Sudoku

project

Galya Dodova | Opject oriented programming with Java

Contents

[Introduction 2](#_Toc87368490)

[1.1. Descrition and Idea 2](#_Toc87368491)

[1.2. Aim and tasks to be done 2](#_Toc87368492)

[1.3. Structure of the documentation 2](#_Toc87368493)

[Reviewing the subject 2](#_Toc87368494)

[2.1. Simple definitions, concepts and algorithms 2](#_Toc87368495)

[2.2. Finding the problems and the difficulty of the problem 2](#_Toc87368496)

[2.3. Methods for solving the problems 2](#_Toc87368497)

[2.4. Functional needs 2](#_Toc87368498)

[Making the project 2](#_Toc87368499)

[3.1. Architecture and diagrams 2](#_Toc87368500)

[Testing and realization 2](#_Toc87368501)

[4.1. Realization of classes 2](#_Toc87368502)

[4.2. Memory control and optimization 2](#_Toc87368503)

[4.3. Planning and creating testing scenarios 2](#_Toc87368504)

[Conclusion 2](#_Toc87368505)

[5.1. Review and future plans 2](#_Toc87368506)

[Literature 2](#_Toc87368507)

# Introduction

## Descrition and Idea

Write a RMI JFX client application that plays the popular Japanese game SUDOKU. Sudoku puzzles are 9 x 9 grids, and each square in the grid consists of a 3 x 3 subgrid called a minigrid. Your goal is to fill in the squares so that each column, row, and minigrid contains the numbers 1 through 9 exactly once. Implement the Sudoku game using JavaFX.

## Aim and tasks to be done

• Create and reusable user defined visual component of JavaFX packaged in a JAR file • Ensures that only valid numbers are allowed to be placed in the currently selected cell • Checks whether a Sudoku puzzle has been solved • Keeps track of the time needed to solve a Sudoku puzzle • Undo and redo previous moves • Can request and start another game from the server b) The RMI server (JavaFX application) • Generates Sudoku puzzles games with 3 levels of difficulty (consult the text sources for this project), where each puzzle should be different than the rest • Serves each client by generating and passing to the client a Sudoku puzzle of the selected level of difficulty • Presents a solution to the given puzzle at the end of the game, if it is terminated without success by the players. • Records statistics in a file about the client username, the level of difficulty and the outcome of the game (total time played and game solved/ unsolved result).

# Reviewing the subject

## 2.1. Simple definitions, concepts and algorithms

In GameParts I am using lambda expressions and Lists to access the cells on the board.

In the UndoAndRedo the information is released by two Stack containters. The first one is for the cells to get back using undo and the other to save undone cells for the redo operation. For valid operations redo stack is cleaning itself.

## 2.2. Finding the problems and the difficulty of the problem

The biggest problem with the game is connecting to the RMI server. ERROR java.rmi.NotBoundException: Sudoku. This error came on the last moment and could not find a way to solve this problem.

## 2.3. Methods for solving the problems

# Making the project

## 

## 3.1. Architecture and diagrams

The architecture of the project is divided by 3 parts: Server, Client and helpers for cleaner code.

The server contains 3 files:

* Iserver – Interface for the server
* Server - The actual server
* StartServer – Implements the interface

