Bitte beachten:

Die Aufgaben wurden nur an der Schule
168040
verwendet.

Sie dürfen nicht veröffentlicht werden!

Name: _____

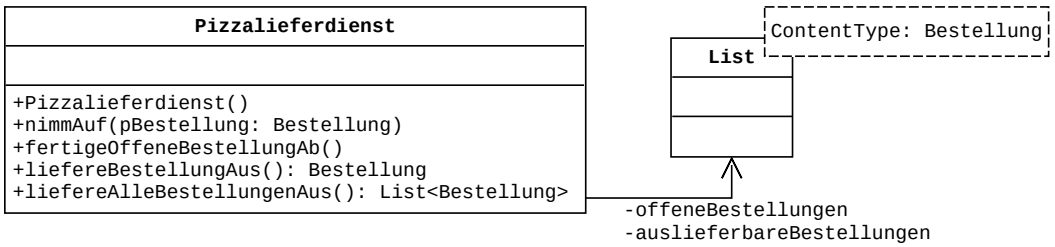
Abiturprüfung 2019

Informatik, Grundkurs

Aufgabenstellung:

Bei dem Start-up-Unternehmen *Pizza2Call* können Kunden Pizza bestellen und sich liefern lassen. Durch ein Informatiksystem sollen Bestellungen noch effizienter verwaltet und abgearbeitet werden können.
Dafür wird jede Bestellung – nach Ermessen der Mitarbeiterin oder des Mitarbeiters – mit einer Priorität versehen und in der Liste `offeneBestellungen` verwaltet. Es stehen lediglich die Prioritäten 1 (niedrig), 2 (mittel) und 3 (hoch) zur Verfügung. Je höher die Priorität einer Bestellung ist, desto weiter vorne wird sie in die Liste eingereiht. Bei gleicher Priorität wird eine neue Bestellung hinter die bereits bestehenden Bestellungen mit dieser Priorität eingereiht.
Wenn die erste Bestellung aus der Liste `offeneBestellungen` abgefertigt bzw. zubereitet wurde, dann wird sie aus der Liste `offeneBestellungen` entfernt und an die Liste `auslieferbareBestellungen` angehängt.
In der ersten Version soll jede Bestellung nur aus jeweils einem Gericht bestehen.

Abbildung 1 zeigt einen Ausschnitt des Implementationsdiagramms des zugehörigen Informatiksystems. Eine Dokumentation relevanter Klassen finden Sie im Anhang.



Bestellung

```
-bestellungID: int
-gericht: String
-anschrift: String
-prioritaet: int
-preis: double
```

```
+Bestellung(pBestellungID: int, pGericht: String, pAnschrift: String,
            pPrioritaet: int, pPreis: double)
+gibBestellungID(): int
+gibGericht(): String
+gibAnschrift(): String
+gibPrioritaet(): int
+gibPreis(): double
+setzePrioritaet(pPrioritaet: int)
```

Abbildung 1: Ausschnitt aus dem Implementationsdiagramm

Name: _____

a) Gehen Sie davon aus, dass die Listenobjekte `offeneBestellungen` und `auslieferbareBestellungen` leer sind.

Folgende Aktionen werden nacheinander durchgeführt:

i. Eine Bestellung aufnehmen:

BestellungID	Gericht	Anschrift	Priorität	Preis in Euro
1	Pizza Salami	Schulstr. 1	2	7,00

ii. Eine Bestellung aufnehmen:

BestellungID	Gericht	Anschrift	Priorität	Preis in Euro
2	Pizza Hawaii	Schulstr. 2	3	8,00

iii. Eine Bestellung aufnehmen:

BestellungID	Gericht	Anschrift	Priorität	Preis in Euro
3	Salat	Bahnhofstr. 10	3	5,00

iv. Eine offene Bestellung abfertigen.

v. Eine Bestellung aufnehmen:

BestellungID	Gericht	Anschrift	Priorität	Preis in Euro
4	Pizzabrötchen	Schulstr. 1	1	3,00

vi. Eine Bestellung ausliefern.

vii. Eine offene Bestellung abfertigen.

viii. Eine offene Bestellung abfertigen.

ix. Alle Bestellungen ausliefern.

Dokumentieren Sie nach jeder der neun genannten Aktionen die Belegungen der Listenobjekte `offeneBestellungen` und `auslieferbareBestellungen`*, indem Sie lediglich die IDs und die Prioritäten der Bestellungen in den jeweiligen Listenobjekten angeben.*

Beschreiben Sie die Beziehungen zwischen den Klassen `Pizzalieferdienst`, `List` *und* `Bestellung`.

(8 Punkte)

Name: _____

b) Die Klasse `Pizzalieferdienst` verfügt über folgende neue Methode:

```
1  private void tueEtwas() {
2    List<Bestellung> neueListe = new List<Bestellung>();
3    auslieferbareBestellungen.toFirst();
4    while (auslieferbareBestellungen.hasAccess()) {
5      Bestellung aktBestellung
6          = auslieferbareBestellungen.getContent();
7      auslieferbareBestellungen.remove();
8      neueListe.toFirst();
9      while (neueListe.hasAccess()
10            && neueListe.getContent().gibAnschrift()
11               .compareTo(aktBestellung.gibAnschrift()) < 0) {
12       neueListe.next();
13     }
14     if (neueListe.hasAccess()) {
15       neueListe.insert(aktBestellung);
16     } else {
17       neueListe.append(aktBestellung);
18     }
19   }
20   auslieferbareBestellungen = neueListe;
21 }
```

Gegeben ist die Liste `auslieferbareBestellungen` mit folgenden Testdaten:

BestellungID	Gericht	Anschrift	Priorität	Preis in Euro
1	Pizza Schinken	Schulstraße 1	3	7,00
5	Pizza Funghi	Kindergartenweg 100	3	7,00
6	Pizza Hawaii	Schulstraße 3	3	8,00
3	Pizzabrötchen	Schulstraße 1	2	3,00
8	Pizza Schinken	Bergweg 80	2	7,00

Abbildung 2: Testdaten für die Liste `auslieferbareBestellungen`

Analysieren Sie die Methode `tueEtwas`*, indem Sie für die Testdaten der Liste* `auslieferbareBestellungen` *die Informationen* `BestellungID`*,* `Anschrift` *und* `Priorität` *nach Methodenausführung bestimmen.*

Erläutern Sie die Funktionsweise der Methode und die Funktionalität der Methode im Sachkontext.

(12 Punkte)

Name: _____

c) Da einzelne Kunden sich über zu lange Wartezeiten beschwert haben, soll eine Möglichkeit geschaffen werden, die Prioritäten von ausgewählten Bestellungen zu erhöhen. In der Klasse `Pizzalieferdienst` wird eine neue Methode `erhoehePrioritaet` benötigt, die wie folgt dokumentiert ist:

```
public void erhoehePrioritaet(int pBestellungID)
```

Die Methode erhöht die Priorität für die Bestellung mit der Bestellnummer `pBestellungID` in der Liste `offeneBestellungen` um den Wert 1.
Die Priorität wird nie größer als 3.

Hinweis: Die Liste `offeneBestellungen` ist auch nach Methodenausführung nach Prioritäten absteigend sortiert.

Entwickeln Sie eine Strategie für die Arbeitsweise der Methode `erhoehePrioritaet` *und stellen Sie diese in geeigneter Form dar.*

Implementieren Sie die Methode `erhoehePrioritaet` *entsprechend Ihrer Strategie.*

(12 Punkte)

d) Das Start-Up-Unternehmen *Pizza2Call* möchte für spätere Werbemaßnahmen die Kundeninformationen (Vorname, Nachname und Telefonnummer) verwalten. Außerdem soll für statistische Zwecke protokolliert werden, welcher Kunde welche Bestellungen getätigt hat.

In der Modellerweiterung müssen folgende Anforderungen berücksichtigt werden:

i. Die Daten (Vorname, Nachname und Telefonnummer) eines Kunden müssen verwaltet werden.
   Alle Kunden müssen verwaltet werden.
ii. Ein neuer Kunde muss angelegt werden können.
iii. Jedem Kunden müssen alle seine Bestellungen zugeordnet werden.
iv. Es sollen alle Kunden ermittelt werden, die mindestens so viele Bestellungen getätigt haben, wie dies durch einen Schwellenwert vorgegeben wird.

Modellieren Sie die oben genannten Anforderungen als Erweiterung des Implementationsdiagramms aus Abbildung 1.
Hinweis: Unveränderte Attribute, Methoden und Assoziationen aus dem Implementationsdiagramm (Abbildung 1) müssen nicht aufgeführt werden.

Erläutern Sie, wie Sie jede der genannten Anforderungen in Ihrem Implementationsdiagramm realisieren.

(11 Punkte)

Name: _____

e) *Nehmen Sie Stellung, inwieweit man für die Verwaltung der offenen Bestellungen statt der Datenstruktur* `List` *die Datenstruktur* `Queue` *nehmen könnte, indem Sie die Vor- und Nachteile der beiden Datenstrukturen im vorliegenden Sachzusammenhang darstellen.*

(7 Punkte)

Zugelassene Hilfsmittel:

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

Name: _____

Anhang

Dokumentationen der verwendeten Klassen

Die Klasse String

Ausschnitt aus der Dokumentation der Klasse `String`

Anfrage **`compareTo(String pAnotherString) : int`**
Diese Anfrage liefert den Wert `0`, wenn die Strings lexikografisch identisch sind. Sie liefert einen Wert kleiner als `0`, wenn `pAnotherString` lexikografisch größer als der String ist. Sie liefert einen Wert größer als `0`, wenn `pAnotherString` lexikografisch kleiner als der String ist.

Die Klasse Bestellung

Ein Objekt der Klasse `Bestellung` verwaltet die Informationen einer Bestellung.

Ausschnitt aus der Dokumentation der Klasse `Bestellung`

Konstruktor
```
Bestellung(int pBestellungID,
           String pGericht, String pAnschrift,
           int pPrioritaet, double pPreis)
```
Ein Bestellungsobjekt mit den im Parameter übergebenen Werten wird erzeugt.

Anfrage **`int gibBestellungID()`**
Die ID der Bestellung wird zurückgeliefert.

Anfrage **`String gibGericht()`**
Der Name des Gerichts wird zurückgeliefert.

Anfrage **`String gibAnschrift()`**
Die Lieferanschrift wird zurückgeliefert.

Anfrage **`int gibPrioritaet()`**
Die Priorität wird zurückgeliefert.

Anfrage **`double gibPreis()`**
Der Preis wird zurückgeliefert.

Auftrag **`void setzePrioritaet(int pPrioritaet)`**
Die Priorität wird auf den im Parameter übergebenen Wert gesetzt.

Name: _____

Die Klasse Pizzalieferdienst

Ein Objekt der Klasse `Pizzalieferdienst` dient zur Verwaltung der Bestellungen.

Ausschnitt aus der Dokumentation der Klasse `Pizzalieferdienst`

Konstruktor **`Pizzalieferdienst()`**
Ein neues Pizzalieferdienstobjekt wird erzeugt. Die Listen `offeneBestellungen` und `auslieferbareBestellungen` werden initialisiert und sind leer.

Auftrag **`void nimmAuf(Bestellung pBestellung)`**
Das Objekt `pBestellung` wird anhand seiner Priorität absteigend in die Liste `offeneBestellungen` eingereiht. Bei gleicher Priorität wird das Objekt `pBestellung` hinter die bestehenden Bestellungen mit dieser Priorität eingereiht.

