Hilfsmittel: *Java-IDE Netbeans, Umlet (Tool zum Zeichnen des Klassendiagramms), XAMPP (MySQL-DBMS, SQL-Client phpmyadmin)*

Die Aufgaben umfassen 6 Seiten.

Aufgabe 1

Themenkreis: Objektorientierte Programmentwicklung

Fussballsoftware gewinnt in den letzten Jahren in den Vereinen, Verbänden und hier in den Bereichen Vereinsverwaltung, Trainings- und Spielplanung sowie Analyse und Leistungsbewertung eine immer größere Rolle.

Hinzu kommt das Geschäft mit Computerspielen. Hierbei gehen die kräftigsten Impulse von den sog. Online-Spielen aus, an denen gerade Spielsoftware aus den Bereichen Fussballmanagement und Fussballsimulation einen großen Anteil haben.

Trotz des sehr unterschiedlichen Einsatzes der Software im Bereich Fussball gibt es gerade bei der Stammdatenverwaltung und –pflege viele Übereinstimmungen und Gemeinsamkeiten.

In diesem Kontext sollen Sie in der Rolle eines Mitarbeiters einer Firma für Softwareentwicklung Aufgaben im Bereich der Stammdatenpflege übernehmen.

Im ersten Schritt sollen Programme für die Stammdatenpflege von Mannschaftsdaten (Spieler, Trainer) erstellt werden.

Für die Datenhaltung ist ein relationales Datenbanksystem (MySQL bzw. Oracle) vorgesehen. Im weiteren Projektverlauf wurde in der sog. Konzeptionellen Phase der Datenbankentwicklung folgendes Datenmodell für die Stammdaten des Fussballmanagers verabschiedet:

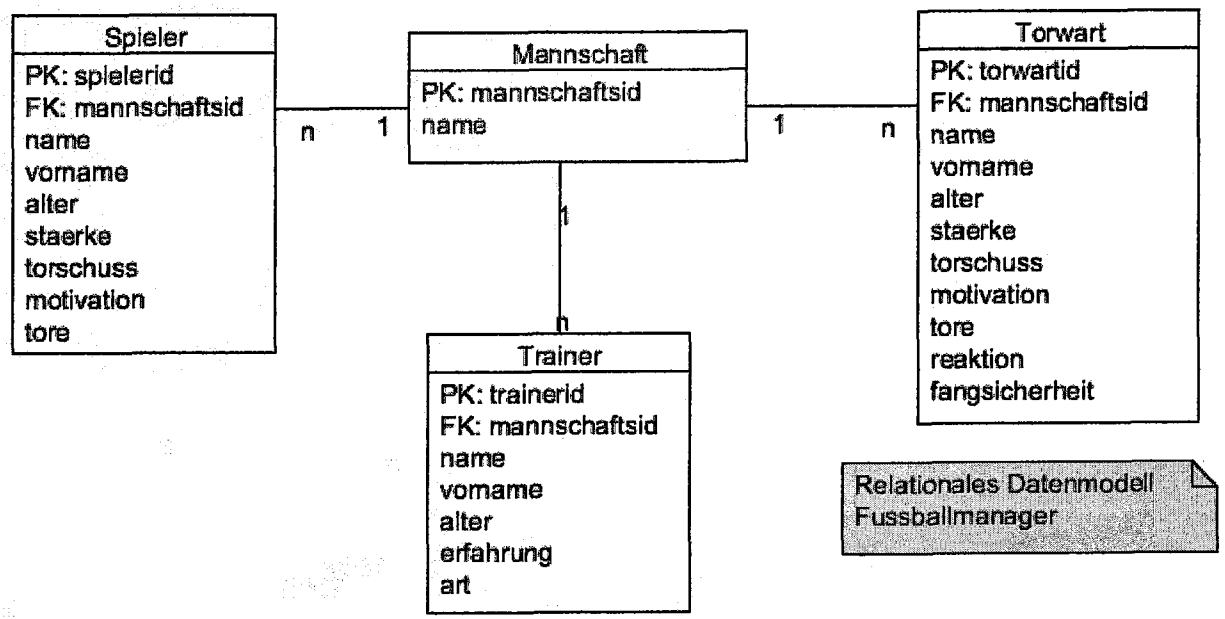

Abb.: Relationales Datenmodell der Stammdaten des Fussballmanagers

Im Rahmen des Projektes wird Ihr Part die Planung und Implementierung der Stammdatenpflege sein.

Das Pflichtenheft: Stammdatenpflege Fussballmanager

Zielbestimmung

Im ersten Schritt soll die Stammdatenpflege (Spieler, Trainer, Mannschaften) realisiert werden. Hierfür sollen zunächst alle notwendigen Methoden entwickelt werden. Die Entwicklung der zur Stammdatenpflege vorgesehenen Dialoge erfolgt zu einem späteren Zeitpunkt.

