Name: _____

Abiturprüfung 2013

Informatik, Grundkurs

Aufgabenstellung:

An der „*Grundschule am Zuckerberg*“ möchte man ein computergestütztes „Soziales Netzwerk“ installieren. Im Folgenden sollen einige Aspekte dieses Netzwerks modelliert werden.

In einem ersten, sehr einfachen Modell wird jedes Kind, das in einem sozialen Netzwerk angemeldet ist, unter seinem Vor- und Nachnamen verwaltet. Wir nehmen dabei an, dass es an der Schule keine zwei Kinder mit gleichen Vor- und Nachnamen gibt. Jedes Kind kann jederzeit andere Kinder einseitig als Freund angeben. Dabei muss in dieser ersten Version das als Freund angegebene Kind nicht zustimmen.

Die Schule verwaltet dann eine Liste aller Kinder, die an dem Netzwerk teilnehmen.

Auszüge aus den Dokumentationen der Klassen `Kind` und `Schule` befinden sich im Anhang.

a) In den Objekten der Klasse `Kind` werden die Freunde eines Kindes in einer Liste unter dem Namen `freunde` (Klasse `List`) verwaltet. In den Objekten der Klasse `Schule` werden alle Kinder (Klasse `Kind`) in einer Liste unter dem Namen `mitglieder` (Klasse `List`) verwaltet.

Überführen Sie die Dokumentationen der Klassen `Kind` *und* `Schule` *in ein Implementationsdiagramm.*

(8 Punkte)

Name: _____

b) Hier sehen Sie einen Ausschnitt aus dem Quelltext der Klasse `Kind`:

```
1 public class Kind {

2   private String vorname;
3   private String nachname;
4   private List freunde;
    ...
5   public void fuegeFreundHinzu(Kind pKind) {
6     freunde.append(pKind);
7   }
    ...
8 }
```

Mit der so implementierten Methode `fuegeFreundHinzu` ist es möglich, ein Kind mehrfach zu der Freundesliste hinzuzufügen. Auch sich selber kann man als Freund wählen.

Erweitern Sie die Implementation der Methode `fuegeFreundHinzu` *so, dass*

- *Kinder nicht mehrfach als Freund hinzugefügt werden können,*
- *ein Kind sich nicht selber als Freund hinzufügen kann.*

(8 Punkte)

c) In der Klasse `Schule` ist eine Methode `wasLiefereIch` implementiert:

```
 1 public int wasLiefereIch(Kind pKind) {
 2   int xyz = 0;
 3   mitglieder.toFirst();
 4   while (mitglieder.hasAccess()) {
 5     Kind dieses = (Kind) mitglieder.getObject() ;
 6     List freunde = dieses.gibFreunde();
 7     freunde.toFirst();
 8     while (freunde.hasAccess()) {
 9       Kind jenes = (Kind) freunde.getObject();
10       if (pKind.gleicht(jenes)) {
11         xyz = xyz + 1;
12       }
13       freunde.next();
14     }
15     mitglieder.next();
16   }
17   return xyz;
18 }
```

Analysieren Sie die Methode und beschreiben Sie deren Funktionalität im Sachzusammenhang anhand eines Beispiels.

Geben Sie insbesondere an, welche Aufgabe die Schleife in den Zeilen 8 bis 14 und welche Bedeutung die Variable `xyz` *hat.*

(10 Punkte)

Name: _____

d) In der Klasse `Schule` muss es möglich sein, ein Kind zu löschen, z. B. wenn es die Schule verlässt.

Implementieren Sie in der Klasse `Schule` *die Methode* `loescheKind` *mit dem Methodenkopf*

```
public void loescheKind(Kind pKind)
```

die das als Parameter übergebene Kind aus der Mitgliederliste der Schule sowie aus den Freundeslisten aller Kinder löscht.

(12 Punkte)

e) In dem bisher beschriebenen Netzwerk kann ein Kind A ein anderes Kind B zum Freund wählen, ohne dass B zustimmt. Dieses soll so abgeändert werden, dass ein Kind A den Wunsch äußern kann, dass es Freund eines Kindes B werden kann. Kind A wird aber erst als Freund registriert, wenn Kind B die Freundschaft bestätigt. Für jedes Kind können beliebig viele Freundschaftswünsche vorliegen.

Erweitern Sie das Netzwerk so, dass die oben beschriebene Funktionalität erreicht werden kann.

Geben Sie dazu die Änderungen in den Klassen an, es dürfen auch Klassen hinzugefügt werden. Dokumentieren Sie neue bzw. veränderte Methoden.

(12 Punkte)

Zugelassene Hilfsmittel:

- Wörterbuch zur deutschen Rechtschreibung
- Taschenrechner

Name: _____

Anhang

Auszug aus der Dokumentation der Klasse Kind

Ein Objekt dieser Klasse verwaltet ein Kind.

Konstruktor **`Kind(String pName, String pVorname)`**
Ein neues Objekt der Klasse `Kind` mit den angegebenen Werten wird erstellt. Das Kind hat (noch) keinen Freund.

Anfrage **`String gibVorname()`**
liefert den Vornamen des Kindes.

Anfrage **`String gibNachname()`**
liefert den Nachnamen des Kindes.

Auftrag **`void fuegeFreundHinzu(Kind pKind)`**
`pKind` wird an die Liste der Freunde dieses Kindes angehängt.

Auftrag **`void loescheFreund(Kind pKind)`**
`pKind` wird (sofern dort enthalten) aus der Liste der Freunde gelöscht.

Anfrage **`List gibFreunde()`**
liefert einen Verweis auf die Liste (Objekt der Klasse `List`), in der die Freunde dieses Kindes verwaltet werden.

Anfrage **`boolean gleicht(Kind pKind)`**
liefert genau dann `true`, wenn Vor- und Nachname dieses Kindes mit denen von `pKind` übereinstimmen.

Auszug aus der Dokumentation der Klasse Schule

Ein Objekt dieser Klasse verwaltet die Schüler der Schule, die an dem sozialen Netzwerk teilnehmen.

Konstruktor **`Schule()`**
Ein neues Objekt vom Typ `Schule` wird erstellt. Die Liste der verwalteten Kinder (im Folgenden Mitgliederliste genannt) ist leer.

Auftrag **`void kindHinzu(Kind pKind)`**
Das als Parameter übergebene Objekt `pKind` wird der Mitgliederliste hinzugefügt, sofern es noch nicht in der Mitgliederliste enthalten ist.

Anfrage **`boolean istEnthalten(Kind pKind)`**
liefert genau dann `true`, wenn `pKind` in der Mitgliederliste enthalten ist.

Auftrag **`void loescheKind(Kind pKind)`**
siehe Aufgabenteil d)

Name: _____

Die Klasse `List`

Objekte der Klasse `List` verwalten beliebig viele, linear angeordnete Objekte. 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 oder ein Listenobjekt an das Ende der Liste angefügt werden.

Dokumentation der Klasse `List`

Konstruktor **`List()`**
Eine leere Liste wird erzeugt.

Anfrage **`boolean isEmpty()`**
Die Anfrage liefert den Wert `true`, wenn die Liste keine Objekte enthält, sonst liefert sie den Wert `false`.

Anfrage **`boolean hasAccess()`**
Die Anfrage liefert den Wert `true`, wenn es ein aktuelles Objekt gibt, sonst liefert sie den Wert `false`.

Auftrag **`void next()`**
Falls die Liste nicht leer ist, es ein aktuelles Objekt gibt und dieses nicht das letzte Objekt der Liste ist, wird das dem aktuellen Objekt in der Liste folgende Objekt zum aktuellen Objekt, andernfalls gibt es nach Ausführung des Auftrags kein aktuelles Objekt, 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 **`Object getObject()`**

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 setObject(Object pObject)`**

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

Auftrag **`void append(Object pObject)`**

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

Auftrag **`void insert(Object pObject)`**

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

Auftrag **`void concat(List pList)`**

Die Liste `pList` wird an die Liste angehängt. Anschließend wird `pList` eine leere Liste. Das aktuelle Objekt bleibt unverändert. Falls `pList` `null` oder eine leere Liste ist, bleibt die Liste unverändert.

Auftrag **`void remove()`**

Falls es ein aktuelles Objekt gibt (`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 2013

Informatik, Grundkurs

1. Aufgabenart

Aufgabenart	Modellierung einer Problemstellung, Entwurf und Implementation von Algorithmen
Syntaxvariante	Java

2. Aufgabenstellung[1]

siehe Prüfungsaufgabe

3. Materialgrundlage

- entfällt

4. Bezüge zu den Vorgaben 2013

1. *Inhaltliche Schwerpunkte*
   Objektorientiertes Modellieren und Implementieren von kontextbezogenen Anwendungen
   - Konzepte des objektorientierten Modellierens
   - Datenstrukturen
     - Lineare Strukturen mit den Akzenten
       Lineare Liste

2. *Medien/Materialien*
   - entfällt

5. Zugelassene Hilfsmittel

- Wörterbuch zur deutschen Rechtschreibung
- Taschenrechner

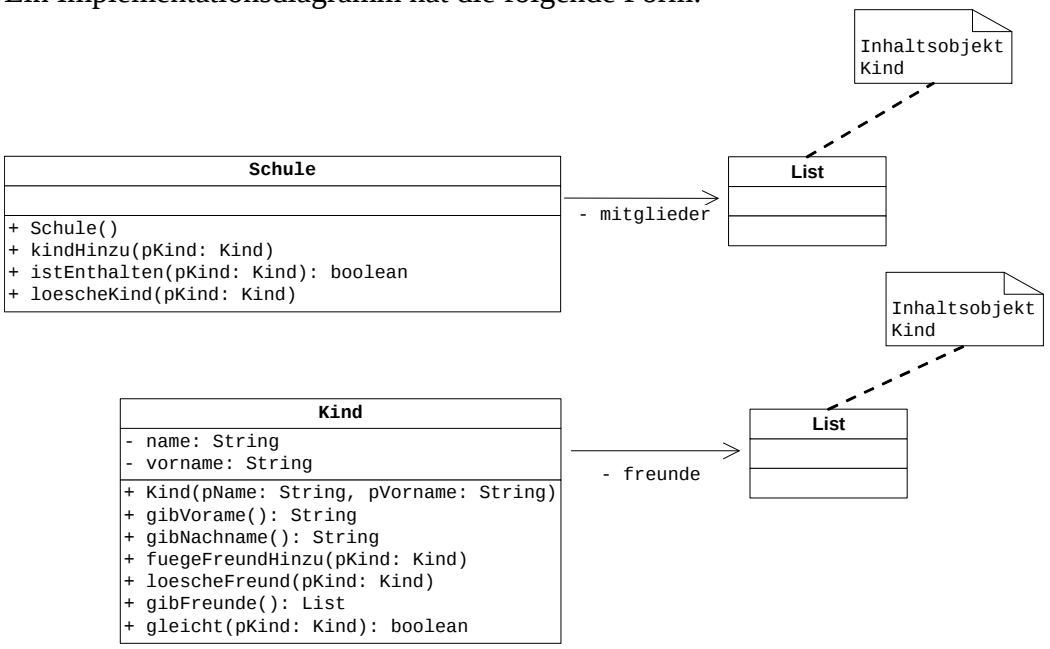
[1] Die Aufgabenstellung deckt inhaltlich alle drei Anforderungsbereiche ab.

6. Vorgaben für die Bewertung der Schülerleistungen

6.1 Modelllösungen

Teilaufgabe a)

Ein Implementationsdiagramm hat die folgende Form:



Nicht notwendig ist evtl. die zweifache Angabe des Klassendiagramms für die Klasse `List` mit dem zugehörigen Inhaltsobjekt `Kind`.

Teilaufgabe b)

Eine mögliche Lösung könnte folgende Form haben (ggf. kann die Überprüfung, ob ein Kind bereits als Freund eingetragen ist, auch in der Methode `fuegeFreundHinzu` geschehen, ohne eine Hilfsmethode einzuführen):

```
private boolean istFreund(Kind pKind) {
  freunde.toFirst();
  while (freunde.hasAccess()) {
    Kind dies = (Kind)freunde.getObject();
    if (dies.gleicht(pKind)) {
      return true;
    }
    freunde.next();
  }
  return false;
}

public void fuegeFreundHinzu(Kind pKind) {
  if (!istFreund(pKind) && !this.gleicht(pKind)) {
    freunde.append(pKind);
  }
}
```

Teilaufgabe c)

Die Methode findet heraus, wie viele Kinder dieses Kind als Freund haben. Die innere Schleife durchläuft dabei die Liste der Freunde eines (in der äußeren Schleife festgelegten) Kindes, und inkrementiert den Wert der Variablen `xyz`, wenn es als Freund gewählt wurde.

Die Variable `xyz` verwaltet also die Zahl der Kinder, die das als Parameter übergebene Kind als Freund haben.

Die Methode könnte also dazu dienen, die Beliebtheit eines Kindes zu erforschen. Hierzu sollte ein passendes Beispiel angegeben werden.