Auftrag **`void fertigeOffeneBestellungAb()`**
Das erste Objekt aus der Liste `offeneBestellungen` wird entfernt und an die Liste `auslieferbareBestellungen` angehängt.
Falls die Liste `offeneBestellungen` leer ist, geschieht nichts.

Anfrage **`Bestellung liefereBestellungAus()`**
Die Methode liefert das vorderste Objekt aus der Liste `auslieferbareBestellungen` zurück und entfernt dieses aus der Liste.
Falls die Liste `auslieferbareBestellungen` leer ist, wird `null` zurückgegeben.

Anfrage **`List<Bestellung> liefereAlleBestellungenAus()`**
Die Methode liefert alle Objekte aus der Liste `auslieferbareBestellungen` zurück und entfernt diese aus der Liste.
Falls die Liste `auslieferbareBestellungen` leer ist, wird eine leere Liste zurückgeliefert.

Name: _____

Die generische Klasse List<ContentType>

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

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

Dokumentation der Klasse `List<ContentType>`

Konstruktor **`List()`**

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

Anfrage **`boolean isEmpty()`**

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

Anfrage **`boolean hasAccess()`**

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

Auftrag **`void next()`**

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

Auftrag **`void toFirst()`**

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

Auftrag **`void toLast()`**

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

Name: _____

Anfrage **`ContentType getContent()`**

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

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

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

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

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

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

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

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

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

Auftrag **`void remove()`**

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

Unterlagen für die Lehrkraft

Abiturprüfung 2019

Informatik, Grundkurs

1. Aufgabenart

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

2. Aufgabenstellung[1]

siehe Prüfungsaufgabe

3. Materialgrundlage

entfällt

4. Bezüge zum Kernlehrplan und zu den Vorgaben 2019

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

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

   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 (grafikfähiger 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)

Zustand der Listen nach `i`:
`offeneBestellungen`:
(Listenanfang) ➔ ID 1, Prio: 2 ➔
`auslieferbareBestellungen`:
(Listenanfang) ➔

Zustand der Listen nach `ii`:
`offeneBestellungen`:
(Listenanfang) ➔ ID 2, Prio: 3 ➔ ID 1, Prio: 2 ➔
`auslieferbareBestellungen`:
(Listenanfang) ➔

Zustand der Listen nach `iii`:
`offeneBestellungen`:
(Listenanfang) ➔ ID 2, Prio: 3 ➔ ID 3, Prio: 3 ➔ ID 1, Prio: 2 ➔
`auslieferbareBestellungen`:
(Listenanfang) ➔

Zustand der Listen nach `iv`:
`offeneBestellungen`:
(Listenanfang) ➔ ID 3, Prio: 3 ➔ ID 1, Prio: 2 ➔
`auslieferbareBestellungen`:
(Listenanfang) ➔ ID 2, Prio: 3 ➔

Zustand der Listen nach `v`:
`offeneBestellungen`:
(Listenanfang) ➔ ID 3, Prio: 3 ➔ ID 1, Prio: 2 ➔ ID 4, Prio: 1 ➔
`auslieferbareBestellungen`:
(Listenanfang) ➔ ID 2, Prio: 3 ➔

Zustand der Listen nach `vi`:
`offeneBestellungen`:
(Listenanfang) ➔ ID 3, Prio: 3 ➔ ID 1, Prio: 2 ➔ ID 4, Prio: 1 ➔
`auslieferbareBestellungen`:
(Listenanfang) ➔

Zustand der Listen nach `vii`:
`offeneBestellungen`:
(Listenanfang) ➔ ID 1, Prio: 2 ➔ ID 4, Prio: 1 ➔
`auslieferbareBestellungen`:
(Listenanfang) ➔ ID 3, Prio: 3 ➔

Zustand der Listen nach `viii`:
`offeneBestellungen`:
(Listenanfang) ➔ ID 4, Prio: 1 ➔
`auslieferbareBestellungen`:
(Listenanfang) ➔ ID 3, Prio: 3 ➔ ID 1, Prio: 2 ➔

Zustand der Listen nach `ix`:
`offeneBestellungen`:
(Listenanfang) ➔ ID 4, Prio: 1 ➔
`auslieferbareBestellungen`:
(Listenanfang) ➔

Ein Objekt der Klasse `Pizzalieferdienst` verwaltet in der linearen Liste `offeneBestellungen` Objekte vom Typ `Bestellung`.
Ein Objekt der Klasse `Pizzalieferdienst` verwaltet in der linearen Liste `auslieferbareBestellungen` Objekte vom Typ `Bestellung`.

Teilaufgabe b)

Belegung der Liste `auslieferbareBestellungen` nach Methodenausführung:

BestellungID	Anschrift	Priorität
8	Bergweg 80	2
5	Kindergartenweg 100	3
3	Schulstraße 1	2
1	Schulstraße 1	3
6	Schulstraße 3	3

Abbildung 1: Belegung der Liste `auslieferbareBestellungen` nach Methodenausführung

Die Methode `tueEtwas` bekommt keine Parameter übergeben und liefert nichts zurück.
Zunächst wird in der Methode eine neue lokale Liste namens `neueListe` deklariert und initialisiert (vgl. Zeile 2).
Die äußere Schleife läuft, solange sie Zugriff auf ein aktuelles Listenobjekt der Liste `auslieferbareBestellungen` hat (vgl. Zeilen 3 – 4): Die vorderste Bestellung wird unter der lokalen Variablen `aktBestellung` gespeichert und aus der Liste `auslieferbareBestellungen` entfernt (vgl. Zeilen 5 – 7).

Die innere Schleife (vgl. Zeilen 9 – 13) durchläuft die Liste `neueListe`. Damit wird die richtige Einfügeposition für `aktBestellung` gesucht. Wenn das aktuelle Bestellungsobjekt aus der Liste `neueListe` eine im Sinne der lexikografischen Ordnung kleinere Adresse hat als die des Objektes `aktBestellung` (vgl. Zeilen 10 – 11), dann wird das nächste Listenobjekt der Liste `neueListe` zum aktuellen Listenobjekt.
Wenn man Zugriff auf ein aktuelles Listenobjekt der Liste `neueListe` hat (vgl. Zeile 14), dann wird `aktBestellung` vor diesem eingefügt (vgl. Zeile 15). Sonst wird `aktBestellung` an die Liste `neueListe` angehängt (vgl. Zeile 17).
Der Referenz `auslieferbareBestellungen` wird die Referenz des Objekts `neueListe` zugewiesen (vgl. Zeile 20).

Die Methode sortiert – im Sinne von Insertionsort – im Sachkontext die fertigen Bestellungen lexikografisch aufsteigend nach der Anschrift.

Teilaufgabe c)

Strategie:
- Die Bestellung mit der gesuchten Bestellnummer muss aus der Liste `offeneBestellungen` gesucht, in einer lokalen Variablen gespeichert werden.
- Wenn die Bestellung vorhanden ist und die Priorität kleiner als 3 ist:
  - Die Priorität bei dem entsprechenden Bestellungsobjekt wird um 1 erhöht.
  - Das entsprechende Bestellungsobjekt wird aus der Liste `offeneBestellungen` entfernt und mithilfe der vorgegebenen Methode `nimmAuf` in die Liste `offeneBestellungen` eingefügt.

Implementierung:
```
public void erhoehePrioritaet(int pBestellungID) {
  offeneBestellungen.toFirst();
  while (offeneBestellungen.hasAccess()
         && offeneBestellungen.getContent().gibBestellungID()
            != pBestellungID) {
    offeneBestellungen.next();
  }
  Bestellung aktuelleBestellung = offeneBestellungen.getContent();
  if (aktuelleBestellung != null
      && aktuelleBestellung.gibPrioritaet() < 3) {
    offeneBestellungen.remove();
    aktuelleBestellung.setzePrioritaet(
        aktuelleBestellung.gibPrioritaet() + 1);
    nimmAuf(aktuelleBestellung);
  }
}
```

Teilaufgabe d)

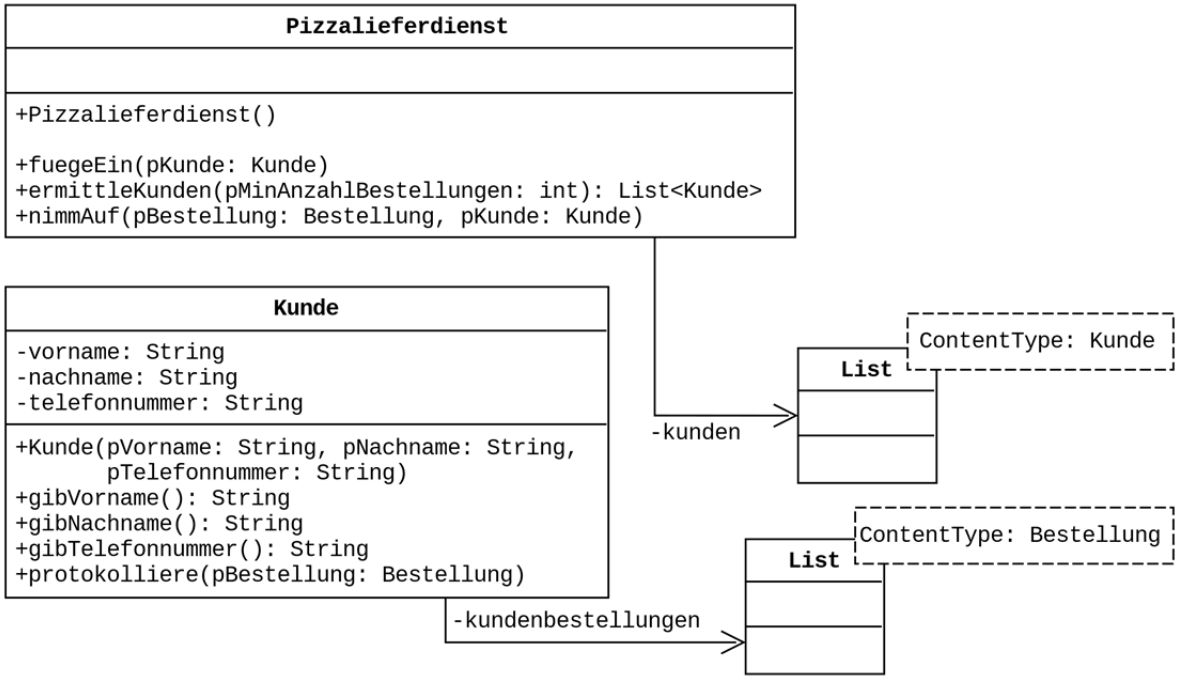


Abbildung 2: Erweiterung des Implementationsdiagramms

Zu i) Der Vorname, der Nachname und die Telefonnummer eines Kunden werden mithilfe der Klasse `Kunde` verwaltet.
Die Klasse `Pizzalieferdienst` verwaltet alle Kunden in dem Objekt `kunden` vom Typ `List`.

Zu ii) Die Klasse `Pizzalieferdienst` enthält die Methode `fuegeEin`.

Zu iii) Die Klasse `Kunde` verwaltet die eigenen Bestellungen in dem Objekt `kundenbestellungen` vom Typ `List`.
Außerdem wird in der Klasse `Pizzalieferdienst` die Methode `nimmAuf` ersetzt. Die neue Methode enthält die zwei Parameter `pBestellung` und `pKunde`.
Zusätzlich enthält die Klasse `Kunde` die Methode `protokolliere`, um eine neue Bestellung an die Liste `kundenbestellungen` anzuhängen.

