Bilkent University

Department of Computer Engineering

Object Oriented Software Engineering Project

STARS League

Design Report

Metehan Kaya, Babanazar Gutlygeldiyev, Yılmaz Berkay Beken, Kerem Ayöz

Course Instructor: Eray TÜZÜN

Design Report
03.03.2018

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Object-Oriented Software Engineering course CS-319

Contents

1. Introduction

1.1 Purpose of the system

"STARS League" is a football game similar to "Football Manager". User plays as a manager who can control his team by changing its tactics, swapping players, etc. Also, the user has an opportunity to view groups, elimination stages, fixture, football players, man of the match, top goal scorers. Thus, user can do changes better for his own team in order to improve his team. The user's goal is to be the champion of the "STARS League". We designed the game in such a way that a match will be relatively harder than previous ones. The program aims to be portable, efficient, reliable, usable, and extensible. These features will make the game more enjoyable.

1.2 Design Goals

In this part, it is shown that what we expect and aim from the game. All of the important design goals are listed below.

1.2.1 Usability

To provide ease to use for users, a user-friendly interface is highly required. Therefore, we designed the game such that it will not be complicated to play. Since the game consists of many stages and menus, it is important to show them in a simple way. Simple and attractive interface will help the prevent the users from wasting time to understand the game. The game is designed in a such way that there will be no need to "How to Play" screen.

1.2.2 Reliability

Reliability is important for a software program since users prefers unproblematic program pleasure. Therefore, during the process of developing the game, we will give high importance to find the bugs to prevent possible crashes. The game will be consistent in the boundary conditions. To provide this, we will test the game many times at each stage.

1.2.3 Performance

Performance, or efficiency, is another important design goal of a software program. Users do not prefer a program which has efficiency problems. Therefore, we aim to implement the game such that minimum FPS (frames per second) of the program will be larger than 30.

1.2.4 Portability

Portability, or adaptability, is one of the most important aspects of a software program since a portability issue can make narrow the range of a software's usability for users. Instead of C, C++ or any other language, we are going to implement the game in Java, and thanks to the JVM which provides cross-platform portability, the game will not have such a problem.

1.2.5 Extensibility

Extensibility is a feature that can make a program more preferable from the user's perspective. The game will be extensible such that it will be possible to add some important or unimportant actions, change the possibilities of these actions, etc.

1.3 Trade Offs

1.3.1 Performance vs Memory

The main purpose of the program is to provide entertainment. Therefore, we prefer the game to be efficient rather than having less memory. We should increase the memory space to provide an efficient game.

1.3.2 Usability vs Functionality

In general, the users prefer usability, in other words user-friendliness, rather than functionality. Therefore, we will aim to provide usability which will make the game provide more entertainment. The game will consist of user-friendly interface which will let the user spend enjoyable time rather than struggling to learn it.

1.3.3 Efficiency vs Reusability

As it is indicated before, the main goal is to make the game efficient. Although we consider reusability of the program, we will not spend much time on designing the game in a such way that it will be highly reusable.

1.4 Definitions, acronyms, and abbreviations

[1] Java Virtual Machine (JVM): Java Virtual Machine which is an engine that can run Java programs. So that, operation systems that contain JVM can start Java applications.

[2] Graphical User Interface (GUI): A software that works at the point of contact (interface) between a computer and its user, and which employs graphic elements (dialog boxes, icons, menus, scroll bars, etc.) instead of text characters to let the user give commands to the computer or to manipulate what is on the screen.

[3] Cross Platform: Cross platform refers to ability of software to run in same way on different platforms such as Microsoft Windows, Linux and Mac OS X.

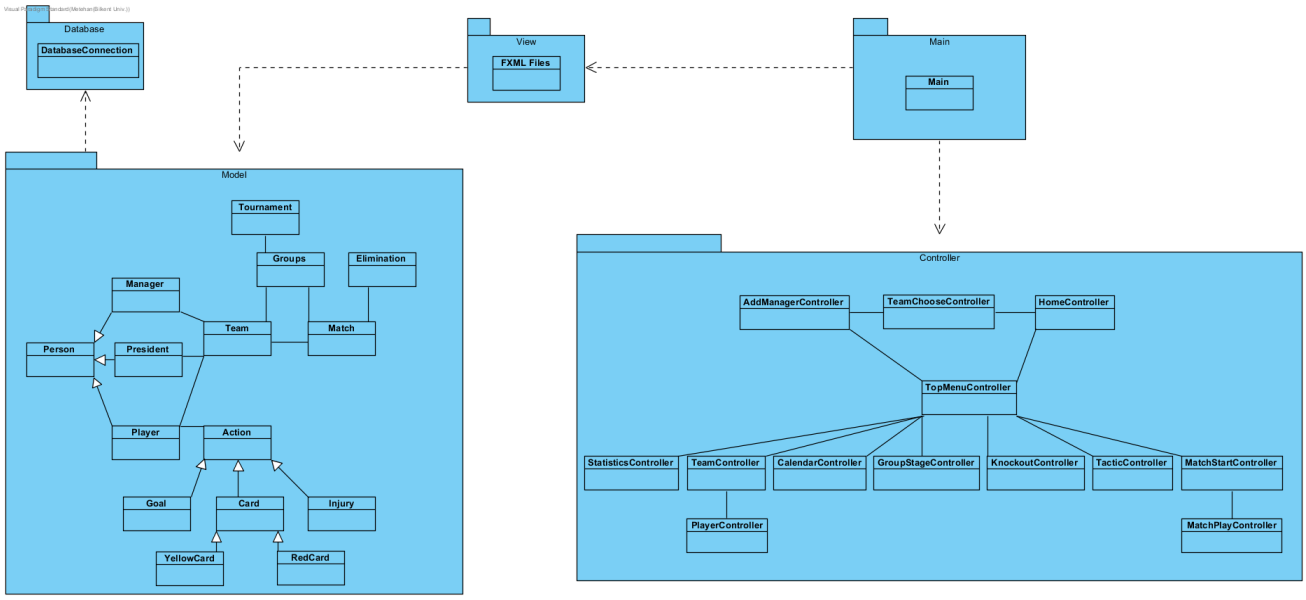
[4] Boundary Conditions: Conditions of the system which may generate run-time errors. They are exceptional cases according to the normal flow of the program. These conditions must be handled carefully for robustness of the system.

