Name: _____

Beispielaufgabe

Informatik, Leistungskurs

Aufgabenstellung

Als Beitrag zum Aktionstag gegen Jugendgewalt möchte die Von-Neumann-Gesamtschule ein Sportfest unter dem Motto „Laufen statt Raufen“ organisieren. Alle Schülerinnen und Schüler, Lehrkräfte und Eltern können nach vorheriger Anmeldung an mehreren Stationen an Laufwettbewerben teilnehmen.
Jede Station ist mit einer Wettkampfanzeige in Form eines Großbildschirms und einem dazugehörigen Rechner ausgestattet. Über dieses System sollen die erzielten Zeiten der Teilnehmenden eingegeben, in eine zentrale Datenbank überspielt und an der jeweiligen Station zusammen mit dem Namen der jeweiligen Person als Bestenliste angezeigt werden. Die aktive Teilnahme am Aktionstag und damit die Anmeldung zu den Wettbewerben erfolgt auf freiwilliger Basis.

Die Modellierung der zentralen Datenbank entspricht dem folgenden Datenbankschema. Beispieldatensätze zu diesem Schema sind im Anhang zu finden.

Teilnehmer(TeilnehmerID, Vorname, Nachname, ↑GruppeID)

Gruppe(GruppeID, Bezeichnung)

Wettkampf(WettkampfID, Bezeichnung)

nimmtTeil(↑TeilnehmerID, ↑WettkampfID, Ergebniszeit)

In einer Relation zum Relationenschema `Teilnehmer` werden alle Personen modelliert, die sich für die Teilnahme an einem Wettkampf angemeldet haben. Die angebotenen Wettkämpfe werden im System in einer Relation zum Schema `Wettkampf` modelliert. Das Relationenschema `Gruppe` definiert Datensätze, die Personengruppen darstellen, zu denen die Teilnehmerinnen und Teilnehmer zugeordnet werden. In der im System verwendeten Relation zum Relationenschema `nimmtTeil` wird eingetragen, welche Teilnehmerin bzw. welcher Teilnehmer für welchen Wettkampf angemeldet ist und in welcher Zeit er oder sie diesen Wettkampf abgeschlossen hat. Als Zeit ist bei der Anmeldung zunächst eine `0` eingetragen. Nach einem Wettkampf wird die korrekte Zeit nachgetragen. Die Zeitangaben erfolgen in Millisekunden.

Name: _____

a) *Überführen Sie das obige Datenbankschema in ein Entity-Relationship-Diagramm.*

Ermitteln Sie auf Grundlage der Beispieldatensätze im Anhang, welche Teilnehmerinnen und Teilnehmer für den `Abenteuerparcours` *angemeldet sind und zu welcher Personengruppe sie jeweils gehören. Erläutern Sie, wie diese Information aus den Beispieldatensätzen abzulesen ist.*

(10 Punkte)

Für die Rechner der einzelnen Wettkampfstationen soll ein Programm entwickelt werden, das folgender Teilmodellierung entspricht:

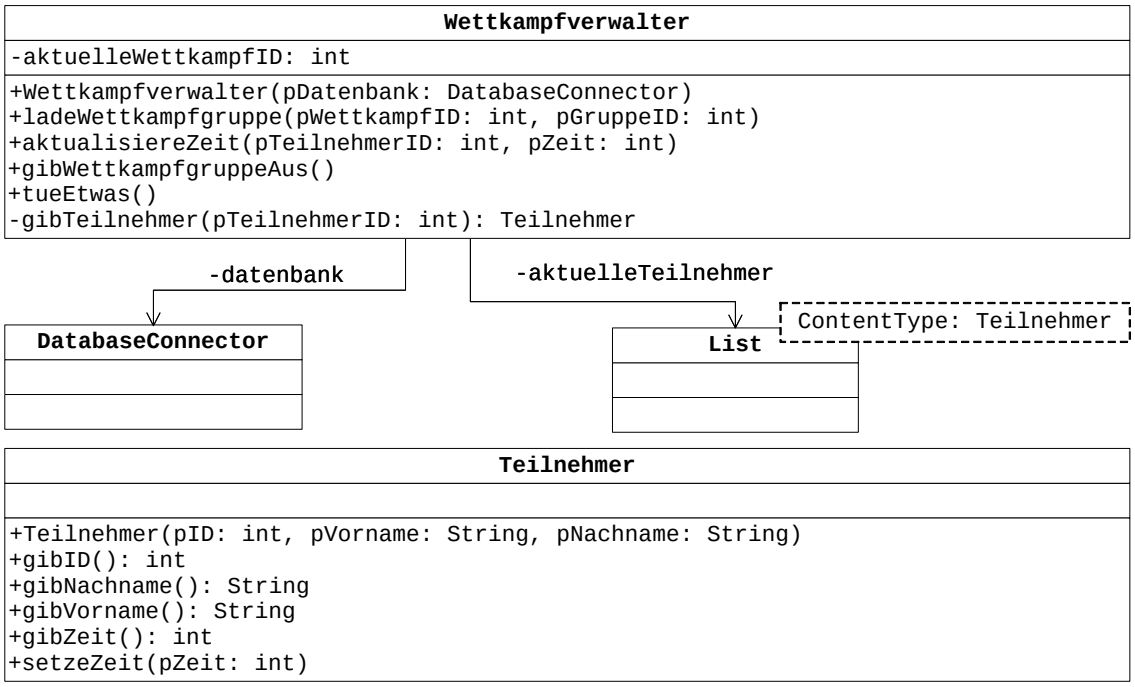


Abbildung 1: Teilmodellierung des Programms für eine Wettkampfstation

Am Tag des Sportfests sollen Teilnehmende derselben Gruppe, die sich für einen bestimmten Wettkampf angemeldet haben, diesen gemeinsam bestreiten.
Kommt z. B. die Gruppe `Klasse 7b` zum Wettkampf `Sprint`, so werden auf dem Rechner der Station mit Hilfe eines Objekts der Klasse `Wettkampfverwalter` unter Verwendung eines Objekts der Klasse `DatabaseConnector` alle entsprechend angemeldeten Teilnehmerinnen und Teilnehmer dieser Gruppe aus der Datenbank abgefragt. Sie werden dann als Objekte vom Typ `Teilnehmer` repräsentiert und in einer Liste vom Typ `List` vorgehalten.

Name: _____

Absolvieren Teilnehmerinnen und Teilnehmer dann den Wettkampf mit einer Ergebniszeit, so wird diese Zeit in dem entsprechenden Objekt gespeichert, so dass sie später in die Datenbank überspielt werden kann. Ist eine Teilnehmerin oder ein Teilnehmer mit ihrer oder seiner Zeit nicht zufrieden, kann er oder sie den Wettkampf wiederholen. Gespeichert wird immer nur die beste Zeit.

Alle Teilnehmerinnen und Teilnehmer werden anhand ihrer ID identifiziert, die sie als Startnummer auf dem Rücken tragen.

b) Die Methode `ladeWettkampfgruppe` der Klasse `Wettkampfverwalter` soll alle Teilnehmerinnen und Teilnehmer der Gruppe mit der ID `pGruppeID`, die zum Wettkampf mit der ID `pWettkampfID` angemeldet sind, aus der Datenbank laden und entsprechende Objekte vom Typ `Teilnehmer` in die neu zu erstellende Liste `aktuelleTeilnehmer` eintragen. Der Wert von `pWettkampfID` wird im Attribut `aktuelleWettkampfID` des an der aktuellen Station genutzten Objekts der Klasse `Wettkampfverwalter` gespeichert, da er später noch benötigt wird.

   Die Methode `ladeWettkampfgruppe` hat den folgenden Methodenkopf:

```
public void ladeWettkampfgruppe(int pWettkampfID, int pGruppeID)
```

   Erläutern Sie, wie mit Hilfe einer SQL-Anweisung die für die Realisierung der Methode `ladeWettkampfgruppe` *erforderlichen Daten aus der Datenbank ermittelt werden können. Gehen Sie dabei insbesondere darauf ein, wie die Verknüpfung der Datensätze realisiert wird.*

   Implementieren Sie die Methode `ladeWettkampfgruppe` *mit der von Ihnen entwickelten SQL-Anweisung.*

   (10 Punkte)

Da geplant ist, auf der Wettkampfanzeige eine Bestenliste der aktuellen Teilnehmerinnen und Teilnehmer anzuzeigen, soll dafür gesorgt werden, dass Teilnehmerinnen und Teilnehmer mit eingetragener Ergebniszeit immer am Anfang der Liste `aktuelleTeilnehmer` stehen und dabei in aufsteigender Reihenfolge nach ihren Ergebniszeiten sortiert sind.