Zu iv) Die Verwaltung stellt die Methode `ermittleKunden` zur Verfügung, die eine Liste mit Objekten von der Klasse `Kunde` zurückliefert. Als Parameter wird der Schwellenwert `pMinAnzahlBestellungen` übergeben.

Teilaufgabe e)

Das Listenobjekt `offeneBestellungen` stellt eine Prioritätswarteschlange dar. Die Bestellungen werden aufsteigend sortiert nach Prioritäten eingefügt. Dies erfordert die Möglichkeit zum Einfügen an jeder Stelle der linearen Datenstruktur. Dafür eignet sich insbesondere die Datenstruktur `List`. Eine vollständige Ausgabe ist mit einem einfachen Durchlauf durch die Liste möglich.

Alternativ dazu werden zwei mögliche Varianten unter Verwendung der Klasse `Queue` genannt, die nach dem FIFO-Prinzip arbeitet:

1. Alternative: Eine einzige `Queue`
Bei einer `Queue` ist ein Einfügen am Ende der Schlange effizient möglich. Ein Einfügen an einer anderen Stelle ist nicht vorgesehen, ließe sich allerdings mit zusätzlichem Aufwand dennoch realisieren. Z. B. könnte man dabei wie folgt vorgehen: Man müsste für das Finden der richtigen Einfügestelle die Queue sukzessive abbauen und die entfernten Objekte in einer lokalen Hilfsschlange speichern und anschließend das neue Bestellungsobjekt einfügen und die Schlangen wieder vereinen.

2. Alternative: Pro Prioritätsstufe eine `Queue`
Falls die Anzahl der Prioritätsstufen im Vorfeld festgelegt und begrenzt ist, könnte man für jede Prioritätsstufe ein `Queue`-Objekt vorsehen. Beim Einfügen wird die neue Bestellung sofort an das richtige `Queue`-Objekt hinten angehängt. Das Suchen der richtigen Einfügestelle, wie bei der `List`, würde entfallen.
Beim Auslesen bzw. Löschen einer Bestellung würde man sukzessive die `Queue`-Objekte abarbeiten und das vorderste Bestellungsobjekt aus dem `Queue`-Objekt mit der höchsten Priorität suchen bzw. löschen.
Wenn man eine oder mehrere Schlangen ausgeben möchte, müsste man die Datenstruktur sukzessive abbauen, das aktuelle Element ausgeben und anschließend die Datenstruktur wieder aufbauen, was mit einem Zusatzaufwand verbunden wäre.

Die Verwaltung der offenen Bestellungen im Sinne einer Prioritätswarteschlange mit einer Liste erscheint geeigneter als die Verwendung mehrerer `Queue`-Objekte oder eines `Queue`-Objektes. Insbesondere das problemlose Hinzufügen von weiteren Prioritätsstufen, die komfortablere Möglichkeit zum Sortieren und die einfache Ausgabe sind überzeugende Argumente für die verwendete Datenstruktur `List`.

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	dokumentiert die Belegungen der Listenobjekte `offeneBestellungen` und `auslieferbareBestellungen`.	6			
2	beschreibt die Beziehungen zwischen den Klassen `Pizzalieferdienst`, `List` und `Bestellung`.	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 die Methode und bestimmt die Belegung der Liste `auslieferbareBestellungen` nach Methoden-ausführung.	4			
2	erläutert die Funktionsweise der Methode.	6			
3	erläutert die Funktionalität der Methode im Sachkontext.	2			
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	entwickelt eine Strategie für die Arbeitsweise der Methode und stellt diese in geeigneter Form dar.	5			
2	implementiert die Methode.	7			
Sachlich richtige Lösungsalternative zur Modelllösung: (12) …… ……					
	Summe Teilaufgabe c)	**12**			

Teilaufgabe d)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	modelliert die Erweiterung als Implementationsdiagramm.	6			
2	erläutert, wie jede Anforderung im Implementations-diagramm realisiert wird.	5			
Sachlich richtige Lösungsalternative zur Modelllösung: (11) …… ……					
	Summe Teilaufgabe d)	**11**			

Teilaufgabe e)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	nimmt Stellung, inwieweit man statt der Datenstruktur `List` die Datenstruktur `Queue` für die Verwaltung der offenen Bestellungen nehmen könnte, indem er die Vor- und Nachteile der beiden Datenstrukturen im vorliegenden Sachzusammenhang darstellt.	7			
Sachlich richtige Lösungsalternative zur Modelllösung: (7) …… ……					
	Summe Teilaufgabe e)	**7**			

	Summe insgesamt	**50**			

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

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

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

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

Unterschrift, Datum:

Grundsätze für die Bewertung (Notenfindung)

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

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

Name: _____

Abiturprüfung 2019

Informatik, Grundkurs

Aufgabenstellung:

Um in den Markt für Lernspiele einzusteigen, möchte ein kleines Softwareunternehmen ein Spiel zur Übung des Kopfrechnens entwickeln.
Beim Spielstart wird ein zufällig zusammengestellter Spielbaum gezeigt, in dessen Blättern ganze Zahlen und in dessen inneren Knoten Rechenzeichen, d. h. `+`, `-`, `*` und `/` stehen. Das Zeichen `/` steht dabei für eine ganzzahlige Division.
Abbildung `1` zeigt einen exemplarischen Spielbaum. Die Großbuchstaben dienen der Bezeichnung der Knoten.

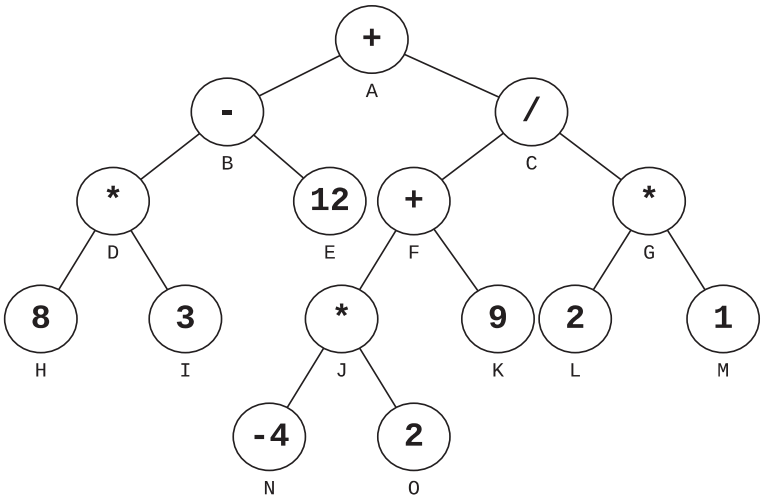


Abbildung 1: Exemplarischer Spielbaum mit alphabetischer Knotenbezeichnung

Für jeden Knoten des Spielbaums kann, wie folgt, ein Punktwert ermittelt werden: Handelt es sich bei dem aktuellen Knoten um ein Blatt, so wird der Wert, der in diesem Blatt eingetragen ist, genommen. Handelt es sich bei dem aktuellen Knoten um einen inneren Knoten, so wird der Wert berechnet, indem die Punktwerte seiner Kinder mit dem Rechenzeichen im aktuellen Knoten verknüpft werden.
Steht im aktuellen Knoten z. B. ein –, muss der Werte seines rechten Teilbaums vom Wert seines linken Teilbaums abgezogen werden. Dabei ist zu bedenken, dass die Werte der Teilbäume ggf. erst nach dem gleichen Verfahren ermittelt werden müssen. Der Punktwert der Wurzel des Spielbaums wird als *Spielwert* des Baums bezeichnet.

Name: _____

Die Spielerin bzw. der Spieler hat nun die Möglichkeit, zwei innere Knoten auszuwählen und die Rechenzeichen in diesen Knoten miteinander zu vertauschen. Diesen Tausch zweier Rechenzeichen bezeichnet man als *Spielzug*. Das Ziel des Spiels ist es, mit einem Spielzug einen Spielbaum mit einem möglichst hohen Spielwert zu erzeugen.

a) *Bestimmen Sie schrittweise den Spielwert für den in Abbildung 1 gegebenen Spielbaum, indem Sie die Werte für jeden Knoten angeben.*

Ermitteln Sie einen Spielzug, der den Spielwert des in Abbildung 1 gegebenen Spielbaums um mindestens 10 Punkte verbessert.

Erläutern Sie, warum jeder Knoten eines gültigen Spielbaums genau zwei oder keinen Nachfolger haben muss.

(8 Punkte)

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

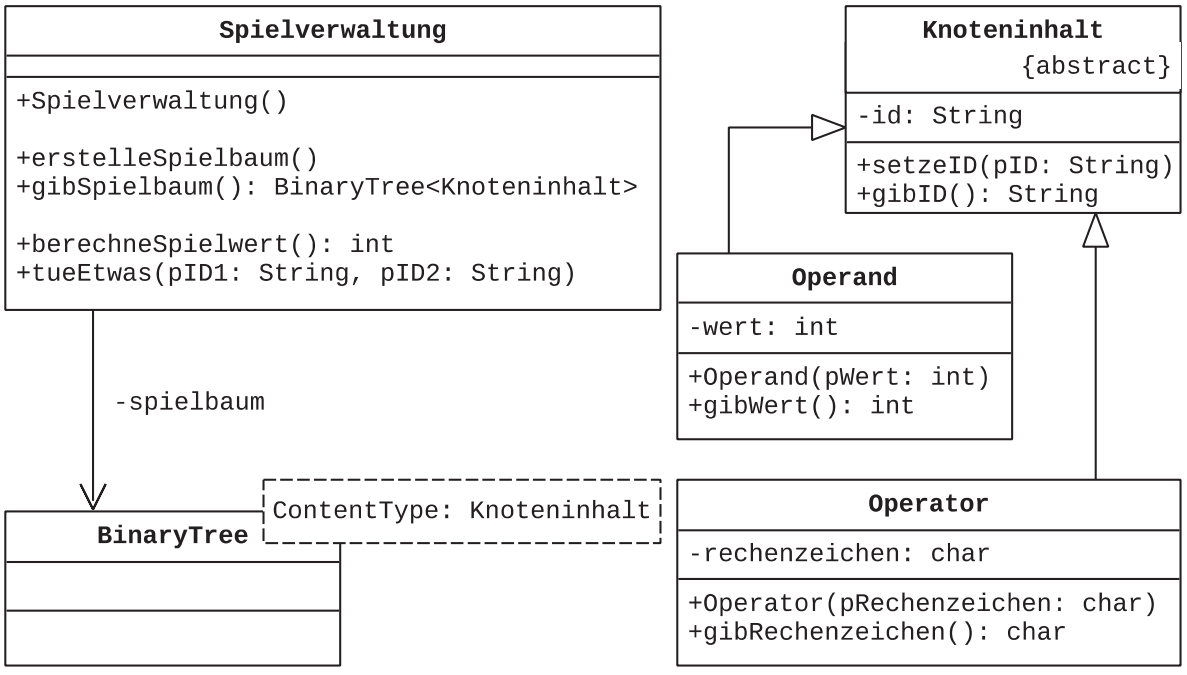


Abbildung 2: Teilmodellierung des Spiels

Name: _____

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

Erläutern Sie, was man unter einer abstrakten Klasse versteht.

Begründen Sie, warum die Klasse `Knoteninhalt` *abstrakt modelliert ist.*

(12 Punkte)

c) Die Methode `berechneSpielwert` der Klasse `Spielverwaltung` berechnet den Spielwert des Baums `spielbaum`.