2. Software architecture

In this part of the report, we present architecture of our software. We plan to divide the system into subsystems to manage and maintain files and codes along the project. By doing this, we will provide coherence and prevent duplicates in the system.

2.1 Subsystem Decomposition

In this section, the system will be decomposed into subsystems. During the decomposition of the system, we have decided to choose MVC (Model View Controller) as the architectural pattern which is commonly used for developing user interfaces that divides our application into three interconnected parts called Model, View and Controller.



Model part will include classes that manage data, logic and rules of the game. For instance, we will have DatabaseConnection class that uses text file to manage data of the game.

"Game Model" subsystem consists of "Model" and "Database" packages. In "Game Model", we have classes Database Connection, Tournament, Team, Groups, Elimination, Person, Manager, President, Player, Match, Action, Goal, Card, Injury, YellowCard, RedCard.

View part will consist of FXML files. We will utilize JavaFX platform to visualize these components of "STARS League". These will be discussed further later.

The controller part is the part which provides coherence along the software. Controller accomplishes that by administering communication among classes and objects in "STARS League". For example, this part will accept, and validate if needed, inputs from the user and forward them to related objects.

"Game Controller" subsystem consists of "Controller" and "Main" packages. In "Game Controller", we have classes AddManagerController, TeamChooseController, HomeController, TopMenuController, StatisticsController, TeamController, PlayerController, CalendarController, GroupStageController, KnockoutController, TacticController, MatchStartController, MatchPlayController.

2.2 Hardware/software mapping

The "STARS League" will be implemented using Java programming language. So the user's device should support Java and should have Java Virtual Machine. Moreover, game utilizes keyboard and mouse as input device and screen and loudspeaker for output devices. Since we are using JavaFX for GUI, graphical requirements are like those JavaFX's.

2.3 Persistent data management

We did not feel the need to utilize complex data management systems like SQL or similar databases. So "STARS League" does not need any internet connection. We will store "STARS League's" data in text files. So whenever game is launched it will load the saved game from relative text file. This depends on choice of the user. If the user clicks "New Game" button on the screen it will ask for confirmation and erase previously saved game and create new game with new text file. Lastly when the user decides to exit and click related button "STARS League" will override the saved data with the newest one.

2.4 Access control and security

Since "STARS League" will not use any databases other than text files and will not connect to internet, there is no need to apply security measures. Moreover, since there will be only one user we also do not need any access to controls. This results that the user cannot continue the saved game in another computer.

