Technische Hochschule Rosenheim
Technical University of Applied Sciences

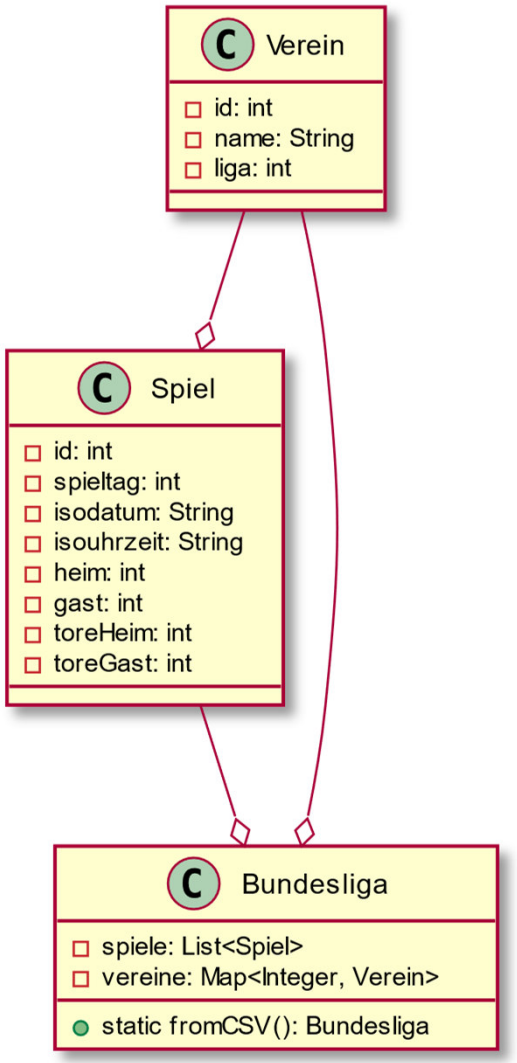
Objektorientierte Programmierung
Kapitel 7b – Annotationen

Prof. Dr. Kai Höfig

Datenmodell

- Wir haben in den vergangenen Wochen die wichtigsten **Datenstrukturen** und **Sortieralgorithmen** kennengelernt. Die Beispiele dazu waren aber meistens knapp und eher abstrakt -- in den wenigsten Fällen aber wirklich anschaulich.

- Heute wollen wir üben, das Gelernte praxisnah anzuwenden, in dem wir exemplarisch einige typische Problemstellungen in der Datenverarbeitung durchgehen. Als Datenbasis nehmen wir die Bundesligaergebnisse der Saison 2017/2018 (Quelle: fussballdaten.de). Es liegen Daten aus der 1. (18 Vereine), 2. (18 Vereine) und 3. Bundesliga (20 Vereine) vor, aus der 1. und 2. Liga bis einschließlich des 32. Spieltags und aus der 3. Liga bis einschließlich des 36. Spieltags.



Datenmodell

Technische Hochschule Rosenheim
Technical University of Applied Sciences

- Man sieht, dass die Vereine Heim und Gast in der Spieltabelle als foreign key auf die Vereinstabelle aufzulösen sind. Für unsere heutigen Beispiele können wir die Fabrikmethode `Bundesliga.loadFromResource()` verwenden, um die Daten aus den beiliegenden CSV Dateien einzulesen.

Beispiel aus Bundesliga_Spiel.csv und Bundesliga_Verein.csv

V_ID	Name	Liga
1	FC Bayern München	1
2	FC Schalke 04	1

Spiel_ID	Spieltag	Datum	Uhrzeit	Heim	Gast	Tore_Heim	Tore_Gast
1	1	2017-08-18	20:30:00	1	5	3	1
2	1	2017-08-19	15:30:00	7	12	1	0

Grundoperationen

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Datenverarbeitung beruht im Wesentlichen auf vier Grundoperationen:

Sortieren: die Daten in eine gewünschte Reihenfolge bringen

- Vereinsliste aufsteigend nach Vereinsname?
- Zuerst nach Liga, dann nach Vereinsname?

