Bitte beachten:

Die Aufgaben wurden nur an den Schulen 168932_192739_188803_168518_166730 verwendet.

Sie dürfen nicht veröffentlicht werden!

Name: _____

Abiturprüfung 2017

Informatik, Grundkurs

Aufgabenstellung:

Das Verwalten von Terminen ist eine Aufgabe, die oft mit Software gelöst wird. Im Folgenden soll eine vereinfachte Variante betrachtet werden, um grundsätzliche Elemente einer solchen Software zu untersuchen.

Die mit einer geeigneten Software gespeicherten Termine können folgende Daten verwalten: Zu jedem Termin gehört zwingend ein Startzeitpunkt, eine Dauer und eine Bezeichnung. Außerdem hat jeder Termin immer eine eindeutige Identifizierungsnummer, abgekürzt „UID“. Die UID eines Termins kann nicht geändert werden. Weitere Daten, die angegeben werden können, sind der Ort, an dem der Termin stattfindet und eine Kategorie, die ebenfalls in Textform angegeben wird.

Die Klasse `TerminVerwaltung` speichert die verwalteten Termine, wobei keine Reihenfolge vorgegeben ist. Dabei soll ein Objekt dieser Klasse garantieren, dass jede gespeicherte UID tatsächlich eindeutig ist. Es können Termine hinzugefügt, gelöscht oder gesucht werden. Zur Verwaltung der Termine wird eine Liste (Datentyp `List`) verwendet.

Terminliste

```
10.05.2017, 09:00   Informatik GK
Dauer: 180 Minuten  UID: abitur:nrw:5B12-0
Kategorie: Prüfung  Raum: A530

03.05.2017, 09:00   Mathematik GK
Dauer: 180 Minuten  UID: abitur:nrw:EEF1-2
Kategorie: Prüfung  Raum: C122

10.05.2017, 09:01   Informatik LK
Dauer: 255 Minuten  UID: abitur:nrw:51BC-1
Kategorie: Prüfung  Raum: C105

03.05.2017, 09:00   Mathematik LK
Dauer: 180 Minuten  UID: abitur:nrw:AA01-9
Kategorie: Prüfung  Raum: A530

09.05.2017, 09:00   Russisch GK
Dauer: 180 Minuten  UID: abitur:nrw:5B12-13
Kategorie: Prüfung  Raum: C122
```

Abbildung 1: Liste mit Terminen

Name: _____

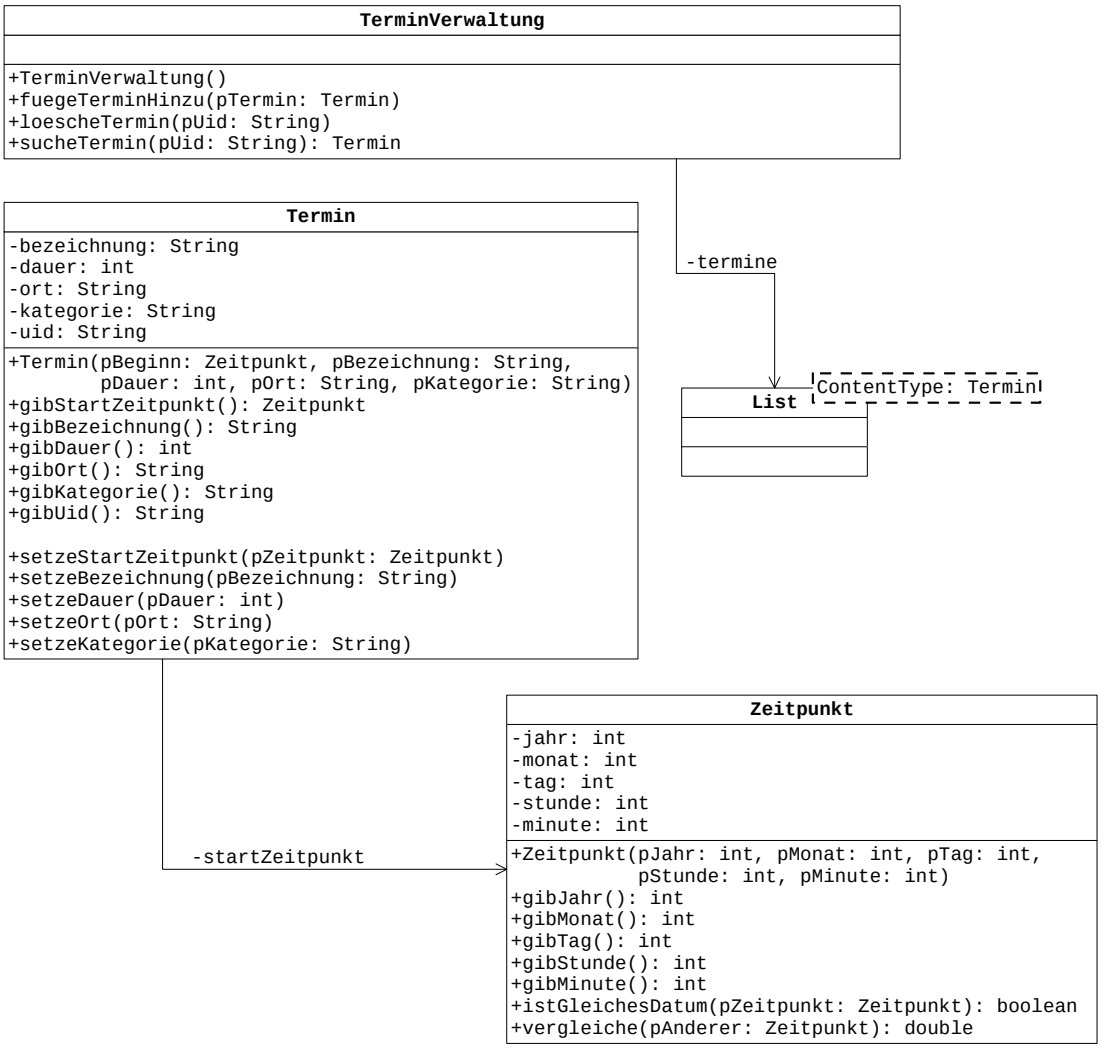


Abbildung 2: Implementationsdiagramm

a) *Beschreiben Sie das in Abbildung 2 dargestellte Implementationsdiagramm in Bezug auf die Beziehungen zwischen den Klassen.*

Erläutern Sie die Assoziationen im Kontext.

(6 Punkte)

Name: _____

b) Da die Liste der Termine unsortiert ist, können neue Termine einfach an die Liste angehängt werden. Um einen neuen Termin hinzufügen zu können, ist es allerdings notwendig zu überprüfen, ob die UID des neuen Termins noch nicht in der Terminliste existiert. Existiert die UID bereits, wird der Termin nicht hinzugefügt.

Die Methode `fuegeTerminHinzu` der Klasse `TerminVerwaltung` soll dies leisten. Sie hat den folgenden Methodenkopf

```
public void fuegeTerminHinzu(Termin pTermin)
```

Entwickeln Sie einen Algorithmus für das Hinzufügen eines Termins unter den genannten Bedingungen als Struktogramm.

Implementieren Sie die Methode `fuegeTerminHinzu`.

(14 Punkte)

c) Die folgende Methode `wasMacheIch` der Klasse `TerminVerwaltung` ist eine nicht dokumentierte Methode im Quellcode:

```
 1 public List<Termin> wasMacheIch(Zeitpunkt pDatum) {
 2   List<Termin> ergebnis = new List<Termin>();
 3   termine.toFirst();
 4   while (termine.hasAccess()) {
 5     Zeitpunkt neuerZeitpunkt
           = termine.getContent().gibStartZeitpunkt();
 6     if (neuerZeitpunkt.istGleichesDatum(pDatum)) {
 7       ergebnis.toFirst();
 8       while (ergebnis.hasAccess()
                 && ergebnis.getContent().gibStartZeitpunkt()
                     .vergleiche(neuerZeitpunkt) < 0.0) {
 9         ergebnis.next();
10       }
11       if (ergebnis.hasAccess()) {
12         ergebnis.insert(termine.getContent());
13       } else {
14         ergebnis.append(termine.getContent());
15       }
16     }
17     termine.next();
18   }
19   return ergebnis;
20 }
```

Name: _____

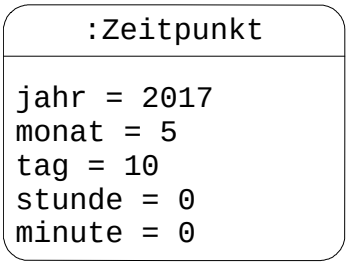


Abbildung 3: Darstellung des als Parameter übergebenen Zeitpunkts

Analysieren Sie die Methode, indem Sie sie auf die Beispieldaten aus Abbildung 1 und Abbildung 3 und anwenden. Dokumentieren Sie dazu immer nach Ausführen von Zeile 3 und Zeile 17 den Zustand der Listen `termine` *und* `ergebnis`*.*

Erläutern Sie im Sachzusammenhang, was die Methode leistet.

(10 Punkte)

Für den Benutzer ist es wichtig, eine Auswahl von Terminen anhand von Kriterien zu ermöglichen. Beispielsweise sollen folgende Termine gesucht werden: alle Termine der Kategorie `"Besprechung"`, alle Termine in Raum `"A530"` oder Termine der Kategorie `"Prüfung"`, die die Bezeichnung `"Klausur"` haben.

Um Termine so zu filtern, wird folgendes Konzept genutzt: Die verwalteten Termine werden mit einem Musterobjekt verglichen. Das Musterobjekt hat die gleichen Attribute wie ein Termin, besitzt aber keine UID. Wenn die Attribute eines gespeicherten Termins gleich denen des Musterobjekts sind, dann ist der Termin Teil des Ergebnisses.

Referenzieren Attribute des Musterobjekts `null` oder besitzen sie einen vereinbarten Wert – zum Beispiel `-1` beim Attribut `dauer` – so werden diese Attribute ignoriert. Die Gleichheit der Attribute wird also nur bei den gesetzten Attributen geprüft.

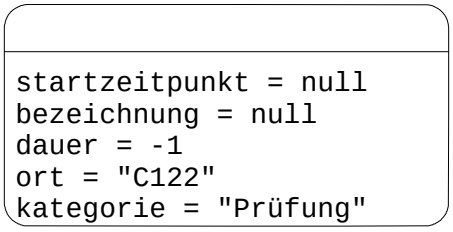


Abbildung 4: Beispiel für ein Musterobjekt zum Filtern nach Ort und Kategorie

Name: _____

d) *Wenden Sie dieses Filterkonzept auf die in Abbildung 1 und Abbildung 4 dargestellten Daten an und ermitteln Sie so eine Ergebnisliste.*

Modifizieren Sie das Implementationsdiagramm aus Abbildung 2, um dieses Filterkonzept implementieren zu können.

Erläutern Sie die Veränderungen an bestehenden Methoden und dokumentieren Sie die ergänzten Methoden.

Hinweis:
Nutzen Sie für die Modifikation des Implementationsdiagramms die Vorlage in der Anlage.

(13 Punkte)

e) Oft möchte man bei einer Terminverwaltung anders filtern, da Termine für einen bestimmten Zeitraum gesucht sind. Beispielsweise können alle Termine gesucht sein, die in der nächsten Woche beginnen, oder man möchte alle Vormittagstermine kennen, die zwischen 9:00 Uhr und 12:00 Uhr beginnen.

Begründen Sie, warum es mit dem gegebenen Filtersystem nicht möglich ist, eine solche Auswahl von Terminen zu filtern.

Erläutern Sie, wie das Filtermodell geändert werden kann, um derartige Filterungen zu ermöglichen.

(7 Punkte)

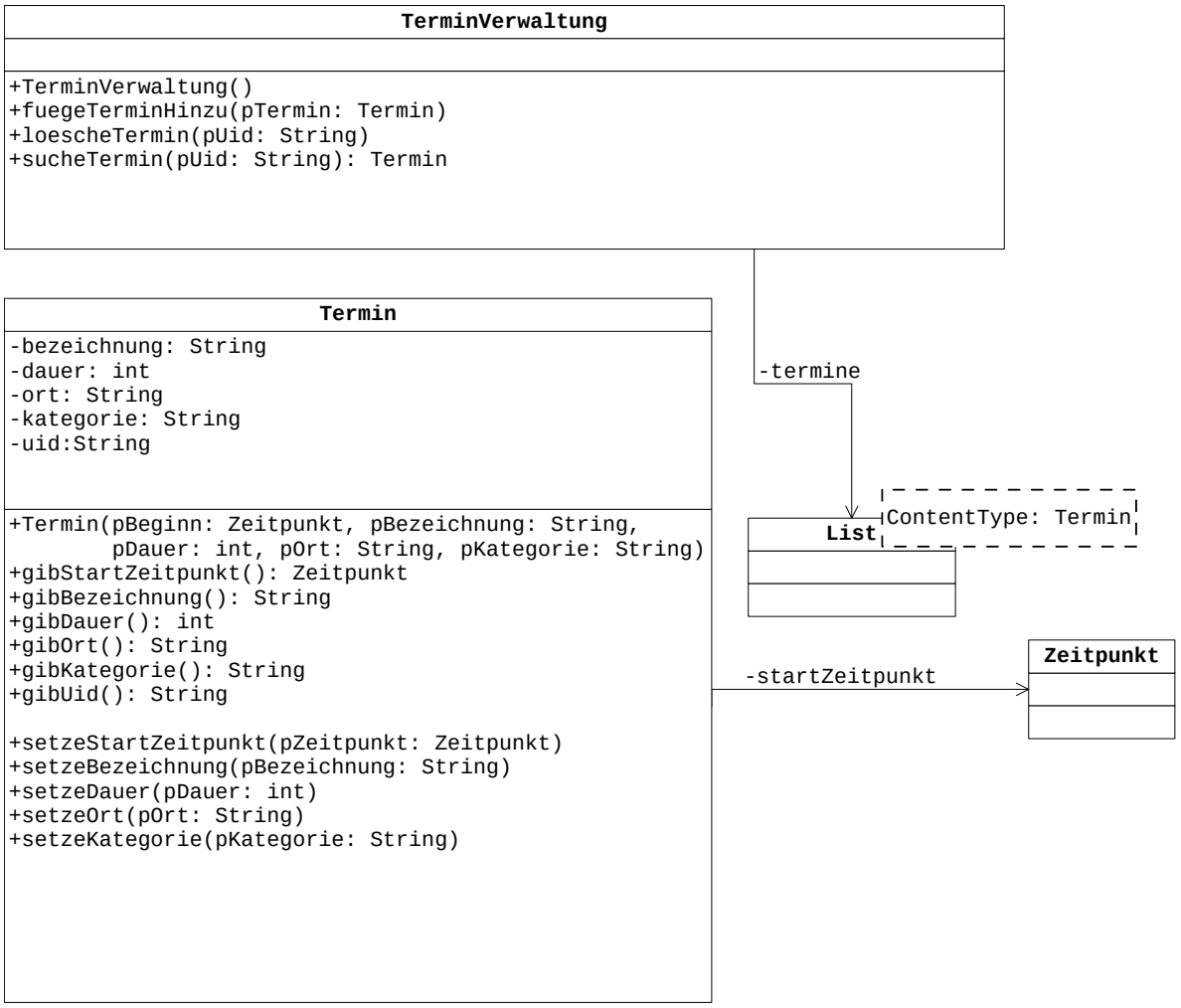
Zugelassene Hilfsmittel:

- Taschenrechner (graphikfähiger Taschenrechner / CAS-Taschenrechner)
- Wörterbuch zur deutschen Rechtschreibung

Name: _____

Anlage zu Aufgabenteil d)

Vorlage für die Modifikation des Implementationsdiagramms.



