Name: _____

Abiturprüfung 2019

Informatik, Grundkurs

Aufgabenstellung:

Fast jeder Radiosender verliest in regelmäßigen Abständen Verkehrsmeldungen, die sich auf Autobahnen beziehen. Im Folgenden sollen Staumeldungen betrachtet werden, wie z. B. die folgende: „A43, Wuppertal Richtung Recklinghausen, zwischen Anschlussstelle Witten-Herbede und Kreuz Herne, 11 Kilometer Stau“.

Eine solche Meldung setzt sich aus vier Elementen zusammen:

1. Autobahnbezeichnung: ein A und eine Autobahnnummer
2. Richtung: zwei Städte
3. Abschnitt: zwei Anschlussstellen (AS) bzw. Kreuze (AK)
4. Länge des Staus: ganzzahlige Kilometerzahl

Mehrere Staumeldungen werden nach der Autobahnnummer aufsteigend sortiert verlesen. Ist die Autobahnnummer gleich, werden sie in beliebiger Reihenfolge genannt. Da die Sendezeit begrenzt ist, können oft nicht alle Staumeldungen berücksichtigt werden, so dass man sich auf Staus ab einer bestimmten Länge beschränkt.

a) Die folgenden unsortierten Staumeldungen sollen zu Testzwecken verarbeitet werden:

Autobahnbezeichnung	Richtung	Abschnitt	Länge
A43	Wuppertal / Recklinghausen	AS Witten-Herbede / AK Herne	11
A1	Köln / Münster	AS Schwerte / AS Kamen-Zentrum	12
A44	Aachen / Mönchengladbach	AS Alsdorf / AS Aldenhoven	2
A46	Düsseldorf / Wuppertal	AS Wuppertal-Elberfeld / AS Wuppertal-Barmen	1
A2	Dortmund / Hannover	AS Bönen / AS Hamm	4
A40	Dortmund / Essen	AS Dortmund-Lütgendortmund / AS Bochum-Werne	1
A43	Wuppertal / Recklinghausen	AS Herne-Eickel / AS Recklinghausen-Hochlarmark	4

Abbildung 1: Tabelle mit Staumeldungen

Name: _____

Geben Sie an, welche Staumeldungen (Autobahnbezeichnung und Länge) in welcher Reihenfolge verlesen werden, wenn nur Staus mit vier oder mehr Kilometern Länge berücksichtigt werden.

Ermitteln Sie, welche Mindeststaulänge angesetzt werden muss, wenn man von den oben gegebenen Staumeldungen möglichst viele, aber maximal drei verlesen möchte.

Erläutern Sie, dass es bei der oben gegebenen Datenlage keine Mindeststaulänge gibt, die zu genau drei Staumeldungen führt.

(8 Punkte)

Um die Liste der Staumeldungen zu verwalten, wird ein Programm verwendet, das der folgenden Teilmodellierung entspricht:

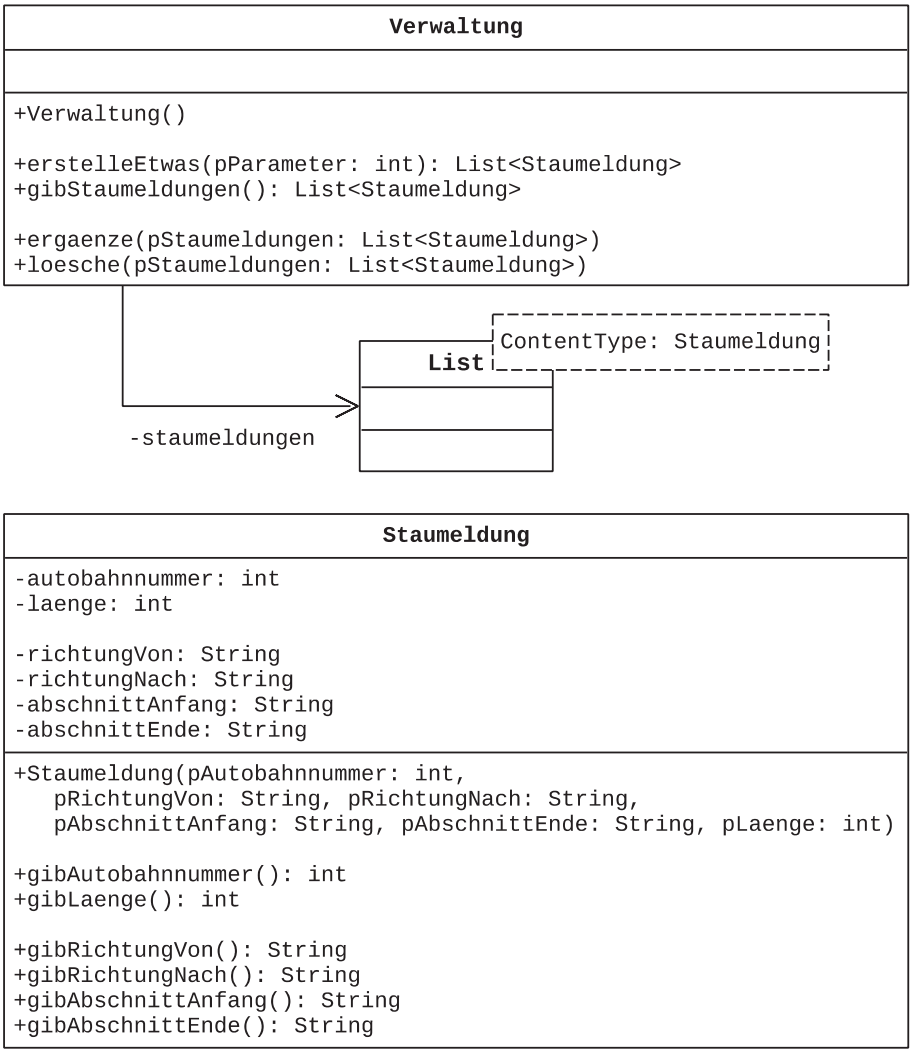


Abbildung 2: Teilmodellierung als Implementationsdiagramm

Name: _____

Die Klasse `Staumeldung` modelliert eine einzelne Staumeldung. Ein Objekt der Klasse `Verwaltung` speichert Staumeldungen in der Liste `staumeldungen` und bietet verschiedene Operationen zu deren Verwaltung an (vgl. Dokumentation im Anhang).

b) Die Methode `loesche` der Klasse `Verwaltung` wird verwendet, um Staumeldungen, die nicht mehr aktuell sind, aus dem Datenbestand zu entfernen. Sie hat den folgenden Methodenkopf:

```
public void loesche(List<Staumeldung> pStaumeldungen)
```

Entwickeln Sie ein algorithmisches Verfahren für die Methode `loesche` *der Klasse* `Verwaltung`.

Implementieren Sie die Methode `loesche` *der Klasse* `Verwaltung`.

(12 Punkte)

c) Die Methode `erstelleEtwas` der Klasse `Verwaltung` ist wie folgt implementiert:

```
 1 public List<Staumeldung> erstelleEtwas(int pParameter) {
 2   List<Staumeldung> liste = new List<Staumeldung>();
 3   staumeldungen.toFirst();
 4   while (staumeldungen.hasAccess()) {
 5     Staumeldung akt = staumeldungen.getContent();
 6     if (akt.gibLaenge() >= pParameter) {
 7       liste.toFirst();
 8       while (liste.hasAccess()
 9              && liste.getContent().gibAutobahnnummer()
10                 < akt.gibAutobahnnummer()) {
11         liste.next();
12       }
13       if (liste.hasAccess()) {
14         liste.insert(akt);
15       } else {
16         liste.append(akt);
17       }
18     }
19     staumeldungen.next();
20   }
21   return liste;
22 }
```

Analysieren Sie die Methode `erstelleEtwas` *der Klasse* `Verwaltung` *und erläutern Sie ihre Funktionsweise.*

Erläutern Sie, welche Information die Methode im Sachzusammenhang ermittelt.

(12 Punkte)

Name: _____

d) Unter Verkehrsmeldungen versteht man in der Regel nicht nur Staumeldungen. Von besonderer Bedeutung sind Warnungen z. B. vor Falschfahrern oder Tieren auf der Fahrbahn. Da solche Meldungen außergewöhnlich wichtig sind, werden sie nicht in die anderen Meldungen eingeordnet, sondern am Anfang und noch einmal am Ende mit einem individuellen Warntext verlesen.

Entwerfen Sie auf Grundlage der Modellierung in Abbildung 2 eine erweiterte Modellierung in Form eines Implementationsdiagramms, die auch Warnmeldungen der obigen Art berücksichtigt.

Erläutern Sie Ihre erweiterte Modellierung.

Hinweis: Die in Teilaufgabe c) zu analysierende Methode `erstelleEtwas` der Klasse `Verwaltung` soll nicht berücksichtigt werden.

(12 Punkte)

e) Um das Programm zu verbessern, macht ein Mitarbeiter folgenden Vorschlag:

> „Wenn es auf einer Autobahn zwei Staus in derselben Richtung gibt, dann kann man beide Meldungen zu einer zusammenfassen, wenn sie unmittelbar aufeinander folgen, d. h. das Ende des einen Staus der Anfang des anderen ist."

Entwerfen Sie ein algorithmisches Verfahren, nach dem die oben beschriebene Verbesserung programmiertechnisch umgesetzt werden kann.

(6 Punkte)

Zugelassene Hilfsmittel:

- GTR (grafikfähiger Taschenrechner) oder CAS (Computer-Algebra-System)
- Wörterbuch zur deutschen Rechtschreibung

Name: _____

Anhang

Dokumentationen der verwendeten Klassen

Die Klasse Verwaltung

Ein Objekt der Klasse `Verwaltung` wird verwendet, um Staumeldungen zu verwalten und auszuwerten.

Ausschnitt aus der Dokumentation der Klasse `Verwaltung`

Konstruktor **`Verwaltung()`**
Initialisiert ein Objekt der Klasse `Verwaltung` und erstellt eine leere Liste für gespeicherte Staumeldungen.

Anfrage **`List<Staumeldung> erstelleEtwas(int pParameter)`**
Diese Methode soll in Teilaufgabe c) analysiert werden.

Anfrage **`List<Staumeldung> gibStaumeldungen()`**
Liefert die Liste der gespeicherten Staumeldungen zurück.

Auftrag **`void ergaenze(List<Staumeldung> pStaumeldungen)`**
Fügt die Staumeldungen in der Liste `pStaumeldungen` am Ende der Liste der gespeicherten Staumeldungen hinzu.

Auftrag **`void loesche(List<Staumeldung> pStaumeldungen)`**
Löscht alle Objekte der Liste `pStaumeldungen` aus der Liste der gespeicherten Staumeldungen.

Name: _____

Die Klasse Staumeldung

Die Klasse `Staumeldung` modelliert eine einzelne Staumeldung.

Ausschnitt aus der Dokumentation der Klasse `Staumeldung`

Konstruktor **`Staumeldung(int pAutobahnnummer, String pRichtungVon, String pRichtungNach, String pAbschnittAnfang, String pAbschnittEnde, int pLaenge)`**
Initialisiert ein Objekt der Klasse `Staumeldung` und speichert die Werte der Parameter.

Anfrage **`int gibAutobahnnummer()`**
Liefert die Autobahnnummer der Staumeldung als `int`.

Anfrage **`String gibRichtungVon()`**
Liefert den ersten Städtenamen der Richtungsangabe des Staus als `String`.

Anfrage **`String gibRichtungNach()`**
Liefert den zweiten Städtenamen der Richtungsangabe des Staus als `String`.

Anfrage **`String gibAbschnittAnfang()`**
Liefert die Anschlussstelle bzw. das Autobahnkreuz, nach welcher bzw. welchem der Stau beginnt, als `String`.

Anfrage **`String gibAbschnittEnde()`**
Liefert die Anschlussstelle bzw. das Autobahnkreuz, vor welcher bzw. welchem der Stau endet, als `String`.

Anfrage **`int gibLaenge()`**
Liefert die Länge des Staus in Kilometern als `int`.

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 2019

Informatik, Grundkurs

1. Aufgabenart

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

2. Aufgabenstellung[1]

siehe Prüfungsaufgabe

3. Materialgrundlage

entfällt

4. Bezüge zum Kernlehrplan und zu den Vorgaben 2019

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

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

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

2. *Medien/Materialien*
   - entfällt

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

5. Zugelassene Hilfsmittel

- GTR (grafikfähiger Taschenrechner) oder CAS (Computer-Algebra-System)
- Wörterbuch zur deutschen Rechtschreibung

6. Modelllösungen

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

Teilaufgabe a)

Wenn nur Staus mit vier oder mehr Kilometern verlesen werden, so sind das die folgenden Staus in der folgenden Reihenfolge:
A1 mit 12 km, A2 mit 4 km, A43 mit 11 km und noch einmal A43 mit 4 km.