Name: _____

c) Sind die Teilnehmerinnen und Teilnehmer aus der Datenbank geladen, können Ergebniszeiten eingetragen werden. Die Methode `aktualisiereZeit` der Klasse `Wettkampfverwalter` trägt die Ergebniszeit `pZeit` im Objekt vom Typ `Teilnehmer` mit der ID `pTeilnehmerID` ein, sofern dort noch keine Ergebniszeit gespeichert ist oder die neue Ergebniszeit besser ist als die dort eingetragene. Gegebenenfalls muss die Position der Teilnehmerin oder des Teilnehmers in der Liste `aktuelleTeilnehmer` angepasst werden, so dass die Liste gemäß der obigen Vorgaben sortiert bleibt.

Die Methode `aktualisiereZeit` hat den folgenden Methodenkopf:

```
public void aktualisiereZeit(int pTeilnehmerID, int pZeit)
```

Entwerfen Sie einen Algorithmus, der die Zeit eines Teilnehmers aktualisiert.

Implementieren Sie die Methode `aktualisiereZeit` *entsprechend Ihrem Algorithmus.*
(12 Punkte)

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

```
1  public void tueEtwas() {
2    aktuelleTeilnehmer.toFirst();
3    while (aktuelleTeilnehmer.hasAccess()
         && aktuelleTeilnehmer.getContent().gibZeit() != 0) {
4      Teilnehmer akt = aktuelleTeilnehmer.getContent();
5      String s =
          "UPDATE nimmtTeil SET Ergebniszeit = "
       +     akt.gibZeit()
       + " WHERE TeilnehmerID = " + akt.gibID()
       + "   AND WettkampfID  = " + aktuelleWettkampfID;
6      datenbank.executeStatement(s);
7      aktuelleTeilnehmer.next();
8    }
9  }
```

Analysieren Sie die Methode `tueEtwas` *und erläutern Sie den Algorithmus, nach dem die Methode arbeitet. Gehen Sie dabei insbesondere auf die SQL-Anweisung in Zeile 5 ein.*

Erläutern Sie, welche Aufgabe die Methode `tueEtwas` *im Sachzusammenhang erfüllt.*
(8 Punkte)

Name: _____

e) Obwohl von Anfang an bekannt war, dass die Ergebnisse der Wettkämpfe auf Großleinwänden präsentiert werden sollen, kommt es kurz vor der Veranstaltung zu vereinzelten Beschwerden von Teilnehmerinnen und Teilnehmern, die nicht möchten, dass ihr Name auf den Anzeigetafeln erscheint. Sie argumentieren, dass es sich dabei um einen Verstoß gegen den Datenschutz handele.

Nehmen Sie Stellung zu der Behauptung, die Anzeigetafeln seien aus der Perspektive des Datenschutzes bedenklich, wenn sie wie geplant eingesetzt werden.

Erläutern Sie unabhängig von Ihrer vorherigen Argumentation, wie dem Wunsch mancher Teilnehmerinnen und Teilnehmer nach Anonymität entsprochen werden könnte und welche hieraus resultierenden Änderungen im Datenbankschema und dem Programm (vgl. Abbildung 1) vorgenommen werden müssten.

(10 Punkte)

Zugelassene Hilfsmittel:

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

Name: _____

Anlage: Beispieldaten aus der zentralen Wettkampfdatenbank

Teilnehmer

TeilnehmerID	Vorname	Nachname	GruppeID
1	Klaudia	Schmidt	2
2	Burak	Yildirim	1
3	Ivonne	Herrmann	3
4	Oliver Paul	Theis	5
5	Thomas	Nagel	5
...	...	...	...

Gruppe

GruppeID	Bezeichnung
1	Klasse 5a
2	Klasse 7d
3	Eltern
4	Klasse 6b
5	Lehrkräfte
...	...

Wettkampf

WettkampfID	Bezeichnung
1	Hürdenlauf
2	Slalomlauf
3	Sprint
4	Abenteuerpar-cours
...	...

nimmtTeil

TeilnehmerID	WettkampfID	Ergebniszeit
2	4	0
3	4	0
5	2	24710
1	3	0
...	...	...

