

Group assignment 4: Refined OO model

Próun hugbúnaðar Spring 2015

Students: (Group F2a) Einar Helgi Þrastarson Hannes Pétur Eggertsson Sigurður Birkir Sigurðsson Teachers: Matthias Book Kristín Fjóla Tómasdóttir

1 Introduction

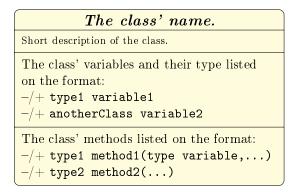
In this document there's the class diagram for group F2a. Group members are: Einar Helgi Prastarson (personal ID number: 110287-2919), Hannes Pétur Eggertsson (240889-2939) and Sigurður Birkir Sigurðsson (120589-2539). Our project is to build an user interface for a fantasy football game. In our class diagram we felt it made sense to split the classes into two categories, back-end classes and front-end classes. Then, in a third diagram there's another diagram that shows the connections between the back-end and We will all present this document on Wednesday, March?th 2015.

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 classes).

Each class in the diagram has four sections shown below:



If the class wasn't created by us it is filled with red. Classes are then interconnected using 3 types of arrows:

$$\begin{array}{ccc} \text{Class A} & \xrightarrow{uses} & \text{Class B} \\ \\ \text{Class A} & \xrightarrow{extends} & \text{Class B} \\ \\ \text{Class A} & \xrightarrow{implements} & \text{Class B} \end{array}$$

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.

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

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

1-N

User

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

- int id
- int money
- List<Integer> score
- String name
- Roster roster
- + User(String name, int id)
- + int getMoney()
- + boolean isAffordable(int price)
- + void changeMoney(int dMoney)
- + Roster getRoster()
- + List<Integer> getScore()
- + List<Integer> getTotalScore()
- + void addScore(int newscore)
- + String getName()
- + void setName(String newname)

Main Game

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

- StatsHistory stats
- List<User> users
- int round
- int currentUser
- + MainGame()
- + void setNumUsers(int num)
- + void nextUser()
- + int getRound()
- + List<User> getUsers()
- + StatsHistory getStatsHistory()
- + User getCurrentUser()
- + int getCurrentUserID()

Roster

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

- List<Player> goalkeepers
- List<Player> goalkeepersOnField
- List<Player> defenders
- List<Player> defendersOnField
- List<Player> midfielders
- List<Player> midfieldersOnField
- List<Player> forwards
- List<Player> forwardsOnField
- int numberOfPlayersOnField
- + Roster()
- + int getNumberOfPlayersOnField()
- + boolean removePlayerFromField(Player player)
- + void removePlayerFromRoster(Player player)
- + boolean addPlayerToField(Player player)
- + boolean addPlayerToRoster(Player player)
- + List < List < Player > getPlayersInRoster()
- + List< List<Player> > getPlayersOnField()
- + boolean isInRoster(Player player)
- + boolean isOnField(Player player)

\0<u>-15</u>

Player «interface»

This class will be made by group F1a. Each instance will contain information about a football player.

Market

Handles listing, buying and selling players.

- Player[] players
- + Market()
- + void fetchPlayersByName()
- + void fetchPlayersByTeam()
- + public void buyPlayer(Player player)
- + public void sellPlayer(Player player)

StatsHistory

A class that has statistical information.

- PlayerScores[] allplayerscores
- int[] rosterscores
- + StatsHistory()
- + void calcTeamScore()
- + int[] getTeamScoresHist()

PlayerScores

A class with information about each player.

- String name
- int[] scores
- + PlayerScores(String name)
- + void setScore(int score)
- + int[] getScores()