



Name: _____

Abiturprüfung 2023

Informatik, Leistungskurs

Aufgabenstellung:

Um bei Sportturnieren den unterschiedlichen Leistungsstärken der Teams gerecht zu werden, sollen möglichst Teams gegeneinander spielen, die ungefähr gleich stark sind. Dazu werden die Teams nach jeder Runde in einer Reihenfolge angeordnet, die dem Erfolg des Teams im Turnier entspricht. Anhand der Reihenfolgen werden die Spiele nach folgendem Prinzip ermittelt:

- Bei Beginn des Turniers werden alle Teams in zufälliger Reihenfolge angeordnet.
- Nach jeder Partie wird die Reihenfolge der Teams anhand ihrer Punktzahlen angepasst: Die Reihenfolge der Teams ist absteigend nach deren Punktzahl geordnet. Für einen Sieg erhält ein Team 1 Punkt, für eine Niederlage -1 Punkt. Bei einem Unentschieden erhalten beide Teams 0 Punkte.
- Um für ein Team A einen Gegner zu bestimmen, wird das Team B als Gegner ausgewählt, das in der Reihenfolge möglichst nah beim Team A ist.
- Um zu vermeiden, dass immer wieder die gleichen Teams gegeneinander spielen, wird eine Maximalzahl gleicher Begegnungen festgelegt: Zwei Teams sollen höchstens so häufig wie in dieser Zahl festgelegt gegeneinander spielen.

Folgendes Softwaremodell ist ein erster Entwurf für die Verwaltung eines solchen Turniers:

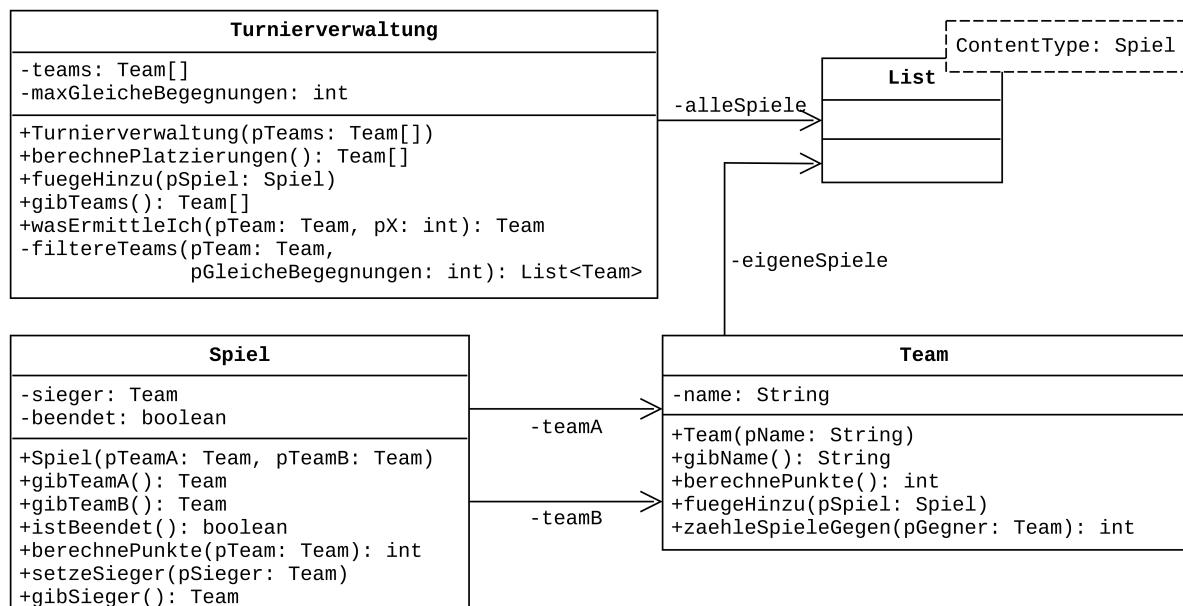


Abbildung 1: Teilmodellierung des Turniers

Abiturprüfung 2023 – Nur für den Dienstgebrauch!



Name: _____

- a) Analysieren Sie das Implementationsdiagramm und erläutern Sie die dargestellten Assoziationen im Sachkontext.

Erläutern Sie anhand des Implementationsdiagramms und der Dokumentation im Anhang, wie laut dem Modell ein Spiel hinzugefügt und wie der Sieger eingetragen wird.

(4 + 4 Punkte)

- b) Die private Methode `filtereTeams` der Klasse `Turnierverwaltung` filtert die Teams heraus, die höchstens eine bestimmte Anzahl von Spielen gegen das übergebene Team gespielt haben. Das als Parameter übergebene Team ist nicht im Ergebnis der Methode enthalten.

Wird für `pTeam` `null` übergeben oder ist der Wert von `pGleicheBegegnungen` kleiner als 0, so wird eine leere Liste zurückgegeben. Die Methode hat den folgenden Methodenkopf:

```
private List<Team> filtereTeams(Team pTeam,  
                                int pGleicheBegegnungen)
```

Entwickeln Sie einen Algorithmus, mit dem das Filtern entsprechend der oben beschriebenen Vorgabe durchgeführt werden kann.

Implementieren Sie die Methode.

(4 + 6 Punkte)



Name: _____

- c) Der Quellcode des Projekts enthält in der Klasse Turnierverwaltung folgende nicht dokumentierte Methode:

```
1 public Team wasErmittleIch(Team pTeam, int pX) {  
2     List<Team> auswahl  
3         = filtereTeams(pTeam, pX);  
4     auswahl.toFirst();  
5     int punkte = pTeam.berechnePunkte();  
6     Team g = auswahl.getContent();  
7     int d = Math.abs(g.berechnePunkte() - punkte);  
8     auswahl.next();  
9     while (auswahl.hasAccess()) {  
10        Team temp = auswahl.getContent();  
11        int diff = Math.abs(temp.berechnePunkte()  
12                      - punkte);  
13        if (diff < d) {  
14            d = diff;  
15            g = temp;  
16        }  
17        auswahl.next();  
18    }  
19    return g;  
20 }
```

Hinweis: Die Methode Math.abs(**int** a) liefert den Absolutbetrag einer Integerzahl a. Ist beispielsweise a = -5, also negativ, so ist die Rückgabe 5. Ist a = 3, also positiv, ist die Rückgabe 3.

Die Methode soll mithilfe der Beispieldaten in der Anlage (Wert des Attributs pX = 0 und Team SF Forst) analysiert werden.

Analysieren Sie die Methode wasErmittleIch, indem Sie sie auf die Beispieldaten anwenden und die Rückgabe angeben.

Erläutern Sie die zugrundeliegende Strategie des Algorithmus.

Erläutern Sie, was die Methode im Sachkontext ermittelt.

Analysieren und erläutern Sie, an welcher Stelle des Quellcodes ein Laufzeitfehler durch einen Zugriff auf ein nicht existentes Objekt (eine sogenannte NullPointerException) entstehen kann.

(4 + 3 + 2 + 4 Punkte)



Name: _____

d) Um die Software unter anderem für verschiedene Sportarten einsetzen zu können, soll die Software flexibler gestaltet werden. Deswegen soll das Modell um folgende Punkte verändert werden:

1. In verschiedenen Sportarten werden Sieg, Unentschieden und Niederlage unterschiedlich bepunktet. Für jedes Turnier sollen diese Punkte zu Turnierbeginn festgelegt werden können.
2. Für jedes Team soll jeweils angefragt werden können, gegen welche Teams es bereits gespielt hat.

Erweitern Sie das Modell aus Abbildung 1 um die angegebenen Erweiterungen.

Erläutern Sie, wie Ihr Modell die Anforderungen umsetzt.

Hinweis: Stellen Sie nur die Änderungen im Modell im Vergleich zu Abbildung 1 dar.

(6 + 5 Punkte)



Name: _____

- e) Bei der Implementierung der Software entsteht die Frage, wie die Spiele für die nächste Runde von der Software ermittelt werden sollen. Zur Vereinfachung soll die Anzahl der bereits gegeneinander gespielten Spiele zweier Mannschaften hier nicht berücksichtigt werden. Außerdem wird von einer geraden Anzahl von Teams ausgegangen.

Die Software soll dafür sorgen, dass für jedes Spiel zu erwarten ist, dass es möglichst ausgeglichen sein wird. Dazu sollen Teams gegeneinander spielen, die ähnlich erfolgreich im Turnier sind: Wenn zwei Teams gegeneinander spielen sollen, soll die Differenz ihrer Punkte möglichst gering sein.

Dabei gibt es zwei Vorschläge:

- (i) Beginne in der Mitte der Rangfolge:

- Wähle das erste der beiden mittleren Teams.
- Dieses Team ist in der Reihenfolge von zwei Teams umgeben: Wähle von den beiden Teams das als Gegner aus, zu dem die Punktdifferenz geringer ist.
- Ist die Punktdifferenz gleich, wähle das untere Team als Gegner.
- Entferne die beiden gewählten Teams aus der Rangfolge.
- Wiederhole diesen Prozess, bis nur noch zwei Teams ungeplant sind.
- Sind nur noch zwei Teams ungeplant, lasse diese beiden gegeneinander spielen.

- (ii) Beginne bei der aktuellen Rangfolge beim ersten Team:

- Wähle das erste Team gegen das zweite Team als Spiel.
- Entferne beide Teams aus der Rangfolge.
- Wiederhole diesen Prozess, bis die Rangfolge leer ist.

Ermitteln Sie für beide Vorschläge die Spiele und die Differenz der Punkte für jedes Spiel anhand der vorliegenden Beispieldaten im Anhang.

Beurteilen Sie welcher Ansatz zu besseren Spielpaarungen führt.

(4 + 4 Punkte)

Zugelassene Hilfsmittel:

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



Name: _____

Anlage: Beispieldaten

Teams (in dieser Reihenfolge im Feld teams):

SF Forst
SG Heide
TV Feld
SSV Bergen
TuS Wiese
SV Wald

Spiele:

Runde 1:

SF Forst - SG Heide: Sieger ist SF Forst
TV Feld - SSV Bergen: Sieger ist TV Feld
TuS Wiese - SV Wald: Unentschieden

Runde 2:

SF Forst - TV Feld: Unentschieden
TuS Wiese - SG Heide: Sieger ist TuS Wiese
SV Wald - SSV Bergen: Unentschieden

Aktueller Punktestand:

Platzierung	Team	Punkte
1.	SF Forst	1
2.	TV Feld	1
3.	TuS Wiese	1
4.	SV Wald	0
5.	SSV Bergen	-1
6.	SG Heide	-2

Ergebnis von **filtere(pTeam, px)**, wobei es sich bei dem Team um SF Forst handelt und px den Wert 0 besitzt:

SSV Bergen
TuS Wiese
SV Wald



Name: _____

Anhang

Dokumentationen der verwendeten Klassen

Die Klasse Team

Objekte der Klasse Team repräsentieren die zu einem Team vorhandenen Daten. Zu jedem Team in einem Turnier soll immer nur genau ein Objekt existieren. Ein Objekt der Klasse verwaltet Referenzen auf die Spiel-Objekte, an denen das Team beteiligt ist.

Dokumentation der Klasse Team

Team(String pName)

Ein Objekt der Klasse wird initialisiert. Die übergebene Zeichenkette wird gespeichert.
Das Team hat keine eigenen Spiele gespeichert.

String gibName()

Die Methode liefert den Namen des Teams zurück.

int berechnePunkte()

Die Methode liefert die Punkte des Teams auf Basis der eigenen Spiele, die bereits beendet sind. Die Rückgabe entspricht der Summe der Einzelbewertungen jedes beendeten Spiels für das Team. Wurde noch kein eigenes Spiel beendet, wird 0 zurückgegeben.

void fuegeHinzu(Spiel pSpiel)

Das durch den Parameter referenzierte Objekt der Klasse Spiel wird zu den eigenen Spielen hinzugefügt.

int zaehleSpieleGegen(Team pGegner)

Die Methode liefert die Anzahl der gespielten Spiele, die das als Parameter übergebene Team gegen dieses Team gespielt hat. Wird null für den Parameter übergeben, so wird -1 zurückgegeben.



Name: _____

Die Klasse Turnierverwaltung

Objekte der Klasse Turnierverwaltung verwaltet alle Teams und alle Spiele eines Turniers. Die Teams werden in der Reihenfolge verwaltet, in der sie sich für das Turnier angemeldet haben.

Dokumentation der Klasse Turnierverwaltung

Turnierverwaltung (Team[] pTeams)

Ein Objekt der Klasse wird initialisiert. Die im Feld übergebenen Teams werden in einer zufälligen Reihenfolge angeordnet. Es existieren keine Spiele.

Team[] berechnePlatzierungen()

Die Methode liefert alle im Turnier vorhandenen Teams in einem Feld. Das Feld ist absteigend nach der Punktzahl der Teams sortiert. Sind zwei Teams punktgleich, ist die Reihenfolge zufällig.

void fuegeHinzu (Spiel pSpiel)

Das übergebene Spiel wird der Liste aller Spiele hinzugefügt. Außerdem wird beiden Teams das übergebene Spiel hinzugefügt. Dabei wird davon ausgegangen, dass die beiden im Spiel referenzierten Objekte der Klasse Team nicht identisch sind und von dem Objekt der Klasse Turnierverwaltung referenziert werden.

Team[] gibTeams()

Die Methode liefert die Teams in der Reihenfolge, in der sie sich zum Turnier angemeldet haben.

Über die öffentlichen Methoden hinaus verfügt die Klasse über folgende private Methode:
Dokumentation dieser privaten Methode der Klasse:

List<Team> filtereTeams (Team pTeam, int pGleicheBegegnungen)

Es wird eine Liste zurückgegeben, die alle Teams enthält, die bisher in dem Turnier höchstens den als den Wert von pGleicheBegegnungen mal gegen das übergebene Team gespielt haben. Referenziert der Parameter pTeam null oder ist der Wert des Parameters pGleicheBegegnungen kleiner als 0, so wird eine leere Liste zurückgegeben.



Name: _____

Die Klasse Spiel

Ein Objekt der Klasse Spiel verwaltet die beteiligten Teams, ob das Spiel bereits beendet wurde, und das siegende Team des Spiels, sofern es eins gibt.

Dokumentation der Klasse Spiel

Spiel(Team pTeamA, Team pTeamB)

Ein Objekt der Klasse wird initialisiert. Die beiden übergebenen Teams werden in dem Spiel gespeichert. Das Spiel gilt als nicht beendet. Es ist kein Sieger eingetragen.

Team gibTeamA()

Die Methode liefert das Team A des Spiels.