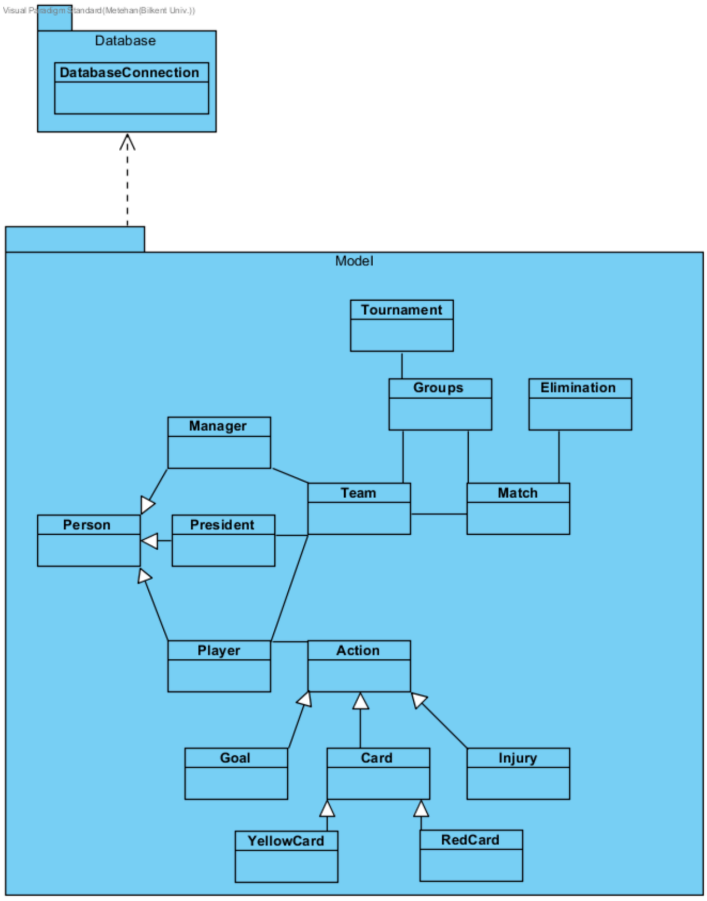
2.5 Boundary conditions

"STARS League" can be run from executable exe file to the computer as well as can be directly launched from jar file. Then, if it is run from exe file it will require permission from operating system. The game can be terminated by clicking the "X" button or clicking appropriate buttons from GUI. After that it will ask for confirmation. If confirmation is given it will save the game by writing to the text file. If there is any exceptional error occurs it will try to ignore that error and launch with that error. For example, if it cannot find an I/O, it will launch without terminating the application. However, an exceptional error related to database occurs it prompts an error message and requires the exception to be handled. For example, when the user tries to load a game while there is no saved game yet, it will prompt a frame with information about an error.

3. Subsystem services

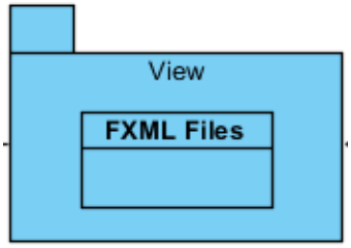
In our software, we choose to apply MVC(Model-View-Controller) design pattern so our software contains these subsystems. Also these subsystems contains different packages.

3.1 Game Model Subsystem



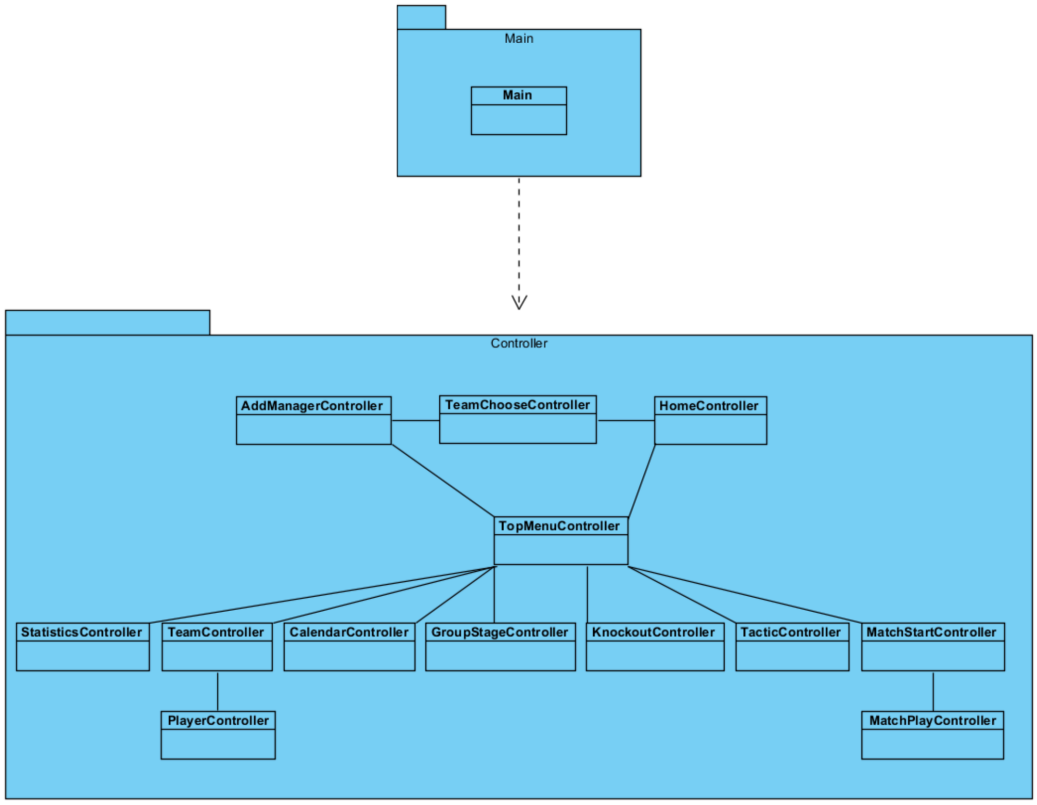
Game Model subsystem contains the model classes of the game and database connection classes. The model classes represents the game elements such as Match, Tournament etc. Model classes are the templates for information that will be used and stored in the game. Also the Game Model Subsystem contains database package that will interact with the database for saving and loading information. Tournament class is used as Facade Class of that system that will interact with the other subsystems. It is also a Singleton Class for sharing information about multiple controller classes.

3.2 Game View Subsystem



Game View subsystem contains the .fxml files that represents the frames of the program. These files contains the layouts of the buttons, links, combo boxes, radio buttons and other GUI elements. Also these files has the static images that will be displayed on the frame such as logo of the game. Controller Subsystem interacts with that .fxml files and do the proper action on these files. All of the controller classes interacts with its .fxml files so there is no need for Facade class in the View Subsystem.

3.3 Game Controller Subsystem



Game Controller subsystem has two packages: Main and Controller packages. Main package connects controller classes to other subsystems, so it is an interface for Game Controller Subsystem.

In the control package, there are classes for frames to changing the states of the views, interact with the user by taking inputs. These classes also communicates with the View subsystem to change the views properly.