Produktfunktionen

- /F10/ Abspeichern der Stammdaten in die vorgesehenen Relationen
- /F20/ Abändern der Stammdaten in den vorgesehenen Relationen über den entsprechenden Primärschlüssel
- /F30/ Löschen von Stammsätzen aus den vorgesehenen Relationen über den betreffenden Primärschlüssel
- /F40/ Selektion der Stammdaten aus den betreffenden Relationen
- /F50/ Eine Konsolenanwendung zum Testen aller implementierten Methoden.

Produktdaten

- /D10/ Es können beliebig viele Stammdatensätze angelegt werden
- /D20/ Der Stammsatz eines Feldspielers besteht aus ID, Name, Vorname, Alter, Stärke, Torschuss, Motivation, Tore
- /D30/ Der Stammsatz eines Torwarts umfasst die Attribute ID, Name, Vorname, Alter, Stärke, Torschuss, Motivation, Tore, Reaktion, Fangsicherheit
- /D40/ Ein Trainerstammsatz umfasst die Eigenschaften Name, Vorname, Alter und Erfahrung und die Art des Trainers
- /D50/ Ein Mannschaftsstammsatz besteht aus der ID und dem Mannschaftsnamen.

Benutzungsoberfläche

- /B10/ Zu Testzwecken zunächst zeichenorientiert.

Auf Grundlage des Pflichtenheftes wird für das Software-System schließlich das sogenannte Fachkonzept erstellt. Dort werden die im Pflichtenheft formulierten Produktfunktionen nochmals spezifiziert.

Aufgabe 1.1

Erstellen Sie das Klassendiagramm

Sie sollen als ersten Schritt der Softwareentwicklung im Rahmen der Fachkonzepterstellung eine objektorientierte Analyse (OOA) für den Problembereich durchführen und Ihre Ergebnisse mit Hilfe eines Klassendiagramms modellieren und darstellen. Im Rahmen des objektorientierten Designs (OOD) wird das Modell immer weiter verfeinert. Verwenden Sie dafür die Unified Modeling Language (UML).

Beachten Sie bei der Erstellung des Klassendiagramms folgende Vorgaben:

- Ausschnittsweise soll hier nur die **Stammdatenpflege des Fussballmanagers** dargestellt werden. Die Methoden hierzu beschränken sich zunächst auf die im Pflichtenheft unter **F40 und F50** beschriebene Funktionalität:
    - Herstellen der Verbindung zur Datenbank
    - Schließen der Verbindung zur Datenbank
    - Selektion der Spieler einer bestimmten Mannschaft (Feldspieler und Torwart) über die MannschaftsID
    - Selektion des Trainers einer bestimmten Mannschaft
    - Selektion der Mannschaft über die MannschaftsID
    - Bilden der Mannschaft aus den selektierten Spielern und dem Trainer
    - Testprogramm.

- Berücksichtigen sie bei der Erstellung der Klassendiagramme das sog. **Drei-Schichten-Modell** (und zwar das Schichtenmodell **linearer Ordnung**!) und ordnen Sie die Klassen entsprechend einer der Schichten zu
- Deuten Sie mit Pfeilen Benutzt-Beziehungen an, in dem Sinne, dass die Klasse im Quellcode (Pfeilspitze) explizit verwendet wird
- Nutzen Sie, dort wo es Sinn macht, bereits beim Klassenentwurf, die **Vorteile der Vererbung**. Beachten Sie außerdem in diesem Zusammenhang, dass es sinnvoll sein kann, wenn von bestimmten Klassen **keine Instanzen** gebildet werden können!

Aufgabe 1.2

Programmieren der Klassen der Applikations- und der Persistenzschicht.

Implementieren Sie gemäß den oben formulierten Vorgaben (Pflichtenheft, Fachkonzept, Klassendiagramme) die entsprechenden Klassen und die Testklasse mit der JAVA-DIE! Berücksichtigen Sie auch hier das sog. 3-Schichtenmodell (Zugriff von einer Schicht auf andere Schichten gemäß dem **Prinzip der linearen Ordnung**!!)!

Implementieren Sie außer den getter- und setter-Methoden exemplarisch in Bezug auf den Datenbank-Zugriff folgende Funktionalität (siehe auch Aufgabe 1.1: Klassendiagramm):

1. Selektion aller Spieler einer Mannschaft über die MannschaftsID
2. Selektion des Trainers einer Mannschaft
3. Selektion der übrigen Daten der Mannschaft, Bilden einer Mannschaft, bestehend aus Spielern und Trainer.

Beachten Sie unbedingt: Nutzen Sie bei der Implementierung der Klassen die **Vorteile der Vererbung**!

Bei der **zwischen der Fachkonzept- und Datenhaltungsschicht vermittelnden Klasse** soll sichergestellt werden, dass von einer solchen Klasse nur **genau ein Objekt** erzeugt werden kann (**Singleton Muster**).

Aufgabe 1.3

Programmieren der Testfälle

Erstellen Sie eine Testklasse, in der alle oben aufgeführten Methoden zur Ausführung kommen!

