte auf ein unbekanntes System übertragen und in ein Programm umsetzen kann. Er soll in diesem Kontext für ein neues Problem eine angemessene Lösung finden..

Unterrichtliche Voraussetzungen:
Im Unterricht wurden endliche Automaten und ihre Simulation behandelt. Die Umsetzung von Automaten in Programme muss bekannt und geübt worden sein.

Zusätzliche Hilfsmittel: keine

Vorgesehene Bearbeitungszeit: 70 min

Lösungsskizze, vorgesehene Bewertungseinheiten und ihre Zuordnung zu den Anforderungsbereichen I, II, III:

<table>
<tr><th></th><th>Lösungsskizze</th><th>I</th><th>II</th><th>III</th></tr>
<tr><td></td><td> 
</td><td>3</td><td>6</td><td>3</td></tr>
<tr><td>b)</td><td>(Quellcode s. Anhang)
Implementierung z. B. als Klasse mit Zustand, Überführungs- und Ausgabefunktion.
Kurze Beschreibung der Benutzung, z. B. der Auswertung der Eingabezeichen.</td><td>9</td><td>12</td><td>-</td></tr>
<tr><td>c)</td><td><pre>
function fuege_zusammen(s1, s2: string): string;
  var i, len, len1, len2: integer;
      h1, h2: string;
      fertig: boolean;
begin
  fertig := false;
  len1 := length(s1);
  len2 := length(s2);
  i := 1;
  if len1 > len2 then len := len2 else len := len1;
  repeat
    h1 := copy(s1, len1-len+i, len+1-i);
    h2 := copy(s2, 1, len+1-i);
    if h1 = h2 then fertig := true
      else i := i + 1;
  until fertig or (i > l);
  if len-i > 1 then result := s1 + copy(s2, len-i+2, len2-len+i-1)
    else result := '';
end;
</pre>
Das längste überlappende Stück finden und die Zeichenketten richtig zusammensetzen</td><td>-</td><td>3</td><td>6</td></tr>
<tr><td></td><td>Insgesamt 42 BWE</td><td>12</td><td>21</td><td>9</td></tr>
</table>

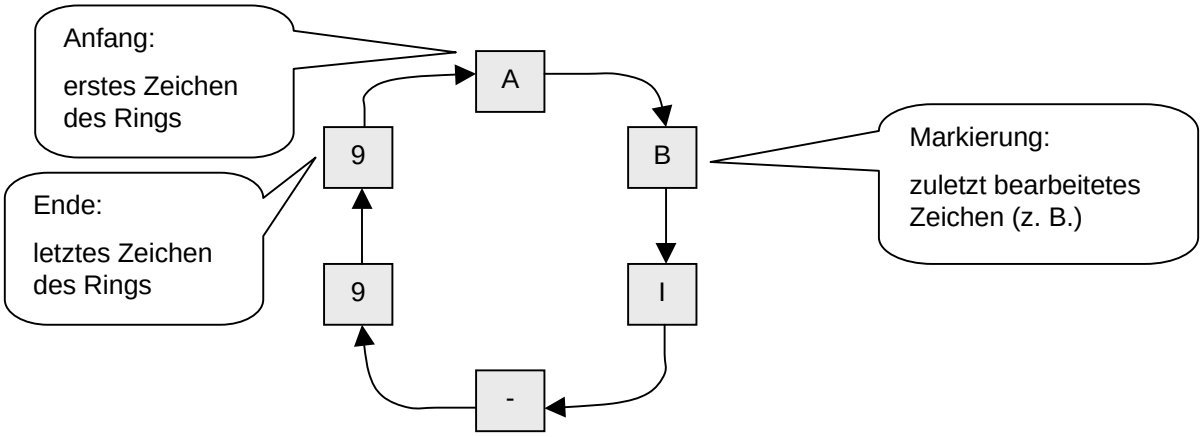
Lösungen zu b) in Delphi: Anhang

```
type
  tZustand = (s0, s1, s2, s3);
  tPolymerase = class                                  //Automat als Klasse
    s : tZustand;                                      //Zustand
    constructor init;                                  //Konstruktor
    function u(z: tZustand; e: char): tZustand;        //Überführungsfunktion
    function g(z: tZustand; e: char): char;            //Ausgabefunktion
    function arbeite(ein: string): string;
  end;

var Sequenz, Kopie: string;
    Polymerase    : tPolymerase;
```

```
constructor tPolymerase.init;
begin
  s := s0
end;

function tPolymerase.arbeite(ein: string): string;
  var h: string;
      i: integer;
      c: char;
begin
  s := s0;
  h := '';
  for i := 1 to length(ein) do begin
    c := ein[i];
    h := h + g(s,c);
    s := u(s,c)
  end;
  result := h
end;

function tPolymerase.u(z: tZustand; e: char): tZustand;
begin
  case z of
    s0: case e of
          'C','G','T': result := s0;
          'A'        : result := s1
          end;
    s1: case e of
          'C','G','T': result := s1;
          'A'        : result := s2
          end;
    s2: case e of
          'A','G','T': result := s0;
          'C'        : result := s3
          end;
    s3:  result := s3;
  end
end;

function tPolymerase.g(z: tZustand; e: char): char;
begin
  case z of
    s0,s1,s2: result := 'e';
    s3: case e of
          'A': result := 'T';
          'C': result := 'G';
          'G': result := 'C';
          'T': result := 'A';
        end;
  end
end;

procedure TForm1.EingabeClick(Sender: TObject);  //Nutzung des Automaten
begin
  Sequenz := Inputbox('Eingabe:', 'Geben Sie eine Basensequenz ein', '');
  Kopie := Polymerase.arbeite(Sequenz);
  ShowMessage('Ergebnis = ' + Kopie);
end;

initialization
  Polymerase := tPolymerase.init;                  //Objekt erzeugen
end.
```