Teilaufgabe d)

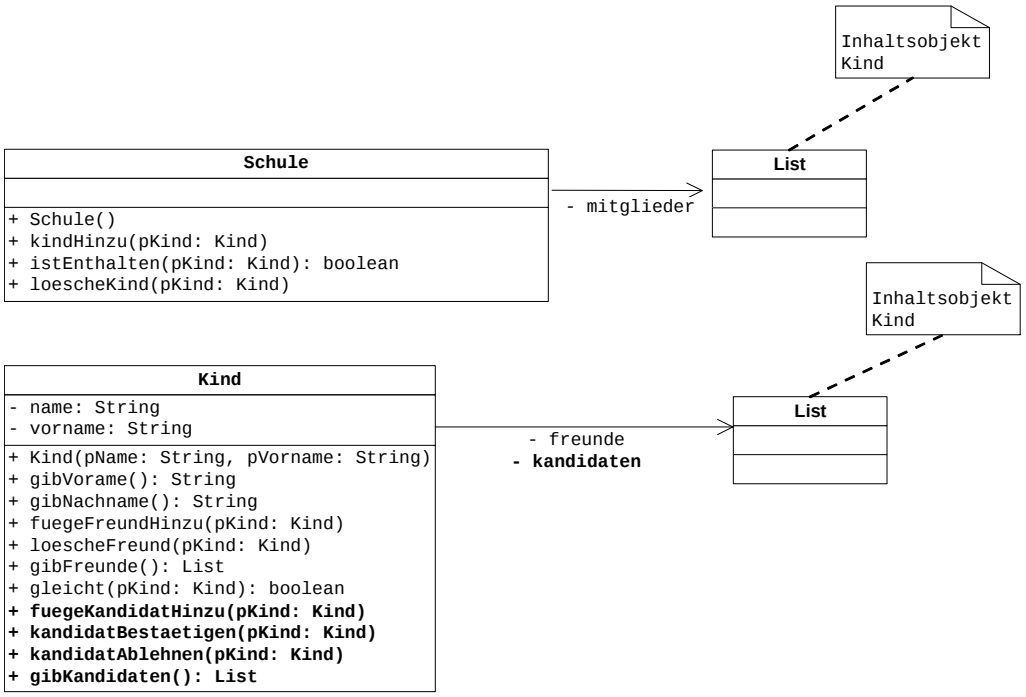
Eine mögliche Implementation kann folgendermaßen erfolgen:

```
public void loescheKind(Kind pKind) {
  mitglieder.toFirst();
  while (mitglieder.hasAccess()) {
    Kind dieses = (Kind)mitglieder.getObject();
    dieses.loescheFreund(pKind);
    mitglieder.next();
  }

  boolean geloescht = false;
  mitglieder.toFirst();
  while (!geloescht && mitglieder.hasAccess()) {
    Kind jenes = (Kind)mitglieder.getObject();
    if (pKind.gleicht(jenes)) {
      mitglieder.remove();
      geloescht = true;
    } else
      mitglieder.next();
  }
}
```

Teilaufgabe e)

Eine mögliche Lösung besteht darin, dass die Klasse `Kind` erweitert wird.



Die Ergänzungen sind im Implementationsdiagramm durch Fettdruck gekennzeichnet.

Ein Kind, das Freund eines anderen Kindes werden will, wird als Kandidat bezeichnet. Die Klasse `Kind` erhält zusätzlich eine Liste, in der die Kandidaten, die Freunde werden wollen, verwaltet werden.

Dokumentation der neuen Methoden:

Auftrag **`void fuegeKandidatHinzu(Kind pKind)`**
`pKind` wird in die Liste der Kandidaten eingefügt.

Auftrag **`void kandidatBestaetigen(Kind pKind)`**
Die Freundschaft wird bestätigt. `pKind` wird aus der Kandidatenliste gelöscht und in die Freundesliste eingetragen.

Auftrag **`void KandidatAblehnen(Kind pKind)`**
Die Freundschaft wird nicht bestätigt. `pKind` wird aus der Kandidatenliste gelöscht.

Anfrage **`List gibKandidaten()`**
Es wird eine Liste aller Kandidaten zurückgeliefert, die als Inhaltobjekte Objekte der Klasse `Kind` hat.

6.2 Teilleistungen – Kriterien

Teilaufgabe a)

	Anforderungen	**maximal erreichbare Punktzahl**
	Der Prüfling	
1	überführt die Klasse `Kind` in den zugehörigen Diagrammteil.	3
2	überführt die Klasse `Schule` in den zugehörigen Diagrammteil.	3
3	fügt die Klasse `List` dem Diagramm hinzu.	2
Der gewählte Lösungsansatz und -weg muss nicht identisch mit dem der Modelllösung sein. Sachlich richtige Alternativen werden an dieser Stelle mit entsprechender Punktzahl bewertet.		

Teilaufgabe b)

	Anforderungen	**maximal erreichbare Punktzahl**
	Der Prüfling	
1	erweitert den Code so, dass ein Kind sich nicht selber als Freund wählen kann.	2
2	erweitert den Code um einen Programmteil, der prüft, ob ein Kind bereits als Freund eingetragen ist.	4
3	erweitert den Code so, dass ein bereits als Freund eingetragenes Kind nicht erneut eingetragen wird.	2
Der gewählte Lösungsansatz und -weg muss nicht identisch mit dem der Modelllösung sein. Sachlich richtige Alternativen werden an dieser Stelle mit entsprechender Punktzahl bewertet.		

Teilaufgabe c)

	Anforderungen	**maximal erreichbare Punktzahl**
	Der Prüfling	
1	gibt die Bedeutung der Schleife in den Zeilen 8 bis 14 an.	4
2	gibt die Bedeutung der Variablen `xyz` an.	2
3	analysiert die Methode, indem die Funktionalität im Sachzusammenhang angegeben wird.	4
Der gewählte Lösungsansatz und -weg muss nicht identisch mit dem der Modelllösung sein. Sachlich richtige Alternativen werden an dieser Stelle mit entsprechender Punktzahl bewertet.		

Teilaufgabe d)

	Anforderungen	**maximal erreichbare Punktzahl**
	Der Prüfling	
1	implementiert eine Schleife, die das angegebene Kind aus der Freundesliste aller Kinder entfernt.	6
2	implementiert den Programmteil, mit dem das angegebene Kind aus der Mitgliederliste entfernt wird.	6
Der gewählte Lösungsansatz und -weg muss nicht identisch mit dem der Modelllösung sein. Sachlich richtige Alternativen werden an dieser Stelle mit entsprechender Punktzahl bewertet.		

Teilaufgabe e)

	Anforderungen	**maximal erreichbare Punktzahl**
	Der Prüfling	
1	gibt die Erweiterungen an, die die Funktionalität ermöglichen.	4
2	gibt die Änderungen in den Klassen an.	4
3	dokumentiert die neuen bzw. veränderten Methoden.	4
Der gewählte Lösungsansatz und -weg muss nicht identisch mit dem der Modelllösung sein. Sachlich richtige Alternativen werden an dieser Stelle mit entsprechender Punktzahl bewertet.		

7. Bewertungsbogen zur Prüfungsarbeit

Name des Prüflings: _____ Kursbezeichnung: _____

Schule: _____

Teilaufgabe a)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK[2]**	**ZK**	**DK**
1	überführt die Klasse …	3			
2	überführt die Klasse …	3			
3	fügt die Klasse …	2			
	Summe Teilaufgabe a)	**8**			

Teilaufgabe b)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	erweitert den Code …	2			
2	erweitert den Code …	4			
3	erweitert den Code …	2			
	Summe Teilaufgabe b)	**8**			

Teilaufgabe c)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	gibt die Bedeutung …	4			
2	gibt die Bedeutung …	2			
3	analysiert die Methode …	4			
	Summe Teilaufgabe c)	**10**			

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

Teilaufgabe d)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	implementiert eine Schleife …	6			
2	implementiert den Programmteil …	6			
	Summe Teilaufgabe d)	**12**			

Teilaufgabe e)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	gibt die Erweiterungen …	4			
2	gibt die Änderungen …	4			
3	dokumentiert die neuen …	4			
	Summe Teilaufgabe e)	**12**			

	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				
Note ggf. unter Absenkung um ein bis zwei Notenpunkte gemäß § 13 Abs. 2 APO-GOSt				
Paraphe				

ggf. arithmetisches Mittel der Punktsummen aus EK und ZK: _____

ggf. arithmetisches Mittel der Notenurteile aus EK und ZK: _____

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 – 39
mangelhaft plus	3	38 – 33
mangelhaft	2	32 – 27
mangelhaft minus	1	26 – 20
ungenügend	0	19 – 0

Name: _____

Abiturprüfung 2013

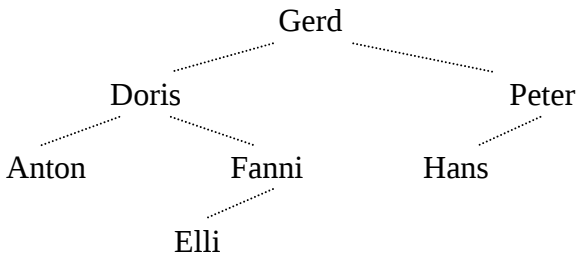
Informatik, Grundkurs

Aufgabenstellung:

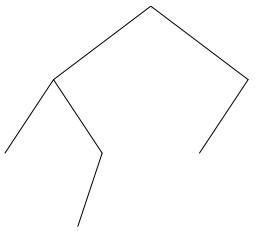
Um sich die Struktur eines binären Suchbaums besser vorstellen zu können, kann man zu jedem Suchbaum ein zugehöriges Baumgerüst zeichnen.

Zum Beispiel:

Suchbaum „Familie“



Baumgerüst



a) Die folgenden Namen sollen in der vorliegenden Reihenfolge in einen neuen Suchbaum eingefügt werden:

Heiner, Dirk, Gritt, Paul, Bert, Ted, Bernd, Ina, Elisabeth

Stellen Sie den entsprechenden binären Suchbaum und das zugehörige Baumgerüst dar.

(6 Punkte)

b) Damit ein Baum vom Computer gezeichnet werden kann, wird eine Klasse `Punkt` benötigt, die durch das nebenstehende Klassendiagramm gegeben ist.

Implementieren Sie die Klasse `Punkt`.

Punkt
- x: double - y: double
+ Punkt(pX: double, pY: double) + gibX(): double + gibY(): double

(6 Punkte)

Name: _____

c) Zur Vorbereitung der grafischen Darstellung soll mit der folgenden Methode aus dem übergebenen Suchbaum zuerst ein Binärbaum erzeugt werden, der die Koordinaten des Baumgerüsts verwaltet.

```
 1 public BinaryTree geruest(BinarySearchTree pSbaum,
                             double pBreite, double pX, double pY) {
 2   BinaryTree gBaum = new BinaryTree();
 3   if (!pSbaum.isEmpty()) {
 4     Punkt startpunkt = new Punkt(pX, pY);
 5     BinarySearchTree lBaum = pSbaum.getLeftTree();
 6     BinarySearchTree rBaum = pSbaum.getRightTree();
 7     BinaryTree links;
 8     BinaryTree rechts;
 9     double neubreite = pBreite / 2;
10     links = geruest(lBaum, neubreite, pX - neubreite, pY + 20);
11     rechts = geruest(rBaum, neubreite, pX + neubreite, pY + 20);
12     gBaum.setObject(startpunkt);
13     gBaum.setLeftTree(links);
14     gBaum.setRightTree(rechts);
15   }
16   return gBaum;
17 }
```

Bestimmen Sie die Punkte, die der entstehende Binärbaum enthält, wenn die Methode als Suchbaum den oben gezeichneten Baum „Familie“ (mit der Wurzel „Gerd“), als aktuellen Wert für `pBreite` *die Zahl* `256` *, als aktuellen Wert für* `pX` *die Zahl* `500` *und als aktuellen Wert für* `pY` *die Zahl* `50` *übergeben bekommt.*

Erläutern Sie zeilenweise den obigen Quellcode und geben Sie die Funktionalität des obigen Algorithmus an. (16 Punkte)

d) Eine Methode

```
public void zeichneLinie(Punkt pP1, Punkt pP2)
```

sei vorgegeben, die bei Übergabe zweier Objekte pP1 und pP2 der Klasse `Punkt` eine Verbindungslinie zwischen den beiden zugehörigen Punkten (`x1`, `y1`) und (`x2`, `y2`) zeichnet.

Implementieren Sie eine Methode

```
public void zeichneGrafikbaum(BinaryTree pBaum)
```

die bei Übergabe eines Binärbaumes mit Objekten der Klasse `Punkt` *in den Knoten diesen Baum unter Zuhilfenahme der Methode* `zeichneLinie` *zeichnerisch darstellt.*

(12 Punkte)

Name: _____

e) Besondere Bedeutung hat die längste Abfolge der Linien von der Wurzel bis zum untersten Blatt (in der Zeichnung ganz oben besteht sie z. B. aus drei Linien).

Analysieren Sie die Bedeutung dieser Anzahl der Linien (Kanten) in einem binären Suchbaum bzgl. der Suche nach einem beliebigen Knoteninhalt, indem Sie Angaben zum Zeitverhalten machen und dieses vergleichen mit dem Zeitverhalten bei der Suche in einer Liste mit denselben Daten. Berücksichtigen Sie insbesondere auch die Situation bei einem ausgeglichenen Suchbaum.

(10 Punkte)

Zugelassene Hilfsmittel:

- Wörterbuch zur deutschen Rechtschreibung
- Taschenrechner

Name: _____

Anhang

Die Klasse BinaryTree

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

Dokumentation der Klasse BinaryTree

Konstruktor **`BinaryTree()`**
Nach dem Aufruf des Konstruktors existiert ein leerer Binärbaum.

Konstruktor **`BinaryTree(Object pObject)`**
Wenn der Parameter `pObject` ungleich `null` ist, existiert nach dem Aufruf des Konstruktors der Binärbaum und hat `pObject` als Inhaltsobjekt und zwei leere Teilbäume. Falls der Parameter `null` ist, wird ein leerer Binärbaum erzeugt.

