Name: _____

Abiturprüfung 2018

Informatik, Grundkurs

Aufgabenstellung:

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

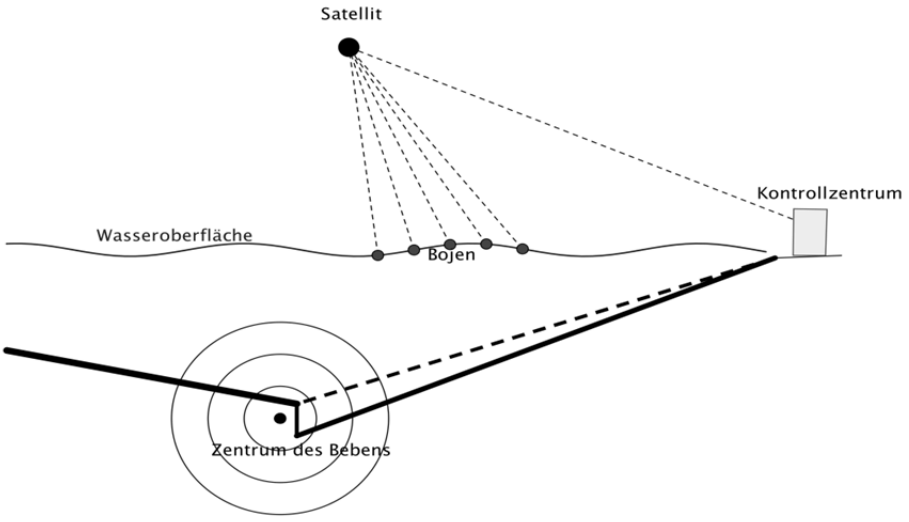


Abbildung 1: Querschnittsskizze der Komponenten des Warnsystems

Name: _____

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

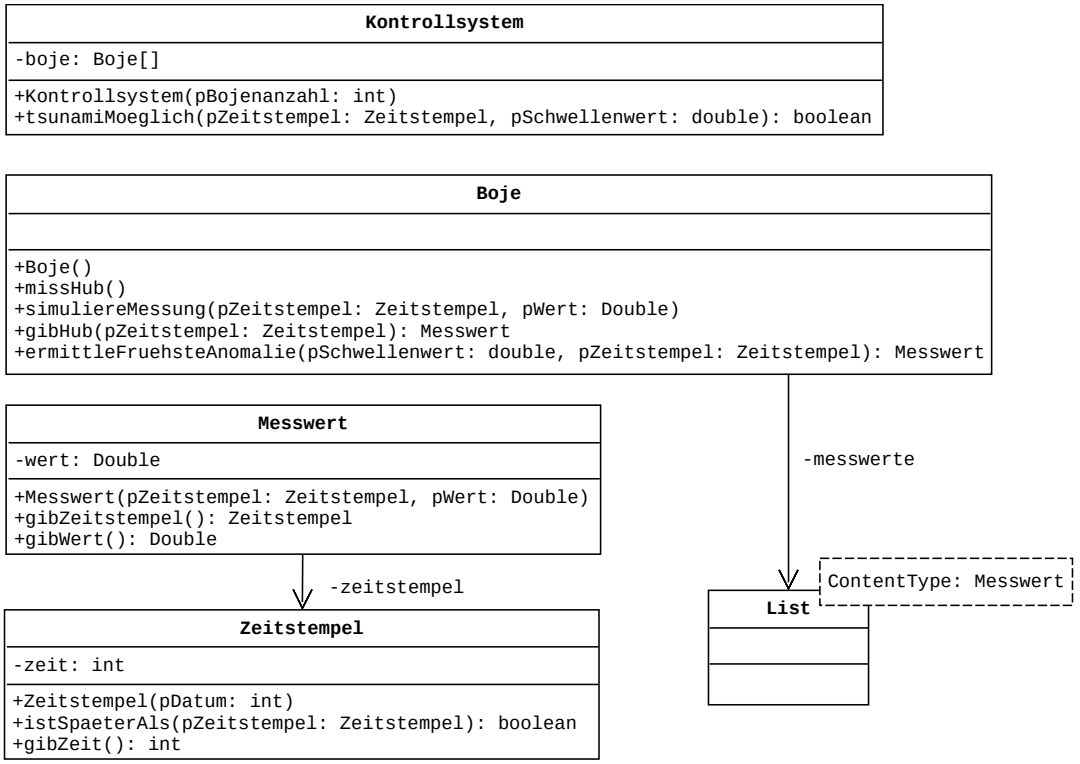


Abbildung 2: Ausschnitt aus dem Implementationsdiagramm

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

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

(8 Punkte)

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

Implementieren Sie diese Methode.

Erläutern Sie die Funktionsweise Ihrer Implementation.

(10 Punkte)

Name: _____

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

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

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

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

(10 Punkte)

Name: _____

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

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

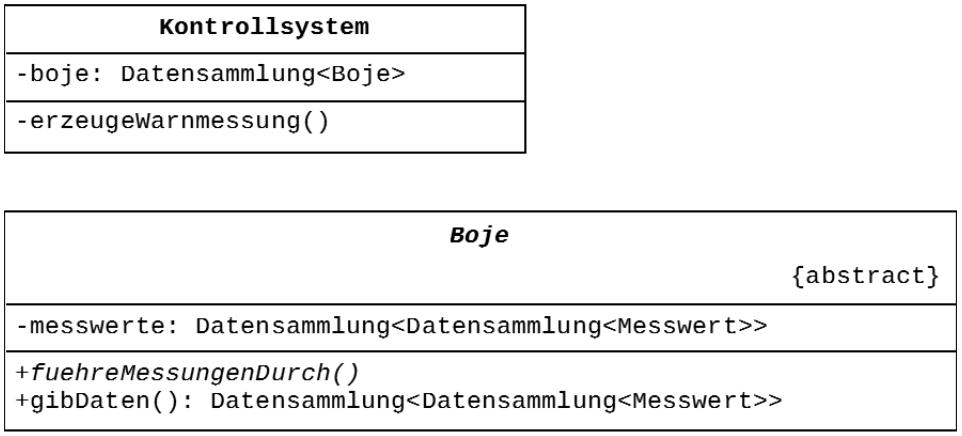


Abbildung 3: Entwurfsdiagramm

Erläutern Sie wesentliche Änderungen zum bisherigen Modell.

Überführen Sie das Entwurfsdiagramm aus Abbildung 3 in ein Implementationsdiagramm.

Erläutern Sie Ihre Entscheidungen hinsichtlich der gewählten Datenstrukturen.

(12 Punkte)

Name: _____

e) Zur Datenanalyse in einem umfassenden Projekt muss der Median[1] (Zentralwert) einer Sammlung von Objekten des Typs `Messwert` bestimmt werden.

Entwickeln Sie ein algorithmisches Verfahren zur Ermittlung des Medians einer Sammlung von Messwerten.

Erläutern Sie dabei insbesondere Ihre Wahl der Datenstrukturen.

