Name: _____

Abiturprüfung 2022

Informatik, Leistungskurs

Aufgabenstellung:

In der Turing-Schule werden häufig unterschiedliche Schulveranstaltungen (z. B. Schulmusical, Schultheater) aufgeführt. In der Vergangenheit bildeten sich vor dem Einlass für eine Veranstaltung teils lange Warteschlangen.
Um mehr Planungssicherheit für alle Personen zu gewährleisten, sollen für eine Veranstaltung die Sitzplätze nun im Vorfeld reserviert werden können. Die Schülerinnen und Schüler des Informatikprojektkurses möchten daher ein Buchungssystem für diese Veranstaltungen entwickeln.
Es hat sich bewährt, dass die Aula für jede Veranstaltung individuell bestuhlt wird. Dabei werden die Stühle immer in Sitzplatzreihen angeordnet. Bei einer Veranstaltung werden pro Sitzplatzreihe immer gleich viele Stühle aufgestellt.
Eine Person kann für eine Veranstaltung auch mehrere Sitzplätze reservieren. Falls eine Veranstaltung bereits ausgebucht ist, kann sich eine Person – für mehrere Platzwünsche auch mehrfach – auf die Warteliste setzen lassen. Für eine Person wird – nur zum Zweck der schnellen Erreichbarkeit bei Veranstaltungsabsagen – die Telefonnummer gespeichert.

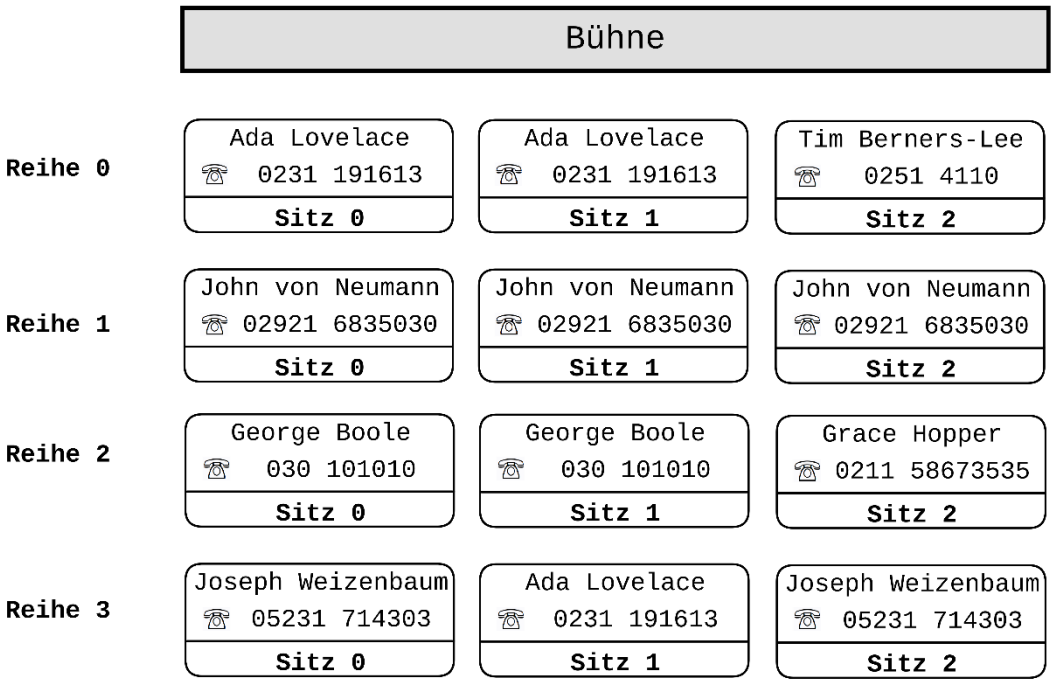


Abbildung 1: Darstellung der Reservierungen der Sitzplätze für eine Veranstaltung mit vier Sitzplatzreihen und jeweils drei Plätzen pro Reihe mit Beispieldaten

Name: _____

Eine erste Modellierung für die Verwaltung der Veranstaltungen ist im folgenden Implementationsdiagramm dargestellt. Die Dokumentationen der Klassen befinden sich im Anhang.

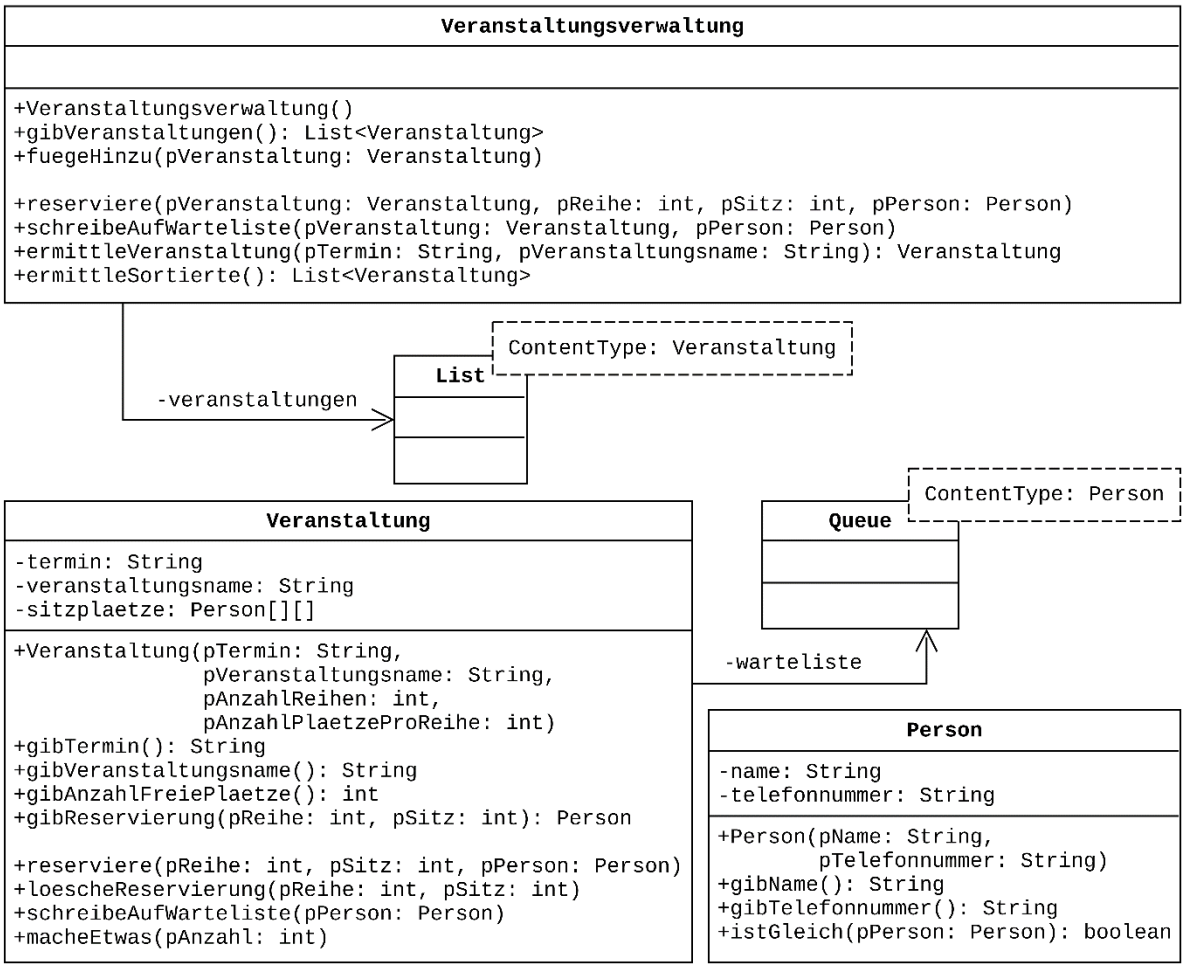


Abbildung 2: Teilmodellierung als Implementationsdiagramm

a) *Beschreiben Sie die Beziehungen zwischen den Klassen der in Abbildung 2 gegebenen Teilmodellierung.*

Begründen Sie im Sachkontext, warum es sinnvoll ist, die Datensammlungen für die Warteliste in Form einer Schlange (`Queue`*) zu realisieren.*

Begründen Sie im Sachkontext, warum es sinnvoll ist, die Sitzplätze mit den zugehörigen Reservierungen der Personen als zweidimensionales Feld (Array) und nicht als dynamische Datenstruktur zu realisieren.

(3 + 2 + 2 Punkte)

Name: _____

b) Eine Schülerin entdeckt in der Klasse `Veranstaltung` die undokumentierte Methode `macheEtwas`.

```
 1 public void macheEtwas(int pAnzahl) {
 2   if (pAnzahl > 0) {
 3     int reihen = sitzplaetze.length;
 4     int sitze = sitzplaetze[0].length;
 5     int rNeu = reihen + pAnzahl;
 6     Person[][] sitzplaetzeNeu = new Person[rNeu][sitze];
 7     for (int r = 0; r < reihen; r++) {
 8       for (int s = 0; s < sitze; s++) {
 9         sitzplaetzeNeu[r][s] = sitzplaetze[r][s];
10       }
11     }
12     for (int r = reihen; r < rNeu; r++) {
13       for (int s = 0; s < sitze; s++) {
14         if (sitzplaetzeNeu[r][s] == null) {
15           sitzplaetzeNeu[r][s] = warteliste.front();
16           warteliste.dequeue();
17         }
18       }
19     }
20     sitzplaetze = sitzplaetzeNeu;
21   }
22 }
```

Angenommen, für die in Abbildung 1 dargestellte Veranstaltung wird die Methode `macheEtwas` mit dem Parameter `pAnzahl = 2` aufgerufen.

Gehen Sie davon aus, dass das Objekt `warteliste` wie folgt belegt ist:

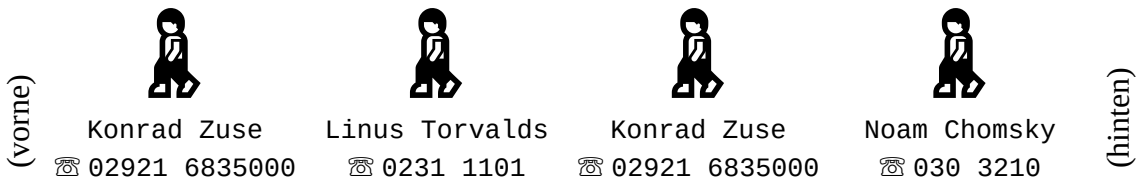


Abbildung 3: Personenobjekte, die von der Warteliste (Objekt `warteliste`) verwaltet werden

Analysieren Sie die Methode für diese Belegungen bei Anwendung auf die Beispieldaten aus Abbildung 1 und erläutern Sie die Funktionsweise der Methode sowie die Veränderungen der Belegung der Sitzplätze und der Warteliste.

Erläutern Sie die Aufgabe der Methode im Sachzusammenhang.

Beurteilen Sie im Sachkontext die Bedeutung der Verzweigung in Zeile 2.

Beurteilen Sie die Notwendigkeit der Verzweigung in Zeile 14.

(6 + 3 + 2 + 2 Punkte)

Name: _____

c) Die Klasse `Veranstaltungsverwaltung` soll die Methode mit dem folgenden Methodenkopf erhalten:

```
public List<Veranstaltung> ermittleSortierte()
```

Die Methode soll eine neue Liste mit allen Veranstaltungsobjekten zurückliefern. Diese neue Liste soll nach der Anzahl der noch freien Plätze absteigend sortiert sein. Bei gleicher Anzahl der noch freien Plätze spielt die Reihenfolge dieser Veranstaltungen untereinander keine Rolle.

Entwickeln und erläutern Sie ein algorithmisches Verfahren für die Methode `ermittleSortierte`.

Implementieren Sie die Methode `ermittleSortierte`.

(5 + 7 Punkte)

d) Die Schülerinnen und Schüler möchten in Rücksprache mit der Schulleitung ihr Projekt erweitern. Folgende Anforderungen sollen zusätzlich umgesetzt werden:

i. Die Schule möchte in Zukunft auch für jede Veranstaltung einen einheitlichen Eintrittspreis pro Sitzplatz erheben können. Dieser Eintrittspreis soll gespeichert werden. Der Eintrittspreis sowie die Summe der Eintrittsgelder auf Basis der Reservierungen sollen auch abgerufen werden können.
ii. Für jede Veranstaltung soll die Frage beantwortet werden können, ob für eine bestimmte Sitzplatzreihe eine bestimmte Anzahl an Sitzplätzen nebeneinander noch frei ist. Falls ja, dann soll die Nummer des ersten freien Sitzes der nebeneinander liegenden Sitzplätze zurückgeliefert werden. Falls für diese Veranstaltung die nebeneinander liegenden Plätze in der entsprechenden Anzahl in der gewünschten Reihe nicht mehr frei sind, muss auch diese Information in einer geeigneten Form zurückgegeben werden.
iii. Jede Person soll jeweils ihre gebuchten Veranstaltungen zurückliefern können.

Modellieren Sie die oben genannten Anforderungen als Erweiterung des Implementationsdiagramms aus Abbildung 2.

Erläutern Sie, wie Sie die zweite und dritte Anforderung in Ihrem Implementationsdiagramm realisieren.

Hinweis: Unveränderte Klassen, Klassenbeziehungen, Methoden und Attribute aus dem Implementationsdiagramm in Abbildung 2 müssen nicht angegeben bzw. dargestellt werden.

(9 + 4 Punkte)

Name: _____

e) Bei kurzfristigen Veranstaltungsabsagen konnte man leider nicht immer alle Personen telefonisch rechtzeitig erreichen. Daher soll nur zum Zweck der schnellen Erreichbarkeit bei Veranstaltungsabsagen zusätzlich zwingend die E-Mail-Adresse einer Person verwaltet werden.
Außerdem legt der Schulleiter der Projektgruppe eine E-Mail mit der Bitte um Prüfung vor. In der E-Mail wird ausschließlich auf zukünftige Schulveranstaltungen hingewiesen. Der Schulleiter argumentiert, dass er in Zukunft gerne monatlich so eine E-Mail als Serviceleistung an alle im Buchungssystem gespeicherten Personen versenden möchte, da sich diese ja ohnehin für die Schulveranstaltungen interessierten.

