Group assignment 4: Refined OO model

Þróun 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 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. 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:

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(...)`

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

Class A —*uses*→ Class B

Class A —*extends*→ Class B

Class A —*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.

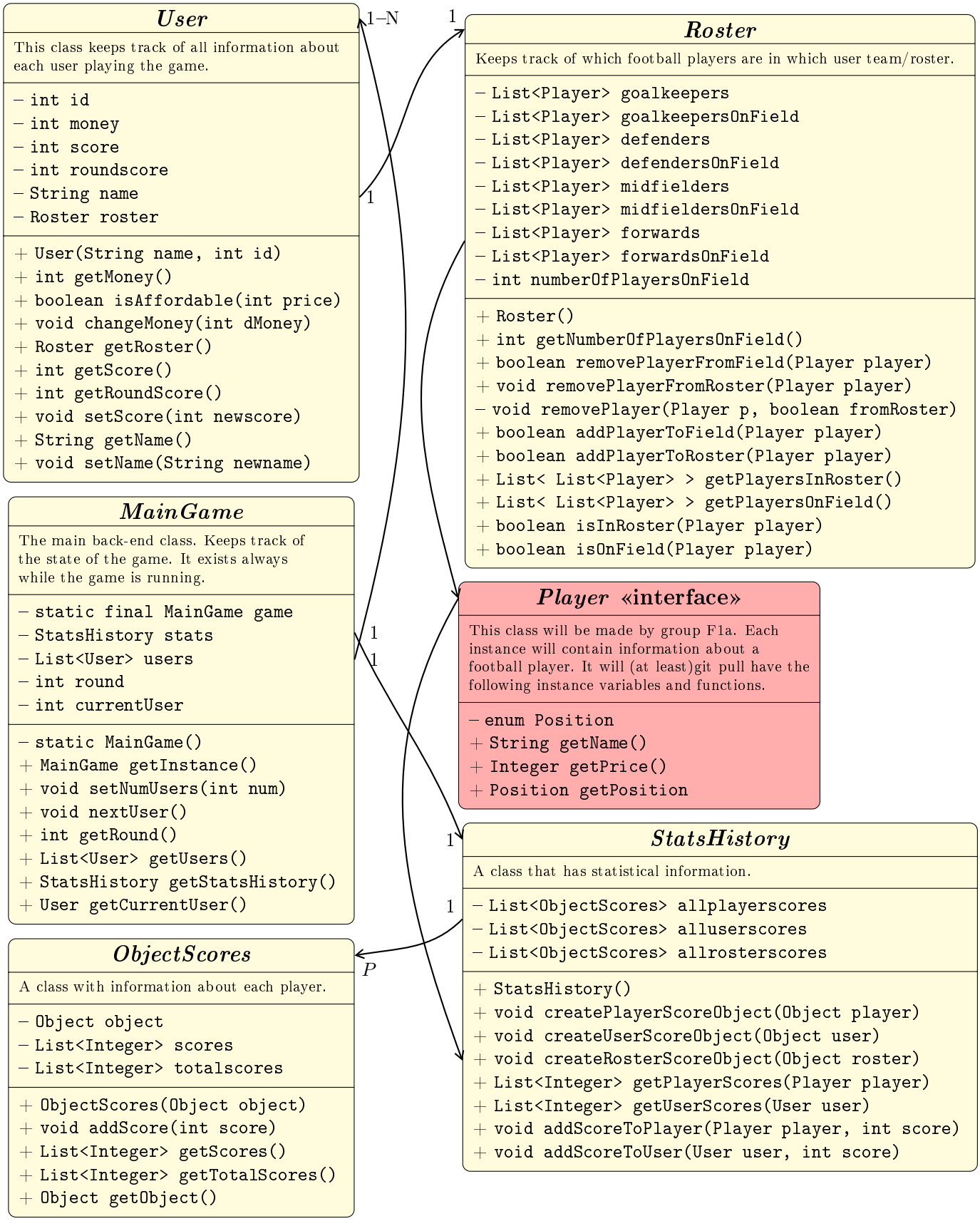
Class A —1 *uses* 0–10→ Class B

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

2 Class diagram

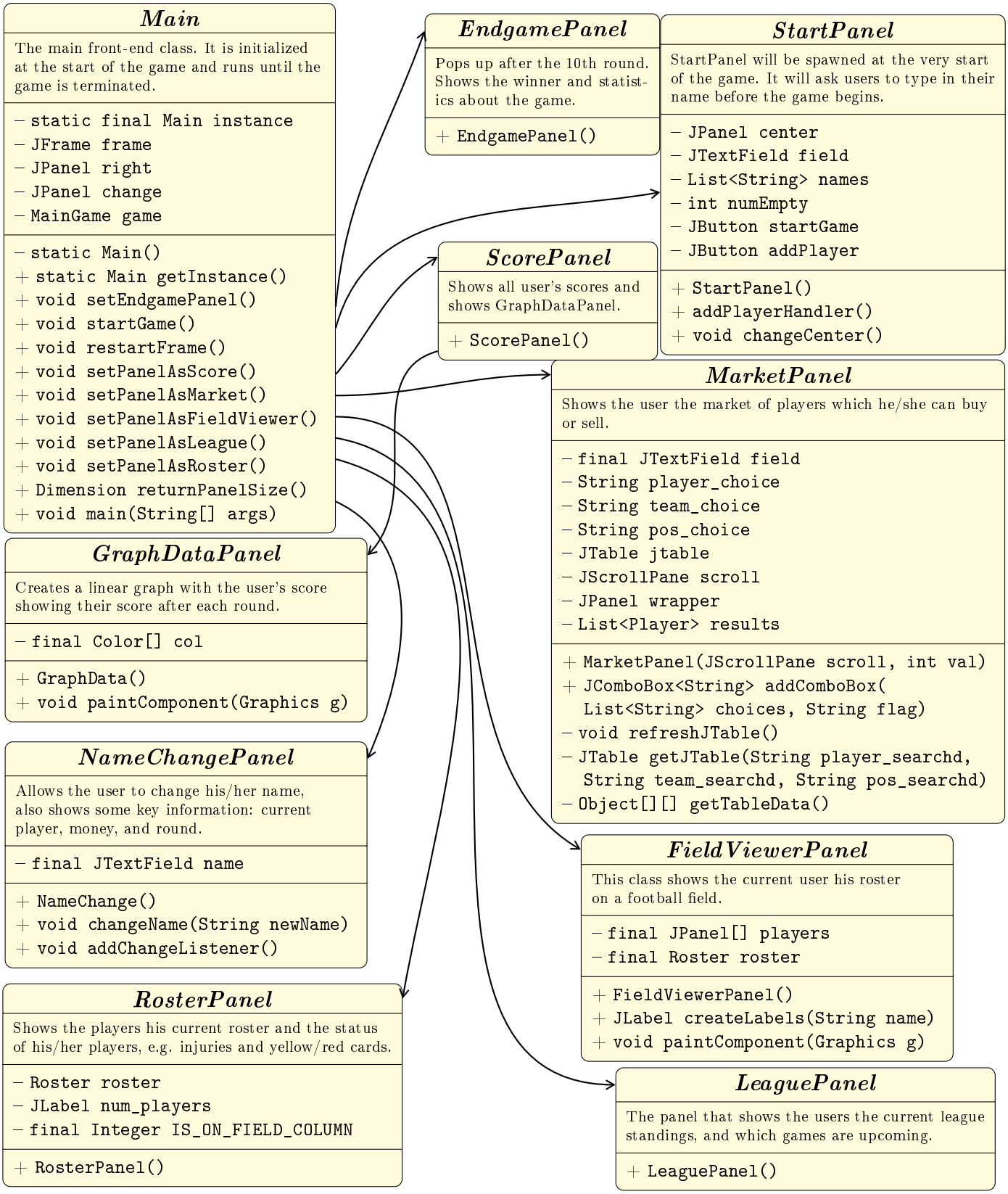
We decided to split our class diagram into two figures: **Back-end classes** and **Front-end classes**. The back-end classes take care of storing and keeping track of all information as the game is running. The front-end classes take care of displaying the information to the users playing the game as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

2.2 Front-end classes



Multiplicities are all 1:1.

2.2.1 Component classes

We are also using the following two component classes

ButtonColumn
Changes a single column of a JTable to buttons.
− JTable table − Action action − int mnemonic − Border originalBorder − Border focusBorder − JButton renderButton − JButton editButton − Object editorValue − boolean isButtonColumnEditor
+ ButtonColumn(JTable table, Action action, int column) + Border getFocusBorder() + void setFocusBorder(Border focusBorder) + int getMnemonic + void setMnemonic(int mnemonic) + Component getTableCellEditorComponent(JTable t, Object val, boolean selected, int row, int col) + Object getCellEditorValue() + Component getTableCellRendererComponent(JTable t, Object val, boolean selected, boolean focus, int row, int col) + void actionPerformed(ActionEvent e) + void mousePressed(MouseEvent e) + void mouseReleased(MouseEvent e)

CustomButton
Our own custom button that implements JButton. Looks much nicer than the default Swing button.
− Color hoverBackgroundColor − Color pressedBackgroundColor − Color fontColor − Color hoverFontColor
+ CustomButton() + CustomButton(String text) + Color getHoverBackgroundColor() + Color getFontColor() + void setFontColor(Color color) + void setContentAreaFilled(boolean b) + void setHoverBGColor(Color color) + Color getHoverFontColor() + void setHoverFontColor(Color color) + Color getPressedBGColor() + void setPressedBGColor(Color color) + void paintComponent(Graphics g)

2.3 Connections between front-end and back-end classes

3 Sequence diagrams