Die Methode hat den folgenden Methodenkopf:

```
public int berechneSpielwert()
```

Entwickeln Sie für die Methode `berechneSpielwert` *und eventuelle Hilfsmethoden eine algorithmische Strategie.*

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

(12 Punkte)

Name: _____

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

```
 1  public void tueEtwas(String pID1, String pID2) {
 2    Queue<BinaryTree<Knoteninhalt>> speicher =
 3                          new Queue<BinaryTree<Knoteninhalt>>();
 4    BinaryTree<Knoteninhalt> k1 = null;
 5    BinaryTree<Knoteninhalt> k2 = null;
 6
 7    speicher.enqueue(spielbaum);
 8    while (!speicher.isEmpty()) {
 9      BinaryTree<Knoteninhalt> akt = speicher.front();
10      speicher.dequeue();
11
12      if (!akt.getLeftTree().isEmpty()
13                            && !akt.getRightTree().isEmpty()) {
14        speicher.enqueue(akt.getLeftTree());
15        speicher.enqueue(akt.getRightTree());
16      }
17
18      if (akt.getContent().gibID().equals(pID1)) {
19        k1 = akt;
20      }
21      if (akt.getContent().gibID().equals(pID2)) {
22        k2 = akt;
23      }
24    }
25
26    Knoteninhalt tmp = k1.getContent();
27    k1.setContent(k2.getContent());
28    k2.setContent(tmp);
29  }
```

Analysieren und erläutern Sie die Methode `tueEtwas`.

Erläutern Sie im Sachzusammenhang, was die Methode `tueEtwas` *leistet.*

(10 Punkte)

Name: _____

e) Nach Fertigstellung des ersten Prototyps für das Lernspiel meldet sich ein Mitarbeiter mit der Anmerkung, dass man auch eine ganz andere Modellierung hätte verwenden können. Diese Modellierung ist in Abbildung 3 gegeben.

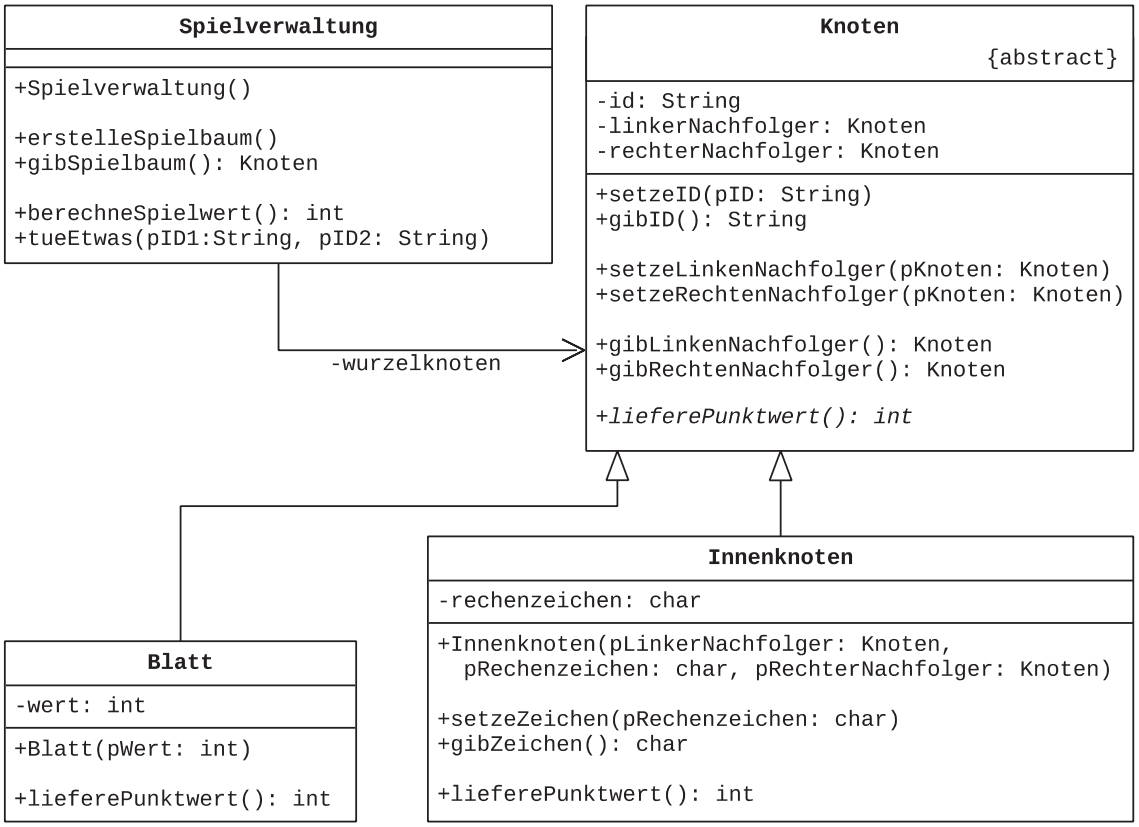


Abbildung 3: Alternativmodellierung des Spiels

Analysieren und vergleichen Sie die in Abbildung 3 gegebene Alternativmodellierung mit der aus Abbildung 2.

Nehmen Sie Stellung zur Alternativmodellierung in Abbildung 3.

(8 Punkte)

Zugelassene Hilfsmittel:

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

Name: _____

Anhang

Dokumentationen der verwendeten Klassen

Dokumentation der Klasse Spielverwaltung

Konstruktor **`Spielverwaltung()`**
Initialisiert ein neues Objekt vom Typ `Spielverwaltung` und erstellt mit Hilfe der Methode `erstelleSpielbaum` einen neuen Spielbaum.

Auftrag **`void erstelleSpielbaum()`**
Erstellt einen neuen Spielbaum.

Anfrage **`BinaryTree<Knoteninhalt> gibSpielbaum()`**
Liefert den aktuellen Spielbaum zurück.

Anfrage **`int berechneSpielwert()`**
Liefert den Spielwert des aktuellen Spielbaums zurück.

Auftrag **`void tueEtwas(String pID1, String pID2)`**
Diese Methode soll in Teilaufgabe d) analysiert werden.

Die Klasse Knoteninhalt

Die abstrakte Klasse `Knoteninhalt` wird zu Modellierung der Inhalte des Spielbaums benötigt.

Dokumentation der abstrakten Klasse Knoteninhalt

Auftrag **`void setzeID(String pID)`**
Setzt die ID des Knotens auf `pID`.

Anfrage **`String gibID()`**
Liefert die ID des Knotens.

Name: _____

Die Klasse Operand

Die Klasse `Operand` modelliert die Zahlenwerte in den Blättern des Spielbaums.

Dokumentation der Klasse Operand

Konstruktor **`Operand(int pWert)`**
Initialisiert ein neues Objekt vom Typ `Operand` mit dem Wert `pWert`.

Anfrage **`int gibWert()`**
Liefert den Wert des Blatts.

Die Klasse Operator

Die Klasse `Operator` modelliert die Rechenzeichen in den inneren Knoten des Spielbaums.

Dokumentation der Klasse Operator

Konstruktor **`Operator(char pRechenzeichen)`**
Initialisiert ein neues Objekt vom Typ `Operator` mit dem Rechenzeichen `pRechenzeichen`.

Anfrage **`int gibRechenzeichen()`**
Liefert das Rechenzeichen 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 2019

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 2019

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

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

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

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

2. *Medien/Materialien*
   - entfällt

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

5. Zugelassene Hilfsmittel

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

6. Modelllösungen

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

Teilaufgabe a)

Der Spielwert des Baums in Abbildung 1 kann wie folgt berechnet werden:

```
H = 8, I = 3
D = H * I = 24
E = 12
B = D - E = 12

N = -4, O = 2
J = N * O = -8
K = 9
F = J + K = 1
L = 2, M = 1
G = L * M = 2
C = F / G = 0

A = B + C = 12
```

Da der Knoten `A` die Wurzel des Spielbaums ist, hat er insgesamt den Spielwert `12`.

Ein Spielzug, der den Spielwert des Baums um mindesten `10` Punkte verbessert, ist z. B. der Tausch der Rechenzeichen in den Knoten `B` und `D`. Daraus ergibt sich:

```
D = H – I = 5
B = D * E = 60
A = B – C = 60
```

Der Spielwert des Baums hat sich durch diesen Zug also von `12` auf `60` Punkte verbessert.

In einem korrekten Spielbaum müssen innere Knoten immer zwei Nachfolger haben, da in ihnen laut Aufgabenstellung immer ein binärer Operator, d. h. `+`, `-`, `*` oder `/` steht und dieser daher auf zwei Werte angewendet werden muss. Alle anderen Knoten sind Blätter. Ein innerer Knoten mit nur einem Nachfolger könnte daher im Aufgabenkontext nicht sinnvoll ausgewertet werden.

Teilaufgabe b)

Die in Abbildung 2 gegebene Modellierung besteht aus fünf Klassen. Die Klasse `Spielverwaltung` stellt allgemeine Dienste zur Verwaltung eines Spielbaums zur Verfügung. Dazu gehört das Erstellen eines Spielbaums (`erstelleSpielbaum`), das Liefern des aktuellen Spielbaums (`gibSpielbaum`) und das Berechnen des Spielwertes des aktuellen Spielbaums (`berechneSpielwert`). Intern speichert ein Objekt der Klasse `Spielverwaltung` den aktuellen Spielbaum unter dem Bezeichner `spielbaum` als Binärbaum vom Typ `BinaryTree`. Dabei werden Objekte vom Typ `Knoteninhalt` als Inhaltsobjekte verwendet.
Die Klasse `Knoteninhalt` ist eine abstrakte Oberklasse für die Klassen `Operand` und `Operator`. Sie modelliert das Attribut `id` als `String` und stellt eine Methode zum Setzen (`setzeID`) und eine Methode zum Abfragen (`gibID`) dieses Attributs zur Verfügung.
Die Klasse `Operand` erbt von der Klasse `Knoteninhalt` und ergänzt das Attribut `wert` vom Typ `int`. Dieses Attribut kann im Konstruktor gesetzt und mit der Methode `gibWert` abgerufen werden.
Die Klasse `Operator` erbt ebenfalls von der Klasse `Knoteninhalt` und ergänzt das Attribut `rechenzeichen` vom Typ `char`. Wie auch in der Klasse `Operand` kann dieses Attribut im Konstruktor gesetzt und mit einer entsprechenden Methode (`gibRechenzeichen`) abgerufen werden.

Unter einer abstrakten Klasse versteht man eine Klasse, aus der keine Objekte erstellt werden können. Ihre Aufgabe ist es, im Sinne der Generalisierung als Oberklasse für andere Klassen zu dienen.

Die Klasse `Knoteninhalt` ist abstrakt modelliert, da es im Kontext der gegebenen Modellierung keinen Sinn macht, einen Baumknoten zu erstellen, der nicht entweder ein Objekt vom Typ `Operand` oder ein Objekt vom Typ `Operator` als Inhaltsobjekt bekommt.