In welcher Reihenfolge die Staus auf der A43 verlesen werden, wird durch die Aufgabenstellung nicht geklärt.

Sollen möglichst viele, aber maximal drei Staumeldungen verlesen werden, so ist eine Mindeststaulänge zwischen 5 und 11 Kilometern anzusetzen. In diesem Fall werden nur jeweils ein Stau auf den Autobahnen A1 und A43 berücksichtigt.

Senkt man die Mindestlänge auf vier Kilometer ab, kommen gleich zwei Staus dazu: auf der A2 und ein weiterer auf der A43. Daraus ergibt sich, dass man bei den gegebenen Staumeldungen nie auf exakt drei Meldungen kommt.

Teilaufgabe b)

Die Methode `loesche` kann entsprechend dem folgenden algorithmischen Verfahren realisiert werden:
Die Liste der zu löschenden Staumeldungen `pStaumeldungen` muss in einer äußeren Wiederholung vom ersten bis zum letzten Element durchlaufen werden.

In einer inneren Wiederholung wird für jedes Element der Liste `pStaumeldungen` die Liste der gespeicherten Staumeldungen `staumeldungen` durchlaufen. Wird dabei das aktuelle Element der Liste `pStaumeldungen` gefunden, wird es aus der Liste `staumeldungen` entfernt. Die innere Wiederholung kann dann beendet werden, so dass beim nächsten Element, das zu löschen ist, fortgefahren wird.

Die Methode `loesche` kann entsprechend diesem Verfahren wie folgt implementiert werden:

```
public void loesche(List<Staumeldung> pStaumeldungen) {
  pStaumeldungen.toFirst();
  while (pStaumeldungen.hasAccess()) {
    staumeldungen.toFirst();
    boolean fertig = false;
    while (staumeldungen.hasAccess() && !fertig) {
      if (staumeldungen.getContent()
          == pStaumeldungen.getContent()) {
        staumeldungen.remove();
        fertig = true;
      } else {
        staumeldungen.next();
      }
    }
    pStaumeldungen.next();
  }
}
```

Teilaufgabe c)

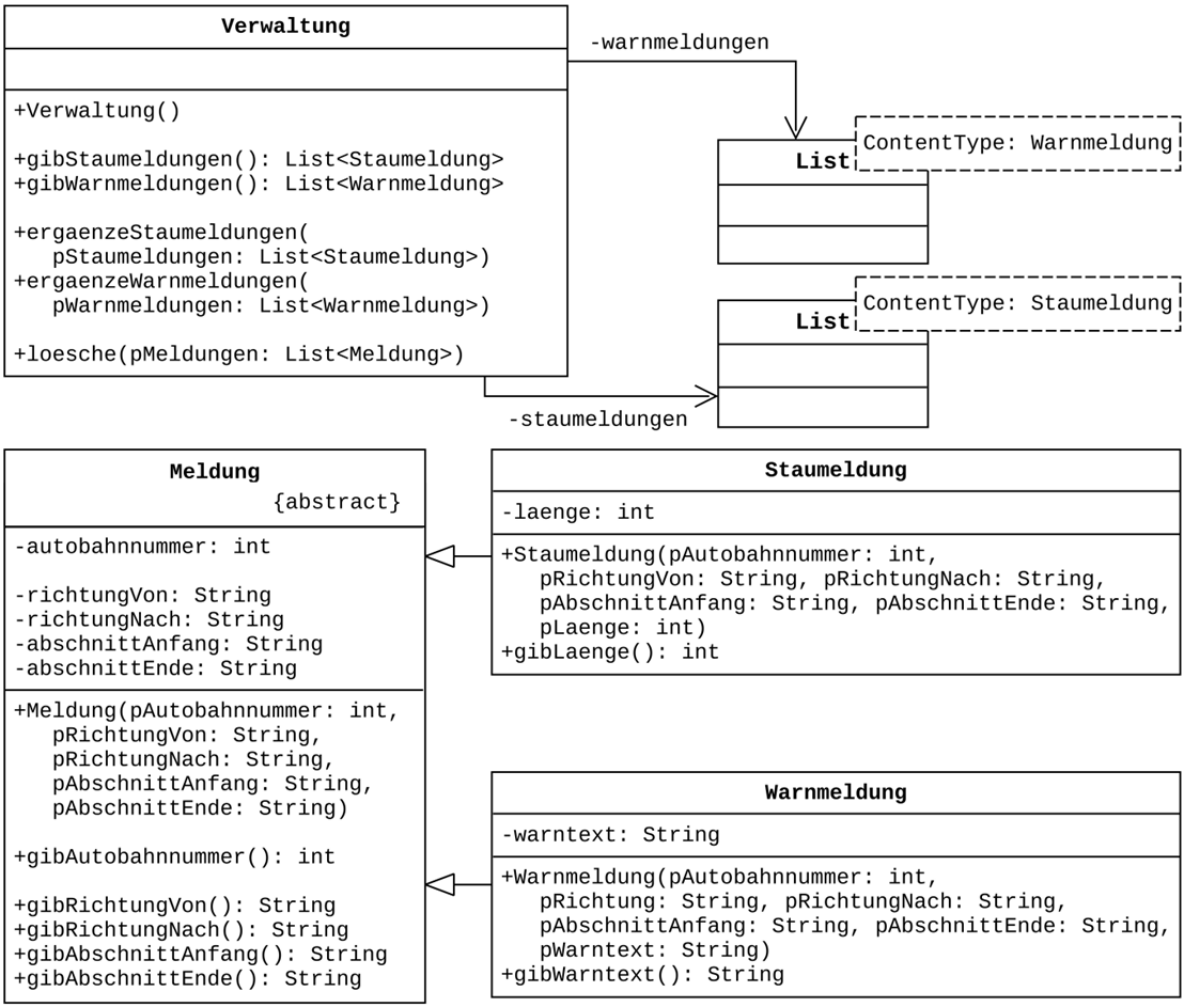
Die Methode `erstelleEtwas` bekommt im Parameter `pParameter` eine Ganzzahl übergeben und liefert eine Liste von Staumeldungen zurück (Zeile `1`).

Zunächst wird in Zeile `2` eine neue Liste für Staumeldungen mit dem Bezeichner `liste` erstellt. Anschließend wird die Liste `staumeldungen` Element für Element durchlaufen (Zeilen `3` bis `20`). Für das jeweils aktuelle Element wird geprüft, ob die Länge des entsprechenden Staus größer oder gleich dem Parameter `pParameter` ist (Zeile `6`). Wenn das der Fall ist, wird das Element in die Liste `liste` eingefügt (Zeile `7` bis Zeile `17`). Dazu wird die Liste `liste` in den Zeilen `7` bis `12` durchlaufen, bis sie am Ende ist oder das aktuelle Element der Liste `liste` eine größere Autobahnnummer aufweist als das einzufügende Element. Dort wird das Element dann eingefügt, d. h. aufsteigend nach der Autobahnnummer sortiert. Die Verzweigung in den Zeilen `13` bis `17` ist erforderlich, um ggf. am Ende der Liste anzuhängen. Die Methode liefert die Liste `liste` in Zeile `21` als Ergebnis zurück.

Im Sachzusammenhang erstellt die Methode eine Liste aller Staumeldungen, deren Länge größer oder gleich dem Wert des Parameters `pParameter` ist. Die Liste ist aufsteigend nach der Autobahnnummer sortiert und somit genau die Liste, die verlesen werden soll, wenn man Staus ab einer Länge von `pParameter` Kilometern berücksichtigen will.

Teilaufgabe d)

Die folgende Modellierung entspricht den erweiterten Anforderungen:



In der obigen Modellierung wird zwischen den Klassen `Staumeldung` und `Warnmeldung` unterschieden. Beide erben von der gemeinsamen abstrakten Oberklasse `Meldung`, welche die Autobahnnummer, die Richtung und den Abschnitt modelliert. In der Staumeldung wird eine Länge ergänzt und in der Warnmeldung ein Warntext.

Die Klasse `Verwaltung` verwaltet zwei Listen, einmal für Warnmeldungen und einmal für Staumeldungen. Elemente für beide Listen können mit den Methoden `ergaenzeStaumeldungen` bzw. `ergaenzeWarnmeldungen` hinzugefügt werden. Beide Listen können mit den Methoden `gibStaumeldungen` und `gibWarnmeldungen` angefragt werden. Zum Entfernen von Meldungen dient die Methode `loesche`. Mit ihr können sowohl Warnmeldungen wie auch Staumeldungen entfernt werden.

Teilaufgabe e)

Das folgende algorithmische Verfahren ist geeignet, den Verbesserungsvorschlag umzusetzen:

Die Liste aller Staumeldungen wird durchlaufen; dabei wird für jede aktuelle Staumeldung `akt` geprüft, ob für diesen Stau ein Folgestau existiert. Dazu muss die Liste aller Staumeldungen erneut durchlaufen werden. Dabei wird ein Stau gesucht, für den die Attributwerte von `autobahnnummer` und `richtungVon` bzw. `richtungNach` mit dem aktuellen Stau übereinstimmen und der Wert von `abschnittEnde` des aktuellen Staus `akt` mit dem Wert von `abschnittAnfang` des anderen Staus übereinstimmt.

Ist ein entsprechender Stau `tmp` gefunden, werden beide aus der Liste aller Staumeldungen entfernt. Stattdessen wird eine neue Staumeldung eingefügt. Diese neue Staumeldung bekommt die gemeinsame Autobahnnummer und Richtung beider Staus, eine Länge, die der Summe der Längen von `akt` und `tmp` entspricht und für `abschnittAnfang` den Wert von `akt` und für `abschnittEnde` den Wert von `tmp`.

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 an, welche Staumeldungen in welcher Reihenfolge verlesen werden.	2			
2	ermittelt, welche Mindeststaulänge angesetzt werden muss, wenn man möglichst viele, aber maximal drei Meldungen verlesen möchte.	3			
3	erläutert, dass es keine Mindeststaulänge gibt, die zu genau drei Meldungen führt.	3			
Sachlich richtige Lösungsalternative zur Modelllösung: (8) …… ……					
	Summe Teilaufgabe a)	**8**			

Teilaufgabe b)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	entwickelt ein algorithmisches Verfahren für die Methode.	5			
2	implementiert die Methode.	7			
Sachlich richtige Lösungsalternative zur Modelllösung: (12) …… ……					
	Summe Teilaufgabe b)	**12**			

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

Teilaufgabe c)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	analysiert die Methode und erläutert die Funktionsweise.	9			
2	erläutert, welche Information im Sachzusammenhang ermittelt wird.	3			
Sachlich richtige Lösungsalternative zur Modelllösung: (12) …… ……					
	Summe Teilaufgabe c)	**12**			

Teilaufgabe d)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	entwirft eine erweiterte Modellierung in Form eines Implementationsdiagramms.	7			
2	erläutert die erweiterte Modellierung.	5			
Sachlich richtige Lösungsalternative zur Modelllösung: (12) …… ……					
	Summe Teilaufgabe d)	**12**			

Teilaufgabe e)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	entwirft ein algorithmisches Verfahren, nach dem die Verbesserung umgesetzt werden kann.	6			
Sachlich richtige Lösungsalternative zur Modelllösung: (6) …… ……					
	Summe Teilaufgabe e)	**6**			

	Summe insgesamt	**50**			

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

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

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

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

Unterschrift, Datum:

Grundsätze für die Bewertung (Notenfindung)

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

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

Name: _____

Abiturprüfung 2019

Informatik, Grundkurs

Aufgabenstellung:

Bei Instant-Messaging-Systemen werden Kurznachrichten innerhalb kurzer Zeit zwischen zwei Geräten übertragen. Diese Systeme werden von vielen Menschen intensiv genutzt. Im Folgenden wird die Verwaltung der Benutzerinnen und Benutzer sowie das Zuordnen von Nachrichten simuliert.

In dem Instant-Messaging-System hat jede Benutzerin beziehungsweise jeder Benutzer ein eigenes Profil, das langfristig genutzt wird. In dem Profil werden eine zufällige, eindeutige, vierstellige Buchstabenfolge als ID sowie ein frei wählbarer Name und ein Passwort verwaltet. Außerdem werden in dem Profil alle bisher gesendeten und empfangenen Nachrichten gespeichert.

Im Arbeitsspeicher des Servers werden die Profile in einem binären Suchbaum verwaltet. Die Profile sind nach der eindeutigen ID alphabetisch im Suchbaum sortiert.