Team gibTeamB()

Die Methode liefert das Team B des Spiels.

boolean istBeendet()

Ist das Spiel beendet, wird true zurückgegeben, andernfalls wird false zurückgegeben.

int berechnePunkte(Team pTeam)

Ist das Spiel nicht beendet oder ist kein Sieger gespeichert (das Spiel also als Unentschieden gewertet), so wird der Wert 0 zurückgegeben. Ist das im Parameter referenzierte Team nicht an dem Spiel beteiligt, wird der Wert -1 zurückgegeben.

void setzeSieger(Team pSieger)

Das Spiel wird als beendet gewertet. Das übergebene Team wird als Sieger gesetzt. Wird null übergeben, gilt das Spiel als unentschieden. Es wird davon ausgegangen, dass entweder null oder ein beteiligtes Team übergeben wird.

Team gibSieger()

Die Methode liefert den Sieger des Spiels. Gibt es keinen Sieger, weil das Spiel unentschieden oder noch nicht beendet ist, wird null geliefert.



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 2023

Informatik, Leistungskurs

1. Aufgabenart

Analyse, Modellierung und Implementation kontextbezogener Problemstellungen mit Schwerpunkt auf den Inhaltenfeldern Daten sowie ihre Strukturierung und Algorithmen

2. Aufgabenstellung¹

siehe Prüfungsaufgabe

3. Materialgrundlage

entfällt

4. Bezüge zum Kernlehrplan und zu den Vorgaben 2023

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

1. Inhaltenfelder und inhaltliche Schwerpunkte

Daten und ihre Strukturierung

- Objekte und Klassen
 - Entwurfsdiagramme und Implementationsdiagramme
 - Lineare Strukturen
 - Array bis zweidimensional
 - 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)

Im Diagramm sind die vier Assoziationen alleSpiele, eigeneSpiele, teamA und teamB dargestellt.

Die Assoziation alleSpiele modelliert den Sachverhalt, dass ein Objekt der Klasse Turnierverwaltung alle Spiele des Turniers in einer Liste (Datentyp List mit Inhaltstyp Spiel) verwaltet – unabhängig davon, ob ein Spiel beendet ist oder nicht.

Die Assoziationen teamA und teamB modellieren die Beteiligung der beiden Teams an einem bestimmten Spiel. Ein Objekt der Klasse Spiel verwaltet also über diese Referenzen zwei Objekte vom Typ Team.

Die Assoziation eigeneSpiele modelliert den Sachverhalt, dass ein Objekt der Klasse Team nur die Spiele verwaltet, an denen das Team beteiligt ist – entweder als Team A oder als Team B.

Ein Spiel wird hinzugefügt, indem zunächst mit zwei Referenzen auf zwei Teams des Turniers ein Objekt der Klasse Spiel erzeugt wird. Dieses Objekt wird der Liste des Objekts der Klasse Turnierverwaltung mittels der Methode fuegeHinzu hinzugefügt. Außerdem wird über die Methode fuegeHinzu der Klasse Team beiden Objekten mit den Referenzen teamA und teamB das Spiel zu der Liste der eigenen Spiele mit dem Bezeichner eigeneSpiele hinzugefügt.

Das siegende Team eines Spiels wird über den Parameter der Methode setzeSieger des Objekts der Klasse Spiel übergeben. Ist das Ergebnis eines Spiels ein Unentschieden, so wird null übergeben.

Teilaufgabe b)

Beim Filtern sollten zunächst die Sonderfälle betrachtet werden, dass die Parameter ungültige Werte enthalten. Tritt einer der beiden genannten Sonderfälle ein, wird null zurückgegeben, andernfalls kann das Filtern beginnen.

Für das Ermitteln der geeigneten Teams wird das Feld vollständig durchlaufen und für jedes Element geprüft, ob das aktuelle Team nicht dem übergebenen Team entspricht und die genannte Voraussetzung der Spielanzahl erfüllt. Ist dies der Fall, wird es an die später zurückgegebene Ergebnisliste angehängt.

```

private List<Team> filtereTeams(Team pTeam,
                                int pGleicheBegegnungen) {
    List<Team> ergebnis = new List<>();
    if (pTeam != null
        && pGleicheBegegnungen >= 0) {
        for (int i = 0; i < teams.length; i++) {
            Team gegner = teams[i];
            if (pTeam != gegner
                && pTeam.zahleSpieleGegen(gegner)
                < pGleicheBegegnungen) {
                ergebnis.append(gegner);
            }
        }
    }
    return ergebnis;
}

```

Teilaufgabe c)

Es wird ein Objekt der Klasse Team mit dem Namen TuS Wiese zurückgegeben.

Die Methode ermittelt durch Aufruf der privaten Methode `filtere` alle Teams, die noch nicht gegen das übergebene Team SF Forst gespielt haben (`pX = 0`).

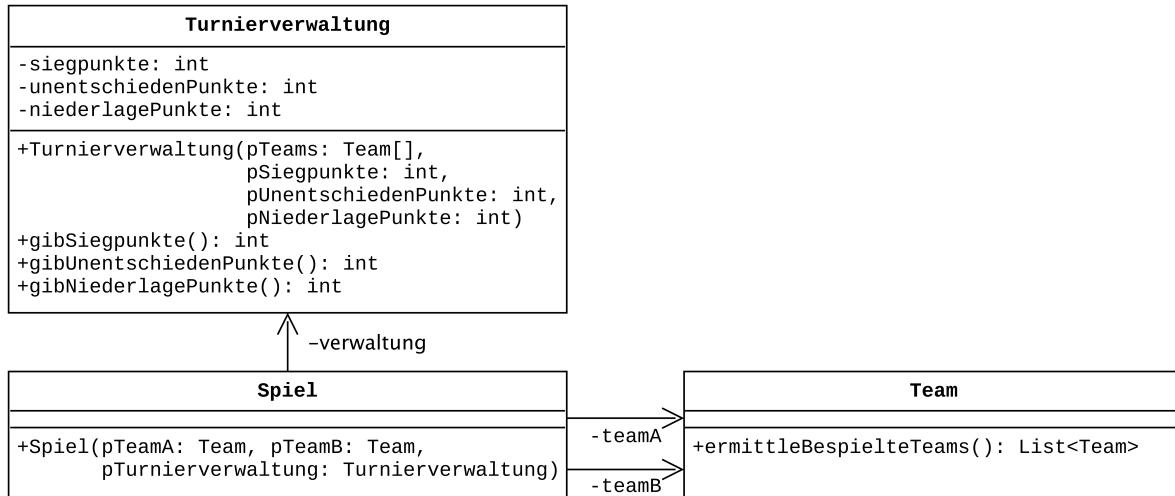
Sofern die Liste nicht leer ist, wird mittels eines Durchlaufs durch die Liste das Minimum des Punkteabstands zum übergebenen Team gesucht. Dazu wird zunächst das erste Element als das mit geringstem Abstand angenommen, bevor die Liste der möglichen Gegner durchlaufen wird und das Team mit der kleinsten absoluten Punktdifferenz ermittelt wird.

Die Methode ermittelt eins der Teams, die die günstigsten Gegner für das übergebene Team sind, deren Punktabstand zum übergebenen Team also am geringsten ist.

Eine Nullpointer-Exception kann in folgendem Fall auftreten:

In Zeile 7, wenn die Methode `filtere` eine leere Liste liefert, referenziert `g` (nach der Zuweisung in Zeile 6) `null`. Damit kann der Methodenaufruf von `berechnePunkte` in Zeile 7 nicht ausgeführt werden.

Teilaufgabe d)



1. Für ein Turnier kann beim Erzeugen eines Objekts der Klasse `Turnierverwaltung` über die entsprechenden Parameter der Wert der Attribute `siegpunkte`, `unentschiedenPunkte` und `niederlagePunkte` einmalig festgelegt werden und über die entsprechenden Methoden `gibSiegpunkte`, `gibUnentschiedenPunkte` und `gibNiederlagePunkte` angefragt werden. Da die Punktauswertung für die Teams in der Klasse `Spiel` vorgenommen wird, wird beim Erzeugen eines Objekts der Klasse `Spiel` eine Referenz auf die `Turnierverwaltung` übergeben.
2. Da jedes Objekt der Klasse `Team` die eigenen Spiele referenziert, müssen keine weiteren Daten gespeichert werden. Über die Methode `ermittleBespielteTeams` kann für jedes Objekt der Klasse `Team` ermittelt werden, gegen welche Teams bereits gespielt wurde.

Teilaufgabe e)

Vorschlag 1:

TuS Wiese – TV Feld	0
SV Wald – SSV Bergen	1
SF Forst – SG Heide	3

Vorschlag 2:

SF Forst – TV Feld	0
TuS Wiese – SV Wald	1
SSV Bergen – SG Heide	1

Beim ersten Vorschlag besteht im Gegensatz zum zweiten Vorschlag das Risiko, dass das Team mit den meisten Punkten gegen das Team mit den wenigsten Punkten spielt. Ein solches Spiel steht diametral gegen die Forderung, dass ungefähr gleich erfolgreiche Teams gegeneinander spielen.

Beim zweiten Vorschlag ist die Auswahl für jedes einzelne Team möglicherweise nicht optimal, es handelt sich jedoch stets um hinreichend ausgewogene Spiele, da einerseits eine große Punktdifferenz zwischen zwei benachbarten Teams nicht zu erwarten ist. Sollte andererseits doch eine große Punktdifferenz existieren, muss sie auf jeden Fall aufgelöst werden. Für den ersten Ansatz können dies jedoch Spiele mit einer größeren Punktdifferenz sein als beim zweiten Ansatz.

Damit ist der zweite Vorschlag als geeigneter zu bewerten.

7. Teilleistungen – Kriterien / Bewertungsbogen zur Prüfungsarbeit

Name des Prüflings: _____ Kursbezeichnung: _____

Schule: _____

Teilaufgabe a)

	Anforderungen	Lösungsqualität			
		maximal erreichbare Punktzahl	EK ²	ZK	DK
1	analysiert das Implementationsdiagramm und erläutert die dargestellten Assoziationen im Sachkontext.	4			
2	erläutert anhand des Implementationsdiagramms und der Dokumentation im Anhang, wie ein Spiel laut dem Modell hinzugefügt und der Sieger eingetragen wird.	4			
Sachlich richtige Lösungsalternative zur Modelllösung: (8)					
.....					
	Summe Teilaufgabe a)	8			

Teilaufgabe b)

	Anforderungen	Lösungsqualität			
		maximal erreichbare Punktzahl	EK	ZK	DK
1	entwickelt einen Algorithmus, mit dem das Filtern entsprechend der beschriebenen Vorgabe durchgeführt werden kann.	4			
2	implementiert die Methode.	6			
Sachlich richtige Lösungsalternative zur Modelllösung: (10)					
.....					
	Summe Teilaufgabe b)	10			

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

Teilaufgabe c)

	Anforderungen	Lösungsqualität				
		Der Prüfling	maximal erreichbare Punktzahl	EK	ZK	DK
1	analysiert die Methode, indem er sie auf die Beispieldaten anwendet, und gibt die Rückgabe an.	4				
2	erläutert die zugrundeliegende Strategie des Algorithmus.	3				
3	erläutert, was die Methode im Sachkontext ermittelt.	2				
4	analysiert und erläutert, an welchen Stellen des Quellcodes ein Laufzeitfehler durch einen Zugriff auf ein nicht existentes Objekt entstehen kann.	4				
Sachlich richtige Lösungsalternative zur Modelllösung: (13)						
.....						
.....						
Summe Teilaufgabe c)		13				

Teilaufgabe d)

	Anforderungen	Lösungsqualität				
		Der Prüfling	maximal erreichbare Punktzahl	EK	ZK	DK
1	erweitert das Modell aus Abbildung 1 um die angegebenen Erweiterungen.	6				
2	erläutert, wie das Modell die Anforderungen umsetzt.	5				
Sachlich richtige Lösungsalternative zur Modelllösung: (11)						
.....						
.....						
Summe Teilaufgabe d)		11				

Teilaufgabe e)

	Anforderungen	Lösungsqualität			
		maximal erreichbare Punktzahl	EK	ZK	DK
1	ermittelt für beide Vorschläge die Spiele und die Differenz der Punkte für jedes Spiel anhand der vorliegenden Beispieldaten im Anhang.	4			
2	beurteilt welcher Ansatz zu besseren Spielpaarungen führt. Sachlich richtige Lösungsalternative zur Modelllösung: (8)	4			
	Summe Teilaufgabe e)	8			
	Summe insgesamt	50			

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

	Lösungsqualität			
	maximal erreichbare Punktzahl	EK	ZK	DK
Übertrag der Punktsumme aus der ersten bearbeiteten Aufgabe	50			
Übertrag der Punktsumme aus der zweiten bearbeiteten Aufgabe	50			
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 2023

Informatik, Leistungskurs

Aufgabenstellung:

Beim Spiel *NobodyKnows* geht es darum, sich für unbekannte Begriffe bzw. Wissensfragen so kreative und glaubwürdig klingende Erklärungen auszudenken, dass die anderen Mitspielerinnen und Mitspieler darauf hereinfallen.

Jede laufende Spielrunde gliedert sich in drei Phasen:

1. Frage-/Antwortphase:

Zu Beginn wird allen Spielenden dieselbe Wissensfrage gestellt, die normalerweise niemand beantworten kann. Nur die (nicht mitspielende) Spielleitung kennt die korrekte Antwort. Die Mitspielerinnen und Mitspieler müssen sich nun eine möglichst kreative Phantasieantwort einfallen lassen. Diese notieren sie und geben sie verdeckt bei der Spielleitung ab.

2. Tipp-Phase:

Sind alle Antworten eingereicht, werden diese, zusammen mit der korrekten Antwort, in zufälliger Reihenfolge allen Spielenden präsentiert. Jeder Mitspieler bzw. jede Mitspielerin muss sich nun für eine der präsentierten Antworten entscheiden und auf diese Weise einen entsprechenden Tipp abgeben.

3. Auswertungsphase:

Hat man auf die korrekte Antwort getippt, bekommt man 2 Punkte. Ist ein anderer Mitspieler oder eine andere Mitspielerin auf die eigene Antwort hereingefallen, indem er oder sie auf diese getippt hat, bekommt man zusätzlich jeweils 3 Punkte.

Fans des Spiels finden es schade, dass in jeder Spielrunde jemand nicht mitspielen kann, weil er oder sie die Spielleitung übernehmen muss. Die Herstellerfirma möchte das Spiel daher nun auch als App für Smartphones und Tablets anbieten und die Rolle der Spielleitung einer Serversoftware übertragen. Die Serversoftware benutzt das Protokoll, das in Anlage 1 dokumentiert ist.

Während des Spiels sollen sich auf dem Bildschirm der Mitspielerinnen und Mitspieler Dialoge der in Abbildung 1 dargestellten Art entwickeln.