4. Low-level design

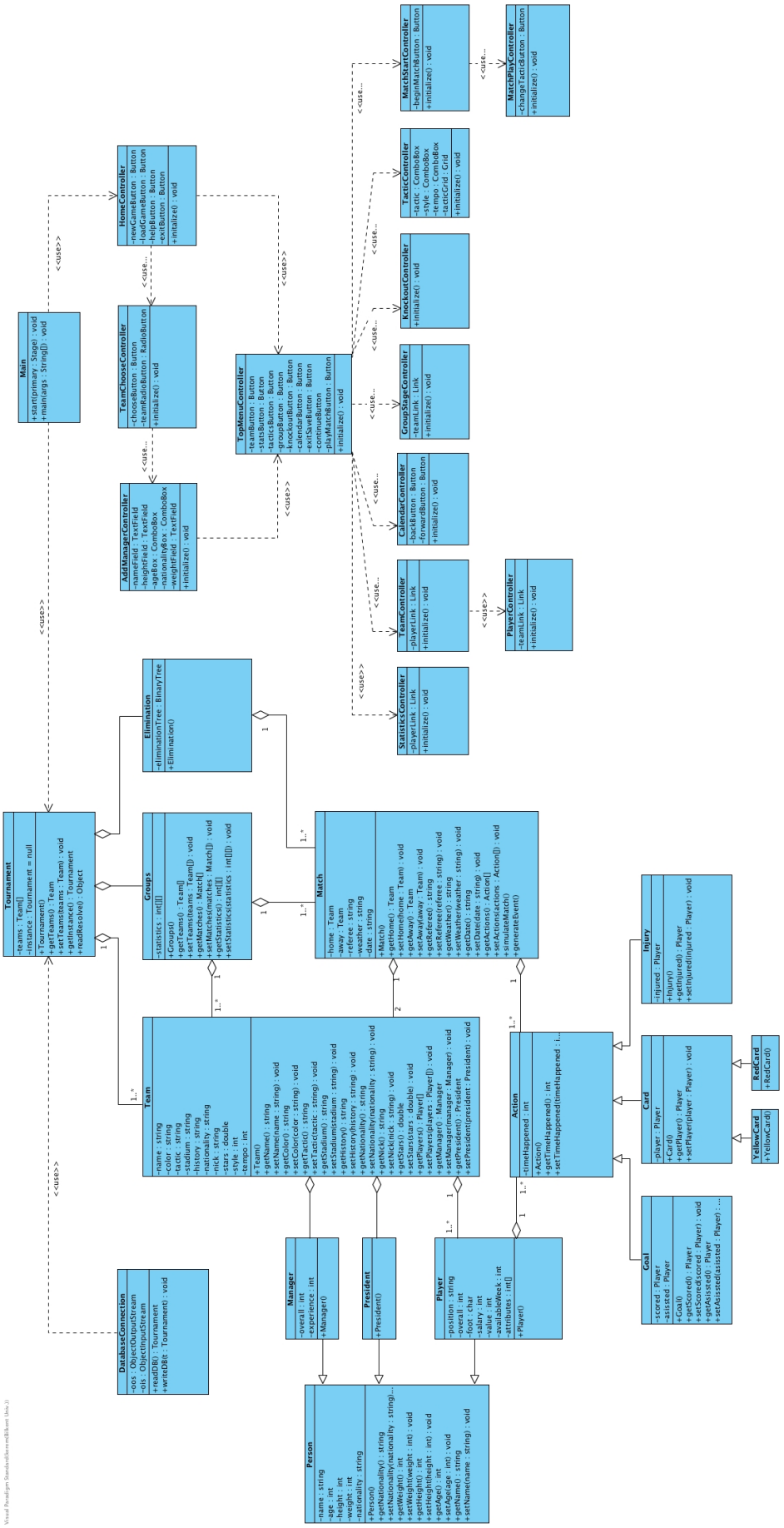
4.1 Object design trade-offs

4.1.1 Database Design vs Object Design

In the application, we implemented the Singleton design pattern in the Tournament class. Because we read the previous game data from our database just after the game is opened, then we modify these data in the main memory. Before exiting from the game, if user selects the option "Save Game", then the current modified data is written to our database. So our program will use database at most twice every time the game is played. Also there is no need for complex database design for that structure.

However, to implement that idea, we need to use the same instance throughout the different controller classes. Therefore we use Singleton design pattern. However, this design may bring some problems such that we need to think where this Singleton object should be created and how it will be accessed in program. So decreasing database complexity brings increased object design complexity.

