Technische Hochschule **Rosenheim**
Technical University of Applied Sciences

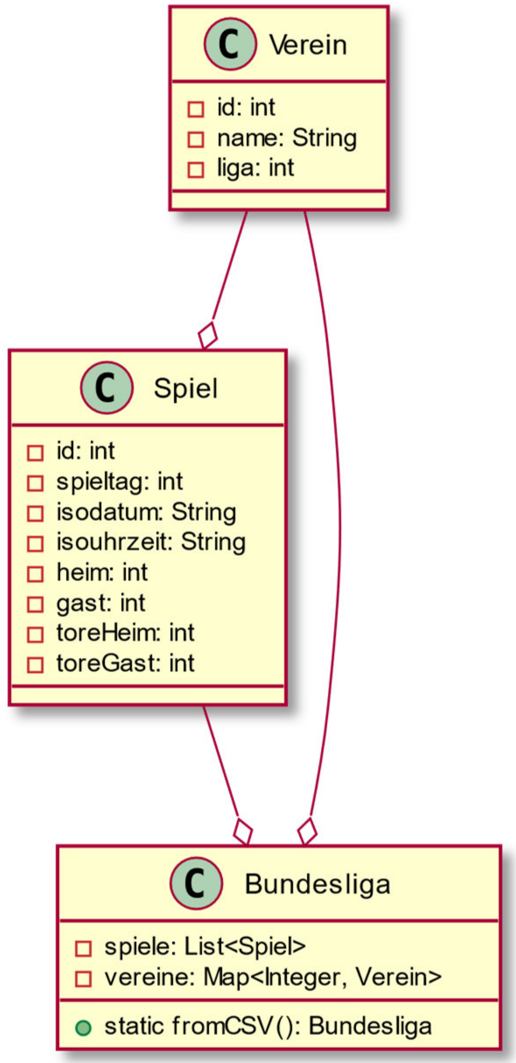
Object-oriented programming
Chapter 8 – Data processing

Prof. Dr Kai Höfig

Data model

- Over the past few weeks we have learned about the most important **data structures** and **sorting algorithms**. But the related examples were mostly brief and rather abstract -- there were only a few cases of really descriptive examples.

- Today we want to apply what we have learned in practice, by going through some rather typical problems in data processing. As the data pool, we will use the Bundesliga results of the 2017/2018 season (source: fussballdaten.de). Data is available from the 1st Bundesliga (18 clubs), 2nd Bundesliga (18 clubs) and 3rd Bundesliga (20 clubs), for the 1st and 2nd league up to and including the 32nd match day, and for the 3rd league up to and including the 36th match day.



Data model

- We can see that the home and away clubs in the games table are to be referenced as a foreign key in the club table. For today’s examples, we can use the factory method `Bundesliga.loadFromResource()` to read the data from the accompanying CSV files.

Example from Bundesliga_Game.csv and Bundesliga_Club.csv

C_ID	Name	League
1	FC Bayern Munich	1
2	FC Schalke 04	1

Game_ID	Match_ day	Date	Time	Home	Away	Goals_ home	Goals_ away
1	1	2017-08-18	20:30:00	1	5	3	1
2	1	2017-08-19	15:30:00	7	12	1	0

Basic operations

Data processing is essentially based on four basic operations:

Sorting: sorting the data into the desired order

- List of clubs in ascending order by club name?
- First by league, then by club name?

Filtering: removing certain data, or only retaining certain data

- *Given*: a list of all clubs
- *Wanted*: a list of all 2nd league clubs

Mapping: converting data from one format to another

- *Given*: a list of games
- *Wanted*: a list of match pairings (date, home, away)

Summarising: summarising/combining data

- *Given*: a list of games
- *Wanted*: how many goals were scored overall?

Sorting

- Often the order in which data is provided is not the order in which we want it to be presented or processed.

