Name: _____

Abiturprüfung 2018

Informatik, Leistungskurs

Aufgabenstellung:

Tsunamis können in Folge eines Seebebens entstehen, wenn sich der Meeresboden absenkt. Vom Ort der Meeresbodenabsenkung breitet sich die Tsunamiwelle mit großer Geschwindigkeit aus. Dies ist dem Effekt vergleichbar, wenn ein Stein in einen See geworfen wird. Die Welle wird höher, je weniger tief der Meeresboden ist.
Zur Warnung vor Tsunamis werden Verwerfungen der Erdkruste, an denen solche Beben zu erwarten sind, überwacht. Ein Bestandteil eines Systems zur Überwachung sind Bojen. Diese Bojen können Messdaten via Satellit an ein Kontrollzentrum übermitteln. Es wird ein System entwickelt, bei dem 10 solcher Bojen in einem Abstand von jeweils einem Kilometer zwischen einer unterseeischen Verwerfungslinie und der Küstenlinie angeordnet sind. Abbildung 1 zeigt eine vereinfachte Darstellung mit 5 Bojen. In der Darstellung hat sich soeben der Meeresboden gesenkt, wobei der ursprüngliche Verlauf des Meeresbodens gestrichelt dargestellt ist.

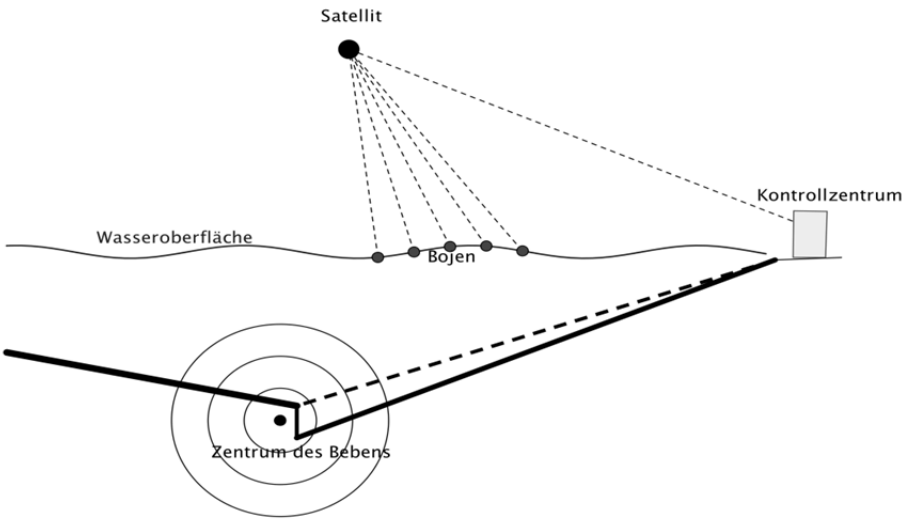


Abbildung 1: Querschnittsskizze der Komponenten des Warnsystems

Name: _____

Zur Erprobung des Warnsystems wurde ein Informatiksystem entwickelt. Abbildung 2 zeigt einen Ausschnitt des Implementationsdiagramms des Informatiksystems. Eine Dokumentation relevanter Klassen finden Sie im Anhang.

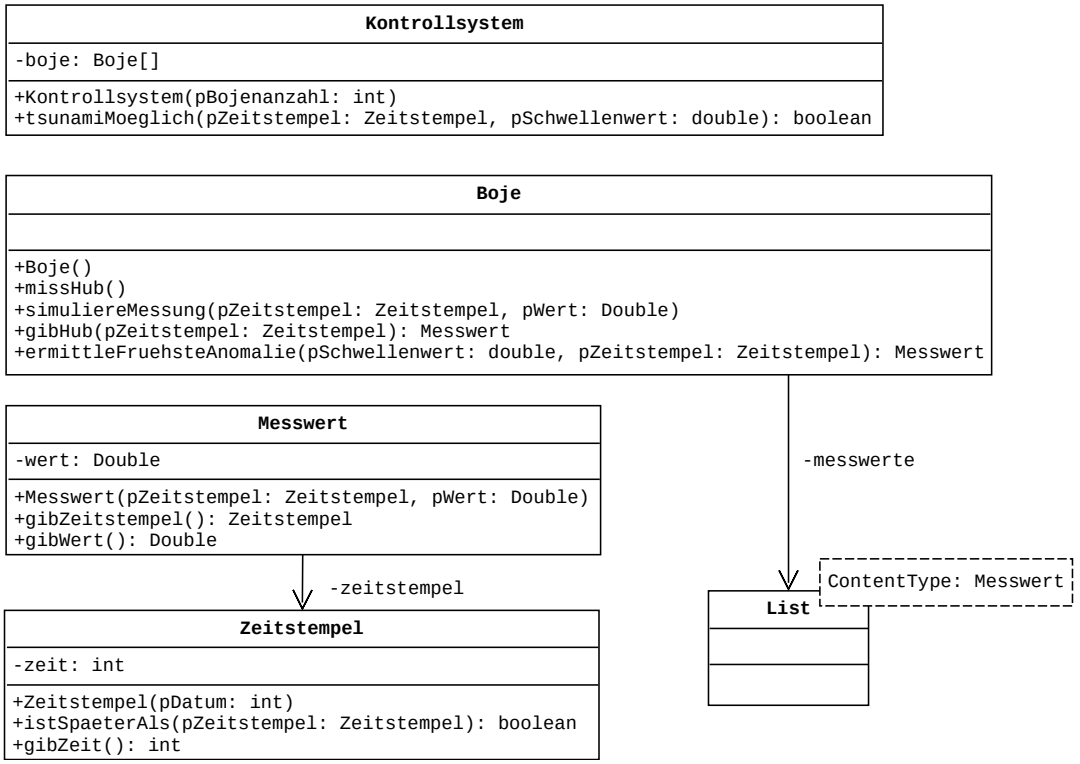


Abbildung 2: Ausschnitt aus dem Implementationsdiagramm

a) *Erläutern Sie die Beziehungen zwischen den Klassen* `Kontrollsystem`, `Boje`, `Messwert` *und* `Zeitstempel`.

Erläutern Sie auf Grundlage der Dokumentation der Klasse `Messwert`, *warum in dieser Klasse Objekte vom Typ* `Double` *verwendet werden und nicht der elementare Typ* `double`.

(6 Punkte)

b) In der Klasse `Boje` wird eine Methode `ermittleFruehsteAnomalie` benötigt, die im Anhang dokumentiert ist.

Implementieren Sie diese Methode.

Erläutern Sie die Funktionsweise Ihrer Implementation.

(10 Punkte)

Name: _____

c) Das Warnsystem erzeugt eine Tsunamiwarnung, wenn die folgende Methode `tsunamiMoeglich` den Wert `true` liefert.

```
 1 public boolean tsunamiMoeglich(Zeitstempel pZeitstempel,
                                          double pSchwellenwert) {
 2   Zeitstempel letzterZeitpunkt = pZeitstempel;
 3   int zaehler = 0;
 4   for (int i = 0; i < boje.length; i++) {
 5     Messwert anomalie = boje[i].ermittleFruehsteAnomalie
                                (pSchwellenwert, letzterZeitpunkt);
 6     if (anomalie != null) {
 7       letzterZeitpunkt = anomalie.gibZeitstempel();
 8       zaehler = zaehler + 1;
 9     }
10   }
11   if (zaehler < boje.length / 2) {
12     return false;
13   } else {
14     return true;
15   }
16 }
```

Analysieren Sie die Methode `tsunamiMoeglich` *der Klasse* `Kontrollsystem` *und erläutern Sie ihre Funktionsweise im Sachzusammenhang, wenn beim Aufruf der Zeitstempel des Seebebens als Parameter übergeben wird.*

Erläutern Sie allgemein, unter welchen Umständen eine Tsunamiwarnung ausgelöst wird.

(10 Punkte)

Name: _____

Neben den fest verankerten Bojen gibt es so genannte Treibbojen, die auf eine Tiefe von bis zu 2000 m sinken und wiederauftauchen können. Diese Bojen können beispielsweise die Salzkonzentration, die Temperatur oder Strömungsverhältnisse auch in Abhängigkeit von der Tauchtiefe und ihrer Position messen. In großen Forschungsprojekten (z. B. GEOSS, ARGO) werden zum Teil bis zu Tausende solcher Bojen ausgesetzt, wobei Bojen häufig ersetzt werden müssen, weil sie defekt sind oder verloren gehen, oder es werden weitere Bojen hinzugefügt.

d) Damit das Kontrollsystem auch für solche Forschungsprojekte eingesetzt werden kann, wurde das Entwurfsdiagramm in Abbildung 3 zur Anpassung der Klassen `Kontrollsystem` und `Boje` des bestehenden Systems vorgeschlagen.

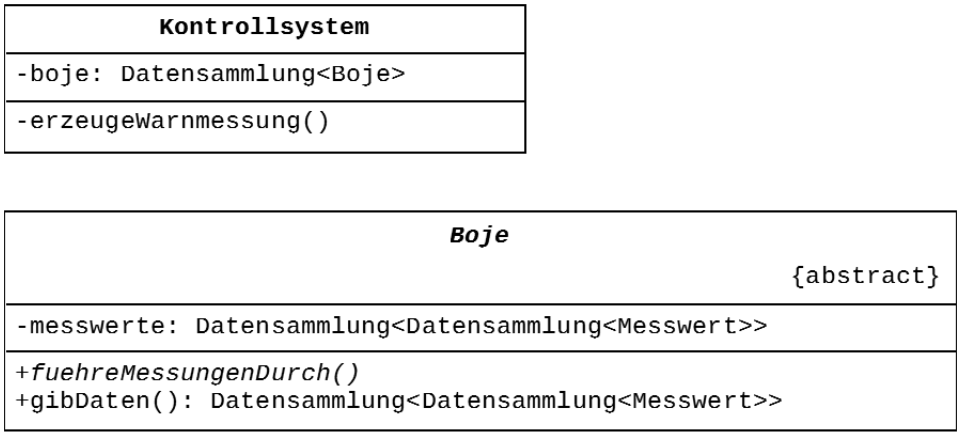


Abbildung 3: Entwurfsdiagramm

Erläutern Sie wesentliche Änderungen zum bisherigen Modell.

Überführen Sie das Entwurfsdiagramm aus Abbildung 3 in ein Implementationsdiagramm.

Erläutern Sie Ihre Entscheidungen hinsichtlich der gewählten Datenstrukturen.

(10 Punkte)

Name: _____

e) Zur Datenanalyse müssen bestimmte statistische Größen einer großen Menge von Messdaten erhoben werden. Die Methode `ermittleEtwas` der Klasse `Kontrollsystem` hilft bei der Bestimmung solcher Größen.

```
 1 private Double ermittleEtwas(List<Double>liste, int k) {
 2   List<Double> kListe = new List<Double>();
 3   int anzahl = 0;
 4   List<Double> gListe = new List<Double>();
 5   liste.toFirst();
 6   Double p = liste.getContent();
 7   liste.next();
 8   while (liste.hasAccess()) {
 9     Double wert = liste.getContent();
10     if (wert.compareTo(p) < 0) {
11       kListe.append(wert);
12       anzahl = anzahl + 1;
13     } else  {
14       gListe.append(wert);
15     }
16     liste.next();
17   }
18   if (anzahl == k-1) {
19     return p;
20   }
21   else if (anzahl > k-1) {
22     return ermittleEtwas(kListe, k);
23   }
24   else {
25     return ermittleEtwas(gListe, k-anzahl-1);
26   }
27 }
```

Gegeben sind drei Listen mit Testdaten:

```
liste1 = [0.0, 5.0, 10.0, 15.0, 20.0, 25.0, 30.0]
liste2 = [30.0, 25.0, 20.0, 15.0, 10.0, 5.0, 0.0]
liste3 = [30.0, 5.0, 20.0, 15.0, 0.0, 10.0, 25.0]
```

Name: _____

Analysieren Sie die Methode `ermittleEtwas`*, indem Sie die Rückgabewerte der Aufrufe* `ermittleEtwas(liste1, 4)`, `ermittleEtwas(liste2, 4)` *und* `ermittleEtwas(liste3, 4)` *ermitteln. Protokollieren Sie für jeden der drei Aufrufe, mit welchen Belegungen der beiden Parameter die Selbstaufrufe der Methode (in Zeile 22 bzw. 25) jeweils erfolgen.*

Erläutern Sie allgemein, was die Methode zurückliefert, wenn sie als Parameter eine Liste mit einer ungeraden Anzahl von `n` *Elementen und den Wert* `k = (n + 1)/2` *erhält.*

Erläutern Sie eine Situation, in welcher in der Methode ein Laufzeitfehler auftreten kann.

(14 Punkte)

Zugelassene Hilfsmittel:

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

Name: _____

Anhang

Dokumentationen der verwendeten Klassen

Die Klasse Kontrollsystem

Ein Objekt der Klasse `Kontrollsystem` dient zur Verwaltung der Bojen.

Ausschnitt aus der Dokumentation der Klasse `Kontrollsystem`

Konstruktor **`Kontrollsystem(int pBojenanzahl)`**

Ein Kontrollsystem zur Verwaltung der gegebenen Anzahl von Bojen wird erzeugt.