Folgende Testfälle sollen nachgestellt werden:

Testfälle für die Selektion der Stammdaten

- Selektion der Spieler über die mannschaftsID
- Selektion der Trainers über die mannschaftsID
- Selektion der übrigen Mannschaftsangaben über die mannschaftsID
- Bilden einer Mannschafts-Instanz
- Ausgabe des Mannschaftsnamens (Eigenschaft der Mannschafts-Instanz)
- Anzeige von Name und Vorname der Spieler und des Trainers (ebenfalls über die zuvor gebildete Mannschafts-Instanz).

Aufgabe 2

Themenkreis: Sortieralgorithmen

Sortieralgorithmen sind in der Informatik von zentraler Bedeutung. Immerhin 25 % der kommerziell verbrauchten Rechnerzeit wird für Sortiervorgänge genutzt.[1] Daher wurde viel Zeit zum Auffinden schneller Algorithmen aufgewendet.

In der nun folgenden Aufgabe sollen zwei Sortieralgorithmen exemplarisch implementiert und bezüglich ihrer Effizienz verglichen werden.

Unten finden Sie einen Quellcode-Auszug eines Sortier-Algorithmus, und zwar des sog. **Bubblesort-Algorithmus**. Der Bubblesort-Algorithmus vergleicht der Reihe nach zwei benachbarte Elemente einer Liste und vertauscht sie, falls sie nicht in der richtigen Reihenfolge vorliegen. Ist er am Ende der Liste angekommen, wird der Vorgang wiederholt.

Implementieren Sie **einen** weiteren Sortieralgorithmus (entweder **Selectionsort**, **Insertionsort** oder **Quicksort**)!

Führen Sie für die beiden Sortieralgorithmen (BubbleSort und dem Algorithmus Ihrer Wahl) eine **Effizienzanalyse** bezüglich

a) der Zahl der Vergleiche
b) der Zahl der Bewegungen und
c) des Laufzeitverhaltens

durch!

[1] Saake, Gunter. Algorithmen und Datenstrukturen: eine Einführung mit Java. 1. Auflage. Heidelberg: dpunkt-Verlag, 2002, S. 116.

Quellcode Bubblesort

```
public void bubbleSort (int[] arr) {
    boolean swapped = true;
    int j = 0;
    int tmp;

    while (swapped) {
      swapped = false;
      j++;
      for (int i = 0; i < arr.length - j; i++) {
        if (arr[i] > arr[i + 1]) {
          tmp = arr[i];
          arr[i] = arr[i + 1];
          arr[i + 1] = tmp;
          swapped = true;
        }
      }
    }
}
```

Führen Sie **an einer eigenen Testklasse** die oben beschriebene Effizienzanalyse der zwei Sortieralgorithmen durch.

Sortiert werden soll ein **10000 Zufallszahlen** umfassendes Array aus Ganzzahlen, die einen Wert zwischen **0 und 100000** annehmen können. Die Bildung des Arrays soll **in einer eigenen Methode** stattfinden.

Die **Ausgabe des Testprogramms** könnte folgendermaßen aussehen:

```
run:
Zeit Insertion Sort: 281 ms Vergleiche: 25090935 Bewegungen: 25090935
Zeit Bubble Sort: 909 ms Vergleiche: 49988214 Bewegungen: 25260789
```

Aufgabe 3

Themenkreis: Datenbankentwicklung

Eine Hochschule bietet Veranstaltungen an, die an einem bestimmten Wochentag und einem bestimmten Block in einem bestimmten Raum stattfinden. Jede dieser Veranstaltungen wird von genau einem Dozenten angeboten, von dem Name und Vorname bekannt sind.

Ein Modul beschreibt die Veranstaltungen, die angeboten werden. Für jedes Modul gibt es eine eindeutige Modulnummer, eine Bezeichnung und die Zahl der CreditPoints, die nach erfolgreichem Absolvieren den Studierenden zugerechnet werden. Ein Modul umfasst in der Regel mehrere Veranstaltungen. Umgekehrt kann eine Veranstaltungen auch mehreren Modulen angehören.

Studierende besuchen Veranstaltungen. Studierende haben einen Namen und Vornamen, sowie eine eindeutige Matrikelnummer.

Folgende Daten sollen also verwaltet werden:

- Veranstaltung: Wochentag, Block, Raum
- Dozent: Name, Vorname
- Modul: Modulnummer, Bezeichnung, Credits
- Student: Matrikelnummer, Name, Vorname.

1. Entwerfen Sie das **semantische Datenmodell in Form eines Entity-Relationship-Diagramms** (ohne Attribute!!), das auch die Kardinalitätstypen der Assoziationen enthält.
2. Konstruieren Sie auf der Gundlage dieses semantischen Datenmodells schrittweise ein logisches Modell (**relationales Datenbankmodell**)! Legen Sie in jeder Relation (Tabelle) einen eindeutigen Schlüssel und alle geforderten Attribute (Spalten) an! Lösen Sie evtl. vorkommende Komplex-Komplex-Beziehungen (n:m) auf! Markieren Sie die Primär- und Fremdschlüssel in den Relationen!