Begründen Sie, warum der Median in diesem Sachzusammenhang im Allgemeinen nicht sinnvoll durch ein Objekt vom Typ `Messwert` *repräsentiert werden kann.*

Hinweis: Eine Implementation ist **nicht** gefordert!

(10 Punkte)

Zugelassene Hilfsmittel:

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

[1] Sind n Elemente einer Stichprobe nach der Größe geordnet, so heißt im Falle n ungerade der an (n+1)/2-ter Stelle stehende Wert Median m, im Falle n gerade der Mittelwert aus den an (n/2)-ter und (n/2+1)-ter Stelle stehenden Werten (vgl.: Bronstein, Semendjajew, Musiol, Mühling: Taschenbuch der Mathematik. Verlag Harri Deutsch. Thun und Frankfurt am Main. 3., überarbeitete und erweiterte Auflage 1997).

Name: _____

Anhang

Dokumentationen der verwendeten Klassen

Die Klasse Kontrollsystem

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

Ausschnitt aus der Dokumentation der Klasse `Kontrollsystem`

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

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

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

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

Auftrag **`void erzeugeWarnmessung()`**

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

Name: _____

Die Klasse Boje

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

Ausschnitt aus der Dokumentation der Klasse `Boje`

Konstruktor **`Boje()`**

Eine Boje wird erzeugt.

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

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

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

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

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

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

Auftrag **`void missHub()`**

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

Name: _____

Die Klasse `Messwert`

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

Ausschnitt aus der Dokumentation der Klasse `Messwert`

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

Ein Messwert mit dem übergebenen Zeitstempel und Wert wird erzeugt.

Anfrage **`Double gibWert()`**

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

Anfrage **`Zeitstempel gibZeitstempel()`**

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

Die Klasse `Zeitstempel`

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

Ausschnitt aus der Dokumentation der Klasse `Zeitstempel`

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

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

Anfrage **`int gibZeit()`**

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

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

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

Name: _____

Die generische Klasse List<ContentType>

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

Das aktuelle Objekt kann gelesen, verändert oder gelöscht werden. Außerdem kann vor dem aktuellen Objekt ein Listenobjekt eingefügt werden.

Dokumentation der Klasse `List<ContentType>`

Konstruktor **`List()`**

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

Anfrage **`boolean isEmpty()`**

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

Anfrage **`boolean hasAccess()`**

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

Auftrag **`void next()`**

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

Auftrag **`void toFirst()`**

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

Auftrag **`void toLast()`**

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

Name: _____

Anfrage **`ContentType getContent()`**

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

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

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

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

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

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

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

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

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

Auftrag **`void remove()`**

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

Unterlagen für die Lehrkraft

Abiturprüfung 2018

Informatik, Grundkurs

1. Aufgabenart

Modellierung, Implementation und Analyse kontextbezogener Problemstellungen

2. Aufgabenstellung[1]

siehe Prüfungsaufgabe

3. Materialgrundlage

entfällt

4. Bezüge zum Kernlehrplan und zu den Vorgaben 2018

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

1. *Inhaltsfelder und inhaltliche Schwerpunkte*
   Daten und ihre Strukturierung
   - Objekte und Klassen
     - Entwurfsdiagramme und Implementationsdiagramme
     - Lineare Strukturen
       Lineare Liste, Array

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

2. *Medien/Materialien*
   - entfällt

5. Zugelassene Hilfsmittel

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

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

6. Modelllösungen

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

Teilaufgabe a)

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

Teilaufgabe b)

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

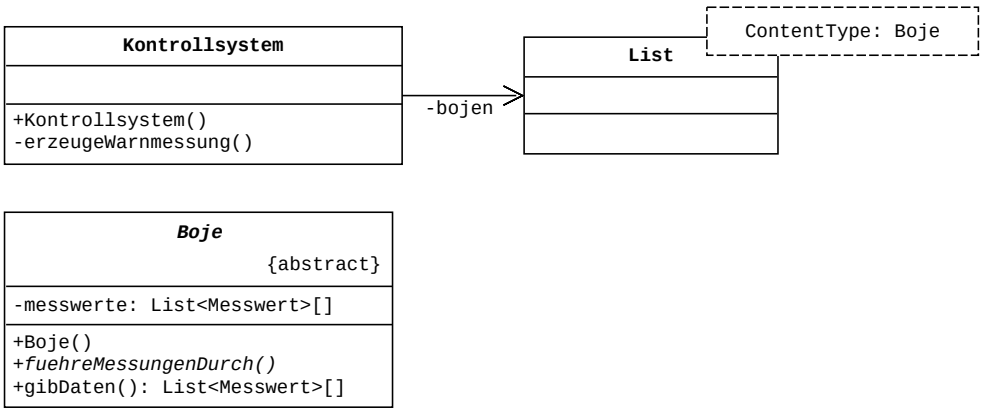
Die Liste der Messwerte wird von ihrem ersten Objekt an durchlaufen (Zeile 4/5). Ein Verweis auf das aktuelle Inhaltsobjekt wird gespeichert (Zeile 6). Zeitstempel und Messwert des aktuellen Objektes werden ermittelt (Zeile 7/8). Ist der Wert größer als der Schwellenwert und liegt zeitlich später als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

Teilaufgabe c)

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

Teilaufgabe d)

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



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

Teilaufgabe e)

Angenommen, die Daten liegen in Form einer nicht leeren linearen Liste vor. Eine Zählvariable `n` wird auf 0 gesetzt. Die Liste wird nun von Beginn an linear durchlaufen und die gültigen Werte (nicht `null`) werden in eine neue Liste `liste` vom Typ `List` mittels Sortieren durch Einfügen (z. B. Insertionsort) eingefügt und dabei `n` um eins erhöht.
Ist `n` ungerade, so wird `liste` von vorne bis zum `(n+1)/2`. Element durchlaufen, dies ist der Median der Liste. Andernfalls wird `liste` bis zum `n/2.` Element durchlaufen. Der Mittelwert aus diesem und dem folgenden Wert bildet den Median.
Enthält die Liste eine gerade Anzahl von Messwerten, so entspricht der Median im Allgemeinen keinem der Werte, der tatsächlich in der Liste gespeichert ist. Darüber hinaus kann in diesem Fall kein sinnvoller Zeitstempel angegeben werden.