Name: _____

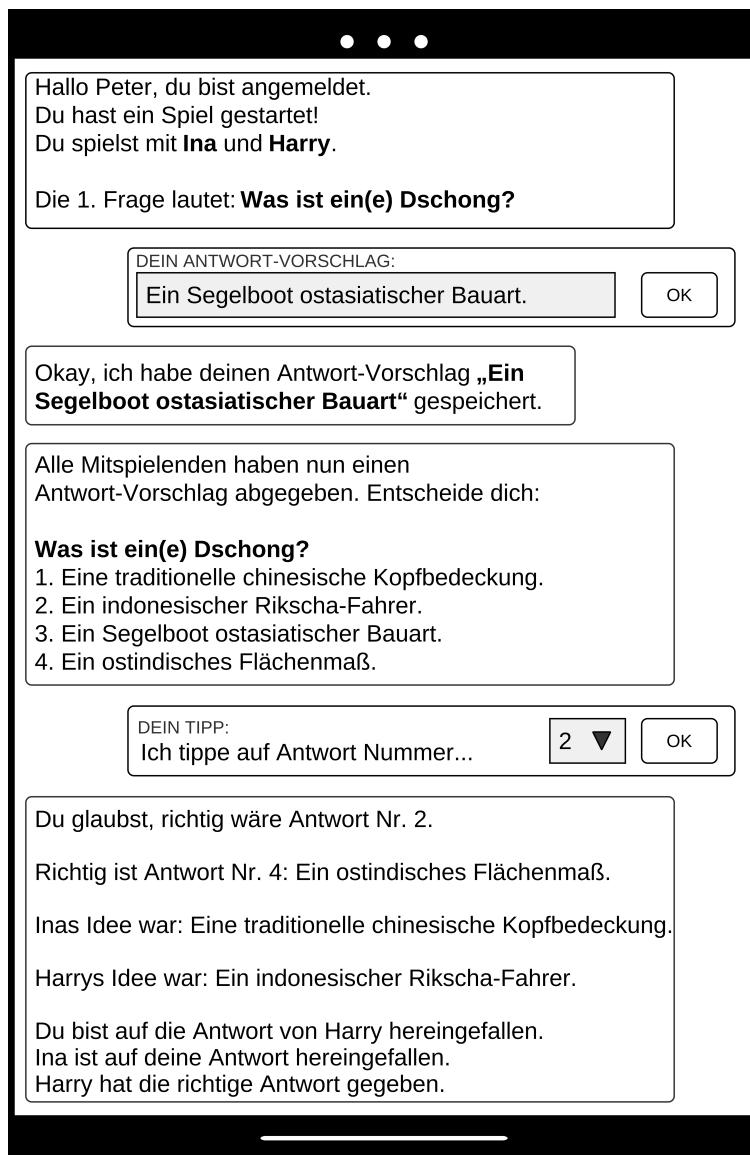


Abbildung 1: Ein Beispielhafter Spieldialog

- a) Der Dialog in Abbildung 1 stammt aus einem Testlauf der Tablet-App von Spieler Peter. Die Kommunikation zwischen Client und Server basiert auf dem Protokoll in Anlage 1.

Ermitteln Sie anhand des in Anlage 1 gegebenen Protokolls, welche Kommandos von Peters Client an den Server gesendet wurden und welche Nachrichten des Servers an Peters Client zurückgesendet wurden, sodass die Bildschirmausgabe in Abbildung 1 entstand.

Anmerkung: Machen Sie kenntlich, welche Servernachricht auf welches Client-Kommando folgt, indem Sie beispielsweise eine tabellarische Darstellung benutzen.

(8 Punkte)



Name: _____

Die Herstellerfirma hat eine erste Version der Serversoftware erstellt. Eine Teilmodellierung ist in Abbildung 2 dargestellt.

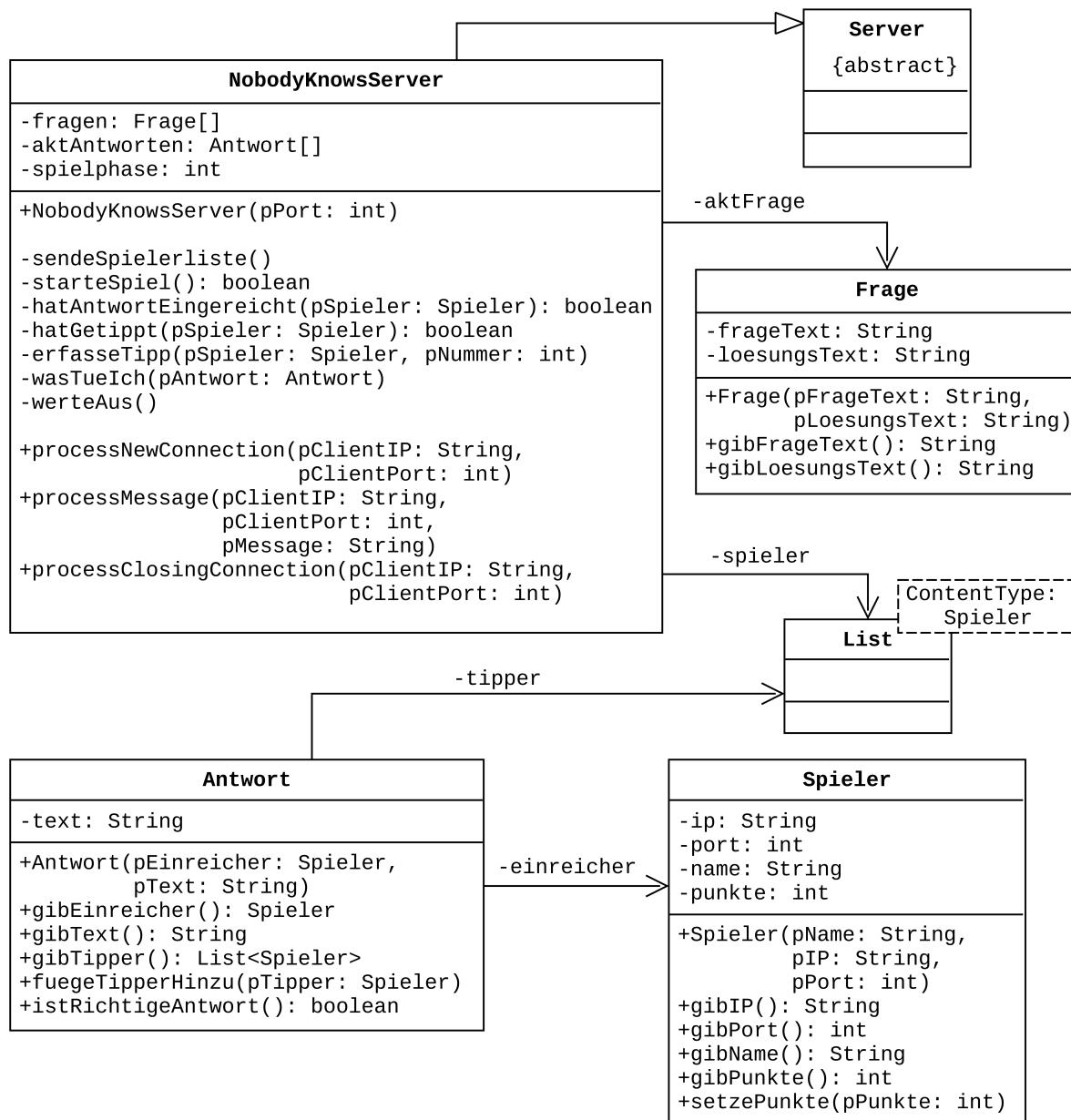


Abbildung 2: Teilmodellierung als Implementationsdiagramm



Name: _____

- b) Erläutern Sie die in Abbildung 2 gegebene Modellierung hinsichtlich der Beziehungen der Klassen NobodyKnowsServer, Server, Antwort, Frage und Spieler untereinander.

Erläutern Sie, weshalb die in der aktuellen Spielrunde gegebenen Antworten in einem Feld (Array) verwaltet werden können.

Erläutern Sie, warum es sinnvoll ist, dass Objekte der Klasse Spieler eine IP-Adresse und einen Port verwalten.

(6 + 3 + 3 Punkte)

- c) In der Klasse NobodyKnowsServer findet sich folgende undokumentierte Methode:

```
1 private boolean wasTueIch(Antwort pAntwort) {  
2     boolean e = false;  
3     boolean s = false;  
4     int pos = 0;  
5     while (!e && !s) {  
6         if (aktAntworten[pos] == null) {  
7             s = true;  
8         } else {  
9             if (aktAntworten[pos].gibEinreicher()  
10                == pAntwort.gibEinreicher()) {  
11                 e = true;  
12             }  
13             pos = pos + 1;  
14         }  
15     }  
16     if (s) {  
17         aktAntworten[pos] = pAntwort;  
18     }  
19     return s;  
20 }
```

Ein laufendes Spiel mit drei Spielern befindet sich in der Frage-/Antwortphase. Die Methode wird nun in drei verschiedenen Situationen aufgerufen:

- i. Niemand hat bisher eine Antwort eingereicht. Das Feld (Array) aktAntworten ist komplett leer, d. h., alle Feldeinträge verweisen auf null.
- ii. Genau eine Person hat bisher eine Antwort eingereicht. Das Feld (Array) aktAntworten verweist an Index 0 bereits auf ein Antwort-Objekt, dessen Attribut einreicher aber nicht auf dasselbe Spieler-Objekt verweist wie das entsprechende Attribut in pAntwort.
- iii. Mehrere Personen haben bereits eine Antwort eingereicht. Das Feld (Array) aktAntworten verweist an Index 0 bereits auf ein Antwort-Objekt, dessen Attribut einreicher auf dasselbe Spieler-Objekt verweist wie in pAntwort.



Name: _____

Analysieren und erläutern Sie die Funktionsweise dieser Methode unter Bezugnahme auf die genannten drei Situationen.

Erläutern Sie, welche Bedeutung die booleschen Variablen s und e im Sachkontext haben.

Erläutern Sie den Zweck der Methode im Sachkontext.

(6 + 4 + 2 Punkte)

- d) Die Methode `werteAus` der Klasse `NobodyKnowsServer` soll die Auswertung der aktuell laufenden Spielrunde vornehmen, d. h. sie vergibt die Punkte gemäß den von den Spielerinnen und Spielern in der aktuellen Spielrunde abgegebenen Tipps und veranlasst das Senden des `AUSWERTUNG`-Kommandos an die Clients. Die Methode hat den folgenden Methodenkopf:

private void werteAus()

Entwickeln Sie eine algorithmische Strategie für die Methode.

Implementieren Sie die Methode.

(4 + 8 Punkte)

- e) In der nächsten Version des Servers sollen dessen Möglichkeiten erweitert werden. Sehr viele Nutzerinnen und Nutzer wünschen sich die Möglichkeit, als angemeldete Spielerinnen und Spieler den Fragenkatalog des Servers mit neuen Fragen und dazugehörigem korrekten Lösungstext ergänzen zu können.

Entwickeln Sie eine entsprechende Erweiterung des Protokolls gemäß Anlage 1.

Nehmen Sie im Sachkontext Stellung zu der Frage, ob die Verwaltung des Fragenkatalogs weiterhin mit einer statischen Datenstruktur oder mit einer dynamischen Datenstruktur erfolgen sollte.

(2 + 4 Punkte)

Zugelassene Hilfsmittel:

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



Name: _____

Anlage 1: Protokoll der Client-Server-Kommunikation

Je nach Spielphase akzeptiert der Server jeweils nur bestimmte Teile des Protokolls. Kommandos, die nicht einer bestimmten Spielphase zugeordnet sind, kann der Server jederzeit verarbeiten. Empfängt der Server ein Kommando, das während der laufenden Spielphase nicht akzeptiert wird, sendet er eine Fehlermeldung der Form -ERR <Fehlercode>.

Client an Server	Server an Client	Kommentar
(Verbindungs-aufnahme)	+NKS OK	
(Fehler oder Fehleingabe)	-ERR <Text>	<Text> beschreibt den Fehler näher.
LOGOUT	+LOGOUT OK	Der Client wird aus der Liste der Spielenden gestrichen und ein evtl. mit ihm laufendes Spiel beendet.
SPIELER <Name>	+SPIELER OK	Der Name <Name> wird der Liste der möglichen Spielenden hinzugefügt. Sobald dies mindestens einmal erfolgreich ausgeführt wurde, wechselt der Server in Spielphase 0.
NEU	+NEU OK	Falls gerade ein Spiel läuft, wird dieses beendet. Alle Spielenden werden auf den Punktestand 0 zurückgesetzt.

Spielphase 0		
Client an Server	Server an Client	Kommentar
SPIELSTART	+SPIELSTART OK	Eine neue Spielrunde beginnt (ggf. wird dafür ein neues Spiel gestartet). Alle aktuell gespeicherten Spielerinnen und Spieler werden in dieses neue Spiel aufgenommen. Der Server wechselt in die Spielphase 1.

Spielphase 1		
Client an Server	Server an Client	Kommentar
	SPIELERLISTE <Liste>	Überträgt zu Beginn von Spielphase 1 die Liste der Spielenden. <Liste> beinhaltet dabei die Namen der Spielenden, gefolgt von ihrer Punktzahl in Klammern (z. B. Peter (5)). Die Inhalte der Liste werden durch das Zeichen voneinander getrennt.
	FRAGE <Text>	Sendet nach der Übermittlung der Spielerliste die Frage mit dem Fragetext <Text> für diese Spielrunde.
ANTWORT <Text>	+ANTWORT OK	<Text> wird als geheime Antwort des Clients gespeichert. Sobald alle Clients eine Antwort übermittelt haben, wechselt der Server in die Spielphase 2.



Name: _____

Spielphase 2		
Client an Server	Server an Client	Kommentar
	ANTWORTEN <Liste>	<p>Beim Einstieg in Spielphase 2 werden die Antworten in zufälliger Reihenfolge als <Liste> übertragen. Die Inhalte der Liste werden durch das Zeichen voneinander getrennt.</p> <p>Jeder Eintrag in der Liste hat die Form <Nummer>,<Name>:<Text>.</p> <p>Hierbei ist <Nummer> die ab 0 aufsteigende Nummer in der Liste, <Name> der Name, unter dem die die Antwort eingereicht wurde und <Text> der Antworttext. Bei der richtigen Antwort hat der Listeneintrag lediglich die Form <Nummer>:<Text>.</p>
TIPP <Nummer>	+TIPP OK oder -ERR <Text>	Der Client gibt mit <Nummer> an, auf welche Antwort er tippt. Versucht ein Client, ein zweites Mal einen Tipp abzugeben oder auf seine eigene Antwort zu tippen, wird eine Fehlermeldung <Text> zurückgegeben. Sobald alle Clients einen Tipp übermittelt haben, wechselt der Server in die Spielphase 3.

Spielphase 3		
Client an Server	Server an Client	Kommentar
	AUSWERTUNG <Reingefallene>@ <Richtige>	<p>Übermittelt zu Beginn von Spielphase 3 die Auswertung der laufenden Spielrunde.</p> <p><Reingefallene> listet die Namen der auf eine falsche Antwort Hereingefallenen auf.</p> <p><Richtige> listet die Namen derjenigen auf, die sich für die richtige Antwort entschieden hatten.</p> <p>In beiden Listen werden die Inhalte durch das Zeichen voneinander getrennt. Die beiden Listen werden durch das Trennzeichen @ voneinander getrennt.</p> <p>Nach der Auswertung wechselt der Server in die Spielphase 0.</p>



Name: _____

Anhang

Dokumentationen der verwendeten Klassen

Die Klasse NobodyKnowsServer

Ein Objekt dieser Klasse stellt einen Server für das Spiel dar. Er verwaltet die Mitspielenden undwickelt ein Spiel inklusive der sich dadurch ergebenden Kommunikation mit den Clients ab. Dabei wird das Protokoll gemäß Anlage 1 benutzt.

Ausschnitt aus der Dokumentation der Klasse NobodyKnowsServer

NobodyKnowsServer(int pPort)

Ein Server für das Spiel NobodyKnows wird initialisiert. Die Liste der Mitspielenden ist zu Beginn leer und es ist kein Spiel gestartet.

void sendeSpielerliste()

Eine Spielerliste wird gemäß dem Kommando SPIELERLISTE an alle Clients übertragen.

boolean starteSpiel()

Sind bereits Mitspielerinnen und Mitspieler vorhanden, wird ein neues Spiel bzw. eine neue Spielrunde gestartet. Dazu wird eine zufällige Frage geladen, in Spielphase 1 gewechselt und true zurückgegeben. Konnte kein Spiel bzw. keine Spielrunde gestartet werden, weil keine Spielerinnen oder Spieler registriert sind, wird false zurückgegeben und die Spielphase nicht verändert.

boolean hatAntwortEingereicht(Spieler pSpieler)

Es wird true zurückgegeben, wenn durch den Client, der durch pSpieler identifiziert wird, bereits eine Antwort für die in der laufenden Spielrunde gestellte Frage eingereicht wurde. Ist dies nicht der Fall oder ist pSpieler == null, wird false zurückgegeben.

boolean hatGetippt(Spieler pSpieler)

Es wird true zurückgegeben, wenn durch den Client, der durch pSpieler identifiziert wird, bereits ein Lösungs-Tipp für die in der laufenden Spielrunde vorgestellten Antworten abgegeben wurde. Ist dies nicht der Fall oder ist pSpieler == null, wird false zurückgegeben.

void erfasseTipp(Spieler pSpieler, int pNummer)

Die Methode speichert, dass der Client, der durch pSpieler identifiziert wird, in der laufenden Spielrunde auf den Antworttext mit der Nummer pNummer getippt hat. Hatte der Client bereits zuvor einen Tipp abgegeben, geschieht nichts. Wird am Ende festgestellt, dass alle Clients einen Tipp abgegeben haben, wechselt der Server in die Spielphase 3.

void wasTueIch(Antwort pAntwort)

Diese Methode wird im Rahmen von Aufgabenteil c) analysiert.



Name: _____

void werteAus()

Auf Basis der abgegebenen Tipps werden die Punktzahlen dieser Spielrunde ermittelt und den entsprechenden Spieler-Objekten angerechnet. Das Kommando AUSWERTUNG wird erstellt und an alle Clients gesendet. Anschließend wechselt der Server in die Spielphase 0.

void processNewConnection(String pClientIP, int pClientPort)

Eine neue Verbindung zu einem Client mit der angegebenen IP-Adresse-/Port-Kombination wird aufgenommen und initialisiert.

void processMessage(String pClientIP, int pClientPort, String pMessage)

Ein von einem Client übermitteltes Kommando pMessage wird interpretiert.

void processClosingConnection(String pClientIP, int pClientPort)

Die Verbindung zu dem Client mit der angegebenen IP-Adresse-/Port-Kombination wurde beendet. Das laufende Spiel wird dadurch beendet und das zugehörige Spieler-Objekt aus der Liste der Mitspielerinnen und Mitspieler gelöscht.

Die Klasse Antwort

Ein Objekt dieser Klasse repräsentiert eine der in jeder Fragerunde möglichen Antworten. Bei einer Spielrunde mit n Spielerinnen und Spielern gibt es $n+1$ mögliche Antworten, n von den Spielerinnen und Spielern und die korrekte Antwort. Zusätzlich verwaltet ein Objekt dieser Klasse auch die Menge derjenigen Spielerinnen und Spieler, die sich für diese Antwort entschieden haben, also auf sie getippt haben.

Ausschnitt aus der Dokumentation der Klasse Antwort

Antwort(Spieler pEinreicher, String pText)

Ein neues Antwort-Objekt wird erstellt. Als Einreicher bzw. Einreicherin des Textes wird das Objekt pEinreicher gespeichert, als dessen bzw. deren Antworttext pText. Hat pEinreicher den Wert null, stellt diese Antwort die korrekte Lösung zur Frage der aktuellen Spielrunde dar.

Spieler gibEinreicher()

Gibt dasjenige Spieler-Objekt zurück, durch das diese Antwort eingereicht wurde. Wird null zurückgegeben, dann handelt es sich bei der durch dieses Objekt referenzierten Antwort um die richtige Lösung zu der Frage der aktuellen Spielrunde.

String gibText()

Der Text dieser Antwort wird zurückgegeben.



Name: _____

List<Spieler> gibTipper()

Die Liste aller Spielerinnen und Spieler, die in der laufenden Spielrunde auf diese Antwort getippt haben, wird zurückgegeben.

void fuegeTipperHinzu(Spieler pTipper)

pTipper wird der Liste derjenigen Spielerinnen und Spieler hinzugefügt, die in der laufenden Spielrunde auf diese Antwort getippt haben.

boolean istRichtigeAntwort()

Gibt true zurück, wenn es sich bei diesem Antwort-Objekt um die richtige Antwort auf die Frage der aktuellen Spielrunde handelt, ansonsten false.

Die Klasse Frage

Ein Objekt dieser Klasse repräsentiert eine Frage (inklusive der korrekten Antwort darauf), wie sie zu Beginn jeder Spielrunde gestellt wird.

Ausschnitt aus der Dokumentation der Klasse Frage

Frage(String pFrageText, String pLoesungsText)

Ein neues Frage-Objekt wird mit dem angegebenen Fragetext und Lösungstext erstellt.

String gibFrageText()

Der Text dieser Frage wird zurückgegeben.

String gibLoesungsText()

Der Lösungstext zu dieser Frage wird zurückgegeben.



Name: _____

Die Klasse Spieler

Ein Objekt dieser Klasse repräsentiert einen Mitspieler oder eine Mitspielerin in der laufenden Spielrunde.

Ausschnitt aus der Dokumentation der Klasse Spieler

Spieler(String pName, String pIP, int pPort)

Ein neues Spieler-Objekt wird mit dem angegebenen Namen erstellt. Die Parameter pIP und pPort identifizieren den Client, von dem aus das Kommando SPIELER zum Erzeugen dieses neuen Objektes gesendet wurde.

String gibIP()

Die IP-Adresse des Clients, von dem die Erzeugung dieses Objekts veranlasst wurde, wird zurückgegeben.

int gibPort()

Die Port-Nummer des Clients, von dem die Erzeugung dieses Objekts veranlasst wurde, wird zurückgegeben.

String gibName()

Der Name des Spielers oder der Spielerin wird zurückgegeben.

int gibPunkte()

Die über alle Spielrunden insgesamt erreichten Punkte dieses Spielers bzw. dieser Spielerin werden zurückgegeben.

void setzePunkte(int pPunkte)

Die Punkte dieses Spielers bzw. dieser Spielerin werden auf den Wert pPunkte gesetzt.



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.



Name: _____

Die Klasse Server

Objekte von Unterklassen der abstrakten Klasse Server ermöglichen das Anbieten von Serverdiensten, sodass Clients Verbindungen zum Server mittels TCP/IP-Protokoll aufbauen können. Zur Vereinfachung finden Nachrichtenversand und -empfang zeilenweise statt, d. h., beim Senden einer Zeichenkette wird ein Zeilentrenner ergänzt und beim Empfang wird dieser entfernt. Verbindungsannahme, Nachrichtenempfang und Verbindungsende geschehen nebenläufig. Auf diese Ereignisse muss durch Überschreiben der entsprechenden Ereignisbehandlungsmethoden reagiert werden. Es findet nur eine rudimentäre Fehlerbehandlung statt, sodass z. B. Verbindungsabbrüche nicht zu einem Programmabbruch führen. Einmal unterbrochene oder getrennte Verbindungen können nicht reaktiviert werden.

Dokumentation der Klasse Server

Server(int pPort)

Ein Objekt vom Typ Server wird erstellt, das über die angegebene Portnummer einen Dienst anbietet. Clients können sich mit dem Server verbinden, sodass Daten (Zeichenketten) zu diesen gesendet und von diesen empfangen werden können. Kann der Server unter der angegebenen Portnummer keinen Dienst anbieten (z. B. weil die Portnummer bereits belegt ist), ist keine Verbindungsaufnahme zum Server und kein Datenaustausch möglich.

boolean isOpen()

Die Anfrage liefert den Wert `true`, wenn der Server auf Port `pPort` einen Dienst anbietet. Ansonsten liefert die Methode den Wert `false`.

boolean isConnectedTo(String pClientIP, int pClientPort)

Die Anfrage liefert den Wert `true`, wenn der Server mit dem durch `pClientIP` und `pClientPort` spezifizierten Client aktuell verbunden ist. Ansonsten liefert die Methode den Wert `false`.

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

Die Nachricht `pMessage` wird – um einen Zeilentrenner erweitert – an den durch `pClientIP` und `pClientPort` spezifizierten Client gesendet. Schlägt der Versand fehl, geschieht nichts.

void sendToAll(String pMessage)

Die Nachricht `pMessage` wird – um einen Zeilentrenner erweitert – an alle mit dem Server verbundenen Clients gesendet. Schlägt der Versand an einen Client fehl, wird dieser Client übersprungen.

void closeConnection(String pClientIP, int pClientPort)



Name: _____

Die Verbindung des Servers zu dem durch pClientIP und pClientPort spezifizierten Client wird getrennt. Zuvor wird die Methode processClosingConnection mit IP-Adresse und Port des jeweiligen Clients aufgerufen. Ist der Server nicht mit dem in der Parameterliste spezifizierten Client verbunden, geschieht nichts.

void close()

Alle bestehenden Verbindungen zu Clients werden getrennt und der Server kann nicht mehr verwendet werden. Ist der Server bereits vor Aufruf der Methode in diesem Zustand, geschieht nichts.

void processNewConnection(String pClientIP, int pClientPort)

Diese Ereignisbehandlungsmethode wird aufgerufen, wenn sich ein Client mit IP-Adresse pClientIP und Portnummer pClientPort mit dem Server verbunden hat. Die Methode ist abstrakt und muss in einer Unterklassie der Klasse Server überschrieben werden, sodass auf den Neuaufbau der Verbindung reagiert wird. Der Aufruf der Methode erfolgt nicht synchronisiert.

void processMessage(String pClientIP, int pClientPort, String pMessage)

Diese Ereignisbehandlungsmethode wird aufgerufen, wenn der Server die Nachricht pMessage von dem durch pClientIP und pClientPort spezifizierten Client empfangen hat. Der vom Client hinzugefügte Zeilentrenner wurde zuvor entfernt. Die Methode ist abstrakt und muss in einer Unterklassie der Klasse Server überschrieben werden, sodass auf den Empfang der Nachricht reagiert wird. Der Aufruf der Methode erfolgt nicht synchronisiert.

void processClosingConnection(String pClientIP, int pClientPort)

Sofern der Server die Verbindung zu dem durch pClientIP und pClientPort spezifizierten Client trennt, wird diese Ereignisbehandlungsmethode aufgerufen, unmittelbar bevor die Verbindungstrennung tatsächlich erfolgt. Wird die Verbindung unvermittelt unterbrochen oder hat der in der Parameterliste spezifizierte Client die Verbindung zum Server unvermittelt getrennt, erfolgt der Methodenaufruf nach der Unterbrechung/Trennung der Verbindung. Die Methode ist abstrakt und muss in einer Unterklassie der Klasse Server überschrieben werden, sodass auf das Ende der Verbindung zum angegebenen Client reagiert wird. Der Aufruf der Methode erfolgt nicht synchronisiert.

Unterlagen für die Lehrkraft

Abiturprüfung 2023

Informatik, Leistungskurs

1. Aufgabenart

Analyse, Modellierung und Implementation kontextbezogener Problemstellungen mit Schwerpunkt auf den Inhalten Daten und ihre Strukturierung, Algorithmen und Informatiksysteme

2. Aufgabenstellung¹

siehe Prüfungsaufgabe

3. Materialgrundlage

entfällt

4. Bezüge zum Kernlehrplan und zu den Vorgaben 2023

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

1. Inhaltenfelder und inhaltliche Schwerpunkte

Daten und ihre Strukturierung

- Objekte und Klassen
 - Entwurfsdiagramme und Implementationsdiagramme
 - Lineare Strukturen
 - Array bis zweidimensional*
 - Lineare Liste (Klasse List)*

Algorithmen

- Analyse, Entwurf und Implementierung von Algorithmen
- Algorithmen in ausgewählten informatischen Kontexten
 - Algorithmen zur Kommunikation in Netzwerken
 - (Klasse Server)*

Formale Sprachen und Automaten

- Syntax und Semantik einer Programmiersprache
 - Java

Informatiksysteme

- Einzelrechner und Rechnernetzwerke

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)

Die Kommunikation zwischen Client und Server gestaltet sich wie folgt:

Peters Client an Server	Server an Peters Client
(Verbindungsaufnahme)	+NKS OK
SPIELER Peter	+SPIELER OK
SPIELSTART	+SPIELSTART OK SPIELERLISTE Peter(0) Ina(0) Harry(0) FRAGE Was ist ein(e) Dschong?
ANTWORT Ein Segelboot ostasiatischer Bauart	+ANTWORT OK
	ANTWORTEN 0, Ina:Eine traditionelle chinesische Kopfbedeckung. 1, Harry:Ein indonesischer Rikscha-Fahrer. 2, Peter:Ein Segelboot ostasiatischer Bauart. 3:Ein ostindisches Flächenmaß.
ENTSCHEIDUNG 1	+ENTSCHEIDUNG OK
	AUSWERTUNG Peter Ina@Harry

Teilaufgabe b)

Ein Objekt der Klasse NobodyKnowsServer erbt von der abstrakten Klasse Server und muss daher deren abstrakte Methoden implementieren. Es verwaltet darüber hinaus die Liste aller eingetragenen Spielerinnen und Spieler sowie ein Feld (Array) aller zur Verfügung stehenden Fragen, die in der aktuellen Spielrunde gestellte Frage und die darauf gegebenen Antworten.

Objekte der Klasse Frage verwalten den Text der Frage und den Text der Lösung zu dieser Frage.

Objekte der Klasse Antwort verwalten den Text der Antwort sowie dasjenige Spieler-Objekt einreicher, durch das diese Antwort eingereicht wurde. Außerdem verwalten sie eine Liste tipper aller Spieler-Objekte, die auf diese Antwort getippt haben, sie also für korrekt halten.

Objekte der Klasse **Spieler** schließlich verwalten die IP-Adresse und den Port ihres Clients sowie den Namen des Spielers oder der Spielerin und dessen oder deren Punkte.

Die in einer laufenden Spielrunde gegebenen Antworten können in einem Feld (Array) verwaltet werden, da die Anzahl der zu verwalteten Antworten konstant ist und mit Beginn der Spielrunde festliegt.

Eine einem **NobodyKnowsServer**-Objekt über den Aufruf der Methode **processMessage** übergebene Nachricht wird durch die IP-Adresse und den Port des Clients identifiziert, der die Nachricht gesendet hat. Da jeder Mitspieler und jede Mitspielerin in dem Spiel einen eigenen Client benutzt, identifizieren dessen IP-Adresse und Port das zugehörige **Spieler**-Objekt. Die Kombination IP-Adresse / Port kommt also einer ID für Spieler-Objekte gleich, sodass der Server darüber jede Nachricht einem **Spieler**-Objekt zuordnen kann.

Teilaufgabe c)

Die Methode besteht im Wesentlichen aus einer zentralen Schleife, die das Feld (Array) **aktAntworten** schrittweise durchgeht und abhängig vom Wert der booleschen Variablen **s** und **e** abbrechen kann (Zeilen 2 – 15). Bei jedem Schleifendurchlauf werden die zu diesen Variablen zugehörigen Kriterien untersucht und ihre Werte gesetzt:

- **s** wird **true**, wenn die gerade untersuchte Stelle **pos** im Feld auf **null** verweist, also insbesondere (noch) kein **Antwort**-Objekt verwaltet (Zeilen 6/7). Sollte dies der Fall sein, wird die Schleife abgebrochen.
- Bleibt **s** bei **false**, verwaltet die gerade untersuchte Stelle **pos** im Feld bereits ein **Antwort**-Objekt (Zeilen 9 – 13). Dann wird untersucht, ob diese Antwort denselben Einreicher bzw. Einreicherin hat wie die mit **pAntwort** übergebene Antwort, und das Ergebnis in **e** abgelegt. Sollte dies der Fall sein, wird die Schleife abgebrochen. In jedem Fall jedoch wird **pos** inkrementiert, sodass die Schleife ggf. mit dem nächsten Feldindex fortfahren kann, falls sie nicht durch **e == true** oder **s == true** abgebrochen wird.

Bezogen auf die drei angegebenen Situationen bedeutet dies, dass die Schleife jeweils mit folgenden Variablenbelegungen terminiert:

Situation	pos	e	s
i.	0	false	true
ii.	1	false	true
iii.	1	true	false

Nach dem Terminieren der Schleife wird in dem Fall, dass **s** den Wert **true** hat, die übergebene Antwort **pAntwort** an die durch **pos** markierte Stelle im Array **aktAntworten** eingesetzt (Zeilen 16 – 18).

Die Bedeutung der Variable **s** ist es anzusehen, ob die gerade untersuchte Stelle **pos** im **aktAntworten**-Array leer ist und folglich dort eine Antwort eingefügt werden kann. Falls dies nicht der Fall ist, zeigt **e** an, ob die Antwort an der gerade untersuchten Stelle vom selben Einreicher stammt wie die mit **pAntwort** übergebene Antwort.

Im Resultat dient die Methode der Aufnahme einer neuen Antwort in die Datensammlung `aktAntworten` aller Antworten für die aktuelle Spielrunde (und liefert genau dann `true` zurück, wenn dies erfolgreich durchgeführt werden konnte).

Teilaufgabe d)

Die Methode muss alle gegebenen Antworten durchgehen und für jede dieser Antworten die Liste `tipper` derjenigen Spielerinnen und Spieler durchgehen, die sich für diese Antwort entschieden haben. Handelt es sich um die korrekte Antwort (diese hat keinen Einreicher bzw. Einreicherin), muss die Punktzahl aller durch `tipper` verwalteten Spielerinnen und Spieler um 2 erhöht werden. Handelt es sich aber um eine ausgedachte, also von einem anderen Spieler oder einer anderen Spielerin eingereichte Antwort, muss die Punktzahl des zugehörigen Einreichers bzw. der Einreicherin (Spieler-Objekt, auf das das Attribut `einreicher` des Antwort-Objekts verweist) um 3 erhöht werden.

Außerdem muss die Methode die Serverausgabe generieren, die den Clients die Auswertung übermittelt. Sie muss also gemäß dem Serverprotokoll das Kommando `AUSWERTUNG` mit den Namenslisten der „Reingefallenen“ und „Richtigen“ erzeugen. Dies kann beim Durchgehen der jeweiligen `tipper`-Liste erledigt werden.

Eine mögliche Implementation der Methode lautet wie folgt:

```
private void werteAus() {
    String reingefallene = "";
    String richtige = "";
    for (int a = 0; a < aktAntworten.length; a++) {
        Spieler aktEinreicher = aktAntworten[a].gibEinreicher();
        List<Spieler> tipper = aktAntworten[a].gibTipper();
        tipper.toFirst();
        while (tipper.hasAccess()) {
            Spieler aktTipper = tipper.getContent();
            if (aktEinreicher == null) {
                if (richtige.length() > 0) {
                    richtige = richtige + "|";
                }
                richtige = richtige + aktTipper.gibName();
                aktTipper.setzePunkte(aktTipper.gibPunkte() + 2);
            } else {
                if (reingefallene.length() > 0) {
                    reingefallene = reingefallene + "|";
                }
                reingefallene = reingefallene + aktTipper.gibName();
                aktEinreicher.setzePunkte(aktEinreicher.gibPunkte() + 3);
            }
            tipper.next();
        }
    }
    sendToAll("AUSWERTUNG " + reingefallene + "@" + richtige);
}
```

Teilaufgabe e)

Eine neue Frage kann prinzipiell jederzeit übermittelt werden, egal, welche Spielphase gerade läuft. Eine Frage im Sinne der Modellierung enthält einen Fragetext und einen Lösungstext. Beide Informationen müssen mit dem neuen Kommando übermittelt werden, sodass beispielsweise folgende Form möglich ist:

NEUEFRAGE <FrageText> | <Lösungstext>

Prinzipiell ist die Verwendung sowohl eines Felds (Arrays) als auch eine dynamische Datenstruktur (wie eine Liste oder ein Binärbaum) zur Verwaltung des Fragekatalogs möglich. Für die dynamischen Datenstrukturen spricht, dass sie mit wenig Aufwand vergrößert werden können, was der neuen Anforderung entgegenkommt. Ein Feld müsste, um die Struktur zu vergrößern, neu angelegt und die bisherigen Fragen umkopiert werden, was sowohl signifikanten Rechen- als auch (zumindest temporär) Speicheraufwand bedeutet.

Die zufällige Auswahl einer Frage für eine neue Spielrunde ist hingegen mit einem Feld mit weniger Aufwand zu erledigen, da diese über eine Zufallszahl geschehen kann, die anhand der (bekannten) Feldgröße erzeugt wird. Der Zugriff auf die Frage kann dann direkt erfolgen. Bei den dynamischen Datenstrukturen ist die Größe a priori nicht bekannt, man könnte diese aber in einem zusätzlichen Attribut speichern oder bei Bedarf jeweils gesondert ermitteln. Auch kann nicht direkt auf ein zufälliges Element direkt zugegriffen werden, sondern es ist eine Traversierung der Datenstruktur erforderlich, was mit entsprechendem Rechen- bzw. Zeitaufwand verbunden ist.

Im Sachkontext der Serveranwendung ist zu erwarten, dass der Umfang des Fragenkatalogs schnell wachsen wird, weshalb der zusätzliche Speicherbedarf bei der Erweiterung der statischen Datenstruktur als kritisch angesehen werden muss. Insofern scheint die Verwendung einer dynamischen Datenstruktur insgesamt die bessere Wahl zu sein.

7. Teilleistungen – Kriterien / Bewertungsbogen zur Prüfungsarbeit

Name des Prüflings: _____ Kursbezeichnung: _____

Schule: _____

Teilaufgabe a)

	Anforderungen	Lösungsqualität			
		maximal erreichbare Punktzahl	EK ²	ZK	DK
1	ermittelt anhand des Protokolls die Kommandos, die von Peters Client an den Server gesendet wurden.	3			
2	ermittelt anhand des Protokolls die Nachrichten, die vom Server an Peters Client gesendet wurden.	5			
Sachlich richtige Lösungsalternative zur Modelllösung: (8)					
.....					
.....					
	Summe Teilaufgabe a)	8			

Teilaufgabe b)

	Anforderungen	Lösungsqualität			
		maximal erreichbare Punktzahl	EK	ZK	DK
1	erläutert die gegebene Modellierung hinsichtlich der Beziehungen der Klassen NobodyKnowsServer, Server, Antwort, Frage und Spieler untereinander.	6			
2	erläutert, weshalb die in der aktuellen Spielrunde gegebenen Antworten in einem Feld (Array) verwaltet werden können.	3			
3	erläutert, warum es sinnvoll ist, dass Objekte der Klasse Spieler eine IP-Adresse und einen Port verwalten.	3			
Sachlich richtige Lösungsalternative zur Modelllösung: (12)					
.....					
.....					
	Summe Teilaufgabe b)	12			

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

Teilaufgabe c)

	Anforderungen	Lösungsqualität			
		maximal erreichbare Punktzahl	EK	ZK	DK
1	analysiert und erläutert die Funktionsweise der Methode unter Bezugnahme auf die genannten drei Situationen.	6			
2	erläutert, welche Bedeutung die booleschen Variablen s und e im Sachkontext haben.	4			
3	erläutert den Zweck der Methode im Sachkontext.	2			
Sachlich richtige Lösungsalternative zur Modelllösung: (12)					
.....					
.....					
	Summe Teilaufgabe c)	12			

Teilaufgabe d)

	Anforderungen	Lösungsqualität			
		maximal erreichbare Punktzahl	EK	ZK	DK
1	entwickelt eine algorithmische Strategie für die Methode.	4			
2	implementiert die Methode.	8			
Sachlich richtige Lösungsalternative zur Modelllösung: (12)					
.....					
.....					
	Summe Teilaufgabe d)	12			

Teilaufgabe e)

	Anforderungen	Lösungsqualität			
		maximal erreichbare Punktzahl	EK	ZK	DK
1	entwickelt eine Erweiterung des Protokolls.	2			
2	nimmt im Sachkontext Stellung zu der Frage, ob die Verwaltung des Fragenkatalogs weiterhin mit einer statischen Datenstruktur oder mit einer dynamischen Datenstruktur erfolgen sollte.	4			
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 2023

Informatik, Leistungskurs

Aufgabenstellung:

Der Informatikprojektkurs möchte ausgewählte Bahnlinien in einer Datenbank abbilden. Dazu sollen in einer ersten Version u. a. die Bahnhöfe und die Streckenabschnitte gespeichert werden.

- Ein Streckenabschnitt ist eine direkte Verbindung von dem einen Bahnhof zu dem anderen Bahnhof ohne Zwischenhalt. Aus Sicherheitsgründen kann ein Streckenabschnitt nur in einer Richtung und somit nicht für die Gegenrichtung genutzt werden.
- Eine Bahnlinie besteht aus Streckenabschnitten, die die Folge der Bahnhöfe auf dieser Bahnlinie beschreiben. Dabei wird für jeden zugehörigen Streckenabschnitt die Fahrzeit in Minuten angegeben.

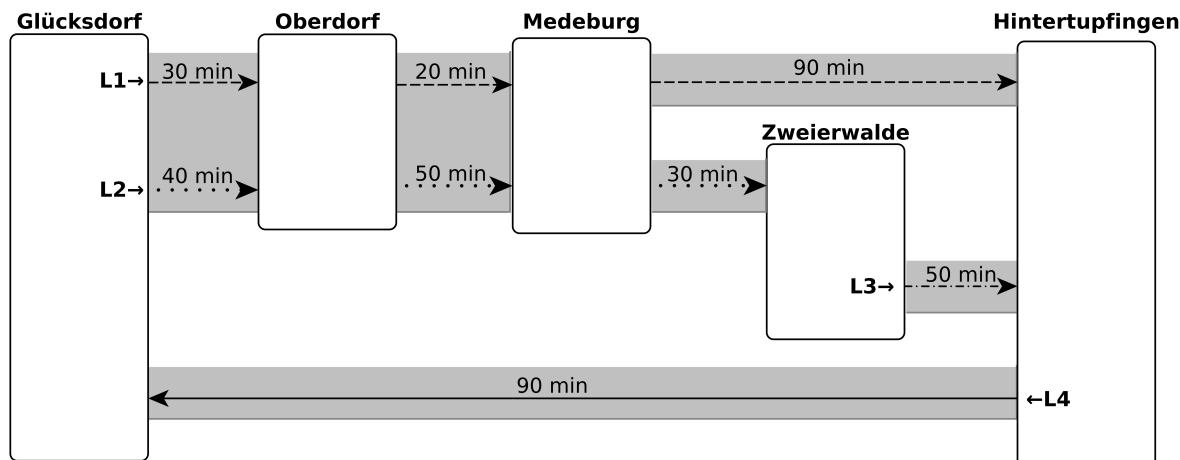


Abbildung 1: Schematische Darstellung der Bahnlinien (L1, L2, L3 und L4) und der Fahrzeiten auf den zugehörigen Streckenabschnitten zwischen den Bahnhöfen (Glücksdorf, Oberdorf, Medeburg, Zweierwalde und Hintertupfingen)



Name: _____

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

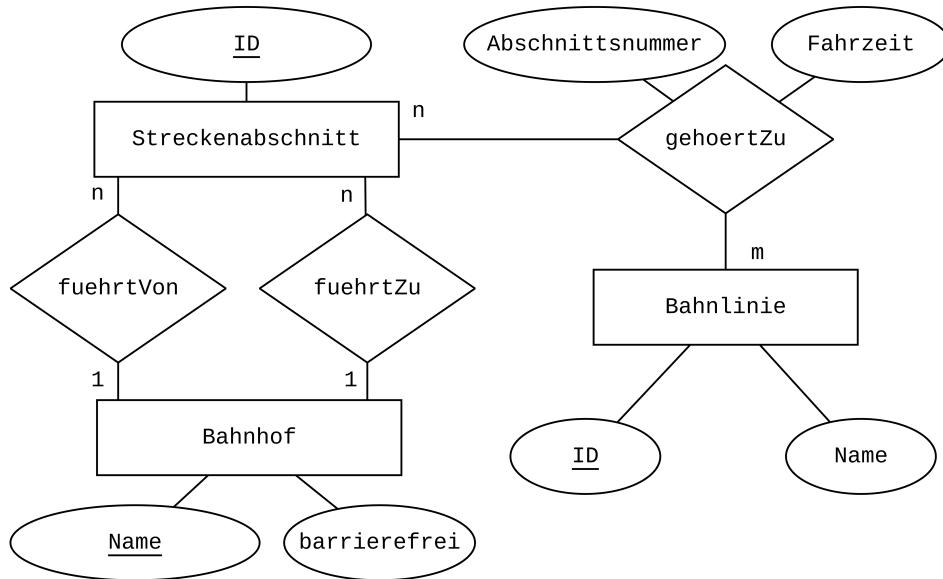


Abbildung 2: Teilmodellierung der Datenbank in Form eines Entity-Relationship-Modells

Im Folgenden soll mit dem in Abbildung 3 gegebenen Datenbankschema gearbeitet werden. Es stellt einen Ausschnitt der Gesamtdatenbank dar. Beispieldaten zu diesem Datenbankschema sind im Anhang zu finden.

Bahnhof (<u>Name</u> , barrierefrei) Streckenabschnitt (<u>ID</u> , ↑vonBahnhofName, ↑zuBahnhofName) Bahnlinie (<u>ID</u> , Name) gehoertZu (↑ <u>StreckenabschnittID</u> , ↑ <u>BahnlinieID</u> , Abschnittsnummer, Fahrzeit)
--

Abbildung 3: Datenbankschema eines Ausschnittes der Datenbank

- a) Erläutern Sie, wie die Bahnlinie L1 mit den zugehörigen Streckenabschnitten in der Datenbank (vgl. Beispieldaten) gespeichert ist.

Beschreiben Sie die in Abbildung 2 gegebene Teilmodellierung des Entity-Relationship-Modells.

Erläutern Sie jeweils unter Berücksichtigung der Kardinalitäten, wie die im Entity-Relationship-Modell abgebildeten Beziehungstypen im Datenbankschema umgesetzt wurden.

(3 + 5 + 5 Punkte)



Name: _____

b) Aus der Datenbank sollen folgende Informationen abgefragt werden:

- i. Gesucht sind für jede Bahnlinie der Name und die jeweilige Gesamtfahrzeit.
- ii. Gesucht sind alle Streckenabschnitte mit der maximalen Fahrzeit. Für diese Streckenabschnitte sollen jeweils die ID der Bahnlinie, der Name des „von-Bahnhofs“ sowie der Name des „zu-Bahnhofs“ und die Fahrzeit des Streckenabschnitts ausgegeben werden.

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

(3 + 5 Punkte)

c) Folgende SQL-Anweisung ist gegeben:

```
1 SELECT G1.BahnlinieID, S.zuBahnhofName
2 FROM (
3     SELECT BahnlinieID,
4            MAX(Abschnittsnummer) AS max
5     FROM gehoertZu
6     GROUP BY BahnlinieID
7 ) AS G1
8 INNER JOIN gehoertZu AS G2
9   ON G1.max = G2.Abschnittsnummer AND
10      G1.BahnlinieID = G2.BahnlinieID
11 INNER JOIN Streckenabschnitt AS S
12   ON G2.StreckenabschnittID = S.ID
```

Analysieren und erläutern Sie die Unterabfrage (Zeilen 3 – 6).

Analysieren und erläutern Sie die gesamte SQL-Anweisung.

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

(3 + 4 + 3 Punkte)



Name: _____

- d) Eine Schülerin aus dem Projektkurs hat noch eine Erweiterung entwickelt. Abbildung 4 zeigt einen Ausschnitt des Implementationsdiagramms.

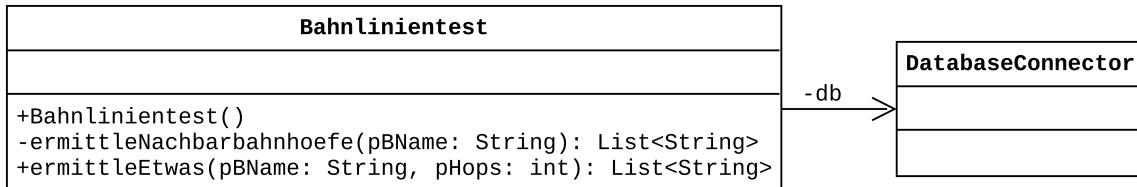


Abbildung 4: Teilmodellierung des Bahnlinientests

Die folgende Methode der Klasse Bahnlinientest ist undokumentiert:

```
1 public List<String> ermittleEtwas(String pBName, int pHops) {  
2     List<String> sammlung = new List<String>();  
3     List<String> nachbarn = ermittleNachbarbahnhoefe(pBName);  
4     while (pHops > 0) {  
5         List<String> temp = new List<String>();  
6         nachbarn.toFirst();  
7         while (nachbarn.hasAccess()) {  
8             String akt = nachbarn.getContent();  
9             sammlung.append(akt);  
10            nachbarn.remove();  
11            temp.concat(ermittleNachbarbahnhoefe(akt));  
12        }  
13        nachbarn = temp;  
14        pHops = pHops - 1;  
15    }  
16    return sammlung;  
17 }
```

Analysieren Sie die öffentliche Methode und erläutern Sie die Funktionsweise der Methode unter besonderer Berücksichtigung der Zeile 11.

Geben Sie jeweils die Rückgabe der Methode an, wenn die Methode mit pBName = "Zweierwalde" einmal mit pHops = 1 sowie ein weiteres Mal mit pHops = 2 für die Beispieldaten im Anhang aufgerufen wird.

Erläutern Sie im Sachkontext, welche Information die Methode ermittelt.

(6 + 6 + 2 Punkte)



Name: _____

- e) Ein Schüler schlägt vor, dass man den Entitätstyp **Bahnhof** ohne nennenswerte Nachteile streichen kann. Die einzelnen Bahnhofsnamen werden sowieso im Entitätstyp **Streckenabschnitt** als **vonBahnhofName** und **zuBahnhofName** aufgeführt. Die Barrierefreiheit für jeden Bahnhof könnte man schließlich auch in den Entitätstyp **Streckenabschnitt** verschieben und dort als Attribute **barrierefreiVonBahnhof** und **barrierefreiZuBahnhof** aufnehmen.

Beurteilen Sie den Vorschlag des Schülers.

(5 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

Bahnhof	
<u>Name</u>	<u>barrierefrei</u>
Glücksdorf	1
Oberdorf	0
Medeburg	1
Zweierwalde	1
Hintertupfingen	1

Streckenabschnitt		
<u>ID</u>	<u>vonBahnhofName</u>	<u>zuBahnhofname</u>
1	Glücksdorf	Oberdorf
2	Oberdorf	Medeburg
3	Medeburg	Hintertupfingen
4	Medeburg	Zweierwalde
5	Zweierwalde	Hintertupfingen
6	Hintertupfingen	Glücksdorf

Bahnlinie	
<u>ID</u>	<u>Name</u>
L1	Bergexpress
L2	Burglinie
L3	Hintertupfingenlinie
L4	Glücksdorfexpress

gehoertZu			
<u>StreckenabschnittID</u>	<u>BahnlinieID</u>	<u>Abschnittsnummer</u>	<u>Fahrzeit</u>
1	L1	1	30
2	L1	2	20
3	L1	3	90
1	L2	1	40
2	L2	2	50
4	L2	3	30
5	L3	1	50
6	L4	1	90



Name: _____

Anhang

Dokumentationen der verwendeten Klassen

Die Klasse Bahnlinentest

Ein Objekt der Klasse wird genutzt, um die Datenbankverbindung und die Datenbankabfragen zu testen.

Bahnlinentest()

Ein Objekt vom Typ DatabaseConnector wird initialisiert und stellt die Datenbankverbindung her.

List<String> ermittleEtwas(String pBName, int pHops)

Diese Methode ist Bestandteil der Teilaufgabe d.

Darüber hinaus verfügt die Klasse über folgende private Methode:

List<String> ermittleNachbarbahnhöfe(String pBName)

Die Methode liefert eine Liste mit den Namen der Nachbarbahnhöfe für den im Parameter übergebenen Bahnhof pBName zurück.

Nachbarbahnhöfe sind die Bahnhöfe, zu denen ein Streckenabschnitt ohne Zwischenhalt von diesem Bahnhof pBName führt.

Falls der Bahnhofsname pBName nicht existiert, wird eine leere Liste zurückgeliefert.



Name: _____

Die Klasse DatabaseConnector

Ein Objekt der Klasse DatabaseConnector ermöglicht die Abfrage und Manipulation einer MySQL-Datenbank.

Beim Erzeugen des Objekts wird eine Datenbankverbindung aufgebaut, sodass anschließend SQL-Anweisungen an diese Datenbank gerichtet werden können.

Dokumentation der Klasse DatabaseConnector

**DatabaseConnector(String pIP, String pPort,
String pDatabase, String pUsername, String pPassword)**

Ein Objekt vom Typ DatabaseConnector wird erstellt, und eine Verbindung zur Datenbank wird aufgebaut. Mit den Parametern pIP und pPort werden die IP-Adresse und die Port-Nummer übergeben, unter denen die Datenbank mit Namen pDatabase zu erreichen ist. Mit den Parametern pUsername und pPassword werden Benutzername und Passwort für die Datenbank übergeben.

void executeStatement(String pSQLStatement)

Der Auftrag schickt den im Parameter pSQLStatement enthaltenen SQL-Befehl an die Datenbank ab.

Handelt es sich bei pSQLStatement um einen SQL-Befehl, der eine Ergebnismenge liefert, so kann dieses Ergebnis anschließend mit der Methode getCurrentQueryResult abgerufen werden.

QueryResult getCurrentQueryResult()

Die Anfrage liefert das Ergebnis des letzten mit der Methode executeStatement an die Datenbank geschickten SQL-Befehls als Objekt vom Typ QueryResult zurück.

Wurde bisher kein SQL-Befehl abgeschickt oder ergab der letzte Aufruf von executeStatement keine Ergebnismenge (z. B. bei einem INSERT-Befehl oder einem Syntaxfehler), so wird null geliefert.

String getErrorMessage()

Die Anfrage liefert null oder eine Fehlermeldung, die sich jeweils auf die letzte zuvor ausgeführte Datenbankoperation bezieht.

void close()

Die Datenbankverbindung wird geschlossen.



Name: _____

Die Klasse QueryResult

Ein Objekt der Klasse `QueryResult` stellt die Ergebnistabelle einer Datenbankanfrage mit Hilfe der Klasse `DatabaseConnector` dar. Objekte dieser Klasse werden nur von der Klasse `DatabaseConnector` erstellt. Die Klasse verfügt über keinen öffentlichen Konstruktor.

Dokumentation der Klasse QueryResult

`String[][] getData()`

Die Anfrage liefert die Einträge der Ergebnistabelle als zweidimensionales Feld vom Typ `String`. Der erste Index des Feldes stellt die Zeile und der zweite die Spalte dar (d. h. `String[zeile][spalte]`).

`String[] getColumnNames()`

Die Anfrage liefert die Bezeichner der Spalten der Ergebnistabelle als Feld vom Typ `String` zurück.

`String[] getColumnTypes()`

Die Anfrage liefert die Typenbezeichnung der Spalten der Ergebnistabelle als Feld vom Typ `String` zurück. Die Bezeichnungen entsprechen den Angaben in der MySQL-Datenbank.

`int getRowCount()`

Die Anfrage liefert die Anzahl der Zeilen der Ergebnistabelle als `int`.

`int getColumnCount()`

Die Anfrage liefert die Anzahl der Spalten der Ergebnistabelle als `int`.

Unterlagen für die Lehrkraft

Abiturprüfung 2023

Informatik, Leistungskurs

1. Aufgabenart

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

2. Aufgabenstellung¹

siehe Prüfungsaufgabe

3. Materialgrundlage

entfällt

4. Bezüge zum Kernlehrplan und zu den Vorgaben 2023

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*
 - Lineare Liste (Klasse List)*

Algorithmen

- Analyse, Entwurf und Implementierung von Algorithmen
- Formale Sprachen und Automaten
- Syntax und Semantik einer Programmiersprache
 - SQL
 - Java

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 Bahnlinie mit der ID L1 und ihrer Bezeichnung Bergexpress ist in der Relation **Bahnlinie** gespeichert. In der Relation **gehoertZu** befinden sich die drei Streckenabschnitte der Bahnlinie mit der ID L1 mit der jeweiligen Abschnittsnummer und der jeweiligen Fahrzeit. Diese drei Streckenabschnitte mit der Angabe, von welchem Bahnhof sie jeweils zu einem anderen Bahnhof führen, befinden sich in der Relation **Streckenabschnitt**.

Die in Abbildung 2 gegebene Teilmodellierung besteht aus drei Entitätstypen und drei Beziehungstypen.

Im Entitätstyp **Bahnhof** wird als Primärschlüssel das Attribut **Name** verwendet.

Im Entitätstyp **Bahnlinie** wird als Primärschlüssel das Attribut **ID** verwendet.

Im Entitätstyp **Streckenabschnitt** wird als Primärschlüssel das Attribut **ID** verwendet.

Entitäten des Typs **Bahnhof** haben das weitere Attribut **barrierefrei**.

Entitäten des Typs **Bahnlinie** haben das weitere Attribut **Name**.

Der Beziehungstyp **gehoertZu** modelliert, welcher Streckenabschnitt zu welcher Bahnlinie gehört. Zusätzlich hat der Beziehungstyp die Attribute **Abschnittsnummer** und **Fahrzeit**.

1:n-Beziehungstyp fuehrtVon:

Ein Bahnhof kann für mehrere Streckenabschnitte der Ausgangsbahnhof sein. Ein Streckenabschnitt besitzt genau einen Ausgangsbahnhof.

1:n-Beziehungstyp fuehrtZu:

Ein Bahnhof kann für mehrere Streckenabschnitte der Zielbahnhof sein. Ein Streckenabschnitt besitzt genau einen Zielbahnhof.

n:m-Beziehungstyp gehoertZu:

Ein Streckenabschnitt kann zu mehreren Bahnlinien gehören. Eine Bahnlinie kann aus mehreren Streckenabschnitten bestehen.

Die **1:n-Beziehungstypen fuehrtVon** und **fuehrtZu** wurden so umgesetzt, dass in der Relation **Streckenabschnitt** zusätzlich die Fremdschlüsselattribute **vonBahnhofName** und **zuBahnhofName** verwaltet werden, welche jeweils den Primärschlüssel aus der Relation **Bahnhof** abbilden.

Der n:m-Beziehungstyp gehoertZu wurde mit dem Relationenschema gehoertZu umgesetzt. In dieser Relation werden die zwei Fremdschlüsselattribute StreckenabschnittID und BahnlinieID verwaltet, welche jeweils die Primärschlüssel einmal in der Relation Streckenabschnitt und einmal in der Relation Bahnlinie darstellen.

Teilaufgabe b)

- i.

```
SELECT Bahnlinie.Name, SUM(gehoertZu.Fahrzeit)
  FROM Bahnlinie
    INNER JOIN gehoertZu
      ON Bahnlinie.ID = gehoertZu.BahnlinieID
 GROUP BY Bahnlinie.ID
```
- ii.

```
SELECT gehoertZu.BahnlinieID, Streckenabschnitt.vonBahnhofName,
          Streckenabschnitt.zuBahnhofname, gehoertZu.Fahrzeit
        FROM Streckenabschnitt
          INNER JOIN gehoertZu
            ON gehoertZu.StreckenabschnittID = Streckenabschnitt.ID
       WHERE Fahrzeit = (
          SELECT MAX(gehoertZu.Fahrzeit)
        FROM gehoertZu
      )
```

Teilaufgabe c)

In der Unterabfrage wird für jede Bahnlinie (vgl. Zeile 6) die größte Abschnittsnummer (vgl. Zeile 4) der zugehörigen Streckenabschnitte bestimmt.

In Zeile 1 findet die Projektion auf die BahnlinienID, welche man aus der Unterabfrage erhält, und den zuBahnhofName statt.

Die Ergebnistabelle der Unterabfrage wird erneut mit der Tabelle gehoertZu über einen Innerjoin verknüpft (vgl. Zeilen 8 – 10). Die Verknüpfung findet über die BahnlinienID und die Abschnittsnummer statt, die der in der Unterabfrage ermittelten größten Abschnittsnummer eines Streckenabschnitts entspricht.

In den Zeilen 11 – 12 findet eine weitere Verknüpfung zur Tabelle Streckenabschnitt mit einem Innerjoin statt. Die Verknüpfung erfolgt über die StreckenabschnittID.

Im Sachzusammenhang wird für jede Bahnlinie der Zielbahnhof ermittelt.

Teilaufgabe d)

Die öffentliche Methode bekommt mit Parameter einen Bahnhofsnamen pBName und eine Anzahl pHops übergeben und liefert eine Liste mit String-Objekten zurück.

In Zeile 2 wird eine lokale anfangs leere Liste für String-Objekte erzeugt, die in Zeile 16 zurückgeliefert wird.

In Zeile 3 wird eine Liste für String-Objekte mit dem Namen nachbarn deklariert.

Diesem Objekt wird durch den Aufruf der privaten Methode eine Liste mit den Nachbarbahnhöfen von dem Bahnhof pBName zugewiesen.

Die Schleife in den Zeilen 4 – 15 stellt sicher, dass, solange die Anzahl hops größer 0 ist, bei jedem Schleifendurchlauf die Liste nachbarn einmal komplett abgearbeitet und gelöscht wird (vgl. Zeilen 6 – 12). Dabei wird eine neue lokale und leere Liste für String-Objekte mit Bezeichner temp erzeugt (vgl. Zeile 5).

Die Abarbeitung der Liste nachbarn erfolgt wie folgt (vgl. Zeile 7 – 12), bis sie leer ist (vgl. Zeile 8): Jedes Objekt von der Liste nachbarn wird an die Liste Sammlung angehängt (vgl. Zeile 9) und von der Liste nachbarn gelöscht (vgl. Zeile 10). In Zeile 11 werden an die Liste temp alle Nachbarbahnhöfe für den aktuellen Bahnhof angehängt.

Nachdem eine Liste mit Nachbarn komplett abgearbeitet wurde, wird die Liste temp zur neuen Liste der Nachbarn (vgl. Zeile 13) und die Anzahl der pHops wird um den Wert 1 verringert.

Rückgaben für den Methodenaufruf mit den folgenden Parametern:

pBName = "Zweierwalde" und pHops = 1

=> Liste: Hintertupfingen

Rückgaben für den Methodenaufruf mit den folgenden Parametern:

pBName = "Zweierwalde" und pHops = 2

=> Liste: Hintertupfingen, Glücksdorf

Im Sachkontext liefert die Methode die Bahnhöfe zurück, die von dem im Parameter übergebenen Bahnhof aus mit einer im Parameter übergebenen Anzahl an maximalen Haltestellen erreichbar sind.

Teilaufgabe e)

Der Vorschlag des Schülers ist abzulehnen, weil eine Datenkonsistenz wie im bisherigen Modell nicht mehr sichergestellt werden kann.

Im bisherigen Modell hängt das Nichtschlüsselattribut barrierefrei eindeutig vom Primärschlüssel Name im Entitätstyp Bahnhof ab. Es wurde somit für jedes Bahnhof eindeutig ein Wert für die Barrierefreiheit zugeordnet. Im von dem Schüler vorgeschlagenen Modell wäre es möglich, dass für den gleichen Bahnhof unterschiedliche Werte für die Barrierefreiheit eingetragen werden.

7. Teilleistungen – Kriterien / Bewertungsbogen zur Prüfungsarbeit

Name des Prüflings: _____ Kursbezeichnung: _____

Schule: _____

Teilaufgabe a)

	Anforderungen	Lösungsqualität			
		maximal erreichbare Punktzahl	EK ²	ZK	DK
1	erläutert, wie die Bahnlinie L1 mit den zugehörigen Streckenabschnitten in der Datenbank (vgl. Beispieldaten) gespeichert ist.	3			
2	beschreibt die in Abbildung 2 gegebene Teilmodellierung des Entity-Relationship-Modells.	5			
3	erläutert jeweils unter Berücksichtigung der Kardinalitäten, wie die im Entity-Relationship-Modell abgebildeten Beziehungstypen im Datenbankschema umgesetzt wurden.	5			
Sachlich richtige Lösungsalternative zur Modelllösung: (13)					
.....					
.....					
	Summe Teilaufgabe a)	13			

Teilaufgabe b)

	Anforderungen	Lösungsqualität			
		maximal erreichbare Punktzahl	EK	ZK	DK
1	entwirft für die Anfrage i eine SQL-Anweisung.	3			
2	entwirft für die Anfrage ii eine SQL-Anweisung.	5			
Sachlich richtige Lösungsalternative zur Modelllösung: (8)					
.....					
.....					
	Summe Teilaufgabe b)	8			

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

Teilaufgabe c)

	Anforderungen	Lösungsqualität			
		maximal erreichbare Punktzahl	EK	ZK	DK
1	analysiert und erläutert die Unterabfrage.	3			
2	analysiert und erläutert die gesamte SQL-Anweisung.	4			
3	erläutert im Sachzusammenhang, welche Information die gesamte SQL-Anweisung ermittelt.	3			
Sachlich richtige Lösungsalternative zur Modelllösung: (10)					
.....					
.....					
	Summe Teilaufgabe c)	10			

Teilaufgabe d)

	Anforderungen	Lösungsqualität			
		maximal erreichbare Punktzahl	EK	ZK	DK
1	analysiert die Methode und erläutert die Funktionsweise der Methode unter Berücksichtigung der Zeile 11.	6			
2	gibt jeweils die Rückgabe der Methode an, wenn die Methode mit den entsprechenden Daten aufgerufen wird.	6			
3	erläutert im Sachkontext, welche Information die Methode ermittelt.	2			
Sachlich richtige Lösungsalternative zur Modelllösung: (14)					
.....					
.....					
	Summe Teilaufgabe d)	14			

Teilaufgabe e)

	Anforderungen	Lösungsqualität			
		maximal erreichbare Punktzahl	EK	ZK	DK
1	beurteilt den Vorschlag des Schülers.	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 2023

Informatik, Leistungskurs

Aufgabenstellung:

Bei der Billard-Variante Snooker befinden sich auf dem Tisch anfangs 15 rote Kugeln, *Rote* (r) genannt, und 6 Kugeln anderer Farben, die zusammenfassend als *Farbige* (f) bezeichnet werden. Darüber hinaus liegt noch eine weiße Kugel, genannt die *Weiß*e (w), auf dem Tisch (vgl. Abbildung 1).

Ziel eines Spielzugs (auch „Stoß“ genannt) ist es, eine der anderen Kugeln durch Anstoßen der Weißen mit dem Billardstock so anzuspielen, dass diese Kugel nach der Kollision mit der Weißen *eingelocht* wird, d. h., diese Kugel fällt in eines der Löcher am Tischrand (die sogenannten „Taschen“).

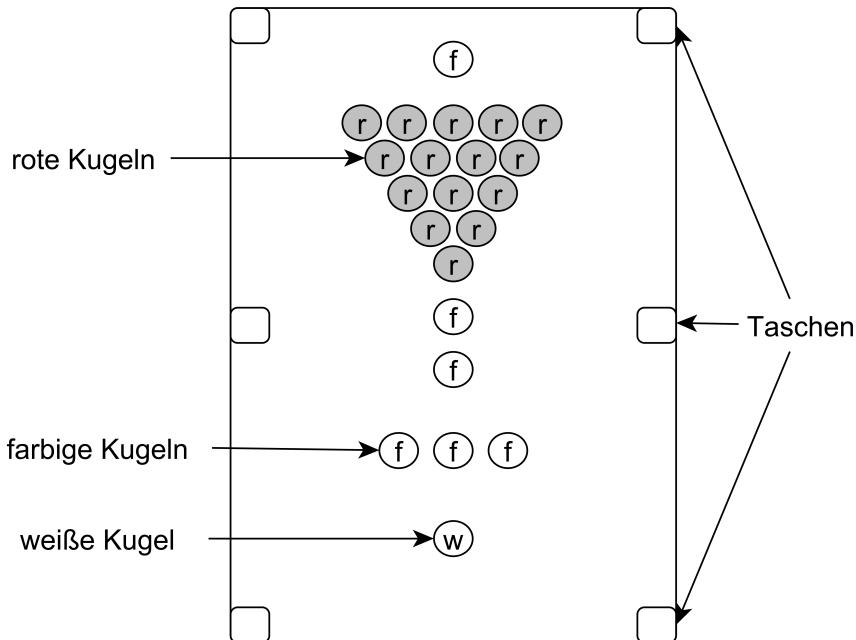


Abbildung 1: Snookertisch zu Beginn eines Spiels (Skizze)

Snookerschüler Jonte und sein Trainer Max spielen eine Trainingsvariante des Spiels, bei der alle eingelochten Kugeln sofort wieder auf den Tisch zurückgelegt werden.

Wer an der Reihe ist, muss beim ersten Stoß genau eine Rote einlochen. Gelingt dies, bleibt man am Spiel und man muss beim nächsten Stoß versuchen, genau eine Farbige einzulochen. Wenn auch dies gelingt, so darf man nun wie zuvor so lange abwechselnd Rote und Farbige einlochen, bis man einen Fehler macht.



Name: _____

Es gilt als Fehler, wenn die falsche Kugel sortiert angespielt und eingelocht wird, wenn die weiße Kugel eingelocht wird oder wenn bei einem Stoß keine Kugel eingelocht wird. Auch fehlerhaft eingelochte Kugeln werden wieder auf die Anfangsposition auf den Tisch gelegt. Nach einem Fehler gibt es einen Spielerwechsel.

Vereinfachend gehen wir davon aus, dass nur folgende Ereignisse im Training eintreten:

- Stoß (s) – Ein neuer Stoß beginnt durch das Anstoßen der Weißen mit dem Billardstock.

Nach dem Anstoßen der Weißen tritt genau eines der folgenden vier Ereignisse ein:

- Eine rote Kugel wird angespielt und eingelocht (r).
- Eine farbige Kugel wird angespielt und eingelocht (f).
- Die weiße Kugel wird eingelocht (w).
- Keine Kugel wird eingelocht (k).

Zusätzlich gehen wir davon aus, dass nie mehr als eine Kugel pro Stoß eingelocht wird.

Jonte beginnt alle Trainingsspiele. Er und sein Trainer trainieren so lange, bis sie keine Lust mehr haben. Dabei gehört das Ergebnis des letzten Stoßes immer zum Spiel dazu.

Jonte und Max protokollieren ihre Spiele, um sie später für das weitere Training auszuwerten. Die Trainingsspiele werden protokolliert, indem die zu den Ereignissen gehörigen Buchstaben in der Reihenfolge ihres Auftretens notiert werden.

Beispiel für ein sehr kurzes Trainingsspiel: Jonte locht eine Rote und eine Farbige ein, beim dritten Stoß locht er keine Kugel ein. Max locht dann mit seinem ersten Stoß eine Rote ein. Das Protokoll dazu lautet: srsfsksr

Um eine Vielzahl von Protokollen auswerten zu können, wurde ein deterministischer endlicher Automat A dazu entworfen, um Spielprotokolle zu untersuchen. In Abbildung 2 ist sein Zustandsübergangsdiagramm dargestellt.

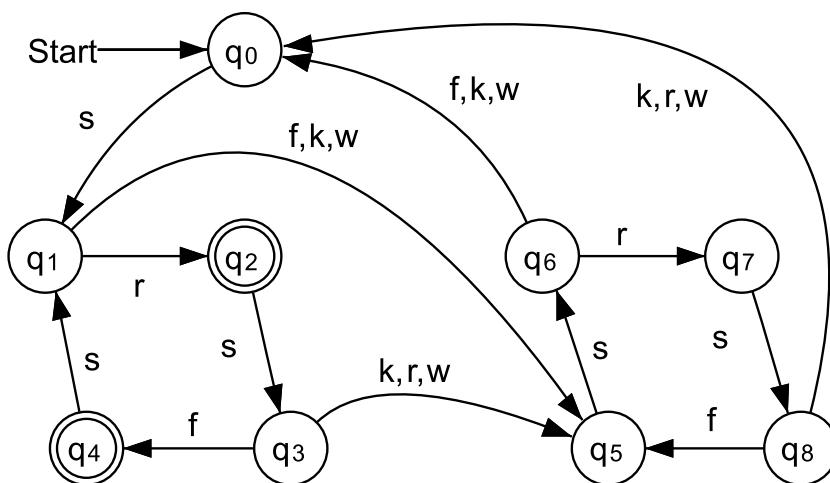


Abbildung 2: Zustandsübergangsdiagramm des Automaten A



Name: _____

- a) Geben Sie die Menge der Zustände, die Menge der Endzustände und den Startzustand des Automaten A an.

Ermitteln Sie ein Wort, das vom Automaten A akzeptiert wird und bei dessen Abarbeitung der Automat alle Zustände besucht, und erläutern Sie den Ablauf des Trainingsspiels, das mithilfe dieses Wortes protokolliert wurde.

Erläutern Sie die Bedeutung der Zustände q_0 , q_2 und q_6 und die Übergänge aus diesen drei Zuständen heraus im Sachzusammenhang.

(3 + 4 + 6 Punkte)

- b) Bei manchen Trainingsspielen passiert es häufig, dass bei einem Stoß aus Versehen die weiße Kugel eingelocht wird. Bei anderen Spielen geschieht das nur maximal einmal im ganzen Trainingsspiel.

Entwerfen Sie eine reguläre Grammatik G, die alle Wörter erzeugt, die Spielprotokolle darstellen, bei denen nur maximal einmal die weiße Kugel eingelocht wurde.

Entscheiden Sie durch Ableiten aus der von Ihnen entworfenen Grammatik G, ob die Zeichenketten srswsfk und swsfsw zur Sprache der Grammatik G gehören.

(6 + 4 Punkte)

- c) Die Spielregeln sollen erweitert werden. So soll unter bestimmten Voraussetzungen erlaubt sein, wenn bei einem Stoß mehr als eine Kugel in die Tasche fällt und auf dem Spielfeld weitere Kugelberührungen passieren:

- Der Spieler, der dran ist, muss nach dem ersten Stoß zuerst eine Rote einlochen, danach sind – beginnend mit einer Farbigen – abwechselnd wieder Farbige und Rote (wie zuvor) erlaubt.
- Außerdem muss bei jedem Stoß mindestens eine Rote oder Farbige eingelocht werden.

Wird die Weiße eingelocht oder fallen die Kugelsorten nicht abwechselnd oder in der falschen Reihenfolge (z. B. erst eine Farbige, obwohl eine Rote zu lochen wäre) in die Taschen, so gilt das immer noch als Fehler.

Es soll ein deterministischer endlicher Automat B mit dem Eingabealphabet $\{f, k, r, s, w\}$ entworfen werden, der ein Spielprotokoll akzeptiert, wenn beim zugehörigen Trainingsspiel die erweiterten Regeln – ohne Fehler – eingehalten wurden.

Geben Sie ein möglichst kurzes Wort zu einem Spielprotokoll an, bei dem Jonte insgesamt 5 Kugeln einlocht, ohne – gemäß den erweiterten Regeln – einen Fehler zu machen.

Geben Sie ein Wort der Länge 7 zu einem Spielprotokoll an, bei dem entsprechend den erweiterten Regeln ohne Fehler genau zweimal die Weiße gestoßen wurde.

Entwerfen Sie ein Zustandsübergangsdiagramm für den Automaten B.

(2 + 2 + 7 Punkte)



Name: _____

- d) Der Trainer Max ist ein erfahrener Snookerspieler, dem kaum noch Fehler, wie das versehentliche Einlochen der weißen Spielkugel, passieren. Er beobachtet die Trainingsspiele genau und hat das Gefühl, dass Jonte mittlerweile auch nicht mehr so häufig aus Versehen die Weiße einlocht.

Er möchte die Spielprotokolle mithilfe eines Kellerautomaten diesbezüglich prüfen lassen. Dazu hängt er an ein Spielprotokoll aus der jüngsten Vergangenheit ein Trennzeichen t an und dann ein Spielprotokoll aus früheren Zeiten.

Da ihn nur das Zeichen w interessiert, werden im zusammengesetzten Eingabewort alle anderen Zeichen – außer w – durch das Zeichen z ersetzt.

Wenn im zweiten Protokoll (nach dem Trennzeichen) das Zeichen w häufiger als im ersten Protokoll (vor dem Trennzeichen) vorkommt, so nimmt er an, dass sich Jonte wirklich verbessert hat.

Entwerfen Sie einen Kellerautomaten, der ein Eingabewort in der beschriebenen Form erhält und entscheidet, ob sich Jonte im direkten Vergleich der beiden zu untersuchenden Trainingsspiele verbessert hat.

Stellen Sie die Erkennung des Wortes zwtwww dar.

(7 + 4 Punkte)

- e) Ab jetzt spielen Jonte und Max eine Spielvariante, bei der die Reihenfolge der eingelochten Kugelfarben egal ist. Nur die eingelochten Roten verbleiben in den Taschen, die eingelochten Farbigen werden wieder auf das Spielfeld gelegt. Es gibt einen Spielerwechsel, wenn bei einem Stoß keine Kugel oder die weiße Kugel eingelocht wird. Wer die letzte der 15 Roten einlocht, gewinnt das Spiel.

Jonte behauptet, dass man ein korrekt protokolliertes Spielprotokoll mithilfe eines endlichen Automaten daraufhin untersuchen kann, ob bei dem dazugehörigen Spiel alle 15 Roten eingelocht wurden.

Max glaubt, dass das nicht möglich ist, weil die Protokolle durch die Spielerwechsel bei Fehlern beliebig lang werden können und endliche Automaten nicht beliebig weit zählen können.

Beurteilen Sie, ob Jonte recht hat.

(5 Punkte)

Zugelassene Hilfsmittel:

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

Unterlagen für die Lehrkraft

Abiturprüfung 2023

Informatik, Leistungskurs

1. Aufgabenart

Analyse, Modellierung und Implementation kontextbezogener Problemstellungen mit Schwerpunkt auf dem Inhaltsfeld Formale Sprachen und Automaten

2. Aufgabenstellung¹

siehe Prüfungsaufgabe

3. Materialgrundlage

entfällt

4. Bezüge zum Kernlehrplan und zu den Vorgaben 2023

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

1. Inhaltsfelder und inhaltliche Schwerpunkte

Formale Sprachen und Automaten

- Endliche Automaten
 - Deterministische endliche Automaten
 - Nichtdeterministische endliche Automaten
- Grammatiken regulärer und kontextfreier Sprachen
- Möglichkeiten und Grenzen von Automaten und formalen Sprachen

2. Medien/Materialien

- entfällt

5. Zugelassene Hilfsmittel

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

¹ Die Aufgabenstellung deckt inhaltlich alle drei Anforderungsbereiche ab.

6. Modelllösungen

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

Teilaufgabe a)

Menge der Zustände: $\{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8\}$

Menge der Endzustände: $\{q_2, q_4\}$

Startzustand: q_0

Das Wort `sfsrsrsrsf` endet in Zustand q_4 und wird somit akzeptiert. Es werden alle Zustände in dieser Reihenfolge $q_0, q_1, q_5, q_6, q_7, q_8, q_0, q_1, q_2, q_3$ und q_4 besucht. Bei dem dazugehörigen Trainingsspiel locht Jonte beim ersten Stoß eine Farbige ein. Dann locht Max zweimal nacheinander eine Rote ein. Jonte locht daraufhin eine Rote und danach eine Farbige ein.

Der Zustand q_0 ist der Startzustand, der allerdings auch erreicht wird, wenn zuvor Max einen Fehler gemacht hat und somit ein Spielerwechsel erfolgt. Als nächstes darf Jonte die Weiße stoßen, und es erfolgt der Übergang zu q_1 .

Der Zustand q_2 stellt den Zustand dar, bei dem Jonte zuletzt regelgerecht eine Rote eingelocht hat. Danach darf Jonte die Weiße stoßen, und es erfolgt der Übergang zu q_3 .

Der Zustand q_6 stellt den Zustand dar, bei dem Max zuletzt die Weiße gestoßen hat, nachdem er regelgerecht eine Farbige eingelocht hat oder durch einen Fehler von Jonte ans Spiel gekommen ist. Locht er mit diesem Stoß eine Rote ein, so erfolgt der Übergang zu q_7 und Max bleibt am Spiel. Locht er keine, die Weiße oder wieder eine Farbige ein, so erfolgt der Übergang zu q_0 . Das bedeutet, dass danach Jonte dran ist.

Teilaufgabe b)

Startsymbol: S_G

Nichtterminale: $N = \{S_G, A, B, C\}$

Terminale: $T = \{f, k, r, s, w\}$

Produktionen:

$$\begin{aligned} P = & \{ S_G \rightarrow sA, \\ & A \rightarrow fS_G \mid kS_G \mid rS_G \mid wB \mid f \mid k \mid r \mid w, \\ & B \rightarrow sC, \\ & C \rightarrow fB \mid kB \mid rB \mid f \mid k \mid r \} \end{aligned}$$

Ableitung des Wortes $srswsfsk$:

$S \rightarrow sA \rightarrow srS_6 \rightarrow srsA \rightarrow srswB \rightarrow srswsC \rightarrow srswsfB \rightarrow srswsfsC \rightarrow srswsfsk$

Das Wort gehört zur Sprache der Grammatik.

Versuchte Ableitung des Wortes $sdfsfs$:

$S \rightarrow sA \rightarrow swB \rightarrow swSC \rightarrow swsfB \rightarrow swfsC \rightarrow ?$

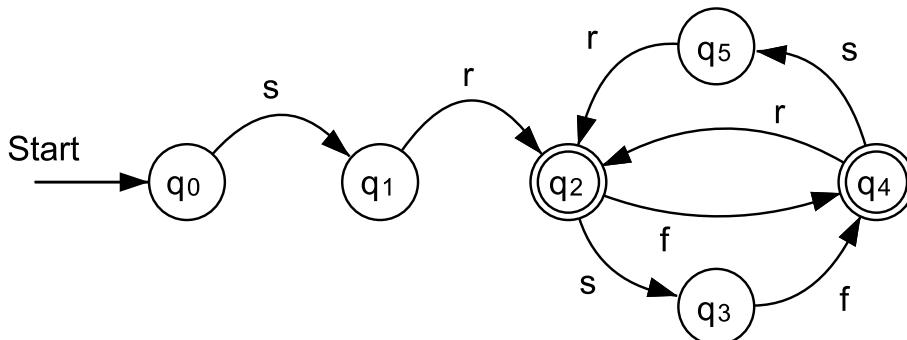
Es gibt keine Produktion mit dem Nichtterminal C auf der linken Seite, dessen rechte Seite mit dem Terminal w beginnt. Das Wort gehört nicht zur Sprache der Grammatik.

Teilaufgabe c)

Ein minimal langes Wort zu einem Spielprotokoll, bei dem Jonte insgesamt 5 Kugeln einlocht, ohne einen Fehler zu machen: $srfrrfr$

Wort der Länge 7 zu einem Spielprotokoll, bei dem ohne Fehler zweimal die Weiße gestoßen wurde: $srsfrfr$

Ein mögliches Zustandsübergangsdiagramm für den Automaten B:



Nicht dargestellte Übergänge führen in einen Fehlerzustand.

Teilaufgabe d)

Kellerautomat:

Zustandsmenge: $\{q_0, q_1, q_2\}$

Eingabealphabet: $\{z, w, t\}$

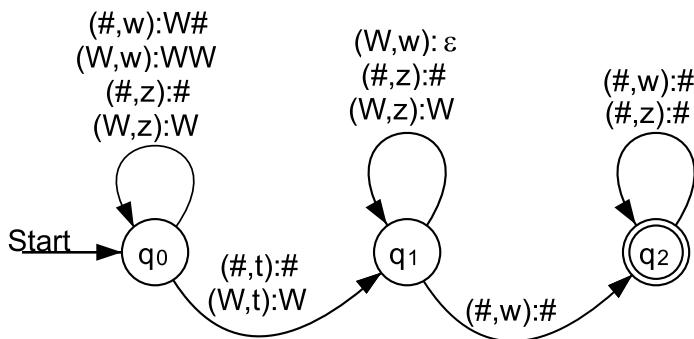
Kelleralphabet: $\{\#, W\}$

Anfangszustand: q_0

Kellerstartsymbol: $\#$

Menge der Endzustände: $\{q_2\}$

Übergangsfunktion:



Erkennung des Wortes zwtwww:

Zustand	Gelesenes Eingabesymbol	Kellerinhalt / oberstes Symbol	Neuer Kellerinhalt / oberstes Symbol	Neuer Zustand
q ₀	z	#	#	q ₀
q ₀	w	#	w	q ₀
q ₀	t	w	w	q ₁
q ₁	w	w	#	q ₁
q ₁	w	#	#	q ₂
q ₂	w	#	#	q ₂

Teilaufgabe e)

Jonte ist im Recht. Solch einen Automaten kann man entwickeln.

Der Automat könnte dabei diese Strategie verfolgen:

- Der Automat verbleibt im Startzustand q₀, solange er eines der Zeichen k, w, f oder s liest, und er wechselt in den Zustand q₁, wenn er das Zeichen r liest.
- Analog verhält er sich im Zustand q₁ als neuem „Startzustand“ und allen folgenden Zuständen.
- So geht es weiter bis zur 15. Roten. Ist also der Zustand q₁₅ erreicht, so akzeptiert der Automat das gelesene Wort.

Der so beschriebene Automat hat endlich viele Zustände und akzeptiert, wenn 15 Mal das Zeichen r gelesen wurde.

7. Teilleistungen – Kriterien / Bewertungsbogen zur Prüfungsarbeit

Name des Prüflings: _____ Kursbezeichnung: _____

Schule: _____

Teilaufgabe a)

	Anforderungen	Lösungsqualität			
		maximal erreichbare Punktzahl	EK²	ZK	DK
1	gibt die Menge der Zustände, die Menge der Endzustände und den Startzustand des Automaten an.	3			
2	ermittelt ein Wort, das vom Automaten akzeptiert wird und bei dessen Abarbeitung der Automat alle Zustände besucht, und erläutert den Ablauf des Trainingsspiels, das mithilfe des Wortes protokolliert wurde.	4			
3	erläutert die Bedeutung der drei genannten Zustände und die Übergänge aus den Zuständen heraus im Sachzusammenhang.	6			
Sachlich richtige Lösungsalternative zur Modelllösung: (13)					
.....					
.....					
	Summe Teilaufgabe a)	13			

Teilaufgabe b)

	Anforderungen	Lösungsqualität			
		maximal erreichbare Punktzahl	EK	ZK	DK
1	entwirft eine Grammatik, die alle Wörter erzeugt, die Spielprotokolle darstellen, bei denen nur maximal einmal die weiße Kugel eingelocht wurde.	6			
2	entscheidet durch Ableiten aus der entworfenen Grammatik, dass das Wort srswsfk zur Sprache der Grammatik gehört und dass das Wort swsfsw nicht zur Sprache der Grammatik G gehört.	4			
Sachlich richtige Lösungsalternative zur Modelllösung: (10)					
.....					
.....					
	Summe Teilaufgabe b)	10			

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

Teilaufgabe c)

	Anforderungen	Lösungsqualität			
		maximal erreichbare Punktzahl	EK	ZK	DK
1	gibt ein möglichst kurzes Wort zu einem Spielprotokoll an, bei dem Jonte insgesamt 5 Kugeln einlocht, ohne einen Fehler zu machen.	2			
2	gibt ein Wort der Länge 7 zu einem Spielprotokoll an, bei dem genau zweimal die Weiße gestoßen wurde.	2			
3	entwirft ein Zustandsübergangsdiagramm für den Automaten B.	7			
Sachlich richtige Lösungsalternative zur Modelllösung: (11)					
.....					
.....					
	Summe Teilaufgabe c)	11			

Teilaufgabe d)

	Anforderungen	Lösungsqualität			
		maximal erreichbare Punktzahl	EK	ZK	DK
1	entwirft einen Kellerautomaten, der ein Eingabewort in der beschriebenen Form erhält, und entscheidet, ob sich Jonte im direkten Vergleich der beiden untersuchten Trainingsspiele verbessert hat.	7			
2	stellt die Erkennung des Wortes dar.	4			
Sachlich richtige Lösungsalternative zur Modelllösung: (11)					
.....					
.....					
	Summe Teilaufgabe d)	11			

Teilaufgabe e)

	Anforderungen	Lösungsqualität			
		maximal erreichbare Punktzahl	EK	ZK	DK
1	beurteilt, ob Jonte recht hat.	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