Name: _____

Anhang:

Dokumentation der Klasse `Wettkampfverwalter`

Die Klasse **`Wettkampfverwalter`** ermöglicht es, Teilnehmerinnen und Teilnehmer an einem Wettkampf aus der Datenbank zu laden und zu verwalten.

`Wettkampfverwalter(DatabaseConnector pDatenbank)`

Erstellt ein Objekt der Klasse `Wettkampfverwalter` und speichert die übergebene Verbindung zur Datenbank des Sportfests.

`void ladeWettkampfgruppe(int pWettkampfID, int pGruppeID)`

Die Methode lädt aus der verbundenen Datenbank alle Teilnehmerinnen und Teilnehmer der Gruppe mit der ID `pGruppeID`, die für den Wettkampf mit der ID `pWettkampfID` angemeldet sind, und speichert sie als Objekte vom Typ `Teilnehmer` in der Liste `aktuelleTeilnehmer`. Der Inhalt des Parameters `pWettkampfID` wird im Attribut `aktuelleWettkampfID` gespeichert.

`void aktualisiereZeit(int pTeilnehmerID, int pZeit)`

Die Methode trägt die Zeit `pZeit` bei der in der lokalen Liste gespeicherten Teilnehmerin bzw. dem in der lokalen Liste gespeicherten Teilnehmer mit der ID `pTeilnehmerID` ein, sofern noch keine Zeit oder eine längere Zeit eingetragen ist. Der Wert für `pZeit` muss größer als `0` sein.
Nach Aufruf der Methode ist die Sortierung der Liste so, dass am Anfang alle Teilnehmerinnen und Teilnehmer mit einem von `0` verschiedenen Zeiteintrag stehen und nach diesem Zeiteintrag aufsteigend sortiert sind. Im Anschluss stehen ohne weitere Sortierung alle Teilnehmerinnen und Teilnehmer, deren Zeiteintrag den Wert `0` hat.

`void gibWettkampfgruppeAus()`

Die Methode gibt alle in der lokalen Liste gespeicherten Teilnehmerinnen und Teilnehmer entsprechend ihrer Reihenfolge in der Liste mit ID, Nachnamen, Vornamen und Ergebniszeit auf der Wettkampfanzeige aus.

`void tueEtwas()`

Die Methode soll in Teilaufgabe d) analysiert werden und wird daher hier nicht dokumentiert.

Zusätzlich verfügt die Klasse `Wettkampfverwalter` über die folgende private Methode:

`Teilnehmer gibTeilnehmer(int pTeilnehmerID)`

Die Methode liefert aus der aktuell geladenen Wettkampfgruppe das Teilnehmerobjekt mit der ID `pTeilnehmerID`. Gibt es in der aktuell geladenen Wettkampfgruppe kein entsprechendes Objekt, wird `null` geliefert.

Name: _____

Dokumentation der Klasse `Teilnehmer`

Die Klasse **`Teilnehmer`** speichert die Daten einer Teilnehmerin bzw. eines Teilnehmers.

`Teilnehmer(int pTeilnehmerID, String pVorname, String pNachname)`

Erstellt ein Objekt der Klasse `Teilnehmer` und speichert die ID `pTeilnehmerID`, den Vornamen `pVorname` und den Nachnamen `pNachname`. Die aktuelle Ergebniszeit wird mit dem Wert `0` initialisiert.

`int gibID()`

Die Methode liefert die ID der Teilnehmerin bzw. des Teilnehmers als Integer.

`String gibNachname()`

Die Methode liefert den Nachnamen der Teilnehmerin bzw. des Teilnehmers als String.

`String gibVorname()`

Die Methode liefert den Vornamen der Teilnehmerin bzw. des Teilnehmers als String.

`int gibZeit()`

Die Methode liefert die aktuelle Ergebniszeit der Teilnehmerin bzw. des Teilnehmers als Integer in Millisekunden.

`void setzeZeit(int pZeit)`

Die Methode setzt die Ergebniszeit der Teilnehmerin bzw. des Teilnehmers auf den Wert von `pZeit`.

Name: _____

Die Klasse DatabaseConnector

Ein Objekt der Klasse **`DatabaseConnector`** ermöglicht die Abfrage und Manipulation einer relationalen Datenbank.
Beim Erzeugen des Objekts wird eine Datenbankverbindung aufgebaut, so dass anschließend SQL-Anweisungen an diese Datenbank gerichtet werden können.