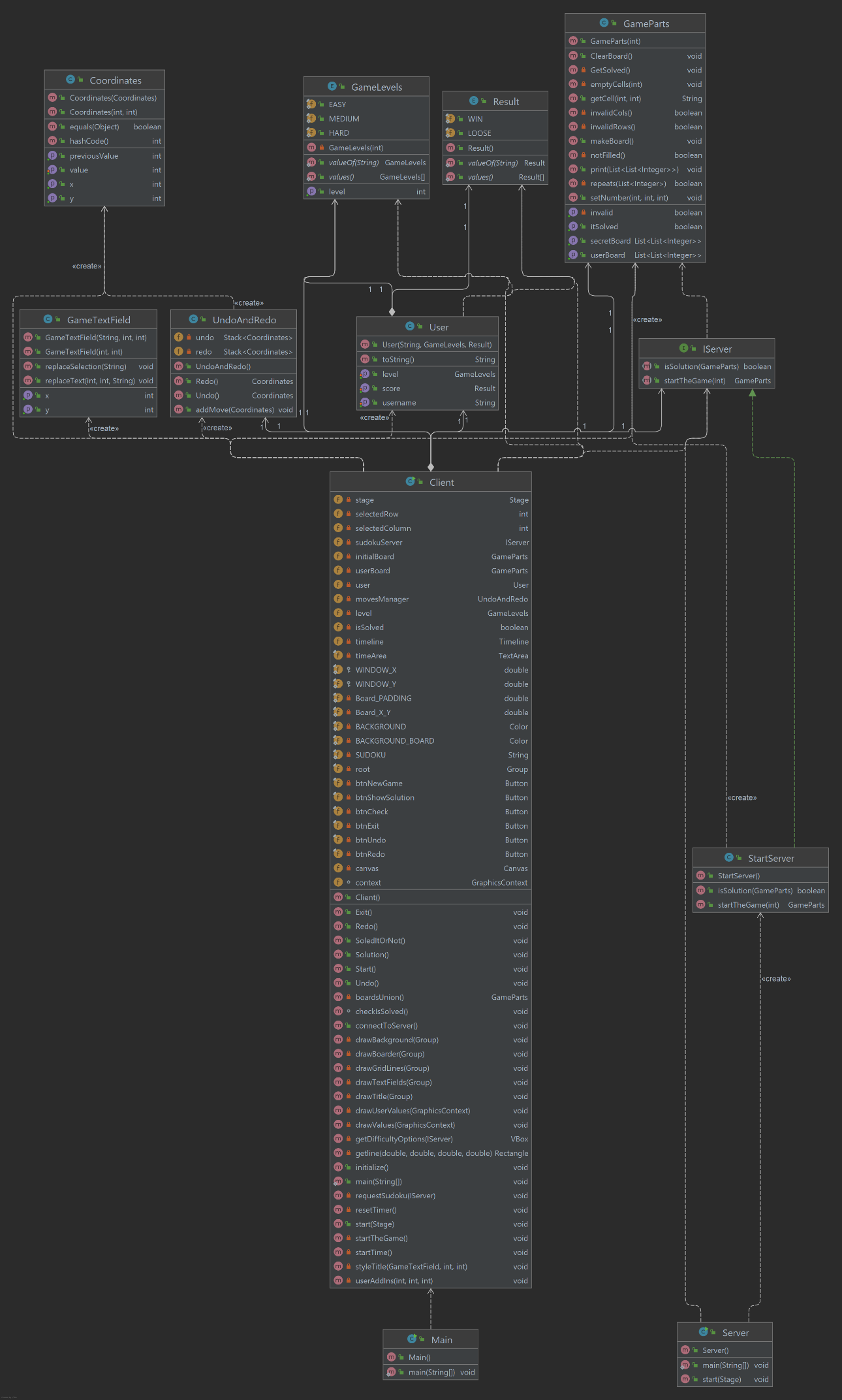
Helper Package is a package with basic parts of the game. As:

* Coordinates- basically represents the cells of the Sudoku with their x and y coordinates and the value of the cell
* GameLevels – enumerable type representing levels with their blank squares
* GameParts- represents the game board. Here we can find the functions that are checking if the game is finished
* GameTextField – this is basically the class TextField but redefined to escape undefined behavior
* Results- enumerable that represents the results. The only purpose is cleaner code
* UndoAndRedo – this class makes undo and redo buttons functional

The client contains the most important parts of the program:

* User- all information about the user
* Client- graphic parts, buttons, connecting to server

The main function is separated just for esthetic and maybe for easy compilation.



# 

# Testing and realization

## 4.1. Realization of classes

Shown in the upper UML diagram

## 4.3. Planning and creating testing scenarios

The testing is made with ready Sudoku solver. <https://www.sudoku-solutions.com/>​. All test are made before making the server is added.

# Conclusion

## 5.1. Review and future plans

For future plans the game can have a random option which is to get a random of difficulty level. Also interactive pictures to congrats your winning and be sad when you lose. Also to solve the most important problem: CONNECTING TO THE SERVER.

# Literature

1. Programming Sudoku
2. <https://www.youtube.com/watch?v=qH9mWpYMtYU>
3. Course from Ryan Kay. Check out his YouTube channel: <https://www.youtube.com/c/wiseAss>
4. Евгений Кръстев, Lecture11c.pdf, Lecture14bFX.pdf