Beurteilen Sie den Vorschlag des Schulleiters aus der Perspektive des Datenschutzes.
(5 Punkte)

Zugelassene Hilfsmittel:

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

Name: _____

Anhang

Dokumentationen der verwendeten Klassen

Die Klasse Veranstaltungsverwaltung

Objekte der Klasse **`Veranstaltungsverwaltung`** verwalten Veranstaltungen.

Auszug aus der Dokumentation der Klasse `Veranstaltungsverwaltung`

`Veranstaltungsverwaltung()`

Ein Objekt der Klasse `Veranstaltungsverwaltung` wird initialisiert.

`List<Veranstaltung> gibVeranstaltungen()`

Eine Liste mit allen Veranstaltungen wird zurückgeliefert. Wenn keine Veranstaltung verwaltet wird, dann wird eine leere Liste zurückgegeben.

`void fuegeHinzu(Veranstaltung pVeranstaltung)`

Die im Parameter übergebene Veranstaltung wird hinzugefügt.

`void reserviere(Veranstaltung pVeranstaltung, int pReihe, int pSitz, Person pPerson)`

Für die Veranstaltung `pVeranstaltung` wird für den Platz (`pReihe`, `pSitz`) für die Person eine Reservierung vorgenommen. Falls für die entsprechende Veranstaltung an dem Platz bereits eine Reservierung vorliegt, dann wird diese überschrieben. Falls der Platz nicht existiert, `pVeranstaltung` `null` ist oder `pPerson` `null` ist, dann geschieht nichts.

`void schreibeAufWarteliste(Veranstaltung pVeranstaltung, Person pPerson)`

Das Personenobjekt `pPerson` wird an die Warteliste für die Veranstaltung `pVeranstaltung` angehängt. Falls `pVeranstaltung` `null` ist oder `pPerson` `null` ist, dann geschieht nichts.

`Veranstaltung ermittleVeranstaltung(String pTermin, String pVeranstaltungsname)`

Für den im Parameter übergebenen Termin und Veranstaltungsnamen wird das entsprechende Veranstaltungsobjekt zurückgeliefert. Falls es zu dem Termin keine Veranstaltung mit dem Namen gibt, wird `null` zurückgeliefert.

`List<Veranstaltung> ermittleSortierte()`

Die Methode soll eine neue Liste mit allen Veranstaltungsobjekten zurückliefern. Diese neue Liste soll nach der Anzahl der noch freien Plätze absteigend sortiert sein. Bei gleicher Anzahl der noch freien Plätze spielt die Reihenfolge dieser Veranstaltungen untereinander keine Rolle.

Name: _____

Die Klasse Veranstaltung

Objekte der Klasse **`Veranstaltung`** verwalten Termine, Veranstaltungsnamen und die Sitzplätze mit den entsprechenden Reservierungen der Personen.

Auszug aus der Dokumentation der Klasse `Veranstaltung`

```
Veranstaltung(String pTermin, String pVeranstaltungsname,
              int pAnzahlReihen,
              int pAnzahlPlaetzeProReihe)
```

Ein Objekt der Klasse `Veranstaltung` wird initialisiert mit dem Termin, dem Veranstaltungsnamen, der Anzahl der Sitzplatzreihen und der Anzahl der Plätze pro Reihe.

```
String gibTermin()
```

Der Termin der Veranstaltung wird zurückgeliefert.

```
String gibVeranstaltungsname()
```

Der Veranstaltungsname der Veranstaltung wird zurückgeliefert.

```
int gibAnzahlFreiePlaetze()
```

Die Anzahl der noch freien Plätze der Veranstaltung wird zurückgeliefert.

```
Person gibReservierung(int pReihe, int pSitz)
```

Das Personenobjekt mit einer Reservierung für den Platz (`pReihe`, `pSitz`) wird zurückgeliefert. Falls der Platz nicht reserviert ist, wird `null` zurückgeliefert.

```
void reserviere(int pReihe, int pSitz, Person pPerson)
```

Für die Veranstaltung wird für den Platz (`pReihe`, `pSitz`) für die Person eine Reservierung vorgenommen. Falls an dem Platz bereits eine Reservierung vorliegt, dann wird diese überschrieben. Falls der Platz nicht existiert oder `pPerson` `null` ist, dann geschieht nichts.

```
void loescheReservierung(int pReihe, int pSitz)
```

Für die Veranstaltung wird die Reservierung für den Platz (`pReihe`, `pSitz`) gelöscht. Falls der Platz nicht existiert, dann geschieht nichts.

```
void schreibeAufWarteliste(Person pPerson)
```

Das Personenobjekt `pPerson` wird an die Warteliste für die Veranstaltung angehängt. Falls `pPerson` `null` ist, dann geschieht nichts.

```
void macheEtwas(int pAnzahl)
```

Diese Methode ist Bestandteil der Teilaufgabe b).

Name: _____

Die Klasse `Person`

Objekte der Klasse **`Person`** verwalten einen Namen und eine Telefonnummer einer Person.

Auszug aus der Dokumentation der Klasse `Person`

`Person(String pName, String pTelefonnummer)`

Ein Objekt der Klasse `Person` wird initialisiert mit dem Namen und der Telefonnummer.

`String gibName()`

Der Name der Person wird zurückgeliefert.

`String gibTelefonnummer()`

Die Telefonnummer der Person wird zurückgeliefert.

`boolean istGleich(Person pPerson)`

Wenn das Personenobjekt inhaltlich mit dem im Parameter übergebenen Personenobjekt `pPerson` gleich ist, dann wird `true` zurückgeliefert und sonst `false`.
Für die Inhaltsgleichheit müssen die jeweiligen Namen und Telefonnummern übereinstimmen.

Name: _____

Die generische Klasse `Queue`

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

Dokumentation der Klasse `Queue<ContentType>`

`Queue()`

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

`boolean isEmpty()`

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

`void enqueue(ContentType pContent)`

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

`void dequeue()`

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

`ContentType front()`

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

Name: _____

Die generische Klasse `List`

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 kann 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>`

`List()`

Eine leere Liste wird erzeugt.

`boolean isEmpty()`

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

`boolean hasAccess()`

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

`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`.

`void toFirst()`

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

`void toLast()`

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

`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.

Name: _____

`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.

`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.

`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.

`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.

`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 2022

Informatik, Leistungskurs

1. Aufgabenart

Modellierung, Implementation und Analyse kontextbezogener 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 2022

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
       Array bis zweidimensional
       Schlange (Klasse `Queue`*)*
       Lineare Liste (Klasse `List`*)*

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

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

2. *Medien/Materialien*
   - entfällt

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

5. Zugelassene Hilfsmittel

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

6. Modelllösungen

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

Teilaufgabe a)

Ein Objekt der Klasse `Veranstaltungsverwaltung` verwaltet im Attribut `veranstaltungen` seine Veranstaltungen in einer linearen Liste mit Objekten der Klasse `Veranstaltung`. Jedes Objekt der Klasse `Veranstaltung` verwaltet seine Sitzplätze in einem zweidimensionalen Array mit dem Bezeichner `sitzplaetze`, welches Objekte vom Typ `Person` verwaltet. Zusätzlich wird unter dem Bezeichner `warteliste` eine Schlange (`Queue`) mit Objekten vom Typ `Person` gespeichert.

Da vorab nicht bekannt ist, wie viele Personenobjekte in der Warteliste für eine Veranstaltung verwaltet werden müssen, ist es sinnvoll, eine dynamische Datenstruktur zu verwenden und nicht ein statistisches Feld (Array). In diesem Anwendungsfall ist eine Schlange sinnvoll, da diese Datenstruktur die Verwaltung der Reihenfolge der Objekte durch das FIFO-Prinzip sicherstellt, ein faires Nachrücken nach Sitzplatzfreigabe garantiert und ein wahlfreier Zugriff auf jedes Objekt in der Warteliste nicht notwendig ist.

Für eine Veranstaltung wird zunächst eine Bestuhlungsvorgabe (Sitzplatzreihen mit Sitzen) festgelegt, die sich grundsätzlich nicht verändern soll. Wenn alle Plätze reserviert sind, dann können sich Personen auf eine Warteliste setzen lassen. Der Zugriff auf ein Personenobjekt im zweidimensionalen Array über die Indizes lässt sich im Gegensatz zu einer dynamischen Datenstruktur effizienter umsetzen.

Teilaufgabe b)

Nur falls der Wert des Parameters `pAnzahl` größer `0` ist, geschieht etwas.

In Zeile 6 wird ein neues zweidimensionales Array für Personenobjekte deklariert und initialisiert, welches um `pAnzahl` mehr Sitzplatzreihen verfügt.

Die ersten zwei geschachtelten Zählschleifen (vgl. Zeilen 7 – 11) stellen sicher, dass alle vorhandenen Reservierungen in das neue Array übertragen werden.

Die zweiten zwei geschachtelten Zählschleifen (vgl. Zeilen 12 – 19) stellen sicher, dass – nur für die neuen Sitzplatzreihen – die Sitzplätze zunächst reihenweise von vorne nach hinten und in jeder Reihe von links nach rechts durchlaufen werden. Wenn ein Sitzplatz frei ist

(vgl. Zeile 14), dann wird das erste Personenobjekt von der Warteliste als neue Reservierung (vgl. Zeile 15) eingetragen. Falls Die Warteliste leer ist, wird an der Stelle `null` eingetragen (vgl. Zeile 15). Das erste Personenobjekt wird anschließend aus der Warteliste gelöscht (vgl. Zeile 16).

In Zeile 20 wird die Referenz `sitzplaetze` auf das neue Array-Objekt gesetzt.

Die Bestuhlung der Veranstaltung wird in diesem Beispiel um zwei Sitzplatzreihen mit je drei Sitzplätzen nach hinten erweitert. Die vier Personen von der Warteliste füllen dann die neuen Sitzplätze auf. Die Warteliste ist anschließend leer. Zwei Sitzplätze sind noch frei.

Der Parameter `pAnzahl` bestimmt, um wie viele Reihen die Bestuhlung erhöht wird, falls `pAnzahl` größer `0` ist. Die Personen von der Warteliste rücken automatisch auf die freien Plätze nach.

Die Verzweigung in Zeile 2 verhindert, dass durch einen negativen Wert des Parameters `pAnzahl` Sitzplatzreihen entfernt werden. So wird sichergestellt, dass die Anzahl der Reihen nur erhöht werden kann.

Die Verzweigung in Zeile 14 ist nicht notwendig, da die Schleife in Zeile 12 nur über die neu hinzugefügten Sitzplatzreihen läuft. Die neu hinzugefügten Sitzplatzreihen beinhalten zunächst leere Sitzplätze, die dann mit Personen von der Warteliste aufgefüllt werden.

Teilaufgabe c)

Erzeuge eine lokale, anfangs leere Liste `sortierte`, die Objekte vom Typ `Veranstaltung` verwaltet.

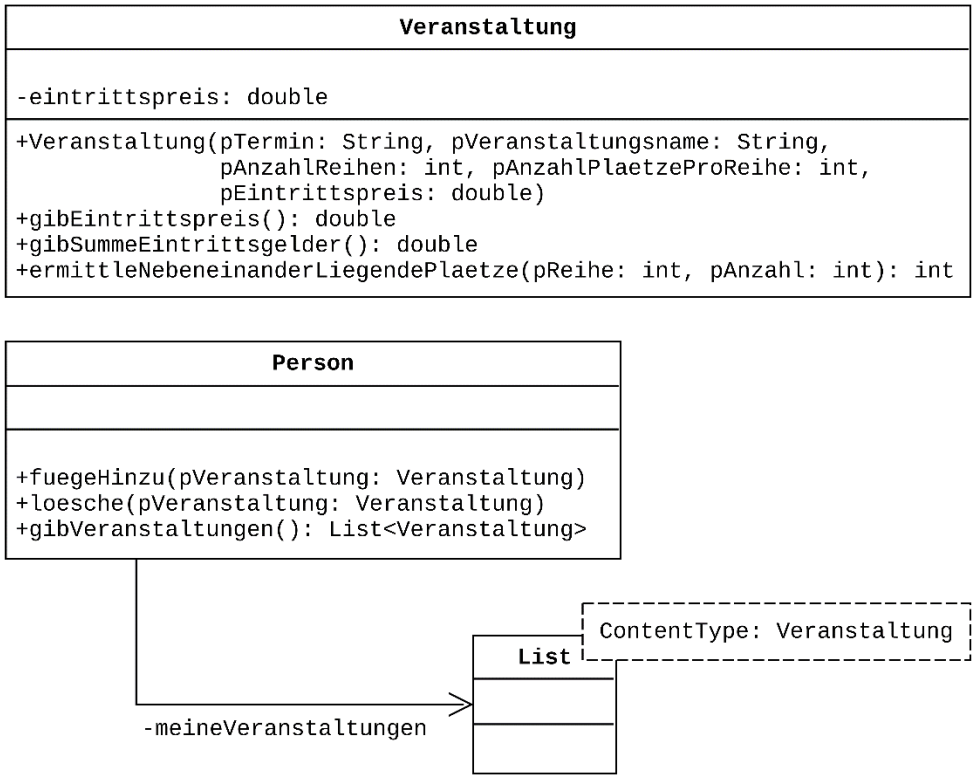
Durchlaufe die Liste `veranstaltungen` mit allen Objekten vom Typ `Veranstaltung` und füge schrittweise jedes Objekt sortiert in die Liste `sortierte` ein. Suche dafür in der Liste `sortierte` – beginnend am Anfang – die richtige Einfügestelle, d. h. man geht in der Liste der sortierten Veranstaltungsobjekte so lange weiter, wie die Anzahl der freien Plätze von der neu einzufügenden Veranstaltung maximal so groß ist wie die Anzahl der freien Plätze des aktuellen Veranstaltungsobjekts der sortierten Liste.

Diese lokale Liste mit den sortierten Veranstaltungen wird zurückgeliefert.

Eine mögliche Implementation der Methode:

```
public List<Veranstaltung> ermittleSortierte() {
  List<Veranstaltung> sortierte = new List<Veranstaltung>();
  veranstaltungen.toFirst();
  while (veranstaltungen.hasAccess()) {
    Veranstaltung veranstaltung = veranstaltungen.getContent();
    sortierte.toFirst();
    while (sortierte.hasAccess()
           && veranstaltung.gibAnzahlFreiePlaetze() <=
              sortierte.getContent().gibAnzahlFreiePlaetze()) {
      sortierte.next();
    }
    if (sortierte.hasAccess()) {
      sortierte.insert(veranstaltung);
    } else {
      sortierte.append(veranstaltung);
    }
    veranstaltungen.next();
  }
  return sortierte;
}
```

Teilaufgabe d)



Realisierung der zweiten Anforderung:
Die Methode `ermittleNebeneinanderLiegendePlaetze` in der Klasse `Veranstaltung` bekommt die Nummer einer Sitzplatzreihe und die gewünschte Anzahl der nebeneinanderliegenden Plätze übergeben.
Die Methode liefert einen Integerwert zurück, der die Nummer des ersten freien Sitzes der nebeneinanderliegenden Plätze in der gewünschten Reihe angibt. Falls keine nebeneinanderliegenden Plätze in der gewünschten Anzahl in der Reihe vorhanden sind, wird `-1` zurückgegeben.