Teilaufgabe c)

Eine algorithmische Strategie für die Methode `berechneSpielwert` ist die folgende:

Um den Spielwert des Spielbaums zu berechnen, muss der Punktwert der Wurzel des Baums `spielbaum` berechnet werden. Dazu wird eine rekursiv arbeitende Hilfsmethode verwendet, die den Punktwert desjenigen Teilbaums berechnet, der im Parameter übergeben wird. Diese Hilfsmethode prüft zuerst, ob der im Parameter übergebene Teilbaum ein Blatt ist. Ist das der Fall, wird der Punktwert dieses Blatts zurückgeliefert. Ansonsten werden die Punktwerte der beiden Teilbäume durch rekursive Aufrufe ermittelt und anschließend abhängig vom Rechenzeichen im aktuellen Knoten verknüpft. Das Ergebnis wird dann zurückgeliefert.

Die Methode `berechneSpielwert` und ihre Hilfsmethode können wie folgt entsprechend der obigen Strategie implementiert werden:

```
public int berechneSpielwert() {
  return berechnePunktwert(spielbaum);
}

private int berechnePunktwert(BinaryTree<Knoteninhalt> pBaum) {
  int ergebnis = 0;
  if (pBaum.getLeftTree().isEmpty()
                              && pBaum.getRightTree().isEmpty()) {
    ergebnis = ((Operand) pBaum.getContent()).gibWert();
  } else {
    int l = berechnePunktwert(pBaum.getLeftTree());
    int r = berechnePunktwert(pBaum.getRightTree());
    char zeichen = ((Operator)
                           pBaum.getContent()).gibRechenzeichen();
    switch (zeichen) {
      case '+':  ergebnis = l + r; break;
      case '-':  ergebnis = l - r; break;
      case '*':  ergebnis = l * r; break;
      case '/':  ergebnis = l / r; break;
    }
  }
  return ergebnis;
}
```

Teilaufgabe d)

Bei der Methode `tueEtwas` handelt es sich um einen Auftrag ohne Rückgabewert. Sie bekommt beim Aufruf Werte für die Strings `pID1` und `pID2` übergeben.
In den Zeilen `2` bis `5` werden eine Schlange für Objekte vom Typ `BinaryTree` mit Inhaltsobjekten vom Typ `Knoteninhalt` (`speicher`) und zwei Hilfsvariablen für Objekte vom Typ `BinaryTree` mit `Knoteninhalt` als Inhaltsobjekt (`k1` und `k2`) erstellt.
Anschließend (Z. `7`) wird die Wurzel des Baums `spielbaum` in die Schlange `speicher` eingefügt. Dann wird von Zeile `8` bis Zeile `24` eine Wiederholung durchlaufen, solange die Schlange `speicher` nicht leer ist. In dieser Wiederholung wird das erste Element der Schlange entnommen und unter dem Bezeichner `akt` gespeichert (Z. `9f`). Anschließend werden beide Kinder dieses Elements an die Schlange `speicher` angefügt, sofern es sie gibt (Z. `12-16`). In den Zeilen `18` bis `23` wird geprüft, ob der Knoten in `akt` eine ID hat, die einer der beiden IDs entspricht, die in den Parametern der Methode übergeben wurden. Ist das der Fall, wird der entsprechende Baumknoten in `k1` oder in `k2` gespeichert.
Ist die Wiederholung beendet, werden in den Zeilen `26` bis `28` die Inhaltselemente von `k1` und `k2` getauscht.

Die Methode führt also eine iterative Traversierung des Spielbaums durch und sucht nach den beiden Knoten, deren IDs in den Parametern übergeben wurden, um deren Inhalte zu tauschen. Im Sachzusammenhang kann diese Methode also verwendet werden, um einen Spielzug entsprechend der Aufgabenstellung durchzuführen.

Teilaufgabe e)

Die Modellierung in Abbildung 3 besteht aus vier statt fünf Klassen, da sie zur Modellierung des Spielbaums auf die Verwendung der Klasse `BinaryTree` verzichtet. Stattdessen speichert die Klasse `Spielverwaltung` direkt ein Objekt vom Typ `Knoten` als Wurzel des Spielbaums.
Die Klasse `Knoten` verwaltet neben der ID nun auch noch einen linken und einen rechten Nachfolgeknoten und verfügt über Methoden zum Setzen und Abfragen beider Nachfolgeknoten. Des Weiteren schreibt die abstrakte Klasse `Knoten` ihren Unterklassen die Implementation der Methode `lieferePunktwert` vor.
Die Unterklasse `Blatt` übernimmt die Funktion der Klasse `Operand` in der ursprünglichen Modellierung und realisiert neben dieser Methode `lieferePunktwert` noch einen Konstruktor, in dem der Wert übergeben werden kann. Die Unterklasse `Innenknoten` bekommt im Konstruktor neben einem Rechenzeichen zusätzlich ihre beiden Nachfolgeknoten übergeben. Sie kann ihr Rechenzeichen nicht nur liefern, sondern auch neu setzen. Sie übernimmt die Funktion der Klasse `Operator` in der alten Modellierung.

Die alternative Modellierung verzichtet auf die Verwendung der Klasse `BinaryTree` und modelliert stattdessen die Baumstruktur in der Klasse `Knoten`. Damit vermeidet sie die im Kontext dieser Aufgabe unnötige Unterscheidung zwischen einem Baumelement und einem Inhaltselement und verzichtet gänzlich auf die Generik der Klasse `BinaryTree`.
Mit der Methode `lieferePunktwert` können Knoten nun deutlich einfacher selbst ihren Punktwert ermitteln, da sie entweder ihren eigenen Punktwert liefern oder diesen durch den Zugriff auf ihre Nachfolgeknoten berechnen können.
Gegen die alternative Modellierung spricht, dass Teile der Funktionalität der Klasse `BinaryTree` neu implementiert werden müssen.

Insgesamt handelt es sich um eine legitime Alternative zum ursprünglichen Ansatz in Abbildung 2.

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	bestimmt schrittweise den Spielwert für den Spielbaum, indem er die Werte für jeden inneren Knoten angibt.	3			
2	ermittelt einen Spielzug, der den Spielwert des Spielbaums um mindestens 10 Punkte verbessert.	3			
3	erläutert, warum jeder Knoten eines gültigen Spielbaums genau zwei oder keinen Nachfolger 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 die Modellierung des Spiels und erläutert die Klassen und ihre Beziehungen zueinander.	8			
2	erläutert, was man unter einer abstrakten Klasse versteht.	2			
3	begründet, warum die Klasse `Knoteninhalt` abstrakt modelliert ist.	2			
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	entwickelt für die Methode `berechneSpielwert` und eventuelle Hilfsmethoden eine algorithmische Strategie.	5			
2	implementiert die Methode `berechneSpielwert` und eventuelle Hilfsmethoden entsprechend seiner Strategie.	7			
Sachlich richtige Lösungsalternative zur Modelllösung: (12) …… ……					
	Summe Teilaufgabe c)	**12**			

Teilaufgabe d)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	analysiert und erläutert die Methode `tueEtwas`.	7			
2	erläutert im Sachzusammenhang, was die Methode `tueEtwas` leistet.	3			
Sachlich richtige Lösungsalternative zur Modelllösung: (10) …… ……					
	Summe Teilaufgabe d)	**10**			

Teilaufgabe e)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	analysiert und vergleicht die gegebene Alternativ-modellierung mit der aus Abbildung 2.	4			
2	nimmt Stellung zur Alternativmodellierung.	4			
Sachlich richtige Lösungsalternative zur Modelllösung: (8) …… ……					
	Summe Teilaufgabe e)	**8**			

	Summe insgesamt	**50**			

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

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

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

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

Unterschrift, Datum:

Grundsätze für die Bewertung (Notenfindung)

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

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

Name: _____

Abiturprüfung 2019

Informatik, Grundkurs

Aufgabenstellung:

Ein Großteil des Handels mit Computerspielen läuft heutzutage über sogenannte Online-Vertriebsplattformen. Hat man sich bei dem entsprechenden Anbieter als Benutzerin oder Benutzer angemeldet, kann man aus einem großen Angebot von Spielen auswählen. Hat man ein Spiel gekauft, kann man seine Kopie des Spiels mit Hilfe eines Internetzugangs beliebig oft auf seinem Rechner installieren bzw. deinstallieren.
Im Folgenden soll eine relationale Datenbank für eine einfache Vertriebsplattform für Computerspiele entworfen werden. Abbildung 1 zeigt eine Teilmodellierung dieser Datenbank.

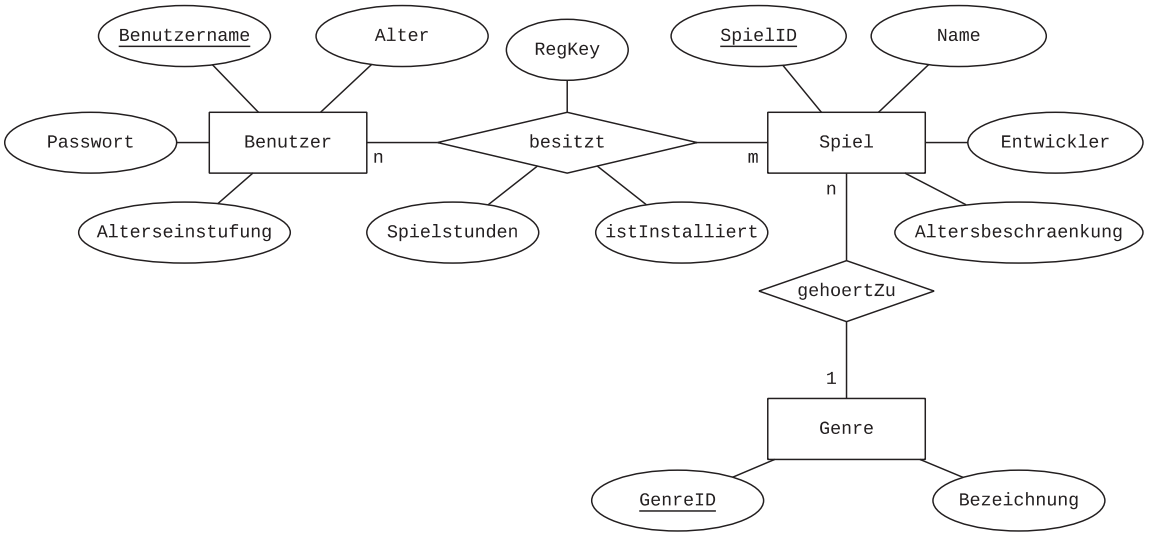


Abbildung 1: Teilmodellierung einer Vertriebsplattform für Computerspiele

Das folgende Datenbankschema setzt die Modellierung aus Abbildung 1 um:

```
Benutzer(Benutzername, Passwort, Alterseinstufung, Alter)
besitzt(↑Benutzername, ↑SpielID, RegKey, Spielstunden,
                                                istInstalliert)
Spiel(SpielID, Name, Entwickler, Altersbeschraenkung, ↑GenreID)
Genre(GenreID, Bezeichnung)
```

Abbildung 2: Datenbankschema zur Teilmodellierung aus Abbildung 1

Name: _____

a) *Erläutern Sie am Beispiel der Modellierung in Abbildung 1 und des Datenbankschemas in Abbildung 2, wie aus einem Entity-Relationship-Diagramm ein Datenbankschema entwickelt wird, und gehen Sie dabei insbesondere auf die Umsetzung der Beziehungstypen ein.*

(8 Punkte)

b) *Geben Sie die Eigenschaften an, die ein Relationenschema in der dritten Normalform erfüllen muss.*

Überführen Sie das in Abbildung 2 gegebene Datenbankschema in die dritte Normalform und erläutern Sie die Änderungen, die zur Überführung in die dritte Normalform nötig sind.

Hinweis: Beispieldaten zum Datenbankschema in Abbildung 2 finden Sie in der Anlage.