Filtern: Entfernen gewisser Daten, bzw. das Behalten von nur gewissen Daten

- *Gegeben*: Liste aller Vereine
- *Gesucht*: Liste aller 2.-Liga-Vereine

Abbilden: Daten von einem Format auf ein anderes umrechnen

- *Gegeben*: Liste der Spiele
- *Gesucht*: Liste der Spielpaarungen (Datum, Heim, Gast)

Reduzieren: Daten zusammenfassen

- *Gegeben*: Liste der Spiele
- *Gesucht*: Wie viele Tore wurden insgesamt geschossen?

Sortieren

Technische Hochschule Rosenheim
Technical University of Applied Sciences

- Oft ist die Reihenfolge, in der man Daten zur Verfügung gestellt bekommt, nicht die Reihenfolge, in welcher man diese darstellen oder weiterverarbeiten möchte.

- In unserem Beispiel sind die Vereine nach aktuellem Tabellenplatz und nach Liga sortiert. Möchte man aber die Liste aufsteigend nach Vereinsnamen sortiert haben, so muss man diese Liste entsprechend sortieren (vgl. Kapitel Sortieren).

```
// neue Liste erstellen, alle Vereine hinzufügen
List<Verein> nachName = new LinkedList<>(b.vereine.values());

// nach Vereinsnamen sortieren
nachName.sort(new Comparator<Verein>() {
    @Override
    public int compare(Verein o1, Verein o2) {
        return o1.getName().compareTo(o2.getName());
    }
});
```

Noch eine Sortierung

Technische Hochschule Rosenheim
Technical University of Applied Sciences

- Möchte man erst nach Liga und dann nach Namen sortieren, so muss man einen Comperator schreiben, der dieses Verhalten erfüllt:

```
List<Verein> nachLigaName = new LinkedList<>(b.vereine.values());
nachLigaName.sort(new Comparator<Verein>() {
    @Override
    public int compare(Verein o1, Verein o2) {
        // erst wenn Ligen gleich sind, dann nach Name!
        if (o1.getLiga() == o2.getLiga())
            return o1.getName().compareTo(o2.getName());
        else
            return Integer.compare(o1.getLiga(), o2.getLiga());
    }
});
```

Filtern

Technische Hochschule Rosenheim
Technical University of Applied Sciences

- ***Entfernen*** *gewisser Daten, bzw. das Behalten von nur gewissen Daten*
- e nach Anwendung kann es aber gut sein, dass wir gar nicht an allen Vereinen interessiert sind, sondern **nur an den Vereinen der 2. Liga**.

```
List<Verein> zweiteLiga = new LinkedList<>();
for (Verein v : b.vereine.values()) {
    if (v.getLiga() == 2)
        zweiteLiga.add(v);
}

zweiteLiga.sort(new Comparator<Verein>() {
    @Override
    public int compare(Verein o1, Verein o2) {
        return o1.getName().compareTo(o2.getName());
    }
});
```

- Man sieht hier auch: Es ist sinnvoll *vor* dem Sortieren zu filtern, da man sonst Daten sortiert, an denen man nicht interessiert ist.

Abbilden

Technische Hochschule Rosenheim
Technical University of Applied Sciences

- *Daten von einem Format auf ein anderes umrechnen*
- Wenn wir nur an den Spielpaarungen interessiert sind, also an welchem Datum welche Mannschaften gegeneinander spielen, so müssen wir ein Spiel auf ein Tripel abbilden.
- Dieses besteht aus dem Datum sowie den Namen von jeweils Heim- und Gastverein. Wir wollen also eine `List<Spiel>` in eine `List<Triple<String, String, String>>` abbilden.

```
List<Triple<String, String, String>> paarungen = new LinkedList<>();
for (Spiel s : b.spiele) {
    // Verwende Vereinstabelle um ID in Verein aufzulösen
    Verein heim = b.vereine.get(s.getHeim());
    Verein gast = b.vereine.get(s.getGast());

    // Erstelle neues Triple aus Datum sowie Vereinsnamen
    paarungen.add(Triple.of(s.getDatum(), heim.getName(), gast.getName()));
}
```