Realisierung der dritten Anforderung:
Ein Objekt der Klasse `Person` verwaltet eine Liste mit Veranstaltungsobjekten. Über die Methode `fuegeHinzu` kann eine Veranstaltung der genannten Liste hinzugefügt und über die Methode `loesche` aus ihr gelöscht werden. Die Methode `gibVeranstaltungen` der Klasse `Person` liefert alle Veranstaltungen für dieses Personenobjekt als Liste zurück.

Teilaufgabe e)

Der Vorschlag der Schulleitung verstößt gegen das Grundprinzip des Verbots mit Erlaubnisvorbehalt.

Die Verwaltung der zusätzlichen personenbezogenen Daten (E-Mail-Adressen der Personen) wird mit dem Zweck legitimiert, eine schnelle Erreichbarkeit bei Absagen zu gewährleisten. Wenn die personenbezogenen Daten dann für Werbezwecke für weitere Veranstaltungen verwendet würden, entspräche dies nicht mehr dem konkreten Zweck, dem zugestimmt wurde.

Die Verarbeitung, d. h. zum Beispiel die Erhebung, Speicherung, Weitergabe oder allgemeine Verwendung personenbezogener Daten ist grundsätzlich verboten – es sei denn, die betroffene Person hat der Verarbeitung für einen konkreten Zweck zugestimmt oder es gibt eine explizite gesetzliche Regelung, die eine Verarbeitung für einen konkreten Zweck erlaubt.

7. Teilleistungen – Kriterien / Bewertungsbogen zur Prüfungsarbeit

Name des Prüflings:_____ Kursbezeichnung:_____

Schule: _____

Teilaufgabe a)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK[2]**	**ZK**	**DK**
1	beschreibt die Beziehungen zwischen den Klassen der in Abbildung 2 gegebenen Teilmodellierung.	3			
2	begründet im Sachkontext, warum es sinnvoll ist, die Datensammlungen für die Warteliste in Form einer Schlange (`Queue`) zu realisieren.	2			
3	begründet im Sachkontext, warum es sinnvoll ist, die Sitzplätze mit den zugehörigen Reservierungen der Personen als zweidimensionales Feld (Array) und nicht als dynamische Datenstruktur zu realisieren.	2			
Sachlich richtige Lösungsalternative zur Modelllösung: (7) …… ……					
	Summe Teilaufgabe a)	**7**			

Teilaufgabe b)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	analysiert die Methode für diese Belegungen bei Anwendung auf die Beispieldaten aus Abbildung 1 und erläutert die Funktionsweise der Methode sowie die Veränderungen der Belegung der Sitzplätze und der Warteliste.	6			
2	erläutert die Aufgabe der Methode im Sachzusammenhang.	3			
3	beurteilt im Sachkontext die Bedeutung der Verzweigung in Zeile 2.	2			
4	beurteilt die Notwendigkeit der Verzweigung in Zeile 14.	2			
Sachlich richtige Lösungsalternative zur Modelllösung: (13) …… ……					
	Summe Teilaufgabe b)	**13**			

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

Teilaufgabe c)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	entwickelt und erläutert ein algorithmisches Verfahren für die Methode.	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 erste Anforderung (i) als Erweiterung des Implementationsdiagramms aus Abbildung 2.	3			
2	modelliert die zweite Anforderung (ii) als Erweiterung des Implementationsdiagramms aus Abbildung 2.	2			
3	modelliert die dritte Anforderung (iii) als Erweiterung des Implementationsdiagramms aus Abbildung 2.	4			
4	erläutert, wie die zweite und dritte Anforderung in seinem Implementationsdiagramm realisiert werden.	4			
Sachlich richtige Lösungsalternative zur Modelllösung: (13) …… ……					
	Summe Teilaufgabe d)	**13**			

Teilaufgabe e)

	Anforderungen	Lösungsqualität			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	beurteilt den Vorschlag des Schulleiters aus der Perspektive des Datenschutzes.	5			
Sachlich richtige Lösungsalternative zur Modelllösung: (5) …… ……					
	Summe Teilaufgabe e)	**5**			

	Summe insgesamt	**50**			

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

	Lösungsqualität			
	maximal erreichbare Punktzahl	**EK**	**ZK**	**DK**
Übertrag der Punktsumme aus der ersten bearbeiteten Aufgabe	**50**			
Übertrag der Punktsumme aus der zweiten bearbeiteten Aufgabe	**50**			
Übertrag der Punktsumme aus der dritten bearbeiteten Aufgabe	**50**			
Punktzahl der gesamten Prüfungsleistung	**150**			
aus der Punktsumme resultierende Note gemäß nachfolgender Tabelle				
Note ggf. unter Absenkung um bis zu zwei Notenpunkte gemäß § 13 Abs. 2 APO-GOSt				
Paraphe				

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

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

Unterschrift, Datum:

Grundsätze für die Bewertung (Notenfindung)

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

Note	**Punkte**	**Erreichte Punktzahl**
sehr gut plus	15	150 – 143
sehr gut	14	142 – 135
sehr gut minus	13	134 – 128
gut plus	12	127 – 120
gut	11	119 – 113
gut minus	10	112 – 105
befriedigend plus	9	104 – 98
befriedigend	8	97 – 90
befriedigend minus	7	89 – 83
ausreichend plus	6	82 – 75
ausreichend	5	74 – 68
ausreichend minus	4	67 – 60
mangelhaft plus	3	59 – 50
mangelhaft	2	49 – 41
mangelhaft minus	1	40 – 30
ungenügend	0	29 – 0

Name: _____

Abiturprüfung 2022

Informatik, Leistungskurs

Aufgabenstellung:

Der Zoo „Tweety's Home" soll renoviert werden. Dabei muss für Notfälle auch ein Konzept zur Evakuierung der Besucherinnen und Besucher erstellt werden. Der Zoo besteht aus verschiedenen Attraktionen, die durch Wege miteinander verbunden sind. Für jede Attraktion ist eine maximale Anzahl an Gästen festgelegt, die sich dort gleichzeitig aufhalten dürfen. Da die Wege zwischen den Attraktionen unterschiedlich breit sind, können sich darauf pro Minute unterschiedlich viele Gäste von einer Attraktion zur anderen bewegen. Für den Evakuierungsfall ist für jede Attraktion immer dasjenige Wegstück als bevorzugter Notausgang (🚶➔) gekennzeichnet, über das sich die größtmögliche Anzahl an Gästen pro Minute bewegen kann; nur der Eingang und die Ausgänge verfügen nicht über diese Markierung.

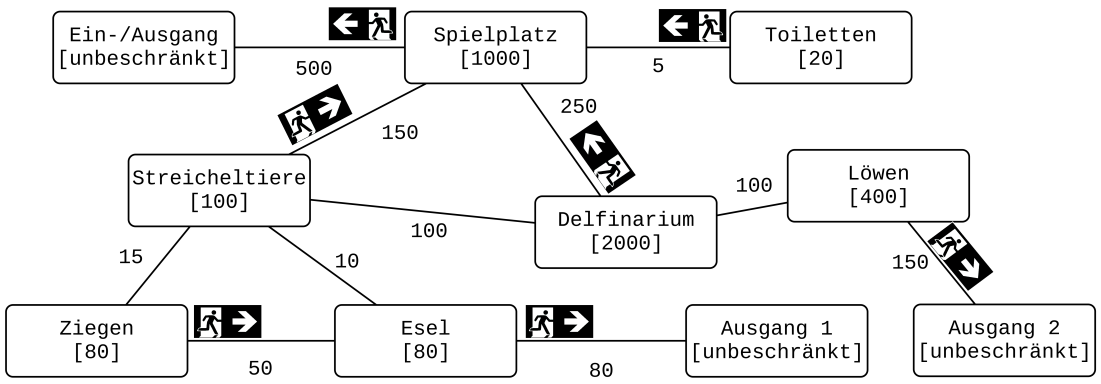


Abbildung 1: Ausschnitt aus dem Lageplan mit gekennzeichneten Notausgängen

Abbildung 1 zeigt beispielsweise:

- `Ein-/Ausgang` und `Spielplatz` sind über einen großzügigen Weg verbunden, über den jeweils bis zu `500` Personen pro Minute in jede der beiden möglichen Richtungen zwischen diesen Attraktionen wechseln können.
- Für das `Delfinarium` sind maximal `2000` Gäste zur gleichen Zeit vorgesehen.
- Im Evakuierungsfall sollen sich die Gäste des `Ziegen`-Geheges auf den Weg zum `Esel`-Gehege begeben.

Name: _____

a) Der Zoo soll um eine Attraktion `Pinguine` erweitert werden. Diese Attraktion soll alle folgenden Eigenschaften erfüllen:

1. Die Attraktion kann maximal `100` Gäste aufnehmen.
2. Die neue Attraktion ist durch direkte Wege jeweils mit den `Ziegen`- und `Esel`-Attraktionen verbunden.
3. Von der `Ziegen`-Attraktion können `30` Gäste pro Minute zur neuen Attraktion gelangen.
4. Sollte die neue Attraktion geräumt werden müssen, sollen die Gäste als Fluchtweg zunächst zur `Esel`-Attraktion gehen.

Geben Sie zunächst unabhängig von der geplanten Erweiterung an, wie viele Besucherinnen und Besucher sich maximal pro Minute von den Nachbarattraktionen insgesamt zum `Delfinarium` *begeben können.*

Erweitern Sie Abbildung 1 um die Attraktion `Pinguine` *und bestimmen Sie dabei eine geeignete Kapazität des Weges von den Pinguinen zur* `Esel`*-Attraktion.*

Hinweis: Für die Erweiterung sind lediglich die Änderungen im Vergleich zu Abbildung 1 darzustellen.

(2 + 4 Punkte)

Eine Softwarefirma hat bereits für andere (Freizeit-)Parks eine Software zur Verwaltung der Besucherströme entwickelt. Diese Software soll auf ihre Tauglichkeit für den Einsatz in diesem Zoo überprüft werden. Die Firma legt dazu einen Ausschnitt aus einem Implementationsdiagramm (Abbildung 2) vor. Eine Dokumentation der verwendeten Klassen findet sich im Anhang.

Name: _____

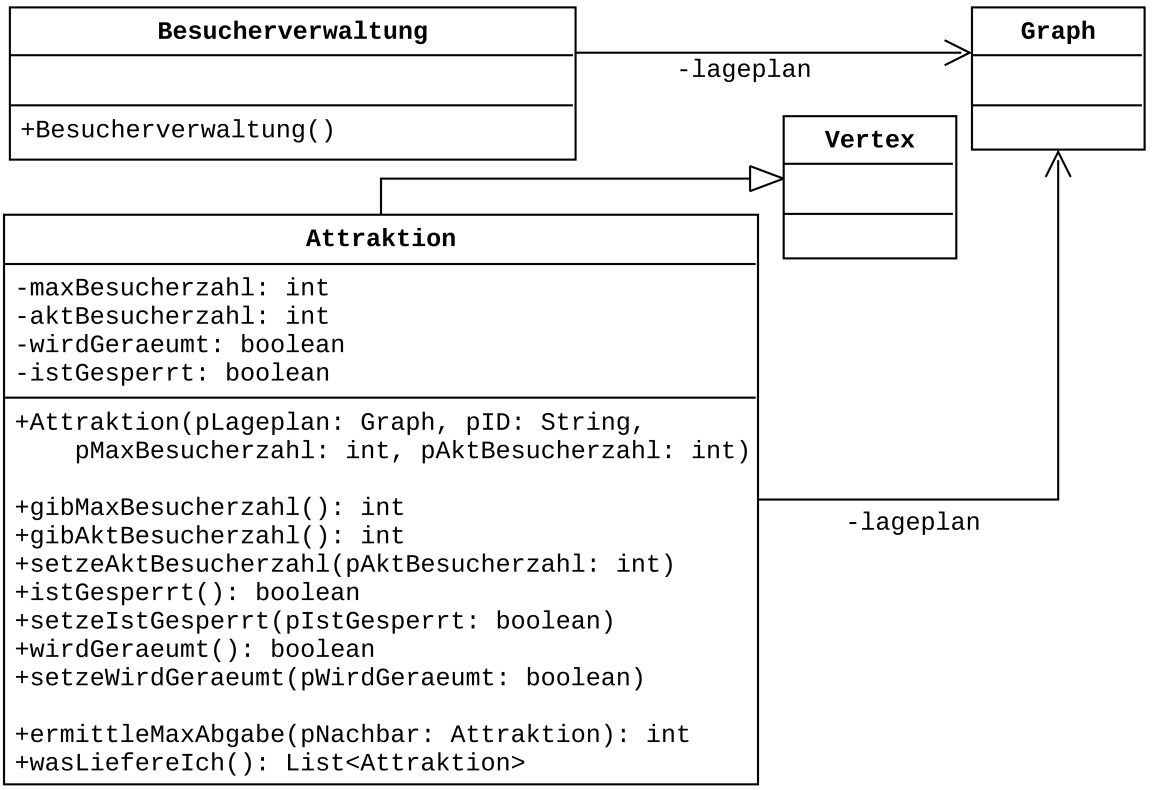


Abbildung 2: Ausschnitt aus dem Implementationsdiagramm

b) *Erläutern Sie die Beziehungen zwischen den Klassen* `Besucherverwaltung`, `Attraktion`, `Graph` *und* `Vertex` *in der durch Abbildung 2 gegebenen Teilmodellierung.*

Begründen Sie, warum es sinnvoll ist, dass jedes Objekt der Klasse `Attraktion` *in der gegebenen Modellierung Zugriff auf ein Objekt der Klasse* `Graph` *hat.*

(6 + 3 Punkte)

Ein Teil des Evakuierungskonzepts des Zoos beschäftigt sich mit der Notfall-Räumung einzelner Attraktionen. In einem solchen Fall kann mit einer Attraktion wie folgt verfahren werden: Eine Attraktion …

- **wird geräumt:** Alle Gäste, die sich an dieser Attraktion befinden, müssen sie verlassen. Dabei wird für die Simulation festgelegt, dass die Gäste die zur Verfügung stehenden Wege absteigend nach ihrer Kapazität nutzen, also zunächst den markierten Notausgang und als Ausweichmöglichkeit alle anderen Wege in absteigender Reihenfolge ihrer Kapazität. Zu räumende Attraktionen dürfen aber immer noch von Gästen neu betreten werden, wenn die Attraktion auf deren Fluchtweg zum Ausgang des Zoos liegt. Die Gäste aber dürfen dort nicht verbleiben, sondern müssen (wie alle anderen) die Attraktion schnellstmöglich verlassen.
- **ist gesperrt:** Gesperrte Attraktionen werden geräumt und dürfen von niemandem neu betreten werden.