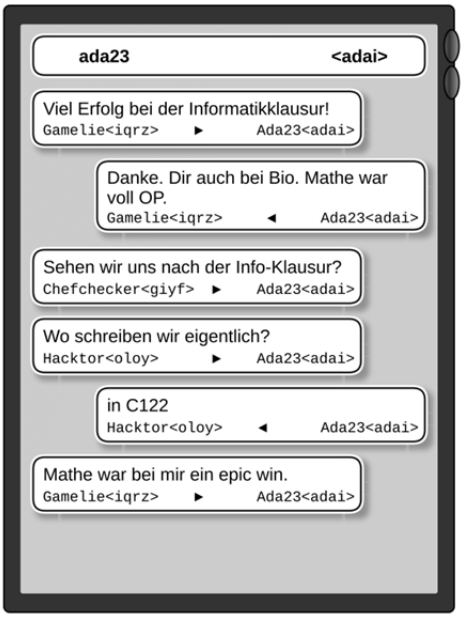


Abbildung 1:
Beispiel für gesendete und empfangene Nachrichten von `ada23` mit der ID `adai` in einem Instant-Messaging-System. Die IDs sind in spitzen Klammern <> dargestellt.

a) In einen leeren Suchbaum werden die folgenden Profile in der angegebenen Reihenfolge eingefügt. Zunächst sind die Profilnamen genannt, dann die IDs in spitzen Klammern <>:

```
Gamelie <iqrz>
Chefchecker <giyf>
ada23 <adai>
Gig <tgif>
Hacktor <oloy>
Fehler451 <asap>
```

Beschreiben Sie, wie ein Element in einen binären Suchbaum eingefügt wird und erläutern Sie dabei die Struktur des binären Suchbaums.

Stellen Sie den Suchbaum dar, der entsteht, wenn die Profile in der angegebenen Reihenfolge in einen leeren Suchbaum eingefügt werden.

Erläutern Sie für diese Verwaltung von Profilen den wesentlichen Vorteil, den ein Suchbaum gegenüber einer sortierten Liste hat.

(13 Punkte)

Name: _____

Im Folgenden ist eine Teilmodellierung in Form eines Implementationsdiagramms dargestellt:

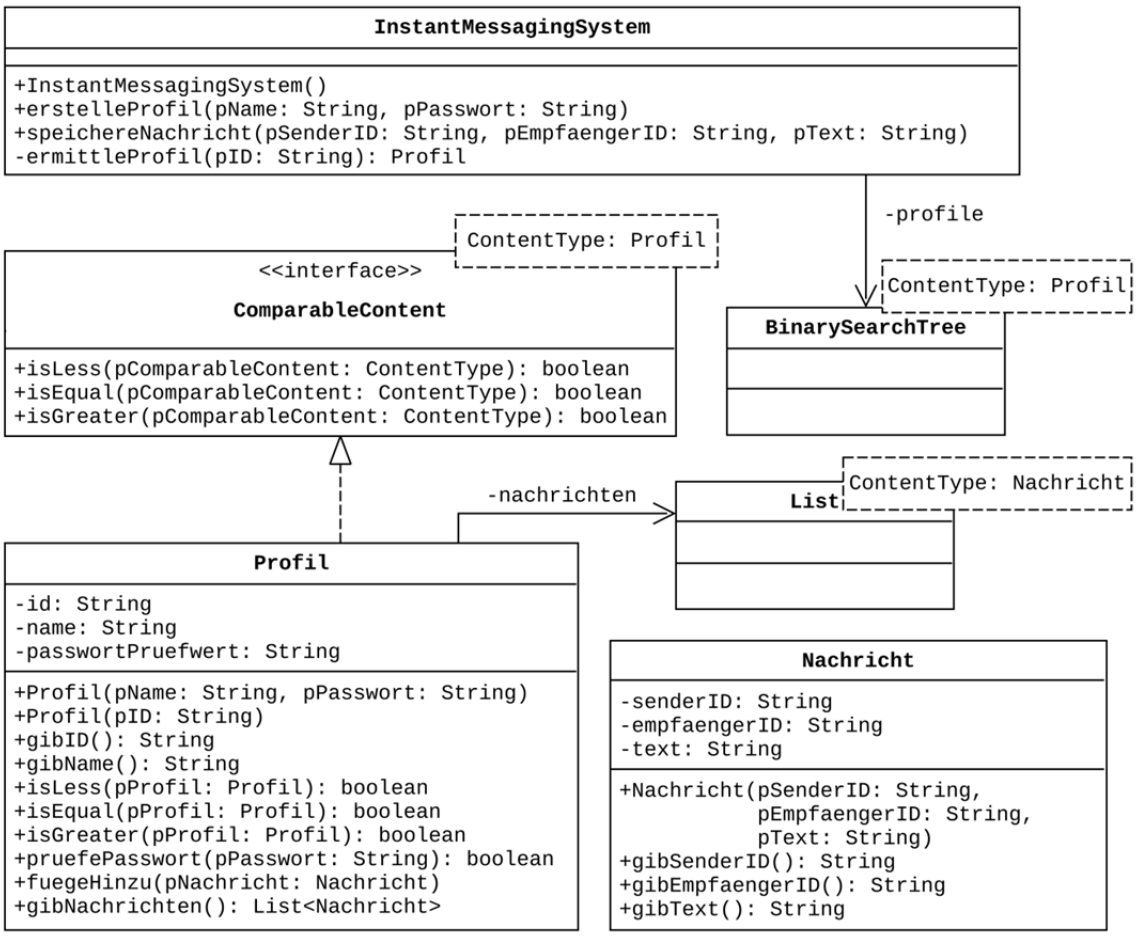


Abbildung 2: Teilmodellierung in Form eines Implementationsdiagramms

b) *Erläutern Sie die Beziehungen der Klassen* `Profil`, `InstantMessagingSystem` *und* `BinarySearchTree` *sowie des Interfaces* `ComparableContent`.

(8 Punkte)

Name: _____

Eine Nachricht besteht aus der ID des Senders, der ID des Empfängers und dem Nachrichtentext. Wird eine Nachricht dem System hinzugefügt, wird das Objekt vom Typ `Nachricht` sowohl im Profil des Senders als auch im Profil des Empfängers gespeichert, wenn zu beiden IDs ein Profil vorhanden ist. Andernfalls wird die Nachricht nicht gespeichert.

c) Die Methode `speichereNachricht` der Klasse `InstantMessagingSystem` speichert die Daten einer Nachricht wie beschrieben.

Sie hat den folgenden Methodenkopf:

```
public void speichereNachricht(String pSenderID,
                               String pEmpfaengerID,
                               String pText)
```

Implementieren Sie die Methode `speichereNachricht`.

Dokumentieren Sie die Methode durch geeignete Kommentare.

(8 Punkte)

d) Gegeben ist der folgende Suchbaum für das Attribut mit der Bezeichnung `profile` in der Klasse `InstantMessagingSystem`:

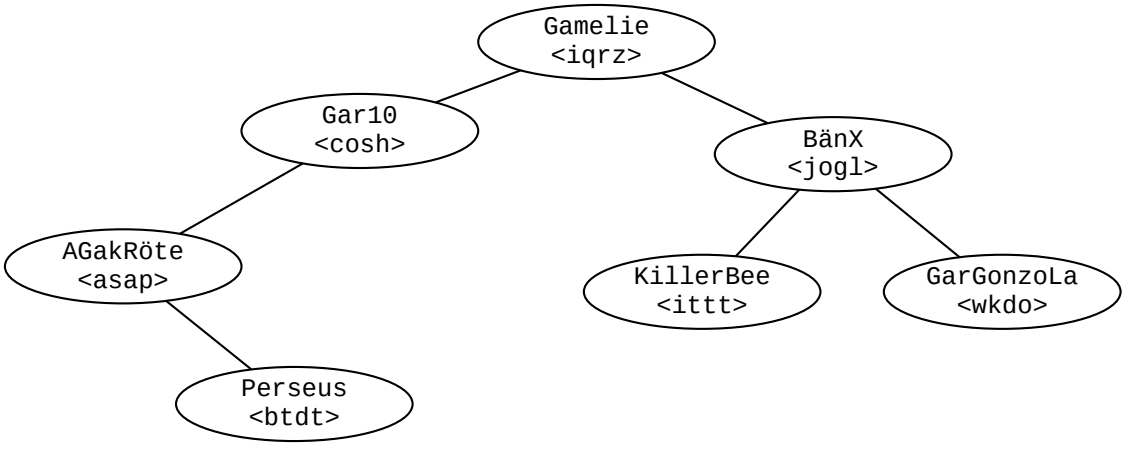


Abbildung 3: Darstellung der Beispieldaten des binären Suchbaums

Name: _____

Die folgenden Methoden der Klasse `InstantMessagingSystem` sind unkommentiert im Quellcode enthalten und nicht dokumentiert:

```
 1  public List<Profil> suche(String pP) {
 2    return suche(profile, pP);
 3  }
 4
 5  private List<Profil> suche(BinarySearchTree<Profil> pBaum,
 6                             String pP) {
 7    List<Profil> ergebnis = new List<Profil>();
 8    if (!pBaum.isEmpty()) {
 9      Profil aktuell = pBaum.getContent();
10      ergebnis.concat(suche(pBaum.getLeftTree(), pP));
11      if (aktuell.gibName().startsWith(pP)) {
12        ergebnis.append(aktuell);
13      }
14      ergebnis.concat(suche(pBaum.getRightTree(), pP));
15    }
16    return ergebnis;
17  }
```

Die öffentliche Methode `suche` soll mit dem Parameter `pP = "Ga"` aufgerufen werden.

Analysieren Sie die beiden angegebenen Methoden, indem Sie sie auf die Beispieldaten anwenden und angeben, in welcher Reihenfolge die Profile im Baum besucht werden.

Stellen Sie dar, wie die Rückgaben der Methoden für die Beispieldaten ermittelt werden.

Erläutern Sie im Sachkontext, welche Information die öffentliche Methode ermittelt.

(12 Punkte)

Name: _____

e) Die in Abbildung 1 dargestellten Nachrichten wirken sehr durcheinander, weil alle Nachrichten in der Reihenfolge angezeigt werden, in der sie eintrafen beziehungsweise gesendet wurden. In der Simulation des Instant-Messaging-Systems werden bisher alle Nachrichten, die zu einem Profil gehören, in einer Liste gespeichert.

Abbildung 4: Kommunikation von `ada23` mit zwei verschiedenen Kommunikationspartnern.

Die Speicherung der Nachrichten in einem Profil soll nun in ihrer Struktur verändert werden. Die Nachrichten sollen nach Gesprächspartnern gruppiert werden: Alle Nachrichten eines Profils, die mit einem bestimmten anderen Profil ausgetauscht wurden, sollen mittels einer geeigneten Datenstruktur gesammelt werden.

Modellieren Sie diese Änderung in Form eines Implementationsdiagramms.

Erläutern Sie den Ansatz für die Speicherung der Nachrichten in Ihrem Modell.

Hinweis: Stellen Sie nur die Änderung gegenüber dem Modell in Abbildung 2 dar.

(9 Punkte)

Zugelassene Hilfsmittel:

- GTR (grafikfähiger Taschenrechner) oder CAS (Computer-Algebra-System)
- Wörterbuch zur deutschen Rechtschreibung

Name: _____

Anhang

Dokumentationen der verwendeten Klassen

Die Klasse `String`

Ein Objekt der Klasse **`String`** repräsentiert eine Zeichenkette.

Auszug aus der Dokumentation der Klasse String

Anfrage **`boolean startsWith(String pPrefix)`**

Die Methode überprüft, ob der String mit dem als Parameter übergebenen Präfix `pPrefix` beginnt. Ist dies der Fall, wird `true` zurückgegeben, andernfalls wird `false` zurückgegeben.

Die Klasse `InstantMessagingSystem`

Ein Objekt der Klasse **`InstantMessagingSystem`** verwaltet die Profile der Benutzerinnen und Benutzer.

Auszug aus der Dokumentation der Klasse `InstantMessagingSystem`

Konstruktor **`InstantMessagingSystem()`**

Ein Objekt der Klasse wird initialisiert.

Auftrag **`void erstelleProfil(String pName, String pPasswort)`**

Ein neues Profil wird mit den übergebenen Parametern erzeugt und gespeichert.

Auftrag

```
void speichereNachricht(String pSenderID,
                        String pEmpfaengerID,
                        String pText)
```

Sofern die Parameter `pSenderID` und `pEmpfaengerID` zu einem existierenden Profil gehören, wird ein neues Nachrichten-Objekt mit den übergebenen Informationen erzeugt und in jedem der beiden Profile referenziert.

Dokumentation einer privaten Methode der Klasse `InstantMessagingSystem`:

Anfrage **`Profil ermittleProfil(String pID)`**

Sofern ein Profil mit der übergebenen ID gespeichert ist, wird das entsprechende Objekt zurückgegeben, anderenfalls wird `null` zurückgegeben.