7. Teilleistungen – Kriterien / Bewertungsbogen zur Prüfungsarbeit

Name des Prüflings:_____ Kursbezeichnung:_____

Schule: _____

Teilaufgabe a)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK[2]**	**ZK**	**DK**
1	erläutert die Beziehungen zwischen den Klassen.	6			
2	erläutert, warum in der Klasse `Messwert` der Typ `Double` verwendet wird.	2			
Sachlich richtige Lösungsalternative zur Modelllösung: (8) …… ……					
	Summe Teilaufgabe a)	**8**			

Teilaufgabe b)

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

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

Teilaufgabe c)

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

Teilaufgabe d)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	erläutert wesentliche Änderungen zum bisherigen Modell.	3			
2	überführt das Diagramm in ein Implementationsdiagramm.	6			
3	erläutert seine Entscheidungen bzgl. der gewählten Datenstrukturen.	3			
Sachlich richtige Lösungsalternative zur Modelllösung: (12) …… ……					
	Summe Teilaufgabe d)	**12**			

Teilaufgabe e)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	entwickelt ein algorithmisches Verfahren zur Ermittlung des Medians einer Sammlung von Messwerten.	4			
2	erläutert die Wahl der Datenstrukturen.	3			
3	begründet, warum der Median nicht durch ein Objekt vom Typ `Messwert` repräsentiert werden kann.	3			
Sachlich richtige Lösungsalternative zur Modelllösung: (10) …… ……					
	Summe Teilaufgabe e)	**10**			

	Summe insgesamt	**50**			

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

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

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

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

Unterschrift, Datum:

Grundsätze für die Bewertung (Notenfindung)

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

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

Name: _____

Abiturprüfung 2018

Informatik, Grundkurs

Aufgabenstellung:

Ein kleines Softwareunternehmen möchte ein neues, innovatives Denkspiel namens *ThinkTree* entwickeln. Beim Start des Spiels wird ein zufällig generierter Binärbaum angezeigt, in dem alle inneren Knoten eine Richtung anzeigen, die entweder auf den linken oder den rechten Teilbaum weist. In allen Blättern stehen Punktwerte. Folgt man von der Wurzel aus den Richtungsangaben, kommt man zwangsläufig zu einem bestimmten Blatt, d. h. einem bestimmten Punktwert. Abbildung 1 zeigt einen Beispielbaum, der zum Punktwert 42 führt.

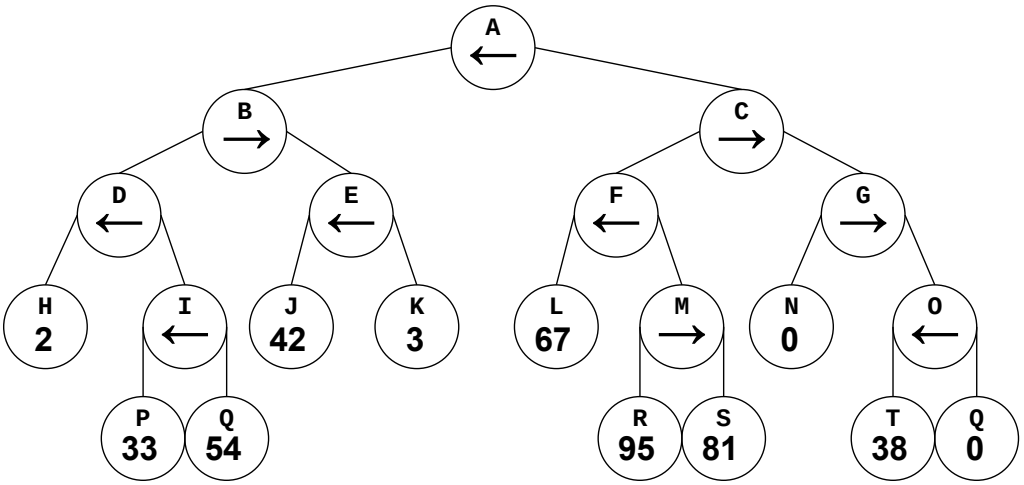


Abbildung 1: Exemplarischer Spielbaum

Die Spielerin oder der Spieler hat nun einen begrenzten Zeitraum von wenigen Sekunden, um bei einer vorgegebenen Anzahl von inneren Knoten die Richtung zu wechseln. Ziel des Spiels ist es, auf diese Weise zu einem Blatt mit möglichst hohem Punktwert zu gelangen.

a) *Erläutern Sie, welcher Punktwert erzielt wird, wenn in Abbildung 1 die Richtungen der Knoten* `B` *und* `D` *gewechselt werden.*

Erläutern Sie, welcher Punktwert bestenfalls erreicht werden kann, wenn die Richtungen zweier Knoten gewechselt werden dürfen.

Begründen Sie im Sachzusammenhang, warum jeder innere Knoten genau zwei Teilbäume haben muss.

(8 Punkte)

Name: _____

Das Implementationsdiagramm in Abbildung 2 zeigt einen Ausschnitt aus der Modellierung des Spiels.

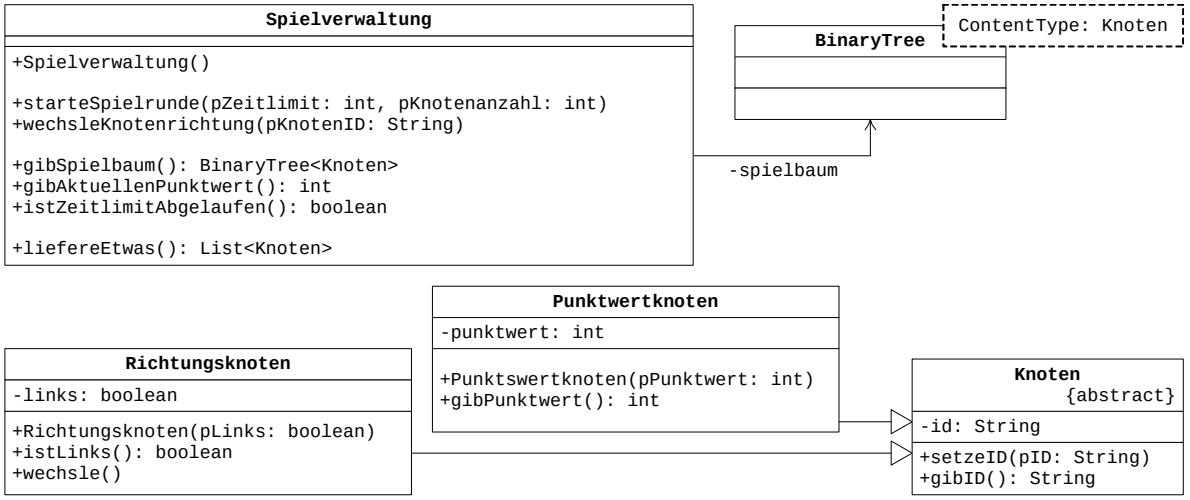


Abbildung 2: Teilmodellierung des Spiels *ThinkTree*

b) *Analysieren Sie auf Grundlage des in Abbildung 2 gegebenen Implementationsdiagramms und der Dokumentation im Anhang die Modellierung des Spiels ThinkTree und erläutern Sie die Klassen und ihre Beziehungen untereinander.*

Ermitteln Sie, wie viele Objekte der Klassen `Spielverwaltung`, `Richtungsknoten` *und* `Punktwertknoten` *benötigt werden, um den in Abbildung 1 gegebenen Baum zu spielen, und begründen Sie ihre Antwort.*

(12 Punkte)

c) Die Methode `wechsleKnotenrichtung` der Klasse `Spielverwaltung` soll den Knoten mit dem im Parameter übergebenen ID-Wert im Binärbaum `spielbaum` suchen und seine Richtung wechseln, wenn das Zeitlimit für die aktuelle Spielrunde noch nicht abgelaufen ist.

