Group assignment 2: Class diagram

Students: (Group F2a)

Einar Helgi Þrastarson

Hannes Pétur Eggertsson

Sigurður Birkir Sigurðsson

Teachers:

Matthias Book

Kristín Fjóla Tómasdóttir

1 Introduction

In this document there's the class diagram for group F2a. Group members are: Einar Helgi Þrastarson (personal ID number: 110287-2919), Hannes Pétur Eggertsson (240889-2939) and Sigurður Birkir Sigurðsson (120589-2539). Our project is to build an user interface for a fantasy football game. In our class diagram we felt it made sense to split the classes into two categories, back-end classes and front-end classes. In the appendix we put our current idea how the UI will look like when the game is ready. The presenter on Wednesday, March 4th 2015, will be Hannes Pétur Eggertsson.

1.1 Notation

In our class diagrams we use the following notation:
  − means a private variable or method (not directly accessable by other classed).
  + means a public variable or method (directly accessable by other methods).

Also, each class in the diagram has four sections:

The class' name.
Short description of the class.
The class' variables and their type listed on the format: −/+ `type1 variable1`   −/+ `anotherClass variable2`
The class' methods listed on the format: −/+ `type1 method1(type variable,...)`   −/+ `type2 method2(...)`

Classes are then connected using arrows:

Class A $\xrightarrow{uses}$ Class B

Class A $\xrightarrow{extends}$ Class B

Class A $\xrightarrow{implements}$ Class B

In most cases we can tell how many classes 'Class A' and 'Class B' will be associated with, this is shown by placing an arrow at the beginning and end of an arrow, e.g.

Class A $\xrightarrow[]{1 \quad uses \quad 0-10}$ Class B

if each instance of 'Class A' will use 'Class B' in a range of 0 to 10 instances.

1.2 Terminologies and Concepts

User A (human) user actually playing the game.

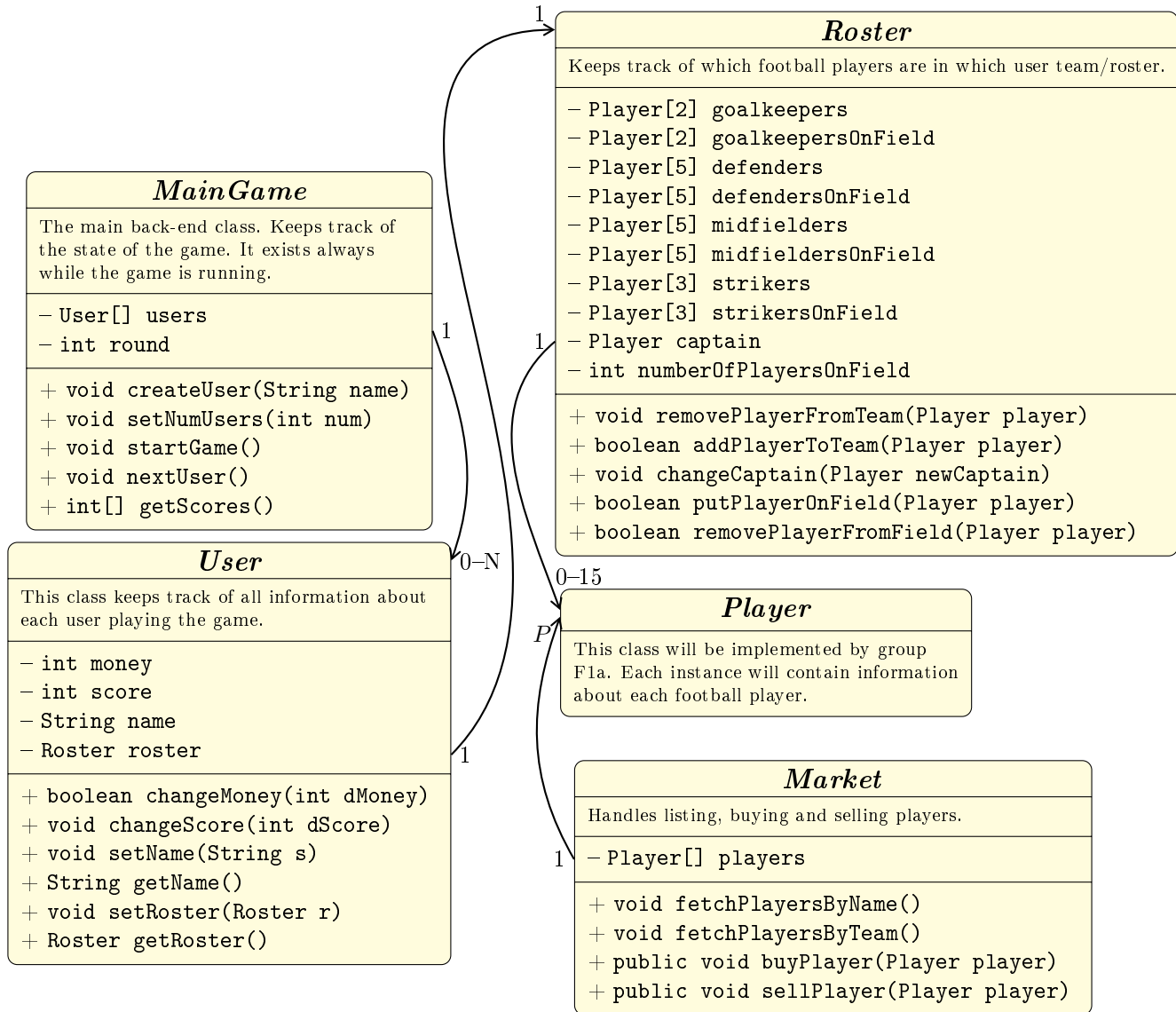**Player** A football player in the game, e.g. Gylfi Sigurðsson.