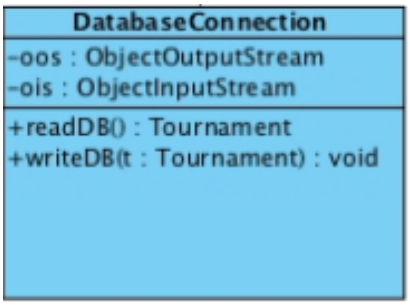
4.2 Final object design



Tournament Class

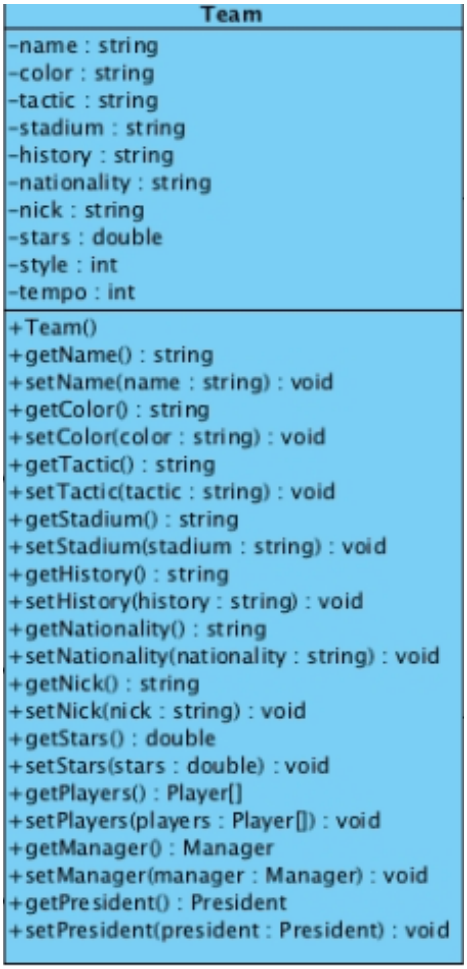
Tournament class represents the STARS League itself. It contains the 2 stages of the STARS League. Also it has the list of teams that are in the tournament. Having these instances of objects enables the Tournament class to be a 'Facade Class' in the Facade design pattern. It is the interface of model for interacting with the controller and view subsystems. Also all the information about tournament could be accessed from the Tournament object, so storing only Tournament object in database is enough for our program to load the previous game that is saved.

Also, all the other controller classes interact with the Tournament instance, therefore there should be only one Tournament instance during the program is running. Therefore, we decided to apply 'Singleton' design pattern for tournament. This Singleton instance is created at the beginning of the program-either loaded from database or created newly- and used by other controller classes.



DatabaseConnection Class

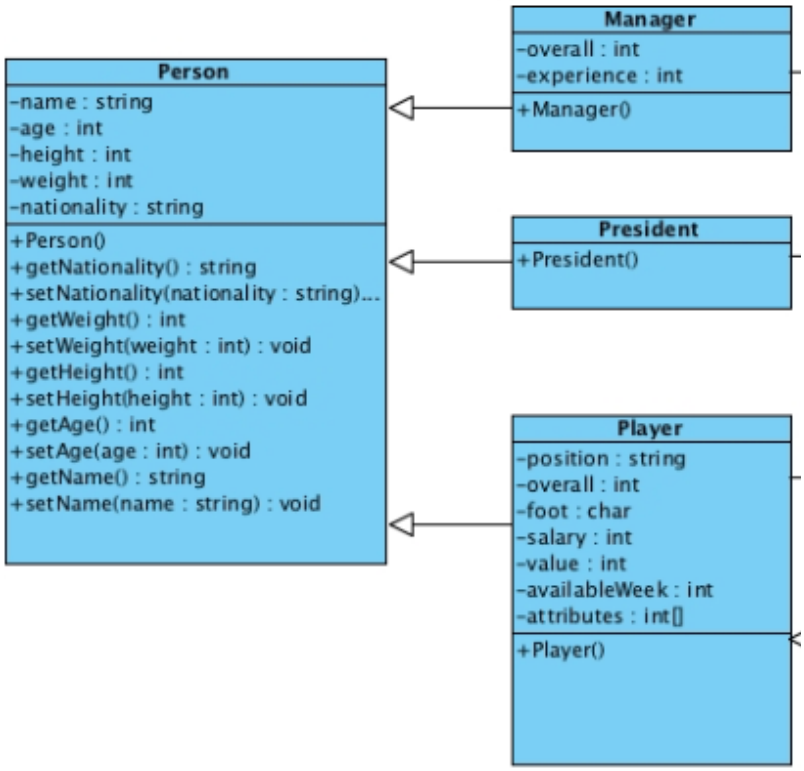
DatabaseConnection class enables program to read data from database and write data to database. The database itself is a simple .txt file. Object Serialization is used for reading and writing the data to .txt file. There is no need for the instance of the DatabaseConnection class thus read and write methods are static.



Team Class

Team class is a team's features. Team has a name which is string, teams have to have color(s) that's why, we have string color for every teams. Teams has to have a tactic(formation) for the incoming matches. Every team has to have a stadium for playing their own matches at their home. There would be information about every team's cups, championships, records at history string. Every team has a nationality because our game is consist of a international tournament and participants of the tournament from different countries. Every team has a nickname which is determined by their fans. (Galatasaray - Cimbom, Manchester United - Red Devils, etc.)

Every team would have stars which represents to team's powers according to their players' ratings. Evert team has a style for their incoming matches(ultra-attacking, defensive, normal etc.). Styles of the teams will affect their match results. For example if a team's style is defensive there would be less goals than ultra-attacking tactic.) Every team is affecting by their previous matches results this represents at tempo variable and that will affect team's incoming matches.



Person class

The "Person" class stores the properties of a person. Every person has a name, age, height, weight and nationality.