Dokumentation der Klasse `DatabaseConnector`

```
DatabaseConnector(String pIP, int 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 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`.

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, Leistungskurs

1. Aufgabenart

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

2. Aufgabenstellung

siehe Prüfungsaufgabe

3. Materialgrundlage

entfällt

4. Bezüge zu dem Kernlehrplan und zu den Vorgaben[1]

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 (lineare Strukturen)
   - Datenbanken

   Algorithmen:
   - Analyse, Entwurf und Implementierung von Algorithmen

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

   Informatik, Mensch und Gesellschaft:
   - Wirkungen der Automatisierung (Grundprinzipien des Datenschutzes)

2. Medien / Materialien
   – keine –

5. Zugelassene Hilfsmittel

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

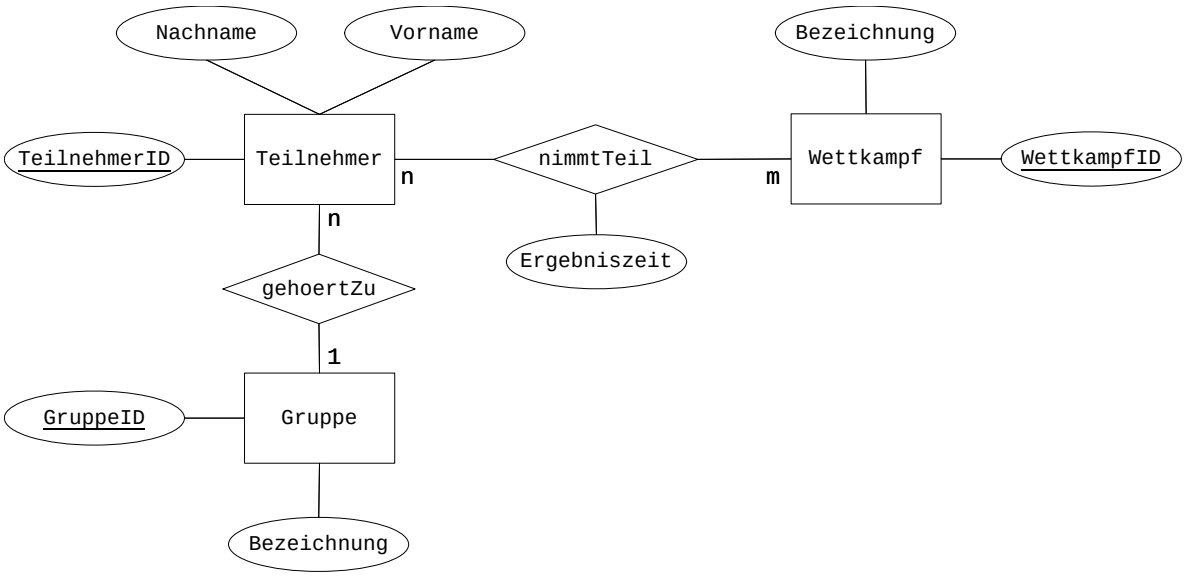
[1] Die vorliegende Beispielaufgabe setzt voraus, dass die SQL-Befehle Update und Insert in die Zentralabiturvorgaben aufgenommen bzw. im Anhang der Aufgabe erklärt werden. Des Weiteren wird vorausgesetzt, dass Grundprinzipien des Datenschutz in den Vorgaben stehen. Dazu gehören z.B. folgende: Verbot mit Erlaubnisvorbehalt, Datensparsamkeit und Datenvermeidung, Erforderlichkeit, Zweckbindung.

6. 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)

Das folgende Entity-Relationship-Diagramm entspricht dem gegebenen Datenbankschema:



Für den Wettkampf `Abenteuerparcours` sind `Burak Yildirim` aus der Gruppe `Klasse 5a` und `Ivonne Herrmann` aus der Gruppe `Elter`“ angemeldet. Das ist wie folgt aus den Datensätzen im Anhang abzulesen:

In der angegebenen Tabelle `Wettkampf` ist für den Wettkampf mit der Bezeichnung `Abenteuerparcours` unter dem Primärschlüssel `WettkampfID` der Wert `4` ausgewiesen.
In der angegebenen Tabelle `nimmtTeil` werden den Einträgen, die im Fremdschlüsselattribut `WettkampfID` den Wert `4` haben, die Werte `2` und `3` im Fremdschlüsselattribut `TeilnehmerID` zugeordnet. Die Personen mit den Werten `2` und `3` im Primärschlüsselattribut `TeilnehmerID` der angegebenen Tabelle `Teilnehmer` sind also zum Wettkampf `Abenteuerparcours` angemeldet.
In der angegebenen Tabelle `Teilnehmer` ist im Primärschlüsselattribut `TeilnehmerID` unter den Werten `2` und `3` abzulesen, dass es sich dabei um `Burak Yildirim` und `Ivonne Herrmann` handelt. Im Fremdschlüsselattribut `GruppeID` hat `Burak Yildirim` den Wert `1` und `Ivonne Herrmann` den Wert `3` eingetragen.
In der angegebenen Tabelle `Gruppe` ist nun abschließend abzulesen, dass der Wert `1` im Primärschlüsselattribut `GruppeID` zur Gruppe `Klasse 5a` und der Wert `3` zur Gruppe `Eltern` gehört.

Teilaufgabe b)

Um die Methode `ladeWettkampfgruppe` zu realisieren, müssen Werte für `TeilnehmerID`, `Vorname` und `Nachname` von bestimmten Teilnehmerinnen und Teilnehmern aus der Datenbank abgefragt werden. Diese Werte sind zwar alle in der Tabelle `Teilnehmer` gespeichert, da aber nur Werte von Personen ermittelt werden sollen, die zu einem bestimmten Wettkampf und einer bestimmten Gruppe gehören, müssen mehrere Tabellen der Datenbank berücksichtigt werden.
Eine entsprechende SQL-Anweisung muss dem Datenbankschema folgend daher jeweils einen *inner join* zwischen den Tabellen `Teilnehmer` und `Gruppe`, `Teilnehmer` und `nimmtTeil` sowie `nimmtTeil` und `Wettkampf` umsetzen.
Durch eine Selektion wird die Ergebnismenge anschließend auf die Einträge beschränkt, die der Gruppe mit der ID `pGruppeID` und dem Wettkampf mit der ID `pWettkampfID` zugeordnet sind. Durch eine Projektion werden die den geforderten Attributen entsprechenden Spalten in die Ergebnistabelle aufgenommen.

Die folgende Implementierung der Methode `ladeWettkampfgruppe` entspricht den Anforderungen:

```
public void ladeWettkampfgruppe(int pWettkampfID, int pGruppeID) {
  // Neue leere Liste anlegen.
  aktuelleTeilnehmer = new List<Teilnehmer>();

  // Wert von pWettkampfID in aktuelleWettkampfID speichern.
  aktuelleWettkampfID = pWettkampfID;

  // SQL-Anweisung wird als String vorbereitet.
  String sql
    = "SELECT Teilnehmer.TeilnehmerID, Teilnehmer.Vorname, "
    + "       Teilnehmer.Nachname "
    + "FROM Teilnehmer "
    + "  INNER JOIN nimmtTeil "
    + "    ON Teilnehmer.TeilnehmerID = nimmtTeil.TeilnehmerID "
    + "  INNER JOIN Wettkampf "
    + "    ON nimmtTeil.WettkampfID = Wettkampf.WettkampfID "
    + "  INNER JOIN Gruppe " +
    + "    ON Teilnehmer.GruppeID = Gruppe.GruppeID "
    + "WHERE Gruppe.GruppeID = " + pGruppeID + " AND "
    + "      Wettkampf.WettkampfID = "+ pWettkampfID;

  /* SQL-Anweisung wird an die Datenbank uebermittelt.
   * Das Ergebnis wird in sqlErgebnis gespeichert. */
  datenbank.executeStatement(sql);
  QueryResult sqlErgebnis = datenbank.getCurrentQueryResult();