Die Methode hat den folgenden Methodenkopf:

```
public void wechsleKnotenrichtung(String pKnotenID)
```

Erläutern Sie für die Methode `wechsleKnotenrichtung` *und eventuelle Hilfsmethoden eine algorithmische Strategie.*

Implementieren Sie die Methode `wechsleKnotenrichtung` *und eventuelle Hilfsmethoden entsprechend Ihrer Strategie.*

(10 Punkte)

Name: _____

d) Die Methode `liefereEtwas` der Klasse `Spielverwaltung` ist wie folgt implementiert:

```
 1 public List<Knoten> liefereEtwas(){
 2   return liefereEtwas(spielbaum);
 3 }
 4
 5 private List<Knoten> liefereEtwas(BinaryTree<Knoten> pAktuell) {
 6   if (pAktuell.getLeftTree().isEmpty()
                            && pAktuell.getRightTree().isEmpty()) {
 7     List<Knoten> neu = new List<Knoten>();
 8     neu.append(pAktuell.getContent());
 9     return neu;
10   } else {
11     List<Knoten> links = liefereEtwas(pAktuell.getLeftTree());
12     links.toFirst();
13     int punktwertLinks =
             ((Punktwertknoten) links.getContent()).gibPunktwert();
14
15     List<Knoten> rechts = liefereEtwas(pAktuell.getRightTree());
16     rechts.toFirst();
17     int punktwertRechts =
            ((Punktwertknoten) rechts.getContent()).gibPunktwert();
18
19     if (punktwertLinks < punktwertRechts) {
20       rechts.append(pAktuell.getContent());
21       return rechts;
22     } else {
23       links.append(pAktuell.getContent());
24       return links;
25     }
26   }
27 }
```

Analysieren und erläutern Sie die öffentliche Methode `liefereEtwas` *mit der von ihr aufgerufenen privaten Methode* `liefereEtwas` *und gehen Sie dabei insbesondere auf die verwendete Rekursion ein.*

Erläutern Sie im Sachzusammenhang, was die Methoden liefern.

(12 Punkte)

Name: _____

e) Die in Abbildung 2 gegebene Modellierung soll für eine Folgeversion des Spiels erweitert werden. Für die neue Version des Spiels sollen folgende Änderungen realisiert werden:
   - Alle Knoten, auch die inneren Knoten, sollen Punktwerte bekommen.
   - Innere Knoten sollen durch die Spielerin oder den Spieler „blockiert“ werden können. Gelangt man beim Durchlaufen des Baums zu einem blockierten inneren Knoten, wird nicht weiter in seine Teilbäume verzweigt. Stattdessen wird der Punktwert des blockierten Knotens als Ergebnis des Zuges angenommen.

   Entwerfen Sie auf Grundlage des in Abbildung 2 gegebenen Implementationsdiagramms eine Modellierung, welche die oben gegebenen Erweiterungen berücksichtigt, und geben Sie die neue Modellierung in Form eines Implementationsdiagramms an.

   (8 Punkte)

Zugelassene Hilfsmittel:

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

Name: _____

Anhang

Dokumentationen der verwendeten Klassen

Dokumentation der Klasse Spielverwaltung

Konstruktor `Spielverwaltung()`

Ein neues Objekt der Klasse `Spielverwaltung` wird erstellt. Es existiert noch kein Spielbaum.

Auftrag `void starteSpielrunde(int pZeitlimit, int pKnotenanzahl)`

Ein neuer Spielbaum wird per Zufall erstellt. Er verfügt über `pKnotenanzahl` an inneren Knoten. Anschließend wird eine neue Spielrunde gestartet, d. h. für `pZeitlimit` Sekunden nach Aufruf dieser Methode können Richtungen von Knoten gewechselt werden.

Auftrag `void wechsleKnotenrichtung(String pKnotenID)`

Ist ein Knoten mit `pKnotenID` als ID im Spielbaum, so wird seine Richtung gewechselt, sofern das Zeitlimit der aktuellen Spielrunde noch nicht abgelaufen ist. Ansonsten passiert nichts.

Anfrage `BinaryTree<Knoten> gibSpielbaum()`

Liefert den aktuellen Spielbaum zurück.

Anfrage `int gibAktuellenPunktwert()`

Liefert den Punktwert zurück, den man bei der aktuellen Ausrichtung der Knoten im Spielbaum erreicht. Gibt es noch keinen Spielbaum wird `0` geliefert.

Anfrage `boolean istZeitlimitAbgelaufen()`

Liefert `true`, sofern das Zeitlimit seit dem Start der aktuellen Spielrunde abgelaufen ist oder noch keine Spielrunde gestartet wurde. Ansonsten wird `false` geliefert.

Anfrage `List<Knoten> ermittleEtwas()`

Diese Methode soll in Teilaufgabe d) analysiert werden.

Name: _____

Dokumentation der Klasse `Richtungsknoten`

Konstruktor `Richtungsknoten(boolean pLinks)`

Ein neues Objekt der Klasse `Richtungsknoten` wird erstellt. Ist `pLinks` `true`, so ist die Richtung des Knotens auf den linken Teilbaum gesetzt, ansonsten auf den rechten.

Anfrage `boolean istLinks()`

Liefert `true`, wenn der Knoten auf den linken Teilbaum weist, ansonsten `false`.

Auftrag `void wechsle()`

Wechselt die Richtung des Knotens auf den jeweils anderen Teilbaum.

Dokumentation der Klasse `Punktwertknoten`

Konstruktor `Punktwertknoten(int pPunktwert)`

Ein neues Objekt der Klasse `Punktwertknoten` wird erstellt. Dieses Objekt speichert den in `pPunktwert` übergebenen Wert als Punktwert.

Anfrage `int gibPunktwert()`

Liefert den gespeicherten Punktwert des Objekts.

Dokumentation der abstrakten Klasse `Knoten`

Auftrag `setzeID(String pID)`

Setzt die ID des Knotens auf `pID`.

Anfrage `String gibID()`

Liefert die ID des Knotens.

Name: _____

Die Klasse BinaryTree<ContentType>

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

Dokumentation der Klasse `BinaryTree<ContentType>`

Konstruktor **`BinaryTree()`**

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

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

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

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

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

Anfrage **`boolean isEmpty()`**

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

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

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

Name: _____

Anfrage `ContentType getContent()`

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

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

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

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

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

Anfrage `BinaryTree<ContentType> getLeftTree()`

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

Anfrage `BinaryTree<ContentType> getRightTree()`

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

Unterlagen für die Lehrkraft

Abiturprüfung 2018

Informatik, Grundkurs

1. Aufgabenart

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

2. Aufgabenstellung[1]

siehe Prüfungsaufgabe

3. Materialgrundlage

entfällt

4. Bezüge zum Kernlehrplan und zu den Vorgaben 2018

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

1. *Inhaltsfelder und inhaltliche Schwerpunkte*
   Daten und ihre Strukturierung
   - Objekte und Klassen
     - Entwurfsdigramme und Implementationsdiagramme
     - Lineare Strukturen
       Lineare Liste
     - Nicht-lineare Strukturen
       Binärbaum

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

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

2. *Medien/Materialien*
   - entfällt

5. Zugelassene Hilfsmittel

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

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

6. Modelllösungen

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

Teilaufgabe a)

Werden die Richtungen der Knoten `B` und `D` gewechselt, werden `33` Punkte erzielt. Die Wurzel weist auf den linken Teilbaum, daher wird der Knoten `B` angesteuert. Dieser Knoten weist auf den rechten und nach dem Wechsel auf den linken Teilbaum, so dass zum Knoten `D` gegangen wird. Dieser weist nach dem Wechsel in Richtung des Knotens `I` und dieser auf das Blatt `P` mit dem Punktwert `33`.