Manager class

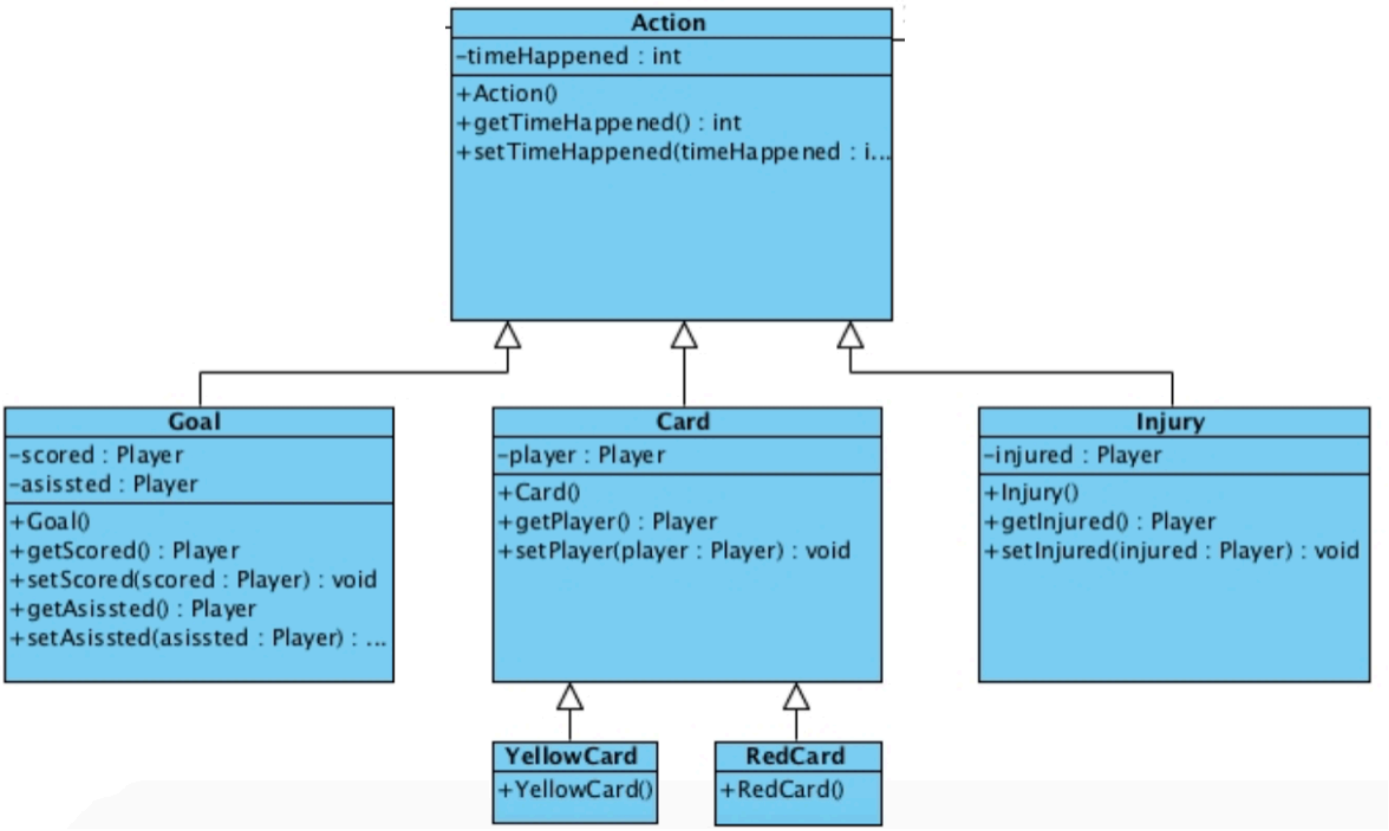
The "Manager" class stores the manager's attributes total by overall. That overall will affect the team's power which is managed by a manager significantly. Managers has an experienced year which is affecting manager's overall critically.

President class

The "President" class stores the president's name, age, height, weight and nationality according to be a subclass of "Person" class.

Player class

The "Player class stores the information of a player. Player has need to have position(s) that's why, there is a position variable. Player has a preferred foot by foot variable. Player has a wage information which stores at salary variable. Player has a value which is determined by player's attributes and overall and it stores at value variable. Player can get injured or suspended so they can't play for the incoming matches, that's why, there is an available week variable shows the availability of the player for incoming match(es).



Action Class

The "Action" class has a timeHappened variable for displaying the action's time to the user.

Goal Class

The "Goal" class has a variable for storing the player who scored.

Card Class

The "Card" class stores the player who got card during the match.

YellowCard Class

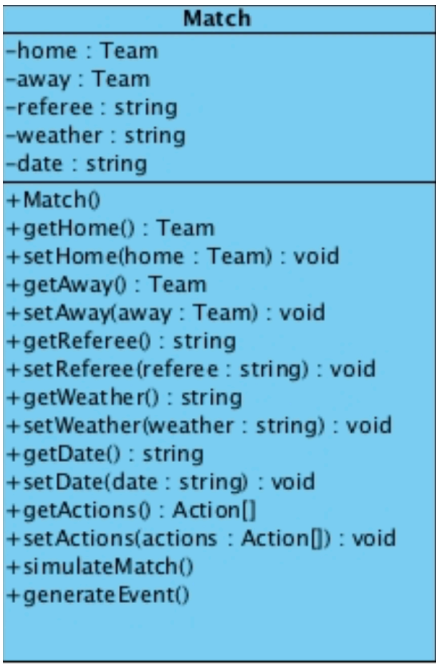
The "YellowCard" class stores the player who got yellow card during the match.

RedCard Class

The "RedCard" class stores the player who got red card during the match.