Name: _____

Ausschnitt aus der Dokumentation der Klasse `TerminVerwaltung`

Diese Klasse verwaltet beliebig viele Termine. Die Termine werden anhand ihrer UID unterschieden. Die UID jedes verwalteten Termins muss eindeutig sein.

Konstruktor `TerminVerwaltung()`

Nach dem Aufruf des Konstruktors ist ein Objekt der Klasse erzeugt.

Auftrag `void fuegeTerminHinzu(Termin pTermin)`

Die Methode fügt einen Termin in die Terminverwaltung ein, sofern die UID des Termins noch nicht existiert.

Auftrag `void loescheTermin(String pUid)`

Die Methode löscht den Termin mit der übergebenen UID aus der Terminverwaltung, sofern der Termin existiert. Referenziert `pUid` `null`, wird keine Veränderung vorgenommen.

Anfrage `Termin sucheTermin(String pUid)`

Die Methode gibt das Objekt mit der als Parameter übergebenen UID zurück, sofern ein Termin mit dieser UID in der Terminverwaltung existiert. Andernfalls wird `null` zurückgegeben. Referenziert `pUid` `null`, wird `null` zurückgegeben.

Name: _____

Ausschnitt aus der Dokumentation der Klasse `Termin`

Diese Klasse repräsentiert einen Termin.

Konstruktor **`Termin(Zeitpunkt pBeginn, String pBezeichnung, int pDauer, String pOrt, String pKategorie)`**
Nach dem Aufruf des Konstruktors sind die übergebenen Werte der Parameter und die übergebenen Referenzen der Parameter gespeichert. Das Objekt hat eine UID.

Anfrage **`Zeitpunkt gibStartZeitpunkt()`**
Die Methode gibt eine Referenz auf den Startzeitpunkt des Termins zurück.

Anfrage **`String gibUid()`**
Die Methode gibt die UID des Termins zurück.

Anfrage **`String gibBezeichnung()`**
Die Methode gibt die Bezeichnung des Termins zurück.

Anfrage **`int gibDauer()`**
Die Methode gibt die Dauer des Termins zurück.

Anfrage **`String gibOrt()`**
Die Methode gibt den Ort des Termins zurück.

Anfrage **`String gibKategorie()`**
Die Methode gibt die Kategorie des Termins zurück.

Auftrag **`void setzeStartZeitpunkt(Zeitpunkt pStartZeitpunkt)`**
Referenziert `pStartZeitpunkt` nicht `null`, wird die übergebene Referenz als Startzeitpunkt gespeichert.

Auftrag **`void setzeBezeichnung(String pBezeichnung)`**
Referenziert `pBezeichnung` nicht `null`, wird die übergebene Referenz als Bezeichnung gespeichert.

Auftrag **`void setzeDauer(int pDauer)`**
Sofern der übergebene Wert nicht negativ ist, wird er als Dauer gespeichert.

Auftrag **`void setzeOrt(String pOrt)`**
Referenziert `pOrt` nicht `null`, wird die übergebene Referenz als Ort gespeichert.

Auftrag **`void setzeKategorie(String pKategorie)`**
Referenziert `pKategorie` nicht `null`, wird die übergebene Referenz als Kategorie gespeichert.

Name: _____

Ausschnitt aus der Dokumentation der Klasse `Zeitpunkt`

Objekte dieser Klasse repräsentieren einen Zeitpunkt, der einerseits durch das Kalenderdatum, andererseits durch eine Uhrzeit festgelegt ist, wobei die Sekunden vernachlässigt werden.

Konstruktor **`Zeitpunkt(int pJahr, int pMonat, int pTag, int pStunde, int pMinute)`**

Nach dem Aufruf des Konstruktors sind die übergebenen Werte der Parameter gespeichert.

Anfrage **`int gibJahr()`**

Die Methode gibt das Jahr des Zeitpunkts zurück.

Anfrage **`int gibMonat()`**

Die Methode gibt den Monat des Zeitpunkts zurück.

Anfrage **`int gibTag()`**

Die Methode gibt den Tag des Zeitpunkts zurück.

Anfrage **`int gibStunde()`**

Die Methode gibt die Stunde des Zeitpunkts zurück.

Anfrage **`int gibMinute()`**

Die Methode gibt die Minute des Zeitpunkts zurück.

Anfrage **`boolean istGleichesDatum(Zeitpunkt pZeitpunkt)`**

Die Methode liefert als Ergebnis `true`, wenn bei beiden Zeitpunkte am gleichen Tag liegen: Das Jahr, der Monat und der Tag sind identisch. Andernfalls liefert die Methode `false`.

Anfrage **`double vergleiche(Zeitpunkt pAnderer)`**

Die Methode liefert als Ergebnis einen negativen Wert, wenn das Objekt vor dem als Parameter übergebenen Zeitpunkt liegt. Sie liefert den Wert `0`, wenn beide Zeitpunkte identisch sind, also das Datum und die Uhrzeit übereinstimmen. Andernfalls liefert die Methode einen positiven Wert.

Name: _____

Dokumentationen der verwendeten Klassen

Die generische Klasse List<ContentType>

Objekte der generischen Klasse **`List`** verwalten beliebig viele, linear angeordnete Objekte vom Typ **`ContentType`**. Auf höchstens ein Listenobjekt, aktuelles Objekt genannt, kann jeweils zugegriffen werden. Wenn eine Liste leer ist, vollständig durchlaufen wurde oder das aktuelle Objekt am Ende der Liste gelöscht wurde, gibt es kein aktuelles Objekt. Das erste oder das letzte Objekt einer Liste können durch einen Auftrag zum aktuellen Objekt gemacht werden. Außerdem kann das dem aktuellen Objekt folgende Listenobjekt zum neuen aktuellen Objekt werden.
Das aktuelle Objekt kann gelesen, verändert oder gelöscht werden. Außerdem kann vor dem aktuellen Objekt ein Listenobjekt eingefügt werden.

Dokumentation der Klasse `List<ContentType>`

Konstruktor **`List()`**

Eine leere Liste wird erzeugt. Objekte, die in dieser Liste verwaltet werden, müssen vom Typ `ContentType` sein.

Anfrage **`boolean isEmpty()`**

Die Anfrage liefert den Wert `true`, wenn die Liste keine Objekte enthält, sonst liefert sie den Wert `false`.

Anfrage **`boolean hasAccess()`**

Die Anfrage liefert den Wert `true`, wenn es ein aktuelles Objekt gibt, sonst liefert sie den Wert `false`.

Auftrag **`void next()`**

Falls die Liste nicht leer ist, es ein aktuelles Objekt gibt und dieses nicht das letzte Objekt der Liste ist, wird das dem aktuellen Objekt in der Liste folgende Objekt zum aktuellen Objekt, andernfalls gibt es nach Ausführung des Auftrags kein aktuelles Objekt, d. h., `hasAccess()` liefert den Wert `false`.

Auftrag **`void toFirst()`**

Falls die Liste nicht leer ist, wird das erste Objekt der Liste aktuelles Objekt. Ist die Liste leer, geschieht nichts.

Auftrag **`void toLast()`**

Falls die Liste nicht leer ist, wird das letzte Objekt der Liste aktuelles Objekt. Ist die Liste leer, geschieht nichts.

Name: _____

Anfrage **`ContentType getContent()`**

Falls es ein aktuelles Objekt gibt (`hasAccess() == true`), wird das aktuelle Objekt zurückgegeben. Andernfalls (`hasAccess() == false`) gibt die Anfrage den Wert `null` zurück.

Auftrag **`void setContent(ContentType pContent)`**

Falls es ein aktuelles Objekt gibt (`hasAccess() == true`) und `pContent` ungleich `null` ist, wird das aktuelle Objekt durch `pContent` ersetzt. Sonst bleibt die Liste unverändert.

Auftrag **`void append(ContentType pContent)`**

Ein neues Objekt `pContent` wird am Ende der Liste eingefügt. Das aktuelle Objekt bleibt unverändert. Wenn die Liste leer ist, wird das Objekt `pContent` in die Liste eingefügt und es gibt weiterhin kein aktuelles Objekt (`hasAccess() == false`).
Falls `pContent` gleich `null` ist, bleibt die Liste unverändert.

Auftrag **`void insert(ContentType pContent)`**

Falls es ein aktuelles Objekt gibt (`hasAccess() == true`), wird ein neues Objekt `pContent` vor dem aktuellen Objekt in die Liste eingefügt. Das aktuelle Objekt bleibt unverändert.
Falls die Liste leer ist und es somit kein aktuelles Objekt gibt (`hasAccess() == false`), wird `pContent` in die Liste eingefügt und es gibt weiterhin kein aktuelles Objekt.
Falls es kein aktuelles Objekt gibt (`hasAccess() == false`) und die Liste nicht leer ist oder `pContent == null` ist, bleibt die Liste unverändert.

Auftrag **`void concat(List<ContentType> pList)`**

Die Liste `pList` wird an die Liste angehängt. Anschließend wird `pList` eine leere Liste. Das aktuelle Objekt bleibt unverändert. Falls es sich bei der Liste und `pList` um dasselbe Objekt handelt, `pList == null` oder eine leere Liste ist, bleibt die Liste unverändert.

Auftrag **`void remove()`**

Falls es ein aktuelles Objekt gibt (`hasAccess() == true`), wird das aktuelle Objekt gelöscht und das Objekt hinter dem gelöschten Objekt wird zum aktuellen Objekt. Wird das Objekt, das am Ende der Liste steht, gelöscht, gibt es kein aktuelles Objekt mehr (`hasAccess() == false`).
Wenn die Liste leer ist oder es kein aktuelles Objekt gibt (`hasAccess() == false`), bleibt die Liste unverändert.

Unterlagen für die Lehrkraft

Abiturprüfung 2017

Informatik, Grundkurs

1. Aufgabenart

Modellierung, Implementation und Analyse kontextbezogener Problemstellungen mit Schwerpunkt auf den Inhaltsfeldern Daten und ihre Strukturierung und Algorithmen

2. Aufgabenstellung[1]

siehe Prüfungsaufgabe

3. Materialgrundlage

entfällt

4. Bezüge zum Kernlehrplan und zu den Vorgaben 2017

Die Aufgaben weisen vielfältige Bezüge zu den Kompetenzerwartungen und Inhaltsfeldern des Kernlehrplans bzw. zu den in den Vorgaben ausgewiesenen Fokussierungen auf. Im Folgenden wird auf Bezüge von zentraler Bedeutung hingewiesen.

1. *Inhaltsfelder und inhaltliche Schwerpunkte*
   Daten und ihre Strukturierung
   - Objekte und Klassen
     - Entwurfsdiagramme und Implementationsdiagramme
     - Lineare Strukturen
       Lineare Liste (Klasse `List`*)*

   Algorithmen
   - Analyse, Entwurf und Implementierung von Algorithmen
   - Algorithmen in ausgewählten informatischen Kontexten

   Formale Sprachen und Automaten
   - Syntax und Semantik einer Programmiersprache
     - Java

2. *Medien/Materialien*
   - entfällt

[1] Die Aufgabenstellung deckt inhaltlich alle drei Anforderungsbereiche ab.

5. Zugelassene Hilfsmittel

- Taschenrechner (graphikfähiger Taschenrechner / CAS-Taschenrechner)
- Wörterbuch zur deutschen Rechtschreibung

6. Modelllösungen

Die jeweilige Modelllösung stellt eine mögliche Lösung bzw. Lösungsskizze dar. Der gewählte Lösungsansatz und -weg der Schülerinnen und Schüler muss nicht identisch mit dem der Modelllösung sein. Sachlich richtige Alternativen werden mit entsprechender Punktzahl bewertet (Bewertungsbogen: Zeile „Sachlich richtige Lösungsalternative zur Modelllösung“).

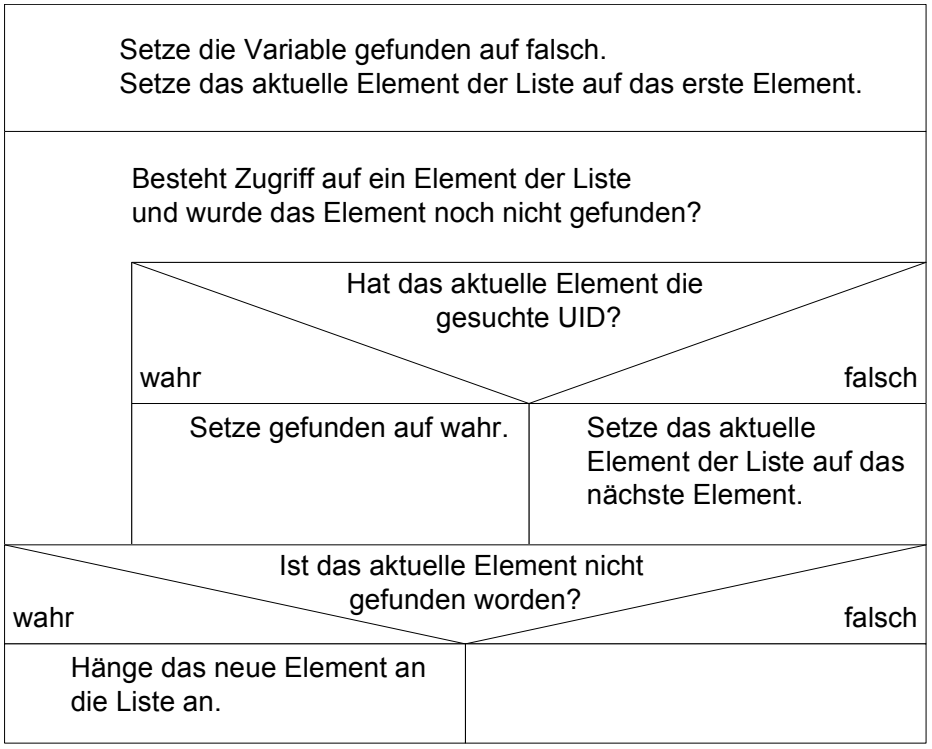
Teilaufgabe a)

Im Implementationsdiagramm in Abbildung 2 sind zwei gerichtete Assoziationen durch Assoziationspfeile dargestellt: Eine Assoziation mit dem Bezeichner `startZeitpunkt` von der Klasse `Termin` zur Klasse `Zeitpunkt` und eine Assoziation mit dem Bezeichner `termine` von der Klasse `TerminVerwaltung` zur parametrisierten Klasse `List`, deren generischer Typ `Termin` entspricht. Beide Assoziationen sind privat deklariert.

Die Assoziation mit dem Bezeichner `termine` ist im Kontext so zu interpretieren, dass die mit einem Objekt der Klasse `TerminVerwaltung` verwalteten Termine in einem Objekt der Klasse `List` verwaltet werden.

Die Assoziation mit dem Bezeichner `startZeitpunkt` bedeutet, dass der Zeitpunkt, an dem ein Termin beginnt, von einem Objekt der Klasse `Termin` verwaltet wird und durch ein Objekt der Klasse `Zeitpunkt` dargestellt wird.

