# Homework Documentation

## Internet Laboratory

**BMEVIAUA048**

**Spring 2017**

# TicTacTooo


**Made by:**

| | | |
|---|---|---|
| **Edvárd Róbert Bayer** | **NOBXBW** | **bayeredi@gmail.com** |
| **Richárd Csáky** | **BNLUAG** | **ricsinaruto@hotmail.com** |
| **Dániel András Dobos** | **N44ZMM** | **dobos.andras95@gmail.com** |
| **Péter Majoros** | **MZEGDJ** | **majorpeti007@gmail.com** |

# 1. Description

As homework, we decided to write a new version of the well known game, the TicTacToe. The peculiarity of our implementation is that the players do not play the game on the same device. Both of them can use their own devices, and they are connected via internet. This way they can play even if they are not near each other or while they are traveling. We wrote not only a computer program, but an Android application as well. Furthermore, the communication via internet is independent of the platform, thus we can further broaden the range of the compatible devices. The solvable tasks included making both the java and android version of the program, working out how to implement the game logic in code, and making the server updates automatic.

# 2. Requirement analysis

There were a couple of requirements that we set out to accomplish from the inception of the idea. First of all probably the most important one was that we had to make the communication with the server platform independent and automatic, so users don't have to press extra buttons in order to update and get data from the server. Another important requirement was the idea that the user can't get the program in a state where it stops working, which we mostly achieved by rigorous testing and bug fixing. Finally we wanted to make the GUI as clean and intuitive as possible.

# 3. Specification

**About the Game in General**

The game is played by two players, one of them is the O, the other one is the X. The board consists of 3*3 fields. The next player in line can chose one of the empty fields, where he wants to place his marker. The goal of the game is to achieve three markers of our own in a line (in a row, in a column or in one of the diagonals) before the other player. If one of the players has achieved it, the game ends. The game also ends, if there are no more empty fields, in this case it is a tie.

**Computer Program GUI:**

When we start the program, we can see the board with the 3*3 fields, a START button in the lower left corner and a RESTART button in the lower right corner. We can change our user name before starting a game by writing our new name in the TextBox in the upper right corner. Additional information can be read throughout the game above the board. At the end of the game, the result is displayed at the bottom of the window between the start and the restart buttons.

**Android Application GUI:**
When we start the application, we can see the following: the title of the game at the top of the screen, the board in the middle of the screen, a TextBox to write in our new name under the title, a START button next to the previous and a RESTART button at the bottom of the screen. The opponent's name can be read under our own name, and additional information is displayed under the board. At the end of the game, the result is displayed at the bottom of screen in a toast message.

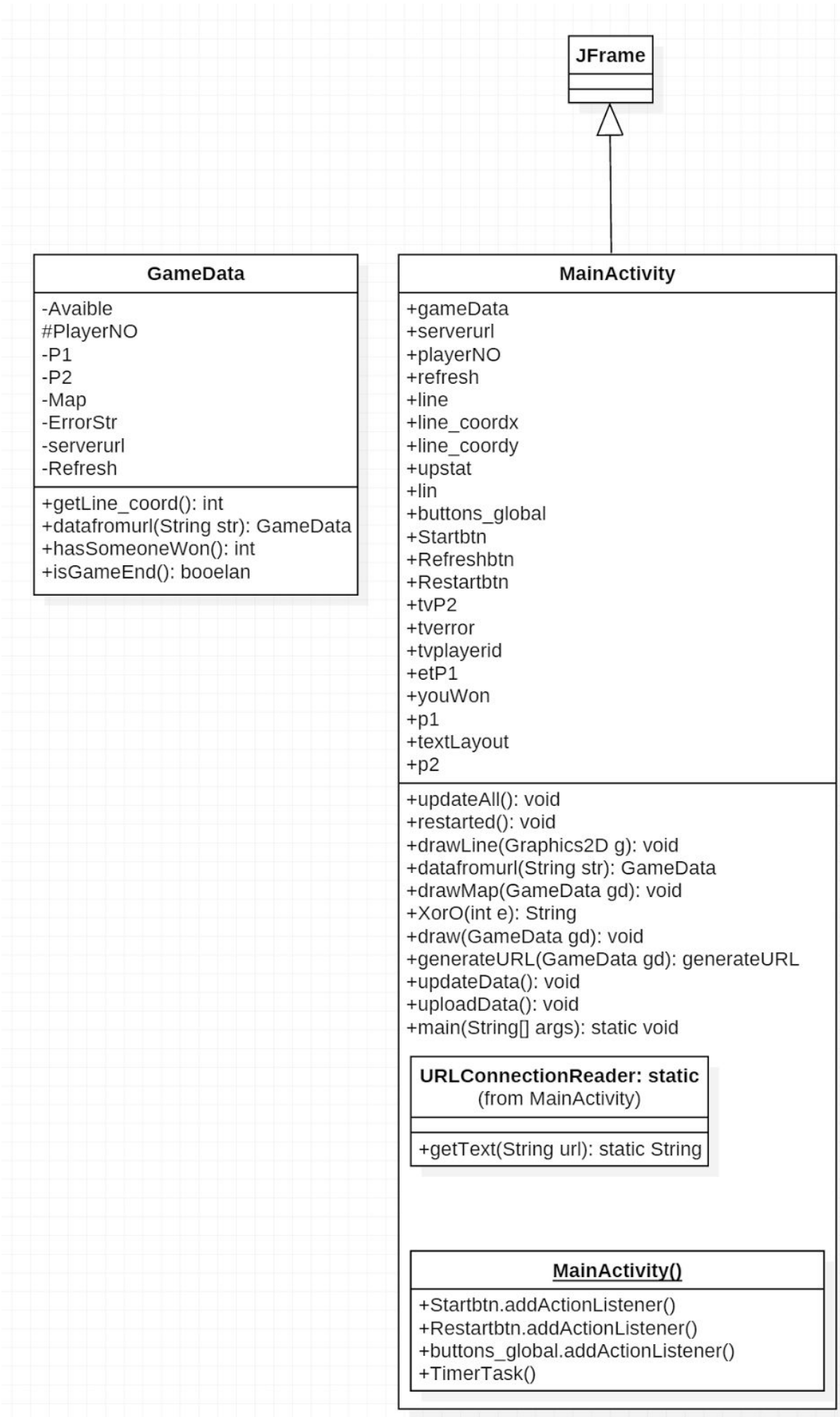
**Usage of the Program**
By pressing the start button we can apply for a game. If someone has already applied for a game, the game starts otherwise we have to wait for someone else to apply as well. Only one game can run at a time. The beginner is the one who applied first. The players can place their markers intermittently. When the game ends, the program displays the results. By pressing the restart button, we can ask for a rematch, if the opponent also restarts, we can start a new game with the start buttons. Restart can also occur during a game, if it does, the game cannot be continued, and after the other player also hit the restart button a new game can be started.