Injury Class

The "Injury" class stores the injured player during the match.

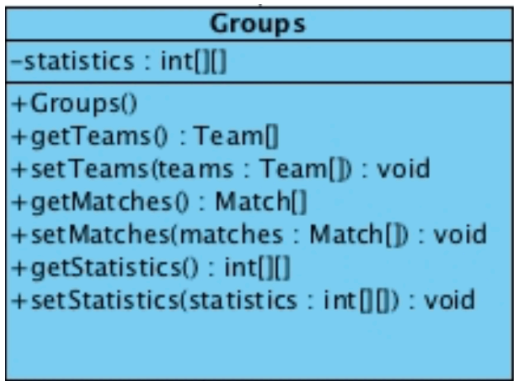


Match Class

The "match" class stores the home team by variable home. It also stores the away team by variable away. It stores the match's referee and display to the user by referee variable. It stores the weather of the match-day at the stadium and display it to the user. It stores the date of the match and it displays the date to the user.

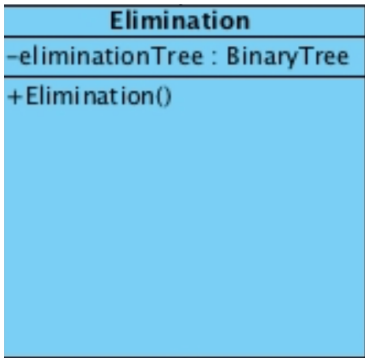
Match class contains the simulateMatch() and generateEvent() functions that are important for the implementation of the game. These functions determine the actions that will be created during the match. Event generation depends on the factors such as tactics and playing styles of the teams, overalls and specific skills of the players-for instance an aggressive player is tend to take a red card more than others-, previous match results and also the randomized algorithms. So even all the factors of one team is always greater than other, the less powerful team may have chance to beat his powerful rival in the STARS League.

Matches do not differ in the various stages of the tournament. The same pattern is used in both knockout matches and group stage matches.



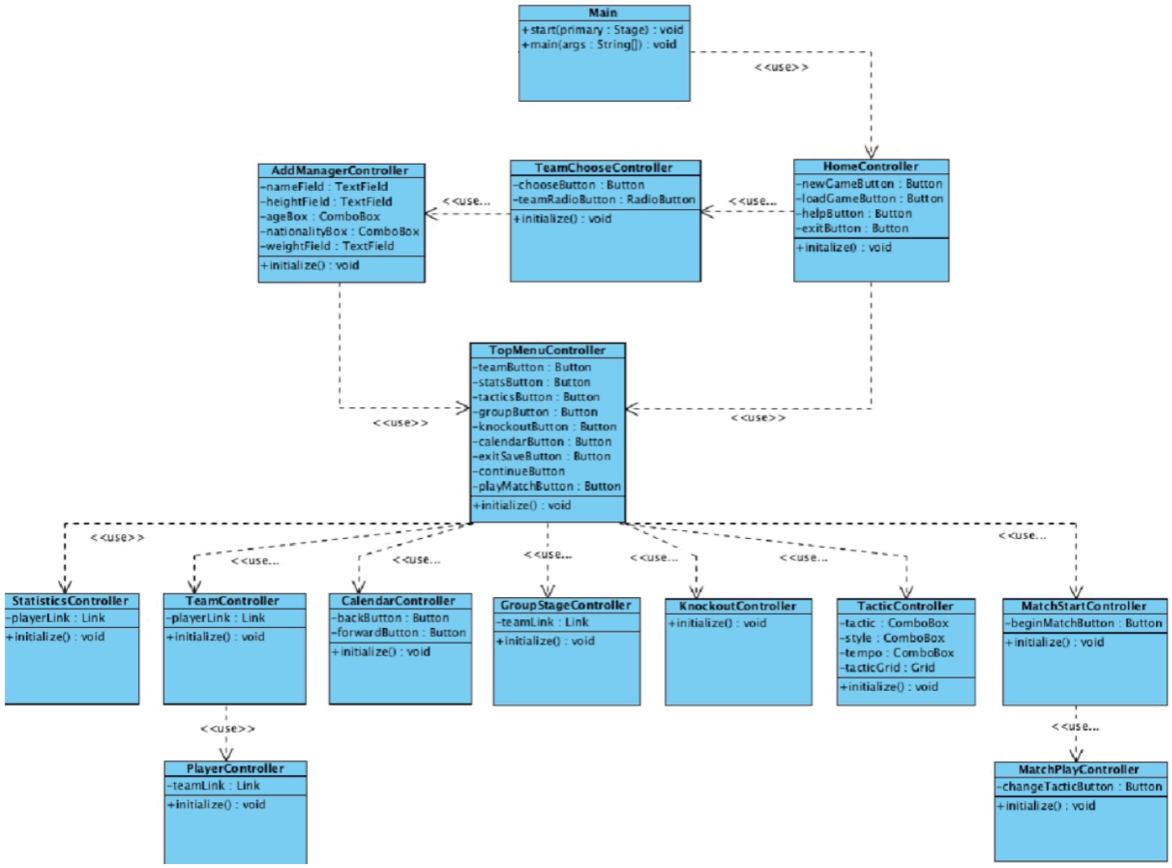
Groups Class

The "Groups" class stores the statistics of the tournament. This statistics consist of the number of matches that is currently played, goals scored in these matches, goals conceded in these matches and points gained from these matches for every team in that group. Groups will be consisted of 4 teams and there will be 8 groups at the tournament. 2 teams of every group which finished group stage first and second will be qualified to elimination stage. GroupStage has Match instances to store the matches that is played between that group members.