Konstruktor **`BinaryTree(Object pObject, BinaryTree pLeftTree, BinaryTree pRightTree)`**
Wenn der Parameter `pObject` ungleich `null` ist, wird ein Binärbaum mit `pObject` als Inhaltsobjekt und den beiden Teilbäume `pLeftTree` und `pRightTree` erzeugt. Sind `pLeftTree` oder `pRightTree` gleich `null`, wird der entsprechende Teilbaum als leerer Binärbaum eingefügt. Wenn der Parameter `pObject` gleich `null` ist, wird ein leerer Binärbaum erzeugt.

Anfrage **`boolean isEmpty()`**
Diese Anfrage liefert den Wahrheitswert `true`, wenn der Binärbaum leer ist, sonst liefert sie den Wert `false`.

Auftrag **`void setObject(Object pObject)`**
Wenn der Binärbaum leer ist, wird der Parameter `pObject` als Inhaltsobjekt sowie ein leerer linker und rechter Teilbaum eingefügt. Ist der Binärbaum nicht leer, wird das Inhaltsobjekt durch `pObject` ersetzt. Die Teilbäume werden nicht geändert. Wenn `pObject` `null` ist, bleibt der Binärbaum unverändert.

Anfrage **`Object getObject()`**
Diese Anfrage liefert das Inhaltsobjekt des Binärbaums. Wenn der Binärbaum leer ist, wird `null` zurückgegeben.

Name: _____

Auftrag **`void setLeftTree(BinaryTree pTree)`**
Wenn der Binärbaum leer ist, wird `pTree` nicht angehängt. Andernfalls erhält der Binärbaum den übergebenen Baum als linken Teilbaum. Falls der Parameter `null` ist, ändert sich nichts.

Auftrag **`void setRightTree(BinaryTree pTree)`**
Wenn der Binärbaum leer ist, wird `pTree` nicht angehängt. Andernfalls erhält der Binärbaum den übergebenen Baum als rechten Teilbaum. Falls der Parameter `null` ist, ändert sich nichts.

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

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

Die abstrakte Klasse `Item`

Die Klasse **`Item`** ist abstrakte Oberklasse aller Klassen, deren Objekte in einen Suchbaum (`BinarySearchTree`) eingefügt werden sollen. Die Ordnungsrelation wird in den Unterklassen von `Item` durch Überschreiben der drei abstrakten Methoden `isEqual`, `isGreater` und `isLess` festgelegt.

Dokumentation der Klasse `Item`

Anfrage **`abstract boolean isEqual(Item pItem)`**
Wenn festgestellt wird, dass das Objekt, von dem die Methode aufgerufen wird, bzgl. der gewünschten Ordnungsrelation gleich dem Objekt `pItem` ist, wird `true` geliefert. Sonst wird `false` geliefert.

Anfrage **`abstract boolean isLess(Item pItem)`**
Wenn festgestellt wird, dass das Objekt, von dem die Methode aufgerufen wird, bzgl. der gewünschten Ordnungsrelation kleiner als das Objekt `pItem` ist, wird `true` geliefert. Sonst wird `false` geliefert.

Anfrage **`abstract boolean isGreater(Item pItem)`**
Wenn festgestellt wird, dass das Objekt, von dem die Methode aufgerufen wird, bzgl. der gewünschten Ordnungsrelation größer als das Objekt `pItem` ist, wird `true` geliefert. Sonst wird `false` geliefert.

Name: _____

Die Klasse BinarySearchTree

In einem Objekt der Klasse **`BinarySearchTree`** werden beliebig viele Objekte in einem Binärbaum (binärer Suchbaum) entsprechend einer Ordnungsrelation verwaltet. Ein Objekt der Klasse stellt entweder einen leeren Baum dar oder verwaltet ein Inhaltsobjekt sowie einen linken und einen rechten Teilbaum, die ebenfalls Objekte der Klasse `BinarySearchTree` sind. Dabei gilt:

Die Inhaltsobjekte sind Objekte einer Unterklasse von `Item`, in der durch Überschreiben der drei Vergleichsmethoden `isLess`, `isEqual`, `isGreater` (siehe Item) eine eindeutige Ordnungsrelation festgelegt sein muss.

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.

Die Klasse `BinarySearchTree` ist keine Unterklasse der Klasse `BinaryTree`, so dass deren Methoden nicht zur Verfügung stehen.

Dokumentation der Klasse BinarySearchTree

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(Item pItem)`**

Falls ein bezüglich der verwendeten Vergleichsmethode `isEqual` mit `pItem` übereinstimmendes Objekt im geordneten Baum enthalten ist, passiert nichts. Andernfalls wird das Objekt `pItem` entsprechend der vorgegebenen Ordnungsrelation in den Baum eingeordnet. Falls der Parameter `null` ist, ändert sich nichts.

Anfrage **`Item search(Item pItem)`**

Falls ein bezüglich der verwendeten Vergleichsmethode `isEqual` mit `pItem` übereinstimmendes Objekt im binären Suchbaum enthalten ist, liefert die Anfrage dieses, ansonsten wird `null` zurückgegeben. Falls der Parameter `null` ist, wird `null` zurückgegeben.

Name: _____

Auftrag **`void remove(Item pItem)`**

Falls ein bezüglich der verwendeten Vergleichsmethode `isEqual` mit `pItem` übereinstimmendes Objekt im binären Suchbaum enthalten ist, wird dieses entfernt. Falls der Parameter `null` ist, ändert sich nichts.

Anfrage **`Item getItem()`**

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

Anfrage **`BinarySearchTree 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 getRightTree()`**

Diese Anfrage liefert den rechten Teilbaum des Suchbaumes. Der Suchbaum ändert sich nicht. Wenn er leer ist, wird `null` zurückgegeben.

Unterlagen für die Lehrkraft

Abiturprüfung 2013

Informatik, Grundkurs

1. Aufgabenart

Aufgabenart	Aufgabenstellungen aus dem Bereich Binärbäume
Syntaxvariante	Java

2. Aufgabenstellung[1]

siehe Prüfungsaufgabe

3. Materialgrundlage

- entfällt

4. Bezüge zu den Vorgaben 2013

1. *Inhaltliche Schwerpunkte*
   Objektorientiertes Modellieren und Implementieren von kontextbezogenen Anwendungen
   - Datenstrukturen
     - Baumstrukturen mit den Akzenten
       Binärbaum (Anwendung der Standardoperationen; Traversierungsalgorithmen)
       Binärer Suchbaum (Anwendung der Standardoperationen)

2. *Medien/Materialien*
   - entfällt

5. Zugelassene Hilfsmittel

- Wörterbuch zur deutschen Rechtschreibung
- Taschenrechner

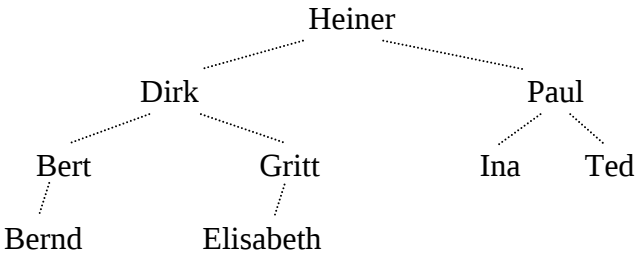
[1] Die Aufgabenstellung deckt inhaltlich alle drei Anforderungsbereiche ab.

6. Vorgaben für die Bewertung der Schülerleistungen

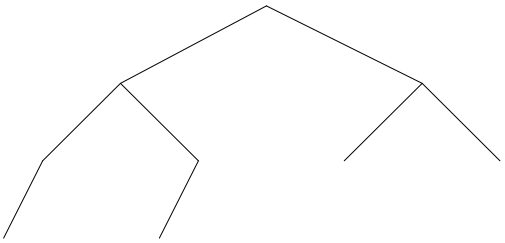
6.1 Modelllösung

Teilaufgabe a)

Suchbaum mit Namen



Baumgerüst



Teilaufgabe b)

```
public class Punkt
{
   private double x, y;

   public Punkt( double pX, double pY) {
       x = pX;
       y = pY;
    }

    public double gibX() {
        return x;
    }

    public double gibY() {
        return y;
    }
}
```

Teilaufgabe c)

Entsprechend der Baumstruktur erhält man die folgenden Koordinaten:

(500,50)
(372,70) (628,70)
(308, 90) (436,90) (564,90)
(404,110)

Der binäre Suchbaum wird mit einer Preorder-Traversierung durchlaufen.

Aufgrund der übergebenen aktuellen Parameter wird ein Punkt mit den Koordinaten (pX, pY) erzeugt und in einen Binärbaum eingegeben. Die beiden linken und rechten binären Teilbäume entstehen durch den rekursiven Abstieg.

Im Detail:

In Zeile 2 wird der Binärbaum erzeugt.

Die Bedingung in Zeile 3 terminiert den rekursiven Abstieg im binären Suchbaum.

In der Zeile 4 wird ein neuer Punkt mit den aktuellen Koordinaten erzeugt, der in Zeile 12 als Wurzelinhalt des neuen Binärbaumes bestimmt wird.

Die Zeilen 5 bzw. 8 deklarieren die linken und rechten Teilbäume des Suchbaumes bzw. des Binärbaumes. Die beiden Teilbäume des Suchbaumes werden zugewiesen.

In der Zeile 9 wird die Breite der neuen Bäume als halb so groß wie beim bisherigen Baum festgelegt.

In den beiden folgenden Zeilen werden der linke und der rechte Binärbaum rekursiv erzeugt.

Durch die aktuellen Parameter wird festgelegt, dass die neuen Bäume in ihren Wurzeln eine y-Koordinate erhalten, die jeweils um 20 größer ist als die bisherige. Daraus ergibt sich, dass die Ebenen äquidistant sind.

Die Verschiebung nach links oder rechts in x-Richtung ergibt sich aus der neuen Breite der beiden (Teil-)Binärbäume, die in Zeile 9 festgelegt wurde.

In den Zeilen 12 bis 14 werden dem neuen Binärbaum der Wurzelinhalt und die beiden Teilbäume zugewiesen.

Dieser wird in Zeile 16 schließlich zurückgegeben.

Insgesamt wird ein Binärbaum erstellt, der die gleiche Struktur wie der übergebene Suchbaum hat. Die Knoten des neuen Binärbaumes enthalten die Punkte mit den passenden Koordinaten, deren Verbindungen von der Wurzel nach unten das Gerüst des Suchbaumes abbilden.

Teilaufgabe d)

```
public void zeichneGrafikbaum(BinaryTree pBaum) {
  if (!pBaum.isEmpty()) {
    Punkt  p= (Punkt) pBaum.getObject();
    BinaryTree links, rechts;
    links = pBaum.getLeftTree();
    if (!links.isEmpty()) {
       Punkt linkerPunkt = (Punkt) links.getObject();
       zeichneLinie(p , linkerPunkt);
       zeichneGrafikbaum(links);
    }
    rechts = pBaum.getRightTree();
    if (!rechts.isEmpty()) {
       Punkt rechterPunkt = (Punkt) rechts.getObject();
       zeichneLinie(p , rechterPunkt);
       zeichneGrafikbaum(rechts);
    }
  }
}
```

Teilaufgabe e)

Die längste Abfolge der Linien von der Wurzel bis zum untersten Blatt bestimmt die Höhe des zugehörigen binären Suchbaumes. Wie der Name schon sagt, ist die wichtigste Aufgabe in einem solchen Baum das Suchen. Der Aufwand dafür wird maximal, wenn der längste Pfad von der Wurzel bis zum untersten Blatt durchlaufen werden muss. Im schlimmsten Fall ist der Baum zur Liste entartet und die Suchzeit entspricht der in einer Liste. In einem ausgeglichenen Suchbaum mit n Knoten ist dieser Aufwand im Durchschnitt proportional zu log(n). In einer vergleichbaren Liste mit n Knoten kann die Suche im schlimmsten Fall durch den Vergleich mit n Knoten bestimmt werden. Im Durchschnitt müssen n/2 Knoten verglichen werden. Der Suchaufwand ist also proportional zu n. Aus diesem Unterschied zwischen log(n) und n ergibt sich der entscheidende Vorteil einer Suche im ausgeglichenen binären Suchbaum.

6.2 Teilleistungen – Kriterien

Teilaufgabe a)

	Anforderungen	**maximal erreichbare Punktzahl**
	Der Prüfling	
1	stellt den binären Suchbaum dar.	3
2	stellt das zugehörige Baumgerüst dar.	3
Der gewählte Lösungsansatz und -weg muss nicht identisch mit dem der Modelllösung sein. Sachlich richtige Alternativen werden an dieser Stelle mit entsprechender Punktzahl bewertet.		

Teilaufgabe b)

	Anforderungen	**maximal erreichbare Punktzahl**
	Der Prüfling	
1	implementiert den Konstruktor.	3
2	implementiert die beiden Methoden.	3
Der gewählte Lösungsansatz und -weg muss nicht identisch mit dem der Modelllösung sein. Sachlich richtige Alternativen werden an dieser Stelle mit entsprechender Punktzahl bewertet.		

Teilaufgabe c)

	Anforderungen	**maximal erreichbare Punktzahl**
	Der Prüfling	
1	gibt die Koordinaten der Punkte an.	6
2	erläutert die Programmzeilen/-bereiche.	7
3	gibt die Funktionalität des Algorithmus an.	3
Der gewählte Lösungsansatz und -weg muss nicht identisch mit dem der Modelllösung sein. Sachlich richtige Alternativen werden an dieser Stelle mit entsprechender Punktzahl bewertet.		

Teilaufgabe d)

	Anforderungen	**maximal erreichbare Punktzahl**
	Der Prüfling	
1	implementiert die Methode `zeichneGrafikbaum`.	12
Der gewählte Lösungsansatz und -weg muss nicht identisch mit dem der Modelllösung sein. Sachlich richtige Alternativen werden an dieser Stelle mit entsprechender Punktzahl bewertet.		

