Java For Industry: Take Home Assignment

The coursework is out of 60, including 10 marks for coding style and organisation. You mustn't use imports other than those found in java.util or java.io or java.lang. When you have completed your answers. Upload the completed IntelliJ project to learn.gold as a zip file.

You code will be checked against other students for similarity using a plagiarism detection tool that an analyses code structure regardless of variable names, comments and white space.

Java World Cup

In this assignment you will handle team data from a fictional association football world cup.

You have been provided with 2 external files that include player and manager data. You will need to load this data into your application, organise it it into squads, pick a match team and finally simulate a tournament from the data.

The team data is all fictitious, generated randomly. However, names of the teams reflect the teams in the 2022 World Cup. You will notice that the players don't only have traditional male names as this is a mixed tournament.

Inspect the template

An IntelliJ template has been created for you. Inspect its contents. You will see several skeleton classes and a complete implementation of a class called Squad and a further class team that extends Squad. There is also 2 data files, called Managers.csv and Players.csv.

Create Person, Player and Manager classes [5 marks]

In the provided template create a new class called Person. Person should have the following properties

Modifier	Type	Identifier
private	String	firstName
private	String	surname
private	String	team

Create a constructor that sets these properties. Alongside getter and setter methods.

Further to person. Create a Player and Manager class that is a child of the Person class. Each should have the following properties. All the properties should be set in a constructor and have a getter and setter method.

Player

Modifier	Type	Identifier
private	String	position
private	double	fitness
private	double	passingAccuracy
private	double	shotAccuracy
private	double	shotFrequency
private	double	defensiveness
private	double	aggression
private	double	positioning
private	double	dribbling
private	double	chanceCreation
private	double	offsideAdherence

Manager

Modifier	Type	Identifier
private	String	favouredFormation
private	double	respect
private	double	ability
private	double	knowledge

private	double	belief

Load in external data [15 marks]

In the Main class, load in the 2 data files (Players.csv and Managers.csv) using Scanners. The files are stored as CSV (comma separated values) files. Open them in IntelliJ to see their structure. Each file contains various information about the players and managers. For players this includes their name, team, position and a series of attributes. Each of these represent how effective the player is at particular skill as a proportion. Likewise, manager includes information on their name and attributes, also their preferred formation for the team.

Process each data row of the files to create players and managers in your program using the Manager and Player classes you wrote earlier. Arrange these objects in the static squad array in the Main class. When you are done the squad array will be populated with 32 objects of the Squad class. Squad has a Manager property to fill with a new Manager object from the data. Also, an arrayList players to populate with Player Objects you make from the scanned data.

The following hints will help you

- The first row of each CSV file is a header row, you can ignore its contents using the nextLine() method of your Scanner object
- Read in the rest of the file row by row using nextLine() in a loop
- Split the row up into individual data items using the split() method of String with the regular expression “\\,” as the only parameter. It will return an array of strings which you can use in the manager / player’s constructor to populate all the properties.

Create match teams [15 marks]

The squads you made in the previous part should have 26 players in each. Now you want to pick a team to play in a match.

Complete the static method getTeam in the Main class it should return a team that has 11 players matching the managers preferred formation. The formation is organised so the first digit is for defenders, the second for midfielders and the final digit for forwards. The goalie is not included but you must have one in your team. For example if the preferred formation is “4-4-2” your team will need 4 defenders, 4 midfielders, 2 forwards and 1 goal keeper.

Pick the best players for your team.To rank the players use an instance of the Comparator interface to organise your squad’s players. Combine all the players attributes and average them to find the best players.

Run tournament [15 marks]

This last part of the assignment is designed to be challenging and should only be attempted when you have completed the other parts. Don’t worry if you aren’t sure what to do or how to start. You can still get a very good mark on the assignment without completing it.

In this final part you are to design and run a tournament using the classes and methods you have created above. The tournament must follow the same pattern as a world cup. A group stage with 8 groups of 4 teams, where the top two teams go through. Followed by knockout stages to find a final winner.

You could go about this task in many ways with varying degrees of complexity. The most simple version might determine a winner of a match randomly. More involved versions will use the player data of each team to determine a winner of a match. You can appropriate that data how you want. For example you might determine that unfit players are more likely to get injuries during the tournament or aggressive players are more likely to get booked. Also, consider whether you wish to just predict a winner or score lines and more detailed match statistics.

When you have completed your implementation complete a short description of how it works in the text file TournamentDescription.txt in the template. The static method runTournament() should start it running.

Finally, make sure your implementation outputs effective information to the terminal describing match results and data.