1.2.8 Abstrakter Datentyp tRing *GF*

Diese Aufgabe beschäftigt sich mit dem abstrakten Datentyp (ADT) *tRing*, der eine begrenzte Anzahl von einzelnen Zeichen (CHAR) aufnehmen kann, die in Form eines Ringes verwaltet werden. Solche Ringstrukturen werden z. B. benötigt, um „Endlostexte“ wie im Vorspann der Fernsehnachrichten oder auf den Anzeigetafeln der Sportstadien zu verwalten, bei denen immer die gleiche Zeichenfolge in Form einer Endlosschleife dargestellt wird. Das zuletzt „bearbeitete“ (also z. B. zum Zweck der Darstellung gelesene) Zeichen wird besonders markiert:



Auf diesen ADT soll mit den folgenden Methoden zugegriffen werden können:

Init	erzeugt einen neuen leeren Ring
IstLeer	liefert TRUE, falls der Ring leer ist, sonst FALSE
FuegeEin	fügt ein neues Zeichen nach dem zuletzt bearbeiteten Zeichen ein
HoleNaechstes	holt das nächste nach dem zuletzt bearbeiteten Zeichen (entfernt es aber nicht aus dem Ring) und setzt die Markierung auf dieses Zeichen
LoescheErstes	löscht das erste Zeichen im Ring

a) Erläutern Sie anhand einiger Skizzen, wie nacheinander in einen zunächst leeren Ring die Zeichen <A><B><I> eingefügt werden.

b) Geben Sie eine Implementierung des ADT als Zeigerstruktur wahlweise konventionell (durch Datentypen und Unterprogramme) oder als Klasse an.

c) Gesucht ist eine Prozedur *LoescheZiffern*, die alle eventuell im Ring auftretenden Ziffern (0..9) löscht. Die Prozedur soll nur mit den Zugriffsmethoden des ADT arbeiten. Leider fehlt in der Definition des ADT mindestens eine Methode, um das Gewünschte zu erreichen. Beschreiben Sie kurz die Probleme, die beim Schreiben von *LoescheZiffern* auftreten und ergänzen Sie den ADT geeignet. Schreiben Sie dann *LoescheZiffern* mit Hilfe der zusätzlichen Methode(n).

d) Bei Suchvorgängen in komplexen Strukturen z. B. in Labyrinthen besteht das Problem zu erkennen, ob Unterstrukturen mehrfach hintereinander auftreten oder ob sich der Suchvorgang in einem Zyklus befindet. Vergleichen Sie dieses Problem mit den im ADT *tRing* auftretenden Problemen. Beschreiben Sie die Analogien detailliert (ggf. auch durch Skizzen) und diskutieren Sie Lösungsmöglichkeiten.

Anmerkungen

Zielsetzung:
Der Prüfling soll nachweisen, dass er bekannte Sachverhalte auf ein ähnliches System übertragen und in ein Programm umsetzen kann. Er soll Vorgänge analysieren, auftretende Probleme erkennen und Analogien zu anderen Gebieten erkennen.

Unterrichtliche Voraussetzungen:
Im Unterricht sind entweder Zeigerstrukturen oder Implementierungen dynamischer Klassen erarbeitet worden. Abstrakte Datentypen sind bekannt, wurden ähnlich wie in der Aufgabenstellung vereinbart und mithilfe einer geeigneten Programmiersprache implementiert. Das Suchen in Bäumen ist geübt worden.

Zusätzliche Hilfsmittel: keine

Vorgesehene Bearbeitungszeit: 90 min

Lösungsskizze, vorgesehene Bewertungseinheiten und ihre Zuordnung zu den Anforderungsbereichen I, II, III:

	Lösungsskizze	I	II	III
a)	Skizzen und kurze Erläuterung	6	-	-
b)	(Quellcode s. Anhang) Datentypen festlegen und in Abhängigkeit davon die gesuchten Methoden schreiben	9	15	6

c)	`function tRing.HoleErstes: char;`
	`begin`
	`  if not IstLeer then HoleErstes := Anfang^.Inhalt`
	`  else HoleErstes := '|'`
	`end;`
	