Teilaufgabe e)

	Anforderungen	**maximal erreichbare Punktzahl**
	Der Prüfling	
1	gibt den Zusammenhang zwischen der Länge des Pfades und der Höhe des Baumes an.	2
2	erläutert, dass die Höhe des Suchbaumes die maximale Suchzeit im Suchbaum bestimmt.	2
3	erläutert die log(n) Beziehung zwischen der Anzahl der Knoten und der Höhe in einem nicht entarteten Suchbaum.	2
4	entwickelt das Zeitverhalten ~ n für Listen.	2
5	vergleicht das Zeitverhalten für die Suche im Suchbaum und in einer Liste.	2
Der gewählte Lösungsansatz und -weg muss nicht identisch mit dem der Modelllösung sein. Sachlich richtige Alternativen werden an dieser Stelle mit entsprechender Punktzahl bewertet.		

7. Bewertungsbogen zur Prüfungsarbeit

Name des Prüflings: _____ Kursbezeichnung: _____

Schule: _____

Teilaufgabe a)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK[2]**	**ZK**	**DK**
1	stellt den binären …	3			
2	stellt das zugehörige …	3			
	Summe Teilaufgabe a)	**6**			

Teilaufgabe b)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	implementiert den Konstruktor.	3			
2	implementiert die beiden …	3			
	Summe Teilaufgabe b)	**6**			

Teilaufgabe c)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	gibt die Koordinaten …	6			
2	erläutert die Programmzeilen …	7			
3	gibt die Funktionalität …	3			
	Summe Teilaufgabe c)	**16**			

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

Teilaufgabe d)

	Anforderungen	Lösungsqualität			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	implementiert die Methode …	12			
	Summe Teilaufgabe d)	**12**			

Teilaufgabe e)

	Anforderungen	Lösungsqualität			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	gibt den Zusammenhang …	2			
2	erläutert, dass die …	2			
3	erläutert die log(n) …	2			
4	entwickelt das Zeitverhalten …	2			
5	vergleicht das Zeitverhalten …	2			
	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				
Note ggf. unter Absenkung um ein bis zwei Notenpunkte gemäß § 13 Abs. 2 APO-GOSt				
Paraphe				

ggf. arithmetisches Mittel der Punktsummen aus EK und ZK: _____

ggf. arithmetisches Mittel der Notenurteile aus EK und ZK: _____

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 – 39
mangelhaft plus	3	38 – 33
mangelhaft	2	32 – 27
mangelhaft minus	1	26 – 20
ungenügend	0	19 – 0

Name: _____

Abiturprüfung 2013

Informatik, Grundkurs

Aufgabenstellung:

Als es für Computer noch keine graphischen Oberflächen gab, die z. B. mit einer Maus zu steuern sind, konnten Computer ausschließlich über die Tastatur bedient werden. Das Betriebssystem stellte dabei eine große Vielfalt von Befehlen bereit, die über die Tastatur eingegeben werden mussten. Diese Befehle existieren in den aktuellen Betriebssystemen auch heute noch. Ähnlich wie bei der Programmierung müssen diese Befehle nach einem bestimmten Schema eingegeben werden. Im Folgenden soll exemplarisch eine vereinfachte Version des Befehls `dir` behandelt werden. Er listet Dateien und Unterverzeichnisse eines Verzeichnisses auf.

Der Befehl `dir` hat sogenannte *Optionen* und *Parameter*.

Sprache der Optionen und Parameter des Befehls `dir`

Beispiel:
`-A-L/media`

Jede Option beginnt immer mit einem Minuszeichen „-“. Es folgt genau ein Buchstabe. Optionen werden immer vor den Parametern angegeben.

Für den Befehl `dir` existieren die Optionen `A` (alle Dateien und Verzeichnisse auflisten, auch versteckte) und `L` (für die Darstellung einer Datei oder eines Verzeichnisses pro Zeile). Die Optionen dürfen beliebig oft und in beliebiger Reihenfolge angegeben oder weggelassen werden.

Als Parameter wird entweder nur der *Name des Verzeichnisses* oder ein *Verzeichnispfad* angegeben, dessen Inhalt aufgelistet werden soll. Der Parameter muss angegeben werden. Im obigen Beispiel ist der Teil`/media` nach den Optionen `-A` und `-L` ein Verzeichnispfad.

Name: _____

Der Name eines Verzeichnisses besteht aus (mindestens einem) Buchstaben von `a` bis `z`. Der Verzeichnispfad beginnt mit einem Schrägstrich „`/`“ oder einem Verzeichnisnamen und wird mit den Namen beliebig vieler Verzeichnisse fortgesetzt. Die Verzeichnisnamen werden ebenfalls durch einen Schrägstrich voneinander getrennt. Der Schrägstrich darf nicht zweimal hintereinander vorkommen. Er darf aber am Ende eines Verzeichnispfads stehen. Das oberste Verzeichnis wird durch den Schrägstrich „`/`“ angegeben.

Beispiele für Verzeichnispfade:

```
home/ich/musik/
home/ich
/etc/sshd
/var/log/
/
```

Werden solche Optionen und Parameter eingegeben, muss überprüft werden, ob sie syntaktisch korrekt sind. Diese Überprüfung kann durch einen deterministischen endlichen Automaten dargestellt werden.

Der folgende Automat überprüft angeblich die Eingabe der Optionen und Parameter des Befehls `dir`.

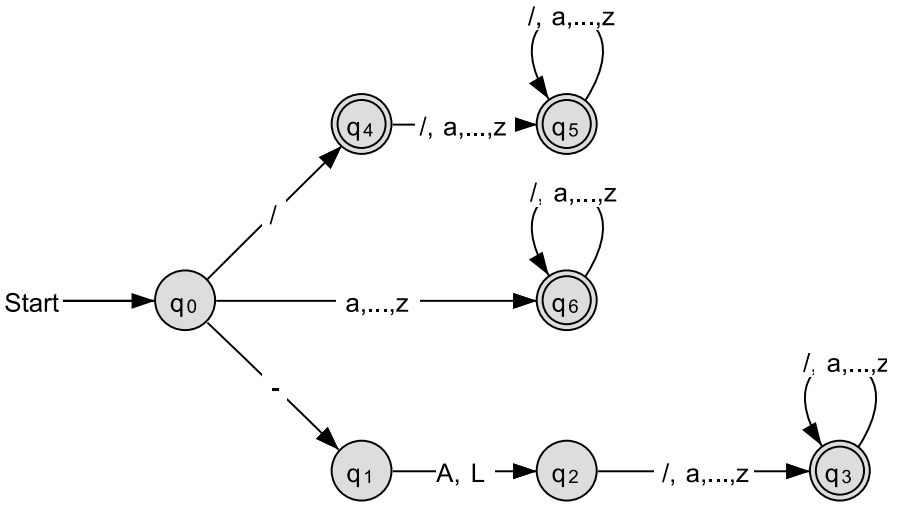


Abbildung 1: Automat 1

Nicht dargestellte Übergänge führen in einen Fehlerzustand, der im Graphen und in der Tabelle nicht dargestellt wird.

a) Geben Sie das Eingabealphabet, die Menge der Endzustände und den Startzustand des Automaten an.

Überführen Sie den Übergangsgraphen in eine Übergangstabelle.

(8 Punkte)

Name: _____

Der in Abbildung 1 dargestellte Automat 1 erfüllt zwei der für den Befehl `dir` festgelegten Bedingungen der Sprache der Optionen und Parameter nicht.

b) *Analysieren Sie den Automaten, indem Sie für die folgenden Eingaben die Zustandsfolgen darstellen.*
Geben Sie auch an, ob die Eingaben akzeptiert werden.

Erläutern Sie gegebenenfalls anhand Ihrer Ergebnisse, inwiefern diese mit den Bedingungen der oben dargestellten Sprache der Optionen und Parameter nicht übereinstimmen.

```
–Amusik
/var//inf/
–L/
–A–L/
```

Überführen Sie anschließend den Automaten in einen neuen deterministischen endlichen Automaten, der die beschriebenen Bedingungen für den Befehl `dir` *erfüllt.*

(17 Punkte)

c) *Entwerfen Sie eine reguläre Grammatik für die oben (Seite 1) dargestellte Sprache der Optionen und Parameter.*

Beschreiben Sie Ihr Vorgehen.

(16 Punkte)

Name: _____

d) Der in Abbildung 1 dargestellte Automat 1 enthält einige überflüssige Zustände. Der folgende Automat wurde entworfen, um diesen Automaten zu vereinfachen.

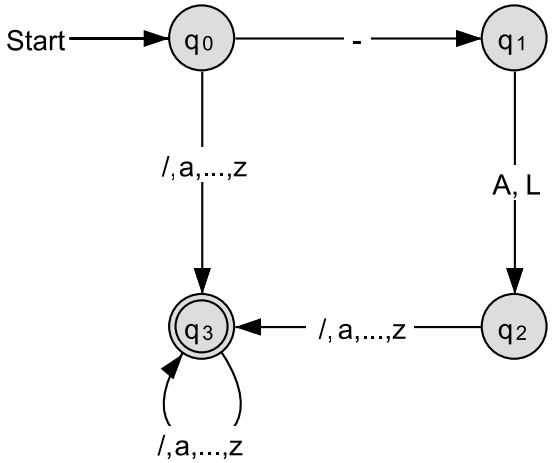


Abbildung 2: Automat 2

Nicht dargestellte Übergänge führen in einen Fehlerzustand, der im Graphen und in der Tabelle nicht dargestellt wird.

Begründen Sie, dass beide Automaten die gleiche Sprache erkennen.

(9 Punkte)

Zugelassene Hilfsmittel:

- Wörterbuch zur deutschen Rechtschreibung
- Taschenrechner

Unterlagen für die Lehrkraft

Abiturprüfung 2013

Informatik, Grundkurs

1. Aufgabenart

Aufgabenart	Aufgabenstellung aus dem Bereich endliche Automaten und formale Sprachen

2. Aufgabenstellung[1]

siehe Prüfungsaufgabe

3. Materialgrundlage

- entfällt

4. Bezüge zu den Vorgaben 2013

1. Inhaltliche Schwerpunkte
Endliche Automaten und formale Sprachen
- Modellieren kontextbezogener Problemstellungen als deterministische endliche Automaten
- Darstellung von deterministischen endlichen Automaten als Graph und als Tabelle
- Formale Sprachen: Reguläre Sprachen und ihre Grammatiken

2. Medien/Materialien
- entfällt

5. Zugelassene Hilfsmittel

- Wörterbuch zur deutschen Rechtschreibung
- Taschenrechner

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

6. Vorgaben für die Bewertung der Schülerleistungen

6.1 Modelllösung

Teilaufgabe a)

Eingabealphabet: $A = \{a, \ldots, z, /, -, A, L\}$

Endzustände: $E = \{ q_3, q_4, q_5, q_6\}$

Startzustand: q_0

Darstellung der Übergangsfunktion als Tabelle

	/	a,...,z	A	L	-
q_0	q_4	q_6			q_1
q_1			q_2	q_2	
q_2	q_3	q_3			
q_3	q_3	q_3			
q_4	q_5	q_5			
q_5	q_5	q_5			
q_6	q_6	q_6			

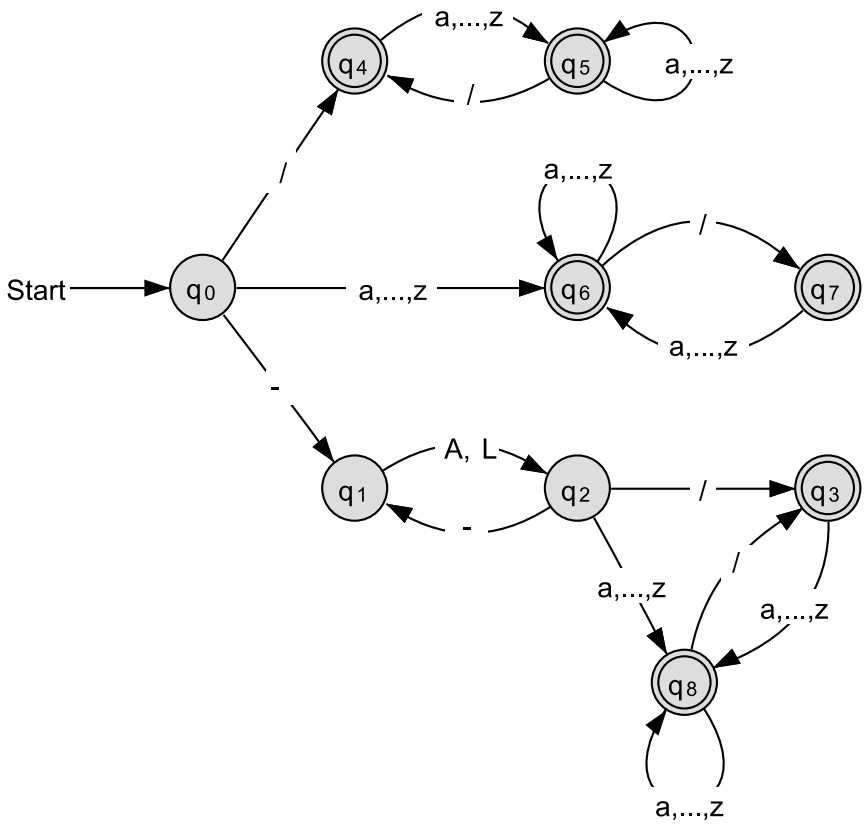
Teilaufgabe b)

`-Amusik`
$q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_3 \rightarrow q_3 \rightarrow q_3 \rightarrow q_3 \rightarrow q_3$
Das Wort wird korrekterweise akzeptiert.