Name: _____

Sobald bei einer Attraktion ein Alarm gemeldet wird, ist diese Attraktion gesperrt. Führt der von deren Gästen eingeschlagene Fluchtweg zu einer Attraktion, die diese momentan nicht mehr aufnehmen kann, so wird diese Attraktion geräumt (ist aber nicht gesperrt), um Platz zu schaffen.

c) In der Klasse `Attraktion` findet sich folgende undokumentierte Methode:

```
1  public List<Attraktion> wasLiefereIch() {
2    List<Attraktion> ret = new List<Attraktion>();
3    List<Vertex> alleNachbarn = lageplan.getNeighbours(this);
4    List<Vertex> nachbarn = new List<Vertex>();
5    alleNachbarn.toFirst();
6    while (alleNachbarn.hasAccess()) {
7      Attraktion aktN = (Attraktion) alleNachbarn.getContent();
8      if (!aktN.istGesperrt()) {
9        nachbarn.append(aktN);
10     }
11     alleNachbarn.next();
12   }
13   nachbarn.toFirst();
14   while (nachbarn.hasAccess()) {
15     Attraktion aktN = (Attraktion) nachbarn.getContent();
16     ret.toFirst();
17     while (ret.hasAccess()
18            && lageplan.getEdge(this, aktN).getWeight()
19               <= lageplan.getEdge(this, ret.getContent())
20                                                .getWeight()) {
21       ret.next();
22     }
23     if (ret.hasAccess()) {
24       ret.insert(aktN);
25     } else {
26       ret.append(aktN);
27     }
28     nachbarn.next();
29   }
30   return ret;
31 }
```

Analysieren Sie die Methode und erläutern Sie ihre Funktionsweise.

Ermitteln Sie schrittweise die Rückgabe der Methode, wenn diese für das Objekt aufgerufen wird, das in Abbildung 1 die Attraktion `Streicheltiere` *repräsentiert und diese die einzige Attraktion ist, die gesperrt ist.*

Erläutern Sie den Zweck der Methode im Sachkontext.

(5 + 5 + 4 Punkte)

Name: _____

d) Bei der Räumung einer Attraktion werden Gäste zu benachbarten Attraktionen evakuiert. Die Methode `ermittleMaxAbgabe` der Klasse `Attraktion` soll ermitteln, wie viele Gäste aktuell zu einer bestimmten Nachbarattraktion wechseln und dort aufgenommen werden können. Die Methode besitzt den folgenden Methodenkopf:

```
public int ermittleMaxAbgabe(Attraktion pNachbar)
```

Verweist `pNachbar` nicht auf eine benachbarte Attraktion, soll die Methode als Fehlerwert `-1` zurückliefern.

Entwickeln Sie einen algorithmischen Ansatz für diese Methode.

Implementieren Sie die Methode.

(5 + 7 Punkte)

e) Sollte ein schwerer Notfall eintreten, kann es sein, dass mehrere Attraktionen zur gleichen Zeit einen Alarm melden. Dadurch können Szenarien entstehen, in denen eine Räumung schwierig wird. Werden beispielsweise in Abbildung 1 gleichzeitig `Spielplatz` und `Toiletten` gesperrt, können die `Toiletten` nicht geräumt werden, da das Betreten der `Spielplatz`-Attraktion durch deren Sperrung nicht mehr möglich ist.
Die Softwarefirma geht davon aus, dass man dieses Problem durch Finden einer geeigneten Reihenfolge der notwendigen Sperrungen umgehen kann, deren Räumungen man zunächst Schritt für Schritt durchführt und erst dann mit der nächsten Sperrung fortfährt, wenn die zuvor gesperrte Attraktion erfolgreich geräumt wurde. Zur Umsetzung dieser Idee wird der Algorithmus gemäß Abbildung 3 vorgeschlagen:

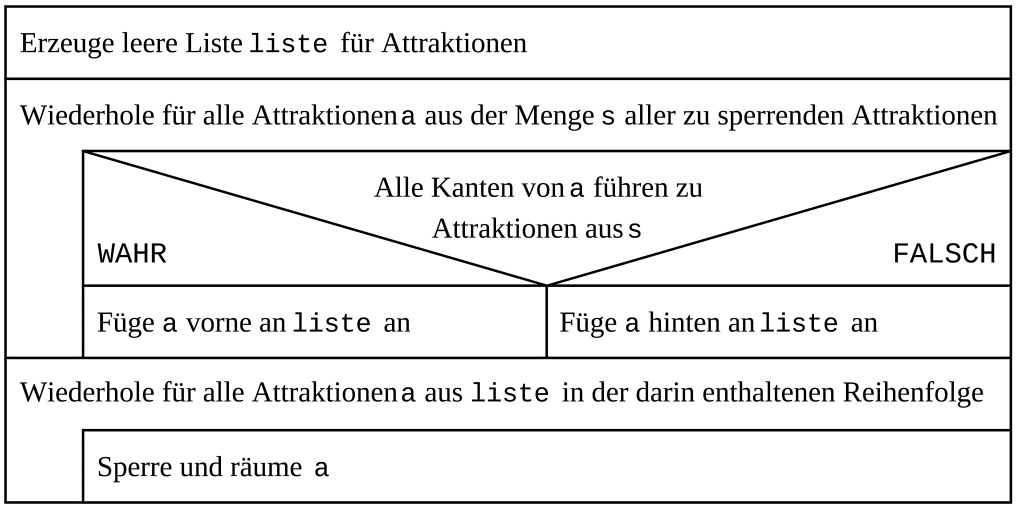


Abbildung 3: Algorithmus zur schrittweisen Räumung mehrerer Attraktionen

Name: _____

Der Algorithmus wird nun anhand des ursprünglichen Lageplans gemäß Abbildung 1 betrachtet.

Analysieren und erläutern Sie den Algorithmus im Sachkontext.

Beurteilen Sie den Algorithmus im Sachkontext vor dem Hintergrund folgender Szenarien:

i) Die Attraktionen `Streicheltiere`, `Ziegen` *und* `Esel` *melden Alarm.*

ii) Die Attraktionen `Streicheltiere`, `Spielplatz`, `Delfinarium` *und* `Esel` *melden Alarm.*

(4 + 5 Punkte)

Zugelassene Hilfsmittel:

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

Name: _____

Anhang

Dokumentationen der verwendeten Klassen

Die Klasse `Besucherverwaltung`

Ein Objekt dieser Klasse verwaltet den Lageplan des Zoos mit seinen Attraktionen und aktueller Anzahl von Besucherinnen und Besuchern.

Ausschnitt aus der Dokumentation der Klasse `Besucherverwaltung`

`Besucherverwaltung()`

Eine Besucherverwaltung mit dem aktuellen Lageplan des Zoos, allen Attraktionen und den gegenwärtigen Besucherzahlen wird initialisiert. Alle Attraktionen befinden sich im Normalbetrieb, d. h., sie sind nicht gesperrt und werden nicht geräumt.

Die Klasse `Attraktion`

Ein Objekt dieser Klasse verwaltet eine Attraktion des Zoos, an der sich Gäste sammeln können, z. B. ein Gehege oder einen besonderen Platz, wie Spielplätze oder Sanitärhäuser.

Ausschnitt aus der Dokumentation der Klasse `Attraktion`

`Attraktion(Graph pLageplan, String pID, int pMaxBesucherzahl, int pAktBesucherzahl)`

Eine Attraktion wird mit den angegebenen Werten in folgender Weise initialisiert:

- Die Attraktion befindet sich auf dem durch `pLageplan` angegebenen Lageplan.
- Die Attraktion bekommt die interne eindeutige Identifikationsbezeichnung `pID`.
- Die Attraktion kann maximal `pMaxBesucherzahl` Besucherinnen und Besucher zur gleichen Zeit aufnehmen. Ist diese Attraktion im Evakuierungsfall als Ausgang anzusehen, wird die Anzahl der maximalen Besucherinnen und Besucher auf `Integer.MAX_VALUE` festgelegt.
- Die Attraktion ist aktuell mit `pAktBesucherzahl` Besucherinnen und Besuchern besetzt.

`int gibMaxBesucherzahl()`

Die maximale Anzahl der Besucherinnen und Besucher, die diese Attraktion gleichzeitig betreten dürfen, wird zurückgegeben.

`int gibAktBesucherzahl()`

Die aktuelle Anzahl an Besucherinnen und Besuchern dieser Attraktion wird zurückgegeben.

`void setzeAktBesucherzahl(int pAktBesucherzahl)`

Die aktuelle Anzahl der Besucherinnen und Besucher dieser Attraktion wird auf den Wert `pAktBesucherzahl` gesetzt.

Name: _____

`boolean istGesperrt()`

Wenn diese Attraktion zum Zeitpunkt des Methodenaufrufs gesperrt ist, wird `true` zurückgegeben, ansonsten `false`. Ist die Attraktion gesperrt, darf sie nicht mehr betreten werden und wird so schnell wie möglich von noch darauf befindlichen Besucherinnen und Besuchern geräumt.

`void setzeIstGesperrt(boolean pIstGesperrt)`

Der Sperrstatus dieser Attraktion wird auf den von `pIstGesperrt` angegebenen Wert gesetzt. Ist eine Attraktion gesperrt, darf sie nicht mehr betreten werden und wird so schnell wie möglich von noch darauf befindlichen Besucherinnen und Besuchern geräumt.
Hat `pIstGesperrt` den Wert `true`, wird die Attraktion gleichzeitig als „zu räumen“ markiert (siehe Methoden `wirdGeraeumt` / `setzeWirdGeraeumt`). Hat `pIstGesperrt` den Wert `false`, wird der Räumungsstatus nicht verändert.

`boolean wirdGeraeumt()`

Wenn diese Attraktion zum Zeitpunkt des Methodenaufrufs geräumt werden muss, wird `true` zurückgegeben, ansonsten `false`. Muss eine Attraktion geräumt werden, müssen die noch darauf befindlichen Besucherinnen und Besucher diese so schnell wie möglich verlassen. Dieser Fall kann eintreten, wenn eine Attraktion gesperrt wird oder im Rahmen einer laufenden Evakuierung geräumt werden muss, damit sie Besucherinnen und Besucher von anderen Attraktionen aufnehmen kann, die ebenfalls geräumt werden müssen.

`void setzeWirdGeraeumt(boolean pWirdGeraeumt)`

Der Räumungsstatus dieser Attraktion wird auf den von `pWirdGeraeumt` angegebenen Wert gesetzt. Muss eine Attraktion geräumt werden, müssen die noch darauf befindlichen Besucherinnen und Besucher diese so schnell wie möglich verlassen. Dieser Fall kann eintreten, wenn eine Attraktion gesperrt wird oder im Rahmen einer laufenden Evakuierung geräumt werden muss, damit sie Besucherinnen und Besucher von anderen Attraktionen aufnehmen kann, die ebenfalls geräumt werden müssen.

`int ermittleMaxAbgabe(Attraktion pNachbar)`

Ermittelt die maximale Anzahl an Besucherinnen und Besuchern, die von dieser Attraktion aktuell an die durch `pNachbar` angegebene Nachbarattraktion abgegeben werden können. Falls `pNachbar` nicht auf eine Nachbarattraktion verweist, liefert die Methode `-1` zurück.

`List<Attraktion> wasLiefereIch()`

Diese Methode wird im Rahmen von Aufgabenteil c) analysiert.

Name: _____

Die Klasse `Graph`

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

Dokumentation der Klasse `Graph`

`Graph()`

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

`void addVertex(Vertex pVertex)`

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

`void addEdge(Edge pEdge)`

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

`void removeVertex(Vertex pVertex)`

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

`void removeEdge(Edge pEdge)`

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

`Vertex getVertex(String pID)`

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

`List<Vertex> getVertices()`

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

Name: _____

`List<Vertex> getNeighbours(Vertex pVertex)`

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

`List<Edge> getEdges()`

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

`List<Edge> getEdges(Vertex pVertex)`

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

`Edge getEdge(Vertex pVertex, Vertex pAnotherVertex)`

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

`void setAllVertexMarks(boolean pMark)`

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

`boolean allVerticesMarked()`

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

`void setAllEdgeMarks(boolean pMark)`

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

`boolean allEdgesMarked()`

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

`boolean isEmpty()`

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

Name: _____

Die Klasse `Vertex`

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

Dokumentation der Klasse `Vertex`

`Vertex(String pID)`

Ein neues Objekt vom Typ `Vertex` mit der ID `pID` wird erstellt. Seine Markierung hat den Wert `false`.

`String getID()`

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

`void setMark(boolean pMark)`

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

`boolean isMarked()`

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

Name: _____

Die Klasse `Edge`

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

Dokumentation der Klasse `Edge`

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

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

`void setWeight(double pWeight)`

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

`double getWeight()`

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

`Vertex[] getVertices()`

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

`void setMark(boolean pMark)`

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

`void setWeight(double pWeight)`

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

`boolean isMarked()`

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

Name: _____

Die generische Klasse `List`

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 kann 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>`

`List()`

Eine leere Liste wird erzeugt.

`boolean isEmpty()`

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

`boolean hasAccess()`

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

`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`.

`void toFirst()`

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

`void toLast()`

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

`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.

Name: _____

`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.

`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.

`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.

`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.