	`procedure LoescheZiffern(var r: tRing);`
	`var h: tRing;`
	`  c: char;`
	`begin`
	`  h := tRing.Init;`
	`  while not r.IstLeer do begin`
	`  c := r.HoleErstes;`
	`  r.LoescheErstes;`
	`  if not (c in ['0'..'9']) then h.FuegeEin(c)`
	`  end;`
	`  r := h`
	`end;`
	
	Periodizitätsprobleme erkennen, z. B. HoleErstes definieren und LoescheZiffern schreiben
		3	6	3

d)	Jeweils ein Beispiel für eine Wiederholung bzw. Periodizität im Ring und im Labyrinth geben. Analogien nennen und Lösungsmöglichkeiten, z. B. das Setzen von Markierungen oder den Aufbau von Datenstrukturen nennen und auf das Problem bezogen diskutieren.	-	5	3
	Insgesamt 56 BWE	18	26	12

Lösungen zu b) in Delphi: Anhang

```
type tZeiger = ^tKnoten;
     tKnoten = record
       Inhalt   : char;
       Naechster: tZeiger
     end;
     tRing  = class
       Anfang,Ende,Markierung: tZeiger;
       constructor Init;
       function IstLeer: boolean;
       procedure FuegeEin(c: char);
       function HoleNaechstes: char;
       function HoleErstes: char;
       procedure LoescheErstes;
     end;

var ring: tRing;

constructor tRing.Init;
begin
  Anfang := NIL;
  Ende := NIL;
  Markierung := NIL
end;

function tRing.IstLeer: boolean;
begin IstLeer := Anfang = NIL end;

procedure tRing.FuegeEin(c: char);
begin
  if IstLeer then begin
    new(Anfang); Ende := Anfang;
    Markierung := Anfang
    end
  else begin
    new(Ende^.Naechster);
    Ende := Ende^.Naechster
  end;
  Ende^.Inhalt := c;
  Ende^.Naechster := Anfang
end;

function tRing.HoleNaechstes: char;
begin
  HoleNaechstes := '|';
  if not IstLeer then begin
    HoleNaechstes := Markierung^.Inhalt;
    Markierung := Markierung^.Naechster
  end
end;

procedure tRing.LoescheErstes;
begin
  if not IstLeer then
   if Anfang = Ende then begin
     Anfang := NIL;
     Ende := NIL; Markierung := NIL
     end
```

```
   else begin                                    end
     if Markierung = Anfang                   end;
       then Markierung:=
Markierung^.Naechster;
     Anfang := Anfang^.Naechster;
     Ende^.Naechster := Anfang
```

1.3 Weitere Beispiele für das Leistungskursfach

1.3.1 E-Mail-Adressen *LF*

E-Mail-Adressen sind nach syntaktischen Regeln aufgebaut. Die folgenden drei Beispiele geben die typische Struktur von E-Mail-Adressen wieder:

info@jugend-forscht.de
thomas.schulze@gym.dd.sn.schule.de
info-fwu@t-online.de

Eine E-Mail-Adresse soll nur aus Kleinbuchstaben und aus den folgenden Zeichen aufgebaut sein:

@	.	-

a) Geben Sie einen endlichen Automaten an, der die folgende Sprache akzeptiert: Die Menge aller Zeichenfolgen, die eine syntaktisch korrekte E-Mail-Adresse darstellen.

b) Erläutern Sie an der folgenden E-Mail-Adresse, wie der von Ihnen in Teilaufgabe a) angegebene endliche Automat arbeitet:

rektorat@uni-jena.de

c) Beschreiben Sie in einer geeigneten Darstellungsform die Syntax von E-Mail-Adressen. Erläutern Sie an dieser Syntaxbeschreibung, was man unter einem Terminalsymbol, einem Nichtterminalsymbol, dem Startsymbol und einer Produktionsregel versteht.

d) Entscheiden Sie, ob es sich bei der von Ihnen in Teilaufgabe c) angegebenen Grammatik um die Grammatik einer regulären Sprache handelt. Begründen Sie Ihre Entscheidung.

Anmerkungen

Unterrichtliche Voraussetzungen:
Im Unterricht sind formale Sprachen und Automaten behandelt worden. Die Schülerinnen und Schüler haben Sprachbeschreibungen erarbeitet, sie kennen Syntaxdiagramme und EBNF. Die Struktur von E-Mail-Adressen ist ihnen aus dem praktischen Umgang geläufig.

Zusätzliche Hilfsmittel: keine

Vorgesehene Bearbeitungszeit: 45 min

1.3.2 Verwaltung eines Warenlagers *LF*

Diese Aufgabe ist mit dem PC zu lösen

Warenlager spielen in der Wirtschaft eine wichtige Rolle. In dieser Aufgabe geht es um ein einfaches Warenlager, das aus 2000 Lagerplätzen besteht, die in einer Reihe angeordnet sind. Die Lagerplätze sind in dieser Reihe fortlaufend von 1 bis 2000 nummeriert. In dem Warenlager können Kartons der Größen A, B und C aufbewahrt werden. Ein Karton der Größe A belegt zwei Lagerplätze, ein Karton der Größe B drei und ein Karton der Größe C acht Lagerplätze. Jeder Karton besitzt eine ganzzahlige Karton-Nummer, die ihn eindeutig

identifiziert. Die Karton-Nummern stammen aus dem Intervall von 1 bis 10000. Keine zwei Kartons besitzen die gleiche Karton-Nummer.