`/var//inf/`
$q_0 \rightarrow q_4 \rightarrow q_5 \rightarrow q_5 \rightarrow q_5 \rightarrow q_5 \rightarrow q_5 \rightarrow q_5 \rightarrow q_5 \rightarrow q_5$
Das Wort wird fälschlicherweise akzeptiert. Hier wird der Bedingung nicht entsprochen, dass der Schrägstrich nicht zweimal direkt hintereinander auftreten darf. Solche Worte darf der Automat nicht akzeptieren.

`-L/`
$q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_3$
Das Wort wird korrekterweise akzeptiert.

`-A-L/`
$q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow$`Abbruch`
Das Wort wird fälschlicherweise nicht akzeptiert. Laut Vorgabe dürfen beliebig viele Optionen hintereinander aufgeführt werden, wobei jeder Buchstabe als Bezeichner einer Option als Präfix ein Minuszeichen erhält. Bei dem gegebenen Wort sind die beiden für `dir` zulässigen Optionen `A` und `L`, jeweils mit führendem Minuszeichen, hintereinander an der korrekten Position angegeben. Trotzdem gelangt der Automat in einen nicht definierten Zustand.



Dieser als Lösung angegebene Automat verwendet die ursprünglichen Zustände mit minimalen Änderungen. Ein vollkommen neuer Automat ist ebenso als korrekte Lösung zu werten, sofern er den Vorgaben entspricht.

Teilaufgabe c)

Terminale T = {a, … , z, /, –,A, L}
Nichtterminale N = {S, O, V, D}
Startsymbol: S

Produktionen (mögliche Lösung mit ε-Produktionen):

```
{S → -O | /D | aV | …  | zV,
 O → AS | LS,
 D → aV | …  | zV | ε,
 V → /D | aV | …  | zV | ε }
```

Produktionen (mögliche Lösung ohne ε-Produktionen):

```
{S → /  | a  | … | z  | -O | /D | aV | … | zV,
 O → AS | LS,
 D → a  | …  | z | aV | …  | zV,
 V → /  | a  | … | z  | /D | aV | …  | zV }
```

Zunächst müssen die Optionen erzeugt werden. Da beliebig viele Angaben der beiden Optionen A und L mit führenden Minuszeichen angegeben werden können, muss vom Startzustand aus ein Kreis angelegt werden, der jeweils eine der beiden Optionen anlegen kann. Da

Optionen auch entfallen können, muss der Kreislauf wieder zum Startsymbol führen. Da nach den Optionen auf jeden Fall ein Parameter folgen muss, gibt es in diesen Regeln stets ein Nichtterminal.

Beim Parameter ist zu beachten, dass ein Terminalzeichen nach dem Startsymbol für ein gültiges Wort ausreicht, so dass hier auch Regeln ohne Nichtterminale eingefügt werden müssen.

Nach dem Startsymbol ist eine fast beliebig lange Verkettung der Terminalen `a` bis `z` und `/` zu erzeugen. Zu beachten ist, dass der Schrägstrich nicht zweimal hintereinander vorkommen darf, weshalb ein weiteres Nichtterminal `D` eingeführt werden muss, welchem die gleichen Produktion von `V`, ohne die mit einem Schrägstrich, zugeordnet sind.

Teilaufgabe d)

Ein formaler Beweis ist nicht gefordert. Eine verständliche und vollständige Begründung ist ausreichend. Hierzu können vielfältige Ansätze gewählt werden.

Die Übergänge zwischen den Zuständen q_0 über q_1 und q_2 bis q_3 sind identisch geblieben. Außerdem ist q_0 weiterhin der Startzustand und q_0 bis q_2 sind weiterhin keine Endzustände. Dieser Teil des Automaten wurde nicht verändert.

Eine Änderung gibt es nur bei den Übergängen von q_0 zu q_3 durch den Schrägstrich und die Buchstaben. Der Automat 1 geht sofort jeweils in einen akzeptierenden Teil über, der beliebige weitere Kombinationen aus dem Schrägstrich und den Buchstaben akzeptiert. Bei diesem Automaten ist dieser Bereich nur in drei Unterbereiche mit den Zuständen q_4 und q_5, dem Zustand q_6 sowie dem Zustand q_3 aufgeteilt.

Der Automat 2 vereinigt diesen Bereich auf einen Zustand, der ebenfalls alle beliebigen Kombinationen aus Buchstaben und dem Schrägstrich akzeptiert.

6.2 Teilleistungen – Kriterien

Teilaufgabe a)

	Anforderungen **Der Prüfling**	**maximal erreichbare Punktzahl**
1	gibt das Eingabealphabet, die Menge der Endzustände und den Startzustand an.	3
2	überführt den Übergangsgraphen in eine Übergangstabelle.	5
Der gewählte Lösungsansatz und -weg muss nicht identisch mit dem der Modelllösung sein. Sachlich richtige Alternativen werden an dieser Stelle mit entsprechender Punktzahl bewertet.		

Teilaufgabe b)

	Anforderungen	**maximal erreichbare Punktzahl**
	Der Prüfling	
1	stellt die Zustandsfolgen dar.	4
2	gibt an, ob die Wörter akzeptiert werden.	2
3	erläutert für das zweite und vierte Wort den Widerspruch zu den Bedingungen.	4
4	überführt den Automaten.	7
Der gewählte Lösungsansatz und -weg muss nicht identisch mit dem der Modelllösung sein. Sachlich richtige Alternativen werden an dieser Stelle mit entsprechender Punktzahl bewertet.		

Teilaufgabe c)

	Anforderungen	**maximal erreichbare Punktzahl**
	Der Prüfling	
1	entwirft die Menge der Nichtterminale.	2
2	entwirft die Menge der Terminale.	2
3	entwirft die Menge der Produktionen.	6
4	beschreibt das Vorgehen.	6
Der gewählte Lösungsansatz und -weg muss nicht identisch mit dem der Modelllösung sein. Sachlich richtige Alternativen werden an dieser Stelle mit entsprechender Punktzahl bewertet.		

Teilaufgabe d)

	Anforderungen	**maximal erreichbare Punktzahl**
	Der Prüfling	
1	begründet die Entsprechung des Startzustands und der Endzustände.	3
2	begründet die Entsprechung der Übergänge.	6
Der gewählte Lösungsansatz und -weg muss nicht identisch mit dem der Modelllösung sein. Sachlich richtige Alternativen werden an dieser Stelle mit entsprechender Punktzahl bewertet.		

7. Bewertungsbogen zur Prüfungsarbeit

Name des Prüflings: _____ Kursbezeichnung: _____

Schule: _____

Teilaufgabe a)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK[2]**	**ZK**	**DK**
1	gibt das Eingabealphabet …	3			
2	überführt den Übergangsgraphen …	5			
	Summe Teilaufgabe a)	**8**			

Teilaufgabe b)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	stellt die Zustandsfolgen …	4			
2	gibt an, ob …	2			
3	erläutert für das …	4			
4	überführt den Automaten.	7			
	Summe Teilaufgabe b)	**17**			

Teilaufgabe c)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	entwirft die Menge …	2			
2	entwirft die Menge …	2			
3	entwirft die Menge …	6			
4	beschreibt das Vorgehen.	6			
	Summe Teilaufgabe c)	**16**			

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

Teilaufgabe d)

	Anforderungen	Lösungsqualität			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	begründet die Entsprechung …	3			
2	begründet die Entsprechung …	6			
	Summe Teilaufgabe d)	**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				
Note ggf. unter Absenkung um ein bis zwei Notenpunkte gemäß § 13 Abs. 2 APO-GOSt				
Paraphe				

ggf. arithmetisches Mittel der Punktsummen aus EK und ZK: _____

ggf. arithmetisches Mittel der Notenurteile aus EK und ZK: _____

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 – 39
mangelhaft plus	3	38 – 33
mangelhaft	2	32 – 27
mangelhaft minus	1	26 – 20
ungenügend	0	19 – 0

Name: _____

Abiturprüfung 2013

Informatik, Grundkurs

Aufgabenstellung:

Bei einer Quizshow stellt der Showmaster der Kandidatin bzw. dem Kandidaten eine Frage und gibt ihm oder ihr 4 mögliche Antworten (mit a, b, c und d bezeichnet). Nur eine dieser Antworten ist richtig. Die Kandidatin bzw. der Kandidat entscheidet sich für eine der Antworten. Der Showmaster entscheidet dann, ob die Antwort richtig war oder nicht. Varianten dieses Spiels sollen als Netzwerkspiel implementiert werden.

Beispiel:

Frage: Was baute Konrad Zuse?

Antwortmöglichkeiten:
a) Kaffeemaschine
b) Telefon
c) Computer
d) Auto

Richtig ist Antwort c.

In einer ersten Version arbeitet der Quiz-Server nach folgendem Protokoll:

Tabelle 1:

Client an Server	**Server an Client**
`Baut Verbindung auf`	`+OK Herzlich willkommen bei unserem Spiel` `<`*`sendet zufällig gewählte Frage mit vier Antwortmöglichkeiten`*`>`

`ANTWORT <a, b, c oder d>`	`+OK Richtige Antwort`
	`<`*`sendet zufällig gewählte Frage mit vier Antwortmöglichkeiten`*`>`
	oder
	`+OK Falsche Antwort`
	`<`*`sendet zufällig gewählte Frage mit vier Antwortmöglichkeiten`*`>`

`ENDE`	`+OK Vielen Dank für die Teilnahme an unserem Spiel` `<`*`beendet Verbindung`*`>`

Name: _____

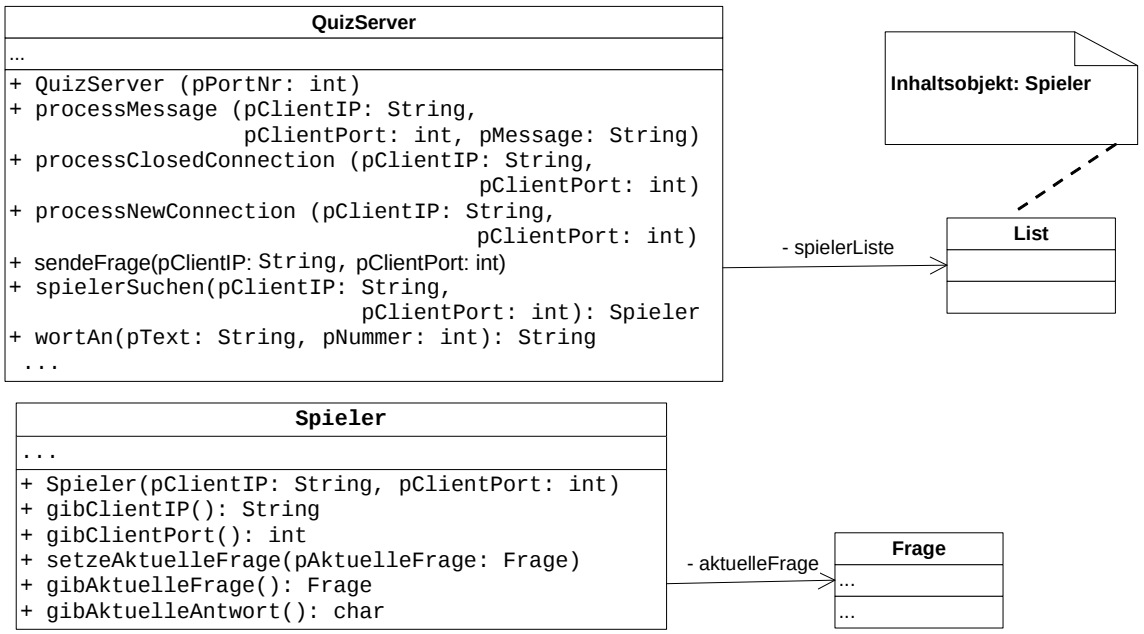
a) *Stellen Sie die Kommunikation zwischen Client und Server nach dem Protokoll aus Tabelle 1 in folgendem Fall dar: Der Client meldet sich an, erhält zufällig die Frage aus der Aufgabenstellung (Beispiel Seite1), antwortet falsch, erhält zufällig wieder dieselbe Frage, antwortet richtig und meldet sich dann wieder ab.*

Beschreiben Sie mögliche Eingabefehler und erweitern sie das Protokoll um eine Fehlerbehandlung für diese.

(10 Punkte)

Zur Implementierung des Spiels werden die Klassen `QuizServer`, `Spieler` und `Frage` zur Verfügung gestellt. Die Klasse `QuizServer` implementiert einen Server und verwaltet die Spieler in einer Liste. Hier sehen Sie einen Ausschnitt des Implementationsdiagramms.

Diagramm 1:



Für die Aufgabe sind nur die Klassen `QuizServer` und `Spieler` von Bedeutung.
Die Klasse `Frage` wird nicht weiter thematisiert. Sie finden Dokumentationen der für die Aufgabe wichtigen Teile im Anhang.

b) *Implementieren Sie die Methode* `processMessage` *der Klasse* `QuizServer` *gemäß dem in Tabelle 1 auf Seite 1 angegebenen Protokoll.*

Berücksichtigen Sie dabei die Dokumentation der Klasse `QuizServer` im Anhang. Eine Fehlerbehandlung ist nicht notwendig.