Anfrage **`boolean tsunamiMoeglich(Zeitstempel pZeitstempel, double pSchwellenwert)`**

Wenn ein Tsunami gemäß den Modellannahmen möglich ist, wird `true` zurückgegeben, sonst `false`.

Auftrag **`void erzeugeWarnmessung()`**

Daten, die einer tatsächlichen Tsunami-Situation entsprechen, werden in das System eingespeist.

Name: _____

Die Klasse Boje

Ein Objekt der Klasse `Boje` dient zur Repräsentation einer Boje. Es speichert Objekte vom Typ `Messwert` in einer linearen Liste.

Ausschnitt aus der Dokumentation der Klasse `Boje`

Konstruktor **`Boje()`**

Eine Boje wird erzeugt.

Anfrage **`Messwert ermittleFruehsteAnomalie(double pSchwellenwert, Zeitstempel pZeitstempel)`**

Es wird der Messwert ermittelt, dessen Zeitstempel der früheste nach dem Zeitstempel `pZeitstempel` ist und größer ist als der Schwellenwert. Existiert kein entsprechender Messwert, wird `null` zurückgegeben.

Anfrage **`Messwert gibHub(Zeitstempel pZeitstempel)`**

Liefert den Messwert zu dem gegebenen Zeitstempel. Existiert kein entsprechender Wert, wird `null` zurückgegeben.

Auftrag **`void simuliereMessung(Zeitstempel pZeitstempel, Double pWert)`**

Ein Messwert mit dem gegebenen Zeitstempel und Wert wird als neuer Messwert an die Liste der Messwerte angehängt. Diese Methode dient zur Erzeugung von Erprobungsdaten. Der `Double`-Wert kann `null` sein, um einen Ausfall des Sensors zu simulieren.

Auftrag **`void missHub()`**

Eine Messung des Hubsensors mit dem aktuellen Zeitstempel wird als neuer Messwert an die Liste der Messwerte angehängt. Die Messung kann (mit bestimmten Wahrscheinlichkeiten) fehlschlagen (Störung des Sensors): Der `Double`-Wert des zu erzeugenden Messwertes kann `null` sein oder der Messwert wird gar nicht in die Liste der Messwerte eingetragen.

Name: _____

Die Klasse `Messwert`

Ein Objekt der Klasse `Messwert` dient zur Verwaltung eines Messwertes. Jeder Messwert hat einen Zeitstempel und einen Wert.

Ausschnitt aus der Dokumentation der Klasse `Messwert`

Konstruktor **`Messwert(Zeitstempel pZeitstempel, Double pWert)`**

Ein Messwert mit dem übergebenen Zeitstempel und Wert wird erzeugt.

Anfrage **`Double gibWert()`**

Liefert den Messwert in Form eines Objektes vom Typ `Double` zurück. Ist der Wert ungültig (durch Defekt), wird `null` zurückgegeben.

Anfrage **`Zeitstempel gibZeitstempel()`**

Liefert den Zeitpunkt der Erhebung eines Messwertes in Form eines Objektes vom Typ `Zeitstempel` zurück.

Die Klasse `Zeitstempel`

Ein Objekt der Klasse `Zeitstempel` dient zur Repräsentation eines Zeitpunktes.

Ausschnitt aus der Dokumentation der Klasse `Zeitstempel`

Konstruktor **`Zeitstempel(int pDatum)`**

Ein Zeitstempel wird erzeugt. Dabei entspricht `pDatum` der Anzahl an Sekunden, die seit dem 01.01.1970 (um 0.00 Uhr) bis zum zu speichernden Zeitpunkt vergangen ist.

Anfrage **`int gibZeit()`**

Liefert die Anzahl an Sekunden, die seit dem 01.01.1970 (um 0.00 Uhr) bis zum gespeicherten Zeitpunkt vergangen ist.

Anfrage **`boolean istSpaeterAls(Zeitstempel pZeitstempel)`**

Liefert `true`, wenn der gespeicherte Zeitpunkt zeitlich später liegt als der durch `pZeitstempel` gegebene. Andernfalls wird `false` zurückgegeben.

Name: _____

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 2018

Informatik, Leistungskurs

1. Aufgabenart

Modellierung, Implementation und Analyse kontextbezogener Problemstellungen

2. Aufgabenstellung[1]

siehe Prüfungsaufgabe

3. Materialgrundlage

entfällt

4. Bezüge zum Kernlehrplan und zu den Vorgaben 2018

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, Array

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

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)

Ein Objekt vom Typ `Kontrollsystem` verwaltet ein eindimensionales Array vom Typ `Boje`.
Objekte vom Typ `Boje` verwalten Objekte vom Typ `Messwert` in einer entsprechend typisierten linearen Liste, welche mehrere Objekte des Typs `Messwert` assoziieren kann.
Ein Objekt der Klasse `Messwert` verwaltet jeweils eine Gleitkommazahl in einem Objekt vom Typ `Double` und ein Objekt vom Typ `Zeitstempel`.
Die Klasse `Messwert` verwendet Objekte vom Typ `Double`, damit der Ausfall eines Sensors zu einem durch ein Objekt des Typs `Zeitstempel` gegebenen Zeitpunkt simuliert werden kann. In diesem Fall ist der entsprechende Wert `null`.

Teilaufgabe b)

```
 1 public Messwert ermittleFruehsteAnomalie(double pSchwellenwert,
                                            Zeitstempel pZeitstempel) {
 2   Messwert ergebnis = null;
 3   Double schwelle = new Double(pSchwellenwert);
 4   messwerte.toFirst();
 5   while (ergebnis == null && messwerte.hasAccess()) {
 6     Messwert dieserMesswert = messwerte.getContent();
 7     Zeitstempel dieserStempel = dieserMesswert.gibZeitstempel();
 8     Double dieserWert = dieserMesswert.gibWert();
 9     if (dieserWert != null && dieserWert.compareTo(schwelle) > 0
                  && dieserStempel.istSpaeterAls(pZeitstempel)) {
10       ergebnis = dieserMesswert;
11     }
12     messwerte.next();
13   }
14   return ergebnis;
15 }
```

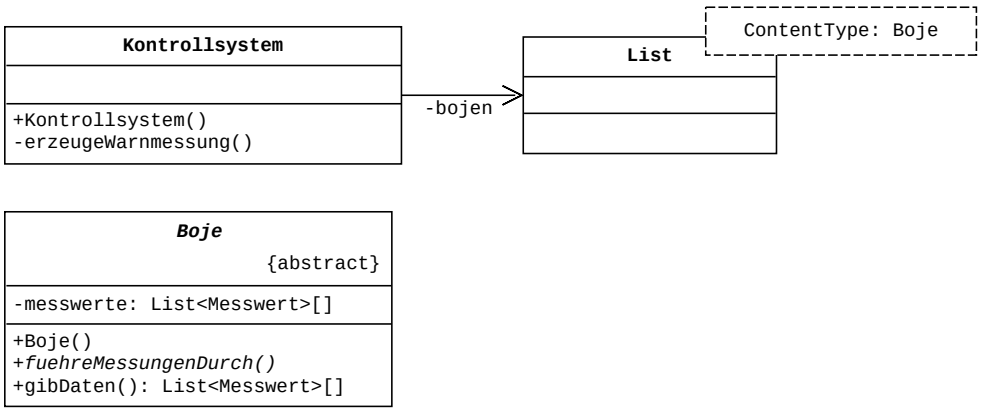
Die Liste der Messwerte wird von ihrem ersten Objekt an durchlaufen (Zeile 4/5). Ein Verweis auf das aktuelle Inhaltsobjekt wird gespeichert (Zeile 6). Zeitstempel und Messwert des aktuellen Objektes werden ermittelt (Zeile 7/8). Ist der Wert größer als der Schwellenwert und liegt zeitlich später als der übergebene Zeitstempel, ist das Ergebnis gefunden (Zeile 9). Der Schleifendurchlauf bricht jetzt ab, da `ergebnis` nicht mehr `null` ist. Das Ergebnis wird zurückgeliefert. Wurde kein passender Messwert gefunden, ist die Rückgabe `null`.

Teilaufgabe c)

Die Methode bestimmt zunächst die Anzahl der Bojen, die – in der Zeit nach dem durch seinen Zeitstempel gegebenen Seebeben – einen Messwert ermittelt haben, der über dem gegebenen Schwellenwert liegt.
Der Zeitstempel des Bebens wird in der Referenz `letzterZeitpunkt` festgehalten.
In der Schleife ab Zeile 4 werden alle Bojen durchlaufen. Dabei wird jeweils die früheste Anomalie (Messwert der ersten Überschreitung des gegebenen Schwellenwertes), welche diese Boje nach dem letzten festgelegten Zeitpunkt festgestellt hat, ermittelt.
Existiert eine solche Anomalie, wird ihr Zeitpunkt zum neuen Wert von `letzterZeitpunkt`, und die Anzahl der Bojen mit einer Anomalie wird um eins erhöht.
Entspricht schließlich die Anzahl der so gefundenen Bojen (mit einer Anomalie nach dem Zeitpunkt des Seebebens) mindestens der Hälfte aller Bojen, wird eine Tsunamiwarnung ausgelöst.

Teilaufgabe d)

Die Bojen werden mit Hilfe einer abstrakten Klasse modelliert. Es können also keine Objekte vom Typ `Boje` mehr direkt erzeugt werden. Die Unterklassen der Klasse `Boje` müssen die abstrakte Methode `fuehreMessungenDurch` implementieren. Die Messwerte werden in einer Datensammlung von Datensammlungen von Objekten vom Typ `Messwert` verwaltet. Auf diese Datensammlung kann durch die Methode `gibDaten` zugegriffen werden.
In der Klasse `Kontrollsystem` wird nicht mehr zwingend ein Feld zur Verwaltung der Bojen verwendet. Die Anzahl der Bojen wird nicht mehr im Konstruktor gegeben.



Zur Verwaltung der Bojen wird die dynamische Datenstruktur einer Liste verwendet, da sich die Anzahl der verwalteten Bojen häufig ändert.
Es wird ein Feld von Listen von Messwerten verwendet, denn die Bojen werden in der Regel über eine feste Anzahl von Sensoren verfügen. Jedem Sensor ist eine Liste von Messwerten zugeordnet, in welcher seine Messdaten gespeichert werden können. Die Anzahl dieser Daten ist dynamisch.

Teilaufgabe e)

Fall 1:

```
ermittleEtwas([0.0, 5.0, 10.0, 15.0, 20.0, 25.0, 30.0], 4)
ermittleEtwas([5.0, 10.0, 15.0, 20.0, 25.0, 30.0], 3)
ermittleEtwas([10.0, 15.0, 20.0, 25.0, 30.0], 2)
ermittleEtwas([15.0, 20.0, 25.0, 30.0], 1)
liefert 15.0
```

Fall 2:

```
ermittleEtwas([30.0, 25.0, 20.0, 15.0, 10.0, 5.0, 0.0], 4)
ermittleEtwas([25.0, 20.0, 15.0, 10.0, 5.0, 0.0], 4)
ermittleEtwas([20.0, 15.0, 10.0, 5.0, 0.0], 4)
ermittleEtwas([15.0, 10.0, 5.0, 0.0], 4)
liefert 15.0
```

Fall 3:

```
ermittleEtwas([30.0, 5.0, 20.0, 15.0, 0.0, 10.0, 25.0], 4)
ermittleEtwas([5.0, 20.0, 15.0, 0.0, 10.0, 25.0], 4)
ermittleEtwas([20.0, 15.0, 10.0, 25.0], 2)
ermittleEtwas([15.0, 10.0], 2)
liefert 15.0
```

Die Methode wählt zunächst das erste Element der übergebenen Liste als Pivotelement `p`.
Anschließend werden zwei Listen `kListe` und `gListe` gebildet, wobei `kListe` alle Werte enthält, die kleiner als `p` sind, und `gListe` alle Werte, die größer als (oder gleich) `p` sind.
Beim Einfügen in `kListe` wird die Anzahl `anzahl` der Werte in dieser Liste ermittelt.
Ist die Anzahl der Werte kleiner `p` genau `k-1`, so ist `p` der k-kleinste Wert in der ursprünglichen Liste und wird zurückgegeben.
Ist `anzahl` größer als `k-1`, erfolgt ein Selbstaufruf mit der Liste der kleineren Werte unter Beibehaltung von `k` (der insgesamt k-kleinste Wert ist in dieser Liste).
Ist schließlich `anzahl` kleiner als `k-1`, erfolgt ein Selbstaufruf mit der Liste der größeren Werte, wobei `k` nun zu `k-anzahl-1` wird (der (k-anzahl-1)-te Wert dieser Liste ist der insgesamt k-kleinste). Die Methode ermittelt also den k-kleinsten Wert in der Liste.
In diesem Fall liefert die Liste den Median der Werte zurück.
Wenn die Methode mit einem Verweis auf das Nullobjekt aufgerufen wird, kommt es zu einem Laufzeitfehler (Zeile 5 - `Nullpointer- Exception`).