a) Entwerfen und implementieren Sie ein Modul, das die folgenden Operationen zur Verwaltung des Warenlagers realisiert: Einlagern eines Kartons, Auslagern eines Kartons und Ermitteln, ob sich ein Karton mit einer bestimmten Karton-Nummer im Warenlager befindet. Bei Bedarf sind Kartons zusammenzuschieben, um Platz für einen neuen Karton zu schaffen.

b) Entwerfen und implementieren Sie ein Programm, das die Verwaltung des Warenlagers realisiert. Das Programm nutzt das externe Modul von Teilaufgabe a). Das Warenlager ist zu Beginn leer.

Anmerkungen

Unterrichtliche Voraussetzungen:
Im Unterricht wurden Probleme unterschiedlicher Komplexität mit Hilfe des Computers bearbeitet. Bei der Bearbeitung wurden die Phasen Entwurf, Implementierung und Reflexion unterschieden. Die Schülerinnen und Schüler haben im Unterricht Schnittstellen von Modulen entworfen und umgesetzt.

Zusätzliche Hilfsmittel: Den Prüflingen steht zur Lösung der Aufgabe ein PC mit Softwareentwicklungsumgebung zur Verfügung.

Vorgesehene Bearbeitungszeit: 100 min

1.4 Weitere Beispiele für das Grundkursfach

1.4.1 Klammerstrukturen *GF*

In den meisten Programmiersprachen wird von Klammerstrukturen ausgiebig Gebrauch gemacht. In Java werden die Klammersymbole (,), [,], { und } verwendet. Klammersymbole müssen korrekt geschachtelt sein.
Die Zeichenketten

$S_1(S_2)S_3$,
$S_1[S_2]S_3$,
$S_1\{S_2\}S_3$

sind korrekt geklammert, wenn die Zeichenketten S_1, S_2 und S_3 jeweils korrekt geklammert sind. Dabei können S_1, S_2 und S_3 auch Zeichenketten ohne Klammern oder leer sein.

Zur Analyse einer Zeichenkette auf syntaktisch korrekte Klammerung ist die Verwendung eines Stapels zweckmäßig. Die Stapeloperationen push, pop, isEmpty, top seien in der üblichen Weise definiert.

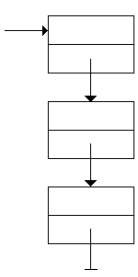
a) Entwickeln Sie einen Algorithmus in verbaler Form, mit dem die Syntaxanalyse in Bezug auf die Klammerung durchgeführt werden kann.

   Demonstrieren Sie den Algorithmus an den Beispielen {...(...)...[...]...}, {...(...[...]...)...} und {...(...)...[...}...]. Die Leerstellen ... sollen keine weiteren Klammern enthalten.

b) Schreiben Sie in Java eine Operation `boolean isKorrektgeklammert(String ausdruck)`, mit der die syntaktisch korrekte Klammerung einer Zeichenkette `ausdruck` überprüft wird. Sie können voraussetzen, dass die Stapel-Operationen Parameter vom Typ char übernehmen können (siehe Aufgabenteil c).

c) Implementieren Sie die Stapel-Operationen in Java mit einer verketteten Liste und erläutern Sie sie. Benutzen Sie dazu die Klasse Zelle.

```
class Zelle {
  char inhalt;
  Zelle nachfolger;

  Zelle() {
    inhalt = ' ';
    nachfolger = null;
  }
}
```



d) Vergleichen Sie diese Implementation mit der durch Reihungen.

e) Für manche Anwendungen von Stapeln werden zwei Stapel benötigt. Beschreiben Sie, wie zwei Stapel effizient in einer einzigen Reihung implementiert werden können.

Anmerkungen

Unterrichtliche Voraussetzungen:
Im Unterricht ist der abstrakte Datentyp „Stapel“ und die Implementation durch eine Reihung behandelt worden. Der Aufbau einer Zeiger-Struktur könnte zum Beispiel an Bäumen erarbeitet worden sein; die Implementation des Stapels durch eine verkettete Liste sollte nicht explizit behandelt worden sein, so dass die Bearbeitung des Aufgabenteils c) über eine Reproduktion hinausgeht.