Teilaufgabe b)




```
public void fuegeTerminHinzu(Termin pTermin) {
  boolean gefunden = false;
  termine.toFirst();
  while (termine.hasAccess() && !gefunden) {
    if (termine.getContent().gibUid().equals(pTermin.gibUid())) {
      gefunden = true;
    } else {
      termine.next();
    }
  }
  if (!gefunden) {
    termine.append(pTermin);
  }
}
```

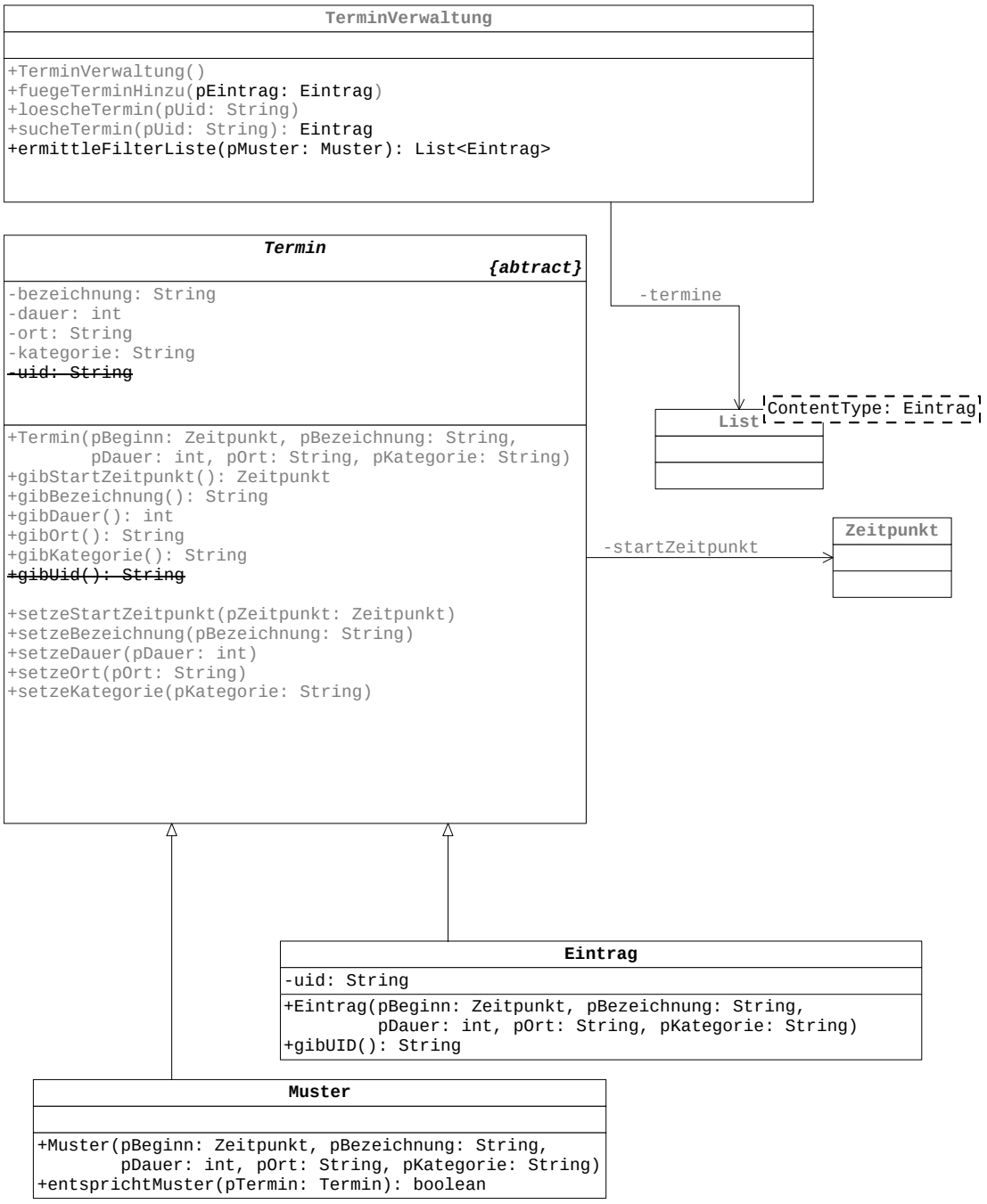
Teilaufgabe c)

Die Methode filtert die Liste der Termine dahingehend, dass alle Termine zurückgegeben werden, die an einem bestimmten Datum liegen. Dieses Filterdatum wird durch das als Parameter übergebene Objekt festgelegt. Die Uhrzeit ist für die Auswahl nicht relevant. Die Ergebnisliste ist aufsteigend nach Uhrzeit geordnet.

Teilaufgabe d)

Relevant sind für das Filtern sind nur die beiden Attribute `ort` und `kategorie`, da bei den anderen Attributen `null` referenziert wird bzw. das Attribut `dauer` den Wert `-1` hat. Für jedes Objekt in der Terminliste müssen also die Attribute `ort` und `kategorie` auf Gleichheit mit den Beispieldaten geprüft werden.

Dies trifft nur auf die Objekte mit UID `abitur:nrw:EEF1-2` und mit UID `abitur:nrw:5B12-`13 zu, die dann beide in einer Ergebnisliste zu finden wären.



Erläuterung der Änderungen:

Die Methoden der Klasse `TerminVerwaltung` werden dahingehend geändert, dass statt der Klasse `Termin` die Unterklasse `Eintrag` verwendet wird. Dies ergibt sich daraus, dass die Klasse `Termin` abstrakt sein soll und die Unterklasse `Eintrag` die Verwaltung des Attributs `uid` übernimmt.
Bei der Klasse `Termin` ändert sich bezüglich des Konstruktors, dass keine UID erzeugt wird.

Dokumentation der ergänzten Methoden:

Klasse `Terminverwaltung`

Anfrage **`List<Eintrag> ermittleFilterListe(Muster pMuster)`**
Die Methode ermittelt eine Liste von Terminen nach dem beschriebenen Filterverfahren: Ein Termin aus der Terminliste ist in der Ergebnisliste enthalten, wenn alle Attribute mit den Attributen des übergebenen Objekts übereinstimmen, sofern die Attribute des übergebenen Objekts nicht `null` referenzieren bzw. den Wert `-1` haben. Die Objekte in der Ergebnisliste befinden sich in der gleichen Reihenfolge wie in der Terminliste. Wird ein Objekt der Klasse `Termin` übergeben, dessen UID nicht `null` referenziert, wird eine leere Liste zurückgegeben.

Die Klasse `Eintrag`

Konstruktor
```
Eintrag(Zeitpunkt pBeginn, String pBezeichnung,
        int pDauer, String pOrt, String pKategorie)
```
Nach dem Aufruf des Konstruktors sind die übergebenen Werte der Parameter und die übergebenen Referenzen der Parameter gespeichert. Das Objekt hat eine eindeutige UID. Referenzieren die Parameter `pBeginn`, `pBezeichnung`, `pOrt` oder `pKategorie` `null`, so wird stattdessen ein leerer String gespeichert. Ist der Wert von `pDauer` kleiner als `0`, so wird `0` gespeichert.

Anfrage **`String gibUID()`**
Die Methode gibt die UID des Eintrags zurück.

Die Klasse `Muster`

Konstruktor `Muster(Zeitpunkt pBeginn, String pBezeichnung, int pDauer, String pOrt, String pKategorie)`

Nach dem Aufruf des Konstruktors sind die übergebenen Werte der Parameter und die übergebenen Referenzen der Parameter gespeichert.

Anfrage `boolean entsprichtMuster(Termin pTermin)`

Die Methode prüft, ob das als Parameter übergebene Objekt der Klasse `Eintrag` dem vorgegebenen Muster entspricht: Abgesehen von der UID des übergebenen Eintrags werden die Ausprägungen aller geerbten Attribute auf Gleichheit mit denen im Muster geprüft: sind diese identisch, wird `true` zurückgegeben, andernfalls `false`.

Teilaufgabe e)

Eine solche Auswahl von Terminen ist mit dem beschriebenen Filtersystem nicht möglich, da jedes Terminobjekt für ein Attribut nur ein Datum speichern kann, nicht mehrere. Damit kann bei einem Vergleich zwischen dem Filterobjekt und den gespeicherten Terminen nur auf Gleichheit oder Ungleichheit geprüft werden. Es muss also eine Möglichkeit geboten werden, auf Gleichheit zu mehreren Daten bzw. Zugehörigkeit zu einem Intervall zu prüfen.

Eine Möglichkeit hierfür wäre, eine neue Klasse `FilterTermin` zu entwerfen und ein Objekt dieser Klasse der Methode `ermittleFilterListe` als Parameter zu übergeben. Der Unterschied der Klasse `FilterTermin` zur Klasse `Termin` liegt darin, dass für jedes Attribut (abgesehen von der UID) in der Klasse `FilterTermin` ein Ausdruck mit Platzhaltern angelegt wird – dieser kann durch eine Zeichenkette oder eine eigene Klasse repräsentiert werden. Ist die Ausprägung eines Attributs eines Objekts der Klasse `Termin` nun durch den entsprechenden Ausdruck darstellbar, wird der Termin in die Ergebnisliste aufgenommen, andernfalls nicht.

7. Teilleistungen – Kriterien / Bewertungsbogen zur Prüfungsarbeit

Name des Prüflings:_____ Kursbezeichnung:_____

Schule: _____

Teilaufgabe a)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	beschreibt und erläutert die Beziehung zwischen `Termin` und `Zeitpunkt`.	2			
2	beschreibt und erläutert die Beziehung zwischen `TerminVerwaltung` und `List`.	4			
Sachlich richtige Lösungsalternative zur Modelllösung: (6) …… ……					
	Summe Teilaufgabe a)	**6**			

Teilaufgabe b)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**[2]	**ZK**	**DK**
1	stellt den entwickelten Algorithmus als Struktogramm dar.	7			
2	implementiert das Durchsuchen der Liste.	4			
3	implementiert das Einfügen.	3			
Sachlich richtige Lösungsalternative zur Modelllösung: (14) …… ……					
	Summe Teilaufgabe b)	**14**			

[2] EK = Erstkorrektur; ZK = Zweitkorrektur; DK = Drittkorrektur

Teilaufgabe c)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	dokumentiert den Durchlauf durch die Liste `termine`.	4			
2	dokumentiert den Durchlauf durch die Liste `ergebnis`.	4			
3	erläutert im Sachzusammenhang, was die Methode leistet.	2			
Sachlich richtige Lösungsalternative zur Modelllösung: (10) …… ……					
	Summe Teilaufgabe c)	**10**			

Teilaufgabe d)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	ermittelt die Ergebnisliste.	2			
2	modifiziert das Diagramm.	4			
3	erläutert die Veränderungen bei bestehenden Methoden.	3			
4	dokumentiert die ergänzten Methoden.	4			
Sachlich richtige Lösungsalternative zur Modelllösung: (13) …… ……					
	Summe Teilaufgabe d)	**13**			

Teilaufgabe e)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	begründet, dass eine solche Filterung nicht möglich ist.	3			
2	erläutert eine mögliche Veränderung des Filtermodells.	4			
Sachlich richtige Lösungsalternative zur Modelllösung: (7) …… ……					
	Summe Teilaufgabe e)	**7**			

	Summe insgesamt	**50**			

Festlegung der Gesamtnote (Bitte nur bei der letzten bearbeiteten Aufgabe ausfüllen.)

	Lösungsqualität			
	maximal erreichbare Punktzahl	EK	ZK	DK
Übertrag der Punktsumme aus der ersten bearbeiteten Aufgabe	**50**			
Übertrag der Punktsumme aus der zweiten bearbeiteten Aufgabe	**50**			
Punktzahl der gesamten Prüfungsleistung	**100**			
aus der Punktsumme resultierende Note gemäß nachfolgender Tabelle				
Note ggf. unter Absenkung um bis zu zwei Notenpunkte gemäß § 13 Abs. 2 APO-GOSt				
Paraphe				

Berechnung der Endnote nach Anlage 4 der Abiturverfügung auf der Grundlage von § 34 APO-GOSt

Die Klausur wird abschließend mit der Note _____ (____ Punkte) bewertet.

Unterschrift, Datum:

Grundsätze für die Bewertung (Notenfindung)

Für die Zuordnung der Notenstufen zu den Punktzahlen ist folgende Tabelle zu verwenden:

Note	**Punkte**	**Erreichte Punktzahl**
sehr gut plus	15	100 – 95
sehr gut	14	94 – 90
sehr gut minus	13	89 – 85
gut plus	12	84 – 80
gut	11	79 – 75
gut minus	10	74 – 70
befriedigend plus	9	69 – 65
befriedigend	8	64 – 60
befriedigend minus	7	59 – 55
ausreichend plus	6	54 – 50
ausreichend	5	49 – 45
ausreichend minus	4	44 – 40
mangelhaft plus	3	39 – 33
mangelhaft	2	32 – 27
mangelhaft minus	1	26 – 20
ungenügend	0	19 – 0

Name: _____

Abiturprüfung 2017

Informatik, Grundkurs

Aufgabenstellung:

Zur Erforschung der Tiefsee werden häufig ferngesteuerte Tauchroboter eingesetzt. Da sich Funkwellen im Wasser schlecht ausbreiten, wird die Kommunikation mit diesen Robotern in der Regel über ein Kabel realisiert, das die Roboter mit einem Forschungsschiff an der Oberfläche verbindet.

Bei der Neuentwicklung eines solchen Tauchroboters soll auf dieses Kabel verzichtet werden. Dazu muss die Funkkommunikation möglichst effizient, d. h. mit geringer Bandbreite, realisiert werden. Die Entwicklungsabteilung ist sicher, dass alle Befehle an den Roboter mit den folgenden Zeichen realisiert werden können:

```
{ A,B,C,D,E,F,0,1,2,3,4,5,6,7,8,9 }
```

Statt jedes Zeichen mit `8` Bit zu kodieren, wie z. B. im ASCII-Code, soll der in Abbildung 1 gezeigte Kodierungsbaum verwendet werden.

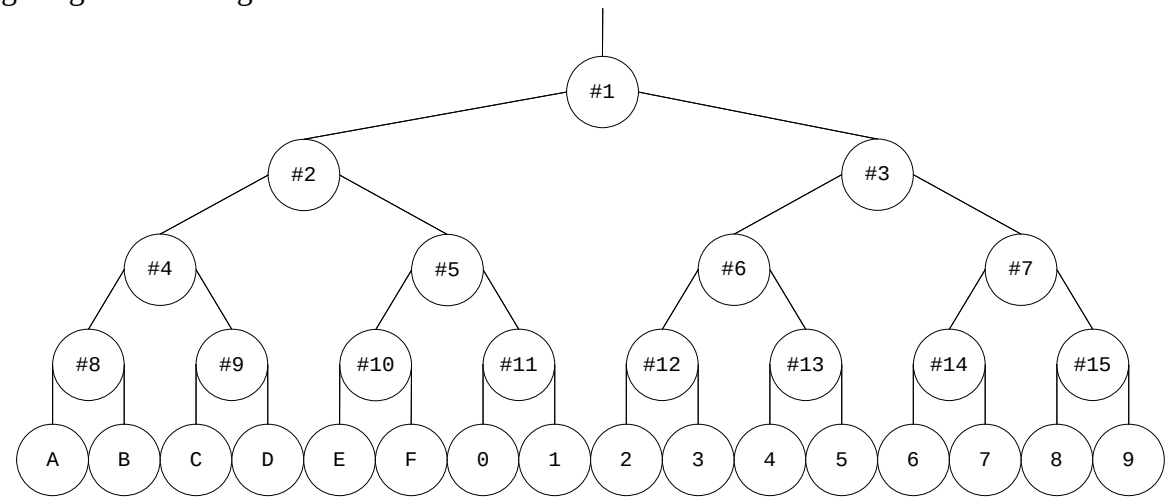


Abbildung 1: Kodierungsbaum zur Kommunikation mit einem Tauchroboter

Die Kodierung eines Zeichens ergibt sich aus dem Pfad von der Wurzel zu dem Blatt, in dem das Zeichen steht. Jeder Wechsel in einen linken Teilbaum steht für eine `0`, ein Wechsel in einen rechten Teilbaum für eine `1`. Für die Kodierung des Buchstaben `C` ergibt sich nach dem Kodierungsbaum in Abbildung 1 somit die Bitfolge `0010`.

Name: _____

a) *Geben Sie die Kodierung der Zeichenfolge* `FD519` *auf Grundlage des Kodierungsbaums in Abbildung 1 an.*

Geben Sie auf Grundlage des Kodierungsbaums in Abbildung 1 an, welche Zeichenfolge mit der folgenden Bitfolge kodiert ist:

```
0100 1010 1000 1101 1111
```

Erläutern Sie allgemein, aus wie vielen Bits die Kodierung einer Zeichenfolge der Länge n besteht, die mit dem Kodierungsbaum in Abbildung 1 erstellt wurde.