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 Beziehungen zwischen den Klassen.	4			
2	erläutert, warum in der Klasse `Messwert` der Typ `Double` verwendet wird.	2			
Sachlich richtige Lösungsalternative zur Modelllösung: (6) …… ……					
	Summe Teilaufgabe a)	**6**			

Teilaufgabe b)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	implementiert die Methode.	6			
2	erläutert die Funktionsweise der Implementierung.	4			
Sachlich richtige Lösungsalternative zur Modelllösung: (10) …… ……					
	Summe Teilaufgabe b)	**10**			

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

Teilaufgabe c)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	analysiert die Methode und erläutert die Funktionsweise im Sachzusammenhang.	6			
2	erläutert, unter welchen Umständen eine Warnung ausgelöst wird.	4			
Sachlich richtige Lösungsalternative zur Modelllösung: (10) …… ……					
	Summe Teilaufgabe c)	**10**			

Teilaufgabe d)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	erläutert wesentliche Änderungen zum bisherigen Modell.	3			
2	überführt das Diagramm in ein Implementationsdiagramm.	4			
3	erläutert seine Entscheidungen bzgl. der gewählten Datenstrukturen.	3			
Sachlich richtige Lösungsalternative zur Modelllösung: (10) …… ……					
	Summe Teilaufgabe d)	**10**			

Teilaufgabe e)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	analysiert die Methode, ermittelt die Rückgabe und protokolliert die Selbstaufrufe.	8			
2	erläutert allgemein, was die Methode in der beschriebenen Situation zurückliefert.	4			
3	erläutert eine Situation in der ein Laufzeitfehler auftreten kann.	2			
Sachlich richtige Lösungsalternative zur Modelllösung: (14) …… ……					
	Summe Teilaufgabe e)	**14**			

	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**			
Übertrag der Punktsumme aus der dritten bearbeiteten Aufgabe	**50**			
Punktzahl der gesamten Prüfungsleistung	**150**			
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	150 – 143
sehr gut	14	142 – 135
sehr gut minus	13	134 – 128
gut plus	12	127 – 120
gut	11	119 – 113
gut minus	10	112 – 105
befriedigend plus	9	104 – 98
befriedigend	8	97 – 90
befriedigend minus	7	89 – 83
ausreichend plus	6	82 – 75
ausreichend	5	74 – 68
ausreichend minus	4	67 – 60
mangelhaft plus	3	59 – 50
mangelhaft	2	49 – 41
mangelhaft minus	1	40 – 30
ungenügend	0	29 – 0

Name: _____

Abiturprüfung 2018

Informatik, Leistungskurs

Aufgabenstellung:

Eine Gruppe von Studierenden forscht an der Vernetzung von Fahrzeugen. Sie sind daran interessiert, wie sich Fahrzeuge im laufenden Verkehr selbstständig miteinander vernetzen können.
Für jedes Fahrzeug ist es wichtig, die Fahrzeuge in seiner räumlichen Nähe zu erkennen und sich mit ihnen zu verbinden. Mit diesen benachbarten Fahrzeugen kann es direkt kommunizieren. Eine solche Verbindung wird **direkte Verbindung** genannt.

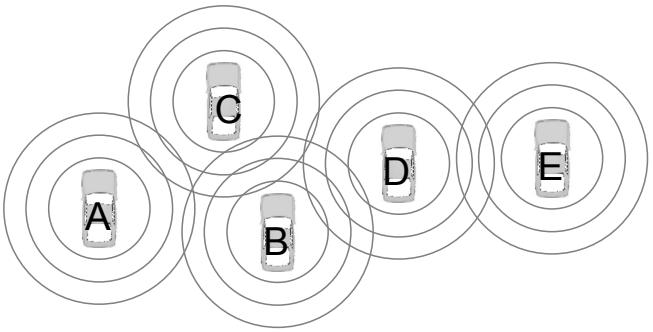


Abbildung 1: Miteinander vernetzte Fahrzeuge

Jedes Fahrzeug kann auch mit bestimmten Fahrzeugen kommunizieren, mit denen es nicht direkt verbunden ist – sofern diese nicht zu weit entfernt sind. Zu diesen Fahrzeugen besteht eine **indirekte Verbindung**.
Damit die Fahrzeuge identifiziert werden können, verfügen sie über eine eindeutige Identifikationsnummer (ID).
Die Vernetzung der Fahrzeuge untereinander soll durch ungerichtete Graphen modelliert werden. Dabei sollen die Fahrzeuge als Knoten dargestellt werden und die Verbindungen zwischen zwei Fahrzeugen als Kanten.

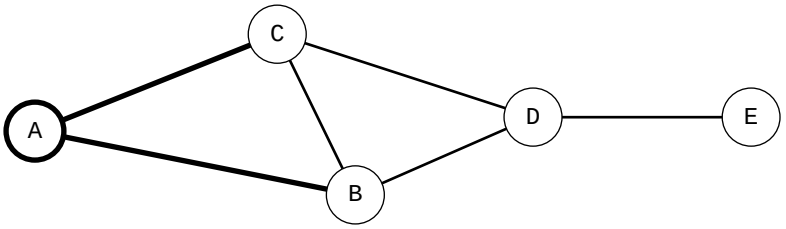


Abbildung 2: Beispiel für eine Netzwerkrepräsentation
Das Fahrzeug mit der ID A und seine direkten Verbindungen sind hervorgehoben

Name: _____

Im fließenden Verkehr verändert sich die Position der Fahrzeuge zueinander ständig: Eine direkte Verbindung kann daher verloren gehen oder eine neue direkte Verbindung aufgebaut werden.

a) Im Beispiel-Netzwerk (siehe Abbildung 2) treten Veränderungen auf:

   i. Die Verbindung zwischen dem Fahrzeug mit der ID `A` und dem Fahrzeug mit der ID `B` geht verloren.
   ii. Ein neues Fahrzeug mit der ID `F` wird ins Netz aufgenommen und eine Verbindung zwischen diesem Fahrzeug und dem Fahrzeug mit der ID `D` ergänzt.

   Geben Sie zunächst in Form einer Tabelle für jeden Knoten des Graphen aus Abbildung 2 an, zu welchen Knoten er eine Kante besitzt.

   Stellen Sie die Veränderungen aus i. und ii. im Netzwerk aus Abbildung 2 grafisch dar.

   Erläutern Sie, wie die Veränderungen aus i. und ii. am Graphen eines Netzwerks mithilfe der Methoden der Klassen `Graph`, `Edge` *und* `Vertex` *umgesetzt werden können.*

   (10 Punkte)

Da es keinen zentralen Server gibt, bei dem das gesamte Netzwerk verwaltet wird, muss jedes Fahrzeug die direkten und indirekten Verbindungen zu den anderen Fahrzeugen selbstständig verwalten.
Mit Fahrzeugen, die zu weit weg sind, soll das Fahrzeug nicht kommunizieren. Die Entfernung zwischen zwei Fahrzeugen wird dadurch bestimmt, wie viele Fahrzeuge mindestens nötig sind, um eine Nachricht vom Sender zum Empfänger weiterzuleiten. Die Anzahl dieser Zwischenstationen wird als „Hops“ bezeichnet.
Es wird im Folgenden davon ausgegangen, dass maximal `2` Hops zulässig sind. Für das Beispielnetzwerk aus Abbildung 3 werden also vom Fahrzeug mit der ID `A` zum Fahrzeug mit der ID `B` `0` Hops benötigt, da zwischen ihnen eine direkte Verbindung besteht. Vom Fahrzeug mit der ID `A` zum Fahrzeug mit der ID `H` werden `2` Hops benötigt. Fahrzeuge, die mehr als zwei Hops vom jeweiligen Fahrzeug entfernt sind, werden nicht in dessen Netzwerkrepräsentation gespeichert.

Name: _____

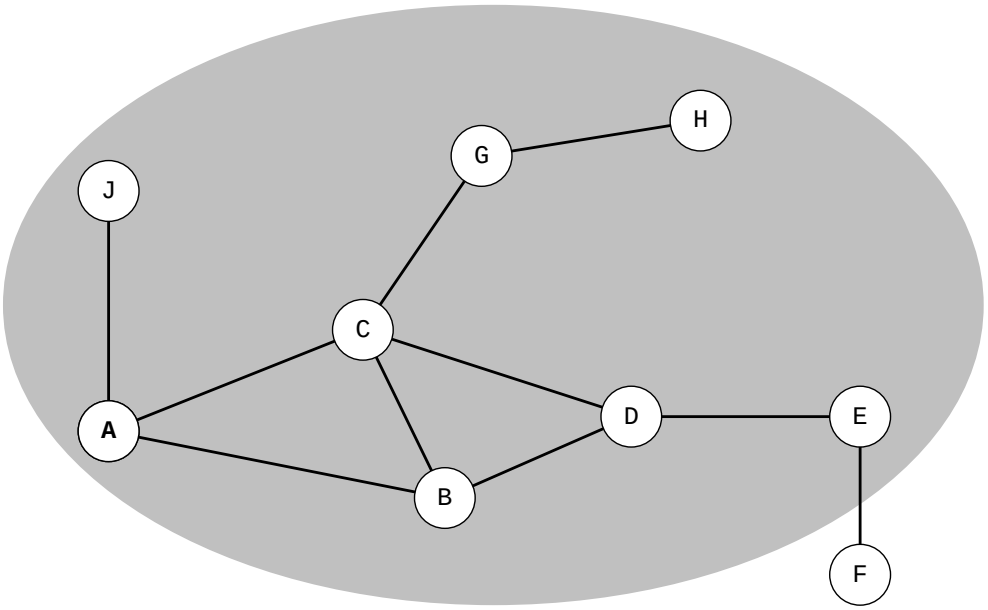


Abbildung 3: Darstellung eines Gesamtnetzwerks von 9 miteinander kommunizierenden Fahrzeugen. Der grau unterlegte Teil entspricht derjenigen Netzwerkrepräsentation, welche im Fahrzeug mit der ID A gespeichert ist.

b) *Stellen Sie das Teilnetzwerk dar, das vom Fahrzeug mit der ID J gespeichert ist.*

(5 Punkte)

Da sich die Verbindungen zwischen den Fahrzeugen ständig ändern, muss der Graph, der die Netzwerkumgebung eines Fahrzeugs repräsentiert, regelmäßig aktualisiert werden. Dazu wird folgendes Verfahren verwendet:

- Bei jedem Fahrzeug wird regelmäßig automatisch die Netzwerkumgebung geprüft und gegebenenfalls aktualisiert: Verliert es eine bestehende direkte Verbindung zu einem anderen Fahrzeug oder kommt eine neue direkte Verbindung hinzu, so werden diese Veränderungen in die Repräsentation des eigenen Netzwerks eingebaut. Außerdem wird an alle direkt verbundenen Fahrzeuge eine Liste der direkten Verbindungen des einen Fahrzeugs mit der Veränderung gesendet.
- Empfängt ein Fahrzeug neue Informationen über das Netzwerk, so wird das eigene Netzwerkmodell entsprechend angepasst. Darüber hinaus werden diese Informationen an alle direkten Nachbarn weitergeleitet (außer an den, der die Information gesendet hat), sofern diese nicht mehr als 2 Hops vom ursprünglichen Sender der Informationen entfernt sind.

Name: _____

Dieses Verfahren soll beispielhaft implementiert werden. Dazu wird im Wesentlichen die Klasse `Fahrzeug` verwendet.

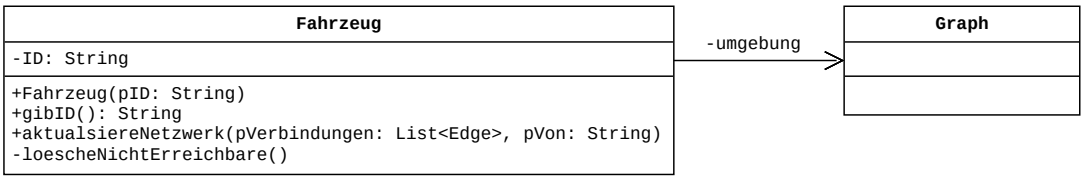


Abbildung 4: Teilmodellierung in Form eines Implementationsdiagramms

c) Die Methode `aktualisiereNetzwerk` der Klasse `Fahrzeug` nimmt die erforderlichen Änderungen an der gespeicherten Repräsentation des Netzwerks vor, wenn entsprechende Daten von einem anderen Fahrzeug eintreffen. Dabei wird bei der Methode davon ausgegangen, dass die Daten nicht von einem Fahrzeug stammen, das zu weit entfernt ist.
Die Repräsentation soll mit folgenden Beispieldaten analysiert werden:
ID des Fahrzeugs, das die Daten empfängt: `H`
ID des Fahrzeugs, von dem gesendet wurde (`pVon`): `C`
Durch `pVerbindung` übergebene Kanten:
`[C – G, D – C, C – B, A – C]`

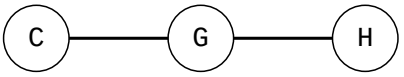


Abbildung 5: Gespeicherter Graph `umgebung` des Fahrzeugs `H`

Name: _____

```
 1 public void aktualisiereNetzwerk(List<Edge> pVerbindungen,
                                    String pVon) {
 2   Vertex von = new Vertex(pVon);
 3   umgebung.addVertex(von);
 4   List<Edge> verbindungen = umgebung.getEdges(von);
 5   verbindungen.toFirst();
 6   while (verbindungen.hasAccess()) {
 7     umgebung.removeEdge(verbindungen.getContent());
 8     verbindungen.next();
 9   }
10   pVerbindungen.toFirst();
11   while (pVerbindungen.hasAccess()) {
12     Edge aktuell = pVerbindungen.getContent();
13     if (umgebung.getVertex(
           aktuell.getVertices()[0].getID()) == null) {
14       umgebung.addVertex(aktuell.getVertices()[0]);
15     } else if (umgebung.getVertex(
           aktuell.getVertices()[1].getID()) == null) {
16       umgebung.addVertex(aktuell.getVertices()[1]);
17     }
18     umgebung.addEdge(aktuell);
19     pVerbindungen.next();
20   }
21 }
```

Analysieren Sie die Methode `aktualisiereNetzwerk`*, indem Sie sie auf die gegebenen Beispieldaten anwenden und nach Ausführung von Zeile* `7` *und jeweils nach Ausführung von Zeile* `18` *die Veränderung des Netzwerks darstellen.*
Erläutern Sie die Strategie, nach der die Repräsentation des Netzwerks im Graphen `umgebung` *aktualisiert wird.*

(12 Punkte)

Name: _____

d) Durch den Wegfall von Verbindungen zwischen Fahrzeugen kann es dazu kommen, dass Fahrzeuge in der Netzwerkrepräsentation eines Fahrzeugs enthalten sind, die nicht mehr erreicht werden können: Es gibt nunmehr weder eine direkte noch eine indirekte Verbindung (zur Vereinfachung wird die Einschränkung bzgl. der Hops in dieser Aufgabe nicht betrachtet).
Die Methode `loescheNichtErreichbare` der Klasse `Fahrzeug` soll die nicht erreichbaren Fahrzeuge einschließlich ihrer direkten Verbindungen aus dem Graphen entfernen.
Die Methode hat den folgenden Methodenkopf:

```
private void loescheNichtErreichbare()
```

Entwickeln Sie einen Algorithmus zum Entfernen der nicht erreichbaren Fahrzeuge. Implementieren Sie die Methode `loescheNichtErreichbare`.

(15 Punkte)

e) Ist die Anzahl der miteinander verbundenen Fahrzeuge groß, entsteht das Problem, dass sehr viele Verbindungsdaten übertragen werden. Aus diesem Grund wurde ein neues Verfahren zur Übertragung von Änderungen im Netzwerk entwickelt:
Bauen zwei Fahrzeuge zueinander eine neue direkte Verbindung auf, so übertragen sie nur diese Änderung an ihre direkten Nachbarn. Der Rest des Verfahrens bleibt unverändert.
Beispiel: Das Fahrzeug mit der ID `P` und das Fahrzeug mit der ID `K` bauen zueinander eine neue Verbindung auf.

Abbildung 6: Netzwerk des Fahrzeugs mit der ID `P`.

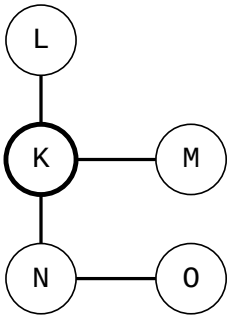


Abbildung 7: Netzwerk des Fahrzeugs mit der ID `K`.

Erläutern Sie anhand der gegebenen Beispieldaten für das alte und für das neue Verfahren, welche Daten zwischen den Fahrzeugen mit den ID `P` *und* `K` *jeweils ausgetauscht werden. Beurteilen Sie das neue Verfahren.*

(8 Punkte)

Zugelassene Hilfsmittel:

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

Name: _____

Anhang

Dokumentationen der verwendeten Klassen

Die Klasse `Fahrzeug`

Die Klasse `Fahrzeug` stellt ein Fahrzeug im Netzwerk dar. Es verwaltet die eigene Repräsentation des Netzwerks, das das Fahrzeug umgibt.

Dokumentation der Klasse `Fahrzeug`

Konstruktor **`Fahrzeug(String pID)`**

Ein Objekt vom Typ `Fahrzeug` wird erstellt. Das gespeicherte Netzwerk enthält nur den Knoten, der das eigene Fahrzeug repräsentiert. Die übergebene ID wird gespeichert.

Anfrage **`String gibID()`**

Eine Referenz auf die gespeicherte ID wird zurückgegeben.

Auftrag **`void aktualisiereNetzwerk(List<Edge> pVerbindungen, String pVon)`**

Siehe Aufgabenteil c).

Name: _____

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.

Name: _____

Die generische Klasse Queue<ContentType>

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

Dokumentation der generischen Klasse `Queue<ContentType>`

Konstruktor **`Queue()`**

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

Anfrage **`boolean isEmpty()`**

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

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

Das Objekt `pContent` wird an die Schlange angehängt. Falls `pContent` gleich `null` ist, bleibt die Schlange unverändert.

Auftrag **`void dequeue()`**

Das erste Objekt wird aus der Schlange entfernt. Falls die Schlange leer ist, wird sie nicht verändert.

Anfrage **`ContentType front()`**

Die Anfrage liefert das erste Objekt der Schlange. Die Schlange bleibt unverändert. Falls die Schlange leer ist, wird `null` zurückgegeben.

Name: _____

Die Klasse Graph

Die Klasse `Graph` stellt einen ungerichteten, kantengewichteten Graphen dar. Es können Knoten- und Kantenobjekte hinzugefügt und entfernt, flache Kopien der Knoten- und Kantenlisten des Graphen angefragt und Markierungen von Knoten und Kanten gesetzt und überprüft werden. Des Weiteren kann eine Liste der Nachbarn eines bestimmten Knoten, eine Liste der inzidenten Kanten eines bestimmten Knoten und die Kante von einem bestimmten Knoten zu einem anderen Knoten angefragt werden. Abgesehen davon kann abgefragt werden, welches Knotenobjekt zu einer bestimmten ID gehört und ob der Graph leer ist.

Dokumentation der Klasse `Graph`

Konstruktor **`Graph()`**

Ein Objekt vom Typ `Graph` wird erstellt. Der von diesem Objekt repräsentierte Graph ist leer.

Auftrag **`void addVertex(Vertex pVertex)`**

Der Auftrag fügt den Knoten `pVertex` vom Typ `Vertex` in den Graphen ein, sofern es noch keinen Knoten mit demselben ID-Eintrag wie `pVertex` im Graphen gibt und `pVertex` eine ID ungleich `null` hat. Ansonsten passiert nichts.

Auftrag **`void addEdge(Edge pEdge)`**

Der Auftrag fügt die Kante `pEdge` in den Graphen ein, sofern beide durch die Kante verbundenen Knoten im Graphen enthalten sind, nicht identisch sind und noch keine Kante zwischen den beiden Knoten existiert. Ansonsten passiert nichts.

Auftrag **`void removeVertex(Vertex pVertex)`**

Der Auftrag entfernt den Knoten `pVertex` aus dem Graphen und löscht alle Kanten, die mit ihm inzident sind. Ist der Knoten `pVertex` nicht im Graphen enthalten, passiert nichts.

Auftrag **`void removeEdge(Edge pEdge)`**

Der Auftrag entfernt die Kante `pEdge` aus dem Graphen. Ist die Kante `pEdge` nicht im Graphen enthalten, passiert nichts.

Anfrage **`Vertex getVertex(String pID)`**

Die Anfrage liefert das Knotenobjekt mit `pID` als ID. Ist ein solches Knotenobjekt nicht im Graphen enthalten, wird `null` zurückgeliefert.

Anfrage **`List<Vertex> getVertices()`**

Die Anfrage liefert eine neue Liste aller Knotenobjekte vom Typ `List<Vertex>`. Enthält der Graph keine Knotenobjekte, so wird eine leere Liste zurückgeliefert.

Name: _____

Anfrage **`List<Vertex> getNeighbours(Vertex pVertex)`**

Die Anfrage liefert alle Nachbarn des Knotens `pVertex` als neue Liste vom Typ `List<Vertex>`. Hat der Knoten `pVertex` keine Nachbarn in diesem Graphen oder ist gar nicht in diesem Graphen enthalten, so wird eine leere Liste zurückgeliefert.

Anfrage **`List<Edge> getEdges()`**

Die Anfrage liefert eine neue Liste aller Kantenobjekte vom Typ `List<Edge>`. Enthält der Graph keine Kantenobjekte, so wird eine leere Liste zurückgeliefert.

Anfrage **`List<Edge> getEdges(Vertex pVertex)`**

Die Anfrage liefert eine neue Liste aller inzidenten Kanten zum Knoten `pVertex`. Hat der Knoten `pVertex` keine inzidenten Kanten in diesem Graphen oder ist gar nicht in diesem Graphen enthalten, so wird eine leere Liste zurückgeliefert.

Anfrage **`Edge getEdge(Vertex pVertex, Vertex pAnotherVertex)`**

Die Anfrage liefert die Kante, welche die Knoten `pVertex` und `pAnother-Vertex` verbindet, als Objekt vom Typ `Edge`. Ist der Knoten `pVertex` oder der Knoten `pAnotherVertex` nicht im Graphen enthalten oder gibt es keine Kante, die beide Knoten verbindet, so wird `null` zurückgeliefert.

Auftrag **`void setAllVertexMarks(boolean pMark)`**

Der Auftrag setzt die Markierungen aller Knoten des Graphen auf den Wert `pMark`.

Anfrage **`boolean allVerticesMarked()`**

Die Anfrage liefert `true`, wenn die Markierungen aller Knoten des Graphen den Wert `true` haben, ansonsten `false`.

Auftrag **`void setAllEdgeMarks(boolean pMark)`**

Der Auftrag setzt die Markierungen aller Kanten des Graphen auf den Wert `pMark`.

Anfrage **`boolean allEdgesMarked()`**

Die Anfrage liefert `true`, wenn die Markierungen aller Kanten des Graphen den Wert `true` haben, ansonsten `false`.

Anfrage **`boolean isEmpty()`**

Die Anfrage liefert `true`, wenn der Graph keine Knoten enthält, ansonsten `false`.

Name: _____

Die Klasse Vertex

Die Klasse `Vertex` stellt einen einzelnen Knoten eines Graphen dar. Jedes Objekt dieser Klasse verfügt über eine im Graphen eindeutige ID als String und kann diese ID zurückliefern. Darüber hinaus kann eine Markierung gesetzt und abgefragt werden.

Dokumentation der Klasse `Vertex`

Konstruktor **`Vertex(String pID)`**

Ein neues Objekt vom Typ `Vertex` ohne Inhaltsobjekt wird erstellt. Der von diesem Objekt repräsentierte Knoten ist noch in keinen Graphen eingefügt. Seine Markierung hat den Wert `false`.

Anfrage **`String getID()`**

Die Anfrage liefert die `ID` des Knotens als `String`.

Auftrag **`void setMark(boolean pMark)`**

Der Auftrag setzt die Markierung des Knotens auf den Wert `pMark`.

Anfrage **`boolean isMarked()`**

Die Anfrage liefert `true`, wenn die Markierung des Knotens den Wert `true` hat, ansonsten `false`.

Name: _____

Die Klasse Edge

Die Klasse Edge stellt eine einzelne, ungerichtete Kante eines Graphen dar. Beim Erstellen werden die beiden durch sie zu verbindenden Knotenobjekte und eine Gewichtung als `double` übergeben. Beide Knotenobjekte können abgefragt werden. Des Weiteren können die Gewichtung und eine Markierung gesetzt und abgefragt werden.

Dokumentation der Klasse `Edge`

Konstruktor **`Edge(Vertex pVertex, Vertex pAnotherVertex, double pWeight)`**

Ein neues Objekt vom Typ `Edge` wird erstellt. Die von diesem Objekt repräsentierte Kante verbindet die Knoten `pVertex` und `pAnotherVertex` mit der Gewichtung `pWeight` und ist noch in keinen Graphen eingefügt. Ihre Markierung hat den Wert `false`.

Auftrag **`void setWeight(double pWeight)`**

Der Auftrag setzt das Gewicht der Kante auf den Wert `pWeight`.

Anfrage **`double getWeight()`**

Die Anfrage liefert das Gewicht der Kante als `double`.

Anfrage **`Vertex[] getVertices()`**

Die Anfrage gibt die beiden Knoten, die durch die Kante verbunden werden, als neues Feld vom Typ `Vertex` zurück. Das Feld hat genau zwei Einträge mit den Indexwerten `0` und `1`.

Auftrag **`void setMark(boolean pMark)`**

Der Auftrag setzt die Markierung der Kante auf den Wert `pMark`.

Auftrag **`void setWeight(double pWeight)`**

Der Auftrag setzt das Gewicht der Kante auf den Wert `pWeight`.

Anfrage **`boolean isMarked()`**

Die Anfrage liefert `true`, wenn die Markierung der Kante den Wert `true` hat, ansonsten `false`.

Unterlagen für die Lehrkraft

Abiturprüfung 2018

Informatik, Leistungskurs

1. Aufgabenart

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

2. Aufgabenstellung[1]

siehe Prüfungsaufgabe

3. Materialgrundlage

entfällt

4. Bezüge zum Kernlehrplan und zu den Vorgaben 2018

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

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

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

   Informatiksysteme
   - Rechnernetzwerke

2. *Medien/Materialien*
   - entfällt

5. Zugelassene Hilfsmittel

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

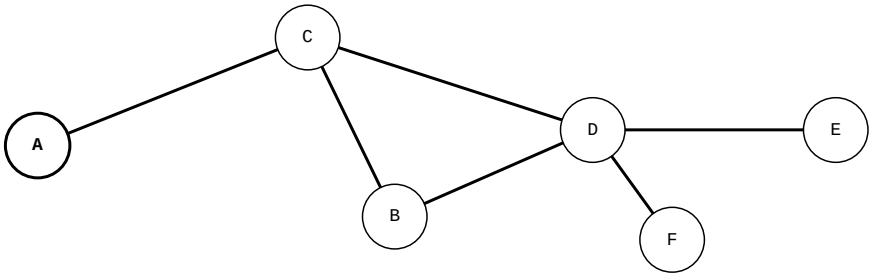
[1] Die Aufgabenstellung deckt inhaltlich alle drei Anforderungsbereiche ab.

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)

	A	B	C	D	E
A	0	1	1	0	0
B	1	0	1	1	0
C	1	1	0	1	0
D	0	1	1	0	1
E	0	0	0	1	0

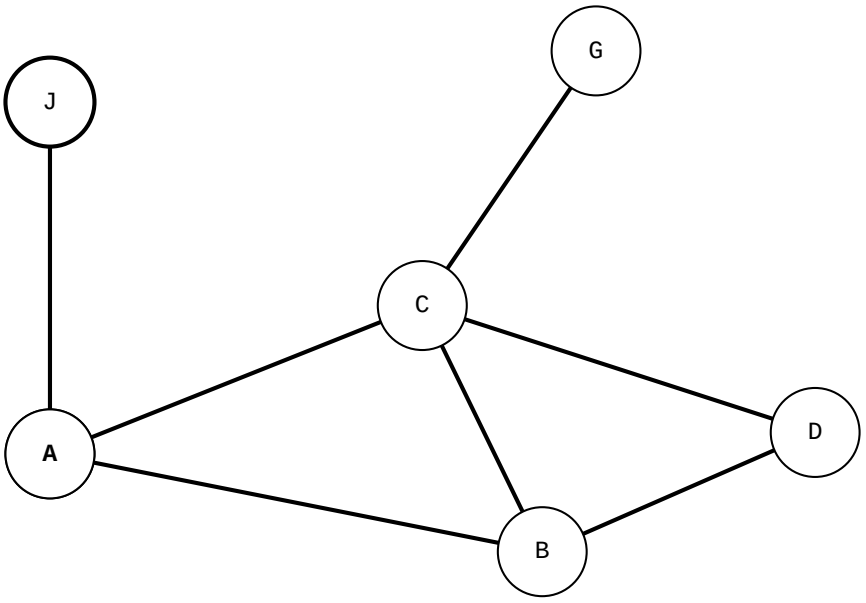


Umsetzung der Veränderungen mit der Klasse `Graph`:

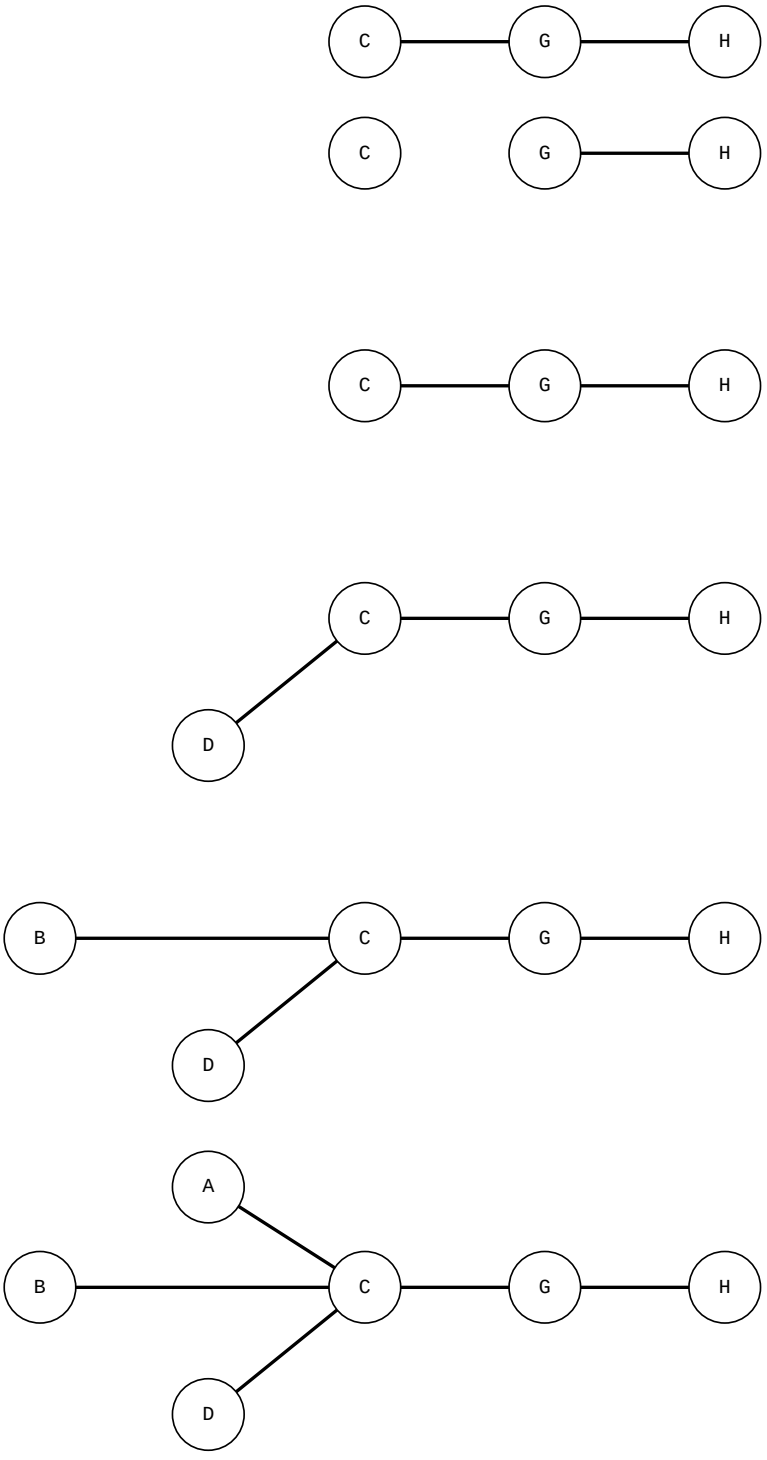
I. Die Knoten in dem ungerichteten Graphen, der das Netzwerk repräsentiert, erhalten als ID die ID des Fahrzeugs, das sie repräsentieren. Diese Knoten lassen sich daher mit der Methode `getVertex` ermitteln.
   Mit der Methode `getEdge` können Kanten ermittelt werden, die bestimmte Knoten verbinden. Dies entspricht im Sachzusammenhang einer direkten Verbindung zwischen Fahrzeugen mit den IDs der entsprechenden Knoten. Mit der Methode `removeEdge` können solche Kanten aus dem Graphen entfernt werden, was dem Wegfall der Verbindung entspricht.

ii. Um einen neuen Knoten mit der gegebenen ID zu erzeugen, wird der Konstruktor der Klasse `Vertex` aufgerufen.
   Mittels der Methode `addVertex` der Klasse `Graph` lässt sich dieses neue Knoten-Objekt einfügen, indem eine Referenz auf das Objekt als Parameter übergeben wird.
   Des Weiteren lässt sich der Knoten mit der ID `D` wie in i. beschrieben ermitteln. Mittels des Konstruktors der Klasse `Edge` und den als Parameter übergebenen Knoten-Objekten lässt sich die neue Kante erzeugen, die mittels der Methode `addEdge` der Klasse `Graph` in den Graph eingebunden werden kann.

Teilaufgabe b)



Teilaufgabe c)



Zunächst wird der Knoten, von dem die Information gesendet wurde, gegebenenfalls dem Graphen hinzugefügt. Damit ist gewährleistet, dass der sendende Knoten im Graphen enthalten ist, sofern er das noch nicht war.
Es werden alle Kanten, die diesen Knoten enthalten, gelöscht. Damit wird gewährleistet, dass weggefallene direkte Verbindungen auf jeden Fall auch im Graphen wegfallen.
Anschließend werden für diesen Knoten alle gesendeten Kanten hinzugefügt: Diese Kanten enthalten gegebenenfalls Knoten, die nicht im Graph enthalten sind. Deswegen wird zunächst für beide inzidenten Knoten der Kante geprüft, ob die Knoten im Graphen vorhanden sind. Ist einer nicht enthalten, wird dieser ergänzt. Die zugehörige Kante wird hinzugefügt.
Diese Kanten repräsentieren genau die direkten Verbindungen des sendenden Knotens. Folglich ist damit gewährleistet, dass alle direkten Verbindungen des sendenden Knotens im Graphen ergänzt werden.

Teilaufgabe d)

Um zunächst herauszufinden, welche Knoten im Graphen vom Knoten (der das Fahrzeug repräsentiert) aus nicht erreichbar sind, kann folgender Algorithmus verwendet werden:
Im Startzustand ist kein Knoten markiert.
Der Startknoten entspricht dem Knoten des Fahrzeugs und ist das erste Element einer Liste von nicht markierten Knoten. Nun werden folgende Schritte wiederholt, bis diese Liste leer ist:

- Der erste Knoten wird markiert und der aktuelle Knoten aus der Liste gelöscht.
- Alle nicht markierten Nachbarn dieses Knotens werden an die Liste angehängt.

Im Anschluss wird die Liste aller Knoten des Graphen durchlaufen und die nicht markierten Knoten werden gelöscht.

```
private void loescheNichtErreichbare() {
  //Setze alle Knoten als nicht markiert.
  umgebung.setAllVertexMarks(false);
  //Initialisiere eine Liste mit noch zu bearbeitenden Knoten.
  List<Vertex> unmarkierte = new List<Vertex>();
  unmarkierte.append(umgebung.getVertex(ID));
  unmarkierte.toFirst();

  //Wiederhole, solange unmarkierte Knoten vorhanden sind.
  while (unmarkierte.hasAccess()) {
    //Entferne den ersten Knoten und markiere ihn.
    Vertex aktuell = unmarkierte.getContent();
    aktuell.setMark(true);
    unmarkierte.remove();

    //Fuege alle unmarkierten Nachbarn der Liste hinzu.
    List<Vertex> nachbarn = umgebung.getNeighbours(aktuell);
    nachbarn.toFirst();
    while (nachbarn.hasAccess()) {
      if (!nachbarn.getContent().isMarked()) {
        unmarkierte.append(nachbarn.getContent());
      }
      nachbarn.next();
    }
  }
  //Loesche alle unmarkierten Knoten im Graphen.
  List<Vertex> alleKnoten = umgebung.getVertices();
  alleKnoten.toFirst();
  while (alleKnoten.hasAccess()) {
    if (!alleKnoten.getContent().isMarked()) {
      umgebung.removeVertex(alleKnoten.getContent());
    }
    alleKnoten.next();
  }
}
```

Teilaufgabe e)

Da das Fahrzeug mit der ID P keine weiteren direkten Verbindungen als die neu aufgebaute hat, überträgt es sowohl nach dem alten Verfahren, als auch nach dem neuen Verfahren genau eine Kante an das Fahrzeug mit der ID K: [P – K]

Das Fahrzeug mit der ID K überträgt jedoch unterschiedliche Daten. Nach dem alten Verfahren überträgt es alle direkten Verbindungen an das Fahrzeug mit der ID P:

[K – P, K – L, K – M, K – N]

Nach dem neuen Verfahren wird nur die Änderung übertragen:

[K – P]

Die zu K benachbarten Fahrzeuge mit den IDs L, M und N erhalten alle die Information über die neue Verbindung, aber nicht Informationen über bereits bestehende Verbindungen. Damit werden tatsächlich weniger Daten übertragen.

Das neu hinzukommende Fahrzeug mit der ID P erhält jedoch keine neuen Informationen vom Fahrzeug K, sondern nur die, die es bereits kennt. Damit kann es die Repräsentation des eigenen Netzwerkmodells nicht erweitern.

Das neue Verfahren erscheint noch sehr unausgewogen, da zwar weniger Daten übertragen werden, jedoch weiterhin redundante Daten weitergegeben werden und im Vergleich zum alten Modell insbesondere der Aufbau der Netzwerkrepräsentationen für neu hinzukommende Fahrzeuge nur deutlich langsamer möglich oder gar unmöglich ist.

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 in einer geeigneten Tabelle für jeden Knoten des Graphs aus Abbildung 1 an, zu welchem Knoten eine Kante besteht.	3			
2	stellt die Veränderungen im Netzwerk grafisch dar.	3			
3	erläutert, wie die Veränderung am Netzwerk mithilfe der Methoden der angegeben Klassen umgesetzt werden können.	4			
Sachlich richtige Lösungsalternative zur Modelllösung: (10) …… ……					
	Summe Teilaufgabe a)	**10**			

Teilaufgabe b)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	stellt das Teilnetzwerk dar, das von dem Fahrzeug mit der angegebenen ID gespeichert ist.	5			
Sachlich richtige Lösungsalternative zur Modelllösung: (5) …… ……					
	Summe Teilaufgabe b)	**5**			

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

Teilaufgabe c)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	wendet die Methode auf Beispieldaten an und stellt die Veränderung des Netzwerks dar.	7			
2	erläutert die Strategie, nach der das Modell des Netzwerks aktualisiert wird.	5			
Sachlich richtige Lösungsalternative zur Modelllösung: (12) …… ……					
	Summe Teilaufgabe c)	**12**			

Teilaufgabe d)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	entwickelt einen Algorithmus zum Entfernen der nicht erreichbaren Fahrzeuge.	6			
2	implementiert die Methode.	9			
Sachlich richtige Lösungsalternative zur Modelllösung: (15) …… ……					
	Summe Teilaufgabe d)	**15**			

Teilaufgabe e)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	erläutert anhand der gegebenen Beispieldaten für das alte und das neue Verfahren, welche Daten zwischen den Fahrzeugen mit den ID J und K ausgetauscht werden.	4			
2	beurteilt das neue Verfahren.	4			
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**			
Übertrag der Punktsumme aus der dritten bearbeiteten Aufgabe	**50**			
Punktzahl der gesamten Prüfungsleistung	**150**			
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	150 – 143
sehr gut	14	142 – 135
sehr gut minus	13	134 – 128
gut plus	12	127 – 120
gut	11	119 – 113
gut minus	10	112 – 105
befriedigend plus	9	104 – 98
befriedigend	8	97 – 90
befriedigend minus	7	89 – 83
ausreichend plus	6	82 – 75
ausreichend	5	74 – 68
ausreichend minus	4	67 – 60
mangelhaft plus	3	59 – 50
mangelhaft	2	49 – 41
mangelhaft minus	1	40 – 30
ungenügend	0	29 – 0

Name: _____

Abiturprüfung 2018

Informatik, Leistungskurs

Aufgabenstellung:

Ein Kino möchte mit einer relationalen Datenbank die wesentlichen Informationen zu seiner Platzvergabe verwalten.

Das Kino hat mehrere Säle in denen unterschiedliche Vorstellungen laufen können. Jeder Saal hat eine bestimmte Anzahl an Reihen und in jeder Reihe eine bestimmte Anzahl an Plätzen.

In der ersten Version wird folgende Teilmodellierung in Form eines Entity-Relationship-Diagramms zugrunde gelegt:

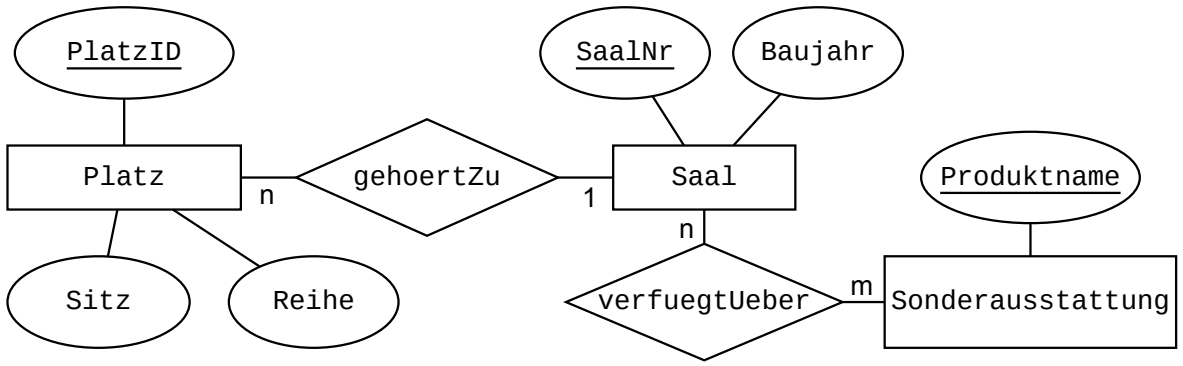


Abbildung 1: Teilmodellierung der Datenbank

a) *Erläutern Sie die in Abbildung 1 gegebene Teilmodellierung und gehen Sie dabei auch auf die Beziehungstypen ein.*

Geben Sie das Relationenschema für den Beziehungstyp `verfuegtUeber` *an.*

Erläutern Sie allgemein, wie 1:n- und n:m-Beziehungstypen in Relationenschemata überführt werden können.

(10 Punkte)

Name: _____

b) Für die Sonderausstattungen der Säle liegt ein alternativer Vorschlag vor, der im folgenden Relationenschema dargestellt ist:

Sonderausstattung(SaalNr, SaalBaujahr, ProduktID, Produktname, KategorieID, KategorieName)

Zudem sind in folgender Tabelle einige Beispieldatensätze angegeben:

Sonderausstattung					
SaalNr	SaalBaujahr	ProduktID	Produktname	KategorieID	KategorieName
5	1990	17	SuperKlang	9	Soundsysteme
5	1990	21	3DProjektor	2	Projektoren
9	2015	17	SuperKlang	9	Soundsysteme
9	2015	121	Rüttelsessel	4	Sessel
10	1990	42	BassExtrem	9	Soundsysteme

Überführen Sie das Relationenschema Sonderausstattung *zunächst in die erste, dann in die zweite und schließlich in die dritte Normalform und erläutern Sie jeweils ihre Änderungen anhand der Anforderungen der einzelnen Normalformen.*

(8 Punkte)

Name: _____

Für konkrete Datenbankanfragen soll nun das folgende weiterentwickelte Datenbankschema als Grundlage verwendet werden.

Film(FilmID, Titel, Mindestalter, Erscheinungsjahr)

Saal(SaalNr, Baujahr, Bodenflaeche)

Vorstellung(VorstellungID, ↑FilmID, ↑SaalNr, Termin)

Buchung(BuchungsNr, ↑VorstellungID, ↑KundenNr)

Platz(PlatzID, Reihe, Sitz, ↑SaalNr)

belegt(↑BuchungsNr, ↑PlatzID)

Abbildung 2: Teilmodellierung der Datenbank bezüglich der Vorstellungen

c) Es seien die folgenden SQL-Abfragen an die Datenbank gegeben.

i)
```
1 SELECT Vorstellung.Termin, Film.Titel
2 FROM Vorstellung
3   INNER JOIN Buchung
4     ON Vorstellung.VorstellungID = Buchung.VorstellungID
5   INNER JOIN Film
6    ON Film.FilmID = Vorstellung.FilmID
7 WHERE Buchung.Kundennummer = 2
```

ii)
```
1 SELECT DISTINCT Buchung.Kundennummer
2 FROM Buchung
3   INNER JOIN
4     (SELECT Vorstellung.VorstellungID
5      From Film
6        INNER JOIN Vorstellung
7          ON Film.FilmID = Vorstellung.FilmID
8      WHERE Film.Titel = 'Zeus') AS a
9   ON Buchung.VorstellungID = a.VorstellungID
```

Analysieren und erläutern Sie die obigen SQL-Anweisungen.

Erläutern Sie im Sachzusammenhang, welche Informationen die SQL-Anweisungen ermitteln.
(10 Punkte)

Name: _____

d) Aus der Datenbank mit den Relationenschemata aus Abbildung 2 sollen nun die folgenden Informationen abgefragt werden:

i) Für die Vorstellung mit der `VorstellungID 1` sollen Reihe und Sitz der gebuchten Plätze ermittelt werden. Dabei soll zunächst nach Reihe und dann nach Sitz aufsteigend sortiert werden.

ii) Es sollen die Titel der Filme ermittelt werden, für die keine Buchungen vorgenommen wurden.

iii) In der Datenbank ist die Bodenfläche der Kinosäle in Quadratmetern verzeichnet. Die Teppichböden sollen nun erneuert werden. Säle vor 1995 bekommen einen neuen Teppich für 50 Euro pro Quadratmeter. Säle, die später gebaut wurden, erhalten einen neuen Teppichboden für 10 Euro pro Quadratmeter. Es soll für jeden Saal ermittelt werden, wie viel dessen Renovierung kostet.

Entwickeln Sie SQL-Anweisungen, mit denen die geforderten Informationen aus der Datenbank abgefragt werden können.

(14 Punkte)

e) Ein Datenbankexperte ist mit der Modellierung des Entitätstyps `Platz` aus Abbildung 2 nicht einverstanden. Er behauptet, dass das Attribut `PlatzID`, das nur als Primärschlüssel dient, in diesem Entitätstyp überflüssig sei. Es könne nämlich ein aus anderen Attributen der Datenbank zusammengesetzter Primärschlüssel gewählt werden, der die Datensätze des Entitätstyps `Platz` eindeutig identifiziert. Allerdings behauptet er auch, dass dieser zusammengesetzte Primärschlüssel in der üblichen Notationsform des Entity-Relationship-Diagramms nicht dargestellt werden könne.

Begründen Sie, warum die Behauptungen des Datenbankexperten zutreffen und entwerfen Sie ein neues Relationenschema für den Entitätstyp `Platz`*, das die eindeutige Identifikation der Datensätze aus anderen Attributen der Datenbank ermöglicht.*

Vergleichen Sie das Relationenschema zu `Platz` *aus Abbildung 2 mit Ihrem Entwurf im Gesamtkontext der zu entwickelnden Datenbank.*

(8 Punkte)

Zugelassene Hilfsmittel:

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

Unterlagen für die Lehrkraft

Abiturprüfung 2018

Informatik, Leistungskurs

1. Aufgabenart

Analyse, Modellierung und Abfrage relationaler Datenbanken

2. Aufgabenstellung[1]

siehe Prüfungsaufgabe

3. Materialgrundlage

entfällt

4. Bezüge zum Kernlehrplan und zu den Vorgaben 2018

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)

Die in Abbildung 1 gegebene Teilmodellierung besteht aus drei Entitätstypen und zwei Beziehungstypen.
Im Entitätstyp `Platz` wird der Primärschlüssel in Form einer ID modelliert (Attribut: `PlatzID`). Beim Entitätstyp `Platz` werden zusätzlich zum Primärschlüssel die Nichtschlüsselattribute `Reihe` und `Sitz` geführt.
Im Entitätstyp `Saal` wird der Primärschlüssel in Form einer Nummer modelliert (Attribut: `SaalNr`). Hinzu kommt noch das Nichtschlüsselattribut `Baujahr`.
Der Entitätstyp `Sonderausstattung` verfügt nur über das Primärschlüsselattribut `Produktname`.
Der Beziehungstyp `gehoertZu` modelliert, welcher Platz zu welchem Saal gehört. Da es sich um einen 1:n-Beziehungstyp handelt, kann ein Platz nur einem Saal zugeordnet werden. Ein Saal kann hingegen mehrere Plätze haben.
Der Beziehungstyp `verfuegtUeber` modelliert, welcher Saal welche Sonderausstattungen hat. Da es sich hier um einen n:m-Beziehungstyp handelt, kann ein Saal mehrere Sonderausstattungen haben und eine Sonderausstattung in mehreren Sälen verbaut sein.

Das Relationenschema für den Beziehungstyp `verfuegtUeber` könnte folgendermaßen aussehen:
`verfuegtUeber`(↑`SaalNr`, ↑`Produktname`)

Einem Datensatz der rechten Seiten des 1:n-Beziehungstyps kann höchstens ein Datensatz der linken Seite des 1:n-Beziehungstyps zugeordnet werden. Umgekehrt können einem Datensatz der linken Seite des 1:n-Beziehungstyps beliebig viele Datensätze der rechten Seite des 1:n-Beziehungstyps zugeordnet werden. Es ist demnach möglich, die Beziehung über ein eindeutiges Fremdschlüsselattribut in einem Datensatz der rechten Seite des 1:n-Beziehungstyps zu übersetzen.
Bei einem n:m-Beziehungstyp wird eine Beziehungsrelation erstellt, die die Datensätze der beiden Entitätentypen miteinander verbindet. Der Primärschlüssel wird in der Regel aus den Primärschlüsselattributen beider Entitätstypen gebildet.

Teilaufgabe b)

Die erste Normalform ist erfüllt, wenn alle Attribute atomar sind. Weder die Attributbezeichner noch die Beispieldaten lassen daran zweifeln.
Ein Datenbankschema ist in der 2. Normalform, wenn es in der 1. Normalform ist und zusätzlich jedes Attribut, das nicht selbst zum Schlüssel gehört, nur von allen Schlüsselattributen funktional abhängig ist und nicht bereits von einem Teil der Schlüsselattribute.
In der vorliegenden Relation hängt das Attribut `SaalBaujahr` ausschließlich von der `SaalNr` und die Attribute `Produktname, KategorieID` und `Kategorienname` ausschließlich von der `ProduktID` ab.

Saal(SaalNr, Baujahr)
Sonderausstattung(ProduktID, Produktname, KategorieID,
KategorieName)
verfuegtUeber(↑SaalNr, ↑ProduktID)

Ein Datenbankschema ist in der 3. Normalform, wenn es in der 2. Normalform ist und es zusätzlich kein Nichtschlüsselattribut gibt, das transitiv von einem Schlüsselattribut abhängig ist. Es darf also keine funktionalen Abhängigkeiten von Attributen geben, die selbst nicht zum Schlüssel gehören.

Im Relationenschema `Produkt` hängt das Attribut `KategorieName` von der `KategorieID` ab. Dabei ist `KategorieID` kein Schlüsselattribut.

Saal(SaalNr, Baujahr)
Produkt(ProduktID, Produktname, ↑KategorieID)
Kategorie(KategorieID, KategorieName)
verfuegtUeber(↑SaalNr, ↑ProduktID)

Teilaufgabe c)

i) Mit den Attributen `Termin` und `Titel` wird je ein Attribut aus den Relationen `Vorstellung` und `Film` über eine mithilfe von zwei `JOIN`-Befehlen zusammengestellte Relation genommen. Die beiden `JOIN`-Befehle beziehen sich auf die Relationen `Vorstellung`, `Buchung` und `Film`, wobei zunächst `Vorstellung` und `Buchung` genau dann zusammengeführt werden, wenn in den jeweiligen Datensätzen der Relationen die Attributausprägungen im Attribut `VorstellungID` gleich sind. Die Datensätze der daraus entstandenen Relation werden dann mit denen der Relation Film zusammengeführt, wenn die Attributausprägungen bezüglich der `FilmID` gleich sind. Die so entstandenen Datensätze werden abschießend über eine `WHERE`-Klausel insofern reduziert, dass nur Datensätze mit der Kundennummer `2` betrachtet werden.
Es werden die Termine und zugehörigen Filmtitel ausgegeben, die der Kunde mit der Kundennummer `2` gebucht hat.

ii) Es werden mit einfacher Häufigkeit die Kundennummern aus den Datensätzen selektiert, die bei der Verbindung der Tabelle `Buchung` mit einer weiteren aus einem Verbund hervorgekommenen Tabelle entstanden sind. Die Verbindung der Tabellen der Unterabfrage bezieht sich auf die Tabellen `Film` und `Vorstellung`. Dabei werden nur Datensätze verknüpft, bei denen die `FilmID` identisch ist. Zudem wird die Menge der Datensätze dahingehend eingeschränkt, dass nur Datensätze übrigbleiben, bei denen der Titel „Zeus“ lautet. Die Unterabfrage wird `a` genannt und wird anschließend mit der Tabelle `Buchung` zusammengeführt. Dabei werden nur Datensätze gebildet, bei denen `VorstellungsID` identisch ist.
Es werden die Nummern der Kunden ermittelt, die den Film „Zeus“ gebucht haben. Jede so ermittelte Nummer kommt in der Auflistung nur einmal vor.

Teilaufgabe d)

(i)

```
1 SELECT Platz.Reihe, Platz.Sitz
2 FROM Platz
3   INNER JOIN Vorstellung
4     ON Platz.SaalNr = Vorstellung.SaalNr
5 WHERE Vorstellung.VorstellungID = 1
6 ORDER BY Platz.Reihe, Platz.Sitz
```

(ii)

```
1 SELECT Film.Titel
2 FROM Film
3 WHERE Film.FilmID NOT IN
4       (SELECT DISTINCT Vorstellung.FilmID
5        FROM Vorstellung
6             INNER JOIN Buchung
7                ON Vorstellung.VorstellungID
8                         = Buchung.VorstellungID
9       )
```

(iii)

```
1 SELECT Saal.SaalNr, Saal.Bodenflaeche * 50
2 FROM Saal
3 WHERE Saal.Baujahr < 1995
4 UNION
5 SELECT Saal.SaalNr, Saal.Bodenflaeche * 10
6 FROM Saal
7 WHERE Saal.Baujahr >= 1995
```

Teilaufgabe e)

Der Datenbankexperte hat insofern Recht, als das in einem Entity-Relationship-Diagramm keine Fremdschlüssel markiert werden. Diese Möglichkeit gibt es erst bei den relationalen Modellen.
Ein zusammengesetzter Primärschlüssel im Relationenschema für `Platz` könnte so aussehen:

`Platz`(↑`SaalNr`, `Reihe`, `Sitz`)

Der in Abbildung 2 gewählte Entwurf bietet den Vorteil, dass die Relation `belegt` nur je ein Attribut (den jeweiligen Primärschlüssel) referenzieren muss. Würde man den aus drei Attributen zusammengesetzten natürlichen Primärschlüssel nehmen, müsste man all diese Attribute in `belegt` aufführen. Nachvollziehbarer und auch wirklichkeitsnäher als ein künstlicher Schlüssel ist die Wahl des natürlichen zusammengesetzten Primärschlüssels.

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 gegebene Teilmodellierung und geht auch auf die Beziehungstypen ein.	5			
2	gibt das Relationenschema für den Beziehungstyp `verfuegtUeber` an.	3			
3	erläutert allgemein, wie 1:n- und n:m-Beziehungstypen in Relationenschemata überführt werden können.	2			
Sachlich richtige Lösungsalternative zur Modelllösung: (10) …… ……					
	Summe Teilaufgabe a)	**10**			

Teilaufgabe b)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	erläutert, dass die erste Normalform erfüllt ist.	2			
2	überführt das Relationenschema in die zweite und dann in die dritte Normalform.	4			
3	erläutert die Änderungen anhand der Anforderungen für die entsprechenden Normalformen.	2			
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	analysiert und erläutert die erste SQL-Anweisung.	4			
2	analysiert und erläutert die zweite SQL-Anweisung.	4			
3	erläutert im Sachzusammenhang, welche Informationen die SQL-Anweisungen ermitteln.	2			
Sachlich richtige Lösungsalternative zur Modelllösung: (10) …… ……					
	Summe Teilaufgabe c)	**10**			

Teilaufgabe d)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	entwickelt für die erste Anfrage eine SQL-Anweisung.	4			
2	entwickelt für die zweite Anfrage eine SQL-Anweisung.	5			
3	entwickelt für die dritte Anfrage eine SQL-Anweisung.	5			
Sachlich richtige Lösungsalternative zur Modelllösung: (14) …… ……					
	Summe Teilaufgabe d)	**14**			

Teilaufgabe e)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	begründet, warum die Behauptungen zutreffen.	3			
2	entwirft ein neues Relationenschema, welches die eindeutige Identifikation der Datensätze aus anderen Attributen der Datenbank ermöglicht.	3			
3	vergleicht das Relationenschema `Platz` mit dem eigenen Entwurf.	2			
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**			
Übertrag der Punktsumme aus der dritten bearbeiteten Aufgabe	**50**			
Punktzahl der gesamten Prüfungsleistung	**150**			
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	150 – 143
sehr gut	14	142 – 135
sehr gut minus	13	134 – 128
gut plus	12	127 – 120
gut	11	119 – 113
gut minus	10	112 – 105
befriedigend plus	9	104 – 98
befriedigend	8	97 – 90
befriedigend minus	7	89 – 83
ausreichend plus	6	82 – 75
ausreichend	5	74 – 68
ausreichend minus	4	67 – 60
mangelhaft plus	3	59 – 50
mangelhaft	2	49 – 41
mangelhaft minus	1	40 – 30
ungenügend	0	29 – 0

Name: _____

Abiturprüfung 2018

Informatik, Leistungskurs

Aufgabenstellung:

Die Fastfoodkette OwnBurger4U möchte einen Burgerprüfer entwickeln, mit dem kontrolliert werden kann, ob die Burger zulässig zusammengestellt sind. Die Burger werden dabei von unten nach oben belegt.

Zeichen	Erklärung
b	Scheibe Brot
f	Fleisch
s	Salat
k	Ketchup
c	Käse (Cheese)

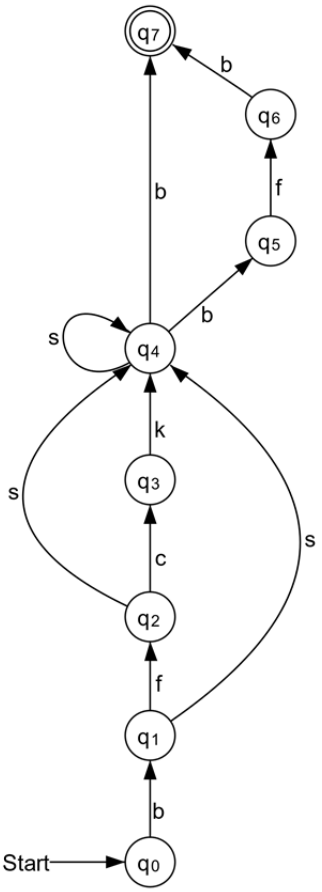


Abbildung 1: Zustandsübergangsdiagramm zum Automaten A

Name: _____

a) Das Zustandsübergangsdiagramm (Abbildung 1) gibt an, in welcher Reihenfolge die Burgerzutaten gewählt werden dürfen.

Gegeben sind die Zeichenketten `bfsbfb` und `bsskb`, die als Eingabewörter mit Hilfe des Automaten `A` überprüft werden sollen.

Geben Sie die Zustandsübergangstabelle des Automaten `A` *an.*

Geben Sie alle Zustandsfolgen des Automaten bei der Abarbeitung beider Eingabewörter an.

Erläutern Sie anhand des Automaten `A`*, warum eines der beiden Eingabewörter akzeptiert wird und das andere nicht akzeptiert wird.*

Geben Sie ein Wort minimaler Länge und ein Wort mit der Länge 10 an, welches jeweils vom Automaten akzeptiert wird.

Begründen Sie, warum der Automat `A` *ein nichtdeterministischer endlicher Automat ist.*

(13 Punkte)

b) Der Automat `A'` ist definiert als der Automat `A` ohne die Zustände q_5 und q_6 und die entsprechenden Zustandsübergänge.

Die Methode `testeEingabe` soll prüfen, ob die als Parameter übergebene Zeichenkette `pEingabe` eine gültige Burgerkonfiguration für den Automaten `A'` dargestellt.

Die Methode gibt genau dann `true` zurück, wenn das Eingabewort vom Automaten `A'` akzeptiert wird.

Methodenkopf:

```
public boolean testeEingabe(String pEingabe)
```

Hinweis zur Klasse `String`

Anfrage **`char charAt(int i)`**
Gibt das Zeichen an Position `i` zurück. Die vorderste Position ist `0`.

Anfrage **`int length()`**
Gibt die Länge des Strings zurück.

Implementieren Sie die Methode `testeEingabe`.

(8 Punkte)

Name: _____

c) Es soll eine zweite Version für den Burgerprüfer entwickelt werden. Die Anforderungen sind wie folgt:

i) Ein Burger kann beliebig groß werden, d. h., es dürfen beliebig viele Scheiben Brot (`b`) und beliebig viele Zutaten verwendet werden, solange folgende Regeln berücksichtigt werden:

ii) Ein Burger beginnt und endet mit einer Scheibe Brot (`b`).

iii) Zwei oder mehr Scheiben Brot (`b`) dürfen nicht direkt aufeinanderliegen.

iv) Als Füllung stehen in dieser Version nur die Zutaten Fleisch (`f`) und Salat (`s`) und Ketchup (`k`) zur Verfügung.
Zwischen zwei Schreiben Brot (`b`) liegt entweder eine Zutat oder zwei unterschiedliche Zutaten oder drei unterschiedliche Zutaten. Die Zutaten sollen in einer beliebigen Reihenfolge erlaubt sein.

Entwerfen sie einen nichtdeterministischen endlichen Automaten (NEA), der die genannten Anforderungen erfüllt.

Erläutern Sie, wie Sie die ersten drei Anforderungen in Ihrem NEA berücksichtigt haben.

(12 Punkte)

d) Die folgende Grammatik `G` erzeugt die Sprache für einen Burgerkonfigurator:

Startsymbol: `S`

Nichtterminale: `N = {S, A, F}`

Terminale: `T = {b, f, s, v}`

Produktionen:
```
P = {S → bAb,
     A → Fs | FsbA,
     F → f | v
   }
```

Terminal	Erklärung
`b`	Scheibe Brot
`f`	Fleisch
`s`	Salat
`v`	vegetarischer Bratling

Erläutern Sie im Sachkontext, welche Burger (auf Grundlage der Grammatik `G`*) erzeugt werden.*

Zeigen Sie, dass sich die Zeichenkette `bfsbvsb` *aus der Grammatik* `G` *ableiten lässt.*

Erläutern Sie, warum die Grammatik `G` *keine reguläre Grammatik ist.*

Entwerfen Sie eine reguläre Grammatik, aus der sich genau die Wörter ableiten lassen, die sich aus der Grammatik `G` *ableiten lassen, und zusätzlich die Wörter, die Burger beschreiben, die nach Salat Ketchup (Terminal* `k`*) enthalten können.*

(12 Punkte)

Name: _____

e) In einer ganz neuen Version des Burgerprüfers soll geprüft werden, ob ein Burger symmetrisch aufgebaut ist. Ein symmetrischer Burger ist von unten nach oben sowie von oben nach unten von den Zutaten identisch (z. B. `bfkkfb` und `bfckcfb`).

Beurteilen Sie, inwiefern sich diese Version des Burgerprüfers jeweils als deterministischer endlicher Automat oder als nichtdeterministischer endlicher Automat realisieren lässt.

(5 Punkte)

Zugelassene Hilfsmittel:

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

Unterlagen für die Lehrkraft

Abiturprüfung 2018

Informatik, Leistungskurs

1. Aufgabenart

Analyse und Modellierung 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 2018

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
   - Grammatiken regulärer Sprachen
   - Scanner, Parser und Interpreter für eine reguläre Sprache
   - 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)

Zustandsübergangstabelle für den Automaten A:

	b	**c**	**f**	**k**	**s**
q_0	q_1				
q_1			q_2		q_4
q_2		q_3			q_4
q_3				q_4	
q_4	$\{q_5, q_7\}$				q_4
q_5			q_6		
q_6	q_7				
q_7					

Die möglichen Zustandsfolgen für die Zeichenkette `bfsbfb` sind folgende:

$q_0 \xrightarrow{b} q_1 \xrightarrow{f} q_2 \xrightarrow{s} q_4 \xrightarrow{b} q_5 \xrightarrow{f} q_6 \xrightarrow{b} q_7$ (Automat akzeptiert)

oder: $q_0 \xrightarrow{b} q_1 \xrightarrow{f} q_2 \xrightarrow{s} q_4 \xrightarrow{b} q_7 \xrightarrow{f}$ FEHLER, denn es gibt vom Endzustand keinen weiteren Zustandsübergang.

Die mögliche Zustandsfolge für die Zeichenkette `bsskb` ist folgende:

$q_0 \xrightarrow{b} q_1 \xrightarrow{s} q_4 \xrightarrow{s} q_4 \xrightarrow{k}$ FEHLER, denn es geht nur mit dem Zeichen `s` oder `b` weiter.

Nach Abarbeitung des ersten Eingabewortes befindet sich der Automat `A` bei einer der Zustandsfolgen im Endzustand q_7. Der Automat akzeptiert also die Eingabe.
Bei der Abarbeitung des zweiten Eingabewortes befindet sich der Automat nach dem dritten Zeichen im Fehlerzustand. Daher akzeptiert der Automat die zweite Eingabe nicht.

Das Wort minimaler Länge: `bsb`
Ein Wort mit der Länge 10: `bfckssbfb`

Der Automat A ist ein nichtdeterministischer endlicher Automat, weil es vom Zustand q_4 zwei Zustandsübergänge zu unterschiedlichen Folgezuständen mit dem gleichen Eingabezeichen `b` gibt. Der Automat ist endlich, weil die Menge der Zustände und das Eingabealphabet endlich sind.

Teilaufgabe b)

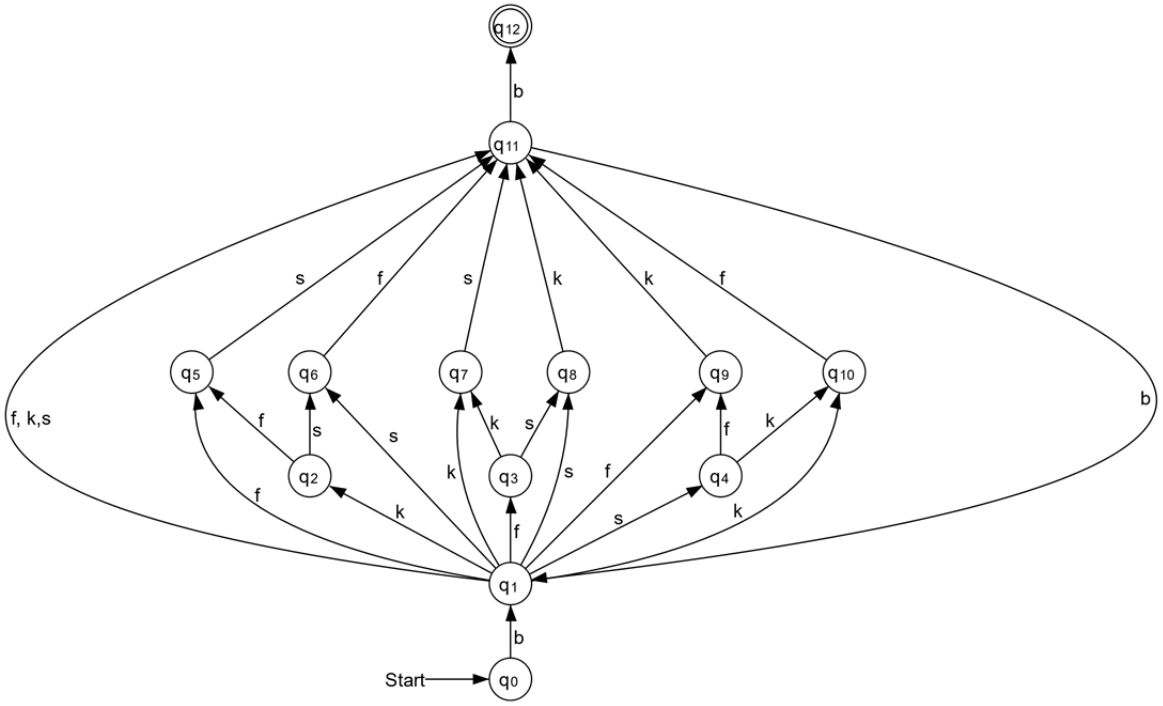
```
public boolean testeEingabe(String pEingabe) {
  int zustand = 0;
  for (int i = 0; i < pEingabe.length(); i++) {
    char terminal = pEingabe.charAt(i);
    switch (zustand) {
      case 0: // Startzustand
        if (terminal == 'b') {
          zustand = 1;
        } else {
          zustand = -1;
        }
        break;
      case 1:
        if (terminal == 's') {
          zustand = 4;
        } else if (terminal == 'f') {
          zustand = 2;
        } else {
          zustand = -1;
        }
        break;
      case 2:
        if (terminal == 's') {
          zustand = 4;
        } else if (terminal == 'c') {
          zustand = 3;
        } else {
          zustand = -1;
        }
        break;
      case 3:
        if (terminal == 'k') {
          zustand = 4;
        } else {
          zustand = -1;
        }
        break;
      case 4:
        if (terminal == 's') {
          zustand = 4;
        } else if (terminal == 'b') {
          zustand = 7;
        } else {
          zustand = -1;
        }
        break;
      case 7:
        zustand = -1;
        break;
    }
  }
  return (zustand == 7);
}
```

Teilaufgabe c)

Der folgende nichtdeterministische endliche Automat entspricht den Anforderungen.

Startzustand: q_0
Endzustände: $\{q_{12}\}$
Zustände: $\{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9, q_{10}, q_{11}, q_{12}\}$
Eingabealphabet: {b, f, s, k}

Zustandsübergangsdiagramm:



Erläuterung zur Berücksichtigung der Anforderung i:
Mit dem Zustandsübergang aus dem Endzustand q_{11} nach q_1 wird ein neuer „Zutat(en)-Brot-Zyklus“ eingeleitet.

Erläuterung zur Berücksichtigung der Anforderung ii:
Das erste Zeichen vom Startzustand und das letzte Zeichen in den Endzustand müssen das Eingabezeichen b sein müssen (vgl. Zustandsübergänge von q_0 nach q_1 und von q_{11} nach q_{12}).

Erläuterung zur Berücksichtigung der Anforderung iii:
Es ist sichergestellt, dass es keine zwei aufeinander folgenden Zustandsübergänge für das Eingabezeichen b gibt.

Teilaufgabe d)

Die Burger, die (auf Grundlage der Grammatik G) erzeugt werden, verfügen über folgende Eigenschaften:

- Die Burger können beliebig groß werden.
- Die Burger beginnen und enden mit jeweils einer Scheibe Brot.
- Zwischen zwei Scheiben Brot liegt die Füllung. In einem Burger sind mehrere Füllungen zulässig, wenn diese durch eine Scheibe Brot getrennt sind.
- Als Füllung zwischen zwei Brotscheiben liegt zunächst entweder einmal Fleisch oder ein vegetarischer Bratling. Danach folgt einmal Salat.

Das gegebene Wort bfsbvsb kann wie folgt aus der Grammatik G abgeleitet werden:

S → bAb
→ bFsbAb
→ bfsbAb
→ bfsbFsb
→ bfsbvsb

Eine reguläre Grammatik ist entweder rechtslinear oder linkslinear. Bei einer rechtslinearen Grammatik haben alle Produktionen auf der rechten Seite der Produktion entweder ein Terminal oder ein Terminal gefolgt von einem Nichtterminal. Bei einer linkslinearen Grammatik haben alle Produktionen auf der rechten Seite der Produktion entweder ein Terminal oder ein Nichtterminal gefolgt von einem Terminal.
Die Produktion S → bAb verstößt gegen diese Kriterien, da sie ein Terminal gefolgt von einem Nichtterminal gefolgt von einem Terminal auf der rechten Seite der Produktion hat.

Aus der folgenden regulären Grammatik lassen sich dieselben Wörter ableiten wie aus der Grammatik G und zusätzlich die Wörter, die Burger beschreiben, die nach Salat Ketchup (Terminal k) enthalten können.

Startsymbol: S

Nichtterminale: N = { S, A, B, C, D}

Terminale: T = { b, f, k, s, v}

Produktionen: P = { S → bA,
A → fB | vB,
B → sC,
C → kD | b | bA
D → b | bA
}

Teilaufgabe e)

Um den Aufbau eines Burgers mit Hilfe eines deterministisch endlichen Automaten (DEA) auf Symmetrie zu überprüfen, müsste man sich die Zutatenfolge bis zur Mitte des Burgers merken, um sie anschließend mit der zweiten Hälfte vergleichen zu können. Da die Burgergröße nicht begrenzt ist, wären unendlich viele Zustände notwendig.
Mit einem deterministisch endlichen Automaten kann man nicht überprüfen, ob ein beliebig großer Burger symmetrisch aufgebaut ist, weil ein endlicher Automat nur über seine Zustände „zählen“ kann und die Anzahl der Zustände endlich ist.
Da ein nichtdeterministischer endlicher Automat (NEA) in einen äquivalenten DEA umgewandelt werden kann, kann auch ein NEA nicht beliebig weit zählen.
Wenn man die Größe des Burgers allerdings begrenzt, könnte man sowohl einen DEA als auch einen NEA mit endlich vielen Zuständen für diese Version des Burgerprüfers realisieren.

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 Zustandsübergangstabelle an.	4			
2	gibt alle Zustandsfolgen bei der Abarbeitung beider Eingabewörter an.	3			
3	erläutert anhand des Automaten, warum eines der beiden Eingabewörter akzeptiert wird und das andere nicht akzeptiert wird.	2			
4	gibt ein Wort minimaler Länge und ein Wort der Länge 10 an, welches jeweils vom Automaten akzeptiert wird.	2			
5	begründet, warum der Automat A ein nichtdeterministischer endlicher Automat ist.	2			
Sachlich richtige Lösungsalternative zur Modelllösung: (13) …… ……					
	Summe Teilaufgabe a)	**13**			

Teilaufgabe b)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	implementiert die Methode	8			
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	entwirft einen NEA, der die genannten Anforderungen erfüllt.	9			
2	erläutert, wie die ersten drei Anforderungen berücksichtigt wurden.	3			
Sachlich richtige Lösungsalternative zur Modelllösung: (12) …… ……					
	Summe Teilaufgabe c)	**12**			

Teilaufgabe d)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	erläutert im Sachkontext, welche Burger (auf Grundlage der Grammatik G) erzeugt werden.	3			
2	zeigt, dass sich die Zeichenkette aus der Grammatik ableiten lässt.	2			
3	erläutert, warum die Grammatik keine reguläre Grammatik ist.	2			
4	entwirft eine reguläre Grammatik, aus der sich genau die Wörter ableiten lassen, die sich aus der Grammatik G ableiten lassen und zusätzlich die Wörter, die nach Salat Ketchup enthalten können.	5			
Sachlich richtige Lösungsalternative zur Modelllösung: (12) …… ……					
	Summe Teilaufgabe d)	**12**			

Teilaufgabe e)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	beurteilt, inwiefern sich diese Version als DEA oder als NEA realisieren lässt.	5			
Sachlich richtige Lösungsalternative zur Modelllösung: (5) …… ……					
	Summe Teilaufgabe e)	**5**			

	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**			
Übertrag der Punktsumme aus der dritten bearbeiteten Aufgabe	**50**			
Punktzahl der gesamten Prüfungsleistung	**150**			
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	150 – 143
sehr gut	14	142 – 135
sehr gut minus	13	134 – 128
gut plus	12	127 – 120
gut	11	119 – 113
gut minus	10	112 – 105
befriedigend plus	9	104 – 98
befriedigend	8	97 – 90
befriedigend minus	7	89 – 83
ausreichend plus	6	82 – 75
ausreichend	5	74 – 68
ausreichend minus	4	67 – 60
mangelhaft plus	3	59 – 50
mangelhaft	2	49 – 41
mangelhaft minus	1	40 – 30
ungenügend	0	29 – 0