  /* Datensaetze der Ergebnistabelle werden durchlaufen.
   * Fuer jeden Datensatz wird ein entsprechendes Objekt
   * vom Typ Teilnehmer erzeugt und in die Liste eingefuegt. */
  for (int i = 0; i < sqlErgebnis.getRowCount(); i++) {
    String[] aktZeile = sqlErgebnis.getData()[i];
    Teilnehmer aktTeilnehmer = new Teilnehmer(
        Integer.parseInt(aktZeile[0]), aktZeile[1], aktZeile[2]);
    aktuelleTeilnehmer.append(aktTeilnehmer);
  }
}
```

Teilaufgabe c)

Um die Methode `aktualisiereZeit` zu realisieren, muss das Objekt vom Typ `Teilnehmer` gesucht werden, dessen Attribut `TeilnehmerID` den übergebenen Wert `pTeilnehmerID` hat. Dazu kann die private Methode `gibTeilnehmer` verwendet werden.
Gibt es ein entsprechendes Objekt und hat dieses Objekt einen Zeiteintrag, der den Wert `0` hat oder größer als `pZeit` ist, muss aktualisiert werden, ansonsten ist nichts zu tun und die Ausführung der Methode kann beendet werden.
Im Fall der Aktualisierung muss die im zu aktualisierenden Objekt die Ergebniszeit angepasst und die korrekte Sortierung der Liste `aktuelleTeilnehmer` bestimmt werden.
Hierzu wird das Objekt zunächst aus der Liste entfernt und dann an der richtigen Stelle neu eingefügt. Um dies zu bewirken, durchläuft man die Liste `aktuelleTeilnehmer` von vorne an, bis man ein Teilnehmerobjekt mit größerer Ergebniszeit oder der Ergebniszeit `0`

findet. Vor diesem wird das aktualisierte Objekt eingefügt. Wird keine Einfügestelle gefunden, wird das aktualisierte Objekt angehängt.

```
public void aktualisiereZeit(int pTeilnehmerID, int pZeit) {
  // Objekt des zu aktualisierenden Teilnehmers suchen.
  Teilnehmer akt = gibTeilnehmer(pTeilnehmerID);

  // Aktualisieren, wenn Teilnehmer existiert und keine oder
  // laengere Zeit eingetragen hat.
  if (akt != null
      && (akt.gibZeit() == 0 || akt.gibZeit() > pZeit)) {
    akt.setzeZeit(pZeit);
    aktuelleTeilnehmer.remove();