(8 Punkte)

Um die Kommunikation zwischen Forschungsschiff und Tauchroboter zu realisieren, soll auf beiden Seiten ein Programm zum Einsatz kommen, das der in Abbildung 2 gegebenen Teilmodellierung entspricht.

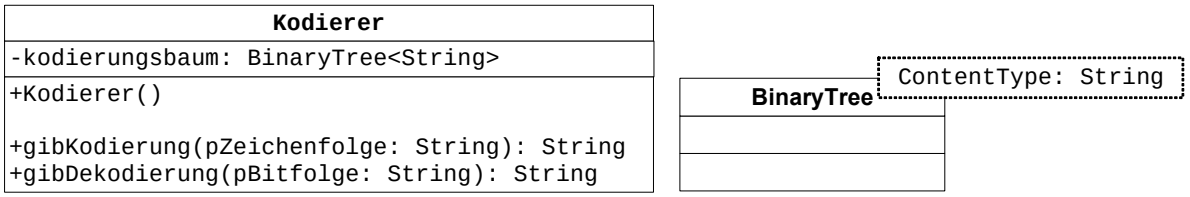


Abbildung 2: Teilmodellierung des Kommunikationsprogramms

Die Methoden `gibKodierung` und `gibDekodierung` verwenden den Kodierungsbaum `kodierungsbaum` vom Typ `BinaryTree<String>` entsprechend der obigen Beschreibung (vgl. Anhang).

b) Die Methode `gibDekodierung` hat den folgenden Methodenkopf:

```
public String gibDekodierung(String pBitfolge)
```

Erläutern Sie ein algorithmisches Verfahren für die Methode `gibDekodierung` *der Klasse* `Kodierer`.

Implementieren Sie die Methode `gibDekodierung` *der Klasse* `Kodierer` *entsprechend der obigen Beschreibung und dem von Ihnen gewählten algorithmischen Verfahren.*

Hinweis: Die Methode `gibDekodierung` muss nur gültige Bitfolgen verarbeiten. Eine Fehlerbehandlung ist nicht erforderlich.

(12 Punkte)

Name: _____

Da man befürchtet, dass auch Unbefugte über die Funkverbindung Anweisungen an den Roboter senden könnten, soll die Kommunikation verschlüsselt werden. Dazu wird die Klasse `Kodierer` entsprechend des Implementationsdiagramms in Abbildung 3 erweitert.

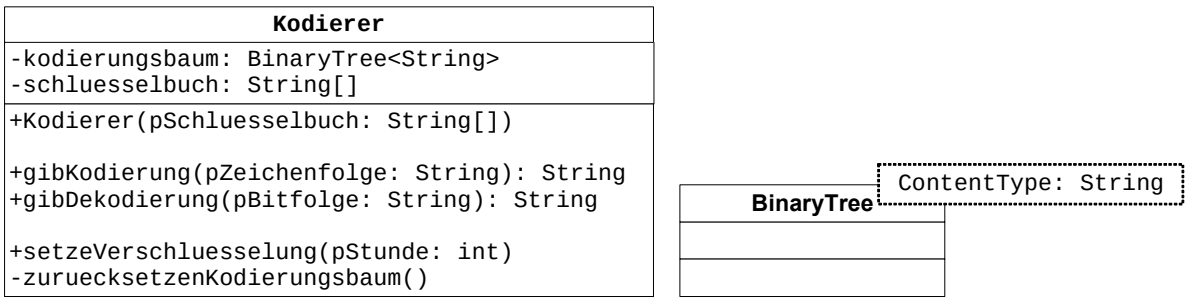


Abbildung 3: Erweiterte Teilmodellierung des Kommunikationsprogramms

Der Tauchroboter führt nun ein „Schlüsselbuch“ mit sich, in dem für jede Stunde des Tages ein Schlüssel in Form von jeweils `15` Bits gespeichert ist. Dieses Schlüsselbuch wird im Konstruktor der Klasse `Kodierer` als Feld von Typ `String` übergeben.

Zur Verschlüsselung wird der Kodierungsbaum aus Abbildung 1 abhängig vom Schlüssel zur aktuellen Stunde nach dem folgenden Verfahren verändert.

Der Kodierungsbaum aus Abbildung 1 wird traversiert, indem zunächst der aktuelle Knoten, anschließend der linke und dann der rechte Teilbaum bearbeitet wird. Auf diese Weise werden alle Knoten des Baums durchlaufen. Relevant für das Verfahren sind nur innere Knoten. Blätter werden nicht berücksichtigt.

Beim Durchlauf durch die inneren Knoten des Kodierungsbaums werden gleichzeitig die einzelnen Bits des aktuellen Schlüssels von links nach rechts durchlaufen. Allen inneren Knoten wird also in der Reihenfolge, in der sie bei der Traversierung bearbeitet werden, ein Bit des aktuellen Schlüssels zugeordnet. Ist das zugeordnete Bit eine `1`, werden die Teilbäume des aktuellen Knotens getauscht. Ist das Bit eine `0`, passiert nichts. Anschließend wird die Traversierung fortgesetzt.

Diese Baummanipulation wird durch die Methode `setzeVerschluesselung` der Klasse `Kodierer` durchgeführt.

c) Angenommen auf den Kodierungsbaum in Abbildung 1 soll einmal der Schlüssel `010000000000000` (Schlüssel `A`) und einmal der Schlüssel `110000000000000` (Schlüssel `B`) angewendet werden.

Geben Sie für beide Schlüssel an, bei welchen Knoten die Teilbäume getauscht werden, und erläutern Sie, warum sich das zweite Bit in beiden Schlüsseln auf unterschiedliche Knoten bezieht.

Name: _____

Entwickeln Sie schrittweise den Kodierungsbaum, der für eine Kommunikation mit dem Tauchroboter um `13:34` *Uhr verwendet würde und stellen Sie in geeigneter Weise dar, wie sich der neue Kodierungsbaum durch den Tausch von Teilbäumen aus dem Schlüssel ergibt. Das entsprechende Schlüsselbuch finden Sie im Anhang.*

(14 Punkte)

d) Das Entwicklerteam hat die Methode `setzeVerschluesselung` der Klasse `Kodierer` bereits fertiggestellt:

```
 1   public void setzeVerschluesselung(int pStunde) {
 2     zuruecksetzenKodierungsbaum();
 3     String schluessel = schluesselbuch[pStunde];
 4     Stack<BinaryTree<String>> speicher =
                                new Stack<BinaryTree<String>>();

 5     speicher.push(kodierungsbaum);
 6     for (int i = 0; i < 15; i++) {
 7       BinaryTree<String> akt = speicher.top();
 8       speicher.pop();

 9       if (schluessel.charAt(i) == '1') {
10         BinaryTree<String> tmp = akt.getLeftTree();
11         akt.setLeftTree(akt.getRightTree());
12         akt.setRightTree(tmp);
13       }

14       if (akt.getRightTree().getContent().charAt(0) == '#') {
15         speicher.push(akt.getRightTree());
16       }
17       if (akt.getLeftTree().getContent().charAt(0) == '#') {
18         speicher.push(akt.getLeftTree());
19       }

20     }
21   }
```

Erläutern Sie das algorithmische Verfahren, nach dem die Methode arbeitet und gehen Sie dabei insbesondere auf die Verwendung des Stapels `speicher` *und die Verzweigungen ab Zeile 9 ein.*

(8 Punkte)

Name: _____

Das Verschlüsselungsverfahren für den Tauchroboter wurde von einigen Mitarbeitern des Entwicklerteam kritisiert:

e) „Auch bei regelmäßigem Austausch des Schlüsselbuchs hängt die Sicherheit der Verschlüsselung von der Anzahl der Kodierungsbäume ab, die nach dem vorgeschlagenen Verfahren prinzipiell möglich sind. Leider sind das verhältnismäßig wenige.

Besser wäre es, einen deutlich längeren Schlüssel mit z. B. `1500` Bit zu verwenden. Diese `1500` Bit sollten dann in jeweils `15`-Bit-Sequenzen unterteilt und nacheinander mit dem bekannten Verfahren auf den Kodierungsbaum angewendet werden."

Erläutern Sie, wie viele verschiedene Kodierungsbäume durch das ursprüngliche Verfahren zustande kommen können.

Beurteilen Sie den Vorschlag, die Schlüssellänge wie oben vorgeschlagen auf `1500` *Bits zu erhöhen.*

(8 Punkte)

Zugelassene Hilfsmittel:

- Taschenrechner (graphikfähiger Taschenrechner / CAS-Taschenrechner)
- Wörterbuch zur deutschen Rechtschreibung

Name: _____

Anlage

Beispiel – Schlüsselbuch mit Feldindex:

Index	Schlüssel
0	100000010000101
1	000100010000000
2	101000001110000
3	001000000000000
4	000100000010010
5	001000000010011
6	000000000011000
7	001000000100010
8	000000100000001
9	001001001000000
10	100000001001000
11	100010000010000
12	110000000000000
13	010010000000000
14	100000011010000
15	001100000000000
16	110000010100000
17	101000000101010
18	010000111000010
19	000000000000000
20	001000000010000
21	001100000100000
22	000010100000001
23	001000000101000

Hinweis: Der Index des Feldes entspricht der Stundenangabe des Zeitpunktes.

Name: _____

Anhang

Dokumentation der Klasse `Kodierer` (Version in Abbildung 2)

Konstruktor **`Kodierer()`**
Erstellt ein Objekt vom Typ `Kodierer`.

Anfrage **`String gibKodierung(String pZeichenfolge)`**
Die Anfrage liefert die Kodierung der Zeichenfolge `pZeichenfolge` entsprechend dem Kodierungsbaum.

Anfrage **`String gibDekodierung(String pBitfolge)`**
Die Anfrage liefert die Dekodierung der Bitfolge `pBitfolge` entsprechend dem Kodierungsbaum.

Dokumentation der Klasse `Kodierer` (Version in Abbildung 3)

Konstruktor **`Kodierer(String[] pSchluesselbuch)`**
Erstellt ein Objekt vom Typ `Kodierer` mit `pSchluesselbuch` als Schlüsselbuch.

Anfrage **`String gibKodierung(String pZeichenfolge)`**
Die Anfrage liefert die Kodierung der Zeichenfolge `pZeichenfolge` entsprechend dem aktuellen Kodierungsbaum.

Anfrage **`String gibDekodierung(String pBitfolge)`**
Die Anfrage liefert die Dekodierung der Bitfolge `pBitfolge` entsprechend dem aktuellen Kodierungsbaum.

Auftrag **`void setzeVerschluesselung(int pStunde)`**
Der Auftrag manipuliert den Kodierungsbaum entsprechend dem zu `pStunde` gehörigen Eintrag im aktuellen Schlüsselbuch.

Des Weiteren verfügen Objekte der Klasse über die folgende private Hilfsmethode:

Auftrag **`void zuruecksetzenKodierungsbaum()`**
Der Auftrag setzt den Kodierungsbaum zurück in seinen Ausgangszustand.

Name: _____

Die generische Klasse Stack<ContentType>

Objekte der generischen Klasse **`Stack`** (Keller, Stapel*)* verwalten beliebige Objekte vom Typ **`ContentType`** nach dem Last-In-First-Out-Prinzip, d. h., das zuletzt abgelegte Objekt wird als erstes wieder entnommen. Alle Methoden haben eine konstante Laufzeit, unabhängig von der Anzahl der verwalteten Objekte.

Dokumentation der generischen Klasse `Stack<ContentType>`

Konstruktor **`Stack()`**

Ein leerer Stapel wird erzeugt. Objekte, die in diesem Stapel verwaltet werden, müssen vom Typ `ContentType` sein.

Anfrage **`boolean isEmpty()`**

Die Anfrage liefert den Wert `true`, wenn der Stapel keine Objekte enthält, sonst liefert sie den Wert `false`.

Auftrag **`void push(ContentType pContent)`**

Das Objekt `pContent` wird oben auf den Stapel gelegt. Falls `pContent` gleich `null` ist, bleibt der Stapel unverändert.

Auftrag **`void pop()`**

Das zuletzt eingefügte Objekt wird von dem Stapel entfernt. Falls der Stapel leer ist, bleibt er unverändert.

Anfrage **`ContentType top()`**

Die Anfrage liefert das oberste Stapelobjekt. Der Stapel bleibt unverändert. Falls der Stapel leer ist, wird `null` zurückgegeben.

Name: _____

Die Klasse BinaryTree<ContentType>

Mithilfe der generischen Klasse **`BinaryTree`** können beliebig viele Objekte vom Typ **`ContentType`** in einem Binärbaum verwaltet werden. Ein Objekt der Klasse stellt entweder einen leeren Baum dar oder verwaltet ein Inhaltsobjekt sowie einen linken und einen rechten Teilbaum, die ebenfalls Objekte der generischen Klasse **`BinaryTree`** sind.

Dokumentation der Klasse `BinaryTree<ContentType>`

Konstruktor **`BinaryTree()`**

Nach dem Aufruf des Konstruktors existiert ein leerer Binärbaum. Objekte, die in diesem Binärbaum verwaltet werden, müssen vom Typ `ContentType` sein.

Konstruktor **`BinaryTree(ContentType pContent)`**

Wenn der Parameter `pContent` ungleich `null` ist, existiert nach dem Aufruf des Konstruktors der Binärbaum und hat `pContent` als Inhaltsobjekt und zwei leere Teilbäume. Falls der Parameter `null` ist, wird ein leerer Binärbaum erzeugt.

Konstruktor
```
BinaryTree(ContentType pContent,
           BinaryTree<ContentType> pLeftTree,
           BinaryTree<ContentType> pRightTree)
```

Wenn der Parameter `pContent` ungleich `null` ist, wird ein Binärbaum mit `pContent` als Inhaltsobjekt und den beiden Teilbäumen `pLeftTree` und `pRightTree` erzeugt. Sind `pLeftTree` oder `pRightTree` gleich `null`, wird der entsprechende Teilbaum als leerer Binärbaum eingefügt. Wenn der Parameter `pContent` gleich `null` ist, wird ein leerer Binärbaum erzeugt.

Anfrage **`boolean isEmpty()`**

Diese Anfrage liefert den Wahrheitswert `true`, wenn der Binärbaum leer ist, sonst liefert sie den Wert `false`.

Auftrag **`void setContent(ContentType pContent)`**

Wenn der Binärbaum leer ist, wird der Parameter `pContent` als Inhaltsobjekt sowie ein leerer linker und rechter Teilbaum eingefügt. Ist der Binärbaum nicht leer, wird das Inhaltsobjekt durch `pContent` ersetzt. Die Teilbäume werden nicht geändert. Wenn `pContent` `null` ist, bleibt der Binärbaum unverändert.

Name: _____

Anfrage **`ContentType getContent()`**

Diese Anfrage liefert das Inhaltsobjekt des Binärbaums. Wenn der Binärbaum leer ist, wird `null` zurückgegeben.

Auftrag **`void setLeftTree(BinaryTree<ContentType> pTree)`**

Wenn der Binärbaum leer ist, wird `pTree` nicht angehängt. Andernfalls erhält der Binärbaum den übergebenen Baum als linken Teilbaum. Falls der Parameter `null` ist, ändert sich nichts.

Auftrag **`void setRightTree(BinaryTree<ContentType> pTree)`**

Wenn der Binärbaum leer ist, wird `pTree` nicht angehängt. Andernfalls erhält der Binärbaum den übergebenen Baum als rechten Teilbaum. Falls der Parameter `null` ist, ändert sich nichts.

Anfrage **`BinaryTree<ContentType> getLeftTree()`**

Diese Anfrage liefert den linken Teilbaum des Binärbaumes. Der Binärbaum ändert sich nicht. Wenn der Binärbaum leer ist, wird `null` zurückgegeben.