Name: _____

Die Klasse Profil

Ein Objekt der Klasse **`Profil`** verwaltet Informationen über eine Benutzerin oder einen Benutzer sowie deren gesendeten und empfangenen Nachrichten. Des Weiteren können leere Profile erzeugt werden, die nur eine ID verwalten. Solche Suchprofile werden dazu verwendet, um eine Suche nach einem Objekt mit der gleichen ID zu ermöglichen.

Ausschnitt aus der Dokumentation der Klasse `Profil`

Konstruktor **`Profil(String pName, String pPasswort)`**
Ein neues Profil-Objekt mit den Parametern `pName` und `pPasswort` wird initialisiert. Dem Objekt wird automatisch eine freie ID zugeordnet.

Konstruktor **`Profil(String pID)`**
Mit diesem Konstruktor wird ein Such-Objekt erzeugt. Dieses Objekt dient ausschließlich dazu, nach dem Profil mit der übergebenen ID im Suchbaum zu suchen. Dazu wird die übergebene ID im Objekt gespeichert. Der Name und das Passwort des Profils sind leere Strings. Nachrichten können nicht gespeichert werden.

Anfrage **`String gibID()`**
Die Methode liefert die gespeicherte ID des Profils.

Anfrage **`String gibName()`**
Die Methode liefert den gespeicherten Namen des Profils.

Anfrage **`boolean isLess(Profil pProfil)`**
Die Methode liefert `true`, wenn die ID des übergebenen Profils lexikografisch kleiner als die gespeicherte ID ist. Ansonsten liefert sie `false`.

Anfrage **`boolean isEqual(Profil pProfil)`**
Die Methode liefert `true`, wenn die ID des übergebenen Profils identisch mit der gespeicherte ID ist. Ansonsten liefert sie `false`.

Anfrage **`boolean isGreater(Profil pProfil)`**
Die Methode liefert `true`, wenn die ID des übergebenen Profils lexikografisch größer als die gespeicherte ID ist. Ansonsten liefert sie `false`.

Anfrage **`boolean pruefePasswort(String pPasswort)`**
Sofern es sich nicht um ein Such-Objekt (s. o.) handelt, liefert die Methode `true`, wenn das übergebene Passwort korrekt ist. Ansonsten liefert sie `false`.

Auftrag **`void fuegeHinzu(Nachricht pNachricht)`**
Sofern es sich nicht um ein Such-Objekt (s. o.) handelt, speichert die Methode eine Referenz auf die übergebene Nachricht.

Auftrag **`List<Nachricht> gibNachrichten()`**
Die Methode liefert die Liste der gespeicherten Nachrichten.

Name: _____

Die Klasse Nachricht

Ein Objekt der Klasse **`Nachricht`** verwaltet die IDs des Senders und des Empfängers sowie den eigentlichen Nachrichtentext.

Ausschnitt aus der Dokumentation der Klasse `Nachricht`

Konstruktor **`Nachricht(pSenderID: String, pEmpfaengerID: String, pText: String)`**
Ein Objekt wird initialisiert. Die übergebenen Werte der Parameter werden gespeichert. Das Objekt ist als nicht gelesen markiert.

Anfrage **`String gibSenderID()`**
Die Methode liefert die ID des Senders.

Anfrage **`String gibEmpfaengerID()`**
Die Methode liefert die ID des Empfängers.

Anfrage **`String gibText()`**
Die Methode liefert den Text der Nachricht.

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 Klasse BinarySearchTree<ContentType extends ComparableContent<ContentType>>

Mithilfe der generischen Klasse **`BinarySearchTree`** können beliebig viele Objekte des Typs **`ContentType`** in einem Binärbaum (binärer Suchbaum) entsprechend einer Ordnungsrelation verwaltet werden.
Ein Objekt der Klasse **`BinarySearchTree`** stellt entweder einen leeren Baum dar oder verwaltet ein Inhaltsobjekt vom Typ **`ContentType`** sowie einen linken und einen rechten Teilbaum, die ebenfalls Objekte der Klasse **`BinarySearchTree`** sind.
Die Klasse der Objekte, die in dem Suchbaum verwaltet werden sollen, muss das generische Interface **`ComparableContent`** implementieren. Dabei muss durch Überschreiben der drei Vergleichsmethoden `isLess`, `isEqual`, `isGreater` (siehe Dokumentation des Interfaces) eine eindeutige Ordnungsrelation festgelegt sein.
Die Objekte der Klasse **`ContentType`** sind damit vollständig geordnet. Für je zwei Objekte c1 und c2 vom Typ **`ContentType`** gilt also insbesondere genau eine der drei Aussagen:

- `c1.isLess(c2)` (Sprechweise: c1 ist kleiner als c2)
- `c1.isEqual(c2)` (Sprechweise: c1 ist gleichgroß wie c2)
- `c1.isGreater(c2)` (Sprechweise: c1 ist größer als c2)

Alle Objekte im linken Teilbaum sind kleiner als das Inhaltsobjekt des Binärbaumes. Alle Objekte im rechten Teilbaum sind größer als das Inhaltsobjekt des Binärbaumes.
Diese Bedingung gilt auch in beiden Teilbäumen.

Dokumentation der generischen Klasse BinarySearchTree<ContentType extends ComparableContent<ContentType>>

Konstruktor **`BinarySearchTree()`**

Der Konstruktor erzeugt einen leeren Suchbaum.

Anfrage **`boolean isEmpty()`**

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

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

Falls bereits ein Objekt in dem Suchbaum vorhanden ist, das gleichgroß ist wie `pContent`, passiert nichts.
Andernfalls wird das Objekt `pContent` entsprechend der Ordnungsrelation in den Baum eingeordnet. Falls der Parameter `null` ist, ändert sich nichts.

Anfrage **`ContentType search(ContentType pContent)`**

Falls ein Objekt im binären Suchbaum enthalten ist, das gleichgroß ist wie `pContent`, liefert die Anfrage dieses, ansonsten wird `null` zurückgegeben.
Falls der Parameter `null` ist, wird `null` zurückgegeben.

Name: _____

Auftrag	**`void remove(ContentType pContent)`** Falls ein Objekt im binären Suchbaum enthalten ist, das gleichgroß ist wie `pContent`, wird dieses entfernt. Falls der Parameter `null` ist, ändert sich nichts.
Anfrage	**`ContentType getContent()`** Diese Anfrage liefert das Inhaltsobjekt des Suchbaumes. Wenn der Suchbaum leer ist, wird `null` zurückgegeben.
Anfrage	**`BinarySearchTree<ContentType> getLeftTree()`** Diese Anfrage liefert den linken Teilbaum des binären Suchbaumes. Der binäre Suchbaum ändert sich nicht. Wenn er leer ist, wird `null` zurückgegeben.
Anfrage	**`BinarySearchTree<ContentType> getRightTree()`** Diese Anfrage liefert den rechten Teilbaum des Suchbaumes. Der Suchbaum ändert sich nicht. Wenn er leer ist, wird `null` zurückgegeben.

Das generische Interface (Schnittstelle) ComparableContent<ContentType>

Das generische Interface **`ComparableContent`** muss von Klassen implementiert werden, deren Objekte in einen Suchbaum (**`BinarySearchTree`**) eingefügt werden sollen. Die Ordnungsrelation wird in diesen Klassen durch Überschreiben der drei implizit abstrakten Methoden `isGreater`, `isEqual` und `isLess` festgelegt.

Das Interface **`ComparableContent`** gibt folgende implizit abstrakte Methoden vor:

Anfrage	**`boolean isGreater(ContentType pComparableContent)`** Wenn festgestellt wird, dass das Objekt, von dem die Methode aufgerufen wird, bzgl. der gewünschten Ordnungsrelation größer als das Objekt `pComparableContent` ist, wird `true` geliefert. Sonst wird `false` geliefert.
Anfrage	**`boolean isEqual(ContentType pComparableContent)`** Wenn festgestellt wird, dass das Objekt, von dem die Methode aufgerufen wird, bzgl. der gewünschten Ordnungsrelation gleich dem Objekt `pComparableContent` ist, wird `true` geliefert. Sonst wird `false` geliefert.
Anfrage	**`boolean isLess(ContentType pComparableContent)`** Wenn festgestellt wird, dass das Objekt, von dem die Methode aufgerufen wird, bzgl. der gewünschten Ordnungsrelation kleiner als das Objekt `pComparableContent` ist, wird `true` geliefert. Sonst wird `false` geliefert.

Unterlagen für die Lehrkraft

Abiturprüfung 2019

Informatik, Grundkurs

1. Aufgabenart

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

2. Aufgabenstellung[1]

siehe Prüfungsaufgabe

3. Materialgrundlage

entfällt

4. Bezüge zum Kernlehrplan und zu den Vorgaben 2019

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
     - Entwurfsdigramme und Implementationsdiagramme
     - Lineare Strukturen
       Lineare Liste
     - Nicht-lineare Strukturen
       Binärer Suchbaum

   Algorithmen
   - Analyse, Entwurf und Implementierung von Algorithmen

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

2. *Medien/Materialien*
   - entfällt

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

5. Zugelassene Hilfsmittel

- GTR (grafikfähiger Taschenrechner) oder CAS (Computer-Algebra-System)
- Wörterbuch zur deutschen Rechtschreibung

6. Modelllösungen

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

Teilaufgabe a)

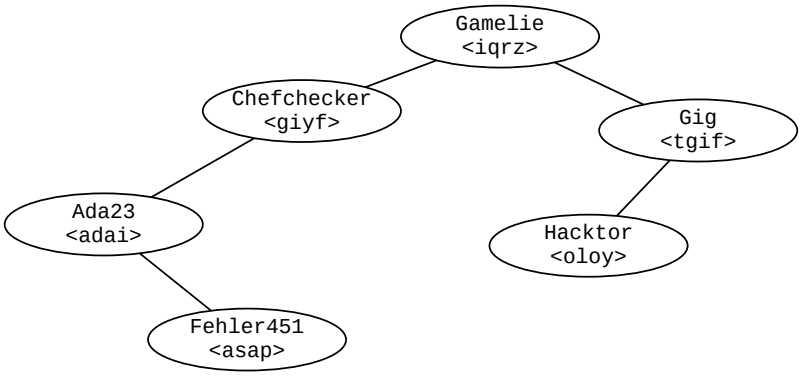
Ein binärer Suchbaum ist eine besondere Art eines Binärbaums. Die Elemente, die in dem Suchbaum gespeichert werden, werden in einer Ordnung abgespeichert, die einer Ordnungsrelation auf den Elementen folgt.

Ist der Binärbaum leer, wird das Element als Wurzelelement eingefügt. Andernfalls wird es zunächst mit dem Wurzelelement verglichen.

Ist es bezüglich des Sortierkriteriums kleiner als das Wurzelelement, so wird das Verfahren im linken Teilbaum fortgesetzt, da alle Elemente im linken Teilbaum kleiner als das aktuelle Element sind.

Ist es größer, so wird das Verfahren im rechten Teilbaum fortgesetzt, da im rechten Teilbaum alle Elemente größer als das aktuelle Element sind.

Ist es gleich, hängt die Entscheidung vom konkreten Entwurf ab, ob das Element überhaupt eingefügt wird oder das Verfahren im linken oder im rechten Teilbaum fortgeführt wird. Beispielsweise wird in der Klasse `BinarySearchTree` bei Gleichheit nicht eingefügt.



Die entstehende Ordnung eines binären Suchbaums ermöglicht in der Regel eine schnellere Suche nach einem Element als in einer vergleichbaren linearen Struktur – sofern der Suchbaum durch eine ungünstige Einfügekonstellation nicht sehr unausgeglichen aufgebaut ist.

In diesem Kontext werden häufig die Profile anhand ihres Identifikators gesucht, z. B., um eine Nachricht zu speichern. Deswegen ist es wichtig, dass das Suchen eines Profils eine geringe Laufzeit benötigt. Genau diesen Vorteil bietet ein Suchbaum gegenüber einer sortierten Liste.

Teilaufgabe b)

Ein Objekt der Klasse `InstantMessagingSystem` verwaltet mehrere Objekte der Klasse `Profil`. Für die Verwaltung dieser Objekte wird ein Objekt der Klasse `BinarySearchTree` mit dem Inhaltstyp `Profil` verwendet. Das Objekt hat den Bezeichner `profile`. Der Datentyp der in diesem Objekt der generischen Klasse `BinarySearchTree` verwalteten Objekte wird als `Profil` festgelegt.

Damit Objekte der Klasse `Profil` mit anderen Objekten dieser Klasse (oder einer Unterklasse) verglichen und somit in einem binären Suchbaum verwaltet werden können, muss die Klasse `Profil` die Schnittstelle `ComparableContent<Profil>` implementieren. Dadurch werden die Methoden `isLess`, `isEqual` und `isGreater` in Bezug auf `Profil` implementiert. Der generische Datentyp des Interface `ComparableContent` wird als Datentyp `Profil` festgelegt.