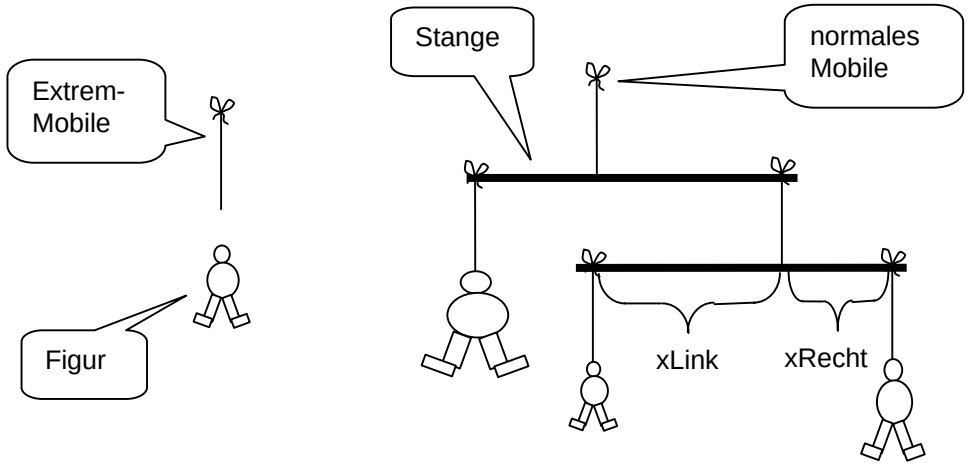
Es wird vorausgesetzt, dass die Anwendung von Stapeln bei der Bewertung von Postfix-Termen bekannt ist.

Zusätzliche Hilfsmittel: keine

Vorgesehene Bearbeitungszeit: 80 min

1.4.2 Mobile *GF*

Ein Mobile ist ein „Kunstwerk“, an dem Figuren an Stangen hängen, die wiederum durch Fäden verbunden sind. Im Extremfall besteht ein Mobile nur aus einer einzelnen Figur. Ein „normales“ Mobile enthält mehrere Stangen, an deren Ende jeweils entweder eine weitere Stange oder eine Figur hängt.



Wenn als „Mobile“ der Faden verstanden wird, an dem das Gebilde hängt, dann besteht ein Mobile aus (Teil)Mobiles, bei denen es sich entweder um Extremmobiles handelt (also einzelne Figuren) oder um normale Mobiles (also um Stangen mit Anhang).

Vereinfachend wird angenommen:

- Figuren haben eine punktförmige Masse, angegeben in Gramm.
- Fäden und Stangen haben keine Masse.
- Die Erdbeschleunigung betrage 10 m/s^2.
- Ein allgemeines Mobile wird durch eine Referenz auf ein Anhängsel beschrieben.

Das Anhängsel enthält Informationen über die Art (Figur oder Stange), über die Masse, den Aufhängepunkt und eventuell weitere Referenzen.

a) Geben Sie geeignete Datenstrukturen für den Typ *tMobile* an. In diesen soll u. a. gespeichert sein, ob es sich beim Mobile um eine einzelne Figur oder um eine Stange handelt.

b) Schreiben Sie eine Prozedur *NeuesMobile*, in der „per Hand“ (durch direktes Setzen der Referenzen) ein Mobile erzeugt wird, das dem oben rechts dargestellten entspricht (also zwei Stangen und drei Figuren enthält). Das Mobile soll im Gleichgewicht sein (s. 1.c)!

c) Analysieren Sie die vorgegebene Funktion *Gesamtmasse* und beschreiben Sie, was sie ermittelt.

```
function tMobile.Gesamtmasse: single;
begin
  if typ=Figur
    then result := masse
    else result := aLinks.Gesamtmasse + aRechts.Gesamtmasse;
end;
```

d) Schreiben Sie unter Benutzung der vorgegebenen Funktion *GesamtMasse* eine Funktion *ImGleichgewicht*. Im Gleichgewicht sind die Beträge der Drehmomente auf beiden Seiten einer Stange `(Masse-links)*xLinks*10` bzw. `(Masse-rechts)*xRechts*10` gleich und alle untergeordneten Stangen befinden sich im Gleichgewicht.

Anmerkungen

Zielsetzung:
Der Prüfling soll nachweisen, dass er für ein neues Problem angemessene Datenstrukturen auswählen und implementieren kann. Er soll Referenzen einzeln setzen können und in einer Baumstruktur rekursive Methoden benutzen.

Unterrichtliche Voraussetzungen:
Im Unterricht sind entweder Zeigerstrukturen oder Implementierungen dynamischer Klassen erarbeitet worden sind. Das Suchen in Bäumen ist geübt worden.

Zusätzliche Hilfsmittel: keine

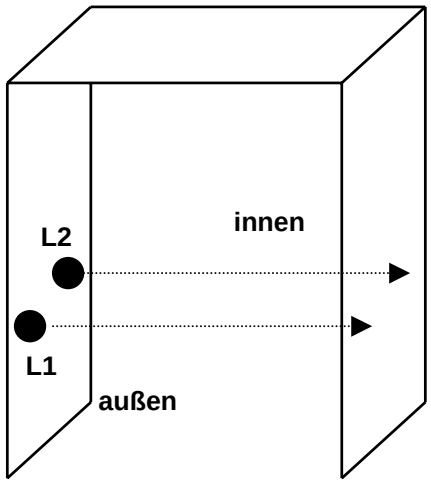
Vorgesehene Bearbeitungszeit: 60 min

1.4.3 Personenschleuse *GF*

Der einzige Eingang eines Computerzentrums soll durch ein Lichtschrankenpaar L1 und L2 in einer Personenschleuse so überwacht werden, dass die Zahl der Personen im Raum jederzeit festgestellt werden kann. Die Schleuse kann nur von einer Person gleichzeitig passiert werden, die dabei zuerst eine, dann beide und zuletzt die andere Lichtschranke unterbricht. Das Lichtschrankenpaar liefert die folgenden Signale:

0: wenn keine Lichtschranke unterbrochen ist,
1: wenn L1 unterbrochen ist,
2: wenn L2 unterbrochen ist,
3: wenn beide Lichtschranken unterbrochen sind.