Anfrage **`BinaryTree<ContentType> getRightTree()`**

Diese Anfrage liefert den rechten Teilbaum des Binärbaumes. Der Binärbaum ändert sich nicht. Wenn der Binärbaum leer ist, wird `null` zurückgegeben.

Unterlagen für die Lehrkraft

Abiturprüfung 2017

Informatik, Grundkurs

1. Aufgabenart

Analyse, Modellierung und Implementation von kontextbezogenen Problemstellungen mit Schwerpunkt auf den Inhaltsfeldern Daten und ihre Strukturierung und Algorithmen

2. Aufgabenstellung[1]

siehe Prüfungsaufgabe

3. Materialgrundlage

entfällt

4. Bezüge zum Kernlehrplan und zu den Vorgaben 2017

Die Aufgaben weisen vielfältige Bezüge zu den Kompetenzerwartungen und Inhaltsfeldern des Kernlehrplans bzw. zu den in den Vorgaben ausgewiesenen Fokussierungen auf. Im Folgenden wird auf Bezüge von zentraler Bedeutung hingewiesen.

1. Inhaltsfelder und inhaltliche Schwerpunkte

Daten und ihre Strukturierung
- Objekte und Klassen
  - Nicht-lineare Strukturen
    Binärbaum (Klasse `BinaryTree`*)*

Algorithmen
- Analyse, Entwurf und Implementierung von Algorithmen

Formale Sprachen und Automaten
- Syntax und Semantik einer Programmiersprache
  - Java

2. Medien/Materialien
- entfällt

[1] Die Aufgabenstellung deckt inhaltlich alle drei Anforderungsbereiche ab.

5. Zugelassene Hilfsmittel

- Taschenrechner (graphikfähiger Taschenrechner / CAS-Taschenrechner)
- Wörterbuch zur deutschen Rechtschreibung

6. Modelllösungen

Die jeweilige Modelllösung stellt eine mögliche Lösung bzw. Lösungsskizze dar. Der gewählte Lösungsansatz und -weg der Schülerinnen und Schüler muss nicht identisch mit dem der Modelllösung sein. Sachlich richtige Alternativen werden mit entsprechender Punktzahl bewertet (Bewertungsbogen: Zeile „Sachlich richtige Lösungsalternative zur Modelllösung“).

Teilaufgabe a)

Die Kodierung der Zeichenfolge `FD519` mit dem Kodierungsbaum in Abbildung 1 ist die folgende:

```
0101 0011 1011 0111 1111
```

Die gegebene Bitfolge kodiert die Zeichen: `E4279`

Der Kodierungsbaum in Abbildung 1 ist ein vollständiger Binärbaum der Tiefe vier. Alle Zeichen befinden sich in Blättern, so dass die Kodierung jedes Zeichens mit vier Bits erfolgt. Bei `n` Zeichen werden also `4n` Bits benötigt.

Teilaufgabe b)

Jeweils vier Bits der in `pBitfolge` übergebenen Bitfolge stellen ein Zeichen der zu dekodierenden Nachricht dar. Die Methode muss nacheinander alle `4`-Bit-Sequenzen der Bitfolge `pBitfolge` durchlaufen und dekodieren. Die Dekodierung erfolgt, indem sie abhängig von dem aktuellen Bit der `4`-Bit-Sequenz durch den Kodierungsbaum bis zu einem Blatt läuft. In diesem Blatt ist das dekodierte Zeichen zu finden.

Zunächst wird eine Zählschleife realisiert, deren Rumpf so oft durchlaufen wird, wie die zu dekodierende Nachricht Zeichen hat, d. h. bei `n` Bits `n/4` mal. In einem Durchlauf dieser Schleife wird genau ein Zeichen abgearbeitet.

Innerhalb dieser Schleife wird die Wurzel des Kodierungsbaums in einer Hilfsreferenzvariable, z. B. `tmp`, gespeichert. Anschließend wird eine innere Zählschleife realisiert, die genau viermal durchläuft und alle Bits, die zu einem Zeichen gehören, abarbeitet. Bei jedem Durchlauf wird von links nach rechts ein weiteres Bit der Bitfolge `pBitfolge` betrachtet. Die Position dieses Bit in der gesamten Bitfolge `pBitfolge` kann mit Hilfe der Zähler beider Schleifen errechnet werden. Ist das aktuelle Bit eine `0`, wird die Hilfsreferenzvariable `tmp` auf den linken Teilbaum von `tmp` gesetzt, ansonsten auf den rechten Teilbaum von `tmp`.

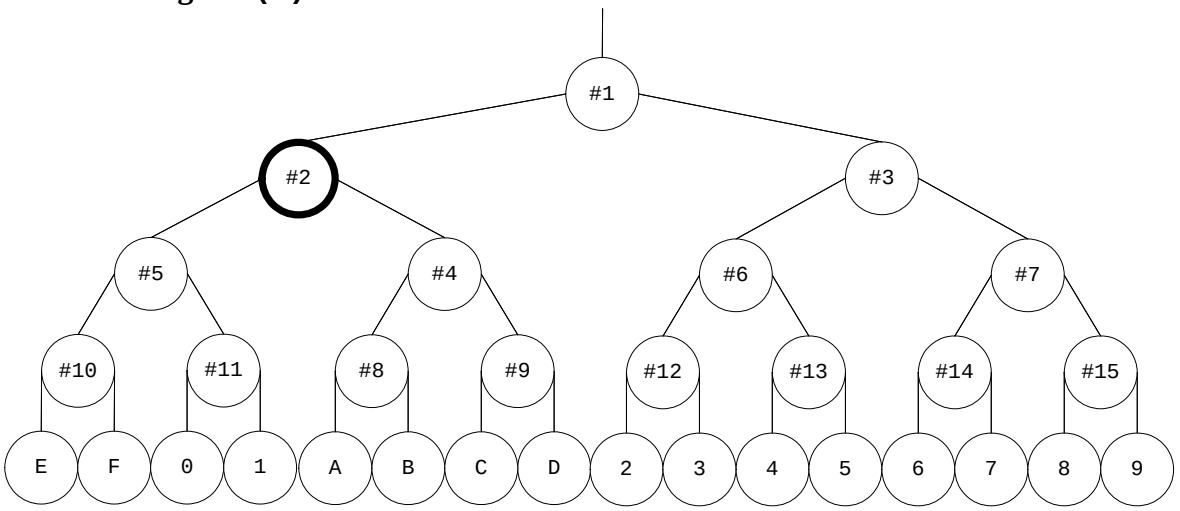
Nachdem die innere Schleife beendet ist, muss das Zeichen in `tmp` an das Ergebnis der Methode angehängt werden.

```
public String gibDekodierung(String pBitfolge) {
  //Initialisierung einer Ergebnisvariable
  String ergebnis = "";
  //Zaehlschleife, die alle Zeichen abarbeitet
  for (int z = 0; z < pBitfolge.length(); z = z+4) {
    //Wurzel des Kodierungsbaum wird gespeichert.
    BinaryTree<String> tmp = kodierungsbaum;
    //Alle Bits eines Zeichens werden durchlaufen.
    for (int b = 0; b < 4; b = b+1) {
      //Die Referenz tmp wechselt in den richtigen Teilbaum.
      if (pBitfolge.charAt(z + b) == '0') {
        tmp = tmp.getLeftTree();
      } else {
        tmp = tmp.getRightTree();
      }
    }
    //Inhalt des Blatts tmp wird an das Ergebnis angefuegt.
    ergebnis = ergebnis + tmp.getContent();
  }
  //Ergebnisvariable wird zurueckgeliefert.
  return ergebnis;
}
```

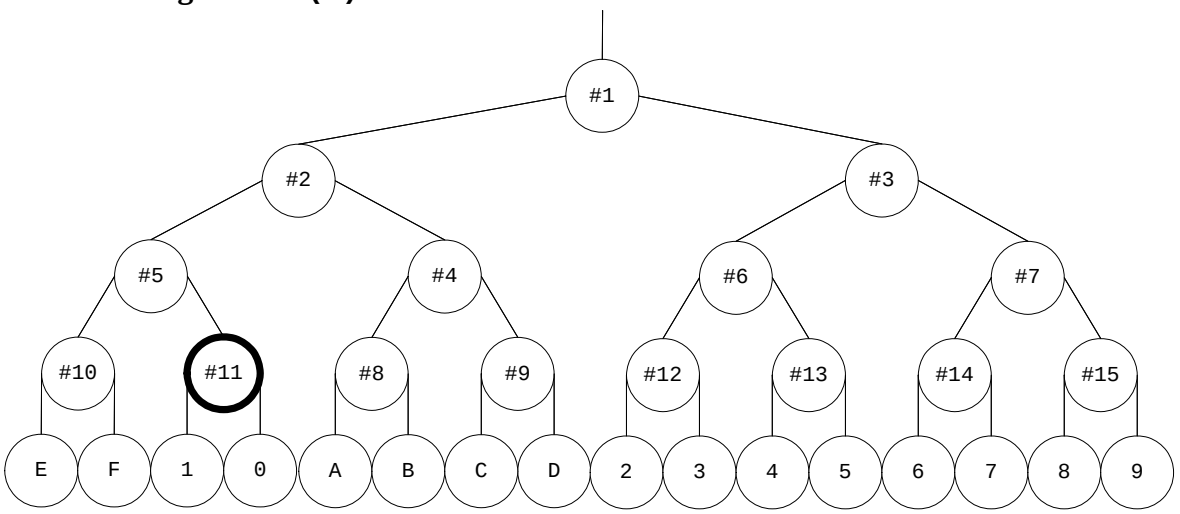
Teilaufgabe c)

Bei Verwendung von Schlüssel `A` (`010000000000000`) werden die Teilbäume des Knotens `#2` getauscht. Bei Verwendung von Schlüssel `B` (`110000000000000`) werden die Teilbäume der Knoten `#1` und `#3` getauscht.
Das zweite Bit der beiden Schlüssel bezieht sich in beiden Fällen auf den zweiten Knoten der Präordertraversierung des Kodierungsbaums. Bei einer Präodertraversierung gegen den Uhrzeigersinn ist das der linke Nachfolger der Wurzel. Bei Schlüssel `A` ist das der Knoten `#2`. Bei Schlüssel `B` werden die Teilbäume der Wurzel getauscht, bevor weiter traversiert wird. Daher bezieht sich das zweite Bit von Schlüssel `B` auf `#3`, den ursprünglich rechten Nachfolger der Wurzel, der durch die Vertauschung zum linken Nachfolger wurde.
Eine Kommunikation um `13.34` Uhr würde mit dem Schlüssel `010010000000000` verschlüsselt. Dieser Schlüssel ist an der Indexstelle `13` des Schlüsselbuchs zu finden. Aufgrund der zwei Einsen in diesem Schlüssel ergibt sich der folgende Kodierungsbaum durch zwei Vertauschungen:

Vertauschung 1: `0(1)0010000000000`



Vertauschung 2: `0100(1)0000000000`



Teilaufgabe d)

Die Methode `setzeVerschluesselung` führt mit einem iterativen Verfahren eine Präordertraversierung des Kodierungsbaum durch und vertauscht abhängig vom Schlüssel zur übergebenen Stunde `pStunde` Teilbäume.

In Zeile 2 wird der Kodierungsbaum auf seinen ursprünglichen Zustand gesetzt, damit ein früher verwendeter Schlüssel keinen Einfluss auf die aktuelle Verschlüsselung nimmt. Anschließend wird der Schlüssel zur Stunde `pStunde` aus dem Feld `schluesselbuch` ausgelesen und in dem lokalen String `schluessel` gespeichert. Des Weiteren wird ein Stapelspeicher mit dem Bezeichner `speicher` für Binärbäume mit Strings als Inhalten erstellt. Anschließend wird von Zeile `5` bis Zeile `20` die eigentliche Traversierung und Manipulation des Baums durchgeführt.

Zu Beginn wird der gesamte Kodierungsbaum auf den Stapel `speicher` gelegt. Anschließend werden in einer Zählschleife (Z. 6 – 20) alle Bits des aktuellen Schlüssels von Bit `0` bis Bit `15` auf den Baum angewendet. Bei jedem Schleifendurchlauf werden folgende Operationen mit jeweils dem nächsten Bit des Schlüssels durchgeführt:
Zunächst wird der oberste Binärbaum aus dem Stapel `speicher` ausgelesen und entfernt (Z. 7 f). In den Zeilen 9 bis 13 werden seine Teilbäume getauscht, sofern das aktuell anzuwendende Bit des Schlüssels eine `1` ist. Anschließend werden in den Zeilen 14 bis 19 seine Teilbäume auf den Stapelspeicher `speicher` gelegt, sofern die Inhalte ihrer Wurzelknoten mit dem Zeichen # beginnen und es sich somit um innere Knoten handelt. Darüber hinaus wird zunächst der rechte Teilbaum und dann der linke Teilbaum auf den Stapel gelegt, damit entsprechend der Funktion eines Stapels der linke Teilbaum beim nächsten Schleifendurchlauf zuerst verarbeitet wird und somit eine Traversierung gegen den Uhrzeigersinn zustande kommt.

Teilaufgabe e)

Jeder Schlüssel steht für genau einen Kodierungsbaum. Da der Schlüssel `15` Bits umfasst, gibt es 2^{15} verschiedene Schlüssel und damit auch Kodierungsbäume.
Der Verbesserungsvorschlag des Mitarbeiters entspricht der Idee nacheinander `100` mal einen `15`-Bit-Schlüssel auf den Kodierungsbaum anzuwenden, ohne ihn dazwischen in seinen Ursprungszustand zurückzuversetzen.
Jeder `15`-Bit-Schlüssel gibt für jeden inneren Knoten des Kodierungsbaums an, ob seine Teilbäume getauscht werden sollen. Die Anwendung jedes weiteren `15`-Bit-Schlüssels führt entweder zusätzliche Vertauschungen durch, die auch mit dem vorangegangenen Schlüssel hätten realisiert werden können, oder nimmt bereits durchgeführte Vertauschungen zurück. Das Ergebnis einer Baummanipulation mit einem `1500`-Bit-Schlüssel ist also ein Baum, der auch mit einem `15`-Bit-Schlüssel hätte erzeugt werden können. Der Vorschlag des Mitarbeiters stellt daher keine Verbesserung dar.

7. Teilleistungen – Kriterien / Bewertungsbogen zur Prüfungsarbeit

Name des Prüflings:_____ Kursbezeichnung:_____

Schule: _____

Teilaufgabe a)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK[2]**	**ZK**	**DK**
1	gibt die Kodierung der Zeichenfolge an.	3			
2	gibt an, welche Zeichenfolge mit der Bitfolge kodiert ist.	3			
3	erläutert allgemein, aus wie vielen Bits die Kodierung einer Zeichenfolge mit n Bits besteht.	2			
Sachlich richtige Lösungsalternative zur Modelllösung: (8) …… ……					
	Summe Teilaufgabe a)	**8**			

Teilaufgabe b)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	erläutert ein algorithmisches Verfahren für die Methode.	4			
2	implementiert die Methode.	8			
Sachlich richtige Lösungsalternative zur Modelllösung: (12) …… ……					
	Summe Teilaufgabe b)	**12**			

[2] EK = Erstkorrektur; ZK = Zweitkorrektur; DK = Drittkorrektur

Teilaufgabe c)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	gibt für beide Schlüssel an, bei welchen Knoten die Teilbäume getauscht werden.	2			
2	erläutert, warum sich das zweite Bit in beiden Schlüsseln auf unterschiedliche Knoten bezieht.	4			
3	entwickelt schrittweise den Kodierungsbaum und stellt in geeigneter Weise dar, wie er sich durch den Tausch von Teilbaumen ergibt.	8			
Sachlich richtige Lösungsalternative zur Modelllösung: (14) …… ……					
	Summe Teilaufgabe c)	**14**			

Teilaufgabe d)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	erläutert das algorithmische Verfahren.	8			
Sachlich richtige Lösungsalternative zur Modelllösung: (8) …… ……					
	Summe Teilaufgabe d)	**8**			

Teilaufgabe e)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	erläutert, wie viele Kodierungsbäume zustande kommen können.	2			
2	beurteilt den Vorschlag, die Schlüssellänge zu erhöhen.	6			
Sachlich richtige Lösungsalternative zur Modelllösung: (8) …… ……					
	Summe Teilaufgabe e)	**8**			

	Summe insgesamt	**50**			

Festlegung der Gesamtnote (Bitte nur bei der letzten bearbeiteten Aufgabe ausfüllen.)