(9 Punkte)

c) Die folgende SQL-Anweisung wird auf dem Datenbankschema in Abbildung 2 ausgeführt.

```
1    SELECT Genre.Bezeichnung, COUNT(besitzt.SpielID) AS C
2    FROM besitzt
3      INNER JOIN Spiel
4        ON besitzt.SpielID = Spiel.SpielID
5      INNER JOIN Genre
6        ON Spiel.GenreID = Genre.GenreID
7    WHERE besitzt.istInstalliert = TRUE
8    GROUP BY Genre.GenreID
9    ORDER BY C
```

Analysieren und erläutern Sie die obige SQL-Anweisung.

Erläutern Sie, welche Information die Anweisung im Sachzusammenhang ermittelt.

(11 Punkte)

d) Die folgenden Anfragen sollen ebenfalls auf dem Datenbankschema in Abbildung 2 realisiert werden.

(i) Gesucht sind die Namen aller Benutzer, die 16 Jahre oder jünger sind.

(ii) Gesucht sind alle Benutzer, die das Spiel „SimTown“ mehr als 300 Stunden lang gespielt haben. Es sollen ihre Namen, die jeweiligen Spielstunden für dieses Spiel und der RegKey ermittelt werden.

(iii) Gesucht sind die Namen aller Spiele, die dem Benutzer mit dem Namen „Baldur80“ noch in seiner Sammlung fehlen.

Entwerfen Sie für die obigen Anfragen jeweils eine SQL-Anweisung.

(12 Punkte)

Name: _____

e) Für eine erweiterte Version soll die Vertriebsplattform um Social-Media-Komponenten ergänzt werden.
In einer ersten Version sollen Benutzerinnen und Benutzer anderen Benutzerinnen und Benutzern Chat-Nachrichten schicken können.
Damit man signalisieren kann, ob man gerade bereit ist, die Chat-Funktion zu nutzen, soll aus mehreren vorgegebenen Alternativen ein Status auswählt werden können, der anderen angezeigt wird (z. B. online, abwesend, beschäftigt usw.).

Modellieren Sie die erweiterte Datenbank als Entity-Relationship-Diagramm, indem Sie den Entitätstypen `Benutzer` *aus Abbildung 1 übernehmen und die erforderlichen neuen Entitätstypen und Beziehungstypen mit Attributen und Kardinalitäten ergänzen.*

Erläutern Sie, wie die zusätzlichen Anforderungen in Ihrer Modellierung umgesetzt sind.

(10 Punkte)

Zugelassene Hilfsmittel:

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

Name: _____

Anlage

Beispieldaten zum Datenbankschema aus Abbildung 2

Benutzer			
Benutzername	**Passwort**	**Alterseinstufung**	**Alter**
Trooper77	§k(652J!	18	22
Baldur80	#)§6dh%“6	16	17
I_will_win123	<ß§&JG51§KK.1&512!	12	12
Averroes79	5%4dK@72/§2n%	18	39

Hinweise: Passwörter werden verschlüsselt in der Datenbank abgelegt. Das Attribut `Alterseinstufung` kann die Werte `0`, `6`, `12`, `16` und `18` annehmen. Jede Person in der Relation `Benutzer` wird entsprechend ihres Alters in die höchstmögliche Alterseinstufung eingeordnet.

Spiel				
SpielID	**Name**	**Entwickler**	**Altersbeschraenkung**	**GenreID**
1	SimTown	Hellhackers & Co	6	5
2	Older Scribbles 5: Skyedge	Mayland Softworks	16	3
3	Need for Quickness 6	Electronic Crafts	12	1

besitzt				
Benutzername	**SpielID**	**RegKey**	**Spielstunden**	**istInstalliert**
Averroes79	2	GDH5-H652-HQQP-BG27	312	TRUE
Baldur80	3	K55G-HBD1-QQ2K-JH12	5	FALSE
Trooper77	1	J7GB-BGG1-GDOQ-77R4	11	TRUE
Baldur80	2	LQUF-7GD5-HQ77-HS2E	122	TRUE

Genre	
GenreID	**Bezeichnung**
1	Sportspiel
2	Ego-Shooter
3	Rollenspiel
4	Lernspiel
5	Simulation

Unterlagen für die Lehrkraft

Abiturprüfung 2019

Informatik, Grundkurs

1. Aufgabenart

Analyse, Modellierung und Abfrage relationaler Datenbanken

2. Aufgabenstellung[1]

siehe Prüfungsaufgabe

3. Materialgrundlage

entfällt

4. Bezüge zum Kernlehrplan und zu den Vorgaben 2019

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

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

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

2. *Medien/Materialien*
   - entfällt

5. Zugelassene Hilfsmittel

- Taschenrechner (grafikfähiger 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)

Um aus einem Entity-Relationship-Diagramm ein Datenbankschema zu entwickeln, wird zu jedem Entitätstyp ein Relationenschema erstellt. In diesem Relationenschema werden dieselben Attribute eingetragen wie im entsprechenden Entitätstyp. Auch die Primärschlüssel werden übernommen. Im Fall der in Abbildung 1 gegebenen Modellierung wird so mit den Entitätstypen `Benutzer`, `Spiel` und `Genre` verfahren.

Um 1:n-Beziehungstypen umzusetzen, kann der Primärschlüssel an der 1-Seite in das Relationenschema an der n-Seite als Fremdschlüssel eingetragen werden. Im gegebenen Beispiel wird für die Umsetzung des Beziehungstyps `gehoertZu` also der Primärschlüssel des Relationenschemas `Genre` als Fremdschlüssel in das Relationenschema `Spiel` aufgenommen.

Jeder n:m-Beziehungstyp wird durch ein eigenes Relationenschema realisiert. Die Attribute des Beziehungstyps werden zusammen mit den Primärschlüsseln beider beteiligter Entitätstypen aufgenommen. Diese Fremdschlüssel werden in der Regel als kombinierter Primärschlüssel der Beziehungsrelation verwendet. Im Fall der gegebenen Modellierung wird so mit dem Beziehungstyp `besitzt` verfahren.

Teilaufgabe b)

Ein Relationenschema ist in der dritten Normalform, wenn alle Attribute einen atomaren Wertebereich haben, sich bei einem kombinierten Primärschlüssel alle anderen Attribute auf alle Teile des Primärschlüssels beziehen und keine transitiven Abhängigkeiten von Nicht-Schlüssel-Attributen zum Primärschlüssel existieren.

Folgendes Datenbankschema ist in der dritten Normalform:

Benutzer(Benutzername, Passwort, Alter)
besitzt(↑Benutzername, ↑SpielID, RegKey, Spielstunden, istInstalliert)
Spiel(SpielID, Name, Entwickler, Altersbeschraenkung, ↑GenreID)
Genre(GenreID, Bezeichnung)

Bereits alle Attribute des ursprünglichen Datenbankschemas in Abbildung 2 sind atomar. Auch beziehen sich alle Attribute des Relationenschemas `besitzt` auf beide Teile des Primärschlüssels. Im Relationenschema `Benutzer` gibt es jedoch eine funktionale Abhän-

gigkeit zwischen den Attributen `Alter` und `Alterseinstufung` und somit auch eine transitive Abhängigkeit zum Primärschlüssel. Das Problem kann gelöst werden, indem das Attribut `Alterseinstufung` gestrichen wird. Es handelt es sich um eine redundante Information, die sich auch aus dem Alter ableiten lässt.

Teilaufgabe c)

Die SQL-Anweisung führt von Zeile `3` bis Zeile `6` Verbundoperationen (`INNER JOIN`) zwischen den Relationen `besitzt`, `Spiel` und `Genre` durch.
Anschließend werden die Datensätze in Zeile `7` mit einer Selektion auf diejenigen einschränkt, die zu Spielkopien gehören, die gerade auf einem Rechner installiert sind.
Diese werden in Zeile `8` nach ihren Genres gruppiert.
Auf diese Gruppen wird in Zeile `1` die Aggregatfunktion `COUNT` angewendet. Darüber hinaus wird zu jeder Gruppe die Bezeichnung des entsprechenden Genres angegebenen.
Die so entstehenden Datensätze aus Genrebezeichnungen und Anzahlen von Spielkopien werden in Zeile `9` aufsteigend nach diesen Anzahlen, die den Alias `C` bekommen, sortiert.

Im Sachzusammenhang liefert die SQL-Anweisung zu jedem Spielegenre die Bezeichnung und die Anzahl der Spielkopien, die aus diesem Genre gerade installiert sind. Die Datensätze werden nach der Anzahl der Spielkopien aufsteigend sortiert zurückgeliefert. Ein Genre, zu dem keine einzige Spielkopie installiert ist, wird nicht berücksichtigt.

Teilaufgabe d)

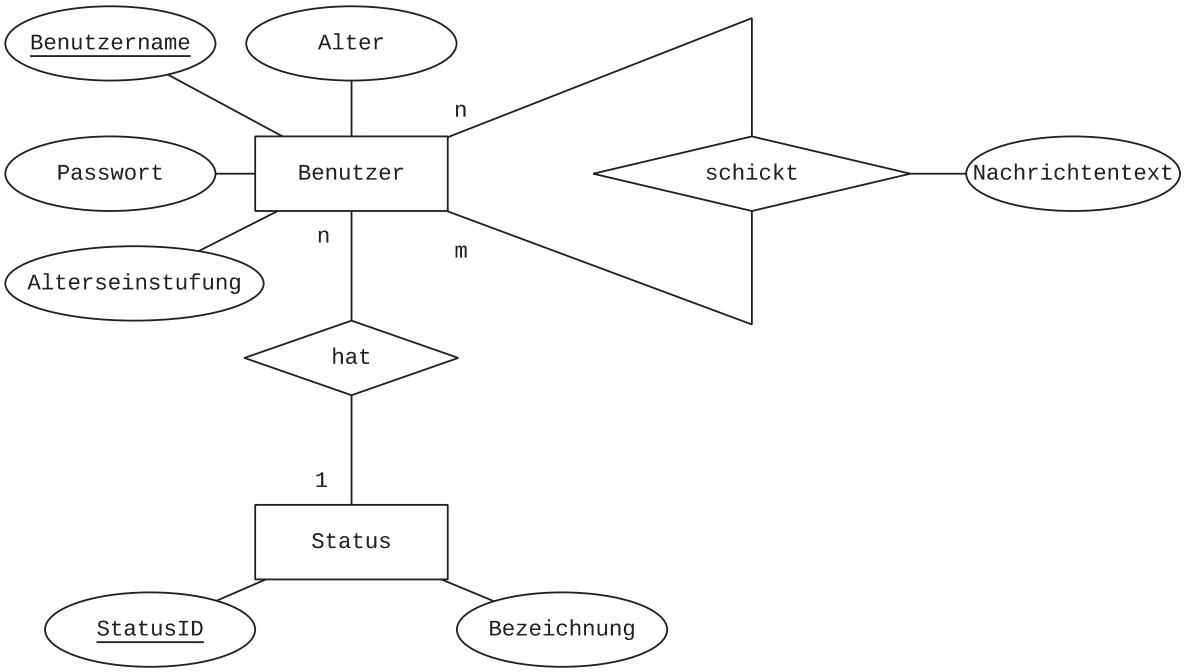
(i)
```
SELECT Benutzer.Benutzername
FROM Benutzer
WHERE Benutzer.Alter <= 16
```

(ii)
```
SELECT Benutzer.Benutzername, besitzt.RegKey, besitzt.Spielstunden
FROM Benutzer
  INNER JOIN besitzt
    ON Benutzer.Benutzername = besitzt.Benutzername
  INNER JOIN Spiel
    ON besitzt.SpielID = Spiel.SpielID
WHERE Spiel.Name = "SimTown" AND
      besitzt.Spielstunden > 300
```

(iii)
```
SELECT Spiel.Name
FROM Spiel
WHERE Spiel.SpielID NOT IN (
  SELECT besitzt.SpielID
  FROM besitzt
  WHERE besitzt.Benutzername = "Baldur80"
)
```

Teilaufgabe e)

Die folgende Datenbankmodellierung setzt die zusätzlichen Anforderungen um:



Um Nachrichten verschicken zu können, wird der n:m-Beziehungstyp `schickt` ergänzt. Datensätze der entsprechenden Beziehungsrelation stellen jeweils eine Nachricht zwischen zwei Benutzern dar. Daher erhält dieser Beziehungstyp auch noch den Text der Chat-Nachricht (`Nachrichtentext`) als Attribut.