`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 2022

Informatik, Leistungskurs

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 2022

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 (Klasse `List`*)*
     - Nicht-lineare Strukturen
       Graphen (Klassen `Graph`, `Vertex`, `Edge`*)*

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

2. *Medien/Materialien*
   - entfällt

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

5. Zugelassene Hilfsmittel

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

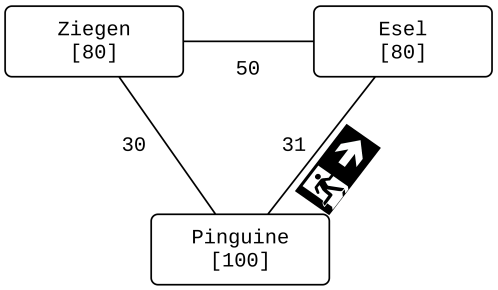
6. Modelllösungen

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

Teilaufgabe a)

Zum Delfinarium können sich pro Minute maximal 100 + 250 + 100 = 450 Personen begeben.

Der um die `Pinguine` erweiterte Ausschnitt aus der Planungskarte hat folgende Form:



Da die Kapazität des Weges von der Attraktion `Pinguine` zur Attraktion `Esel` mit `31` die höchste Kapazität eines von `Pinguine` wegführenden Weges hat, ist dieser Weg als Notausgang markiert.

Teilaufgabe b)

Die Klasse `Besucherverwaltung` verwaltet einen Graphen mit der Bezeichnung `lageplan`. Dieser modelliert den Plan des Zoos. Die Attraktionen werden als Knoten dieses Graphen modelliert und die Wege zwischen den Attraktionen als Kanten zwischen den Knoten, wobei deren Gewicht die Kapazität der Wege darstellt. Die im Sachkontext verwendeten Knoten sind Objekte der Klasse `Attraktion`, die von der Klasse `Vertex` erbt.

Objekte der Klasse `Attraktion` müssen Zugriff auf den Lageplan haben, da die Methode `ermittleMaxAbgabe` auf die Knoten und Kanten zugreifen muss, was durch Methoden der Klasse `Graph` gewährleistet wird.

Teilaufgabe c)

Die Methode erzeugt zunächst eine leere Rückgabeliste `ret` und besorgt sich eine Liste aller Nachbarattraktionen des aufrufenden Knotens (Z. 2 – 3). In der ersten Schleife (Z. 6 – 12) wird diese Liste durchgegangen und alle enthaltenen nicht gesperrten Attraktionen an eine weitere Liste `nachbarn` angehängt.
Im Anschluss wird diese Liste durchgegangen (Z. 13 – 29), also alle nicht gesperrten Nachbarn der Attraktion untersucht. Für jeden dieser Nachbarn `aktN` wird die aktuelle Rückgabeliste `ret` aufs Neue so weit durchlaufen, bis diese entweder am Ende angelangt ist oder das Gewicht der Kante zu `aktN` größer ist, als das Gewicht der Kante zum aktuellen Element von `ret` (Z. 17 – 22). `aktN` wird anschließend vor der Stelle eingefügt, an der die Rückgabeliste gerade ihr aktuelles Element hat (Z. 23 – 27).
Wird für die Attraktion `Streicheltiere` als Liste der Nachbarn beispielsweise [`Ziegen`, `Esel`, `Delfinarium`, `Spielplatz`] angenommen, wird die Rückgabeliste `ret` Schritt für Schritt folgendermaßen aufgebaut:

1. [`Ziegen`]
2. [`Ziegen`, `Esel`] (denn die Kante zu `Esel` hat ein geringeres Gewicht als die Kante zu `Ziegen`)
3. [`Delfinarium`, `Ziegen`, `Esel`] (denn die Kante zu `Delfinarium` hat ein größeres Gewicht als die Kante zu `Ziegen`)
4. [`Spielplatz`, `Delfinarium`, `Ziegen`, `Esel`] (denn die Kante zu `Spielplatz` hat ein größeres Gewicht als die Kante zu `Delfinarium`)

Die Methode liefert im Sachkontext die Nachbarattraktionen der aufrufenden Attraktion in der Reihenfolge zurück, wie sie im Falle einer Räumung der zugehörigen Attraktion von deren Gästen angesteuert würden, also in absteigender Kapazität der Wege, die jeweils dorthin führen.

Teilaufgabe d)

Die Abgabe von Personen an eine Nachbarattraktion ist durch drei Faktoren beschränkt:

- Die Nachbar-Attraktion hat eine maximale Besucherzahl und eine aktuelle Besucherzahl, so dass die maximal mögliche Aufnahme durch deren Differenzbetrag beschränkt ist.
- Der Weg, der beide Attraktionen verbindet, kann nur eine bestimmte Anzahl von Personen pro Minute transportieren.
- Die aufrufende Attraktion kann höchstens so viele Personen abgeben, wie sich gerade auf ihrem Gelände befinden.

Die Methode `ermittleMaxAbgabe` muss also den Minimalwert aus diesen drei beschränkenden Größen zurückliefern. Ist `pNachbar` entweder `null` oder ein Verweis auf einen Knoten, der keine Kante mit dem aufrufenden Knoten gemeinsam hat, muss der Wert `-1` zurückgegeben werden.

Eine mögliche Implementation der Methode lautet wie folgt:

```
public int ermittleMaxAbgabe(Attraktion pNachbar) {
  int ret = -1;
  if (pNachbar != null) {
    Edge weg = lageplan.getEdge(this,pNachbar);
    if (weg != null) {
      int freiePlaetze = pNachbar.gibMaxBesucherzahl()
                                  - pNachbar.gibAktBesucherzahl();
      int wegKapazitaet = (int)weg.getWeight();
      int maxAufnahme = Integer.min(freiePlaetze, wegKapazitaet);
      ret = Integer.min(maxAufnahme, aktBesucherzahl);
    }
  }
  return ret;
}
```

Teilaufgabe e)

Der Algorithmus geht Attraktionen, die Alarm gemeldet haben, in unbestimmter Reihenfolge durch und fügt sie entweder vorn oder hinten in die Liste `liste` ein.
Dabei werden diejenigen Attraktionen vorne eingefügt, die ausschließlich Kanten zu Attraktionen haben, die ebenfalls einen Alarm haben. Attraktionen mit dieser Eigenschaft könnten bei gleichzeitiger Sperrung der Nachbarattraktionen nicht geräumt werden, da deren Gelände nicht betreten werden darf.
Alle anderen Attraktionen werden ans Ende von `liste` gehängt.
Im Anschluss werden die Attraktionen in der durch die Prioritätenliste `liste` vorgegebenen Reihenfolge jeweils gesperrt und geräumt.
Im Resultat werden somit diejenigen Attraktionen, die im schlimmsten Fall keine direkten Auswege mehr hätten, als erste geräumt.

Im ersten Szenario werden `Ziegen` vorn in der Liste `liste` stehen, `Streicheltiere` und `Esel` in unbestimmter Reihenfolge dahinter. Dadurch können alle Gäste dieser Attraktionen evakuiert werden.

Im zweiten Szenario erfüllt keine der Attraktionen die Priorisierungsbedingung. Tritt so der Fall ein, dass die `Streicheltiere` erst nach `Spielplatz`, `Delfinarium` und `Esel` gesperrt und geräumt werden, können die Gäste dieser Attraktion eventuell nicht evakuiert werden, denn die einzige Ausweichattraktion ist `Ziegen`. Diese Attraktion kann aber nicht mehr als 80 Gäste aufnehmen, so dass die ggf. 100 Gäste der `Streicheltiere` auch im günstigsten Fall, dass das `Ziegen`-Gehege gerade keine Besucherinnen oder Besucher hat, nicht vollständig dorthin ausweichen können.

Der Algorithmus liefert also keine zuverlässige Lösung des Problems.

7. Teilleistungen – Kriterien / Bewertungsbogen zur Prüfungsarbeit

Name des Prüflings:_____ Kursbezeichnung:_____

Schule: _____

Teilaufgabe a)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**[2]	**ZK**	**DK**
1	gibt an, wie viele Besucherinnen und Besucher sich maximal pro Minute von den Nachbarattraktionen insgesamt zum `Delfinarium` begeben können.	2			
2	erweitert Abbildung 1 um die Attraktion `Pinguine` und bestimmt dabei die kleinstmögliche Kapazität des Weges von den Pinguinen zur `Esel`-Attraktion.	4			
Sachlich richtige Lösungsalternative zur Modelllösung: (6) …… ……					
	Summe Teilaufgabe a)	**6**			

Teilaufgabe b)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	erläutert die Beziehungen zwischen den Klassen `Besucherverwaltung`, `Attraktion`, `Graph` und `Vertex` in der durch Abbildung 2 gegebenen Teilmodellierung.	6			
2	begründet, warum es sinnvoll ist, dass jedes Objekt der Klasse `Attraktion` in der gegebenen Modellierung Zugriff auf ein Objekt der Klasse `Graph` hat.	3			
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 die Methode und erläutert ihre Funktionsweise.	5			
2	ermittelt schrittweise die Rückgabe der Methode, wenn sie für das Objekt aufgerufen wird, das in Abbildung 1 die Attraktion `Streicheltiere` repräsentiert und dieses die einzige Attraktion ist, die gesperrt ist.	5			
3	erläutert den Zweck der Methode im Sachkontext.	4			
Sachlich richtige Lösungsalternative zur Modelllösung: (14) …… ……					
	Summe Teilaufgabe c)	**14**			

Teilaufgabe d)

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

Teilaufgabe e)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	analysiert und erläutert den Algorithmus im Sachkontext.	4			
2	beurteilt den Algorithmus vor dem Hintergrund der angegebenen Szenarien.	5			
Sachlich richtige Lösungsalternative zur Modelllösung: (9) …… ……					
	Summe Teilaufgabe e)	**9**			

	Summe insgesamt	**50**			

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

	Lösungsqualität			
	maximal erreichbare Punktzahl	EK	ZK	DK
Übertrag der Punktsumme aus der ersten bearbeiteten Aufgabe	**50**			
Übertrag der Punktsumme aus der zweiten bearbeiteten Aufgabe	**50**			
Übertrag der Punktsumme aus der dritten bearbeiteten Aufgabe	**50**			
Punktzahl der gesamten Prüfungsleistung	**150**			
aus der Punktsumme resultierende Note gemäß nachfolgender Tabelle				
Note ggf. unter Absenkung um bis zu zwei Notenpunkte gemäß § 13 Abs. 2 APO-GOSt				
Paraphe				

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

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

Unterschrift, Datum:

Grundsätze für die Bewertung (Notenfindung)

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

Note	**Punkte**	**Erreichte Punktzahl**
sehr gut plus	15	150 – 143
sehr gut	14	142 – 135
sehr gut minus	13	134 – 128
gut plus	12	127 – 120
gut	11	119 – 113
gut minus	10	112 – 105
befriedigend plus	9	104 – 98
befriedigend	8	97 – 90
befriedigend minus	7	89 – 83
ausreichend plus	6	82 – 75
ausreichend	5	74 – 68
ausreichend minus	4	67 – 60
mangelhaft plus	3	59 – 50
mangelhaft	2	49 – 41
mangelhaft minus	1	40 – 30
ungenügend	0	29 – 0

Name: _____

Abiturprüfung 2022

Informatik, Leistungskurs

Aufgabenstellung:

Da Bewertungen von Restaurants durch Gäste in der Regel subjektiv sind, möchte der Projektkurs „Gastro-Informatik“ ein Bewertungssystem für Restaurants aufbauen, bei dem eine professionelle Gutachterin bzw. ein professioneller Gutachter die Speise- und Servicequalität sowie den Hygienezustand auf einer Skala zwischen 0 und 10 bewertet, wobei 10 die Bestnote darstellt.

Die Verwaltung der dabei anfallenden Daten soll in einer relationalen Datenbank umgesetzt werden, die der folgenden Teilmodellierung entspricht:

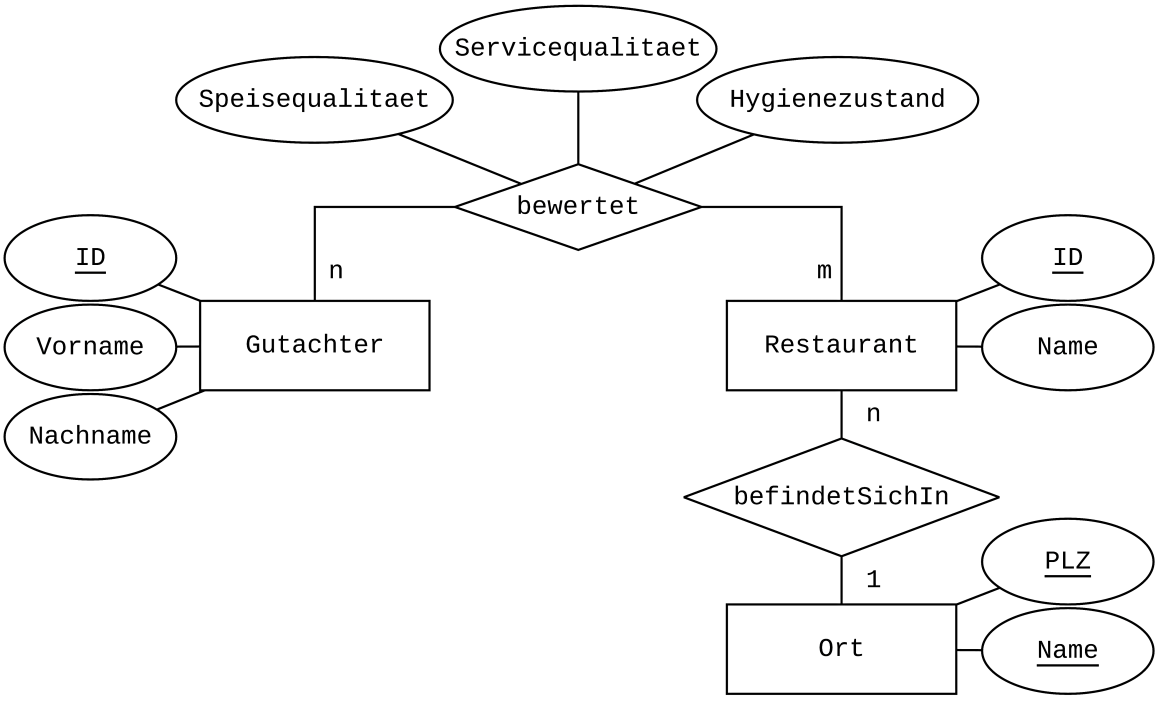


Abbildung 1: Teilmodellierung der Datenbank als Entity-Relationship-Diagramm

Name: _____

Das in Abbildung 2 gegebene Datenbankschema setzt die in Abbildung 1 gegebene Teilmodellierung als relationale Datenbank um und soll im Folgenden näher betrachtet werden. Beispieldaten zu diesem Datenbankschema sind in der Anlage zu finden.

```
Gutachter(ID, Vorname, Nachname)
bewertet(↑GutachterID, ↑RestaurantID, Speisequalitaet,
         Servicequalitaet, Hygienezustand)
Restaurant(ID, Name, ↑PLZ, ↑Ortsname)
Ort(PLZ, Name)
```

Abbildung 2: Datenbankschema eines Ausschnitts der Datenbank

a) *Beschreiben Sie die in Abbildung 1 als Entity-Relationship-Diagramm dargestellte Teilmodellierung.*

Erläutern Sie im Sachkontext die Beziehungstypen inklusive der zugehörigen Kardinalitäten aus dem Entity-Relationship-Diagramm in Abbildung 1.