	Lösungsqualität			
	maximal erreichbare Punktzahl	EK	ZK	DK
Übertrag der Punktsumme aus der ersten bearbeiteten Aufgabe	**50**			
Übertrag der Punktsumme aus der zweiten bearbeiteten Aufgabe	**50**			
Punktzahl der gesamten Prüfungsleistung	**100**			
aus der Punktsumme resultierende Note gemäß nachfolgender Tabelle				
Note ggf. unter Absenkung um bis zu zwei Notenpunkte gemäß § 13 Abs. 2 APO-GOSt				
Paraphe				

Berechnung der Endnote nach Anlage 4 der Abiturverfügung auf der Grundlage von § 34 APO-GOSt

Die Klausur wird abschließend mit der Note _____ (____ Punkte) bewertet.

Unterschrift, Datum:

Grundsätze für die Bewertung (Notenfindung)

Für die Zuordnung der Notenstufen zu den Punktzahlen ist folgende Tabelle zu verwenden:

Note	**Punkte**	**Erreichte Punktzahl**
sehr gut plus	15	100 – 95
sehr gut	14	94 – 90
sehr gut minus	13	89 – 85
gut plus	12	84 – 80
gut	11	79 – 75
gut minus	10	74 – 70
befriedigend plus	9	69 – 65
befriedigend	8	64 – 60
befriedigend minus	7	59 – 55
ausreichend plus	6	54 – 50
ausreichend	5	49 – 45
ausreichend minus	4	44 – 40
mangelhaft plus	3	39 – 33
mangelhaft	2	32 – 27
mangelhaft minus	1	26 – 20
ungenügend	0	19 – 0

Name: _____

Abiturprüfung 2017

Informatik, Grundkurs

Aufgabenstellung:

Am Edgar-F.-Codd-Gymnasium soll als Angebot im Rahmen der individuellen Förderung die Teilnahme an Wettbewerben unterstützt und begleitet werden. Ein Wettbewerb ist eine Veranstaltung, bei der die Teilnehmerinnen und Teilnehmer gegeneinander antreten, um ihre Leistungen miteinander zu vergleichen, und bei der es für die besten Preise gibt. Wettbewerbe haben eindeutige Bezeichnungen und die Teilnahmebedingungen sind häufig auf einer Website beschrieben. Einige Wettbewerbe werden in unterschiedlichen Runden (Schulrunde, Regionalrunde etc.) ausgetragen.
Der Informatikkurs der Q1 wird damit beauftragt, eine Datenbanklösung für die Erfassung der relevanten Daten zu entwickeln. Der Kurs macht folgenden Vorschlag:

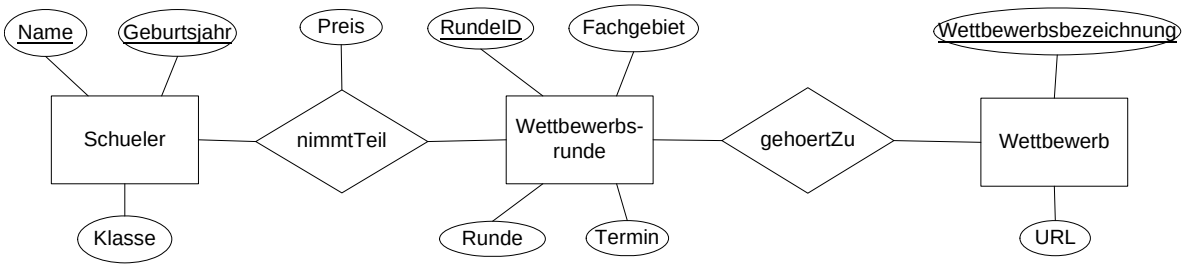


Abbildung 1: ER-Diagramm

```
Schueler(Name, Geburtsjahr, Klasse, ↑RundeID, Preis)
Wettbewerbsrunde(RundeID, Fachgebiet, Runde, Termin)
Wettbewerb(Wettbewerbsbezeichnung, URL)
gehoertZu(↑RundeID, ↑Wettbewerbsbezeichnung)
```

Abbildung 2: Datenbankschema

Eine Beispieldatenbank zu dem Datenbankschema befindet sich im Anhang.

Dieses Schema soll in den nächsten Teilaufgaben auf Praxistauglichkeit getestet werden.

Name: _____

a) *Erläutern Sie die Informationen, die über Heike Heigel in der Beispieldatenbank stehen, im Sachzusammenhang.*

(4 Punkte)

b) Um die Beispieldatenbank zu testen, werden folgende Anfragen ausprobiert:

(i)
```
1 SELECT Schueler.Name
2 FROM Schueler, Wettbewerbsrunde
3 WHERE Schueler.RundeID = WettbewerbsRunde.RundeID
  AND WettbewerbsRunde.Fachgebiet = "Mathematik"
```

(ii)
```
1 SELECT Wettbewerb.Fachgebiet, COUNT(*) AS Anzahl
2 FROM Wettbewerbsrunde
3 GROUP BY Wettbewerbsrunde.Fachgebiet
```

(iii)
```
1 SELECT Wettbewerb.Wettbewerbsbezeichnung,
                                      Wettbewerbsrunde.Termin
2 FROM (Wettbewerb
3   JOIN gehoertZu
      ON gehoertZu.Wettbewerbsbezeichnung =
                          Wettbewerb.Wettbewerbsbezeichnung)
4   JOIN Wettbewerbsrunde
      ON gehoertZu.RundeID = Wettbewerbsrunde.RundeID
```

Anmerkung: In einigen Datenbanksystemen muss das Schlüsselwort `JOIN` durch `INNER JOIN` ersetzt werden.

Analysieren Sie die SQL-Anweisungen und bestimmen Sie die Informationen, die in der Datenbank gesucht werden.

(15 Punkte)

Name: _____

c) Die folgenden Anfragen sollen in SQL umgesetzt werden:

(I) Es sollen alle Oberstufenschülerinnen und -schüler (Klasse EF, Q1, Q2) ermittelt werden, die an einem Wettbewerb teilgenommen haben.
(II) Gesucht sind die Termine, an denen eine Schülerin oder ein Schüler einen 1. Platz belegt hat. Zusätzlich soll das Fachgebiet mit ausgegeben und das Ergebnis nach diesen sortiert werden.
(III) Es sollen die Bezeichnungen der Wettbewerbe ermittelt werden, an denen noch keine Schülerin bzw. kein Schüler der Schule teilgenommen hat.

Entwickeln Sie für die drei genannten Suchaufträge geeignete SQL-Anweisungen, mit deren Hilfe die geforderten Informationen verfügbar werden.

(15 Punkte)

d) Folgende Informationen sollen in die Beispieldatenbank aufgenommen werden:

(i) Der weltweite RoboCup-Wettbewerb findet am 01.07.2017 in Leipzig statt. Genauere Informationen gibt es unter www.robocup.org.
(ii) Marie Winter, am 13.02.2003 geboren, aus der Klasse 9b hat am 24.02.2017 am Regionalwettwerb Jugend forscht mit einem Beitrag zur Informatik teilgenommen und den 1. Platz erreicht.
(iii) Heike Heigel hat am 12.04.2017 am Landeswettbewerb Jugend forscht im Fach Technik den 1. Platz erreicht.

Geben Sie an, wie die Eintragungen in der Beispieldatenbank ergänzt werden müssen, und erläutern Sie diese.

(9 Punkte)

Name: _____

e) Die Arbeit mit dem Entwurf und die Umsetzung im Datenbankschema der Schülerinnen und Schüler der Q1 zeigen Schwachstellen. Daher sollen der Entwurf und die Umsetzung in das Datenbankschema grundsätzlich überarbeitet werden. Dazu wird das folgende ER-Diagramm entworfen:

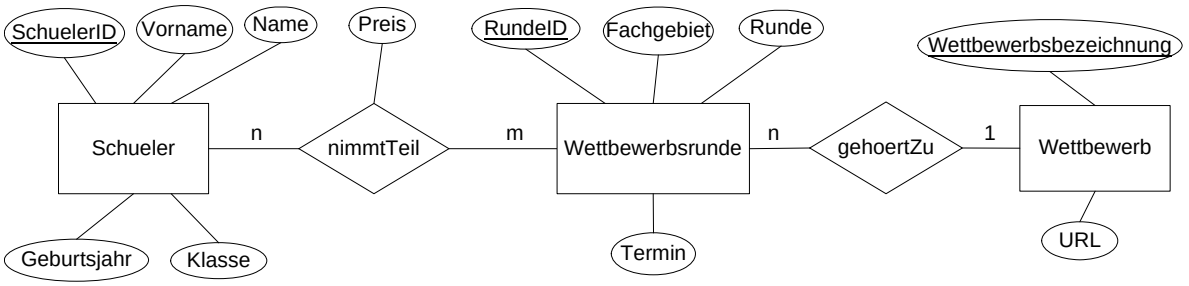


Abbildung 3: ER-Diagramm

Das neue Datenbankschema soll außerdem so aufgebaut sein, dass es der dritten Normalform genügt.

Begründen Sie, dass der ursprünglichen Entwurf (Abbildung 1) fehlerhaft in ein Datenbankschema (Abbildung 2) übertragen worden ist, und erläutern Sie, worin die Verbesserung dieses Entwurfs gegenüber dem ursprünglichen Entwurf besteht.

Überführen Sie das neue ER-Modell in ein Datenbankschema, das in dritter Normalform ist, und erläutern Sie Ihre Entscheidungen.

(7 Punkte)

Zugelassene Hilfsmittel:

- Taschenrechner (graphikfähiger Taschenrechner / CAS-Taschenrechner)
- Wörterbuch zur deutschen Rechtschreibung

Name: _____

Anlage: Auszug aus der Beispieldatenbank

Schueler				
Name	**Geburtsjahr**	**Klasse**	**RundeID**	**Preis**
Eva Hansen	2004	08C	1	Teilnahme
Friedrich Meister	2004	08A	3	Platz 2
Heike Heigel	2002	EF	1	Platz 1
Thomas Meyer	1999	Q2	2	Platz 1

Wettbewerbsrunde			
RundeID	**Fachgebiet**	**Runde**	**Termin**
1	Technik	regional	24.02.2017
2	Informatik	bundesweit	15.11.2016
3	Mathematik	lokal	07.10.2016
4	Mathematik	lokal	24.03.2017

Wettbewerb	
Wettbewerbsbezeichnung	**URL**
Jugend forscht	www.jugend-forscht.de
Informatik-Biber	informatik-biber.de
Mathematik Olympiade	www.mathematik-olympiaden.de
Känguru Wettbewerb	www.mathe-kaenguru.de
Bundeswettbewerb Informatik	www.bwinf.de

gehoertZu	
RundeID	**Wettbewerbsbezeichnung**
1	Jugend forscht
2	Informatik-Biber
3	Mathematik Olympiade
4	Känguru Wettbewerb

Unterlagen für die Lehrkraft

Abiturprüfung 2017

Informatik, Grundkurs

1. Aufgabenart

Analyse, Modellierung und Abfrage relationaler Datenbanken

2. Aufgabenstellung[1]

siehe Prüfungsaufgabe

3. Materialgrundlage

entfällt

4. Bezüge zum Kernlehrplan und zu den Vorgaben 2017

Die Aufgaben weisen vielfältige Bezüge zu den Kompetenzerwartungen und Inhaltsfeldern des Kernlehrplans bzw. zu den in den Vorgaben ausgewiesenen Fokussierungen auf. Im Folgenden wird auf Bezüge von zentraler Bedeutung hingewiesen.

1. *Inhaltsfelder und inhaltliche Schwerpunkte*
   Daten und ihre Strukturierung
   - Datenbanken

   Formale Sprachen und Automaten
   - Syntax und Semantik einer Programmiersprache
     - SQL

2. *Medien/Materialien*
   - entfällt

5. Zugelassene Hilfsmittel

- Taschenrechner (graphikfähiger Taschenrechner / CAS-Taschenrechner)
- Wörterbuch zur deutschen Rechtschreibung

[1] Die Aufgabenstellung deckt inhaltlich alle drei Anforderungsbereiche ab.

6. Modelllösungen

Die jeweilige Modelllösung stellt eine mögliche Lösung bzw. Lösungsskizze dar. Der gewählte Lösungsansatz und -weg der Schülerinnen und Schüler muss nicht identisch mit dem der Modelllösung sein. Sachlich richtige Alternativen werden mit entsprechender Punktzahl bewertet (Bewertungsbogen: Zeile „Sachlich richtige Lösungsalternative zur Modelllösung“).

Teilaufgabe a)

Heike Heigel wurde im Jahr 2002 geboren und besucht die Klasse EF. Sie hat am 24.02.2017 am Regionalwettbewerb Jugend forscht im Fachgebiert Technik teilgenommen und den 1. Platz erreicht.

Teilaufgabe b)

(i) Die Abfrage sucht die Namen aller Schülerinnen und Schüler, die an einer Veranstaltung aus dem Fachgebiet Mathematik teilgenommen haben.
Zunächst werden die Tabellen `Schueler` und `Wettbewerbsrunde` miteinander kombiniert (kartesisches Produkt). Anschließend werden die Zeilen gesucht, in denen `RundeID` der Ausgangstabellen übereinstimmt und das zugehörige Fachgebiet der Mathematik entspricht.

(ii) Die Abfrage gibt die Anzahl der Veranstaltungen in jedem Fachgebiet zusammen mit dem jeweiligen Fachgebiet aus.
In der Abfrage werden die Einträge in der Tabelle `Wettbewerbsrunde` zunächst nach einzelnen Fachgebieten zusammengefasst und gruppiert. Anschließend wird jedes Fachgebiet mit der Anzahl der Elemente jeder Gruppe – `Anzahl` genannt – ausgegeben.

(iii) Die Abfrage gibt die Bezeichnungen aller Wettbewerbe zusammen mit ihren jeweiligen Terminen (bzw. mit den Terminen einer Veranstaltung dieses Wettbewerbs) aus.
Die Tabellen `Wettbewerb`, `gehoertZu` und `Wettbewerbsrunde` werden durch `JOIN` miteinander verbunden. Zuerst werden die Tabellen `Wettbewerb` und `gehoertZu` mit Hilfe der beiden Attribute `Wettbewerbsbezeichnung` verbunden, anschließend wird die entstehende Tabelle mit `Wettbewerbsstufe` anhand der beiden Attribute `RundeID` verbunden.
Aus der Ergebnistabelle, die die drei Ausgangstabellen wie beschrieben miteinander verbunden beinhaltet, werden die gewünschten Spalten ausgegeben.

Teilaufgabe c)

Anmerkung: In einigen Datenbanksystemen muss das Schlüsselwort `JOIN` durch `INNER JOIN` ersetzt werden.