Elimination Class

The "elimination" class stores the teams who qualified to elimination stage from group stages. This class consist of a binary tree for maintaining the tournament. This tree represents the current stage of the knockout. Nodes of this tree are teams and the teams that have the same parent are rivals to each other. Also the match results is stored in that nodes.



Main Class

Main class is the 'Facade Class' of controller classes for interacting with model and view subsystems. It has the initializer functions for the program, so our program will start to run from that class. It organizes the other controller classes and initializes them.

AddManager Class

Managers have names, users will enter their name for adding a new manager by filling the nameField variable. Managers have height, users will enter their name for adding a new manager by filling the heightField variable. Managers have weight, users will enter their name for adding a new manager by filling the weightField variable. Managers have ages, users will enter their age by combobox for adding a new manager by filling the ageBox variable. Managers have nationalities, users will enter their age by combobox for adding a new manager by filling the nationalityBox variable.

TeamChooseController Class

In that controller class, every team has a radio button for user to choose one of them to manage. After selecting that radio button, user should press the chooseTeam button to continue to play the STARS League. This class also enables user to display the teams by clicking the links on the team names. So user can choose the team by evaluating players, play styles of the teams.

HomeController

There will be a new game button to start new game by clicking on newGameButton. There will be a load game button to maintain the game from last saved document by clicking on loadGameButton. There will be a help button for displaying the instructions of the game to the user by clicking on helpButton button. There will be a exit button for quitting the game by clicking on exit button.

TopMenu Controller Class

There will be teams button for displaying the situation of the teams (player list, tactics, previous match score) by clicking on teamsButton. There will be a stats button for displaying the most goal scorer list, the most assist list, the most shown yellow and red card player list by clicking on statsButton. There will be a group button for displaying the group's standing and results of the matches by clicking on groupButton. There will be a knockout button for displaying the knockout tree and results of the matches which is consist of qualified teams from group stage by clicking on knockoutButton. There will be a calendar button for displaying the fixture of a team(see previous matches schedule, result and incoming matches dates) by clicking on scheduleButton. There will be exit save button for saving the game before quitting from the game by clicking on exitSaveButton. There will be a continue button for passing the days by clicking on continueButton. There will be a play match button for displaying the match screen by clicking on playMatchButton.

StatisticsController Class

There will be link of players who consist of the statistics(top goal scorer, top assist, shown yellow card, shown red card) Thus, user will be able to access the player informations by clicking on playerLink link.

TeamController Class

There will be link of players who consist of the player list for every team. Thus, user will be able to access the player informations and can make changes on the teams by clicking on teamLink link.

CalendarController Class

There will be a back Button button for displaying the previous matches of the chosen team by clicking on backButton. There will be a forward Button button for displaying the incoming matches of the chosen team by clicking on forwardButton.

GroupStageController Class

There will be link of teams which consist of the groups. Thus, user will be able to access the team informations by clicking on teamLink link. User could access the all the information about the other teams from that part of the game.

KnockoutController Class

There will be link of teams which consist of the knockouts. Thus, user will be able to access the team informations by clicking on teamLink link.

TacticController Class

There will be a tactic combobox for choosing the team's tactic for incoming matches(4-4-2, 4-3-3, 4-2-3-1 etc.) User will choose the tactic of their own team by choosing a tactic on tactic Combo Box. There will be a style combobox for choosing the teams's style for incoming matches(ultra-attacking, defensive etc.) User will choose the style of their own team by choosing a style on style Combo Box. There will be a tempo combobox for choosing the team's tempo for incoming matches(calm, aggressive, ambitious etc.) User will choose the tempo of their own team by choosing a tempo on tempo Combo Box. There will be a tactic grid to user be able to change the tactic by clicking on tacticGrid.

MatchStartController Class

There will be begin match button to start a new match by clicking on beginMatchButton. User could see the brief information about the match such as referee, weather of the match day, odds of the match, stadium of the match, line-up and tactic of the other player(user could not see it before deciding his/her tactic).

MatchPlayController Class

There will be change tactic button to change the tactic during the match for affecting the result of the match by clicking on changeTacticButton button.

PlayerController Class

This controller is for the frame of player display. User could see all the information about player that is displayed on screen; skills, overall, statistics, team, etc. Also user can display that player's team by clicking on the team logo, this takes user to the Team view screen.

4.3 Packages

In total, the system has 3 subsystems called "Game Controller ", "GameView" and "Game Model".

- "Game Controller" consists of 2 packages called "Controller" package which has almost all classes related to controller of the system and "Main" package which has only "Main" class.
- "Game Model" consists of 2 packages called "Model" package which has almost all classes related to model, and "Database" package which has only "DatabaseConnection" class.
- "Game View" has FXML files.

5. Glossary & references

[1] https://www.w3schools.in/java-tutorial/java-virtual-machine/

[2] http://www.businessdictionary.com/definition/graphical-user-interface-GUI.html

[3] https://www.webopedia.com/TERM/C/cross_platform.html

[4] http://www.klabs.org/history/history_docs/sp-8070/ch4/4p1_design_tradeoffs.htm