Diese Signale werden in schneller Folge an einen Automaten weitergegeben.



a) Interpretieren Sie kurz die folgenden Signalfolgen, um sich mit der Wirkungsweise der Anlage vertraut zu machen.

1.) .....000111133333222000.....
2.) .....0001113333311000.....
3.) .....000222333311113333220000.....
4.) .....000111111111111110000000....
5.) .....000022331100.....

b) Geben Sie zwei Signalfolgen an, die in der Realität nicht auftreten können.
Beschreiben Sie dazu jeweils kurz, weshalb diese Fälle unmöglich sind.

c) Beschreiben Sie einen endlichen Automaten mit Ausgabe durch seinen Transitionsgraphen, der als Eingabe die Signalfolgen des Lichtschrankenpaares erhält und als Ausgabe Signale für eine Zählschaltung generiert, die die Anzahl der Personen im Raum zählt:

- Hat eine Person den Raum betreten, dann wird „+“ (für „+1“) ausgegeben.
- Hat eine Person den Raum verlassen, dann wird „-“ (für „-1“) ausgegeben.
- Sonst wird „n“ (für „nichts“) ausgegeben.

Im Graphen brauchen Sie die unmöglichen Signalfolgen aus b) nicht zu berücksichtigen.

Anmerkungen

Zielsetzung:
Der Prüfling soll nachweisen, dass er ein neues Problem analysieren und durch einen Automaten beschreiben kann.

Unterrichtliche Voraussetzungen:
Im Unterricht wurde der Umgang mit endlichen Automaten ausführlich geübt.

Zusätzliche Hilfsmittel: keine

Vorgesehene Bearbeitungszeit: 50 min

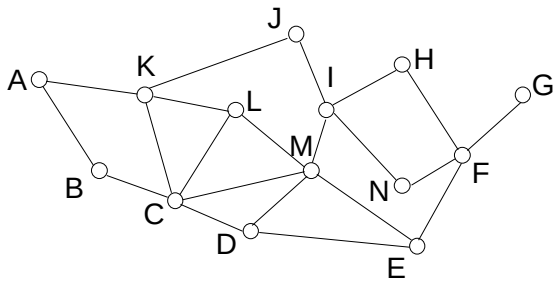
2 Aufgabenbeispiele für die mündliche Prüfung

Die folgenden Aufgabenbeispiele sind Teile von mündlichen Prüfungsaufgaben, die unterschiedliche Vorbereitungs- und Bearbeitungszeiten erfordern. Sie sollen die Eigenart und besondere Zielsetzung von mündlichen Prüfungen im Unterschied zur schriftlichen Prüfung verdeutlichen.

2.1 Austausch von Dokumenten

Wir betrachten den Austausch von HTML-Dokumenten über ein Computernetz.

a) Entwerfen Sie für das skizzierte Netz eine Datenstruktur und einen Suchbaum, mit dem Kommunikationswege von Computer A zu einem anderen Computer gefunden werden können, und erläutern Sie die Grundstruktur eines Algorithmus, mit dem der Suchbaum abgesucht werden kann.



b) Geben Sie ein Prolog-Prädikat für die Suche von Pfaden von *Start* nach *Ziel* im Netz an.

c) Erläutern Sie am Beispiel von HTML Möglichkeiten der Sprachbeschreibung und Sprachverarbeitung.

Lösungsskizze

zu a)

in Prolog z. B.: `kante(a, k).`
Tiefen- oder Breitensuche

zu b)

```
sucheWeg(Start, Pfad, Ziel):-
  Start = Ziel,
  write('Weg gefunden'), nl,
  write('Weg: '), write(Pfad), nl.

sucheWeg(Start, Pfad, Ziel):-
  kante(Start, Computer),
  not(member(Computer, Pfad)),
  sucheWeg(Computer, [Computer|Pfad], Ziel).
```

zu c)

Sprachbeschreibung: Grammatik, Syntaxdiagramm
Grammatik: Terminale, Variablen, Startsymbol, Produktionen (Ableitungsregeln)
Sprache: kontextfrei

Sprachverarbeitung: Browser als Interpreter eines HTML-Dokuments.
Scanner: erkennt HTML-Befehle
Parser: erkennt syntaktische Strukturen, z. B. Tabellenaufbau oder Hyperlinks
Interpreter: stellt den Inhalt des HTML-Dokuments grafisch dar.