- In our example, the clubs are sorted by current table position and by league. However, if we want the list sorted in ascending order by club name, we have to sort this list accordingly (see the chapter on sorting).

```
// create a new list, add all clubs
List<Club> byName = new LinkedList<>(b.clubs.values());

// sort by club name
byName.sort(new Comparator<Club>() {
  @Override
  public int compare(Club o1, Club o2) {
    return o1.getName().compareTo(o2.getName());
    }
});
```

Another sort

- If we want to first sort by league and then by name, we have to write a Comparator that fulfils this behaviour:

```
List<Club> byLeagueName = new LinkedList<>(b.clubs.values());
byLeagueName.sort(new Comparator<Club() {
  @Override
  public int compare(Club o1, Club o2) {
    // only sort by name if leagues are the same!
    if (o1.getLeague() == o2.getLeague())
      return o1.getName().compareTo(o2.getName());
    else
            return Integer.compare(o1.getLeague(), o2.getLeague());
    }
});
```

Filtering

- ***Removing** certain data, or only retaining certain data*
- Depending on the application, it could be that we are not interested in all clubs, but **only in the clubs of the 2nd league**.

```
List<Club> secondLeague = new LinkedList<>();
for (Club c : b.clubs.values()) {
  if (c.getLeague() == 2)
        secondLeague.add(c);
}

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

- We can also see here: it is useful to filter *before* sorting, otherwise we will sort data that we are not interested in.

Mapping

- *Converting data from one format to another*
- If we are only interested in the match pairings, i.e. on what date which teams play against each other, we have to map a match to a triplet (Triple class).
- This consists of the date as well as the names of the respective home and away club. So we want to map a `List<Game>` to a `List<Triple<String, String, String>>`

```
List<Triple<String, String, String>> pairings = new LinkedList<>();
for (Game g : b.games) {
  // use club table to reference ID to club
  Club home = b.clubs.get(g.getHome());
    Club away = b.clubs.get(g.getAway());

  // create new Triple from date and club name
  pairings.add(Triple.of(g.getDate(), home.getName(), away.getName()));
}
```

- *The classes `org.apache.commons.lang3.tuple.Pair` and `org.apache.commons.lang3.tuple.Triple` are generic classes for pairs and triplets which can be easily instantiated using the factory methods `Pair.of(...)` and `Triple.of(...)`.*

Summarising

- Summarising/combining data
- So mapping means converting a list into another list, for example. *Summarising* means combining a list of values into a single value. A simple example would be to add up the values in an array:

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

Summarising goals

- In our Bundesliga example, a similar issue would be: how many goals were scored overall?

```
int goals = 0;
for (Game g : b.games) {
    goals = goals + g.getGoalsHome() + g.getGoalsAway();
}


System.out.println("There was a total of " + goals +
" goals scored in " + b.games.size() + " games");

// "There was a  total of 1741 goals scored in 714 games"
```

Tasks

Goal statistics (G)

- G1 How many goals are scored in each game on average?
- G2 How many goals are scored in each game in the 1st league on average?
- G3 How many goals are scored on a match day in the 2nd league on average?
- G4 Is it true that on average more goals are scored in the afternoon games (15:30:00) than in the evening games?
- G5 Is it is true that on average clubs in the 3rd league score more goals at home than away?

Clubs (C)

- C1 How many goals did FC Bayern Munich (Club 1) score?
- C2 How many goals did FC Schalke 04 (Club 2) score?
- C3 How many points does 1. FC Nürnberg (Club 20) have? A win counts 3 points, a draw 1 point, a defeat 0 points.
- C4 What is the goal difference of VfL Bochum (Club 26), i.e. the number of goals scored vs. goals conceded?
- C5 Which three clubs scored the most goals at home, and how many did they score?
- C6 Which club scored the fewest goals away, and how many did they score?

Solve systematically

- In order to solve these issues (tasks) systematically, you should first answer the following questions for each of the tasks:

  - What is the input data, and what data structure is it available in?
  - What is the output data, and what data structure or form should it be available in?
  - Which operations (sorting, filtering, mapping, summarising) are necessary, and in what order? What are the intermediate results?

- Use the Bundesliga class and the factory method `Bundesliga.loadFromResource()`
- The attributes `List<Game> games` and `Map<Integer, Club> clubs` are publicly visible (as already shown in the examples above).