Erläutern Sie, wie der 1:n-Beziehungstyp `befindetSichIn` *und der n:m-Beziehungstyp* `bewertet` *aus dem Entity-Relationship-Modell in Abbildung 1 im Datenbankschema in Abbildung 2 umgesetzt wurden.*

(3 + 4 + 4 Punkte)

b) Aus der im Anhang gegebenen Datenbank sollen folgende Informationen abgefragt werden:

i. Gesucht sind die Namen der Restaurants im PLZ-Bereich von einschließlich 44100 bis einschließlich 44300. Das Ergebnis soll nach den Restaurantnamen aufsteigend sortiert sein.
ii. Gesucht sind die Restaurants, zu denen noch keine Bewertung abgegeben wurde. Dabei sollen der Restaurantname, die PLZ und der Ortsname ausgegeben werden.
iii. Gesucht ist die Anzahl der Restaurants zu jeder PLZ. Dabei sollen auch PLZ, in denen es kein Restaurant gibt, mit aufgeführt sein. Das Ergebnis soll nach der Anzahl der Restaurants absteigend sortiert sein.

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

(3 + 4 + 5 Punkte)

Name: _____

c) Folgende SQL-Anweisung ist gegeben:

```
 1 SELECT Temp.Ortsname, MAX(Temp.d) AS Max, MIN(Temp.d) AS Min
 2 FROM (
 3     SELECT Restaurant.ID, Restaurant.Ortsname,
 4            AVG(bewertet.Speisequalitaet) AS d
 5     FROM bewertet
 6       INNER JOIN Restaurant
 7         ON bewertet.RestaurantID = Restaurant.ID
 8     GROUP BY Restaurant.ID
 9   ) AS Temp
10 GROUP BY Temp.Ortsname
```

Analysieren und erläutern Sie zunächst die Unterabfrage (Zeilen 3 bis 8) und anschließend die gesamte SQL-Anweisung.

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

(6 + 2 Punkte)

d) Nachdem das Restaurantbewertungssystem in der Praxis überprüft wurde, müssen folgende Aspekte verändert werden:

i. Eine Gutachterin bzw. ein Gutachter soll das gleiche Restaurant nach einigen Tagen oder Wochen erneut bewerten können. Außerdem soll für jede Bewertung die Möglichkeit bestehen, zusätzlich einen freien Bewertungstext eintragen zu können.

ii. Jede Gutachterin bzw. jeder Gutachter soll Wünsche bezüglich ihres bzw. seines Einsatzortes äußern dürfen. Für jede Gutachterin bzw. jeden Gutachter sollen daher mehrere bevorzugte Einsatzorte gespeichert werden.

iii. In jedem Jahr sollen die besten drei Restaurants in den Kategorien Speisequalität, Servicequalität und Hygienezustand besonders ausgezeichnet werden. Die Information, welches Restaurant in welchem Jahr in welcher Bewertungskategorie einen der ersten drei Plätze belegt hat, soll ebenfalls verwaltet werden.

Modellieren Sie die Erweiterung für das Datenbankschema aus Abbildung 2 so, dass der neue Datenbankentwurf zusätzlich die beschriebenen Aspekte enthält.

Erläutern Sie, wie das von Ihnen entwickelte Modell die Anforderungen umsetzt, und begründen Sie die gewählten Primärschlüssel für die Fälle, in denen sich deren Zusammensetzung ändert oder neue Relationenschemata hinzugefügt werden.

Hinweis: Relationenschemata, die unverändert bleiben, müssen nicht dargestellt werden.

(7 + 6 Punkte)

Name: _____

e) Ein Praktikant schlägt vor, das Datenbankschema aus Abbildung 2 durch seinen Vorschlag aus Abbildung 3 zu ersetzen.
Der Praktikant ist davon überzeugt, dass man mit seinem Vorschlag Speicherplatz einsparen und die Übersichtlichkeit erhöhen könne. Man müsse schließlich deutlich weniger Attribute verwalten, weil zusätzliche Fremdschlüssel entfallen. Das Datenbankschema in Abbildung 3 sei zudem einfacher zu lesen und beinhalte im Vergleich zum Datenbankschema in Abbildung 2 keine nennenswerten Nachteile.

```
Gutachter(ID, Vorname, Nachname)
Restaurantbewertung(↑GutachterID, RestaurantID,
                    Restaurantname, PLZ, Ortsname,
                    Speisequalitaet, Servicequalitaet,
                    Hygienezustand)
```

Abbildung 3: Datenbankschema eines Ausschnitts der Datenbank

Beurteilen Sie den Vorschlag des Praktikanten mit Blick auf die vorgebrachten Argumente und unter Berücksichtigung der Normalformen.

(6 Punkte)

Zugelassene Hilfsmittel:

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

Name: _____

Anlage:

Ausschnitt aus den Beispieldaten zum Datenbankschema aus Abbildung 2

Gutachter		
ID	**Vorname**	**Nachname**
1	Ada	Lovelace
2	John	Von Neumann
3	Alan	Turing
4	Konrad	Zuse
5	Tim	Berners-Lee

bewertet				
GutachterID	**RestaurantID**	**Speise-qualitaet**	**Service-qualitaet**	**Hygiene-zustand**
1	2	7	8	6
4	3	1	8	10
4	2	8	5	5
3	3	10	7	8
2	2	5	8	4
1	1	8	9	7

Restaurant			
ID	**Name**	**PLZ**	**Ortsname**
1	Meisterpizza	44269	Dortmund
2	Sushi4Friends	52525	Heinsberg
3	Veggie-Haus	31137	Hildesheim
4	Der Grieche	44135	Dortmund
5	Burgerhaus	52525	Waldfeucht

Ort	
PLZ	**Name**
31137	Hildesheim
44135	Dortmund
44269	Dortmund
52525	Heinsberg
52525	Waldfeucht

Unterlagen für die Lehrkraft

Abiturprüfung 2022

Informatik, Leistungskurs

1. Aufgabenart

Analyse, Modellierung und Implementation kontextbezogener Problemstellungen mit Schwerpunkt auf dem Inhaltsfeld Daten und ihre Strukturierung

2. Aufgabenstellung[1]

siehe Prüfungsaufgabe

3. Materialgrundlage

entfällt

4. Bezüge zum Kernlehrplan und zu den Vorgaben 2022

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

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

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

2. *Medien/Materialien*
   - entfällt

5. Zugelassene Hilfsmittel

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

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

6. Modelllösungen

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

Teilaufgabe a)

Die in Abbildung 1 gegebene Teilmodellierung besteht aus drei Entitätstypen und zwei Beziehungstypen.
Im Entitätstyp `Gutachter` wird als Primärschlüssel `ID` verwendet.
Im Entitätstyp `Restaurant` wird als Primärschlüssel `ID` verwendet.
Im Entitätstyp `Ort` werden als kombinierter Primärschlüssel `PLZ` und `Name` verwendet.
Entitäten des Typs `Gutachter` haben die weiteren Attribute `Vorname` und `Nachname`.
Entitäten des Typs `Restaurant` haben das weitere Attribut `Name`.

Der Beziehungstyp `bewertet` modelliert, welche Gutachterin oder welcher Gutachter welches Restaurant bewertet. Die Attribute des Beziehungstyps `Speisequalitaet`, `Servicequalitaet` und `Hygienezustand` verwalten die Bewertungen in den einzelnen Kategorien.
Eine Gutachterin bzw. ein Gutachter kann mehrere Restaurants bewerten und ein Restaurant kann von mehreren Gutachterinnen bzw. Gutachtern bewertet werden.

Der Beziehungstyp `befindetSichIn` modelliert, welches Restaurant sich in welchem Ort befindet.
Ein Restaurant befindet sich in genau einem Ort. In einem Ort können sich mehrere Restaurants befinden.

Der 1:n-Beziehungstyp `befindetSichIn` wurde so umgesetzt, dass in der Relation `Restaurant` zusätzlich die Fremdschlüsselattribute `PLZ` und `Ortsname` verwaltet werden, welche den Primärschlüssel aus der Relation `Ort` abbilden.

Der n:m-Beziehungstyp `bewertet` wurde mit dem Relationenschema `bewertet` umgesetzt. In dieser Relation wird ein kombinierter Primärschlüssel aus den zwei Fremdschlüsselattributen `GutachterID` und `RestaurantID` verwendet, welche sich jeweils auf die Primärschlüssel einmal in der Relation `Gutachter` und einmal in der Relation `Restaurant` beziehen.

Teilaufgabe b)

Die folgenden SQL-Anweisungen realisieren die Anfragen:

i.
```
SELECT Name
FROM Restaurant
WHERE PLZ >= 44100 AND PLZ <= 44300
ORDER BY Name ASC
```

ii.
```
SELECT Restaurant.Name, Restaurant.PLZ, Restaurant.Ortsname
FROM Restaurant
WHERE Restaurant.ID NOT IN (
    SELECT DISTINCT bewertet.RestaurantID
    FROM bewertet
  )
```

iii.
```
SELECT Ort.PLZ, COUNT(Restaurant.ID) AS Anzahl
FROM Ort
  LEFT JOIN Restaurant
    ON Ort.PLZ = Restaurant.PLZ
    AND Ort.Name = Restaurant.Ortsname
GROUP BY Ort.PLZ
ORDER BY Anzahl DESC
```

Teilaufgabe c)

Die Unterabfrage verknüpft mit einem `INNER JOIN` die Relationen `bewertet` und `Restaurant` über die ID der Restaurants (vgl. Zeilen 5 bis 7). Außerdem werden die Datensätze nach der ID der Restaurants gruppiert (vgl. Zeile 8), um für jedes Restaurant die durchschnittliche Bewertung der Speisequalität mit der Aggregatfunktion `AVG` zu ermitteln (vgl. Zeile 4). Diese wird mit `d` bezeichnet und zusammen mit der ID und dem zugehörigen Ortsnamen des Restaurants zurückgeliefert (vgl. Zeilen 3 bis 4).

Die gesamte SQL-Anweisung gruppiert die mit `Temp` bezeichnete Ergebnisrelation der Unterabfrage nach den Ortsnamen (vgl. Zeilen 9 bis 10). Die Aggregatfunktionen `MAX` und `MIN` stellen sicher, dass für jede Gruppe (d. h. für jeden Ortsnamen) ein Datensatz mit der höchsten (als `Max` bezeichneten) und der niedrigsten (als `Min` bezeichneten) durchschnittlichen Bewertung der Speisequalität zurückgeliefert wird (vgl. Zeile 1). Zusätzlich wird der zugehörige Ortsname zurückgeliefert (vgl. Zeile 1).

Gesucht ist zu jedem Ortsnamen die beste und die schlechteste durchschnittliche Bewertung der Speisequalität (bzw. die Bandbreite dieses Durchschnittswerts).

Teilaufgabe d)

Das folgende ausschnitthafte Datenbankschema stellt eine mögliche Erweiterung dar.

bewertet(↑GutachterID, ↑RestaurantID, Datum,
Speisequalitaet, Servicequalitaet, Hygienezustand,
Bewertungstext)
wuenscht(↑GutachterID, ↑PLZ, ↑Ortsname)
Auszeichnung(Kategorie)
wurdeAusgezeichnet(↑RestaurantID, ↑AuszeichnungKategorie,
Jahr, Platz)

i.
Das Relationenschema `bewertet` wird um das Attribut `Bewertungstext` und das zusätzliche Primärschlüsselattribut `Datum` erweitert. Der gesamte Schlüssel (`Datum`, `GutachterID` und `RestaurantID`) identifiziert somit eine Bewertung. Dadurch wird sichergestellt, dass eine Gutachterin bzw. ein Gutachter das gleiche Restaurant an unterschiedlichen Daten mehrfach bewerten kann.

ii.
Das neue Relationenschema `wuenscht` stellt einen n:m-Beziehungstyp zwischen den Relationen `Gutachter` und `Ort` dar, das als Fremdschlüssel die entsprechenden Primärschlüssel der genanten Relationen enthält. Alle Fremdschlüsselattribute bilden den kombinierten Primärschlüssel der neuen Relation.

iii.
Das neue Relationenschema `Auszeichnung` verwaltet das Attribut `Kategorie`.
Das neue Relationschema `wurdeAusgezeichnet` stellt einen n:m-Beziehungstyp zwischen den Relationen `Restaurant` und `Auszeichnung` dar, das als Fremdschlüssel die entsprechenden Primärschlüssel der genanten Relationen enthält. Der Primärschlüssel besteht aus dem Fremdschlüsselattribut `AuszeichnungKategorie` sowie den beiden Attributen `Jahr` und `Platz`. So wird sichergestellt, dass ein Restaurant im gleichen Jahr unterschiedliche Auszeichnungen und in unterschiedlichen Jahren auch erneut die gleiche Auszeichnung erhalten kann.