(10 Punkte)

Name: _____

c) In einer Verbesserung soll das Protokoll wie folgt verändert werden:

Client an Server	**Server an Client**
ENDE	+OK Vielen Dank für die Teilnahme an unserem Spiel gespielte Fragen: <Anzahl> davon richtig beantwortet:<Anzahl> <*beendet Verbindung*>

Erweitern Sie die Klasse `Spieler` *im Implementationsdiagramm (Diagramm 1 auf Seite 2) so, dass die neue Anweisung umgesetzt werden kann und erläutern Sie kurz, welche Änderungen in der Implementierung vorgenommen werden müssen.*

(14 Punkte)

d) Es soll in Zukunft sogenannten Administratoren des Spiels erlaubt sein, Fragen zu ergänzen. Daher muss man sich mit einem Namen und Passwort anmelden. Das Ergänzen von Fragen ist dann möglich, wenn man sich mit einem Namen angemeldet hat, der als Administrator geführt wird. Die Passwörter sollen natürlich verschlüsselt übergeben werden. Zur Verschlüsselung wird das Verfahren von Vigenère verwendet. In der Anlage finden Sie das Vigenère-Quadrat.

Bestimmen Sie die Chiffre des Wortes `ABITUR`*, wenn es mit dem Schlüssel* `ABCDEF` *verschlüsselt* wird.

(6 Punkte)

e) Der Schüler ALI hat seinen Namen als Passwort verwendet. Es wurde mit einem Wort der Länge 3 nach dem Verfahren von Vigènere verschlüsselt. Als Chiffre erhielt er GHJ.

Bestimmen Sie den Schlüssel und bestimmen Sie anschließend einen Schlüssel, der aus dem Klartext `ABI` *ebenfalls das Chiffre* `GHJ` *erzeugt.*

Beurteilen Sie die Sicherheit des Vigenère-Verfahrens, wenn der Schlüssel genauso lang ist wie der Text.

(10 Punkte)

Zugelassene Hilfsmittel:

- Wörterbuch zur deutschen Rechtschreibung
- Taschenrechner

Name: _____

Anlage: Das Vigenère-Quadrat:

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

Name: _____

Anhang

Die Klasse QuizServer

Die Klasse `QuizServer` implementiert einen Quiz-Server.

Konstruktor **`QuizServer(int pPortNr)`**

Nach dem Aufruf dieses Konstruktors bietet der Quiz-Server seinen Dienst über die Portnummer `pPortNr` an. Clients können sich nun mit dem Server verbinden. Eine Liste mit Fragen und eine (leere) Spielerliste wurden erzeugt.

Auftrag **`void processMessage(String pClientIP, int pClientPort, String pMessage)`**

Der Client mit der angegebenen IP und der angegebenen Portnummer hat dem Server eine Nachricht gesendet. Dieser reagiert gemäß dem Protokoll in der Aufgabenstellung.

Auftrag **`void processClosedConnection(String pClientIP, int pClientPort)`**

Der Client mit der IP-Adresse `pClientIP` und der Portnummer `pClientPort` wird gemäß Protokoll verabschiedet und aus der Spielerliste gelöscht.

Auftrag **`void processNewConnection(String pClientIP, int pClientPort)`**

Der Client mit der IP-Adresse `pClientIP` und der Portnummer `pClientPort` hat eine Verbindung zum Server aufgebaut. Der Server begrüßt den neuen Teilnehmer, nimmt ihn in die Spielerliste auf, bestimmt die erste Frage und sendet sie.

Auftrag **`void sendeFrage(String pClientIP, int pClientPort)`**

Eine Frage wird zufällig aus der Liste ausgewählt und an den Client mit der angegebenen IP-Adresse und der angegebenen Portnummer gesendet.

Anfrage **`Spieler spielerSuchen(String pClientIP, int pClientPort)`**

Diese Anfrage liefert den Spieler mit der IP-Adresse `pClientIP` und der Portnummer `pClientPort`. Gibt es keinen solchen Spieler, so liefert die Methode `null`.

Anfrage **`String wortAn(String pText, int pNummer)`**

Diese Anfrage liefert das Wort mit der angegebenen Wortnummer in dem angegebenen Text. Die Wörter sind dabei beginnend mit 1 nummeriert und durch Leerzeichen voneinander getrennt. Falls kein Wort an der angegebenen Wortnummer existiert, wird `null` zurückgegeben.

Name: _____

Die Klasse Spieler

In der Klasse **`Spieler`** wird die IP-Adresse, die Portnummer und die aktuelle Frage des Spielers gespeichert.

Konstruktor **`Spieler (String pClientIP, int pClientPort)`**

Nach dem Aufruf dieses Konstruktors wurden die in den Parametern übergebenen Daten in entsprechenden Zustandsvariablen gespeichert. Die aktuelle Frage des Spielers ist `null`.

Anfrage **`String gibClientIP()`**

Die IP-Adresse des Spielers wird zurückgegeben.

Anfrage **`int gibClientPort()`**

Die Portnummer des Spielers wird zurückgegeben.

Auftrag **`void setzeAktuelleFrage(Frage pAktuelleFrage)`**

Die aktuelle Frage des Spielers wird neu gesetzt.

Anfrage **`Frage gibAktuelleFrage()`**

Die aktuelle Frage des Spielers wird zurückgeliefert. Gibt es keine aktuelle Frage, so ist der Rückgabewert `null`.

Anfrage **`char gibAktuelleAntwort()`**

Die Nummer (als Buchstabe) der richtigen Antwort auf die aktuelle Frage des Spielers wird zurückgegeben.

Name: _____

Die Klasse Server

Über die Klasse **`Server`** ist es möglich, eigene Serverdienste anzubieten, so dass Clients Verbindungen gemäß dem TCP/IP-Protokoll hierzu aufbauen können. Nachrichten werden grundsätzlich zeilenweise verarbeitet, d. h., beim Senden einer Zeichenkette wird ein Zeilentrenner ergänzt und beim Empfangen wird er entfernt.

Verbindungsaufbau, Nachrichtenempfang und Verbindungsende geschehen nebenläufig. Durch Überschreiben der entsprechenden Methoden kann der Server auf diese Ereignisse reagieren.

Eine Fehlerbehandlung ist in dieser Klasse aus Gründen der Vereinfachung nicht vorgesehen.

Dokumentation der Klasse Server

Konstruktor **`Server(int pPortNr)`**

Nach dem Aufruf dieses Konstruktors bietet ein Server seinen Dienst über die angegebene Portnummer an. Clients können sich nun mit dem Server verbinden.

Auftrag **`void closeConnection(String pClientIP, int pClientPort)`**

Unter der Voraussetzung, dass eine Verbindung mit dem angegebenen Client existiert, wird diese beendet. Der Server sendet sich die Nachricht `processClosedConnection`.

Auftrag **`void processClosedConnection(String pClientIP, int pClientPort)`**

Diese Methode ohne Anweisungen wird aufgerufen, bevor der Server die Verbindung zu dem in der Parameterliste spezifizierten Client schließt. Durch das Überschreiben in Unterklassen kann auf die Schließung der Verbindung zum angegebenen Client reagiert werden.

Auftrag **`void processMessage(String pClientIP, int pClientPort, String pMessage)`**

Der Client mit der angegebenen IP und der angegebenen Portnummer hat dem Server eine Nachricht gesendet. Dieser ruft daraufhin diese Methode ohne Anweisungen auf. Durch das Überschreiben in Unterklassen kann auf diese Nachricht des angegebenen Client reagiert werden.

Name: _____

Auftrag `void processNewConnection(String pClientIP, int pClientPort)`

Der Client mit der angegebenen IP-Adresse und der angegebenen Portnummer hat eine Verbindung zum Server aufgebaut. Der Server hat daraufhin diese Methode aufgerufen, die in dieser Klasse keine Anweisungen enthält. Durch das Überschreiben in Unterklassen kann auf diesen Neuaufbau einer Verbindung von dem angegebenen Client zum Server reagiert werden.

Auftrag `void send(String pClientIP, int pClientPort, String pMessage)`

Wenn eine Verbindung zum angegebenen Client besteht, dann wird diesem Client die angegebene Nachricht – um einen Zeilentrenner erweitert – gesendet.

Auftrag `void sendToAll(String pMessage)`

Die angegebene Nachricht wird – um einen Zeilentrenner erweitert – an alle verbundenen Clients gesendet.

Auftrag `void close()`

Alle bestehenden Verbindungen werden getrennt. Der Server kann nicht mehr verwendet werden.

Name: _____

Die Klasse `List`

Objekte der Klasse **`List`** verwalten beliebig viele, linear angeordnete Objekte. 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 oder ein Listenobjekt an das Ende der Liste angefügt werden.

Dokumentation der Klasse `List`

Konstruktor **`List()`**
Eine leere Liste wird erzeugt.

Anfrage **`boolean isEmpty()`**
Die Anfrage liefert den Wert `true`, wenn die Liste keine Objekte enthält, sonst liefert sie den Wert `false`.

Anfrage **`boolean hasAccess()`**
Die Anfrage liefert den Wert `true`, wenn es ein aktuelles Objekt gibt, sonst liefert sie den Wert `false`.

Auftrag **`void next()`**
Falls die Liste nicht leer ist, es ein aktuelles Objekt gibt und dieses nicht das letzte Objekt der Liste ist, wird das dem aktuellen Objekt in der Liste folgende Objekt zum aktuellen Objekt, andernfalls gibt es nach Ausführung des Auftrags kein aktuelles Objekt, 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 **`Object getObject()`**

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 setObject(Object pObject)`**

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

Auftrag **`void append(Object pObject)`**

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

Auftrag **`void insert(Object pObject)`**

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

Auftrag **`void concat(List pList)`**

Die Liste `pList` wird an die Liste angehängt. Anschließend wird `pList` eine leere Liste. Das aktuelle Objekt bleibt unverändert. Falls `pList` `null` oder eine leere Liste ist, bleibt die Liste unverändert.

Auftrag **`void remove()`**

Falls es ein aktuelles Objekt gibt (`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 2013

Informatik, Grundkurs

1. Aufgabenart

Aufgabenart	Aufgabenstellungen aus dem Bereich Client-Server-Strukturen
Syntaxvariante	Java

2. Aufgabenstellung[1]

siehe Prüfungsaufgabe

3. Materialgrundlage

- entfällt

4. Bezüge zu den Vorgaben 2013

1. *Inhaltliche Schwerpunkte*
   Objektorientiertes Modellieren und Implementieren von kontextbezogenen Anwendungen
   - Modellieren und Implementieren kontextbezogener Problemstellungen als Netzwerkanwendungen
     - Netzwerkprotokolle
     - Client-Server-Anwendungen
       Kryptografie: Symmetrische Verschlüsselungsverfahren (Caesar, Vigenère)

2. *Medien/Materialien*
   - entfällt

5. Zugelassene Hilfsmittel

- Wörterbuch zur deutschen Rechtschreibung
- Taschenrechner

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

6. Vorgaben für die Bewertung der Schülerleistungen

6.1 Modelllösung

Teilaufgabe a)

Die Kommunikation in dem Beispiel:

Client meldet sich an.
Server: +OK Herzlich willkommen bei unserem Spiel.
Server: **Frage:** Was baute Konrad Zuse?
Server: **Antwortmöglichkeiten:** a) Kaffeemaschine; b) Telefon; c) Computer; d) Auto;
Client: ANTWORT b
Server: +OK Falsche Antwort
Server: **Frage:** Was baute Konrad Zuse?
Server: **Antwortmöglichkeiten:** a) Kaffeemaschine; b) Telefon; c) Computer; d) Auto;
Client: ANTWORT c
Server: +OK Richtige Antwort
<Server sendet weitere Frage mit Antwortmöglichkeiten>
Client: ENDE
Server: +OK Vielen Dank für die Teilnahme an unserem Spiel.
Server beendet Verbindung.

Da in der Aufgabe keine Vorgaben gemacht sind, wird auch eine andere Darstellung der vom Server gesendeten Frage und der möglichen Antworten als richtig gewertet.

Die Fehlerbehandlung könnte wie folgt aussehen:

Client an Server	Server an Client
<Jede nicht im Protokoll aufgeführte Anweisung>	-ERR Fehler: Anweisung nicht im Protokoll.
`ANTWORT <Andere Antwort als a, b, c oder d>`	-ERR Fehler: Es sind nur die Antworten a, b, c oder d möglich.

Teilaufgabe b)

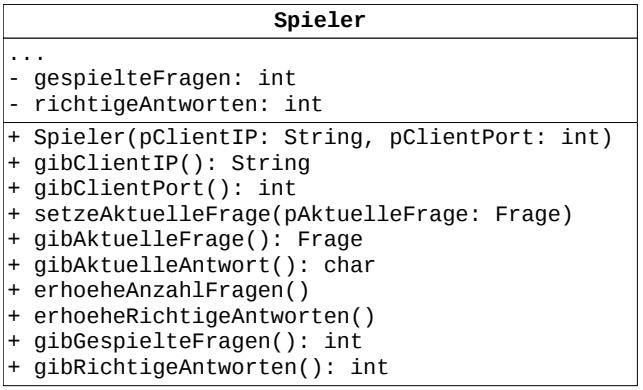
Eine mögliche Lösung:

```
public void processMessage(String pClientIP, int pClientPort,
                                                    String pMessage){
  Spieler derSpieler;
  String anweisung = wortAn(pMessage, 1);
  if (anweisung.equals("ENDE")){
    closeConnection(pClientIP, pClientPort);
  }
  else if (anweisung.equals("ANTWORT")) {
    derSpieler = spielerSuchen(pClientIP, pClientPort);
    if (pMessage.charAt(pMessage.length()-1) ==
                                  derSpieler.gibAktuelleAntwort()){
      send(pClientIP, pClientPort, "+OK Richtige Antwort");
    }
    else {
      send(pClientIP, pClientPort, "+OK Falsche Antwort");
    }
    sendeFrage(pClientIP, pClientPort);
  }
}
```

Es sind hier viele andere Lösungen möglich. In dieser Lösung wird davon ausgegangen, dass die Reaktion auf die Protokollanweisung `ENDE`, wie in der Dokumentation gefordert, in der Methode `processClosedConnection` erfolgt. Wenn der Prüfling dies in `processMessage` durchführt, ist das als richtig zu werten.