Team A football team in the game, e.g. Manchester United.

Roster The user's bought players, the players could be from multiple teams. Note that the roster includes both players which user has set on and off the field. A roster can have up to 15 players.
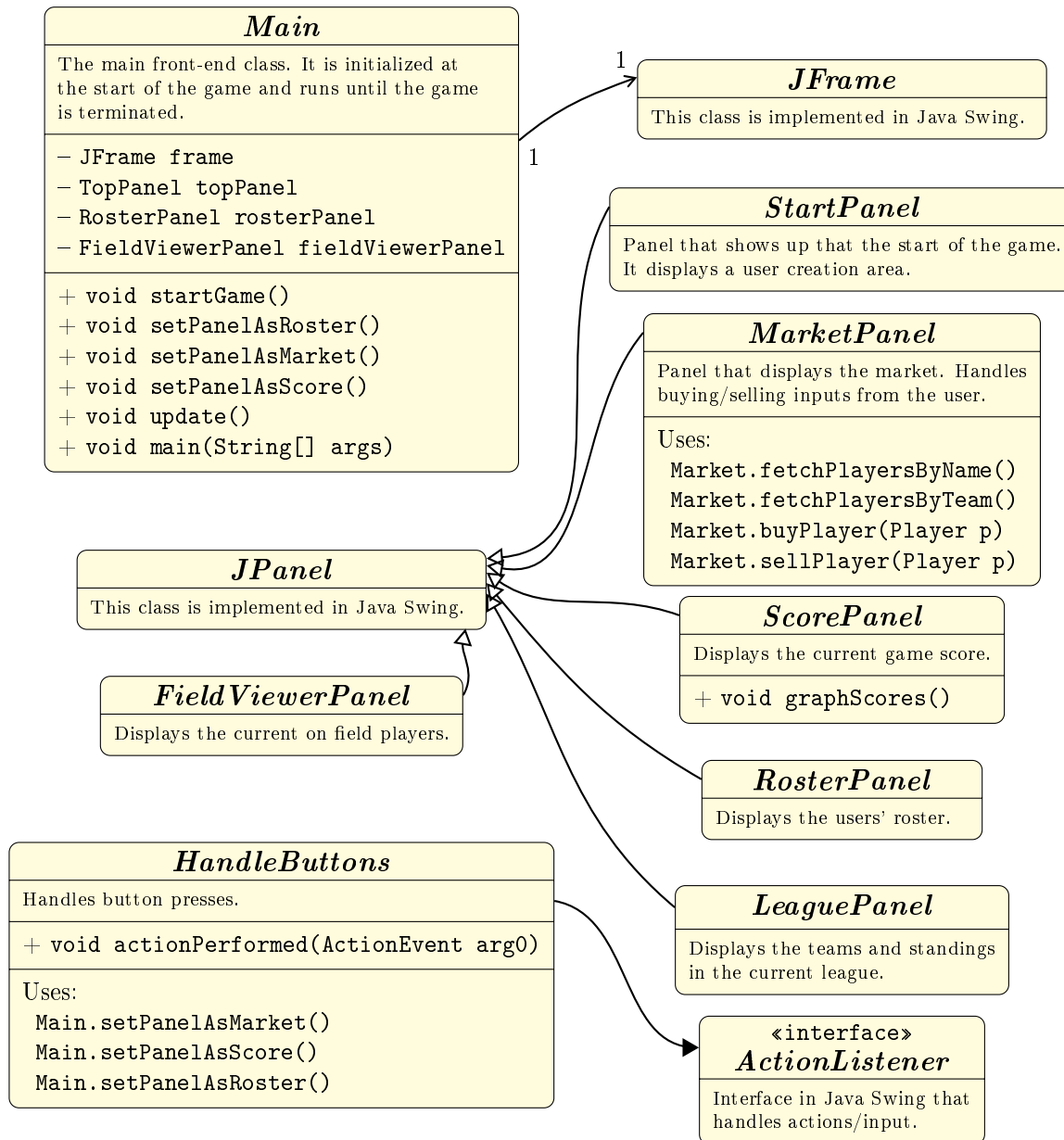
2 Class diagram