Teilaufgabe e)

Die vom Praktikanten vorgeschlagene Modellierung ist aus folgenden Gründen abzulehnen:
Die Relation `Restaurantbewertung` würde sich aufgrund eines atomaren Wertebereichs der Attribute in der ersten Normalform befinden. Allerdings verstößt sie gegen die zweite Normalform, da die Attribute `Restaurantname`, `PLZ` und `Ortsname` bereits von einem Teil des Primärschlüssels, nämlich vom Schlüsselattribut `RestaurantID`, funktional abhängig sind.
Speicherplatz lässt sich entgegen seiner Behauptung nicht einsparen. Auch wenn bei dem Vorschlag durch teilweise wegfallende Fremdschlüsselbeziehungen weniger Attribute verwaltet werden, würden einzelne Informationen in den Datensätzen unnötig redundant gespeichert. Beispielsweise würden bei jeder Bewertung eines Restaurants neben dem Restaurantnamen insbesondere die PLZ und der Ortsname redundant verwaltet.
Wurde ein Restaurant beispielsweise mehrfach bewertet, können Anomalien (z. B. bei der Änderung des Restaurantnamens) auftreten und inkonsistente Datensätze (z. B. bei fehlerhaften Eingaben eines Restaurantnamens) entstehen.
Die Übersichtlichkeit im Modellierungsprozess stellt grundsätzlich kein Kriterium zur Bewertung der Güte eines Modells dar.
Darüber hinaus ist es mit dem vorgeschlagenen Datenbankschema nicht mehr möglich, Restaurants unabhängig vom Vorliegen mindestens einer Bewertung zu speichern, weshalb unbewertete Restaurants nicht mehr abgebildet werden können. Auch Orte, in denen bisher kein bewertetes Restaurant existiert, können nicht mehr erfasst 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	beschreibt die in Abbildung 1 als Entity-Relationship-Diagramm dargestellte Teilmodellierung.	3			
2	erläutert im Sachkontext die Beziehungstypen inklusive der zugehörigen Kardinalitäten aus dem Entity-Relationship-Diagramm in Abbildung 1.	4			
3	erläutert, wie der 1:n-Beziehungstyp `befindetSichIn` und der n:m-Beziehungstyp `bewertet` aus dem Entity-Relationship-Modell in Abbildung 1 im Datenbankschema in Abbildung 2 umgesetzt wurden.	4			
Sachlich richtige Lösungsalternative zur Modelllösung: (11) …… ……					
	Summe Teilaufgabe a)	**11**			

Teilaufgabe b)

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

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

Teilaufgabe c)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	analysiert und erläutert die Unterabfrage.	3			
2	analysiert und erläutert die gesamte SQL-Anweisung.	3			
3	erläutert im Sachzusammenhang, welche Information die gesamte SQL-Anweisung ermittelt.	2			
Sachlich richtige Lösungsalternative zur Modelllösung: (8) …… ……					
	Summe Teilaufgabe c)	**8**			

Teilaufgabe d)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	modelliert die Erweiterung für das Datenbankschema aus Abbildung 2 so, dass der neue Datenbankentwurf zusätzlich den Aspekt i enthält.	2			
2	modelliert die Erweiterung für das Datenbankschema aus Abbildung 2 so, dass der neue Datenbankentwurf zusätzlich den Aspekt ii enthält.	2			
3	modelliert die Erweiterung für das Datenbankschema aus Abbildung 2 so, dass der neue Datenbankentwurf zusätzlich den Aspekt iii enthält.	3			
4	erläutert, wie das entwickelte Modell die Anforderungen umsetzt, und begründet die gewählten Primärschlüssel.	6			
Sachlich richtige Lösungsalternative zur Modelllösung: (13) …… ……					
	Summe Teilaufgabe d)	**13**			

Teilaufgabe e)

	Anforderungen	Lösungsqualität			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	beurteilt den Vorschlag des Praktikanten mit Blick auf die vorgebrachten Argumente und unter Berücksichtigung der Normalformen.	6			
Sachlich richtige Lösungsalternative zur Modelllösung: (6) …… ……					
	Summe Teilaufgabe e)	**6**			

	Summe insgesamt	**50**			

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

	Lösungsqualität			
	maximal erreichbare Punktzahl	**EK**	**ZK**	**DK**
Übertrag der Punktsumme aus der ersten bearbeiteten Aufgabe	**50**			
Übertrag der Punktsumme aus der zweiten bearbeiteten Aufgabe	**50**			
Übertrag der Punktsumme aus der dritten bearbeiteten Aufgabe	**50**			
Punktzahl der gesamten Prüfungsleistung	**150**			
aus der Punktsumme resultierende Note gemäß nachfolgender Tabelle				
Note ggf. unter Absenkung um bis zu zwei Notenpunkte gemäß § 13 Abs. 2 APO-GOSt				
Paraphe				

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

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

Unterschrift, Datum:

Grundsätze für die Bewertung (Notenfindung)

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

Note	**Punkte**	**Erreichte Punktzahl**
sehr gut plus	15	150 – 143
sehr gut	14	142 – 135
sehr gut minus	13	134 – 128
gut plus	12	127 – 120
gut	11	119 – 113
gut minus	10	112 – 105
befriedigend plus	9	104 – 98
befriedigend	8	97 – 90
befriedigend minus	7	89 – 83
ausreichend plus	6	82 – 75
ausreichend	5	74 – 68
ausreichend minus	4	67 – 60
mangelhaft plus	3	59 – 50
mangelhaft	2	49 – 41
mangelhaft minus	1	40 – 30
ungenügend	0	29 – 0

Name: _____

Abiturprüfung 2022

Informatik, Leistungskurs

Aufgabenstellung:

Radioteleskope sind ein wichtiger Bestandteil moderner Astronomie. Mit riesigen Antennen zeichnen sie Signale im Frequenzbereich zwischen `0.01` GHz und `100.00` GHz auf und geben diese Daten an zahlreiche Forschungsprojekte weiter. Eines dieser Projekte ist das sogenannte SETI-Projekt (Search for Extraterrestrial Intelligence), bei dem man sich mit der Möglichkeit befasst, dass manche der empfangenen Radiosignale Kontaktversuche außerirdischer Zivilisationen sein könnten.

Ein einfaches Signalprotokoll der Frequenzen könnte wie folgt aussehen:

```
43.22;0.10;3.21;1.53;14.22
```

Das Protokoll entsteht, indem einmal pro Sekunde ein neuer Eintrag mit dem jeweils stärksten Signal durch ein Semikolon abgetrennt ergänzt wird. Wird ein Signal empfangen, so wird die Frequenz auf zwei Nachkommastellen genau angefügt. Wird kein Signal empfangen, wird `0.00` angefügt. Signalprotokolle können prinzipiell beliebig lang sein.

a) Bei der Übertragung von Signalprotokollen kommt es häufig zu Fehlern, die später die Auswertung der Protokolle behindern. Die Signalprotokolle werden daher auf Korrektheit geprüft. Der Automat A_1 zum Zustandsübergangsdiagramm in Abbildung 1 wird dafür verwendet. Er erfüllt seine Aufgabe aber nur teilweise.

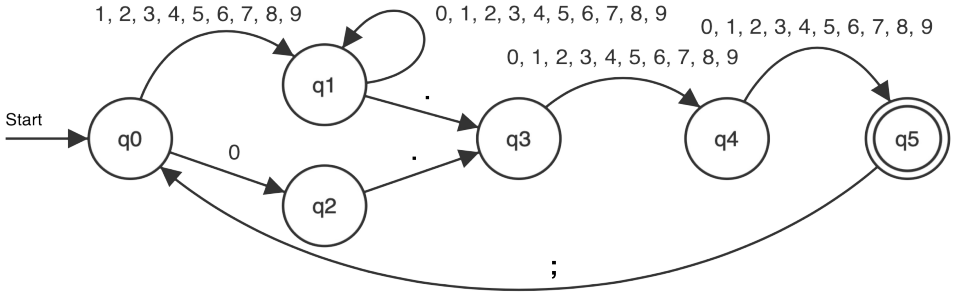


Abbildung 1: Zustandsübergangsdiagramm zum Automaten A_1

Geben Sie Startzustand, Endzustände, Eingabealphabet und Zustandsmenge des Automaten A_1 an.

Analysieren Sie den Automaten A_1 und erläutern Sie, wie eine Eingabezeichenfolge aufgebaut sein muss, um als fehlerfrei akzeptiert zu werden.

Ermitteln Sie eine Zeichenfolge, die vom Automaten A_1 akzeptiert wird, obwohl sie kein korrektes Signalprotokoll darstellt.

(3 + 3 + 3 Punkte)

Name: _____

Beim SETI-Projekt sucht man in Signalprotokollen in erster Linie nach Frequenzen zwischen einschließlich `1.42` GHz und einschließlich `1.66` GHz, dem sogenannten *Wasserloch*. In diesem Frequenzbereich existieren besonders wenig störende natürliche Signale, so dass man davon ausgeht, dass eine außerirdische Zivilisation diesen Frequenzbereich zur Kontaktaufnahme nutzen würde.

b) Ein Mitarbeiter hat die Idee, auch für die inhaltliche Auswertung einen Automaten zur Hilfe zu nehmen. Der nichtdeterministische endliche Automat A_2 soll von allen einfachen Signalprotokollen diejenigen akzeptieren, in denen mindestens ein Signal im Wasserloch-Frequenzbereich enthalten ist. Die formale Korrektheit des Protokolls kann vorausgesetzt werden, so dass sie vom Automaten A_2 nicht überprüft werden muss.

Entwerfen Sie den nichtdeterministischen endlichen Automaten A_2 entsprechend den obigen Vorgaben.

Begründen Sie, dass es sich bei Ihrem Automaten A_2 um einen nichtdeterministischen endlichen Automaten (NEA) handelt.

Zeigen Sie, dass Ihr Automat A_2 das Signalprotokoll `9.01;1.52;0.00` *akzeptiert.*

(7 + 3 + 4 Punkte)

Obwohl der Wasserloch-Frequenzbereich relativ wenig natürliche Störquellen beinhaltet, kommt es immer wieder zu künstlichen Störsignalen, die in der Umgebung des Radioteleskops durch technische Geräte (z. B. Mikrowellenöfen) entstehen. Im Verhältnis zu Signalen, die aus großer Entfernung zu uns kommen, sind diese Störsignale meist sehr energiereich, so dass man sie oft anhand der Signalstärke identifizieren kann.

Beim SETI-Projekt beschließt man daher, in Zukunft mit einem erweiterten Signalprotokoll zu arbeiten, in dem neben der Frequenz auch die Signalstärke verzeichnet ist. Ein solches erweitertes Signalprotokoll könnte wie folgt aussehen:

`(43.22:2);(0.10:1);(3.21:4);(0.00:0);(1.53:b);(14.22:9)`

Hinter jedem Frequenzeintrag steht nun durch einen Doppelpunkt abgetrennt die Stärke des Signals in den Stufen

`0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f`

wobei `1` für ein sehr schwaches und `f` für ein sehr starkes Signal steht. Die Signalstärke `0` wird eingetragen, wenn kein Signal aufgefangen wurde. Darüber hinaus wird jeder so erweiterte Frequenzeintrag in Klammern gesetzt.

Name: _____

c) Die folgende Grammatik G_1 erzeugt die Sprache der erweiterten Signalprotokolle.

Startsymbol: `S`

Terminalsymbole:
```
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9,
 a, b, c, d, e, f, (, ), ., :, ;}
```

Nichtterminalsymbole: `{S, F, I, N, Z, B}`

Produktionen:
```
{S → F:I | F:I;S,
 F → (ZN.NN,
 I → N) | B),
 N → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9,
 Z → ε | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9,
 B → a | b | c | d | e | f
}
```

Erläutern Sie die Produktionen der Grammatik G_1.

Begründen Sie, dass die Grammatik G_1 *nicht regulär ist.*

Begründen Sie, dass die Sprache der Grammatik G_1 *regulär ist.*

(5 + 3 + 4 Punkte)

d) Da Signale mit zu großer Signalstärke nicht als Kontaktversuch in Frage kommen, sollen nur schwache Signale im Wasserloch-Frequenzbereich mit den Signalstärken `1` oder `2` untersucht werden. Um solche Signaleinträge aus einem erweiterten Signalprotokoll zu ermitteln, soll die Java-Methode `ermittleMoeglicheKontakte` der Klasse `Protokollpruefer` implementiert werden. Sie soll sowohl die Signalfrequenz wie auch die Signalstärke überprüfen.

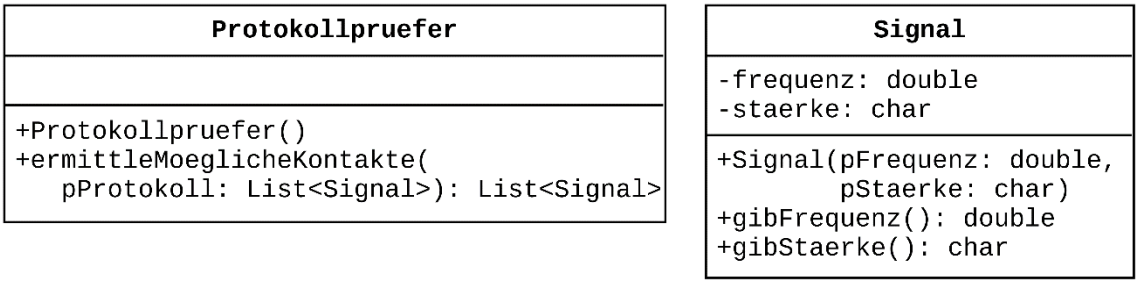


Abbildung 2: Teilmodellierung eines Programms zum Prüfen von Signalprotokollen

Die Methode verfügt über den folgenden Methodenkopf:

```
public List<Signal> ermittleMoeglicheKontakte(
                                    List<Signal> pProtokoll)
```

Entwickeln Sie für die Methode `ermittleMoeglicheKontakte` *ein algorithmisches Verfahren.*

Implementieren Sie die Methode `ermittleMoeglicheKontakte` *entsprechend Ihrem Verfahren.*

(4 + 5 Punkte)

Name: _____

e) Schnell stellt sich heraus, dass die Signalstärke kein zuverlässiges Kriterium zum Identifizieren von Störsignalen ist. Auch Störsignale können schwach sein und vielleicht werden Kontaktversuche durch Außerirdische sehr energiereich abgestrahlt.

Daher möchte man beim SETI-Projekt in Zukunft eine andere Strategie verfolgen und ein zusätzliches Radioteleskop in Nordafrika verwenden. Auf Grund seiner abgelegenen Lage hofft man, dass es weniger Störsignale empfängt.

Um das zu überprüfen, soll ein Protokoll aus Nordafrika mit einem Ausrufezeichen `(!)` getrennt an ein Protokoll aus Europa angehängt werden. Beide Protokolle müssen exakt denselben Zeitraum protokollieren.

Beispiel:

`3.21;0.00;1.53;14.22` (Europa)

`64.22;0.12;1.53;23.11` (Nordafrika)

`3.21;0.00;1.53;14.22!64.22;0.12;1.53;23.11` (kombiniert)

Mit Hilfe eines Kellerautomaten soll nun geprüft werden, ob das Protokoll aus Nordafrika öfter den Eintrag `0.00` hat als das Protokoll aus Europa und der Standort in Afrika somit „ruhiger“ ist.