    // Teilnehmerobjekt neu einsortieren.
    aktuelleTeilnehmer.toFirst();
    while (aktuelleTeilnehmer.hasAccess()
        && aktuelleTeilnehmer.getContent().gibZeit() != 0
        && aktuelleTeilnehmer.getContent().gibZeit()
            <= akt.gibZeit()) {
      aktuelleTeilnehmer.next();
    }
    if (aktuelleTeilnehmer.hasAccess()) {
      aktuelleTeilnehmer.insert(akt);
    } else {
      aktuelleTeilnehmer.append(akt);
    }
  }
}
```

Teilaufgabe d)

Bei der Methode `tueEtwas` handelt es sich einen Auftrag ohne Parameter. In den Zeilen 2 bis 8 wird die Liste `aktuelleTeilnehmer` vom Anfang bis zum Ende oder bis zur ersten Teilnehmerin bzw. zum ersten Teilnehmer ohne Ergebniszeit durchlaufen. Aus den Daten jeder Teilnehmerin bzw. jedes Teilnehmers wird jeweils eine SQL-Anweisung erstellt, die an die Datenbank geschickt wird.

Bei dieser SQL-Anweisung in Zeile 5 handelt es sich um einen Update-Befehl, der die Ergebniszeit von Teilnehmerinnen und Teilnehmern in die Verknüpfungstabelle `nimmtTeil` einträgt. Um die richtige Stelle zu finden, werden die ID der Teilnehmerin bzw. des Teilnehmers und die ID des aktuellen Wettkampfes in `pWettkampfID` verwendet.

Im Sachzusammenhang erfüllt die Methode `tueEtwas` die Aufgabe, die Ergebniszeiten aller Teilnehmerinnen und Teilnehmer, deren Gruppe gerade an der aktuellen Station einen Wettkampf absolviert hat, in die Datenbank einzutragen.

Teilaufgabe e)

Auf den Wettkampfanzeigen werden der Name und die Zeit jeder Teilnehmerin und jedes Teilnehmers aufgeführt. Bei beiden Informationen handelt es sich um personenbezogene Daten, bei der Zeitangabe sogar um eine Leistungsbewertung. Das Datenschutzprinzip des „Verbots mit Erlaubnisvorbehalt“ besagt, dass personenbezogene Daten nur entsprechend gesetzlicher Regelungen oder mit Erlaubnis der betroffenen Personen genutzt werden dürfen. Eine explizite Erlaubnis zur Veröffentlichung der Daten liegt in diesem Fall vermutlich (sonst hätte sich niemand beschwert) nicht vor.

Allerdings ist die Teilnahme an den Wettkämpfen laut Beschreibung freiwillig und die geplante Veröffentlichung dieser Daten auf Anzeigetafeln ist bei der Anmeldung zu den Wettkämpfen bekannt. Es kann also bei jeder angemeldeten Person ein Einverständnis vorausgesetzt werden.

Ein lediglich implizites Einverständnis kann aber nicht ausreichen, um die Verbreitung personenbezogener Daten zu rechtfertigen. Die Kritik ist daher berechtigt.

Eine Möglichkeit, um auf die Einwände der Kritiker einzugehen, besteht darin, bei der Anmeldung abzufragen, ob der Name der Teilnehmerin oder des Teilnehmers auf den Anzeigetafeln erscheinen darf. Soll der Name nicht erscheinen, taucht die Zeit anonym in der Liste auf, so dass sie nur noch von unmittelbar anwesenden Personen zugeordnet werden kann.

Diese Idee könnte wie folgt realisiert werden:

In der Datenbank wird die Information, ob der Name einer Teilnehmerin oder eines Teilnehmers angezeigt werden soll, gespeichert. Dazu bietet sich ein boolesches Attribut in der Relation `Teilnehmer` an. Auch die Klasse `Teilnehmer` des Programms für die Wettkampfstationen wird um dieses Attribut ergänzt. Die Methoden `gibNachname` und `gibVorname` der Klasse `Teilnehmer` lesen dieses Attribut beim Aufruf aus und liefern ggf. nur Auslassungszeichen zurück.

7. Teilleistungen – Kriterien / Bewertungsbogen zur Prüfungsarbeit

Name des Prüflings: _____ Kursbezeichnung: _____

Schule: _____

Teilaufgabe a)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl (AFB)**	**EK**[2]	**ZK**	**DK**
1	überführt das Datenbankschema.	4 (I/II)			
2	ermittelt die Teilnehmerinnen und Teilnehmer mit ihren Gruppen.	2 (II)			
3	erläutert, wie die Information aus den Tabellen abzulesen ist.	4 (I/II)			
Sachlich richtige Lösungsalternative zur Modelllösung: (10) …….. ……..					
	Summe Teilaufgabe a)	10			

Teilaufgabe b)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl (AFB)**	**EK**	**ZK**	**DK**
1	erläutert die Idee einer SQL-Anweisung und geht auf die Joins ein.	4 (II)			
2	implementiert die Methode.	8 (II)			
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 (AFB)**	**EK**	**ZK**	**DK**
1	entwirft einen Algorithmus, nach dem die Zeit eines Teilnehmers aktualisiert werden kann.	4 (II)			
2	implementiert die Methode entsprechend dem Algorithmus.	8 (II)			
Sachlich richtige Lösungsalternative zur Modelllösung: (12) …….. ……..					
	Summe Teilaufgabe c)	12			

Teilaufgabe d)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl (AFB)**	**EK**	**ZK**	**DK**
1	erläutert den Algorithmus, nach dem die Methode arbeitet.	4 (II)			
2	erläutert die SQL-Anweisung in Zeile 5.	2 (II)			
3	erläutert die Aufgabe der Methode im Sachzusammenhang.	2 (II)			
Sachlich richtige Lösungsalternative zur Modelllösung: (8) …….. ……..					
	Summe Teilaufgabe d)	8			

Teilaufgabe e)

	Anforderungen	**Lösungsqualität**			
	Der Prüfling	**maximal erreichbare Punktzahl (AFB)**	**EK**	**ZK**	**DK**
1	erläutert jeweils mindestens ein schlüssiges Argument für und gegen die These und stellt einen Bezug zu den Prinzipien des Datenschutzes her.	4 (III)			
2	entwickelt eine eigene Position durch nachvollziehbare Abwägung der Argumente.	2 (III)			
3	erläutert, wie dem gegebenen Wunsch entsprochen werden kann.	2 (II)			
4	erläutert, wie die Datenbankmodellierung und das Programm geändert werden müssen.	2 (II)			
Sachlich richtige Lösungsalternative zur Modelllösung: (10) …….. ……..					
	Summe Teilaufgabe e)	10			