2.2 Wertpapierbörse

Eine Wertpapier-Börse soll im lokalen Netz in Form eines Börsenspiels simuliert werden. Kunden können ihrer Bank Kauf- und Verkaufsaufträge erteilen. Der Börsencomputer ermittelt aus den eingegangenen Aufträgen den Kassakurs, zu dem die Aufträge abgewickelt werden.

a) Entwerfen Sie eine Datenbanktabelle für die Aufträge und eine Kundenklasse für die Ausführung von Aufträgen. Vergleichen Sie das Relationenmodell mit der objektorientierten Modellierung.

b) Entwerfen Sie eine kleine Transaktionssprache, mit welcher Aufträge beschrieben und abgewickelt werden können.

c) Erläutern Sie Chancen und Risiken des Online-Banking.

Lösungsskizze

zu a)

```
Auftrag(Kundennr, Wertpapiernummer, Anzahl, Preis,
        KaufVerkauf, ErledigtAnzahl, ErledigtPreis)

KundeTyp = object
  Nummer: Integer;
  Konto: String[20];
  Depot: String[20];
  procedure Init;
  procedure Kaufen(,,,);
  procedure Verkaufen (,,,);
end;
```

Vergleich der Modellierung bei relationalen Datenbanken und objektorientiertem Ansatz, z. B. Vererbung, Schlüssel, Methoden, Assoziationen, Aggregationen.

zu b)

```
BUY Wertpapiernummer Anzahl Preis
SELL Wertpapiernummer Anzahl Preis
SOLD Wertpapiernummer Anzahl Preis
BOUGHT Wertpapiernummer Anzahl Preis
```

zu c)

Zugang zur Börse von zuhause, schnellere Börsengeschäfte
Vertraulichkeit, Verbindlichkeit, Datensicherung, Datenschutz

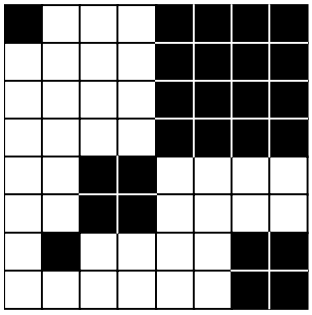
2.3 Quadtree

Quadratische Schwarz-Weiß-Grafiken, deren Seitenlänge (in Pixeln) eine Zweierpotenz ist, können mit Hilfe der Datenstruktur QuadTree gespeichert werden. Ein QuadTree ist dabei ein Baum, der entweder vier Teilbäume hat oder ein Blatt ist.

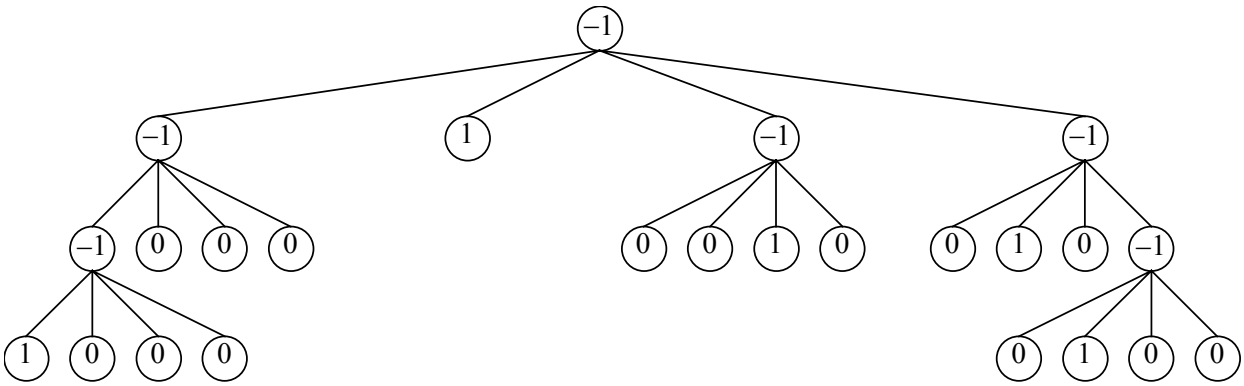
Der Inhalt eines Bildes wird nach folgendem Algorithmus in der Datenstruktur abgelegt:

- Jedem Teilquadrat der Grafik entspricht ein QuadTree.
- Ist die Farbe eines Teilquadrates einheitlich, so wird in den Knoten des zugehörigen QuadTree der Farbwert (1 = schwarz, 0 = weiß) eingetragen. Der QuadTree besitzt in diesem Fall leere Teilbäume und ist somit ein Blatt.
- Ist die Farbe nicht einheitlich, so wird dies im Knoten des zugehörigen QuadTree durch den Wert –1 kenntlich gemacht. Anschließend wird das Quadrat in vier Teilquadrate zerlegt. Deren Bildinhalte werden im Uhrzeigersinn (links oben, rechts oben, rechts unten, links unten) in Teilbäume abgelegt und an den aktuellen Knoten angehängt.