Beurteilen Sie, ob ein Kellerautomat durch Überprüfung des kombinierten Protokolls entscheiden kann, ob der Standort in Nordafrika ruhiger ist.

(6 Punkte)

Zugelassene Hilfsmittel:

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

Name: _____

Anhang

Die Klasse `Protokollpruefer`

Ein Objekt der Klasse `Protokollpruefer` wird verwendet, um erweiterte Signalprotokolle auf eventuelle Kontaktversuche von außerirdischen Zivilisationen zu überprüfen.

Dokumentation der Klasse `Protokollpruefer`

`Protokollpruefer()`

Das Objekt der Klasse `Protokollpruefer` wird initialisiert.

`List<Signal> ermittleMoeglicheKontakte(List<Signal> pProtokoll)`

Die Methode liefert eine Liste aller Signaleinträge aus dem Signalprotokoll `pProtokoll`, deren Frequenzen zwischen einschließlich `1.42` GHz und einschließlich `1.66` GHz liegen und die Signalstärke `1` oder `2` haben.

Die Klasse `Signal`

Ein Objekt der Klasse `Signal` speichert einen einzelnen Signaleintrag bestehend aus einer Signalfrequenz und einer Signalstärke.

Dokumentation der Klasse `Signal`

`Signal(double pFrequenz, char pStaerke)`

Das Objekt der Klasse `Signal` wird mit `pFrequenz` als Signalfrequenz und `pStaerke` als Signalstärke initialisiert.

`double gibFrequenz()`

Die Methode liefert die Signalfrequenz des Eintrags.

`char gibStaerke()`

Die Methode liefert die Signalstärke des Eintrags.

Name: _____

Die generische Klasse `List`

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 kann 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>`

`List()`

Eine leere Liste wird erzeugt.

`boolean isEmpty()`

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

`boolean hasAccess()`

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

`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`.

`void toFirst()`

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

`void toLast()`

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

Name: _____

`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.

`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.

`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.

`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.

`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.

`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 2022

Informatik, Leistungskurs

1. Aufgabenart

Analyse, Modellierung und Implementation kontextbezogener Problemstellungen mit Schwerpunkt auf den Inhaltsfeldern Algorithmen und Formale Sprachen und Automaten

2. Aufgabenstellung[1]

siehe Prüfungsaufgabe

3. Materialgrundlage

entfällt

4. Bezüge zum Kernlehrplan und zu den Vorgaben 2022

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*
   Algorithmen
   - Analyse, Entwurf und Implementierung von Algorithmen

   Formale Sprachen und Automaten
   - Syntax und Semantik einer Programmiersprache
     - Java
   - Endliche Automaten und Kellerautomaten
     - Deterministische endliche Automaten
     - Nichtdeterministische endliche Automaten
     - Nichtdeterministische Kellerautomaten
   - Grammatiken regulärer Sprachen
     - Produktionen mit ε
   - Möglichkeiten und Grenzen von Automaten und formalen Sprachen

2. *Medien/Materialien*
   - entfällt

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

5. Zugelassene Hilfsmittel

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

6. Modelllösungen

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

Teilaufgabe a)

Startzustand: q_0
Endzustände: $\{q_5\}$
Eingabealphabet: $\{., ;, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
Zustandsmenge: $\{q_0, q_1, q_2, q_3, q_4, q_5\}$

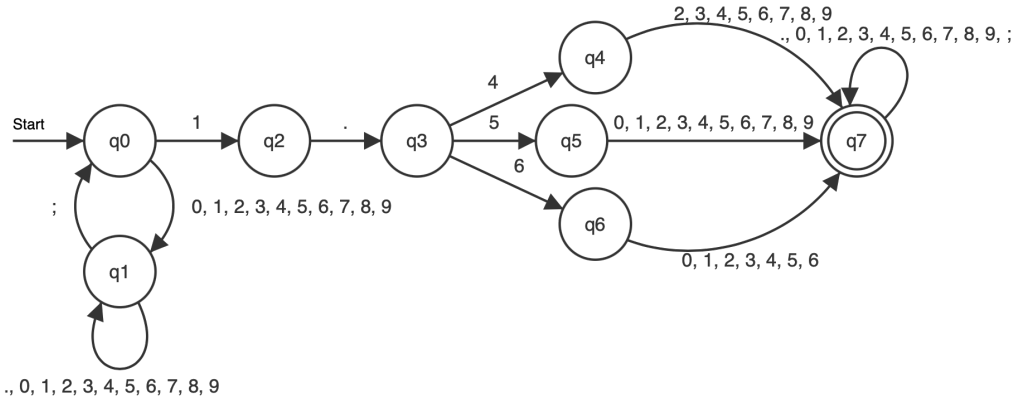
Der Automat A_1 akzeptiert alle Zeichenketten, die aus einem oder mehreren mit Semikolon getrennten Einträgen der folgenden Form bestehen.
Beginnt ein Eintrag mit einer Null, so muss ein Punkt folgen. Beginnt ein Eintrag mit einer Ziffer ungleich der Null, kann ein Punkt oder eine beliebig lange Folge der Ziffern `0` bis `9` mit einem anschließenden Punkt folgen. Nach dem Punkt müssen zwei weitere Ziffern von `0` bis `9` folgen.

Die folgende Zeichenfolge wird vom Automaten A_1 akzeptiert: `0.34;245.23;12.33`
Der zweite Signaleintrag kann nicht korrekt sein, da seine Frequenz außerhalb des Empfangsbereichs der Radioantenne liegt.

Teilaufgabe b)

Der folgende Automat A_2 entspricht den Anforderungen:

Startzustand: q_0
Endzustände: $\{q_7\}$
Eingabealphabet: $\{., ;, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
Zustandsmenge: $\{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7\}$
Zustandsübergangsdiagramm:



Bei dem endlichen Automaten A_2 handelt es sich um einen NEA, da es z. B. vom Zustand q_0 aus mit dem Zeichen `1` möglich ist, in q_1 oder aber in q_2 zu wechseln. Der Automat ist an dieser Stelle nicht deterministisch.

Der Automat A_2 erkennt das Wort `9.01;1.52;0.00` wie folgt:

$$q_0 \xrightarrow{9} q_1 \xrightarrow{.} q_1 \xrightarrow{0} q_1 \xrightarrow{1} q_1 \xrightarrow{;} q_0 \xrightarrow{1} q_2 \xrightarrow{.} q_3 \xrightarrow{5} q_5 \xrightarrow{2} q_7 \xrightarrow{;} q_7 \xrightarrow{0} q_7 \xrightarrow{.} q_7 \xrightarrow{0} q_7 \xrightarrow{0} q_7$$

Teilaufgabe c)

Aus dem Startsymbol `S` kann eine Frequenz `F` und eine mit dem Zeichen `:` abgetrennte Signalstärke `I` abgeleitet werden. Darüber hinaus kann das Startsymbol `S` mit dem Zeichen `;` ein weiteres Mal angehängt werden, so dass eine beliebig lange Kette von Signaleinträgen entstehen kann.
Das Nichtterminal `F` wird zu einem einzelnen Frequenzeintrag `ZN.NN` abgeleitet, vor dem eine öffnende Klammer `(` stehen muss. Das Nichtterminal `I` wird zu einem `N` oder einem `B`, die beide Frequenzstärken darstellen können, abgeleitet. Anschließend muss immer eine schließende Klammer `)` kommen.
Das Nichtterminal `N` wird zu einer Ziffer von `0` bis `9`; das Nichtterminal `Z` zu einer Ziffer zwischen `1` bis `9`, die auch entfallen kann (`ε`-Produktion). Mit Hilfe des Nichtterminals `Z` kann die erste Stelle eines zweistelligen Frequenzeintrags realisiert werden.
Das Nichtterminal `B` kann in einen Buchstaben zwischen `a` und `f` abgeleitet werden. Dieses Nichtterminal ist erforderlich, da Frequenzstärken auch aus Buchstaben bestehen können.

Für eine reguläre Grammatik gilt, dass auf der rechten Seite aller Produktionen entweder ein einzelnes Terminal oder – im Falle einer rechtslinearen Grammatik – ein Terminal gefolgt von einem einzelnen Nichtterminal steht. Auch `ε`-Produktion sind erlaubt. Die Grammatik G_1 enthält aber z. B. die Produktion `S → F:I`, die nicht diesen Anforderungen genügt. Die Grammatik ist daher nicht regulär.

Die Sprache zur Grammatik G_1 ist regulär. Da nur Signalfrequenzen zwischen `0.00` und `100.00` und Signalstärken zwischen `0` und `f` vorkommen können, gibt es nur endliche viele Möglichkeiten für einen Signaleintrag. Daher muss es eine reguläre Grammatik geben, die alle möglichen Signaleinträge erzeugt. Diese Signaleinträge werden nun mit dem Zeichen `;` getrennt aneinandergereiht. Da diese Aneinanderreihung in Einzelschritten aufgebaut werden kann, ist auch das mit einer regulären Grammatik möglich.

Teilaufgabe d)

Zuerst wird eine leere Liste für das Ergebnis der Methode initialisiert. Anschließend wird das übergebene Signalprotokoll mit Hilfe einer Schleife durchlaufen.
Für jeden Eintrag wird die Signalfrequenz und die Signalstärke ermittelt. Ist die Frequenz zwischen einschließlich `1.42` und einschließlich `1.66` und darüber hinaus die Signalstärke eine `1` oder eine `2`, so wird der Eintrag in die Ergebnisliste übernommen.
Nachdem alle Einträge durchlaufen wurden, wird die Ergebnisliste zurückgeliefert.

```
public List<Signal> ermittleMoeglicheKontakte(
                                              List<Signal> pProtokoll){
  List<Signal> ergebnis = new List<Signal>();
  pProtokoll.toFirst();
  while (pProtokoll.hasAccess()) {
    double freq = pProtokoll.getContent().gibFrequenz();
    char st = pProtokoll.getContent().gibStaerke();
    if ((1.42 <= freq && freq <= 1.66) &&
        (st == '1' || st == '2')) {
      ergebnis.append(pProtokoll.getContent());
    }
    pProtokoll.next();
  }
  return ergebnis;
}
```

Teilaufgabe e)

Ein Kellerautomat kann diese Aufgabe lösen, indem er vor dem Auftauchen des Zeichens `!` für jeden Eintrag `0.00` ein vorher definiertes Zeichen in den Keller speichert. Sobald das Zeichen `!` gelesen wird, beginnt der Automat für jedes Auftauchen des Eintrags `0.00` ein Zeichen aus dem Keller zu entfernen. Ist der Keller leer, ohne dass die Eingabe vollständig abgearbeitet ist, muss überprüft werden, ob die Zeichenfolge `0.00` noch mindestens einmal vorkommt. Ist das der Fall, ist der Standort in Nordafrika „ruhiger“.

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 Startzustand, Endzustände, Eingabealphabet und Zustandsmenge des Automaten A_1 an.	3			
2	analysiert den Automaten A_1 und erläutert, wie eine Eingabezeichenfolge aufgebaut sein muss, um als fehlerfrei akzeptiert zu werden.	3			
3	ermittelt eine Zeichenfolge, die vom Automaten A_1 akzeptiert wird, obwohl sie kein korrektes Signalprotokoll sein kann.	3			
Sachlich richtige Lösungsalternative zur Modelllösung: (9) …… ……					
	Summe Teilaufgabe a)	**9**			

Teilaufgabe b)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	entwirft den nichtdeterministischen Automaten A_2.	7			
2	begründet, dass es sich beim Automaten A_2 um einen nichtdeterministischen endlichen Automaten handelt.	3			
3	zeigt, dass der Automat A_2 das Signalprotokoll akzeptiert.	4			
Sachlich richtige Lösungsalternative zur Modelllösung: (14) …… ……					
	Summe Teilaufgabe b)	**14**			

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

Teilaufgabe c)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	erläutert die Produktionen der Grammatik G_1.	5			
2	begründet, dass die Grammatik G_1 nicht regulär ist.	3			
3	begründet, dass die Sprache der Grammatik G_1 regulär ist.	4			
Sachlich richtige Lösungsalternative zur Modelllösung: (12) …… ……					
	Summe Teilaufgabe c)	**12**			

Teilaufgabe d)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	entwickelt für die Methode ein algorithmisches Verfahren.	4			
2	implementiert die Methode entsprechend dem Verfahren.	5			
Sachlich richtige Lösungsalternative zur Modelllösung: (9) …… ……					
	Summe Teilaufgabe d)	**9**			

Teilaufgabe e)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl**	**EK**	**ZK**	**DK**
1	beurteilt, ob ein Kellerautomat durch Überprüfung des kombinierten Protokolls entscheiden kann, ob der Standort in Nordafrika ruhiger ist.	6			
Sachlich richtige Lösungsalternative zur Modelllösung: (6) …… ……					
	Summe Teilaufgabe e)	**6**			

	Summe insgesamt	**50**			

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

	Lösungsqualität			
	maximal erreichbare Punktzahl	EK	ZK	DK
Übertrag der Punktsumme aus der ersten bearbeiteten Aufgabe	**50**			
Übertrag der Punktsumme aus der zweiten bearbeiteten Aufgabe	**50**			
Übertrag der Punktsumme aus der dritten bearbeiteten Aufgabe	**50**			
Punktzahl der gesamten Prüfungsleistung	**150**			
aus der Punktsumme resultierende Note gemäß nachfolgender Tabelle				
Note ggf. unter Absenkung um bis zu zwei Notenpunkte gemäß § 13 Abs. 2 APO-GOSt				
Paraphe				

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

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

Unterschrift, Datum:

Grundsätze für die Bewertung (Notenfindung)

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

Note	**Punkte**	**Erreichte Punktzahl**
sehr gut plus	15	150 – 143
sehr gut	14	142 – 135
sehr gut minus	13	134 – 128
gut plus	12	127 – 120
gut	11	119 – 113
gut minus	10	112 – 105
befriedigend plus	9	104 – 98
befriedigend	8	97 – 90
befriedigend minus	7	89 – 83
ausreichend plus	6	82 – 75
ausreichend	5	74 – 68
ausreichend minus	4	67 – 60
mangelhaft plus	3	59 – 50
mangelhaft	2	49 – 41
mangelhaft minus	1	40 – 30
ungenügend	0	29 – 0