Teilaufgabe c)

Eine mögliche Lösung: Die Klasse `Spieler` wird um zwei Attribute ergänzt: Eines, um sich die Anzahl der gespielten Fragen zu merken, und eines, um die Anzahl der richtigen Antworten zu speichern. Entsprechend werden zwei Methoden zur Abfrage der Werte und zwei Methoden zum Erhöhen der Werte (um 1) ergänzt. Im Konstruktor wird der Anfangswert der Attribute auf Null gesetzt. Es ergibt sich folgendes Klassendiagramm:



Die Attributswerte müsssen an geeigneter Stelle erhöht werden. Man könnte bei einer richtigen Antwort beide Attribute in `processMessage` erhöhen und bei einer falschen Antwort nur die gespielten Fragen.

Das Ergebnis wird dann in `processClosedConnection` gesendet.

Teilaufgabe d)

Die Chiffre ist `ACKWYW`.

Teilaufgabe e)

Der Schlüssel `GWB` verschlüsselt `ALI` zu `GHJ`.

Der Schlüssel `GGB` verschlüsselt `ABI` zu `GHJ`.

Zur Sicherheit: Die Sicherheit des Verfahrens hängt, wenn der Schlüssel genauso lang ist wie der Text, nur noch von der Geheimhaltung des Schlüssels ab, da man mit anderen Schlüsseln jeden Klartext derselben Länge erzeugen kann, wie man an dem Beispiel sieht.

6.2 Teilleistungen – Kriterien

Teilaufgabe a)

	Anforderungen	**maximal erreichbare Punktzahl**
	Der Prüfling	
1	stellt die Kommunikation zwischen Client und Server richtig dar.	6
2	beschreibt zwei mögliche Eingabefehler und erweitert das Protokoll entsprechend.	4
Der gewählte Lösungsansatz und -weg muss nicht identisch mit dem der Modelllösung sein. Sachlich richtige Alternativen werden an dieser Stelle mit entsprechender Punktzahl bewertet.		

Teilaufgabe b)

	Anforderungen	**maximal erreichbare Punktzahl**
	Der Prüfling	
1	implementiert die Fallunterscheidung nach den Anweisungen im Protokoll.	5
2	implementiert die Fallunterscheidung nach richtiger und falscher Antwort.	5
Der gewählte Lösungsansatz und -weg muss nicht identisch mit dem der Modelllösung sein. Sachlich richtige Alternativen werden an dieser Stelle mit entsprechender Punktzahl bewertet.		

Teilaufgabe c)

	Anforderungen	**maximal erreichbare Punktzahl**
	Der Prüfling	
1	erweitert die Klasse `Spieler` geeignet.	8
2	erläutert, welche Änderungen in der Implementierung vorgenommen werden müssen.	6
Der gewählte Lösungsansatz und -weg muss nicht identisch mit dem der Modelllösung sein. Sachlich richtige Alternativen werden an dieser Stelle mit entsprechender Punktzahl bewertet.		

Teilaufgabe d)

	Anforderungen	**maximal erreichbare Punktzahl (AFB)**
	Der Prüfling	
1	bestimmt die Chiffre des Wortes `ABITUR`, wenn es mit dem Schlüssel `ABCDEF` verschlüsselt wird.	6
Der gewählte Lösungsansatz und -weg muss nicht identisch mit dem der Modelllösung sein. Sachlich richtige Alternativen werden an dieser Stelle mit entsprechender Punktzahl bewertet.		

Teilaufgabe e)

	Anforderungen	**maximal erreichbare Punktzahl**
	Der Prüfling	
1	bestimmt den Schlüssel.	3
2	bestimmt einen Schlüssel, der aus dem Klartext `ABI` ebenfalls die Chiffre `GHJ` erzeugt.	3
3	beurteilt die Sicherheit des Vigenère-Verfahrens, wenn der Schlüssel genauso lang ist, wie der Text.	4
Der gewählte Lösungsansatz und -weg muss nicht identisch mit dem der Modelllösung sein. Sachlich richtige Alternativen werden an dieser Stelle mit entsprechender Punktzahl bewertet.		

7. Bewertungsbogen zur Prüfungsarbeit

Name des Prüflings: _____ Kursbezeichnung: _____

Schule: _____

Teilaufgabe a)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK[2]**	**ZK**	**DK**
1	stellt die Kommunikation …	6			
2	beschreibt zwei mögliche …	4			
	Summe Teilaufgabe a)	**10**			

Teilaufgabe b)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	implementiert die Fallunterscheidung …	5			
2	implementiert die Fallunterscheidung …	5			
	Summe Teilaufgabe b)	**10**			

Teilaufgabe c)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	erweitert die Klasse …	8			
2	erläutert, welche Änderungen …	6			
	Summe Teilaufgabe c)	**14**			

Teilaufgabe d)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	bestimmt die Chiffre …	6			
	Summe Teilaufgabe d)	**6**			

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

Teilaufgabe e)

	Anforderungen	Lösungsqualität			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	bestimmt den Schlüssel.	3			
2	bestimmt einen Schlüssel …	3			
3	beurteilt die Sicherheit …	4			
	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				
Note ggf. unter Absenkung um ein bis zwei Notenpunkte gemäß § 13 Abs. 2 APO-GOSt				
Paraphe				

ggf. arithmetisches Mittel der Punktsummen aus EK und ZK: _____

ggf. arithmetisches Mittel der Notenurteile aus EK und ZK: _____

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 – 39
mangelhaft plus	3	38 – 33
mangelhaft	2	32 – 27
mangelhaft minus	1	26 – 20
ungenügend	0	19 – 0

Name: _____

Abiturprüfung 2013

Informatik, Grundkurs

Aufgabenstellung:

Die Abiturienten der verschiedenen Schulen eines Landkreises planen ein gemeinsames soziales Netzwerk für Abiturjahrgänge, das auch nach dem Abitur den Kontakt zwischen den Mitgliedern des Netzwerkes aufrecht halten soll.

Das Netzwerk soll mit Hilfe einer Datenbank verwaltet werden.

a) Für einen ersten Entwurf wurde der folgende Anforderungskatalog an die Datenbank aufgestellt:

- Von den einzelnen Schulen werden der Name und der Ort der Schule gespeichert.
- Von den Mitgliedern werden der Name und die von ihnen besuchte Schule gespeichert. Außerdem wird das Jahr, in dem sie ihr Abitur machen, festgehalten.
- Bei der Registrierung werden für jedes Mitglied ein Anmeldename und ein Passwort festgelegt.
- Mitglieder des Netzwerks können mit anderen Mitgliedern befreundet sein. Diese Freundschaft soll gespeichert werden.
- Jeder Teilnehmer kann Nachrichten verfassen. Andere Teilnehmer können von einer Nachricht festhalten, dass sie ihnen gefällt.

Modellieren Sie die zu erstellende Datenbank im ER-Modell. Geben Sie dazu das ER-Diagramm an, aus dem die Entitätstypen mit den zugehörigen Attributen und ihre Beziehungstypen einschließlich der Kardinalitäten hervorgehen.

Übersetzen Sie das Modell in entsprechende Relationenschemata.

(10 Punkte)

Name: _____

Nach gründlicher Überarbeitung des ersten Entwurfs wurden die Möglichkeiten im Umgang mit Nachrichten erweitert: Es soll nun möglich sein, eine Nachricht mit einer Note von 1 bis 6 zu bewerten. Zur Sicherheit muss jedes Mitglied hier eine eindeutige Mailadresse angeben. Außerdem kann man zu einer Nachricht Kommentare verfassen. Im Folgenden sind die für die Verwaltung von Nachrichten wichtigen Relationen angegeben.

```
Mitglied (idMitglied, Name, Schulname, Schulort,
                                  Abijahrgang, Anmeldename, Passwort)
Nachricht (idNachricht, ↑idMitglied, NachrichtText)
bewertet (↑idMitglied, ↑idNachricht, MailAdresse, Note)
Kommentar (idKommentar, ↑idNachricht, ↑idMitglied, KommentarText)
```

b) In Anlage 1 sind Beispieltabellen zu den obigen Relationen angegeben.

Wenden Sie die folgenden SQL-Anweisungen auf die in der Anlage 1 angegebenen Datensätze an und notieren Sie das Ergebnis der Abfragen in einer Tabelle.

Erläutern Sie die SQL-Anweisungen.

1.
```
SELECT Mitglied.Name
FROM Mitglied
WHERE Abijahrgang = 2013
```

2.
```
SELECT Mitglied.Name, Kommentar.idNachricht
FROM Mitglied, Kommentar, bewertet
WHERE Kommentar.idNachricht = bewertet.idNachricht
      AND Mitglied.idMitglied = Kommentar.idMitglied
      AND Kommentar.idMitglied = bewertet.idMitglied
```

3.
```
SELECT Mitglied.Name, COUNT(Nachricht.idNachricht) AS Anzahl
FROM Mitglied, Nachricht
WHERE Mitglied.idMitglied = Nachricht.idMitglied
      AND Mitglied.Schulname = "Alan-Turing-Gesamtschule"
GROUP BY Mitglied.idMitglied
ORDER BY Mitglied.Name ASC
```

(12 Punkte)

Name: _____

c) *Entwickeln Sie die zugehörigen SQL-Anweisungen, um folgende Abfragen durchzuführen:*

1. Es soll eine Tabelle aller Nachrichtentexte von Lea Braun, die die Mitgliedernummer 3 hat, erstellt werden.
2. Als Gesamtbewertung einer Nachricht wird der Durchschnitt aller Bewertungen genommen. Es soll die Gesamtbewertung der Nachricht mit der Nummer 1 ermittelt werden.
3. Es soll eine Tabelle erstellt werden, in der alle Mitglieder durch ihre Mitgliedsnummer aufgeführt sind, die schon einmal die Nachricht eines anderen Mitgliedes, das nicht im selben Ort zur Schule geht, kommentiert haben. Die Tabelle soll nach der Mitgliedsnummer aufsteigend sortiert werden und außerdem keine doppelten Einträge enthalten.

(16 Punkte)

d) Durch die Normalisierung von Datenbanken werden Redundanzen und Anomalien vermieden.

Geben Sie die Bedingungen für die 1., 2. und 3. Normalform an.

Erläutern Sie für die vor Aufgabenteil b) gegebenen Relationenschemata, welche der Bedingungen für die einzelnen Normalformen verletzt werden. Berücksichtigen Sie ggf. die in Anlage 2 aufgeführten Abhängigkeiten.

Überführen Sie die gegebenen Relationenschemata schrittweise in die 3. Normalform und begründen Sie Ihr Vorgehen.

(12 Punkte)

Zugelassene Hilfsmittel:

- Wörterbuch zur deutschen Rechtschreibung
- Taschenrechner

Name: _____

Anlage 1: Beispieltabellen zu Aufgabenteil b)

Mitglied						
idMitglied	Name	Schulname	Schulort	Abijahrgang	Anmeldename	Passwort
1	Dirk Meier	Konrad-Zuse-Gymnasium	Aburg	2013	dirk	bca!314
2	Thinh Phu	Carl-Petri-Gymnasium	Bedorf	2014	thinh	cba?159
3	Lea Braun	Alan-Turing-Gesamtschule	Cefurt	2013	lea	bac$265
4	Gamze Ergin	Alan-Turing-Gesamtschule	Cefurt	2013	gamze	cab&358
5	Arndt Schmidt	von-Neumann-Gesamtschule	Aburg	2014	andy	acb(979

Nachricht		
idNachricht	↑idMitglied	NachrichtText
1	3	Bald habe ich mein Abi!
2	4	Schule ist doof.
3	3	Ihr seid alle toll!
4	1	Wer von euch kommt zur Party?

bewertet			
↑idMitglied	↑idNachricht	MailAdresse	Note
1	2	dmeier@mail.de	5
1	1	dmeier@mail.de	3
3	1	lea@mail.de	1
4	1	ge13@mail.de	7

Kommentar			
idKommentar	↑idNachricht	↑idMitglied	KommentarText
1	2	3	Bitte hier nicht lästern!
2	2	1	Das finde ich gar nicht.

Name: _____

Anlage 2: Abhängigkeiten zu Aufgabenteil d)

Die Abhängigkeiten geben an, ob Attribute voneinander abhängig sind. So wird durch die Beziehung `Schulname → Schulort` festgelegt, dass der Ort einer Schule vom Namen der Schule abhängt.