(i)
```
SELECT Schueler.Name
FROM Schueler
WHERE Schueler.Klasse = "Q1" OR Schueler.Klasse = "Q2"
      OR Schueler.Klasse = "EF"
```

(ii)
```
SELECT Schueler.Name, Wettbewerbsrunde.Fachgebiet,
                                        Wettbewerbsrunde.Termin
FROM Wettbewerbsrunde JOIN Schueler
      ON Wettbewerbsrunde.RundeID = Schueler.RundeID
WHERE Preis LIKE "Platz 1"
ORDER BY Wettbewerbsrunde.Fachgebiet
```

(iii)
```
SELECT Wettbewerb.Wettbewerbsbezeichnung
FROM Wettbewerb
WHERE Wettbewerb.Wettbewerbsbezeichnung NOT IN
(SELECT DISTINCT gehoertZu.Wettbewerbsbezeichnung
  FROM Schueler JOIN gehoertZu
             ON gehoertZu.RundeID = Schueler.RundeID)
```

Teilaufgabe d)

(i)

`Wettbewerb`

Wettbewerbsbezeichnung	URL
RoboCup	www.robocup.org

`Wettbewerbsrunde`

RundeID	Fachgebiet	Runde	Termin
5	Robotik	weltweit	01.07.2017

`gehoertZu`

RundeID	Wettbewerbsbezeichnung
5	RoboCup

Die Daten des Wettbewerbs und die Daten der Veranstaltung werden in die entsprechenden Tabellen übernommen. Abschließend wird eine Beziehung über die Tabelle `gehoertZu` hergestellt, in die die beiden Primärschlüsselwerte übernommen werden.

(ii)

`Wettbewerbsrunde`

RundeID	Fachgebiet	Runde	Termin
6	Informatik	regional	24.02.2017

Schueler

Name	Geburtsjahr	Klasse	RundeID	Preis
Marie Winter	2003	9b	6	Platz 1

`gehoertZu`

RundeID	Wettbewerbsbezeichnung
6	Jugend forscht

Die noch nicht existierende Schülerin wird in den Datenbestand (Tabelle `Schueler`) aufgenommen. Die Wettbewerbsrunde des Wettbewerbs Jugend forscht wird in der Tabelle `Wettbewerbsrunde` ergänzt, über die Tabelle `gehoertZu` dem Wettbewerb Jugend forscht zugeordnet. Abschließend lassen sich die ID der (neuen) Wettbewerbsrunde und (sofern noch nicht geschehen) der Preis bei der neu eingefügten Schülerin in der Tabelle `Schueler` ergänzen.

(iii)

Das Einfügen dieser Information ist aufgrund der Primärschlüsseleigenschaft von `Name` und `Geburtsjahr` in der Tabelle `Schueler` nicht möglich, da Heike Heigel (2002) darin bereits existiert.

Die Information kann nur eingefügt werden, indem die bisherige Information überschrieben wird.

Teilaufgabe e)

Die Beziehung gehoertZu ist eine 1:n Beziehung, da jede Wettbewerbsrunde einem Wettbewerb zugeordnet werden kann.

Die 1:n-Beziehung zwischen den Entitätstypen Wettbewerb und Wettbewerbsrunde im ursprünglichen Entwurf ist unpassend umgesetzt worden. Die zusätzliche Relation gehoertZu ist überflüssig. Der Schlüssel Wettbewerbsbezeichnung kann als Fremdschlüssel in die Relation Wettbewerbsrunde aufgenommen werden.

Die Beziehung nimmtTeil ist eine n:m-Beziehung, denn an jeder Wettbewerbsrunde können mehrere Schülerinnen und Schüler teilnehmen. Jede Schülerin und jeder Schüler kann an mehreren Wettbewerbsrunden teilnehmen.

Die n:m-Beziehung zwischen den Entitätstypen Wettbewerbsrunde und Schueler ist im ursprünglichen Entwurf falsch umgesetzt worden. Hierfür ist eine zusätzliche Relation erforderlich.

Ein Primärschlüssel SchuelerID beim Entitätstyp Schueler ist sinnvoll, denn an einer Schule können mehrere Schüler mit gleichem Namen und Geburtsjahr existieren. Daraus ergibt sich, dass es keine transitive Abhängigkeit von Nichtschlüsselattributen gibt.

Datenbankschema

```
Wettbewerb(Wettbewerbsbezeichnung, URL)
WettbewerbsRunde(RundeID, Fachgebiet, Runde, Termin,
                                        ↑Wettbewerbsbezeichnung)
Schueler(SchuelerID, Name, Vorname, Geburtsjahr, Klasse)
nimmtTeil(↑StufeID, ↑SchuelerID, Preis)
```

Alle Attribute sind atomar. Damit sind die Bedingungen für die erste Normalform erfüllt.

Die zweite Normalform ist erfüllt, da alle Entitätstypen Primärschlüssel haben, die nur aus einem Attribut bestehen.

In der Relation nimmtTeil hängt das Attribut Preis nicht von einem Teilschlüssel ab.

Die dritte Normalform ist erfüllt, da keine transitiven Abhängigkeiten zwischen den Nichtschlüsselattributen existieren.

7. Teilleistungen – Kriterien / Bewertungsbogen zur Prüfungsarbeit

Name des Prüflings:_____ Kursbezeichnung:_____

Schule: _____

Teilaufgabe a)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK[2]**	**ZK**	**DK**
1	erläutert die Informationen, die in der Datenbank stehen, im Sachzusammenhang.	4			
Sachlich richtige Lösungsalternative zur Modelllösung: (4) …… ……					
	Summe Teilaufgabe a)	**4**			

Teilaufgabe b)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	analysiert die erste SQL-Anweisung und bestimmt die Informationen, die in der Datenbank gesucht werden.	5			
2	analysiert die zweite SQL-Anweisung und bestimmt die Informationen, die in der Datenbank gesucht werden.	5			
3	analysiert die dritte SQL-Anweisung und bestimmt die Informationen, die in der Datenbank gesucht werden.	5			
Sachlich richtige Lösungsalternative zur Modelllösung: (15) …… ……					
	Summe Teilaufgabe b)	**15**			

[2] EK = Erstkorrektur; ZK = Zweitkorrektur; DK = Drittkorrektur

Teilaufgabe c)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	entwickelt für den ersten Suchauftrag eine geeignete SQL-Anweisung.	5			
2	entwickelt für den zweiten Suchauftrag eine geeignete SQL-Anweisung.	5			
3	entwickelt für den dritten Suchauftrag eine geeignete SQL-Anweisung.	5			
Sachlich richtige Lösungsalternative zur Modelllösung: (15) …… ……					
	Summe Teilaufgabe c)	**15**			

Teilaufgabe d)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	gibt an, wie die Eintragungen in der Beispieldatenbank ergänzt werden müssen, und erläutert dieses.	3			
2	gibt an, wie die Eintragungen in der Beispieldatenbank ergänzt werden müssen, und erläutert dieses.	3			
3	erläutert, dass die Beispieldatenbank nicht entsprechend ergänzt werden kann.	3			
Sachlich richtige Lösungsalternative zur Modelllösung: (9) …… ……					
	Summe Teilaufgabe d)	**9**			

Teilaufgabe e)

	Anforderungen	Lösungsqualität			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	begründet, dass der ursprüngliche Entwurf fehlerhaft in ein Datenbankschema überführt worden ist, und erläutert, worin die Verbesserung des Entwurfs gegenüber dem ursprünglichen Entwurf besteht.	4			
2	überführt das neue ER-Modell in ein Datenbankschema, das in dritter Normalform ist, und erläutert die Entscheidungen.	3			
Sachlich richtige Lösungsalternative zur Modelllösung: (7) …… ……					
	Summe Teilaufgabe e)	**7**			

	Summe insgesamt	**50**			

Festlegung der Gesamtnote (Bitte nur bei der letzten bearbeiteten Aufgabe ausfüllen.)

	Lösungsqualität			
	maximal erreichbare Punktzahl	**EK**	**ZK**	**DK**
Übertrag der Punktsumme aus der ersten bearbeiteten Aufgabe	**50**			
Übertrag der Punktsumme aus der zweiten bearbeiteten Aufgabe	**50**			
Punktzahl der gesamten Prüfungsleistung	**100**			
aus der Punktsumme resultierende Note gemäß nachfolgender Tabelle				
Note ggf. unter Absenkung um bis zu zwei Notenpunkte gemäß § 13 Abs. 2 APO-GOSt				
Paraphe				

Berechnung der Endnote nach Anlage 4 der Abiturverfügung auf der Grundlage von § 34 APO-GOSt

Die Klausur wird abschließend mit der Note _____ (____ Punkte) bewertet.

Unterschrift, Datum:

Grundsätze für die Bewertung (Notenfindung)

Für die Zuordnung der Notenstufen zu den Punktzahlen ist folgende Tabelle zu verwenden:

Note	**Punkte**	**Erreichte Punktzahl**
sehr gut plus	15	100 – 95
sehr gut	14	94 – 90
sehr gut minus	13	89 – 85
gut plus	12	84 – 80
gut	11	79 – 75
gut minus	10	74 – 70
befriedigend plus	9	69 – 65
befriedigend	8	64 – 60
befriedigend minus	7	59 – 55
ausreichend plus	6	54 – 50
ausreichend	5	49 – 45
ausreichend minus	4	44 – 40
mangelhaft plus	3	39 – 33
mangelhaft	2	32 – 27
mangelhaft minus	1	26 – 20
ungenügend	0	19 – 0

Name: _____

Abiturprüfung 2017

Informatik, Grundkurs

Aufgabenstellung:

Seit einiger Zeit bieten Fernsehsender ihren Zuschauern an, bereits gesendete Beiträge nachträglich über das Internet anzuschauen. Zu jeder Sendung werden u. a. Angaben zur ursprünglichen Sendezeit abgespeichert.

Eine Servicekraft soll die Anfangs- und Endzeiten (Uhrzeiten) der Sendungen eingeben. Als Trennzeichen zwischen Stunden und Minuten wird der Doppelpunkt vereinbart. Die erste mögliche Zeitangabe eines Tages ist 00:00 Uhr, die letzte 23:59 Uhr. Da insgesamt sehr viele Sendungen zu erfassen sind, sollen Abkürzungen möglich sein: Nullen an der Zehnerstelle bei einstelligen Stunden- und Minutenangaben können weggelassen werden, d. h. neben „05:30“ bzw. „15:05“ sollen also auch die Eingaben „5:30“ bzw. „15:5“ zulässig sein.

Zur Überprüfung, ob es sich um eine syntaktisch zulässige Eingabe handelt, soll ein deterministischer endlicher Automat verwendet werden. Abbildung 1 zeigt den Übergangsgraphen des Automaten. Nicht dargestellte Übergänge führen in einen Fehlerzustand q_F.

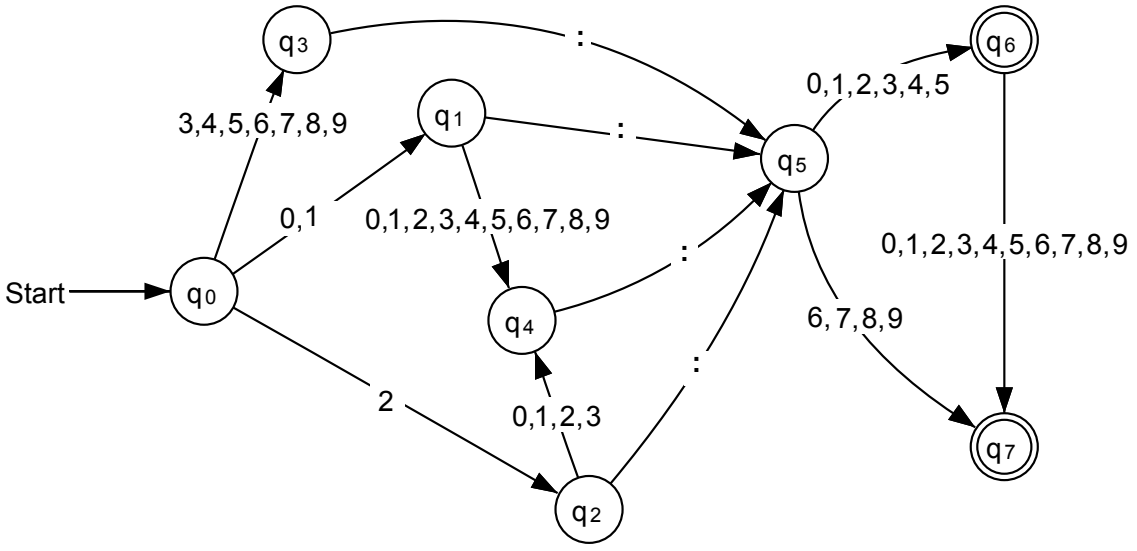


Abbildung 1: Übergangsgraph zum Überprüfen einer Zeitangabe auf Gültigkeit

Name: _____

a) Das Eingabealphabet des deterministischen endlichen Automaten zum Übergangsgraphen aus Abbildung 1 bestehe aus allen zehn Ziffern und dem Doppelpunktzeichen.

Geben Sie die Menge der Zustände, den Startzustand und die Menge der Endzustände des Automaten zum Übergangsgraphen aus Abbildung 1 an.

Geben Sie die Zustandsfolge des Automaten zum Übergangsgraphen in Abbildung 1 bei Eingabe der folgenden drei Wörter an:

`1:5` `12:73` `:41`

Erläutern Sie im Sachzusammenhang, warum die drei Wörter akzeptiert bzw. nicht akzeptiert werden.

Erläutern Sie die Bedeutung der Zustände q_4 *und* q_6 *im Sachzusammenhang.*

(15 Punkte)

b) Der abgebildete Automat enthält mehr Zustände, als erforderlich wären. Zwei der Zustände können zu einem „zusammengelegt“ werden, ohne die akzeptierte Sprache des Automaten zum Übergangsgraphen zu ändern.

Entscheiden Sie, ob die Zustände q_1 *und* q_2 *oder die Zustände* q_3 *und* q_4 *zusammengelegt werden können.*

Stellen Sie dar, welche Änderungen nötig sind, um den überzähligen Zustand im Übergangsgraphen zu entfernen, ohne die akzeptierte Sprache zum Automaten zu ändern.

Hinweis: Sie können zur Darstellung der Änderungen den Übergangsgraphen in der Anlage verwenden.

(8 Punkte)

Name: _____

c) Die folgende Grammatik G erzeugt die „Uhrzeit-Sprache“, also alle Wörter, die syntaktisch korrekten Uhrzeiten entsprechen, die nach obigem Schema gebildet werden:

Startsymbol: S

Menge der Terminalsymbole: T = {:, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

Menge der Nichtterminalsymbole: N = {S, H, M, X, Y, Z}

Menge der Produktionsregeln:

P = {

S → H:M ,
H → 0Z | 1Z | 2X | Z ,
M → YZ | Z ,
X → 0 | 1 | 2 | 3 ,
Y → 0 | 1 | 2 | 3 | 4 | 5 ,
Z → Y | 6 | 7 | 8 | 9

}

Begründen Sie, warum die Grammatik G nicht regulär ist.

Erläutern Sie, warum mit der angegebenen Grammatik G nur Uhrzeiten in der erlaubten Form gemäß dem Automaten zum Übergangsgraphen in Abbildung 1 gebildet werden können.

Entwickeln Sie eine reguläre Grammatik, mit der die Wörter der „Uhrzeit-Sprache“ erzeugt werden können.

(15 Punkte)

d) Um noch mehr Zeit beim Eintippen der Uhrzeiten sparen zu können, sollen fehlende Stunden- bzw. Minutenangaben als „0“ interpretiert werden: Die Eingabe „:15“ entspricht also 00:15 Uhr und „20:“ entspricht 20:00 Uhr, während „:“ Mitternacht repräsentiert.

Erläutern Sie, welche Änderungen aufgrund dieser Erweiterungen am Übergangsgraphen aus Abbildung 1 vorzunehmen sind.