- *Die Klassen `org.apache.commons.lang3.tuple.Pair` und `org.apache.commons.lang3.tuple.`Triple sind generische Klassen für Paare und Tripel, welche mit den Fabrikmethoden `Pair.of(...)` sowie `Triple.of(...)` einfach instanziiert werden können.*

Reduzieren

Technische Hochschule Rosenheim
Technical University of Applied Sciences

- *Daten zusammenfassen*

- Abbilden bedeutet also bspw. das Umwandeln einer Liste in eine andere Liste. *Reduzieren* bedeutet, eine Liste von Werten auf einen einzigen Wert zu reduzieren. Ein einfaches Beispiel wäre das Aufsummieren von Werten in einem Array:

```
int[] a = {1, 2, 3};
int summe = 0;
for (int v : a)
    summe += a;
```

Reduzieren auf Tore

Technische Hochschule Rosenheim
Technical University of Applied Sciences

- In unserem Bundesligabeispiel wäre eine ähnliche Fragestellung: Wie viele Tore wurden insgesamt geschossen?

```
int tore = 0;
for (Spiel s : b.spiele) {
    tore = tore + s.getToreGast() + s.getToreHeim();
}

System.out.println("Es fielen insgesamt " + tore +
" Tore in " + b.spiele.size() + " Spielen");
// "Es fielen insgesamt 1741 Tore in 714 Spielen"
```

Aufgaben

Technische Hochschule Rosenheim
Technical University of Applied Sciences

Torstatistiken (T)

- T1 Wie viele Tore fallen durchschnittlich in jedem Spiel?
- T2 Wie viele Tore fallen durchschnittlich in einem Spiel der 1. Liga?
- T3 Wie viele Tore fallen durchschnittlich an einem Spieltag der 2. Liga?
- T4 Stimmt es, dass in den Nachmittagsspielen (15:30:00) im Schnitt mehr Tore fallen, wie in den Abendspielen?
- T5 Stimmt es, dass Vereine der 3. Liga zuhause im Schnitt mehr Tore schießen als auswärts?

Vereine (V)

- V1 Wie viele Tore hat der FC Bayern München (Verein 1) erzielt?
- V2 Wie viele Tore hat der FC Schalke 04 (Verein 2) kassiert?
- V3 Wie viele Punkte hat der 1. FC Nürnberg (Verein 20)? Ein Sieg zählt 3 Punkte, ein Unentschieden 1, eine Niederlage 0 Punkte.
- V4 Was ist das Torverhältnis des VfL Bochum (Verein 26), also die Rate von erzielten zu kassierten Toren?
- V5 Welche drei Vereine haben die meisten Tore zuhause geschossen, und wie viele?
- V6 Welcher Verein hat die wenigsten Tore auswärts geschossen, und wie viele?

Systematisch lösen

Technische Hochschule Rosenheim
Technical University of Applied Sciences

- Für die systematische Bearbeitung dieser Fragestellungen sollten Sie für jede dieser Fragestellungen zuerst die folgenden Fragen beantworten:

  - Was sind die Eingabedaten, und in welcher Datenstruktur liegen diese vor?
  - Was sind die Ausgabedaten, und in welcher Datenstruktur bzw. Form sollen diese vorliegen?
  - Welche Operationen (sortieren, filtern, abbilden, reduzieren) sind nötig, und in welcher Reihenfolge? Was sind die Zwischenprodukte?

- Verwenden Sie die Klasse Bundesliga sowie die Fabrikmethode `Bundesliga.loadFromResource()`
- die Attribute `List<Spiel> spiele` und `Map<Integer, Verein> vereine` sind öffentlich sichtbar (wie bereits den obigen Beispielen zu entnehmen).