Teilaufgabe c)

```
public void speichereNachricht(String pSenderID,
                               String pEmpfaengerID,
                               String pText) {

  //Ermitteln der Profile anhand ihrer IDs
  Profil sender = ermittleProfil(pSenderID);
  Profil empfaenger = ermittleProfil(pEmpfaengerID);

  //Die Nachricht wird nur genau dann gespeichert,
  //wenn beide Profile existieren.
  if (sender != null && empfaenger != null) {

    //Erzeugen eines Nachrichten-Objekts
    Nachricht neu = new Nachricht(pSenderID, pEmpfaengerID,
                                  pText);

    //Hinzufügen in beiden Profilen
    sender.fuegeHinzu(neu);
    empfaenger.fuegeHinzu(neu);
  }
}
```

Teilaufgabe d)

Es werden alle Knoten des Baums in In-Order-Reihenfolge durchlaufen, also in der folgenden Reihenfolge: `"AGakRöte"`, `"Perseus"`, `"Gar10"`, `"Gamelie"`, `"KillerBee"`, `"BänX"`, `"GarGonzoLa"`.

Die öffentliche Methode ruft die private Methode `suche` mit der Zeichenkette `"Ga"` und dem gesamten Suchbaum als Parameter auf. Sie gibt das Ergebnis der privaten Methode zurück.

In der privaten Methode wird zunächst wird im linken Teilbaum gesucht. Ist die Suche dort abgeschlossen, wird das aktuelle Element betrachtet, danach wird im rechten Teilbaum weitergesucht.

Der Profilname `"AGakRöte"` beginnt nicht mit `"Ga"`, deswegen wird dort, wie beim Profil mit Namen `"Perseus"`, eine leere Liste zurückgegeben. Da der Profilname `"Gar10"` mit `"Ga"` beginnt, wird das Profil in eine Ergebnisliste aufgenommen. Der rechte Teilbaum vom Profil mit Namen `"Gar10"`, der linke vom Profil mit Namen `"AGakRöte"` und der linke und rechte Teilbaum vom Profil mit Namen `"Perseus"` sind leer, dort wird also jeweils eine leere Liste zurückgegeben.

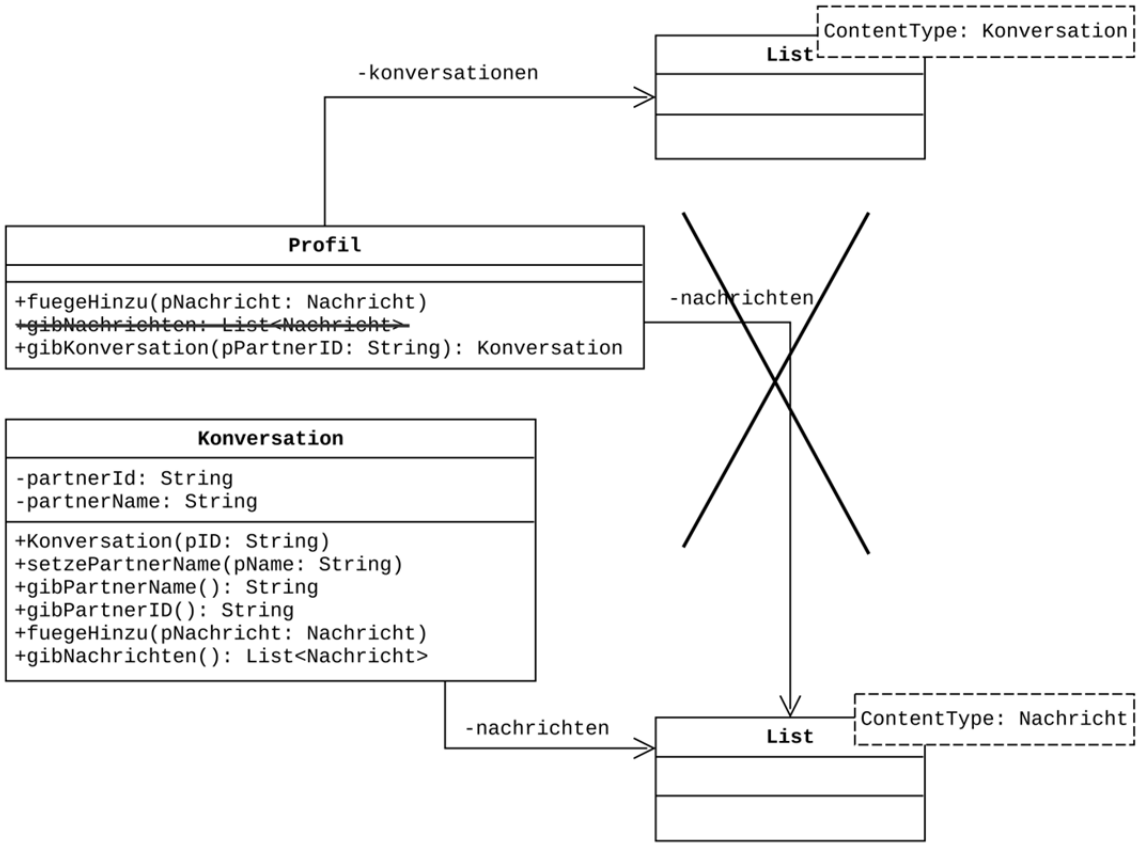
Die Rückgabe des Rekursionsschritts mit `"Gar10"` als Wurzel liefert eine nicht leere Liste an den ersten Rekursionsaufruf zurück. Diese Liste wird mit der Ergebnisliste konkateniert und enthält also zunächst das Profil mit Namen `"Gar10"`, dann das Profil mit Namen `"Gamelie"`.

Nun wird der rechte Teilbaum vom Profil mit Namen `"Gamelie"` durchlaufen, wo auf die gleiche Art nur das Profil mit Namen `"GarGonzoLa"` ermittelt wird.

Es ergibt sich also als Rückgabe eine Liste, die die drei Profile mit den Namen `"Gar10"`, `"Gamelie"` und `"GarGonzoLa"` in dieser Reihenfolge enthält.

Die Methode ermittelt alle Profile, deren Name mit dem Wert des Parameters `pP` beginnt.

Teilaufgabe e)



Sämtliche Nachrichten, die mit einem Kommunikationspartner ausgetauscht werden, werden in einem Objekt der Klasse `Konversation` gesammelt. Der Kommunikationspartner ist bei jeder Nachricht anhand der ID zu identifizieren. In einem Profil werden nicht die Nachrichten selbst, sondern die Konversationen in einer Liste verwaltet. Wird bei einem Profil eine neue Nachricht hinzugefügt, so wird diese an die entsprechende Konversation weitergereicht. Existiert noch keine Konversation, wird eine neue Konversation mit der in der Nachricht enthaltenen ID und gegebenenfalls dem Namen des Partners hinzugefügt.

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	beschreibt, wie ein Element in einen binären Suchbaum eingefügt wird und erläutert dabei die Struktur des binären Suchbaums.	6			
2	stellt den Suchbaum dar, der entsteht, wenn die Profile in der angegebenen Reihenfolge in einen leeren Suchbaum eingefügt werden.	3			
3	erläutert für diese Verwaltung von Profilen den wesentlichen Vorteil, den ein Suchbaum gegenüber einer sortierten Liste hat.	4			
Sachlich richtige Lösungsalternative zur Modelllösung: (13) …… ……					
	Summe Teilaufgabe a)	**13**			

Teilaufgabe b)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	erläutert die Beziehungen der Klassen `Profil`, `InstantMessagingSystem` und `BinarySearchTree` sowie des Interfaces `ComparableContent`.	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	implementiert die Methode.	6			
2	dokumentiert die Methode durch geeignete Kommentare.	2			
Sachlich richtige Lösungsalternative zur Modelllösung: (8) …… ……					
	Summe Teilaufgabe c)	**8**			

Teilaufgabe d)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	wendet die Methoden auf die Beispieldaten an und gibt an, in welcher Reihenfolge die Profile im Baum besucht werden.	7			
2	stellt dar, wie die Rückgaben der Methoden für die Beispieldaten ermittelt werden.	3			
3	erläutert im Sachkontext, welche Information die öffentliche Methode ermittelt.	2			
Sachlich richtige Lösungsalternative zur Modelllösung: (12) …… ……					
	Summe Teilaufgabe d)	**12**			

Teilaufgabe e)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	modelliert die Änderung in Form eines Implementationsdiagramms.	6			
2	erläutert den Ansatz für die Speicherung der Nachrichten in dem entwickelten Modell.	3			
Sachlich richtige Lösungsalternative zur Modelllösung: (9) …… ……					
	Summe Teilaufgabe e)	**9**			

	Summe insgesamt	**50**			

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

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

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

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

Unterschrift, Datum:

Grundsätze für die Bewertung (Notenfindung)

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

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

Name: _____

Abiturprüfung 2019

Informatik, Grundkurs

Aufgabenstellung:

Das Unternehmen *Rent a Bike in a City* verleiht in einer Stadt an mehreren Verleihstationen Fahrräder an Kundinnen und Kunden.

Eine Kundin oder ein Kunde kann an einer Fahrradstation ein Fahrrad ausleihen und an einer beliebigen Fahrradstation zurückgeben.

Wenn ein Kunde oder eine Kundin ein Fahrrad ausleiht, dann wird der Ausleihzeitpunkt erfasst. Bei Rückgabe werden der Zeitpunkt und die gefahrenen Kilometer protokolliert. Ein Fahrrad ist nur an der Station, an der es steht, ausleihbar. Ein verliehenes Fahrrad ist somit nicht ausleihbar.

Zur Verwaltung der Daten möchte das Unternehmen eine relationale Datenbank aufbauen, die der folgenden Teilmodellierung entspricht:

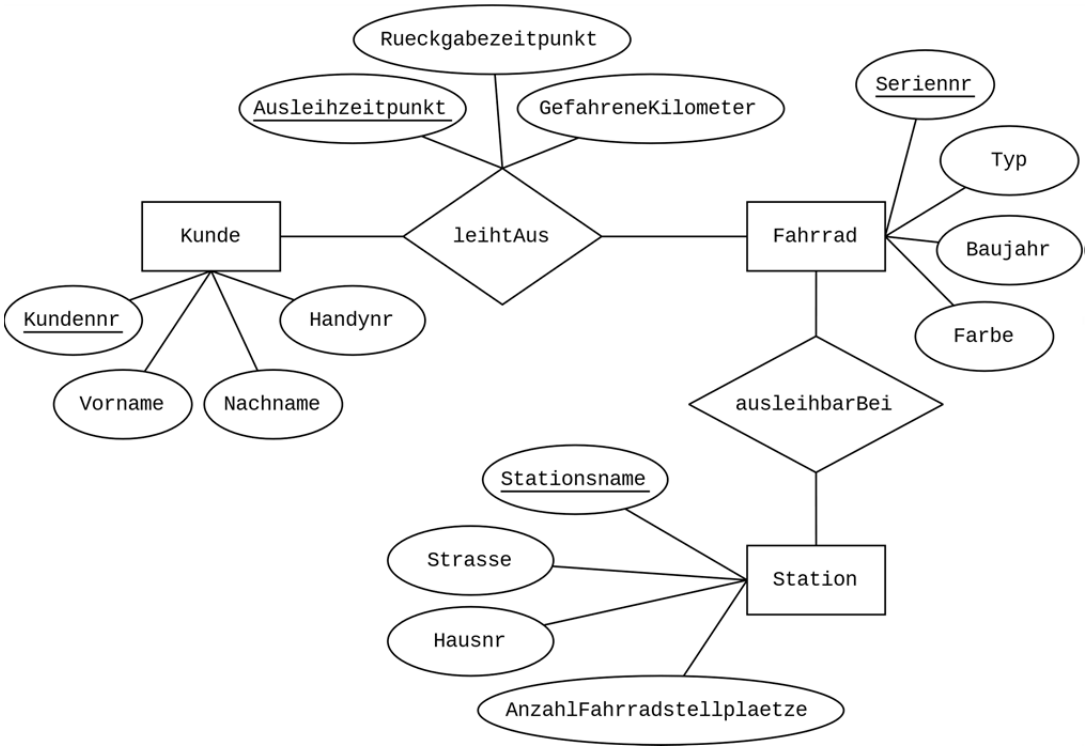


Abbildung 1: Teilmodellierung der Datenbank für *Rent a Bike in a City*

Name: _____

Im Folgenden soll mit dem in Abbildung 2 gegebenen Datenbankschema gearbeitet werden. Es stellt einen Ausschnitt der Gesamtdatenbank dar. Beispieldaten zu diesem Datenbankschema sind in der Anlage zu finden.

```
Kunde(Kundennr, Vorname, Nachname, Handynr)
leihtAus(↑Kundennr, ↑Seriennr, Ausleihzeitpunkt,
         Rueckgabezeitpunkt, GefahreneKilometer)
Fahrrad(Seriennr, Typ, Baujahr, Farbe, ↑Stationsname)
Station(Stationsname, Strasse, Hausnr, AnzahlFahrradstellplaetze)
```

Abbildung 2: Datenbankschema eines Ausschnittes der Datenbank

a) *Beschreiben Sie die in Abbildung 1 gegebene Teilmodellierung.*

Ermitteln Sie die Kardinalitäten für das Entity-Relationship-Modell, so dass diese der Umsetzung im Datenbankschema entsprechen.