We decided to split our class diagram into two figures: **Back-end classes** and **Front-end classes**. The back-end classes take care of storing and keeping track of all information as the game is running. The front-end classes take care of displaying the information to the users playing the game as well as handling their input.

2.1 Back-end classes

Roster

Keeps track of which football players are in which user team/roster.

- Player[2] goalkeepers
- Player[2] goalkeepersOnField
- Player[5] defenders
- Player[5] defendersOnField
- Player[5] midfielders
- Player[5] midfieldersOnField
- Player[3] strikers
- Player[3] strikersOnField
- Player captain
- int numberOfPlayersOnField

+ void removePlayerFromTeam(Player player)
+ boolean addPlayerToTeam(Player player)
+ void changeCaptain(Player newCaptain)
+ boolean putPlayerOnField(Player player)
+ boolean removePlayerFromField(Player player)

MainGame

The main back-end class. Keeps track of the state of the game. It exists always while the game is running.

- User[] users
- int round

+ void createUser(String name)
+ void setNumUsers(int num)
+ void startGame()
+ void nextUser()
+ int[] getScores()

User

This class keeps track of all information about each user playing the game.

- int money
- int score
- String name
- Roster roster

+ boolean changeMoney(int dMoney)
+ void changeScore(int dScore)
+ void setName(String s)
+ String getName()
+ void setRoster(Roster r)
+ Roster getRoster()

Player

This class will be implemented by group F1a. Each instance will contain information about each football player.

Market

Handles listing, buying and selling players.

- Player[] players

+ void fetchPlayersByName()
+ void fetchPlayersByTeam()
+ public void buyPlayer(Player player)
+ public void sellPlayer(Player player)

Where N is er number of total users in the current game and P is the total amount of football players in the game.

2.2 Front-end classes

Main

The main front-end class. It is initialized at the start of the game and runs until the game is terminated.

− JFrame frame
− TopPanel topPanel
− RosterPanel rosterPanel
− FieldViewerPanel fieldViewerPanel

+ void startGame()
+ void setPanelAsRoster()
+ void setPanelAsMarket()
+ void setPanelAsScore()
+ void update()
+ void main(String[] args)

JFrame

This class is implemented in Java Swing.

StartPanel

Panel that shows up that the start of the game. It displays a user creation area.

MarketPanel

Panel that displays the market. Handles buying/selling inputs from the user.

Uses:
  Market.fetchPlayersByName()
  Market.fetchPlayersByTeam()
  Market.buyPlayer(Player p)
  Market.sellPlayer(Player p)

JPanel

This class is implemented in Java Swing.

FieldViewerPanel

Displays the current on field players.

ScorePanel

Displays the current game score.

+ void graphScores()

RosterPanel

Displays the users' roster.

HandleButtons

Handles button presses.

+ void actionPerformed(ActionEvent arg0)

Uses:
  Main.setPanelAsMarket()
  Main.setPanelAsScore()
  Main.setPanelAsRoster()

LeaguePanel

Displays the teams and standings in the current league.

«interface» ActionListener

Interface in Java Swing that handles actions/input.

1

1

Appendix

User interface

ROSTER	MARKET	SCOREBOARD

BIRKZ'S TEAM: