Name: _____

Beispielaufgabe

Informatik, Grundkurs

Aufgabenstellung

Zur U17-Jugendeuropameisterschaft im Frauenfußball haben sich 32 Nationen angemeldet. Da nur acht Mannschaften für die Endrunde zugelassen sind, findet eine Qualifikation in acht Gruppen mit jeweils vier Mannschaften statt, von denen sich die Gruppensieger für das Turnier qualifizieren. Zur Verwaltung der Qualifikationsspiele wurde eine Datenbank mit einem Datenbankschema bestehend aus den folgenden Relationenschemata eingerichtet:

Land(LandID, Name, Gruppe)
Spiel(SpielID, ↑HeimID, ↑AuswaertsID, Datum)
Spielerin(SpielerinID, ↑LandID, Name, Rueckennummer)
erzieltTor(TorID, ↑SpielerinID, ↑SpielID, Spielminute)

Abbildung 1: Datenbankschema

Hinweis: Die Fremdschlüssel ↑`HeimID` und ↑`AuswaertsID` beziehen sich auf den Primärschlüssel `LandID` der Relation Land.

Die Organisatoren vereinbaren, dass jedes geschossene Tor sofort in die Datenbank eingetragen wird und bei Eigentoren immer eine beteiligte Spielerin der Mannschaft, der das Tor angerechnet wird, als Torschützin in der Datenbank gespeichert wird.

Die Tabellen in Anlage 1 stellen Auszüge aus Beispielrelationen zu den vier Relationenschemata dar.

a) *Geben Sie mithilfe der Beispielrelationen die Namen der Heim- und Auswärtsmannschaften an, die* `2017-04-12` *gegeneinander gespielt haben.*

Ermitteln Sie mithilfe der Beispielrelationen die Ergebnisse der beiden Spiele, die Namen der Torschützinnen und die Spielminuten, in denen die Tore erzielt wurden.

Entwickeln Sie eine SQL-Anweisung für die Abfrage, die eine alphabetisch aufsteigend sortierte Liste der Namen der Spielerinnen der englischen Mannschaft generiert.

(9 Punkte)

Name: _____

b) Auf dem angegebenen Datenbankschema sollen die folgenden Anfragen realisiert werden:

i.

```
1  SELECT Spielerin.Name, Spielerin.LandID
2  FROM erzieltTor
3    INNER JOIN Spielerin
4      ON Spielerin.SpielerinID = erzieltTor.SpielerinID
5    INNER JOIN Spiel
6      ON Spiel.SpielID = erzieltTor.SpielID
7  WHERE Spiel.SpielID = 2
```

ii.

```
1  SELECT Spielerin.Name, Land.Name,
2         COUNT(erzieltTor.SpielerinID) AS Tore
3  FROM erzieltTor, Spielerin, Land
4  WHERE erzieltTor.SpielerinID = Spielerin.SpielerinID AND
5        Spielerin.LandID = Land.LandID
6  GROUP BY Spielerin.SpielerinID
7  ORDER BY Tore DESC
```

Analysieren und erläutern Sie die SQL-Anweisungen.

Erläutern Sie die ermittelten Informationen im Sachzusammenhang.

(10 Punkte)

Um während der Qualifikationsspiele ständig einen genauen Überblick über die Spielergebnisse und den aktuellen Tabellenstand zu haben, wird eine Software entwickelt, die diese Informationen aus der Datenbank extrahiert und anzeigt. Die folgenden Abbildungen zeigen die Benutzungsoberfläche der Software.

Abbildung 2: Tabelle einer Gruppe

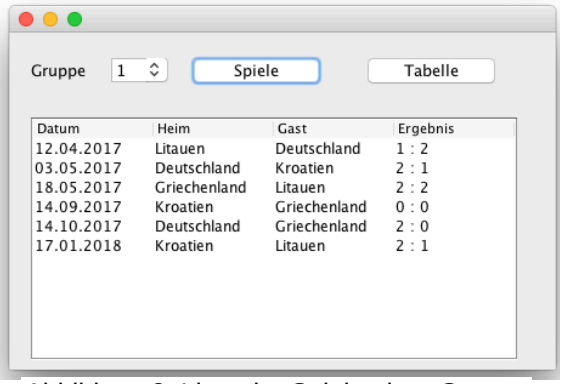


Abbildung 3: Liste der Spiele einer Gruppe

Name: _____

Die folgende Abbildung zeigt einen Ausschnitt aus dem Implementationsdiagramm der Software. Die für die Aufgabenbearbeitung nicht relevanten Methoden wurden aus Gründen der Übersichtlichkeit weggelassen.

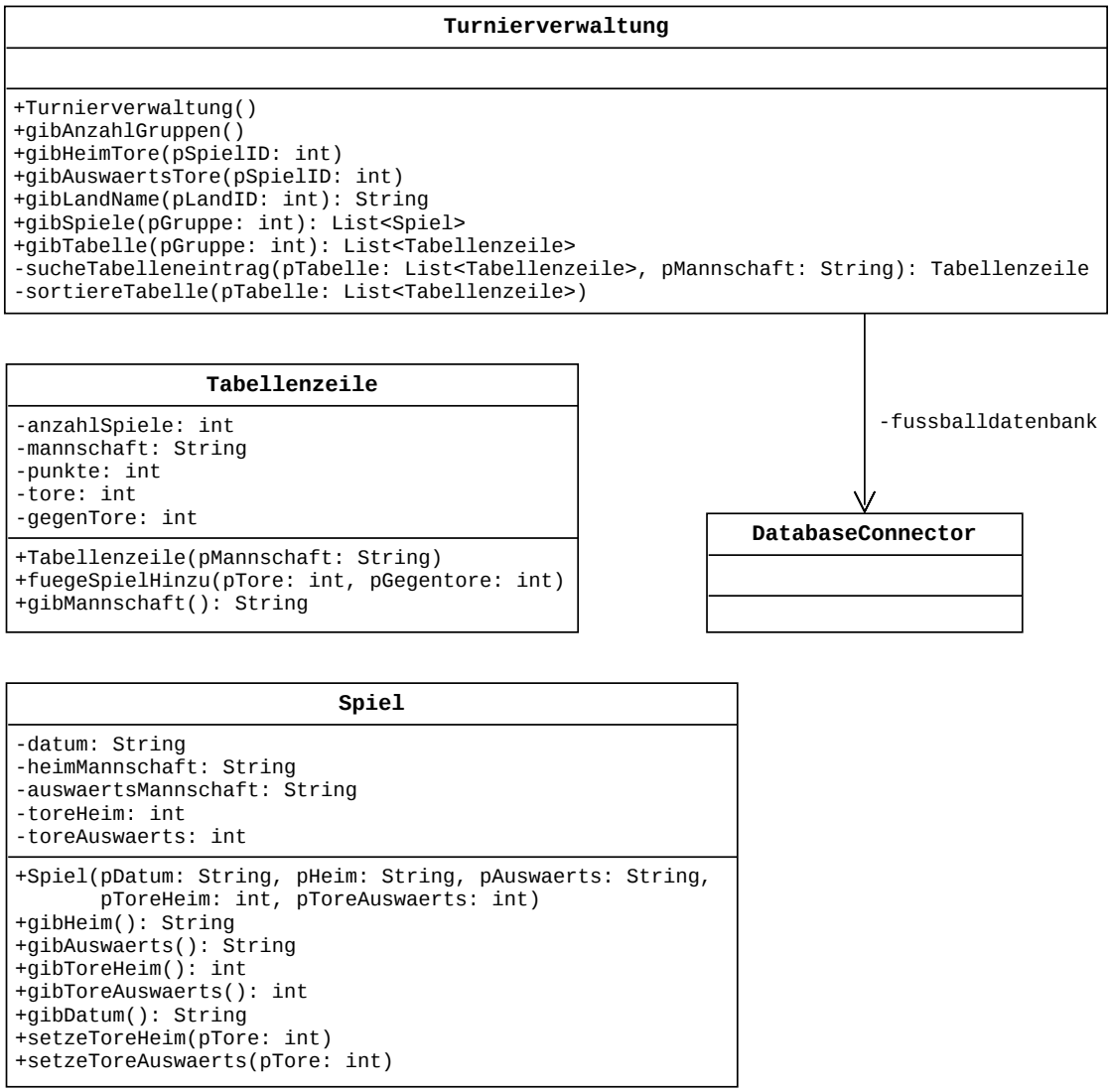


Abbildung 4: Teilmodellierung in Form eines Implementationsdiagramms

In Anlage 2 finden Sie einen Ausschnitt der Dokumentation der Klassen `Turnierverwaltung` und `Tabellenzeile`.

c) *Implementieren Sie die Methode* `sucheTabelleneintrag` *der Klasse* `Turnierverwaltung`.

(10 Punkte)

Name: _____

d) Gegeben ist der Quelltext der Methode `gibTabelle` der Klasse `Turnierverwaltung`.

```
 1  public List<Tabellenzeile> gibTabelle(int pGruppe) {
 2    List<Tabellenzeile> tabelle = new List<Tabellenzeile>();
 3    List<Spiel> spiele = gibSpiele(pGruppe);
 4    spiele.toFirst();
 5    while (spiele.hasAccess()) {
 6      Spiel spielDaten = spiele.getContent();
 9      Tabellenzeile tabzeileHeim
10          = sucheTabellenEintrag(tabelle,
11                                 spielDaten.gibHeim());
12      if (tabzeileHeim == null) {
13         tabzeileHeim = new Tabellenzeile(spielDaten.gibHeim());
14         tabelle.append(tabzeileHeim);
15      }
16      Tabellenzeile tabzeileAuswaerts
17          = sucheTabellenEintrag(tabelle,
18                                 spielDaten.gibAuswaerts());
19      if (tabzeileAuswaerts == null) {
20        tabzeileAuswaerts
21            = new Tabellenzeile(spielDaten.gibAuswaerts());
22        tabelle.append(tabzeileAuswaerts);
23      }
24      tabzeileHeim.fuegeSpielHinzu(
25          spielDaten.gibToreHeim(),
26          spielDaten.gibToreAuswaerts());
27      tabzeileAuswaerts.fuegeSpielHinzu(
28          spielDaten.gibToreAuswaerts(),
29          spielDaten.gibToreHeim());
30      spiele.next();
31    }
32    sortiereTabelle(tabelle);
33    return tabelle;
34  }
```

Analysieren und erläutern Sie die Methode `gibTabelle` *der Klasse* `Turnierverwaltung`.

(10 Punkte)

Name: _____

e) Für die Prämienvergabe benötigen die Fußballverbände die Informationen, in wie vielen Spielen die Spielerinnen eingesetzt wurden und welche Spielzeiten sie hatten. Die Datenbank muss daher erweitert werden. Die folgende Abbildung zeigt das Entity-Relationship-Modell der oben beschriebenen Datenbank ohne vollständige Angabe der Kardinalitäten.

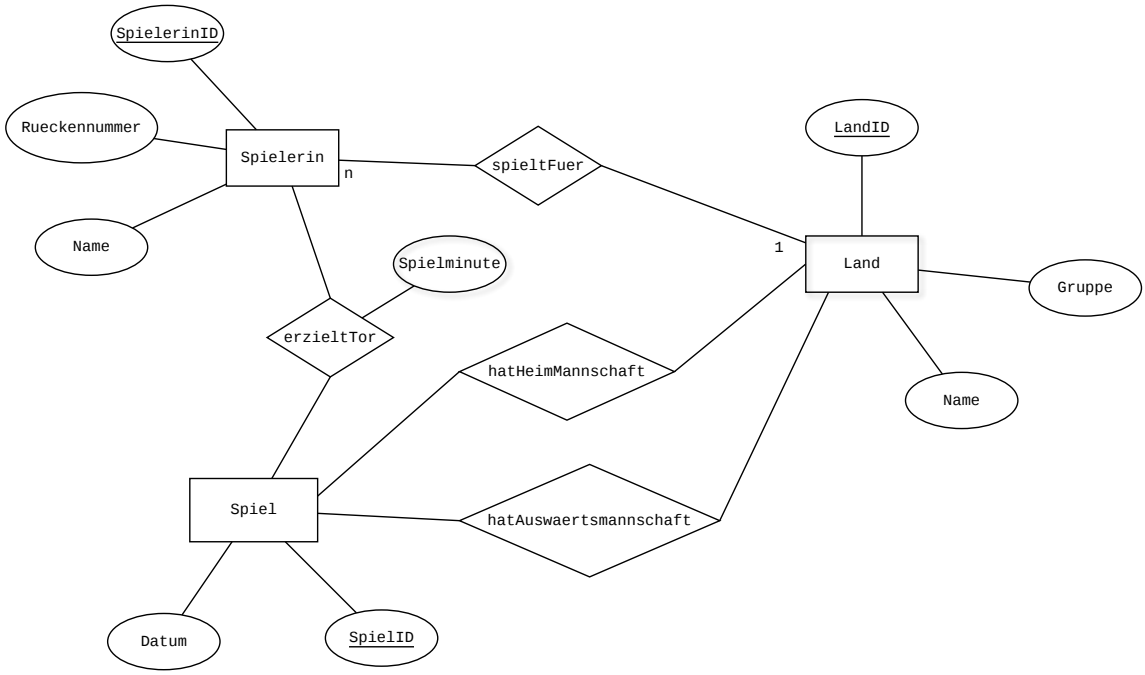


Abbildung 5: Entity-Relationship-Diagramm zur Fußballdatenbank

Erweitern Sie das Entity-Relationship-Diagramm um die fehlenden Kardinalitäten der vier gegebenen Beziehungstypen. Tragen Sie die Kardinalitäten in das als Anlage 3 angefügte Diagramm ein.

Begründen Sie, dass sich aus dem mit Kardinalitäten versehenen Entity-Relationship-Diagramm das Datenbankschema aus Abbildung 1 am Beginn der Aufgabe herleiten lässt.

Erweitern Sie das mit Kardinalitäten versehene Entity-Relationship-Diagramm so, dass die Informationen über den zeitlichen Einsatz der Spielerinnen in der Datenbank gespeichert werden.

(11 Punkte)

Zugelassene Hilfsmittel:

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

Name: _____

Anlage 1: Auszug aus der Beispieldatenbank

Land		
LandID	**Name**	**Gruppe**
1	Deutschland	1
2	England	2
3	Litauen	1
4	Bulgarien	2

Spielerin			
SpielerinID	**Name**	**Rueckennummer**	**LandID**
1	Müller	6	1
2	Heise	8	1
3	Kramer	1	1
4	Meyer	2	1
5	Jones	5	2
6	Smith	8	2
7	Hancock	9	2
8	Cool	4	2
9	Adomat	4	3
10	Butkewitz	6	3
11	Grigull	7	3
12	Matnwitz	9	3
13	Ivanova	14	4
14	Dimitrova	11	4
15	Kostadinova	3	4
16	Stoikova	0	4

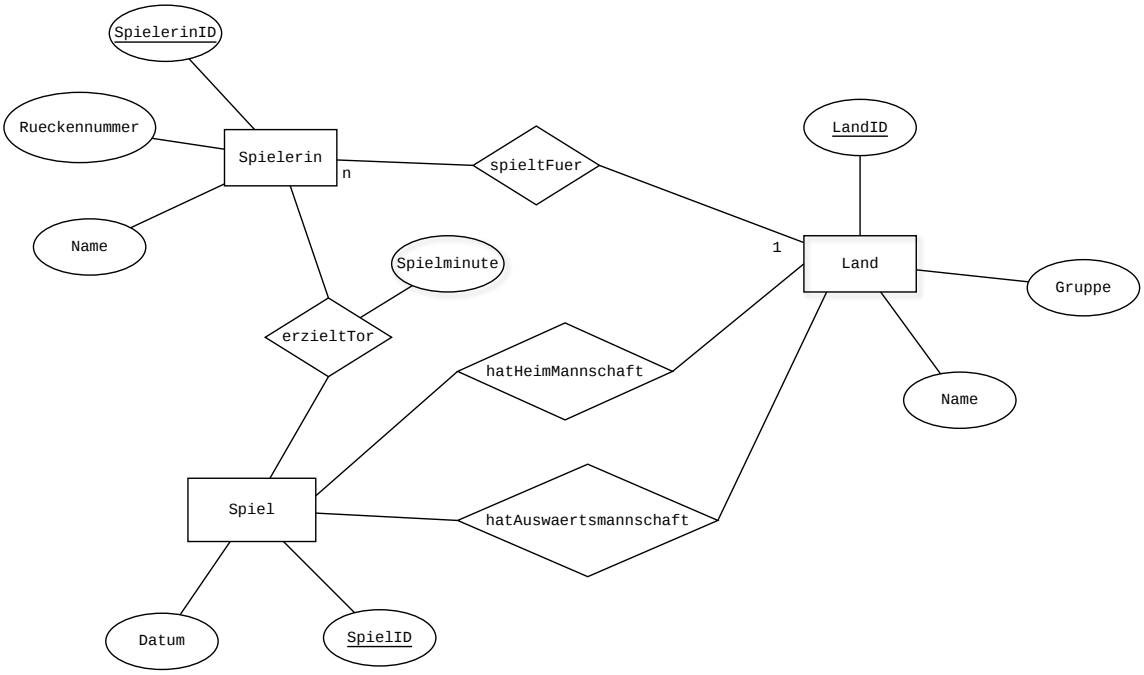
Name: _____

Spiel			
SpielID	**HeimID**	**AuswaertsID**	**Datum**
1	1	3	2017-04-12
2	2	4	2017-04-12
3	3	1	2017-09-08
4	4	2	2017-09-08

erzieltTor			
TorID	**SpielerinID**	**SpielID**	**Spielminute**
1	3	1	5
2	14	2	12
3	13	2	47
4	11	1	52
5	4	1	70
6	14	2	89

Name: _____

Anlage 2: Entity-Relationship-Diagramm zu Aufgabenteil e)



Name: _____

Anhang:

Die Klasse `Turnierverwaltung`

Ein Objekt der Klasse **`Turnierverwaltung`** liest die Daten aus der Fußballdatenbank und liefert auf Anfrage eine Liste aller Spiele einer Gruppe mit den beteiligten Mannschaften und den Ergebnissen sowie die sich aus den Ergebnissen aller durchgeführten Spiele ergebende Tabelle. Zur Reduzierung der Komplexität wird auf die Fehlerbehandlung verzichtet.

Ausschnitt aus der Dokumentation der Klasse `Turnierverwaltung`

`Turnierverwaltung()`

Der Konstruktor Initialisiert ein Objekt der Klasse und öffnet die Fußballdatenbank.

`int gibAnzahlGruppen()`

Die Methode liefert die Anzahl der Gruppen des Turniers aus der Datenbanktabelle `'Land'`.

`int gibHeimTore(int pSpielID)`

Die Methode liefert die Anzahl der erzielten Tore der Heimmannschaft im Spiel mit dem Primärschlüssel `pSpiel` der Datenbanktabelle `'Spiel'`.

`int gibAuswaertsTore(int pSpielID)`

Die Methode liefert die Anzahl der erzielten Tore der Auswärtsmannschaft im Spiel mit dem Primärschlüssel `pSpiel` der Datenbanktabelle `'Spiel'`.

`String gibLandName(int pLandID)`

Die Methode liefert den Namen des Landes, das in der Datenbanktabelle `'Land'` den Primärschlüssel `pLandID` hat.

`List<Spiel> gibSpiele(int pGruppe)`

Die Methode liefert eine nach Datum sortierte Liste aller Spiele der Gruppe `pGruppe.` Falls das Spiel bereits stattgefunden hat, werden auch die erzielten Heim- und Auswärtstore in die Listenknoten eingetragen.

`List<Tabellenzeile> gibTabelle(int pGruppe)`

Die Methode liefert eine Liste der Zeilen der Tabelle der Gruppe `pGruppe`.

Name: _____

Außerdem verfügt die Klasse **`Turnierverwaltung`** über folgende private Methoden:

```
Tabellenzeile sucheTabellenEintrag(List<Tabellenzeile> pTabelle,
                                   String pMannschaft)
```

Die Methode liefert die Tabellenzeile der Mannschaft `pMannschaft` aus der übergebenen Tabelle `pTabelle`. Wird die Mannschaft in der Tabelle nicht gefunden, wird `null` zurückgegeben.

```
void sortiereTabelle(List<Tabellenzeile> pTabelle)
```

Die Methode sortiert die Tabelle nach den Kriterien Punktzahl, Tordifferenz und erzielte Tore. Eine Rückgabe der sortierten Tabelle erübrigt sich, weil bei Objekten nur Referenzen als Parameter übergeben werden.

Die Klasse `Tabellenzeile`

Ein Objekt der Klasse **`Tabellenzeile`** repräsentiert eine Zeile einer Fußballtabelle. Es speichert den Namen der Mannschaft, die Anzahl der Spiele, die Punkte, die erzielten Tore sowie die Gegentore.

Ausschnitt aus der Dokumentation der Klasse `Tabellenzeile`

```
Tabellenzeile(String pMannschaft)
```

Der Konstruktor initialisiert ein Objekt der Klasse für die übergebene Mannschaft und weist den Attributen für Punkte, Tore und Gegentore den Wert `0` zu.

```
void fuegeSpielHinzu(int pTore, int pGegenTore)
```

Die Methode erhöht bei einem Sieg die Punktzahl um `3`, bei einem Unentschieden um `1` und aktualisiert die Werte für die Tore und die Gegentore.

```
String gibMannschaft()
```

Die Methode liefert den Namen der Mannschaft.

Name: _____

Die generische Klasse `List<ContentType>`

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

Das aktuelle Objekt kann gelesen, verändert oder gelöscht werden. Außerdem kann vor dem aktuellen Objekt ein Listenobjekt eingefügt oder ein Listenobjekt an das Ende der Liste angefügt werden.

Dokumentation der Klasse `List`

`List<ContentType>()`

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 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` gleich `null` ist, bleibt die Liste unverändert.

`void concat(List pList)`

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

`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

Beispielaufgabe

Informatik, Grundkurs

1. Aufgabenart

Analyse, Modellierung und Implementation von kontextbezogenen Problemstellungen mit Schwerpunkt auf den Inhaltsfeldern Daten und ihre Strukturierung (Objekte und Klassen, Datenbanken) und Algorithmen

2. Aufgabenstellung

siehe Prüfungsaufgabe

3. Materialgrundlage

entfällt

4. Bezüge zu dem Kernlehrplan und den Vorgaben

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
     - Implementationsdiagramme
     - Lineare Strukturen (Array, lineare Liste)
   - Datenbanken

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

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

2. Medien / Materialien

   entfällt

5. Zugelassene Hilfsmittel

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

6.1 Vorbemerkung zu dieser Beispielklausur

Das Ziel dieser Beispielaufgabe ist es, eine mögliche Verknüpfung der Inhaltsfelder „Daten und ihre Strukturierung“ (Objekte und Klassen, Datenbanken) und „Algorithmen“ für eine inhaltsfeldübergreifende Grundkursklausur zu zeigen.

Diese Beispielaufgabe orientiert sich an den Abiturvorgaben für 2021.

Ab dem Abiturjahrgang 2021 ist für eine Grundkursklausur, die aus zwei Aufgaben besteht, eine Bearbeitungszeit von 3 Stunden und 45 Minuten vorgesehen. Diese Beispielaufgabe stellt nur eine von zwei Aufgaben dar, die zusammen eine vollständige Grundkursklausur ergeben.

6.2 Modelllösungen

Die 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)

Spiele am 12.04.2017:
Deutschland (Heimmannschaft) – Litauen (Auswärtsmannschaft)
England (Heimmannschaft) – Bulgarien (Auswärtsmannschaft)

Ergebnisse, Torschützinnen und Spielminute
Deutschland – Litauen 2:1
Tore: 5. Minute 1:0 Kramer, 52. Minute 1:1 Grigull, 70. Minute 2:1 Meyer

England – Bulgarien 0:3
Tore: 12. Minute 0:1 Dimitrova, 47. Minute 0:2 Ivanova, 89. Minute 0:3 Dimitrova

SQL-Anweisung:

```
SELECT Spielerin.Name
FROM Spielerin
JOIN Land
  ON Land.LandID = Spielerin.LandID
WHERE Land.Name = 'England'
ORDER BY Spielerin.Name ASC
```

Teilaufgabe b)

i. In dieser Abfrage sind die Relationen `erzieltTor`, `Spielerin` und `Spiel` durch `JOIN` so miteinander verknüpft, dass die Ergebnisrelation nur Datensätze enthält, in denen die Attribute `SpielerinID` der Relation `erzieltTor` und `SpielerinID` der Relation `Spielerin` sowie die Attribute `SpielID` der Relation `Spiel` und `SpielID` der Relation `erzieltTor` übereinstimmen. Gesucht sind in dem Verknüpfungsergebnis die Ausprägungen der Attribute `Name` der Relation `Spielerin` und `LandID` der Relation `Spielerin`, für die das Attribut `SpielID` der Relation `Spiel` den Wert `2` hat. Die

Abfrage liefert also für das Spiel mit dem Schlüssel `2` eine Liste der Torschützinnen mit ihren Länderschlüsseln.

ii. Gesucht sind die Ausprägungen der Attribute `Name` der Relation `Spielerin` und `Name` der Relation `Land`, wobei nur Datensätze ausgewählt werden sollen, bei denen die Attribute `LandID` der Relation `Land` und `LandID` der Relation `Spielerin` übereinstimmen. Darüber hinaus enthält die Ergebnisrelation eine Spalte `Tore` mit der Anzahl der Datensätze der Relation `erzieltTor`, gruppiert nach dem Attribut `SpielerinID`, das übersteinstimmt mit dem Attribut `SpielerinID` der Relation `Spielerin`. Die Daten sind nach der Ergebnisspalte `Tore` absteigend sortiert. Es handelt sich um eine Auflistung aller Spielerinnen mit ihren Ländern, die mindestens ein Tor erzielt haben und der Zahl, der von ihnen erzielten Tore, absteigend sortiert nach der Zahl der Tore. Kurz gesagt, eine Torjägerinnenliste.

Teilaufgabe c)

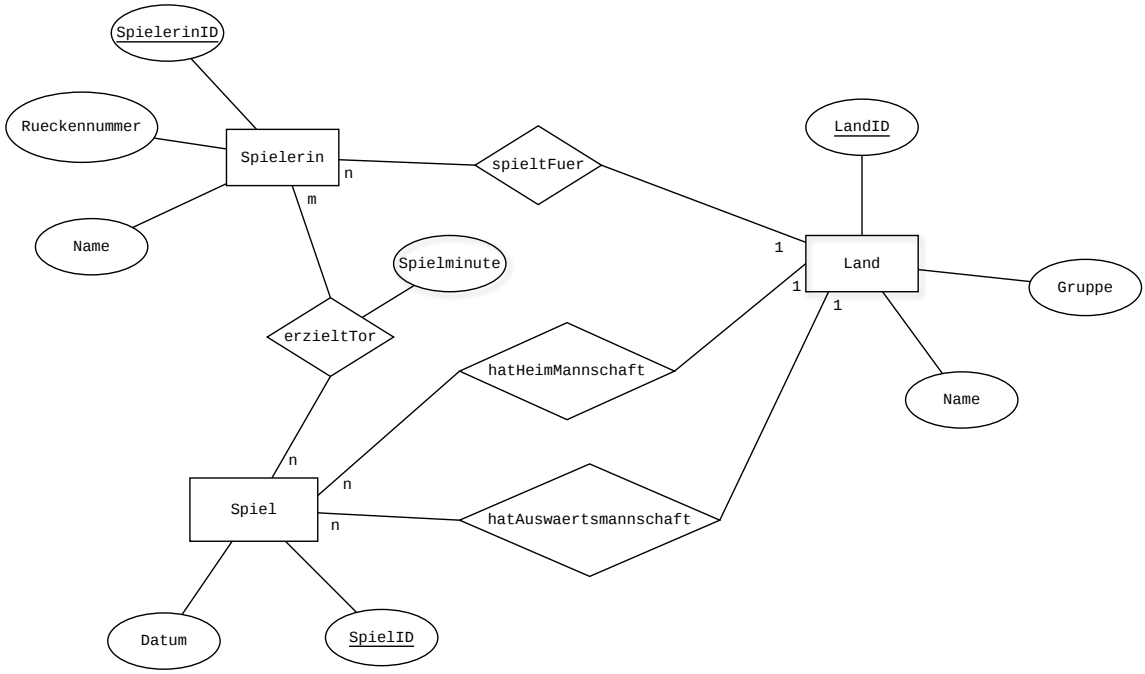
```
private Tabellenzeile sucheTabellenEintrag(
    List<Tabellenzeile> pTabelle, String pMannschaft) {
  pTabelle.toFirst();
  boolean gefunden = false;
  Tabellenzeile tabZeile = null;
  while (!gefunden && pTabelle.hasAccess()) {
    tabZeile = pTabelle.getContent();
    if (pMannschaft.equals(tabZeile.gibMannschaft())) {
      gefunden = true;
    } else {
      pTabelle.next();
    }
  }
  if (!gefunden) {
    tabZeile = null;
  }
  return tabZeile;
}
```

Teilaufgabe d)

Die Methode berechnet und liefert die aktuelle Tabelle des Fußballturniers für die als Parameter übergebene Gruppe unter Berücksichtigung aller in die Datenbank eingetragenen Spiele.
Die Tabelle wird als lineare Liste mit Objekten der Klasse `Tabellenzeile` erstellt. Nachdem eine leere Liste erzeugt wurde, erhält man durch Aufruf der Methode `gibSpiele` eine Liste mit allen Spielen der Gruppe. Elemente dieser Liste sind Objekte der Klasse `Spiel`, die als Attribute die Namen der beiden Mannschaften, das Datum des Spiels sowie die von beiden Mannschaften erzielten Tore haben. Jedes Element der Spielliste wird in einer `while`-Schleife verarbeitet. Wenn die Mannschaften eines Spiels noch nicht in der Tabellenliste enthalten sind, wird ein neues Objekt der Klasse `Tabellenzeile` an die Liste angehängt. Für beide beteiligten Mannschaften eines Spiels wird die Methode `fuegeSpielHinzu` der Klasse `Tabellenzeile` aufgerufen, mit den von der Mannschaft erzielten Toren und den erhaltenen Gegentoren als Parameter. Wenn alle Spiele verarbeitet sind, wird die Tabelle durch Aufruf der Methode `sortiereTabelle` entsprechend der erreichten Punktzahl, der Tordifferenz und der Anzahl erzielter Tore absteigend sortiert.

Teilaufgabe e)

Entity-Relationship-Modell mit Kardinalitäten.



Aus dem ER-Modell lassen sich folgende Relationen-Schemata herleiten:

Land(LandID, Name, Gruppe)
Spiel(SpielID, ↑HeimID, ↑AuswaertsID, Datum)
Spielerin(SpielerinID, ↑LandID, Name, Rueckennummer)
erzieltTor(TorID, ↑SpielerinID, ↑SpielID, Spielminute)

Begründung:
Für jeden Entitätstyp des Entity-Relationship-Diagramms wird ein Relationenschema mit demselben Namen und allen Attributen des Entitätstyps angelegt.

Dem Relationenschema `Land` werden daher die Attribute `LandID` als Primärschlüssel sowie `Name` und `Gruppe` hinzugefügt

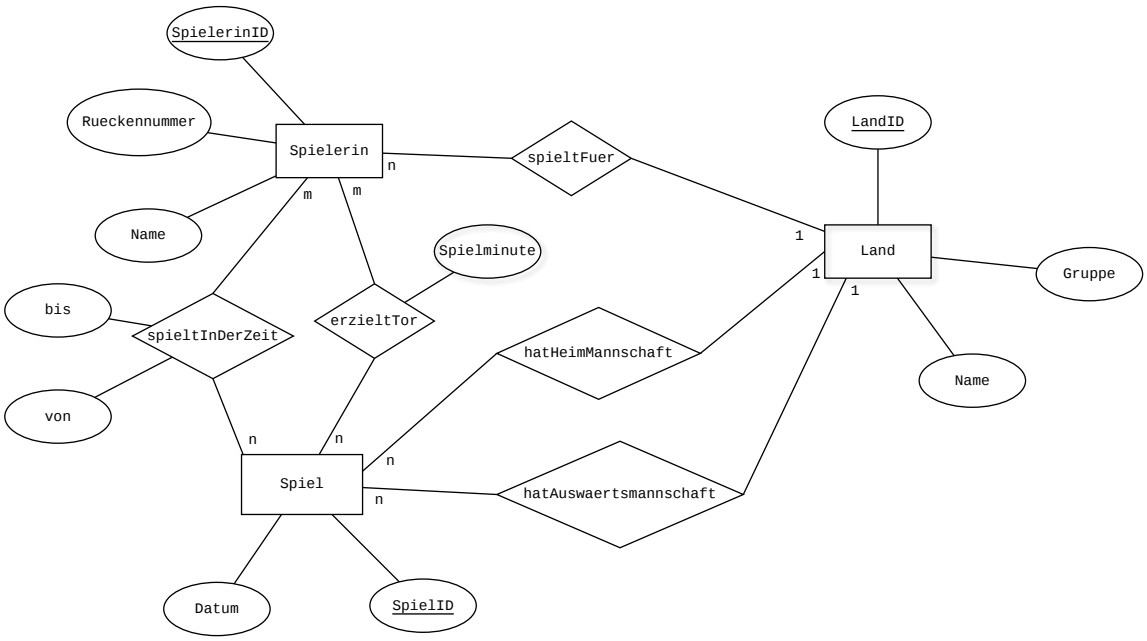
Das Relationenschema `Spiel` erhält die Attribute `SpielID` als Primärschlüssel und `Datum`. Die Zuordnung der Heimmannschaft und der Auswärtsmannschaft eines Spiels zu einem Land wurde durch die beiden 1:n-Beziehungstyp `hatHeimMannschaft` und `hatAuswaertsMannschaft` modelliert. Daher werden der Schlüssel des Entitätstyps `Land` dem Relationenschema `Spiel` einmal als Fremdschlüssel-Attribut mit der Bezeichnungen `HeimID` und einmal als Fremdschlüssel-Attribut mit der Bezeichnung `AuswaertsID` hinzugefügt.

Das Relationenschema `Spielerin` erhält die Attribute `SpielerinID` als Primärschlüssel sowie `Name` und `Rueckennummer`. Jeder Spielerin wird über den 1:n-Beziehungtyp `erzieltTor` eindeutig ein Land zugeordnet. Daher wird der Schlüssel des Entitätstyps `Land` dem Relationenschema `Spielerin` als Fremdschlüssel hinzugefügt.

Da es sich bei dem Beziehungstyp `erzieltTor` zwischen den Entitätstypen `Spiel` und

`Spielerin` um die Darstellung eines n:m-Beziehungstyps handelt, muss ein weiteres Relationenschema modelliert werden. Die Schlüssel der Entitätstypen, zwischen denen die Beziehung besteht, werden als Fremdschlüssel hinzugefügt; hinzu kommt das vorgeschriebene Attribut `Spielminute`. Da eine Spielerin in einem Spiel und einer Spielminute mehrere Tore erzielen könnte, ist die Kombination der beiden Fremdschlüssel nicht eindeutig. Somit muss ein zusätzliches Attribut als Primärschlüssel hinzugefügt werden.

Die folgende Abbildung zeigt eine Möglichkeit der Erweiterung des Entity-Relationship-Diagramms um den n:m-Beziehungstyp `spieltInDerZeit`, mit der die Spielzeiten der Spielerinnen in der Datenbank erfasst werden können.



7. Teilleistungen – Kriterien / Bewertungsbogen zur Prüfungsarbeit

Name des Prüflings: _____ Kursbezeichnung: _____

Schule: _____

Teilaufgabe a)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl (AFB)**	**EK[1]**	**ZK**	**DK**
1	gibt die Namen der Heim- und Auswärtsspielmannschaften an, die am 12.04.2017 gegeneinander gespielt haben.	2 (I)			
2	ermittelt die Ergebnisse, die Torschützinnen und die Spielminuten.	3 (I)			
3	entwickelt eine SQL-Anweisung für die Abfrage, die eine alphabetisch aufsteigend sortierte Liste der Namen der Spielerinnen der englischen Mannschaft generiert.	4 (II)			
Sachlich richtige Lösungsalternative zur Modelllösung: (9) …….. ……..					
	Summe Teilaufgabe a)	9			

Teilaufgabe b)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl (AFB)**	**EK**	**ZK**	**DK**
1	analysiert und erläutert die SQL-Anweisung i. und erläutert die ermittelten Informationen im Sachzusammenhang.	4 (II)			
2	analysiert und erläutert die SQL-Anweisung ii. und erläutert die ermittelten Informationen im Sachzusammenhang.	6 (II)			
Sachlich richtige Lösungsalternative zur Modelllösung: (10) …….. ……..					
	Summe Teilaufgabe b)	10			

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

Teilaufgabe c)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl (AFB)**	**EK**	**ZK**	**DK**
1	implementiert die Methode sucheTabelleneintrag.	10 (II)			
Sachlich richtige Lösungsalternative zur Modelllösung: (10) …….. ……..					
	Summe Teilaufgabe c)	10			

Teilaufgabe d)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl (AFB)**	**EK**	**ZK**	**DK**
1	analysiert und erläutert die Methode gibTabelle.	10 (II)			
Sachlich richtige Lösungsalternative zur Modelllösung: (10) …….. ……..					
	Summe Teilaufgabe d)	10			

Teilaufgabe e)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl (AFB)**	**EK**	**ZK**	**DK**
1	erweitert das ER-Diagramm um die Kardinalitäten.	3 (II)			
2	begründet die Herleitung des Datenbankschemas.	4 (II)			
3	entwickelt ein erweitertes ER-Diagramm.	4 (III)			
Sachlich richtige Lösungsalternative zur Modelllösung: (11) …….. ……..					
	Summe Teilaufgabe e)	11			