Begründen Sie anhand des Kontextes, warum die Attribute `Kundennr` *und* `Seriennr` *gemeinsam mit dem Attribut* `Ausleihzeitpunkt` *in der Relation* `leihtAus` *den kombinierten Primärschlüssel bilden.*

(12 Punkte)

b) Aus der Datenbank sollen folgende Informationen abgefragt werden:

i. Gesucht sind die Seriennummern – aufsteigend sortiert – aller roten Fahrräder im Bestand, die vor dem Jahr 2017 gebaut wurden.

ii. Gesucht sind die Vornamen, Nachnamen und Handynummern der Kunden, die ein Fahrrad ausgeliehen hatten und damit beim Ausleihvorgang mehr als 10 Kilometer gefahren sind.

iii. Gesucht sind die Fahrradtypen und die jeweilige Anzahl, wie oft jeder Fahrradtyp ausgeliehen wurde, sofern sie bereits mindestens einmal ausgeliehen waren.

Entwerfen Sie für die obigen Anfragen i, ii und iii jeweils eine SQL-Anweisung.

(10 Punkte)

Name: _____

c) Folgende SQL-Anweisung ist gegeben:

```
 1  SELECT *
 2  FROM (
 3    SELECT Fahrrad.Seriennr, Fahrrad.Stationsname,
 4      Fahrrad.Baujahr,
 5      SUM(leihtAus.GefahreneKilometer) AS X,
 6      COUNT(*) AS Y
 7    FROM Fahrrad
 8    INNER JOIN leihtAus
 9      ON Fahrrad.Seriennr = leihtAus.Seriennr
10    GROUP BY Fahrrad.Seriennr
11  ) AS Tmp
12  WHERE Tmp.X > 500 OR Tmp.Y > 50
13  ORDER BY Tmp.X DESC
```

Analysieren und erläutern Sie die Unterabfrage (Zeilen 3 – 10).

Analysieren und erläutern Sie die gesamte SQL-Anweisung.

Erläutern Sie im Sachzusammenhang, welche Information die gesamte SQL-Anweisung ermittelt.

(8 Punkte)

d) Damit die Kunden bei einer Fahrradstation schnell das passende Fahrrad finden, sind alle Kinderfahrräder grün, alle Damenfahrräder rot und alle Herrenfahrräder blau.

Das in Abbildung 2 gegebene Datenbankschema befindet sich bereits in der ersten Normalform.

Überführen Sie das in Abbildung 2 gegebene Datenbankschema in ein neues Datenbankschema, das der dritten Normalform entspricht.

Erläutern Sie die Änderungen, die zur Überführung in die dritte Normalform notwendig sind.

Beurteilen Sie, ob Ihr entwickeltes Datenbankschema noch geeignet ist, wenn z. B. durch einen Zukauf noch mehrere gelbe Damen- und Herrenfahrräder hinzukämen.

(10 Punkte)

Name: _____

e) Die Datenbank soll um folgende Aspekte erweitert werden:
- Die standardisierten Wartungsvorgänge müssen für jedes Fahrrad protokolliert werden. Dazu sollen die Informationen über den Zeitpunkt, die durchgeführte Maßnahme und die Person, die die Wartung durchgeführt hat, verwaltet werden.
- Um die Fahrräder flächendeckender auszulasten, sollen Rabattaktionen eingeführt werden. Für eine Rabattaktion muss der Name der Aktion, eine Beschreibung und der prozentuale Rabatt verwaltet werden. Jedes Fahrrad kann höchstens an einer Rabattaktion teilnehmen. Es muss erfasst werden, in welchem Zeitraum ein bestimmtes Fahrrad an einer Rabattaktion teilnimmt.

Modellieren Sie die Erweiterung für das Entity-Relationship-Diagramm aus Abbildung 1 so, dass der neue Datenbankentwurf zusätzlich die beschriebenen Aspekte enthält.

Erläutern Sie, wie Ihr entwickeltes Modell die Anforderungen umsetzt.

Hinweis: Entitäts- und Beziehungstypen, die für die Erweiterung nicht relevant sind, müssen nicht dargestellt werden.

(10 Punkte)

Zugelassene Hilfsmittel:

- GTR (grafikfähiger Taschenrechner) oder CAS (Computer-Algebra-System)
- Wörterbuch zur deutschen Rechtschreibung

Name: _____

Anlage

Beispieldaten zur Teilmodellierung aus Abbildung 2

Fahrrad

Seriennr	Typ	Baujahr	Farbe	Stationsname
1	Kind	2018	grün	Bahnhof
2	Kind	2017	grün	See
3	Herren	2017	blau	*NULL*
4	Damen	2015	rot	Schule
5	Damen	2018	rot	Schwimmbad
6	Herren	2018	blau	Schwimmbad
7	Damen	2015	rot	See

Kunde

Kundennr	Vorname	Nachname	Handynr
1	Ada	Lovelace	016012345678
2	John	von	016398765432
3	Alan	Turing	016001010101
4	Konrad	Zuse	016310101010

leihtAus

Kundennr	Seriennr	Ausleihzeitpunkt	Rueckgabezeitpunkt
1	4	2016-06-16 15:00:00	2016-06-16 18:00:00
1	5	2017-04-13 10:30:00	2017-04-13 19:45:00
2	3	2018-12-24 08:00:00	2018-12-25 01:00:00
3	3	2019-04-16 12:08:00	*NULL*

GefahreneKilometer
7
12
20
NULL

Name: _____

Station			
Stationsname	**Strasse**	**Hausnr**	**AnzahlFahrradstellplaetze**
Bahnhof	Bahnhofsplatz	1	100
Park	Im Park	5	100
Schule	Schulstr.	3	40
Schwimmbad	Schwimmbadweg	1	50
See	Paradieser Weg	64	30

Unterlagen für die Lehrkraft

Abiturprüfung 2019

Informatik, Grundkurs

1. Aufgabenart

Analyse, Modellierung und Abfrage relationaler Datenbanken

2. Aufgabenstellung[1]

siehe Prüfungsaufgabe

3. Materialgrundlage

entfällt

4. Bezüge zum Kernlehrplan und zu den Vorgaben 2019

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

- GTR (grafikfähiger Taschenrechner) oder CAS (Computer-Algebra-System)
- 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 `Kunde` wird als Primärschlüssel `Kundennr` verwendet.
Im Entitätstyp `Fahrrad` wird als Primärschlüssel `Seriennr` verwendet.
Im Entitätstyp `Station` wird als Primärschlüssel `Stationsname` verwendet.

Vom Entitätstyp `Kunde` werden die weiteren Attribute `Vorname`, `Nachname` und `Handynr` verwaltet.
Vom Entitätstyp `Fahrrad` werden die weiteren Attribute `Typ`, `Baujahr` und `Farbe` verwaltet.
Vom Entitätstyp `Station` werden die weiteren Attribute `Strasse`, `Hausnr` und `AnzahlFahrradstellplaetze` verwaltet.

Der Beziehungstyp `leihtAus` modelliert, welche Kundin oder welcher Kunde welches Fahrrad ausgeliehen hat. Zusätzlich verwaltet der Beziehungstyp die weiteren Attribute `Ausleihzeitpunkt`, `Rueckgabezeitpunkt` und `GefahreneKilometer`.

Der Beziehungstyp `ausleihbarBei` modelliert, an welcher Station ein zur Verfügung stehendes Fahrrad steht.

Kardinalitäten beim Beziehungstyp `leihtAus`:
Es handelt sich um einen `n:m`-Beziehungstyp. Ein Kunde kann mehrere Fahrräder ausleihen und ein Fahrrad kann von mehreren Kunden zu unterschiedlichen Zeitpunkten ausgeliehen werden.

Kardinalitäten beim Beziehungstyp `ausleihbarBei`:
Es handelt sich um einen `1:n`-Beziehungstyp. An einer Station können mehrere Fahrräder stehen. Ein Fahrrad steht an einer Station.
Im Datenbankschema wird diese Beziehung durch das Fremdschlüsselattribut `Stationsname` im Entitätstyp `Fahrrad` realisiert.

Der Primärschlüssel für den Beziehungstyp `leihtAus` setzt sich aus der `Kundennr` (Fremdschlüssel von `Kunde`) und der `Seriennr` (Fremdschlüssel von `Fahrrad`) und dem `Ausleihzeitpunkt` zusammen.

Weder die Fremdschlüssel noch der `Ausleihzeitpunkt` genügen, um einen Ausleihvorgang eindeutig bestimmen zu können, da zu einem Zeitpunkt mehrere Kunden unterschiedliche Fahrräder ausleihen könnten und ein Kunde mehrfach ein identisches Fahrrad ausleihen kann.

Teilaufgabe b)

Die folgenden SQL-Anweisungen realisieren die Anfragen:

(i)
```
SELECT Seriennr
FROM Fahrrad
WHERE Farbe = 'rot' AND Baujahr < 2017
ORDER BY Seriennr ASC
```

(ii)
```
SELECT Kunde.Vorname, Kunde.Nachname, Kunde.Handynr
FROM Kunde
  INNER JOIN leihtAus
    ON Kunde.Kundennr = leihtAus.Kundennr
WHERE leihtAus.GefahreneKilometer > 10
```

(iii)
```
SELECT Fahrrad.Typ, COUNT(*)
FROM Fahrrad
  INNER JOIN leihtAus
    ON Fahrrad.Seriennr = leihtAus.Seriennr
GROUP BY Fahrrad.Typ
```

Teilaufgabe c)

Die Unterabfrage beinhaltet einen INNER JOIN von der Tabelle `Fahrrad` auf `leihtAus` über die `Seriennr` (vgl. Zeilen 8 – 9). Somit tauchen alle Fahrräder in der Teilergebnismenge auf, die schon einmal ausgeliehen wurden. In Zeile 10 werden diese Ergebnisse nach der Seriennummer des Fahrrads gruppiert.

Die Unterabfrage ermittelt folgende Informationen (vgl. Zeilen 3 – 6):
- Seriennummer des Fahrrads
- Stationsname, wo das Fahrrad steht
- Baujahr des Fahrrads
- Kilometersumme (Summe der gefahrenen Kilometer des Fahrrads)
- Anzahl der Verleihvorgänge (Anzahl, wie oft das Fahrrad verliehen wurde)

Die gesamte SQL-Anweisung beinhaltet eine Unterabfrage (vgl. Zeilen 3 – 10), die die als `Tmp` bezeichnete Ergebnistabelle zurückliefert.

Aus der Ergebnistabelle der Unterabfrage werden alle Spalten ausgegeben (vgl. Zeile 1) und die Datensätze mit einer Kilometersumme größer 500 Kilometer oder mit mehr als 50 Verleihvorgängen. Die Ergebnismenge ist absteigend nach der Kilometersumme sortiert (vgl. Zeile 13).

Im Sachzusammenhang umfasst die Ergebnismenge die Datensätze von den Fahrrädern, die mehr als 500 Kilometer gefahren oder häufiger als 50-mal verliehen wurden (vgl. Zeile 12).

Teilaufgabe d)

Kunde(Kundennr, Vorname, Nachname, Handynr)

leihtAus(↑Kundennr, ↑Seriennr, Ausleihzeitpunkt,
Rueckgabezeitpunkt, GefahreneKilometer)

Fahrrad(Seriennr, ↑Typname, Baujahr, ↑Stationsname)

Typ(Typname, Farbe)

Station(Stationsname, Strasse, Hausnr, AnzahlFahrradstellplaetze)

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.