Der Entitätstyp `Status` wird mit `StatusID` und einer `Bezeichnung` ergänzt. Zwischen `Benutzer` und `Status` wird mit dem Beziehungstyp `hat` ein 1:n-Beziehungstyp modelliert, da jeder Benutzer immer genau einen Status hat.

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 am Beispiel, wie aus einem Entity-Relationship-Diagramm ein Datenbankschema entwickelt wird, und geht dabei auf die Umsetzung der Beziehungstypen ein.	8			
Sachlich richtige Lösungsalternative zur Modelllösung: (8) …… ……					
	Summe Teilaufgabe a)	**8**			

Teilaufgabe b)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	gibt die Eigenschaften an, die ein Relationenschema in der dritten Normalform erfüllen muss.	3			
2	überführt das gegebene Datenbankschema in die dritte Normalform.	4			
3	erläutert die Änderungen, die zur Überführung in die dritte Normalform nötig sind.	2			
Sachlich richtige Lösungsalternative zur Modelllösung: (9) …… ……					
	Summe Teilaufgabe b)	**9**			

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

Teilaufgabe c)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	analysiert und erläutert die SQL-Anweisung.	7			
2	erläutert, welche Information die Anweisung im Sachzusammenhang ermittelt.	4			
Sachlich richtige Lösungsalternative zur Modelllösung: (11) …… ……					
	Summe Teilaufgabe c)	**11**			

Teilaufgabe d)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	entwirft eine SQL-Anweisung für die erste Anfrage.	3			
2	entwirft eine SQL-Anweisung für die zweite Anfrage.	4			
3	entwirft eine SQL-Anweisung für die dritte Anfrage.	5			
Sachlich richtige Lösungsalternative zur Modelllösung: (12) …… ……					
	Summe Teilaufgabe d)	**12**			

Teilaufgabe e)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	modelliert die erweiterte Datenbank als Entity-Relationship-Diagramm.	6			
2	erläutert, wie die zusätzlichen Anforderungen in der Modellierung umgesetzt sind.	4			
Sachlich richtige Lösungsalternative zur Modelllösung: (10) …… ……					
	Summe Teilaufgabe e)	**10**			

	Summe insgesamt	**50**			

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

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

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

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

Unterschrift, Datum:

Grundsätze für die Bewertung (Notenfindung)

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

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

Name: _____

Abiturprüfung 2019

Informatik, Grundkurs

Aufgabenstellung:

Mit sogenannten Strichcodes ist es möglich, binäre Codes optisch darzustellen. Für eine `1` wird ein schwarzer Streifen verwendet, für eine `0` ein weißer Streifen.
Beim Lesen des Strichcodes kann es zu Lesefehlern kommen. Es wird statt einer `0` eine `1` gelesen oder umgekehrt. Eine Methode zur Erkennung der Fehler besteht darin, dass die Bitfolge, die gelesen werden soll, durch ein letztes Prüfbit ergänzt wird.

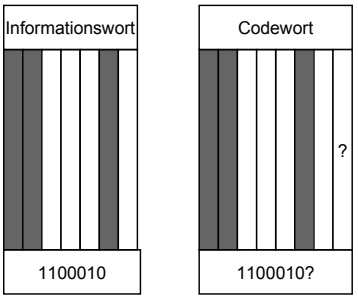


Abbildung 1:
Informationswort und Codewort

Zur Überprüfung, ob das Codewort korrekt gebildet ist, wird hier ein endlicher Automat A benutzt, der durch einen Übergangsgraphen dargestellt wird:

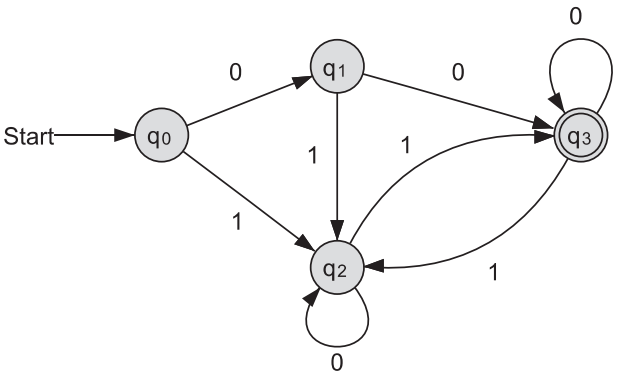


Abbildung 2: Zustandsübergangsdiagramm Automat A

a) Gegeben sind die beiden Codewörter `110111` und `01101101001` , sie bestehen aus einem Informationswort und einem angehängten Prüfbit.

Ermitteln Sie, welche der beiden Codewörter vom Automaten `A` *akzeptiert werden.*

Bestimmen Sie ein möglichst kurzes Codewort, das vom Automaten `A` *akzeptiert wird.*

Erläutern Sie, wieso der Automat `A` *in der Lage ist zu prüfen, ob die Anzahl der Einsen gerade oder ungerade ist.*

Erläutern Sie einen Lesefehler, der durch die Anwendung dieses Automaten erkannt wird, und einen Lesefehler, der durch die Anwendung dieses Automaten nicht erkannt wird.

(10 Punkte)

Name: _____

b) Es werden nun nicht-leere Informationsworte gerader Länge betrachtet. Um mehr Lesefehler zu erkennen, wird jeweils nach zwei Bit des Informationswortes ein Prüfbit ergänzt. Das Bit wird so gewählt, dass die drei Bit eine ungerade Anzahl von Einsen enthalten.

Bestimmen Sie das Codewort zum Informationswort `110110` *nach dem oben beschriebenen Verfahren.*

Entwerfen Sie einen endlichen Automaten A_1*, der nur die Codewörter akzeptiert, die nach dem oben beschriebenen Verfahren gebildet werden.*

(10 Punkte)

Im Folgenden wird ein Code zur Zifferndarstellung benutzt. Jede Ziffer wird durch eine Folge aus sieben Bit definiert. Im Strichcode wird die 1 durch einen schwarzen Streifen und die 0 durch weißen Streifen dargestellt. Benachbarte gleichfarbige Streifen werden zu Balken zusammengefasst, die unterschiedliche Breite haben können. Der Stichcode für eine Ziffer besteht aus zwei schwarzen und zwei weißen Balken. Die Kennzeichnung gibt die Balkenbreiten an.

Beispiel:

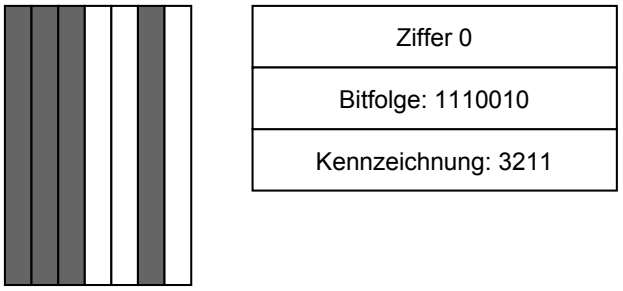


Abbildung 3: Strichcode für die Ziffer `0`

Von links nach rechts steht ein schwarzer Balken dreifacher Breite (3), ein weißer Balken doppelter Breite (2), dann ein schwarzer Balken einfacher Breite (1) und ein weißer Balken einfacher Breite (1). Damit hat der Strichcode die Kennzeichnung `3211`.

Die folgende Grammatik `G = (N,T,S,P)` erzeugt die Sprache, die die oben beschriebenen Kennzeichnungen des Codes enthält. Damit die Produktionen nicht zu umfangreich werden, werden durch die Grammatik `G` die Kennzeichnungen nur für einige Ziffern erzeugt.

Name: _____

Startsymbol: S

Nichtterminale: N = {S, A, B, C, D, E, X, Y, Z}

Terminalsymbole: T = {1, 2, 3}

Produktionen: P = {

S → A B | A E | C D | D C,
A → 1 Y,
B → 1 Z,
C → 2 X,
D → 2 Y,
E → 3 X,
X → 1,
Y → 2,
Z → 3
}

c)

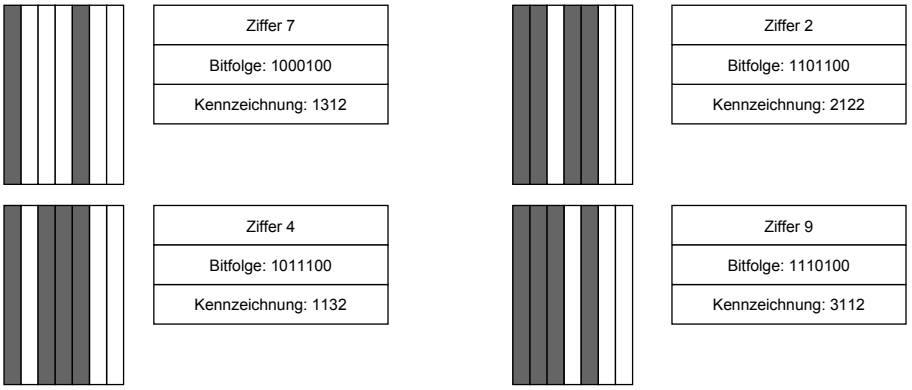


Abbildung 4: Strichcode für Ziffern 7, 2, 4 9

Zeigen Sie, dass die Kennzeichnung für die Ziffer 7 *nicht zur Sprache von* G *gehört und die Kennzeichnung der Ziffer* 2 *zur Sprache von* G *gehört.*

Entwickeln Sie eine Grammatik G_1*, dessen Sprache die Sprache von* G *umfasst, erweitert um die Kennzeichnungen für Ziffer* 7*, Ziffer* 9*, Ziffer* 0 *und Ziffer* 4*.*

(6 Punkte)

d) *Begründen Sie, dass die Grammatik* G *nicht regulär ist.*

Entwerfen Sie eine reguläre Grammatik G_2*, aus der sich genau die Wörter ableiten lassen, die sich aus der Grammatik* G *ableiten lassen.*

(10 Punkte)

Name: _____

e) Ein beliebig langes Informationswort besteht aus einer geraden Anzahl von Binärziffern.

Folgende Behauptung wird aufgestellt:
Wenn man das Informationswort vollständig oder blockweise in Zweierblöcken wiederholt, kann man endliche Automaten konstruieren, die alle Lesefehler beim Lesen erkennen.