(6 Punkte)

Name: _____

e) In der Pause kommt jemand auf die Idee, dass es lustig wäre, „Palindromzeiten“ [1] mithilfe eines endlichen Automaten erkennen zu lassen. „Palindromzeiten“ sollen Uhrzeiten sein, bei denen die Minuten rückwärts gelesen die Stundenzahl ergeben, z. B. 04:40 Uhr. Das Eingabealphabet für den Automaten soll die Menge `{:, 0, 1, 2, 3, 4, 5}` sein. Während einige noch unsicher sind, ob es so einen endlichen Automaten geben kann, behauptet jemand, dass es sogar einen deterministischen endlichen Automaten mit maximal 4 Zuständen gibt, der Palindromzeiten erkennt.

Beurteilen Sie die Behauptung, dass es einen deterministischen endlichen Automaten mit dem genannten Eingabealphabet mit maximal 4 Zuständen gibt, der Palindromzeiten erkennt, und geben Sie ihn ggf. an.

(6 Punkte)

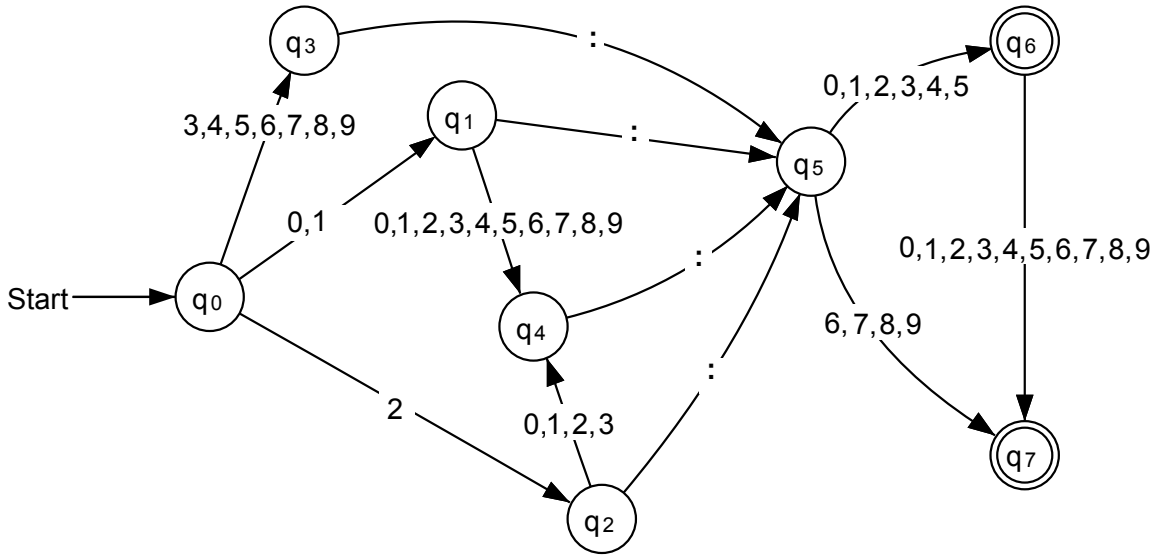
Zugelassene Hilfsmittel:

- Taschenrechner (graphikfähiger Taschenrechner / CAS-Taschenrechner)
- Wörterbuch zur deutschen Rechtschreibung

[1] Hinweis: Palindrome sind Zeichenfolgen, die von hinten und vorne gelesen dasselbe Wort ergeben.

Name: _____

Anlage: Übergangsgraph (zur Bearbeitung von Teilaufgabe b)



Unterlagen für die Lehrkraft

Abiturprüfung 2017

Informatik, Grundkurs

1. Aufgabenart

Analyse, Modellierung und Implementation von kontextbezogenen Problemstellungen mit Schwerpunkt auf dem Inhaltsfeld formale Sprachen und Automaten

2. Aufgabenstellung[1]

siehe Prüfungsaufgabe

3. Materialgrundlage

entfällt

4. Bezüge zum Kernlehrplan und zu den Vorgaben 2017

Die Aufgaben weisen vielfältige Bezüge zu den Kompetenzerwartungen und Inhaltsfeldern des Kernlehrplans bzw. zu den in den Vorgaben ausgewiesenen Fokussierungen auf. Im Folgenden wird auf Bezüge von zentraler Bedeutung hingewiesen.

1. *Inhaltsfelder und inhaltliche Schwerpunkte*
   Formale Sprachen und Automaten
   - Endliche Automaten
     - deterministische endliche Automaten
   - Grammatiken regulärer Sprachen
   - Möglichkeiten und Grenzen von Automaten und formalen Sprachen

2. *Medien/Materialien*
   - entfällt

5. Zugelassene Hilfsmittel

- Taschenrechner (graphikfähiger Taschenrechner / CAS-Taschenrechner)
- Wörterbuch zur deutschen Rechtschreibung

[1] Die Aufgabenstellung deckt inhaltlich alle drei Anforderungsbereiche ab.

6. Modelllösungen

Die jeweilige Modelllösung stellt eine mögliche Lösung bzw. Lösungsskizze dar. Der gewählte Lösungsansatz und -weg der Schülerinnen und Schüler muss nicht identisch mit dem der Modelllösung sein. Sachlich richtige Alternativen werden mit entsprechender Punktzahl bewertet (Bewertungsbogen: Zeile „Sachlich richtige Lösungsalternative zur Modelllösung“).

Teilaufgabe a)

Die Menge der Zustände ist $\{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_F\}$, wobei q_0 der Startzustand ist. Die Menge der Endzustände ist $\{q_6, q_7\}$.

Eingabefolge `1:5`

Zustandsfolge: $q_0 \rightarrow q_1 \rightarrow q_5 \rightarrow q_6$. Die Eingabe wird akzeptiert.
Die Eingabe entspricht der Uhrzeit 01:05 Uhr und ist eine abgekürzte gültige Eingabe. Führende Nullen bei Stunden- und Minutenangaben dürfen weggelassen werden.

Eingabefolge `12:73`

Zustandsfolge: $q_0 \rightarrow q_1 \rightarrow q_4 \rightarrow q_5 \rightarrow q_7 \rightarrow q_F$. Die Eingabe wird nicht akzeptiert.
Da eine Stunde nur 60 Minuten hat, ist die Minutenangabe 73 ungültig.

Eingabefolge `:41`

Zustandsfolge: $q_0 \rightarrow q_F$. Die Eingabe wird nicht akzeptiert.
Die Stundenzahl ist ganz weggelassen worden. Es ist aber vereinbart worden, dass nur die Zehnerstellen weggelassen werden dürfen, die Einerstellen müssen angegeben werden.

Bedeutung des Zustands q_4:
Die bisher erfolgte Eingabe bildet eine Zahl für die Stunde (da noch kein Doppelpunkt gelesen wurde), wobei bislang zwei Ziffern eingelesen wurden: entweder erst eine `2` mit anschließender Ziffer `0` bis `3` (Stundenzahlen 20 bis 23) oder erst eine `0` oder `1` mit anschließender beliebiger Ziffer (Stundenzahlen 00 bis 19). Jetzt wird zwingend ein Doppelpunkt erwartet.

Bedeutung des Zustands q_6:
Bisher wurde (nach der Zahl für die Stunde und dem „:“-Trennzeichen) eine der Ziffern von `0` bis `5` eingelesen. Da diese einzelne Ziffer als verkürzte Schreibweise für eine Minutenangabe gelten kann, handelt es sich bereits um einen akzeptierenden Zustand. Auch jede folgende Ziffer kann noch akzeptiert werden. Dann wird die zuletzt gelesene Ziffer als Zehnerziffer der Minutenangabe interpretiert.

Teilaufgabe b)

Die Zustände q_1 und q_2 können nicht zusammengelegt werden, da man z. B. beim Lesen des Symbols `9` aus dem Zustand q_1 in den Zustand q_4 gelangt, vom Zustand q_2 aber in den Fehlerzustand q_F.
Die Zustände q_3 und q_4 können zu einem einzigen Zustand verschmolzen werden, da man von ihnen durch die gleiche Eingabe jeweils in denselben Folgezustand gelangt: bei der Eingabe des Doppelpunkts ist der Folgezustand jeweils q_5, bei der Eingabe einer Ziffer ist es der Fehlerzustand.
Beispiel für notwendige Änderungen: Die Übergänge von q_0 zu q_3 werden als Übergänge von q_0 zum Zustand q_4 eingetragen. Der Zustand q_3 kann dann entfernt werden.

Teilaufgabe c)

Die angegebene Grammatik G ist nicht regulär, da z. B. die Produktion `S → H:M` zwei Nichtterminale auf der rechten Seite enthält, was für reguläre Grammatiken nicht erlaubt ist.
Die Grammatik stellt sicher, dass jedes Wort der Sprache einen Doppelpunkt enthält (Zwang der Anwendung der ersten Produktionsregel, da diese als einzige auf der linken Seite das Startsymbol aufweist). Danach können keine weiteren Doppelpunkte mehr erzeugt werden.
Die X-Produktionen erzeugen Ziffern von 0 bis 3 (für die Einerstelle für Stunden nach 20 Uhr), die `Y`-Produktionen Ziffern von `0` bis `5` (für die Zehnerstelle bei zweistelligen Minutenangaben) und die Z-Produktionen alle möglichen Ziffern (die Ziffern der Y-Produktionen und zusätzlich die Ziffern von `6` bis `9`).
Links vom Doppelpunkt können über die `H`-Produktionen die Stunden erzeugt werden: entweder werden einstellige Zahlen erzeugt (`H → Z`) für die Stundenangaben 0 bis 9 oder zweistellige Zahlen mit einer führenden Null oder Eins und einer beliebigen Folgeziffer (`H → 0Z | 1Z`) für die Stunden zwischen 00 und 19 Uhr bzw. mit einer führenden `2` und einer folgenden Ziffer von `0` bis `3` (`H → 2X`) für die Stunden 20 bis 23 Uhr.
Rechts vom Doppelpunkt können über die `M`-Produktionen Minutenangaben erzeugt werden: entweder werden einstellige Zahlen erzeugt (`M → Z`) für die Minutenangaben 0 bis 9 oder zweistellige Zahlen mit einer führenden Ziffer von 0 bis 5 und einer beliebigen Folgeziffer (`M → YZ`) für die Minutenangaben von 00 bis 59.

Eine reguläre Grammatik zur „Uhrzeit-Sprache“ ist zum Beispiel die folgende:

Startsymbol: S

T = {:, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

N = {S, A, B, C, D, E}

P = {

S → 0A | 1A | 2B | 3C | 4C | 5C | 6C | 7C | 8C | 9C

A → 0C | 1C | 2C | 3C | 4C | 5C | 6C | 7C | 8C | 9C | :D

B → 0C | 1C | 2C | 3C | :D

C → :D

D → 0E | 1E | 2E | 3E | 4E | 5E | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

E → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

}

Teilaufgabe d)

Um die Anforderung umzusetzen, genügt es, zwei Änderungen vorzunehmen:
Vom Zustand q_0 gelangt man nach dem Lesen des Eingabesymbols : nicht mehr zum Fehlerzustand, sondern in den Zustand q_5. Außerdem wird q_5 zu einem akzeptierenden Zustand.

Die erste Maßnahme sorgt dafür, dass man die Eingabe der Stunden überspringen kann (fehlende Nullen vor dem Doppelpunkt sind nun erlaubt), und durch die zweite Maßnahme wird auch das Fehlen von Ziffern nach dem Doppelpunkt akzeptiert.

Teilaufgabe e)

Man muss zu dem Schluss kommen, dass es keinen solchen Automaten gibt.
Begründung: Der Automat benötigt außer dem Startzustand 3 Zustände, um das erste Zeichen zu speichern, denn das erste und letzte Zeichen jedes Wortes der Sprache müssen entweder beide 0, beide 1 oder beide 2 sein. Das sind schon 4 Zustände. Weitere Zustände werden benötigt, um den Wert der zweiten Stelle zu speichern, denn das zweite Zeichen muss identisch zum vierten Zeichen sein. Das zeigt, dass ein deterministischer endlicher Automat zum Erkennen der Sprache mehr als 4 Zustände braucht.

7. Teilleistungen – Kriterien / Bewertungsbogen zur Prüfungsarbeit

Name des Prüflings:_____ Kursbezeichnung:_____

Schule: _____

Teilaufgabe a)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK[2]**	**ZK**	**DK**
1	gibt die Menge der Zustände, den Startzustand und die Menge der Endzustände an.	3			
2	gibt die Zustandsfolge für alle drei Eingaben an.	3			
3	erläutert im Sachzusammenhang für jedes der drei Wörter, ob das Wort akzeptiert wird.	3			
4	erläutert die Bedeutung des Zustands q_4 im Sachzusammenhang.	3			
5	erläutert die Bedeutung des Zustands q_6 im Sachzusammenhang.	3			
Sachlich richtige Lösungsalternative zur Modelllösung: (15) …… ……					
	Summe Teilaufgabe a)	**15**			

Teilaufgabe b)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	entscheidet begründet, welche Zustände zusammengelegt werden können.	5			
2	stellt die nötigen Änderungen dar.	3			
Sachlich richtige Lösungsalternative zur Modelllösung: (8) …… ……					
	Summe Teilaufgabe b)	**8**			

[2] EK = Erstkorrektur; ZK = Zweitkorrektur; DK = Drittkorrektur

Teilaufgabe c)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	begründet, warum die Grammatik G nicht regulär ist.	2			
2	erläutert, warum mit der Grammatik G nur Uhrzeiten in der erlaubten Form gebildet werden können.	6			
3	entwickelt eine reguläre Grammatik, mit der die Wörter der „Uhrzeit-Sprache“ erzeugt werden können.	7			
Sachlich richtige Lösungsalternative zur Modelllösung: (15) …… ……					
	Summe Teilaufgabe c)	**15**			

Teilaufgabe d)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	erläutert, welche Änderungen am Übergangsgraphen vorzunehmen sind.	6			
Sachlich richtige Lösungsalternative zur Modelllösung: (6) …… ……					
	Summe Teilaufgabe d)	**6**			

Teilaufgabe e)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	beurteilt die genannte Behauptung.	6			
Sachlich richtige Lösungsalternative zur Modelllösung: (6) …… ……					
	Summe Teilaufgabe e)	**6**			

	Summe insgesamt	**50**			

Festlegung der Gesamtnote (Bitte nur bei der letzten bearbeiteten Aufgabe ausfüllen.)

	Lösungsqualität			
	maximal erreichbare Punktzahl	EK	ZK	DK
Übertrag der Punktsumme aus der ersten bearbeiteten Aufgabe	**50**			
Übertrag der Punktsumme aus der zweiten bearbeiteten Aufgabe	**50**			
Punktzahl der gesamten Prüfungsleistung	**100**			
aus der Punktsumme resultierende Note gemäß nachfolgender Tabelle				
Note ggf. unter Absenkung um bis zu zwei Notenpunkte gemäß § 13 Abs. 2 APO-GOSt				
Paraphe				

Berechnung der Endnote nach Anlage 4 der Abiturverfügung auf der Grundlage von § 34 APO-GOSt

Die Klausur wird abschließend mit der Note _____ (____ Punkte) bewertet.

Unterschrift, Datum:

Grundsätze für die Bewertung (Notenfindung)

Für die Zuordnung der Notenstufen zu den Punktzahlen ist folgende Tabelle zu verwenden:

Note	**Punkte**	**Erreichte Punktzahl**
sehr gut plus	15	100 – 95
sehr gut	14	94 – 90
sehr gut minus	13	89 – 85
gut plus	12	84 – 80
gut	11	79 – 75
gut minus	10	74 – 70
befriedigend plus	9	69 – 65
befriedigend	8	64 – 60
befriedigend minus	7	59 – 55
ausreichend plus	6	54 – 50
ausreichend	5	49 – 45
ausreichend minus	4	44 – 40
mangelhaft plus	3	39 – 33
mangelhaft	2	32 – 27
mangelhaft minus	1	26 – 20
ungenügend	0	19 – 0