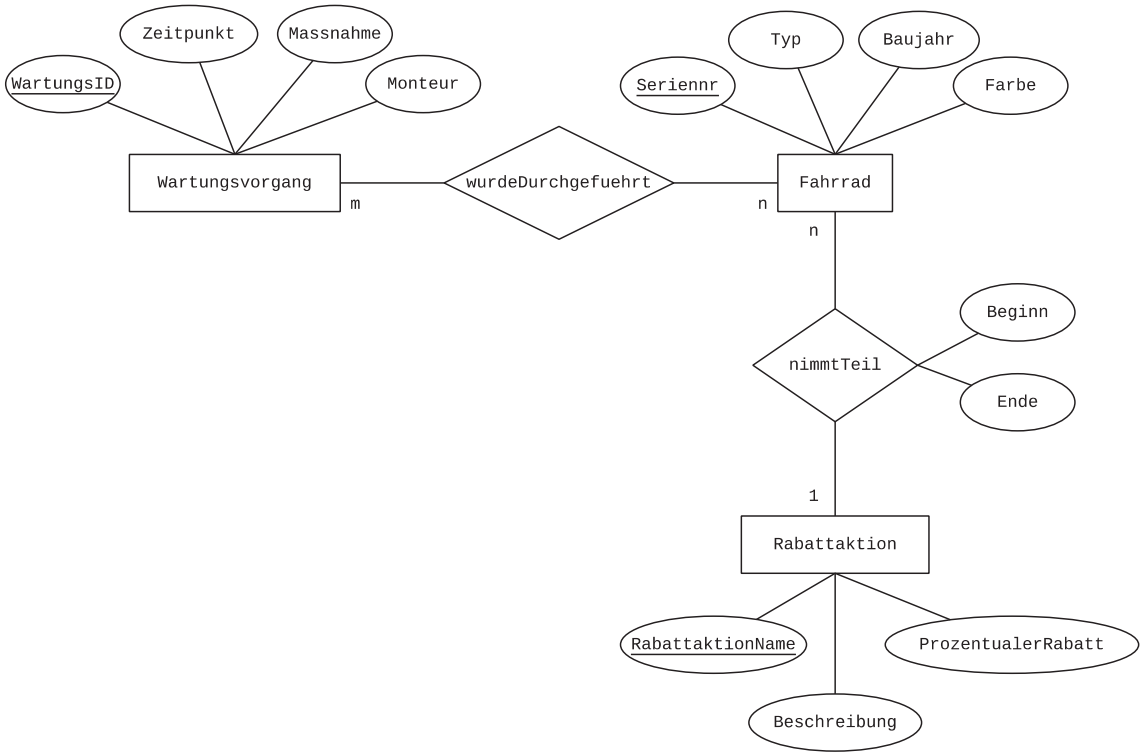
Ein Verstoß gegen die zweite Normalform könnte nur in einer Relation mit einem kombinierten Primärschlüssel auftreten. Da jedes Nichtschlüsselattribut der Relation `leihtAus` vom gesamten Schlüssel abhängt, ist das Datenbankschema auch in der zweiten Normalform.

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.

Der Entitätstyp `Fahrrad` verstößt gegen die dritte Normalform, weil das Attribut `Farbe` funktional vom Attribut `Typ` abhängt. Dieser Verstoß wird gelöst, indem der neue Entitätstyp `Typ` mit dem Primärschlüssel `Typname` und dem Attribut `Farbe` eingerichtet wird. Im Entitätstyp `Fahrrad` werden die Attribute `Typ` und `Farbe` durch das Fremdschlüsselattribut `Typname` ersetzt.

Wenn durch einen Zukauf noch mehrere gelbe Damen- und Herrenfahrräder hinzukämen, wäre das entwickelte Datenbankschema wenig geeignet, weil im Entitätstyp `Typ` das Attribut `Farbe` nicht mehr vom Primärschlüssel `Typname` abhängen würde.

Teilaufgabe e)



Bei einem Fahrrad können `m` Wartungsvorgänge und ein Wartungsvorgang kann an `n` Fahrrädern durchgeführt werden. Die geforderten Informationen werden in Attributen beim Entitätstyp `Wartungsvorgang` verwaltet.

Ein Fahrrad nimmt an höchstens einer Rabattaktion teil und an einer Rabattaktion nehmen `n` Fahrräder teil. Beim Beziehungstyp `nimmtTeil` wird der Zeitraum, in dem das Fahrrad an der entsprechenden Aktion teilnimmt, durch die Attribute `Beginn` und `Ende` abgebildet.
Die restlichen geforderten Informationen werden als Attribute im Entitätstyp `Rabattaktion` verwaltet.

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	beschreibt die gegebene Teilmodellierung.	4			
2	ermittelt die Kardinalitäten für das Entity-Relationship-Modell, so dass diese der Umsetzung im Datenbankschema entsprechen.	4			
3	begründet anhand des Kontextes, warum die Attribute `Kundennr` und `Seriennr` gemeinsam mit dem Attribut `Ausleihzeitpunkt` in der Relation `leihtAus` den kombinierten Primärschlüssel bilden.	4			
Sachlich richtige Lösungsalternative zur Modelllösung: (12) …… ……					
	Summe Teilaufgabe a)	**12**			

Teilaufgabe b)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	entwirft für die erste Anfrage `i` eine SQL-Anweisung.	3			
2	entwirft für die zweite Anfrage `ii` eine SQL-Anweisung.	3			
3	entwirft für die dritte Anfrage `iii` eine SQL-Anweisung.	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 und erläutert die innere SQL-Anweisung.	4			
2	analysiert und erläutert die gesamte SQL-Anweisung.	2			
3	erläutert im Sachzusammenhang, welche Information die gesamte SQL-Anweisung ermittelt.	2			
Sachlich richtige Lösungsalternative zur Modelllösung: (8) …… ……					
	Summe Teilaufgabe c)	**8**			

Teilaufgabe d)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	überführt das gegebene Datenbankschema in ein neues Datenbankschema, das der dritten Normalform entspricht.	4			
2	erläutert die Änderungen, die zur Überführung in die dritte Normalform notwendig sind.	3			
3	beurteilt, ob das entwickelte Datenbankschema noch geeignet ist, wenn z. B. durch einen Zukauf noch mehrere gelbe Damen- und Herrenfahrräder hinzukämen.	3			
Sachlich richtige Lösungsalternative zur Modelllösung: (10) …… ……					
	Summe Teilaufgabe d)	**10**			

Teilaufgabe e)

	Anforderungen	Lösungsqualität			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	modelliert die Erweiterung für das Entity-Relationship-Diagramm so, dass der neue Datenbankentwurf zusätzlich die beschriebenen Aspekte enthält.	6			
2	erläutert, wie sein entwickeltes Modell die Anforderungen umsetzt.	4			
Sachlich richtige Lösungsalternative zur Modelllösung: (10) …… ……					
	Summe Teilaufgabe e)	**10**			

	Summe insgesamt	**50**			

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

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

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

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

Unterschrift, Datum:

Grundsätze für die Bewertung (Notenfindung)

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

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

Name: _____

Abiturprüfung 2019

Informatik, Grundkurs

Aufgabenstellung:

Eine Vektorgrafik ist eine Computergrafik, bei der das gesamte Bild nicht punktweise gespeichert wird, sondern alle in der Grafik enthaltenen Objekte anhand ihrer Eigenschaften (z. B. Farbe, Lage im Koordinatensystem, Länge, Abstände) exakt beschrieben werden. In dieser Aufgabe sollen nur Vektorgrafiken betrachtet werden, die aus Liniensegmenten zusammengesetzt sind.

Bevor Linien gezeichnet werden, wird der Zeichenstift mit dem „move“-Befehl zu einem Punkt des Zeichenbereichs bewegt, an dem die Zeichnung beginnt (Startpunkt). Ein „move“-Befehl beginnt mit dem Symbol `m`, gefolgt von der x- und der y-Koordinate eines Punktes.

Punktkoordinaten sind ein- oder zweistellige ganze Zahlen ohne führende Nullen. Der Ursprung `(0|0)` liegt in der Mitte des Bildschirms. Falls eine Koordinatenangabe negativ ist, beginnt sie mit dem Minuszeichen; positive Koordinatenangaben haben kein Vorzeichen. Die x- und die y-Koordinate werden durch das Trennzeichen `/` getrennt.

Beispiel: Der Befehl `m5/-40` bedeutet, dass der Zeichenstift auf den Punkt mit den Koordinaten `(5|-40)` gesetzt wird.

Der nichtdeterministische endliche Automat (NEA) `A` zum Übergangsgraphen in Abbildung 1 prüft, ob ein „move“-Befehl regelkonform aufgebaut ist.

Ziffern zwischen `1` und `9` werden im Automaten unter dem Eingabezeichen `z` abgearbeitet: Der Übergang $q_1 \xrightarrow{z} q_3$ im Übergangsgraph in Abbildung 1 bedeutet also, dass man vom Zustand q_1 mit dem Eingabezeichen `1`, `2`, `3`, `4`, `5`, `6`, `7`, `8` oder `9` in den Zustand q_3 wechseln kann.

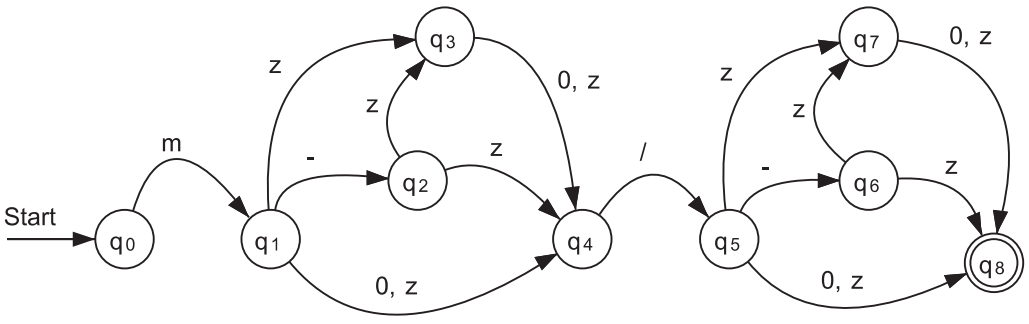


Abbildung 1: Übergangsgraph zum nichtdeterministischen endlichen Automaten A

Name: _____

a) *Geben Sie den Startzustand und die Menge der Endzustände des Automaten* `A` *an.*

Erläutern Sie anhand des Übergangsgraphen in Abbildung 1, warum der Automat `A` *kein deterministischer endlicher Automat (DEA) ist.*

Zeigen Sie anhand geeigneter Zustandsfolgen des Automaten `A` *bei der Abarbeitung der beiden Eingabewörter* `m0/-14` *und* `m04/29`*, dass eines der beiden Eingabewörter akzeptiert wird und das andere nicht akzeptiert wird.*

Erläutern Sie die Bedeutung des Zustands q_3 *im Sachzusammenhang.*

(13 Punkte)

b) *Entwickeln Sie mithilfe eines geeigneten Verfahrens den Übergangsgraphen eines deterministischen endlichen Automaten* `A'`*, der dieselbe Sprache akzeptiert wie der Automat* `A`*.*

(11 Punkte)

Nachdem der Zeichenstift mit dem „move“-Befehl zu einem Punkt des Zeichenbereichs bewegt wurde, können anschließend Linien gezeichnet werden. Sogenannte „Streckenzüge“ bestehen aus aneinandergesetzten Strecken. Für die dazugehörigen Linienbefehle gelten folgende Regeln:

I. Ein „gerade Linie“-Befehl zeichnet eine gerade Linie vom aktuellen Punkt zu einem Zielpunkt. Er wird durch das Symbol `g` eingeleitet, dem zwei Zahlen für die x- und die y-Koordinate des Zielpunktes folgen. Für die Koordinatenangaben sollen hier vereinfachend nur die einstelligen ganzen Zahlen `0,1,2,3,4,5,6,7,8` oder `9` ohne Vorzeichen möglich sein.

II. Ein Himmelsrichtungsbefehl zeichnet eine waagerechte bzw. senkrechte Linie mit Einheitslänge vom aktuellen Punkt aus in die gewünschte Richtung. Dazu muss nur das Symbol einer der vier Himmelsrichtungen angegeben werden: `n` für „north“, `s` für „south“, `e` für „east“ und `w` für „west“.

III. Ein Streckenzug kann aus beliebig vielen „gerade Linie“- und Himmelsrichtungsbefehlen in beliebiger Reihenfolge zusammengestellt werden, er hat aber mindestens einen „gerade Linie“- oder Himmelsrichtungsbefehl.

Name: _____

c) *Entscheiden Sie für die folgenden Wörter auf der Grundlage der genannten Regeln, ob sie einen regelkonformen Streckenzug darstellen:*

`g40n` `e1g6`

Entwerfen Sie eine reguläre Grammatik, die die Sprache aller regelkonformen Streckenzüge erzeugt.

Begründen Sie, dass die von Ihnen entworfene Grammatik regulär ist.

(12 Punkte)

In der im Folgenden betrachteten, vereinfachten Sprache zur Beschreibung einer Vektorgrafik beginnt ein sogenannter „Pfad“ mit einem „move“-Befehl und kann durch beliebig viele „gerade Linie“-Befehle ergänzt werden. Himmelsrichtungsbefehle gibt es bei Pfaden nicht. Für den Aufbau der Befehle gelten zudem die folgenden Regeln:

I. Als Zahlen sind hier für alle Befehle vereinfachend nur die einstelligen ganzen Zahlen `0,1,2` oder `3` ohne Vorzeichen möglich.