Beispiel: Folgendes 8x8-Schwarz-Weiß-Bild soll in einem QuadTree gespeichert werden:



Der zugehörige QuadTree sieht demnach wie folgt aus:

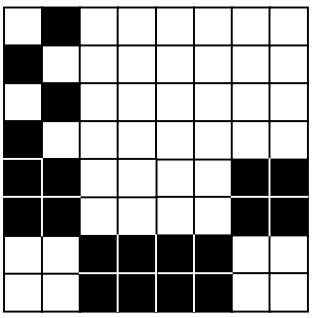


Durchläuft man den Baum in der Reihenfolge Wurzel-Linksaußen-Linksinnen-Rechtsinnen-Rechtsaußen (WLaLiRiRa), so erhält man die untenstehende Folge von Knoteninhalten, die z. B. als lineare Liste intern oder auch zur externen Speicherung der Grafik verwendet werden kann:

–1, –1, –1, 1, 0, 0, 0, 0, 0, 0, 1, –1, 0, 0, 1, 0, –1, 0, 1, 0, –1, 0, 1, 0, 0

Aufgaben:

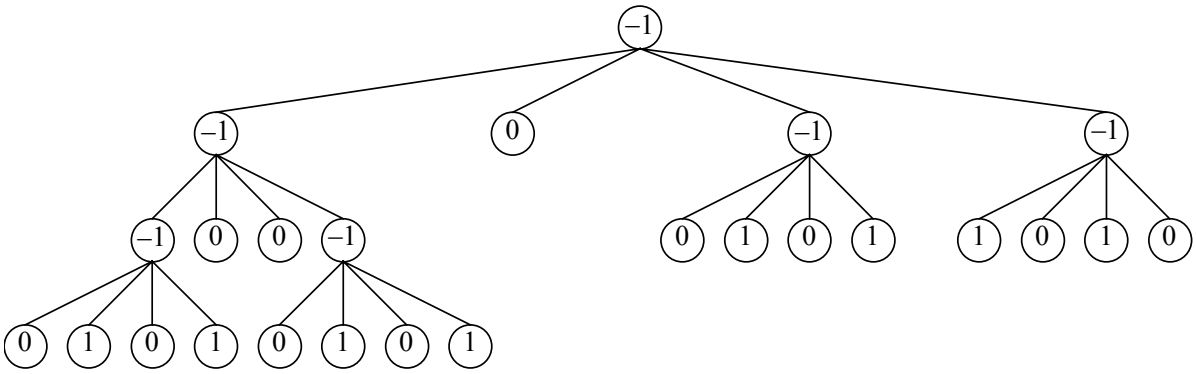
a) Zeichne den QuadTree zur folgenden 8x8-Schwarz-Weiß-Grafik. Gib anschließend die Folge von Knoteninhalten entsprechend der Traversierung WLaLiRiRa an.



b) Erstelle eine Klasse `CQuadTree`, welche Bäume der genannten Art repräsentiert.

Lösungsskizze: QuadTrees

a) Zur angegebenen Grafik gehört der folgende QuadTree



Das Bild entspricht somit der pre-Order erzeugten linearen Liste
–1, –1, –1, 0, 1, 0, 1, 0, 0, –1, 0, 1, 0, 1, 0, –1, 0, 1, 0, 1, –1, 1, 0, 1, 0

b) Eine Möglichkeit, die bereits die beiden Umwandlungsmethoden berücksichtigt, könnte wie folgt aussehen:

```
Klasse:      CQuadTree
Attribute:     Farbwert: shortint;
Teilbäume:     array[1..4] of CQuadTree;
Methoden:      Create;
               Destroy;
               Leer: boolean;
```

Anmerkungen
Unterrichtliche Voraussetzungen:
Die Datenstruktur des QuadTree ist dem Schüler so noch nicht begegnet. Die Struktur von mehrfach verzweigten Bäumen ist im Unterricht an unterschiedlichen Stellen behandelt worden. Auch die lineare Liste als Organisationsstruktur ist bekannt. Die Traversierung in „Preorder“, welche bei der Umformung in eine lineare Liste zum Tragen kommt, ist hinlänglich bekannt. Die Modellierung von Klassen wurde durchgängig im Unterricht geübt.

2.4 Suchmaschine

Die Suchmaschine GUTSUCH verwendet beim Beantworten von Anfragen eine Datenbank. Die Datenbank verwaltet unter anderem Schlüsselwörter und Internet-Adressen.

a) Entwickeln Sie ein ER-Diagramm, das die Daten, die die Suchmaschine beim Beantworten von Anfragen benötigt, modelliert.

b) Eine Anfrage an die Suchmaschine GUTSUCH kann mehrere Schlüsselwörter enthalten. Erläutern Sie, wie solche Anfragen von der Suchmaschine bearbeitet werden können.

c) Erläutern Sie einen Algorithmus, der Schlüsselwörter aus einem Text herauslöst. Der Algorithmus soll solche Wörter wie „der“ und „eine“ als Schlüsselwörter ausschließen.

Anmerkungen

Unterrichtliche Voraussetzungen:
Im Unterricht wurden ER-Modelle und deren Umsetzung in Relationenmodelle behandelt. Die Prüflinge wissen, wie logische Ausdrücke in höheren Programmiersprachen ausgewertet werden. Sie können mit einer Suchmaschine umgehen und haben mit deren Hilfe in mehreren Unterrichtsfächern zahlreiche Recherchen ausgeführt. Im Unterricht entwarfen sie zahlreiche Algorithmen.