Den höchsten Punktwert im Baum stellt das Blatt `R` mit `95` Punkten dar. Dieses Blatt kann erreicht werden, wenn die Richtungen von vier Knoten (`A`, `C`, `F` und `M`) gewechselt werden. Das Blatt `S` mit `81` Punkten stellt die zweithöchste Punktzahl dar und kann nur mit einem Wechsel von drei Knoten (`A`, `C` und `F`) erreicht werden. Der dritthöchste Punktwert steht in Blatt `L` mit `67` Punkten. Dieser Knoten ist mit Richtungswechseln in den Knoten `A` und `C` zu erreichen. Somit handelt es sich hierbei um den höchsten Punktwert, der mit zwei Richtungswechseln zu erreichen ist.

Im Sachzusammenhang müssen alle inneren Knoten zwei Teilbäume haben, da ansonsten für einen inneren Knoten die Richtung so zu setzen wäre, dass man beim Durchlaufen des Baums gar nicht zu einem bepunkteten Blatt gelangt, sondern ins Leere läuft.

Teilaufgabe b)

Ein Objekt der Klasse `Spielverwaltung` verwaltet einen Spielbaum `spielbaum` vom Typ `BinaryTree` und stellt zentrale Funktionen zur Durchführung des Spiels zur Verfügung. Mit seiner Hilfe kann eine neue Spielrunde mit variablem Zeitlimit und mit einem neuen Spielbaum variabler Größe erstellt werden (`starteSpielrunde`). Richtungen einzelner Knoten können gewechselt werden (`wechsleKnotenrichtung`) und der aktuell zu erreichende Punktwert des Baums kann angefragt werden (`gibAktuellenPunktwert`). Des Weiteren kann der Spielbaum geliefert (`gibSpielbaum`) und geprüft werden, ob die Spielrunde bereits beendet ist (`istZeitlimitAbgelaufen`). Die Klasse `Spielverwaltung` beinhaltet auch die undokumentierte Methode `liefereEtwas`.

Ein Objekt der Klasse `Richtungsknoten` stellt einen inneren Knoten des Spielbaums dar. Beim Konstruktoraufruf wird die Anfangsrichtung des Knotens übergeben. Diese kann später beliebig oft auf die jeweils andere Richtung geändert (`wechsle`) und auch abgefragt werden (`istLinks`). Dazu wird sie in dem booleschen Attribut `links` gespeichert.

Ein Objekt der Klasse `Punktwertknoten` stellt ein Blatt des Spielbaums dar. Sein Punktwert wird im Konstruktoraufruf übergeben und kann später nur noch abgerufen werden (`gibPunktwert`). Dazu wird er in dem Attribut `punktwert` gespeichert.

Die abstrakte Klasse `Knoten` ist die Oberklasse für die Klassen `Richtungsknoten` und `Punktwertknoten`. Sie modelliert für jedes Objekt dieser Klassen eine `ID`, welche mit der Methode `setzeID` übergeben und mit der Methode `gibID` abgerufen werden kann.

Die Klasse `BinaryTree` modelliert rekursiv einen Binärbaum und hat in diesem Fall `Knoten` als Inhaltstyp. Sie kann also Objekte der Klassen `Richtungsknoten` und `Punktwertknoten` aufnehmen.

Klasse `Spielverwaltung`:

Da genau ein Spielbaum aus Abbildung 1 verwaltet werden muss und ein Objekt dieser Klasse genau einen Baum verwaltet, ist genau ein Objekt dieser Klasse erforderlich.

Klasse `Richtungsknoten`:

Da ein Objekt der Klasse `Richtungsknoten` einen inneren Knoten eines Spielbaums darstellt, werden so viele Objekte dieser Klasse gebraucht, wie es innere Knoten gibt. Im Fall des Spielbaums in Abbildung 1 sind das `10` Objekte.

Klasse `Punktwertknoten`:

Analog zum Richtungsknoten werden so viele Objekte gebraucht, wie der Spielbaum Blätter hat. In diesem Fall sind das `11` Objekte.

Teilaufgabe c)

Die Methode `wechsleKnotenrichtung` kann mit einer Hilfsmethode folgender Strategie gemäß arbeiten:
Zunächst wird mit Hilfe der Methode `istZeitAbgelaufen` der Klasse `Spielverwaltung` geprüft, ob ein Richtungswechsel des Knotens `pKnoten` überhaupt noch erlaubt ist. Ist das der Fall, wird der Binärbaum `spielbaum` mit einer zusätzlichen, privaten Hilfsmethode rekursiv durchlaufen. Dabei wird der Knoten `pKnoten` gesucht und seine Richtung ggf. geändert.
Der rekursive Durchlauf mit einer privaten Hilfsmethode wird wie folgt realisiert: Die Methode wird zunächst mit der Wurzel des Spielbaums als aktuellem Knoten aufgerufen. Sie prüft, ob dieser aktuelle Knoten ein innerer Knoten ist. Ist das der Fall, wird geprüft, ob es sich um den gesuchten Knoten `pKnoten` handelt, dessen Richtung dann mit einem Aufruf seiner Methode `wechsle` geändert wird. Handelt es sich nicht um den gesuchten Knoten, wird analog mit dem linken und dem rechten Teilbaum des aktuellen Knotens verfahren, indem die private Hilfsmethode rekursiv für diese Teilbäume aufgerufen wird.

Die Methode `wechsleKnotenrichtung` kann wie folgt implementiert werden:

```
public void wechsleKnotenrichtung(String pKnotenID) {
  if (!this.istZeitAbgelaufen()) {
    wechsleKnotenrichtung(spielbaum, pKnotenID);
  }
}

private void wechsleKnotenrichtung(BinaryTree<Knoten> pBaum,
                                                  String pKnotenID) {
  if (!pBaum.getLeftTree().isEmpty() ||
                                !pBaum.getRightTree().isEmpty()) {
    if (pBaum.getContent().gibID().equals(pKnotenID)) {
      ((Richtungsknoten) pBaum.getContent()).wechsle();
    } else {
      wechsleKnotenrichtung(pBaum.getLeftTree(), pKnotenID);
      wechsleKnotenrichtung(pBaum.getRightTree(), pKnotenID);
    }
  }
}
```

Teilaufgabe d)

Die öffentliche Methode `liefereEtwas` liefert das Ergebnis eines Methodenaufrufs der privaten Hilfsmethode `liefereEtwas` zurück. Die private Methode `liefereEtwas` wird dabei mit der Wurzel des Spielbaums aufgerufen (Z. `1` – `3`).

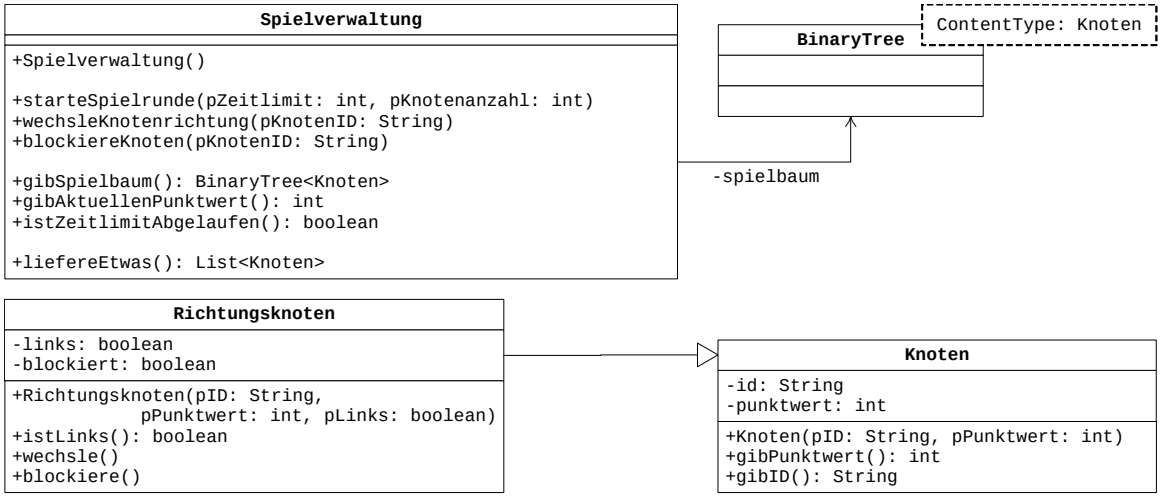
Die Methode `liefereEtwas` prüft in Zeile `6`, ob der in `pAktuell` übergebene Knoten ein Blatt ist. Ist das der Fall, wird eine neue Liste zurückgeliefert, die genau dieses Blatt beinhaltet (Z. 7 – `9`).

Ist `pAktuell` kein Blatt, werden durch rekursive Aufrufe die Liste `links` für den linken Teilbaum und die Liste `rechts` für den rechten Teilbaum erzeugt. Des Weiteren wird für jede dieser Listen der Punktwert des ersten Knotens ermittelt. (Z. 11 – 17) Abschließend wird der aktuelle Knoten an diejenige Liste angehängt, deren erstes Element den höchsten Punktwert hat. Diese Liste wird dann zurückgeliefert. (Z. 19 – 25)

Im Sachzusammenhang liefert die Methode `liefereEtwas` eine Liste aller Knoten, die auf dem Pfad vom punkthöchsten Blatt bis zur Wurzel liegen.

Teilaufgabe e)

Die folgende Modellierung stellt eine Möglichkeit dar, die neuen Vorgaben umzusetzen:



7. Teilleistungen – Kriterien / Bewertungsbogen zur Prüfungsarbeit

Name des Prüflings:_____ Kursbezeichnung:_____

Schule: _____

Teilaufgabe a)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK[2]**	**ZK**	**DK**
1	erläutert, welcher Punktwert erzielt wird, wenn die Richtungen der Knoten B und D gewechselt werden.	3			
2	erläutert, welcher Punktwert bestenfalls erreicht werden kann, wenn die Richtungen zweier Knoten gewechselt werden dürfen.	3			
3	begründet im Sachzusammenhang, warum jeder innere Knoten genau zwei Teilbäume haben muss.	2			
Sachlich richtige Lösungsalternative zur Modelllösung: (8) …… ……					
	Summe Teilaufgabe a)	**8**			

Teilaufgabe b)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	analysiert und erläutert die Klassen und ihre Beziehungen untereinander.	6			
2	ermittelt, wie viele Objekte der Klassen benötigt werden.	3			
3	begründet seine Antwort jeweils.	3			
Sachlich richtige Lösungsalternative zur Modelllösung: (12) …… ……					
	Summe Teilaufgabe b)	**12**			

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

Teilaufgabe c)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	erläutert für die Methode eine algorithmische Strategie.	5			
2	implementiert die Methode entsprechend seiner Strategie.	5			
Sachlich richtige Lösungsalternative zur Modelllösung: (10) …… ……					
	Summe Teilaufgabe c)	**10**			

Teilaufgabe d)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	analysiert und erläutert die Methode.	9			
2	erläutert im Sachzusammenhang, was die Methoden liefern.	3			
Sachlich richtige Lösungsalternative zur Modelllösung: (12) …… ……					
	Summe Teilaufgabe d)	**12**			

Teilaufgabe e)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	entwirft eine Modellierung und gibt sie in Form eines Implementationsdiagramms an.	8			
Sachlich richtige Lösungsalternative zur Modelllösung: (8) …… ……					
	Summe Teilaufgabe e)	**8**			

	Summe insgesamt	**50**			

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

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

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

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

Unterschrift, Datum:

Grundsätze für die Bewertung (Notenfindung)

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

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

Name: _____

Abiturprüfung 2018

Informatik, Grundkurs

Aufgabenstellung:

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

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

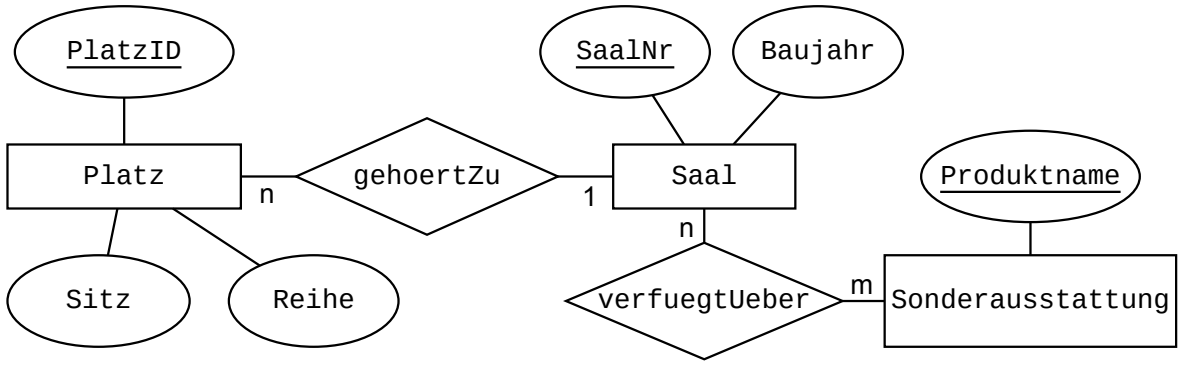


Abbildung 1: Teilmodellierung der Datenbank

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

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

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

(10 Punkte)

Name: _____

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

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

Zudem sind in folgender Tabelle einige Beispieldatensätze angegeben:

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

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

(10 Punkte)

Name: _____

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

Film(FilmID, Titel, Mindestalter, Erscheinungsjahr)

Saal(SaalNr, Baujahr)

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

Buchung(BuchungsNr, ↑VorstellungID, ↑KundenNr)

Platz(PlatzID, Reihe, Sitz, ↑SaalNr)

belegt(↑BuchungsNr, ↑PlatzID)

Abbildung 2: Teilmodellierung der Datenbank bezüglich der Vorstellungen

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

i)
```
1 SELECT Titel
2 FROM Film
3 WHERE Erscheinungsjahr >= 2018 AND Mindestalter < 18
4 ORDER BY Titel ASC
```

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

Analysieren und erläutern Sie die obigen SQL-Anweisungen.

Erläutern Sie im Sachzusammenhang, welche Informationen die SQL-Anweisungen ermitteln.

(10 Punkte)

Name: _____

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

i) Es sollen die IDs der Plätze aus Saal 3 ermittelt werden, die in den Reihen 1, 2 oder 3 liegen.

ii) Es sollen die IDs der Filme ermittelt werden, für die es keine Vorstellungen gibt.

iii) Für die Vorstellung mit `VorstellungID 1` soll die Anzahl der gebuchten Plätze ermittelt werden.

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

(15 Punkte)

e) Das Datenbankschema soll nun nicht nur für ein Kino eingesetzt werden. Dafür müssen folgende zusätzliche Anforderungen erfüllt werden.

i) Ein Kino kann mehrere Säle besitzen.

ii) Ein Kino hat einen Namen und eine Adresse (Straße, Hausnummer und Postleitzahl).

iii) Jedes Kino gehört zu einem Betreiber, dessen ID und Name gespeichert werden. Ein Betreiber kann mehrere Kinos haben.

Erweitern Sie das Entity-Relationship-Diagramm aus Abbildung 1 gemäß den oben genannten Anforderungen.

Hinweis: Abgesehen vom Entitätstyp `Saal` müssen vorhandene Entitätstypen, Attribute und Beziehungstypen nicht aufgeführt werden.

(5 Punkte)

Zugelassene Hilfsmittel:

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

Unterlagen für die Lehrkraft

Abiturprüfung 2018

Informatik, Grundkurs

1. Aufgabenart

Analyse, Modellierung und Abfrage relationaler Datenbanken

2. Aufgabenstellung[1]

siehe Prüfungsaufgabe

3. Materialgrundlage

entfällt

4. Bezüge zum Kernlehrplan und zu den Vorgaben 2018

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

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

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

2. *Medien/Materialien*
   - entfällt

5. Zugelassene Hilfsmittel

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

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

6. Modelllösungen

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

Teilaufgabe a)

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

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

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

Teilaufgabe b)

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

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

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

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

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

Teilaufgabe c)

i) Aus der Relation `Film` werden die Attributausprägungen des Attributs `Titel` derjenigen Datensätze selektiert, für die zwei Einschränkungen gelten. Zum einen muss die Attributausprägung des Attributs `Erscheinungsjahr` größer oder gleich `2018` sein und zum anderen muss in eben diesem Datensatz die Attributausprägung für das Attribut `Mindestalter` kleiner als `18` sein. Alle so ermittelten Datensätze werden in Bezug auf den `Titel` in absteigender Reihenfolge geordnet.
In absteigender Reigenfolge werden also die Filmtitel genannt, die seit diesem Jahr erschienen sind und die für Menschen unter `18` Jahren zugelassen sind.

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

Teilaufgabe d)

(i)

```
1  SELECT PlatzID
2  FROM Platz
3  WHERE Platz.SaalNr = 3
4        AND Platz.Reihe IN (1, 2, 3)
```

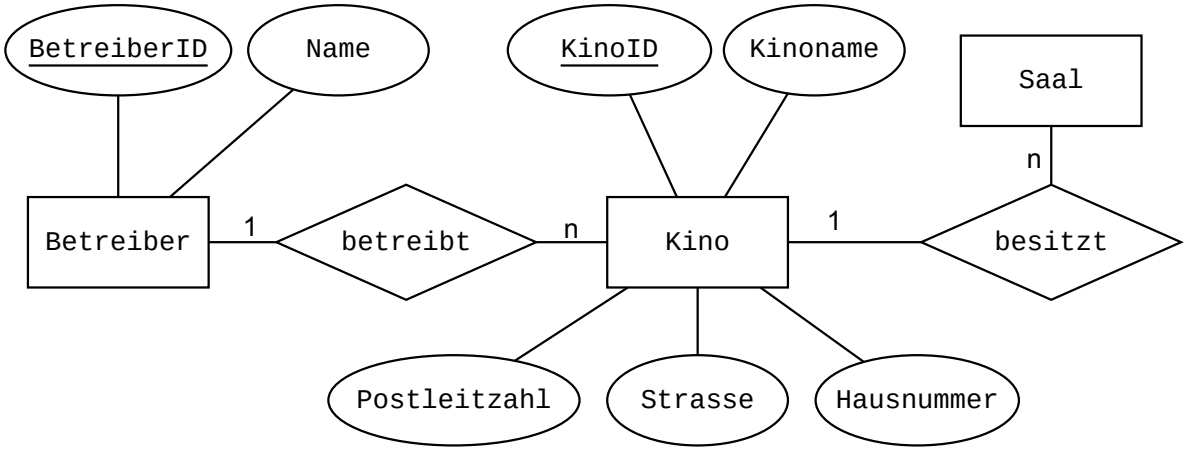
(ii)

```
1  SELECT Film.Titel
2  FROM Film
3  WHERE Film.FilmID NOT IN
4        (SELECT FilmID FROM Vorstellung)
```

(iii)

```
1  SELECT COUNT(*)
2  FROM Buchung
3    INNER JOIN belegt
4      ON Buchung.BuchungsNr = belegt.BuchungsNr
5  WHERE Buchung.VorstellungID = 1
```

Teilaufgabe e)



7. Teilleistungen – Kriterien / Bewertungsbogen zur Prüfungsarbeit

Name des Prüflings:_____ Kursbezeichnung:_____

Schule: _____

Teilaufgabe a)

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

Teilaufgabe b)

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

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

Teilaufgabe c)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	analysiert und erläutert die erste SQL-Anweisung.	4			
2	analysiert und erläutert die zweite SQL-Anweisung.	4			
3	erläutert im Sachzusammenhang, welche Informationen die SQL-Anweisungen ermitteln.	2			
Sachlich richtige Lösungsalternative zur Modelllösung: (10) …… ……					
	Summe Teilaufgabe c)	**10**			

Teilaufgabe d)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	entwickelt für die erste Anfrage eine SQL-Anweisung.	5			
2	entwickelt für die zweite Anfrage eine SQL-Anweisung.	5			
3	entwickelt für die dritte Anfrage eine SQL-Anweisung.	5			
Sachlich richtige Lösungsalternative zur Modelllösung: (15) …… ……					
	Summe Teilaufgabe d)	**15**			

Teilaufgabe e)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	erweitert das Entity-Relationship-Diagramm.	5			
Sachlich richtige Lösungsalternative zur Modelllösung: (5) …… ……					
	Summe Teilaufgabe e)	**5**			

	Summe insgesamt	**50**			

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

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

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

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

Unterschrift, Datum:

Grundsätze für die Bewertung (Notenfindung)

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

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

Name: _____

Abiturprüfung 2018

Informatik, Grundkurs

Aufgabenstellung:

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

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

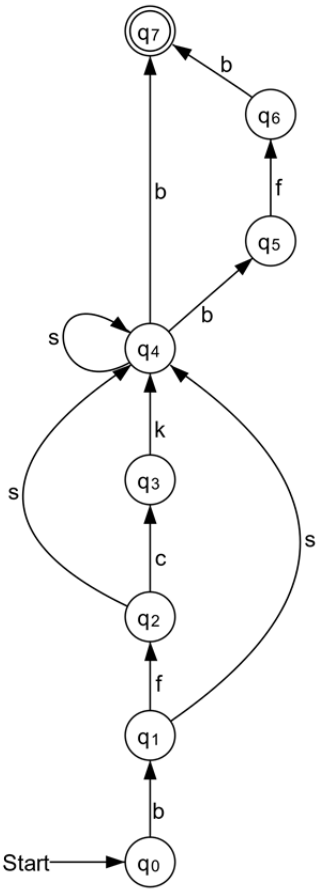


Abbildung 1: Zustandsübergangsdiagramm zum Automaten A

Name: _____

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

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

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

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

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

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

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

(13 Punkte)

b) *Entwickeln Sie für den Automaten* `A` *eine reguläre Grammatik und geben Sie das Startsymbol, die Menge der Nichtterminale, die Menge der Terminale und die Menge der Produktionen an.*

(8 Punkte)

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

i) Ein Burger kann beliebig groß werden, d. h., es dürfen beliebig viele Scheiben Brot (`b`) und beliebig viele Zutaten verwendet werden, solange folgende Regeln berücksichtigt werden:
ii) Ein Burger beginnt und endet mit einer Scheibe Brot (`b`).
iii) Zwei oder mehr Scheiben Brot (`b`) dürfen nicht direkt aufeinanderliegen.
iv) Als Füllung stehen in dieser Version nur die Zutaten Fleisch (`f`) und Salat (`s`) zur Verfügung.
Zwischen zwei Schreiben Brot (`b`) liegt entweder eine Zutat oder zwei unterschiedliche Zutaten. Die Zutaten sollen in einer beliebigen Reihenfolge erlaubt sein.

Entwerfen sie einen deterministischen endlichen Automaten (DEA), der die genannten Anforderungen erfüllt.

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

(12 Punkte)

Name: _____

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

Startsymbol: S
Nichtterminale: N = {S, A, F}
Terminale: T = {b, f, s, v}
Produktionen: P = {S → bAb,
A → Fs | FsbA,
F → f | v
}

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

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

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

Erläutern Sie am Beispiel einer der Produktionen, warum die Grammatik G *keine reguläre Grammatik ist.*

Entwerfen Sie eine reguläre Grammatik, aus der sich genau die Wörter ableiten lassen, die sich aus der Grammatik G *ableiten lassen.*

(12 Punkte)

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

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

(5 Punkte)

Zugelassene Hilfsmittel:

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

Unterlagen für die Lehrkraft

Abiturprüfung 2018

Informatik, Grundkurs

1. Aufgabenart

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

2. Aufgabenstellung[1]

siehe Prüfungsaufgabe

3. Materialgrundlage

entfällt

4. Bezüge zum Kernlehrplan und zu den Vorgaben 2018

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

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

2. *Medien/Materialien*
   - entfällt

5. Zugelassene Hilfsmittel

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

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

6. Modelllösungen

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

Teilaufgabe a)

Zustandsübergangstabelle für den Automaten A:

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

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

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

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

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

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

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

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

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

Teilaufgabe b)

Startsymbol: S

Nichtterminale: $N = \{S, Q_1, Q_2, Q_3, Q_4, Q_5, Q_6\}$

Terminale: $T = \{b, f, s, k, c\}$

Produktionen:

$$
\begin{aligned}
P = \{\ S &\rightarrow bQ_1,\\
Q_1 &\rightarrow fQ_2 \mid sQ_4,\\
Q_2 &\rightarrow cQ_3 \mid sQ_4,\\
Q_3 &\rightarrow kQ_4,\\
Q_4 &\rightarrow sQ_4 \mid bQ_5 \mid b,\\
Q_5 &\rightarrow fQ_6,\\
Q_6 &\rightarrow b\\
\}&
\end{aligned}
$$

Teilaufgabe c)

Der folgende deterministische endliche Automat entspricht den Anforderungen.

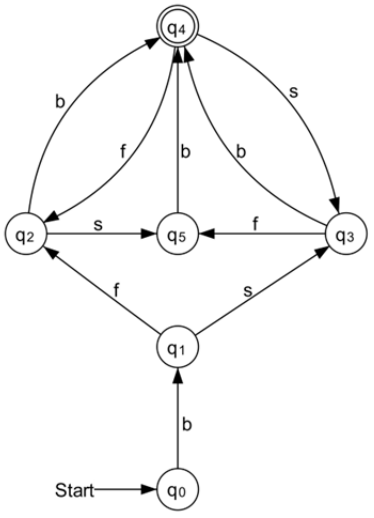
Startzustand: q_0

Endzustände: $\{q_4\}$

Zustände: $\{q_0, q_1, q_2, q_3, q_4, q_5\}$

Eingabealphabet: $\{b, f, s\}$

Zustandsübergangsdiagramm (nicht dargestellte Übergänge führen in einen Fehlerzustand):



Erläuterung zur Berücksichtigung der Anforderung i:
Mit dem Zustandsübergang aus dem Endzustand q_4 nach q_2 und nach q_3 wird ein neuer „Zutat(en)-Brot-Zyklus“ eingeleitet.

Erläuterung zur Berücksichtigung der Anforderung ii:
Das erste Zeichen vom Startzustand und das letzte Zeichen in den Endzustand muss das Eingabezeichen b sein (vgl. Zustandsübergänge von q_0 nach q_1 und die Zustandsübergänge in den Endzustand q_4).

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

Teilaufgabe d)

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

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

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

S → bAb
→ bFsbAb
→ bfsbAb
→ bfsbFsb
→ bfsbvsb

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

Aus der folgenden regulären Grammatik lassen sich dieselben Wörter ableiten wie aus der Grammatik G.

Startsymbol: S

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

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

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

Teilaufgabe e)

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

7. Teilleistungen – Kriterien / Bewertungsbogen zur Prüfungsarbeit

Name des Prüflings:_____ Kursbezeichnung:_____

Schule: _____

Teilaufgabe a)

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

Teilaufgabe b)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	entwickelt für den Automaten eine reguläre Grammatik und gibt das Startsymbol, die Nichtterminale, die Terminale und die Produktionen an.	8			
Sachlich richtige Lösungsalternative zur Modelllösung: (8) …… ……					
	Summe Teilaufgabe b)	**8**			

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

Teilaufgabe c)

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

Teilaufgabe d)

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

Teilaufgabe e)

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

	Summe insgesamt	**50**			

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

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

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

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

Unterschrift, Datum:

Grundsätze für die Bewertung (Notenfindung)

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

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