II. Der „move“-Befehl beginnt mit dem Symbol `m` und wird durch zwei Zahlen ergänzt, die die Koordinaten des Startpunktes darstellen.

III. Der „gerade Linie“-Befehl beginnt mit dem Symbol `g` und wird durch zwei Zahlen ergänzt, die die Koordinaten des Zielpunktes darstellen.

Es wurde eine Grammatik `G = (N,T,P,S)` konstruiert, die die Sprache der regelkonformen Pfade erzeugt.

Startsymbol: `S`
Nichtterminale: `N = {S,A,B}`
Terminale: `T = {0,1,2,3,g,m}`
Produktionen:

```
P =     {
            S →  A0 | A1 | A2 | A3,
            A →  B0 | B1 | B2 | B3,
            B →  Sg | m,
        }
```

d) *Zeigen Sie, dass das Wort* `m01g23` *aus der Grammatik* `G` *abgeleitet werden kann.*

Begründen Sie anhand der Produktionen, dass alle Wörter, die durch die Grammatik `G` *erzeugt werden, mit dem Terminal* `m` *beginnen.*

Erläutern Sie, wie die Regeln I bis III bei den Produktionen berücksichtigt wurden.

(9 Punkte)

Name: _____

e) Aus technischen Gründen sollen nur Pfade möglich sein, die maximal 15 „gerade Linie“-Befehle enthalten. Jemand behauptet:

> „Es ist nicht möglich, einen endlichen Automaten zu konstruieren, der entscheiden kann, ob ein regelkonform notierter Pfad die Bedingung erfüllt, da ein endlicher Automat nicht zählen kann.“

Beurteilen Sie die genannte Behauptung.

(5 Punkte)

Zugelassene Hilfsmittel:

- GTR (grafikfähiger Taschenrechner) oder CAS (Computer-Algebra-System)
- Wörterbuch zur deutschen Rechtschreibung

Unterlagen für die Lehrkraft

Abiturprüfung 2019

Informatik, Grundkurs

1. Aufgabenart

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

2. Aufgabenstellung[1]

siehe Prüfungsaufgabe

3. Materialgrundlage

entfällt

4. Bezüge zum Kernlehrplan und zu den Vorgaben 2019

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
     - Transformation eines nicht-deterministischen endlichen Automaten in einen deterministischen endlichen Automaten
   - Grammatiken regulärer Sprachen
     - Produktionen mit ε
   - Möglichkeiten und Grenzen von Automaten und formalen Sprachen

2. *Medien/Materialien*
   - entfällt

5. Zugelassene Hilfsmittel

- GTR (grafikfähiger Taschenrechner) oder CAS (Computer-Algebra-System)
- 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)

Startzustand: q_0
Menge der Endzustände: $\{q_8\}$

Der Automat A ist kein deterministischer endlicher Automat, weil man z. B. vom Zustand q_2 aus mit dem Zeichen z sowohl in den Zustand q_3 als auch in den Zustand q_4 wechseln kann.

Es gibt eine Zustandsfolge zum Eingabewort m0/-14, die in einem Endzustand endet:

$$q_0 \xrightarrow{m} q_1 \xrightarrow{0} q_4 \xrightarrow{/} q_5 \xrightarrow{-} q_6 \xrightarrow{1} q_7 \xrightarrow{4} q_8$$

Das Wort wird folglich akzeptiert.

Jede mögliche Zustandsfolge zum Eingabewort m04/29 beginnt wie folgt:

$$q_0 \xrightarrow{m} q_1 \xrightarrow{0} q_4$$

Von q_4 aus gibt es im Übergangsgraphen in Abbildung 1 keinen Zustandsübergang mit dem Eingabezeichen 4. Daher wechselt der Automat in einen Fehlerzustand, der kein Endzustand ist, und das Wort wird nicht akzeptiert.

Der Zustand q_3 wird erreicht, wenn zuvor das einleitende Symbol m und eine Ziffer ggf. mit Vorzeichen gelesen wurde. Diese Ziffer muss eine Zehnerziffer sein, auf die eine weitere Ziffer für die erste Koordinatenangabe folgen muss.

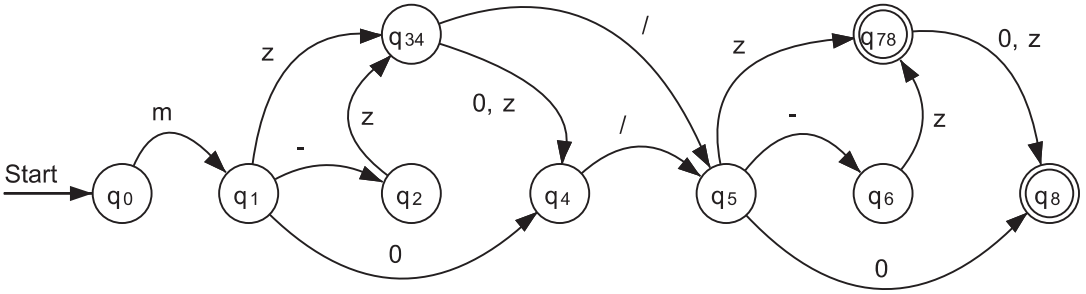
Teilaufgabe b)

Vorgehensweise:
Mit der Potenzmengenkonstruktion lässt sich folgende Zustandsübergangstabelle für den gesuchten DEA A' erstellen:

	z	0	m	-	/
{q_0}	{}	{}	{q_1}	{}	{}
{q_1}	{q_3,q_4}	{q_4}	{}	{q_2}	{}
{q_2}	{q_3,q_4}	{}	{}	{}	{}
{q_3,q_4}	{q_4}	{q_4}	{}	{}	{q_5}
{q_4}	{}	{}	{}	{}	{q_5}
{q_5}	{q_7,q_8}	{q_8}	{}	{q_6}	{}
{q_6}	{q_7,q_8}	{}	{}	{}	{}
{q_7,q_8}	{q_8}	{q_8}	{}	{}	{}
{q_8}	{}	{}	{}	{}	{}

Alle Zustände von A', die einer Menge von Zuständen entsprechen, die einen Endzustand von A enthalten, werden zu Endzuständen von A'. Daher wird neben dem vorherigen Endzustand q_8 auch der Zustand, der die Zustände q_7 und q_8 vereinigt, zum Endzustand.

Im folgenden Übergangsdiagramm werden die Zustände, die zwei Zustände vereinigen, mit den Indizes der enthaltenen Zustände dargestellt:



Teilaufgabe c)

Das Wort `g40n` beschreibt einen regelkonformen Streckenzug, da die erste Linie, eingeleitet durch das Symbol `g`, zwei nachfolgende Zahlen erwartet: `4` und `0` sind erlaubte Zahlendarstellungen. Es schließt sich ein Himmelsrichtungsbefehl an, dargestellt durch das Symbol `n`.

Das Wort `e1g6` beschreibt keinen regelkonformen Streckenzug, weil dem Himmelsrichtungsbefehl `e` eine Zahl folgt, obwohl Himmelsrichtungsbefehle laut Regel II nur aus einem Buchstabensymbol `n`, `s`, `e` oder `w` bestehen. Ein alternatives Argument ist folgendes: Es fehlt für den „gerade Linie“-Befehl eine zweite Zahlenangabe (Regel I).

Mögliche Grammatik:
Startsymbol: `S`
Nichtterminale: `N = {S,A,B,C}`
Terminale: `T = {0,1,2,3,4,5,6,7,8,9,g,n,s,e,w}`
Produktionen:

```
P = {
      S →  gA | nC | sC | eC | wC,
      A →  0B | 1B | 2B | 3B | 4B | 5B | 6B | 7B | 8B | 9B,
      B →  0C | 1C | 2C | 3C | 4C | 5C | 6C | 7C | 8C | 9C,
      C →  gA | nC | sC | eC | wC | ε
    }
```

Es ist auch eine verkürzende Notation denkbar, bei der alle Ziffern-Terminale durch das Symbol `z` dargestellt werden.

Diese Grammatik ist regulär, da für alle Produktionen gilt, dass auf der linken Seite ein Nichtterminal steht und auf der rechten Seite entweder das Symbol `ε` für das leere Wort oder ein Terminalsymbol gefolgt von einem Nichtterminal.

Teilaufgabe d)

Das Wort `m01g23` kann wie folgt abgeleitet werden:
`S → A3 → B23 → Sg23 → A1g23 → B01g23 → m01g23`
Da die Grammatik linkslinear ist, bauen sich die erzeugten Wörter von hinten nach vorne auf. Zuerst wird das letzte Terminal des Wortes erzeugt; zuletzt wird durch die einzige Produktion mit nur einem Terminal auf der rechten Seite (`B → m`) das Zeichen `m` erzeugt. Damit beginnen alle erzeugten Wörter mit dem Terminal `m`.
Die Regeln wurden wie folgt berücksichtigt:
Die Produktionen mit dem Startsymbol `S` auf der linken Seite leiten die Produktion der y-Koordinate eines Zielpunktes ein. Die danach zu verwendende Produktion mit dem Nichtterminal `A` auf der linken Seite erzeugt die x-Koordinate des Zielpunktes. (Regeln II – III)
Als Zahlen für die Koordinatenangaben sind bei den Produktionen nur die Terminale `0`, `1`, `2` und `3` vorgesehen (Regel I).
Durch die Wahl der Produktion mit dem Nichtterminal `B` auf der linken Seite wird festgelegt, zu welchem Befehl die zuvor erzeugten Koordinaten gehören:
Verwendet man die Produktion `B → Sg`, so ist es ein „gerade Linie“-Befehl (Regel III).
Verwendet man die Produktion `B → m`, so ist es der „move“-Befehl (Regel II).

Teilaufgabe e)

Endliche Automaten können nicht beliebig weit zählen, weil man je Zählwert einen eigenen Zustand benötigen würde, der diesen repräsentiert, endliche Automaten aber nur endlich viele Zustände haben.

Hier allerdings gibt es eine vorher feststehende maximale Anzahl an enthaltenen „gerade Linie“-Befehlen und damit auch an unterschiedlichen Pfaden, die akzeptiert werden soll. Die Sprache der akzeptierten Wörter ist also endlich. Daher kann man die Bedingung durchaus mithilfe eines endlichen Automaten überprüfen.

Die Behauptung ist demnach falsch.

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 den Startzustand und die Menge der Endzustände von A an.	2			
2	erläutert anhand des Übergangsgraphen, warum der Automat kein DEA ist.	2			
3	zeigt anhand geeigneter Zustandsfolgen des Automaten bei der Abarbeitung der beiden Eingabewörter, dass das Wort `m0/-14` akzeptiert wird und das Wort `m04/29` nicht akzeptiert wird.	6			
4	erläutert die Bedeutung des Zustands q_3 im Sachzusammenhang.	3			
Sachlich richtige Lösungsalternative zur Modelllösung: (13) …… ……					
	Summe Teilaufgabe a)	**13**			

Teilaufgabe b)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	entwickelt mithilfe eines geeigneten Verfahrens den Übergangsgraphen eines DEA `A'`, der dieselbe Sprache akzeptiert wie `A`.	11			
Sachlich richtige Lösungsalternative zur Modelllösung: (11) …… ……					
	Summe Teilaufgabe b)	**11**			

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

Teilaufgabe c)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	entscheidet auf der Grundlage der genannten Regeln, dass das Wort `g40n` einen regelkonformen Streckenzug darstellt und das Wort `e1g6` keinen regelkonformen Streckenzug darstellt.	4			
2	entwirft eine reguläre Grammatik, die die Sprache aller regelkonformen Streckenzüge erzeugt.	6			
3	begründet, dass die entworfene Grammatik regulär ist.	2			
Sachlich richtige Lösungsalternative zur Modelllösung: (12) …… ……					
	Summe Teilaufgabe c)	**12**			

Teilaufgabe d)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	zeigt, dass das genannte Wort aus der Grammatik `G` abgeleitet werden kann.	2			
2	begründet, dass alle Wörter, die durch die Grammatik `G` erzeugt werden können, mit dem Terminal `m` beginnen.	3			
3	erläutert, wie die Regeln I bis III bei den Produktionen berücksichtigt wurden.	4			
Sachlich richtige Lösungsalternative zur Modelllösung: (9) …… ……					
	Summe Teilaufgabe d)	**9**			

Teilaufgabe e)

	Anforderungen	Lösungsqualität			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	beurteilt die genannte Behauptung.	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**			
Punktzahl der gesamten Prüfungsleistung	**100**			
aus der Punktsumme resultierende Note gemäß nachfolgender Tabelle				
Note ggf. unter Absenkung um bis zu zwei Notenpunkte gemäß § 13 Abs. 2 APO-GOSt				
Paraphe				

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

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

Unterschrift, Datum:

Grundsätze für die Bewertung (Notenfindung)

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

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