Im Aufgabenteil d) werden die folgenden Abhängigkeiten angenommen:

```
Schulname → Schulort

idMitglied → Nachname, Vorname, Mailadresse, Schulname,
                       Schulort, Abijahrgang, Anmeldename, Passwort

idNachricht → NachrichtText

idKommentar → KommentarText
```

Unterlagen für die Lehrkraft

Abiturprüfung 2013

Informatik, Grundkurs

1. Aufgabenart

Aufgabenart	Aufgabenstellungen aus dem Bereich Relationale Datenbanken
Syntaxvariante	–

2. Aufgabenstellung[1]

siehe Prüfungsaufgabe

3. Materialgrundlage

- entfällt

4. Bezüge zu den Vorgaben 2013

1. *Inhaltliche Schwerpunkte*
   Relationale Datenbanken
   - Normalisierung: Überführung einer Datenbank in die 1. bis 3. Normalform
   - Relationenalgebra (Selektion, Projektion, Join)
   - SQL-Abfragen über eine und mehrere verknüpfte Tabellen
   - Datenschutzaspekte

2. *Medien/Materialien*
   - entfällt

5. Zugelassene Hilfsmittel

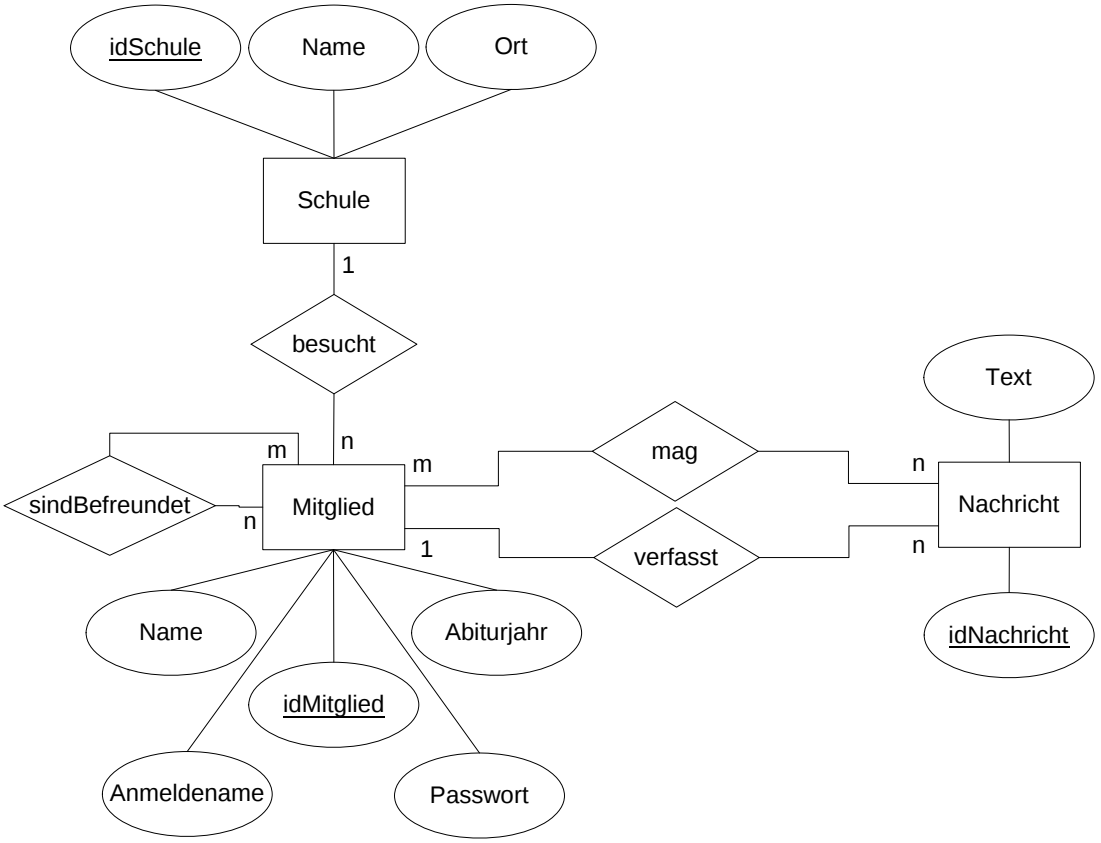
- Wörterbuch zur deutschen Rechtschreibung
- Taschenrechner

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

6. Vorgaben für die Bewertung der Schülerleistungen

6.1 Modelllösung

Teilaufgabe a)




```
Mitglied (idMitglied, Name, Abiturjahr,
                                  Anmeldename, Passwort, ↑idSchule)
sindBefreundet (↑idMitglied, ↑idMitglied)
Schule (idSchule, Name, Ort)
Nachricht (idNachricht, ↑idMitglied, Text)
mag (↑idMitglied, ↑idNachricht)
```

Teilaufgabe b)

1.

Name
Dirk Meier
Lea Braun
Gamze Ergin

Es werden alle Schüler des Abiturjahrgangs 2013 selektiert und anschließend auf den Namen projiziert.

Es wird eine Liste aller Mitglieder, die im Jahr 2013 ihr Abitur ablegen, erstellt.

2.

Mitglied.Name	Kommentar.idNachricht
Dirk Meier	2

Zunächst werden in den Tabellen `Mitglied`, `Kommentar` und `bewertet` alle Datensätze miteinander verbunden und selektiert, bei denen der Kommentar und die Bewertung sich auf die gleiche Nachricht beziehen und bei denen der Kommentar und die Bewertung von dem gleichen Mitglied stammen. Anschließend wird auf den Namen des Mitgliedes und die Nummer der Nachricht projiziert.

Es wird eine Liste aller Mitglieder, die eine Nachricht sowohl bewertet als auch kommentiert haben, zusammen mit der Nummer der Nachricht erstellt.

3.

Mitglied.Name	Anzahl
Gamze Ergin	1
Lea Braun	2

In den Tabellen `Mitglied` und `Nachricht` werden alle Datensätze verbunden und selektiert, die vom gleichen Mitglied stammen und bei denen das Mitglied die Alan-Turing-Gesamtschule besucht.

Die Datensätze werden nach der Mitgliedsnummer gruppiert und alphabetisch aufsteigend nach dem Namen sortiert. Es wird die Anzahl der Nachrichten eines jeden Mitgliedes gezählt. Es wird eine alphabetisch aufsteigend sortierte Liste der Schülerinnen und Schüler aus der Alan-Turing-Gesamtschule erstellt, die bereits wenigstens eine Nachricht geschrieben haben. In der zweiten Spalte steht die Anzahl der Nachrichten.

Teilaufgabe c)

1.

```
SELECT Nachricht.NachrichtText
FROM Nachricht
WHERE Nachricht.idMitglied = 3
```

2.

```
SELECT SUM(bewertet.Note) / COUNT(bewertet.idMitglied)
FROM bewertet
WHERE bewertet.idNachricht = 1
```

3.

```
SELECT DISTINCT Mitglied.idMitglied
FROM Mitglied,
     (select Mitglied.Schulort, Kommentar.idMitglied AS idKom
      FROM Mitglied, Nachricht, Kommentar
      WHERE Mitglied.idMitglied = Nachricht.idMitglied
              and Nachricht.idNachricht = Kommentar.idNachricht) AS
                                                                  tmp
WHERE Mitglied.Schulort <> tmp.Schulort
      AND Mitglied.idMitglied = idKom
ORDER BY Mitglied.idMitglied
```

Teilaufgabe d)

1. Normalform

Eine Tabelle liegt in der ersten Normalform (1NF) vor, wenn jeder Attributwert eine atomare, nicht weiter zerlegbare Dateneinheit ist. Eine Tabelle ist nicht in der 1NF, wenn Attribute mehrfach oder komplex in einer Spalte auftreten.

Das Attribut `Name` ist nicht atomar, denn es lässt sich in die Dateneinheiten `Nachname` und `Vorname` zerlegen.

Um das Relationenschema in 1NF zu überführen, wird das Attribut `Name` durch die oben genannten Dateneinheiten ersetzt.

Die Überführung in die erste Normalform lautet deshalb wie folgt:

```
Mitglied (idMitglied, Nachname, Vorname, Schulname, Schulort,
                                Abijahrgang, Anmeldename, Passwort)
Nachricht (idNachricht, ↑idMitglied, NachrichtText)
bewertet (↑idMitglied, ↑idNachricht, MailAdresse, Note)
Kommentar (idKommentar, ↑idNachricht, ↑idMitglied, KommentarText)
```

2. Normalform

Eine Tabelle liegt in der zweiten Normalform (2NF) vor, wenn sie in der 1NF ist und jedes Nichtschlüsselattribut voll funktional abhängig vom Primärschlüssel ist. Eine Tabelle ist nicht in der 2NF, wenn Attribute von einem Teil des Schlüssels eindeutig identifiziert werden.

In dem Relationenschema hängt das Attribut MailAdresse nur vom Teilschlüssel `idMitglied` ab.

Um das Relationenschema in 2NF zu überführen, wird das Attribut `Mailadresse` in das Relationenschema `Mitglied` eingefügt.

Die Überführung in die zweite Normalform lautet wie folgt:

```
Mitglied (idMitglied, Nachname, Vorname, MailAdresse,
          Schulname, Schulort, Abijahrgang, Anmeldename, Passwort)
Nachricht (idNachricht, ↑idMitglied, NachrichtText)
bewertet (↑idMitglied, ↑idNachricht, Note)
Kommentar (idKommentar, ↑idNachricht, ↑idMitglied, KommentarText)
```

3. Normalform

Eine Tabelle liegt in der dritten Normalform (3NF) vor, wenn sie sich in der 2NF befindet und jedes Nichtschlüsselattribut nicht transitiv abhängig vom Primärschlüssel ist. Eine Tabelle ist nicht in der 3NF, wenn Attribute von anderen Nicht-Schlüsselattributen identifiziert werden.

In der Relation `Mitglied` hängt (nach Anlage 2) das Attribut `Schulort` vom Nichtschlüsselattribut `Schulname` ab.

Um die Relationenschemata in 3NF zu überführen, wird das neue Relationenschema `Schule` eingeführt. Die durch die neuen Relationenschemata aufgefangenen Attribute werden im Relationenschema `Mitglied` durch den entsprechenden Fremdschlüssel ersetzt.

Die Überführung in die dritte Normalform lautet wie folgt:

```
Schule (idSchule, Schulname, Schulort)
Mitglied (idMitglied, Nachname, Vorname, MailAdresse,
                      idSchule, Abijahrgang, Anmeldename, Passwort)
Nachricht (idNachricht, ↑idMitglied, NachrichtText)
bewertet (↑idMitglied, ↑idNachricht, Note)
Kommentar (idKommentar, ↑idNachricht, ↑idMitglied, KommentarText)
```

6.2 Teilleistungen – Kriterien

Teilaufgabe a)

	Anforderungen	**maximal erreichbare Punktzahl**
	Der Prüfling	
1	stellt die Entitäten und Beziehungen in einem ER-Diagramm dar.	4
2	bestimmt die Kardinalitäten.	2
3	übersetzt das Modell in entsprechende Relationenschemata.	4
Der gewählte Lösungsansatz und -weg muss nicht identisch mit dem der Modelllösung sein. Sachlich richtige Alternativen werden an dieser Stelle mit entsprechender Punktzahl bewertet.		

Teilaufgabe b)

	Anforderungen	**maximal erreichbare Punktzahl**
	Der Prüfling	
1	wendet die erste SQL-Anweisung auf die Daten an, notiert das Ergebnis und erläutert die Anweisung.	2
2	wendet die zweite SQL-Anweisung auf die Daten an, notiert das Ergebnis und erläutert die Anweisung.	4
3	wendet die dritte SQL-Anweisung auf die Daten an, notiert das Ergebnis und erläutert die Anweisung..	6
Der gewählte Lösungsansatz und -weg muss nicht identisch mit dem der Modelllösung sein. Sachlich richtige Alternativen werden an dieser Stelle mit entsprechender Punktzahl bewertet.		

Teilaufgabe c)

	Anforderungen	**maximal erreichbare Punktzahl**
	Der Prüfling	
1	entwickelt eine SQL-Anweisung zur ersten Abfrage.	5
2	entwickelt eine SQL-Anweisung zur zweiten Abfrage.	5
3	entwickelt eine SQL-Anweisung zur dritten Abfrage.	6
Der gewählte Lösungsansatz und -weg muss nicht identisch mit dem der Modelllösung sein. Sachlich richtige Alternativen werden an dieser Stelle mit entsprechender Punktzahl bewertet.		

Teilaufgabe d)

	Anforderungen	**maximal erreichbare Punktzahl**
	Der Prüfling	
1	gibt die Bedingungen für die erste, zweite und dritte Normalform an.	6
2	erläutert, dass die Bedingungen für die Normalformen verletzt sind.	3
3	überführt das Relationenschema in die 3. Normalform und begründet das Vorgehen.	3
Der gewählte Lösungsansatz und -weg muss nicht identisch mit dem der Modelllösung sein. Sachlich richtige Alternativen werden an dieser Stelle mit entsprechender Punktzahl bewertet.		

7. Bewertungsbogen zur Prüfungsarbeit

Name des Prüflings: _____ Kursbezeichnung: _____

Schule: _____

Teilaufgabe a)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**[2]	**ZK**	**DK**
1	stellt die Entitäten …	4			
2	bestimmt die Kardinalitäten.	2			
3	übersetzt das Modell …	4			
	Summe Teilaufgabe a)	**10**			

Teilaufgabe b)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	wendet die erste …	2			
2	wendet die zweite …	4			
3	wendet die dritte …	6			
	Summe Teilaufgabe b)	**12**			

Teilaufgabe c)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	entwickelt eine SQL-Anweisung …	5			
2	entwickelt eine SQL-Anweisung …	5			
3	entwickelt eine SQL-Anweisung …	6			
	Summe Teilaufgabe c)	**16**			

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

Teilaufgabe d)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	gibt die Bedingungen …	6			
2	erläutert, dass die …	3			
3	überführt das Relationenschema …	3			
	Summe Teilaufgabe d)	**12**			

	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				
Note ggf. unter Absenkung um ein bis zwei Notenpunkte gemäß § 13 Abs. 2 APO-GOSt				
Paraphe				

ggf. arithmetisches Mittel der Punktsummen aus EK und ZK: _____

ggf. arithmetisches Mittel der Notenurteile aus EK und ZK: _____

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 – 39
mangelhaft plus	3	38 – 33
mangelhaft	2	32 – 27
mangelhaft minus	1	26 – 20
ungenügend	0	19 – 0