Lab exercise 1: Implementing a class
21414-Object Oriented Programming

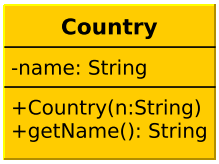
1 Introduction

The aim of this lab session is to implement the design of Seminar 1 consisting of the Player and Team classes.

The session is mandatory and you have to deliver the source code of the Java project and a document describing the implementation.

2 The Country class

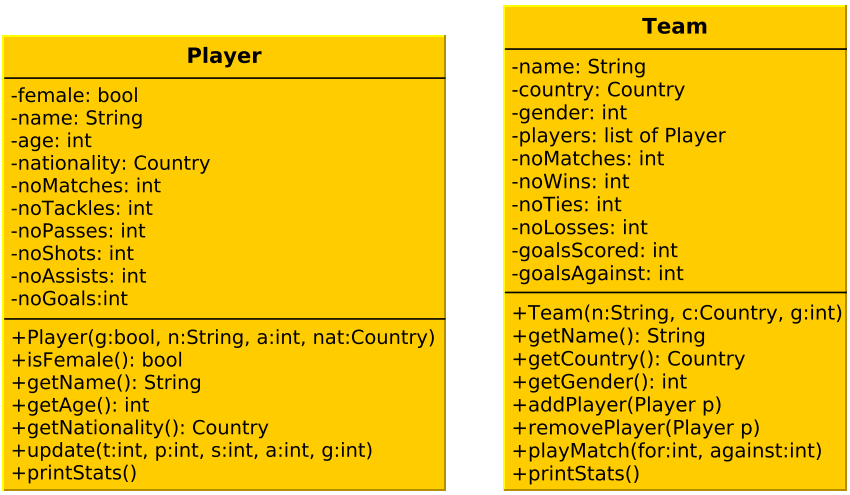
In the supplied code you will find a class Country that represents countries, with the following simple design:



The idea is to use this class in the application to represent countries.

3 The Player and Team classes

The next step is to implement the Player and Team classes from Seminar 1. During implementation, it is essential to define a test class with a main method, create instances of the different classes, apply the various methods and frequently compile to ensure that there are no errors. A sample design solution appears below:



3.1 The Player class

Add all the attributes specified in the design. The constructor should correctly set the initial values of all attributes. You should use the supplied Country class to represent the nationality of a player.

The method `update` should correctly update the statistics of a player as a result of playing a match. The method `printStats` should print the current statistics of the player, after playing multiple matches.

3.2 The Team class

Add all the attributes specified in the design. Again, the constructor should correctly set the initial values of all attributes, and you should use the Country class to represent the country of a team.

Instead of using an integer to represent the gender designation, you can use the `enum` construct in Java. For example, an enumeration of different colors will look as follows:

```
public enum Color { RED, GREEN, BLUE }
```

To use these colors, we can now define variables whose type is `Color`, and assign values from the enumeration:

```
Color myColor = Color.RED;
```

The method `addPlayer` should check that the player satisfies the criteria of playing for the team (i.e. the gender of the player should match the gender designation of the team).

The method `playMatch` should correctly update the statistics of a team as a result of playing a match. The arguments of the methods are the number of goals scored for the team, and the number of goals scored against the team. Note that the goals scored for and against are sufficient to update the other statistics (wins, ties, losses). The method `printStats` should print the current statistics of a team as a result of playing multiple matches.

4 Documentation

Apart from the source code, you should also hand in a document that outlines the solution of the problem. To elaborate the document you can use the following guidelines regarding the content:

1. An introduction where the problem is described. For example, what should the program do? Which classes do you have to define? Which methods do you have to implement for these classes?

2. A description of possible alternative solutions that were discussed, and a description of the chosen solution and the reason for choosing this solution rather than others. It is also a good idea to mention the related theoretical concepts of object-oriented programming that were applied as part of the solution.

3. A conclusion that describes how well the solution worked in practice, i.e. did the tests show that the classes were correctly implemented? You can also mention any difficulties during the implementation as well as any doubts you might have had.

The source code and documentation should all be uploaded to a directory `Lab1` of your Git repository **prior to the next lab session**. Please make sure that the directory name is *exactly* `Lab1` (not `Lab_1`, `Lab-1`, `LAB1`, `lab1` etc.)