Seminar 4: Designing an application using interfaces
24292-Object Oriented Programming

1 Introduction

The objective of this seminar is to learn how to design an application using interfaces. Concretely, we will use interfaces in order to sort the tables of the teams in a league, and to sort the top goal scorers of a competition.

The solution to these exercises will be implemented in Java during Lab session 4.

2 Review

We will use the concept of **interfaces** that we discussed in class. Recall that interfaces include abstract methods, but not implemented methods, making it possible for a class to inherit from multiple interfaces.

Again, to indicate in the design that you want to **override** a method, simply repeat the method again in the appropriate subclass.

In this seminar, it is not necessary to repeat all the classes from the previous seminars (most of them will remain unchanged). Instead, it is sufficient to design a subdiagram including the classes related to teams and team statistics, as well as players and player statistics.

3 Sorting league tables

Until now, the team statistics have been part of the `Team` class. However, this is impractical, since the same team may participate in multiple competitions over time, making it necessary to store team statistics for each such competition. For example, if a team participates in a league during 2022-23 and during 2023-24, we want to keep track of the number of wins, ties and losses for each separate league.

Since team statistics should be separate from the team itself, it makes sense to design a new class especially for this purpose, and relate this class to the existing `Team`. Think about how we can make sure that the correct team statistics

get associated to the correct team and league. Each team will need a method that updates the statistics for a given competition and match. Note that the team statistics may need access to information about the team (such as the name).

In addition, we can use the new class to compare the statistics of two teams. Java contains a method `Collections.sort` for sorting collections of elements. In order to sort collections whose elements are instances of a class `T`, it is necessary for `T` to implement the interface `Comparable`, which contains a single abstract method `int compareTo(Object o)`. The details of this method will be explained in the lab session.

4 Sorting goal scorers

Likewise, until now the player statistics have been part of the `Goalkeeper` and `Outfielder` classes, depending on the type of player. However, a player may also participate in multiple competitions over time, so we would like to store the player statistics for each competition. Just like team statistics, it should be possible to relate player statistics to a given player and a given competition. Note that the player statistics may need access to information about the player (such as the name).

In addition, since there are different player types, it is necessary to store different types of player statistics (e.g. goalkeeper statistics and outfielder statistics). Each player type should have to appropriate player statistics. Each player will need a method that updates the statistics for a given competition and match.

To sort goal scorers, we can again implement the interface `Comparable` and override the abstract method `int compareTo(Object o)`. The details of how to implement the sorting mechanism will be explained in the lab session.