Beispiele:

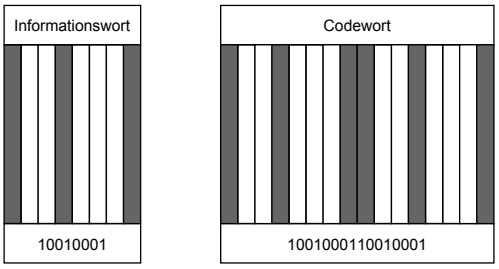


Abbildung 5: Vollständige Wiederholung

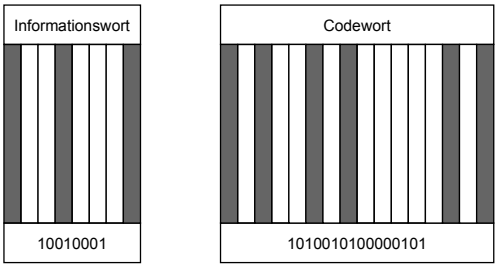


Abbildung 6: Blockweise Wiederholung

Entwerfen Sie einen endlichen Automaten A_2, der nach obigen Verfahren auf Fehlerfreiheit bei einer blockweisen Wiederholung in Zweierblöcken testet.

Beurteilen Sie die Behauptung.

(14 Punkte)

Zugelassene Hilfsmittel:

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

Unterlagen für die Lehrkraft

Abiturprüfung 2019

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 2019

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

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

2. *Medien/Materialien*
   - entfällt

5. Zugelassene Hilfsmittel

- Taschenrechner (grafikfähiger 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)

`110111`

$q_0 \xrightarrow{1} q_2 \xrightarrow{1} q_3 \xrightarrow{0} q_3 \xrightarrow{1} q_2 \xrightarrow{1} q_3 \xrightarrow{1} q_2$

q_2 ist kein Endzustand, das Codewort wird nicht akzeptiert.

`01101101001`

$q_0 \xrightarrow{0} q_1 \xrightarrow{1} q_2 \xrightarrow{1} q_3 \xrightarrow{0} q_3 \xrightarrow{1} q_2 \xrightarrow{1} q_3 \xrightarrow{0} q_3 \xrightarrow{1} q_2 \xrightarrow{0} q_2 \xrightarrow{0} q_2 \xrightarrow{1} q_3$

q_3 ist ein Endzustand, das Codewort wird akzeptiert.

Ein möglichst kurzes Codewort ist `11`.

$q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_3$

q_3 ist ein Endzustand, das Codewort wird akzeptiert.

Das Prüfbit wird so ergänzt, dass die Anzahl der Einsen im Codewort gerade ist. Zum Zustand q_2 kommt man zum ersten Mal durch folgende Bitfolgen: `1`, `01`, `001`.
Von q_2 kommt man mit einer 1 zum Endzustand q_3. In diesen Fällen ist also die Anzahl der Einsen gerade. Von q_3 kommt man durch `0` oder durch `11` wieder zu q_3, wobei zwischen den zwei Einsen beliebig viele Nullen stehen können. Die Anzahl der Einsen bleibt also gerade.

Da das Verändern eines einzelnen Bit zu einer falschen Parität führt, können solche Fehler beim Einlesen erkannt werden. Werden zwei Bit vertauscht, so ergibt sich wieder die korrekte Parität, der Fehler wird nicht erkannt.
Allgemein kann bei einer ungeraden Anzahl an Bitfehlern das fehlerhafte Lesen erkannt werden, bei einer geraden Anzahl an Bitfehlern bleibt der Fehler unbemerkt.

Teilaufgabe b)

Hinweis: Es werden Lücken gelassen, um die Lesbarkeit zu erhöhen.

Informationswort gerader Länge: `110110 bzw. 11 01 10`

Codewort mit Prüfbit jeweils nach 2-Bitblöcken: `111 010 100 bzw. 111010100`

Endlicher deterministischer Automat A_1:

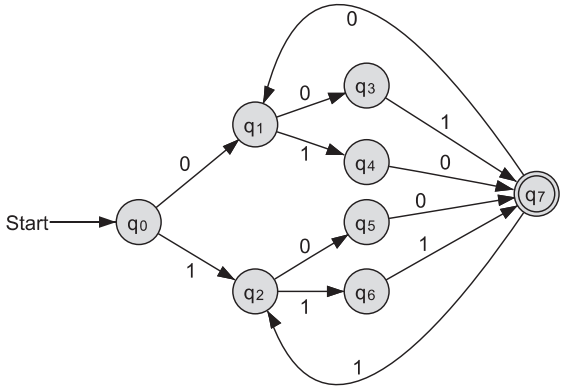
Zustandsmenge: $\{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7\}$

Eingabealphabet: $\{0, 1\}$

Startzustand: q_0

Menge der Endzustände: $\{q_7\}$

Zustandsübergangsdiagramm:



Teilaufgabe c)

Die Ziffer 7 hat die Kennzeichnung 1312.

S → AB → 1YB

1312 gehört nicht zur Sprache, da es bei der Ableitung keine Alternative gibt, die zu 13B führt.

Die Ziffer 2 hat die Kennzeichnung 2122.

S → CD → 2XD → 21D → 212Y → 2122

2122 gehört zur Sprache, wie die Ableitung zeigt.

Grammatik G_1:

Startsymbol: S

Nichtterminale: N = {S, A, B, C, D, E, F, G, X, Y, Z}

Terminalsymbole: T = {1, 2, 3}

Produktionen: P = {

S → A B | A E | C D | D C | BA | EA | FG | GF,
A → 1 Y,
B → 1 Z,
C → 2 X,
D → 2 Y,
E → 3 X,
F → 3 Y,
G → 1 X,
X → 1,
Y → 2,
Z → 3
}

Teilaufgabe d)

Eine reguläre Grammatik kann entweder rechtslinear oder linkslinear sein. Bei einer rechtslinearen Grammatik haben alle Produktionen auf der rechten Seite ein Terminalsymbol oder ein Terminalsymbol gefolgt von einem Nichtterminalsymbol. Bei einer linkslinearen Grammatik haben alle Produktionen auf der rechten Seite entweder ein Terminalsymbol oder ein Nichtterminalsymbol gefolgt von einem Terminalsymbol.

Die Produktion `S → A B | A E | C D | D C` verstößt gegen diese Regel, denn auf der rechten Seite stehen bei jeder Alternative zwei Nichtterminalsymbole.

Die folgende rechtslineare Grammatik G_2 erzeugt die gleiche Sprache:

Startsymbol: `S`

Nichtterminale: `N = {S, A, B, C, D, E, F, G, H}`

Terminalsymbole: `T = {1, 2, 3}`

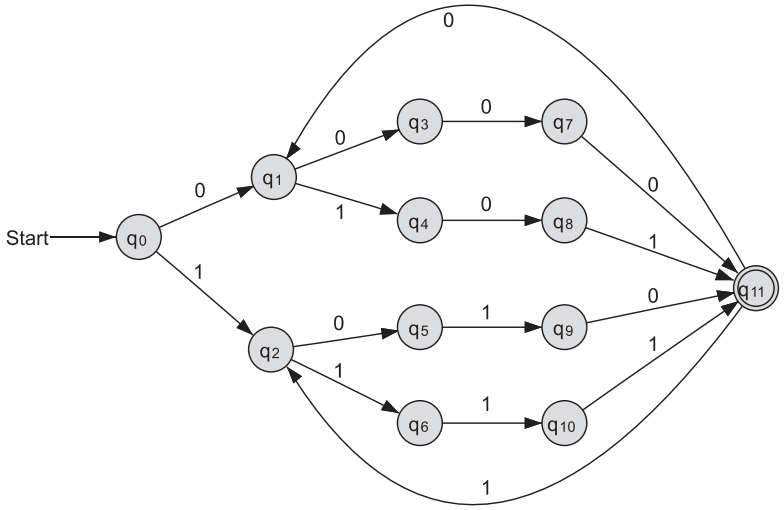
Produktionen:

```
P = {
    S → 1 A | 2 B
    A → 2 C
    B → 1 E | 2 D
    C → 1 F | 3 G
    D → 2 G
    E → 2 H
    F → 3
    G → 1
    H → 2
    }
```

Teilaufgabe e)

Endlicher deterministischer Automat, der die blockweise Wiederholung testet:
Zustandsmenge: $\{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9, q_{10}, q_{11}\}$
Eingabealphabet: $\{0, 1\}$
Startzustand: q_0
Menge der Endzustände: $\{q_{11}\}$

Zustandsübergangsdiagramm:



Bei der blockweisen Wiederholung können auch Lesefehler nicht erkannt werden. Wenn in dem Block an einer Position und in der Wiederholung an der gleichen Position das Bit falsch gelesen wird, wird der Fehler nicht erkannt.

Es wird keine Aussage über die Länge des Informationswortes gemacht. Bei einer vollständigen Wiederholung des Informationswortes im Codewort ist es allgemein nicht möglich, mit einem endlichen Automaten zu überprüfen, ob die Informationswörter identisch sind. Für jede Bitfolge müsste ein eigener Automat konstruiert 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	ermittelt, welches Codewort akzeptiert und welches nicht akzeptiert wird.	2			
2	bestimmt ein möglichst kurzes Codewort, das akzeptiert wird.	2			
3	erläutert, wieso der Automat prüft, ob die Anzahl der Einsen gerade oder ungerade ist.	4			
4	erläutert Lesefehler, die durch den Automaten erkannt und nicht erkannt werden.	2			
Sachlich richtige Lösungsalternative zur Modelllösung: (10) …… ……					
	Summe Teilaufgabe a)	**10**			

Teilaufgabe b)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	bestimmt ein Codewort zu einem gegebenen Informationswort nach dem angegebenen Verfahren.	2			
2	entwirft einen endlichen Automaten, der nur Codewörter akzeptiert, die nach dem beschriebenen Verfahren gebildet werden.	8			
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	zeigt, dass das eine angegebene Wort nicht zur Sprache gehört und dass das andere angegebene Wort zur Sprache gehört.	3			
2	entwickelt eine Grammatik G_1 mit den geforderten Eigenschaften.	3			
Sachlich richtige Lösungsalternative zur Modelllösung: (6) …… ……					
	Summe Teilaufgabe c)	**6**			

Teilaufgabe d)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	begründet, dass die Grammatik nicht regulär ist.	2			
2	entwirft eine reguläre Grammatik mit den angegebenen Eigenschaften.	8			
Sachlich richtige Lösungsalternative zur Modelllösung: (10) …… ……					
	Summe Teilaufgabe d)	**10**			

Teilaufgabe e)

	Anforderungen	Lösungsqualität			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	entwirft einen endlichen Automaten, der die Fehlerfreiheit nach angegebenen Verfahren bei einer blockweisen Wiederholung in Zweierblöcken testet.	8			
2	beurteilt und formuliert für den Fall der blockweisen Wiederholung, dass nicht alle Fehler erkannt werden können.	2			
3	beurteilt und formuliert, dass es für beliebig lange Informationswörter, die vollständig wiederholt werden, keinen endlichen Automaten gibt, der in jedem Fall überprüfen kann, ob das Lesen fehlerfrei war.	4			
Sachlich richtige Lösungsalternative zur Modelllösung: (14) …… ……					
	Summe Teilaufgabe e)	**14**			

	Summe insgesamt	**50**			

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

	Lösungsqualität			
	maximal erreichbare Punktzahl	**EK**	**ZK**	**DK**
Übertrag der Punktsumme aus der ersten bearbeiteten Aufgabe	**50**			
Übertrag der Punktsumme aus der zweiten bearbeiteten Aufgabe	**50**			
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