## 4. Alternatives for solving the tasks

There are a lot of ways to solve this task, we only list a few.
As for the design it is possible to use lines and circles, instead of using buttons to indicate the board grid.
The internet communication could be done in other ways, too. Google's firebase service provides a good solution for it, for free, but other ways of communication are also possible.

## 5. Design

The design of the application is quite simple, and elegant on both platforms. Due to the game's simplicity we used default android buttons/jbuttons. It provided an accurate click, and also gives the developers a lot of opportunity for further customizations, since the style, and size of the buttons can be easily changed. The design of the program architecture and the classes used can be seen below in UML format.


JFrame
GameData
-Avaible
#PlayerNO
-P1
-P2
-Map
-ErrorStr
-serverurl
-Refresh
+getLine_coord(): int
+datafromurl(String str): GameData
+hasSomeoneWon(): int
+isGameEnd(): booelan
MainActivity
+gameData
+serverurl
+playerNO
+refresh
+line
+line_coordx
+line_coordy
+upstat
+lin
+buttons_global
+Startbtn
+Refreshbtn
+Restartbtn
+tvP2
+tverror
+tvplayerid
+etP1
+youWon
+p1
+textLayout
+p2
+updateAll(): void
+restarted(): void
+drawLine(Graphics2D g): void
+datafromurl(String str): GameData
+drawMap(GameData gd): void
+XorO(int e): String
+draw(GameData gd): void
+generateURL(GameData gd): generateURL
+updateData(): void
+uploadData(): void
+main(String[] args): static void
URLConnectionReader: static
(from MainActivity)
+getText(String url): static String
MainActivity()
+Startbtn.addActionListener()
+Restartbtn.addActionListener()
+buttons_global.addActionListener()
+TimerTask()

## 6. Implementation

### Communication between clients

For communicating between two instances of the programs (two player), we set up a server, which was both readable and writeable. This server appears as a webpage, with a single line in it, which is written or read by the game clients, and this line contains every information about the current state of game (whose turn it is, players' usernames, the Xs and Os position, and whether someone requested a restart). The only information which has to be constantly stored in the program is, the player's ID, and every other information is accessed from the server, in every second, using timers in both the PC and the Android version.
The server information is only changed, when one of the player pushes a button, and after that this player can not make any change to the server, until the other player has interacted with it. There is one exception from this, and that is requesting a reset, this can always be made, and the other player can only answer to this with a reset, too.

See implementation code in the appendix.

## 7. Instructions for testing

Start two instances of the program, after writing in the name of the players, click on the START buttons, and the game begins with the player who clicked on the START button first. Place the markers one after the other until the game ends. The server can be reseted by the two players. One of them requests a RESTART, and the other one accepts it by also clicking on the RESTART button. This can be tested after the game has ended, or during a game.

## 8. Team management

The division of the work wasn't planned beforehand, but rather based on voluntary contributions. As we stepped forward in development there were always parts of the program that needed work or problems to be solved, and someone always offered. This way we didn't have to force anyone to do something he didn't want to.
Daniel coded the main part, and the base of the program in Android, by implementing the game logic using the GameData class. He also did the first version of the server communication. Richard implemented Daniel's code in Java, making the first version of the Java program. Furthermore, he worked on bug fixing and the GUI of the Java version. Peter extracted the string resources in the android

version so that they are not hardcoded in the code, rather defined among the other resources. This way it is easier to make the application multilingual in the future. Furthermore, he wrote the functions (hasSomeoneWon, isGameEnd and endGame), which handle the end of the game. Edvard finished the work on the game logic by implementing what happens at the end of a game. He also updated the server communication to be automatic in both versions of the program.

## 9. Summary and conclusions

We managed to create a cross platform version of a classical game, which works seamlessly. However, there is place for further improvements as listed below.

## 10. Further plans

By extracting the string resources we made it easy to make the android application multilingual in the future. Furthermore, the communication with the server is platform independent, thus we could further broaden the range of supported platforms (e.g. iOS, Windows Phone etc.)

A major shortcoming of our current implementation is that, only one pair of players can play at the same time, however it wouldn't be hard to solve this problem using a more sophisticated approach to the server side of the program.

*"A software is never finished, only released"*

## 11. Appendix

### PC - MainActivity.java

https://github.com/ricsinaruto/Eclipse/blob/master/TicTacToe2/src/GameData/MainActivity.java

### PC - GameData.java

https://github.com/ricsinaruto/Eclipse/blob/master/TicTacToe2/src/GameData/GameData.java

### Android - MainActivity.java

https://drive.google.com/file/d/0B5_71w6KIiZDQjFtaEY0enpZaWM/view?usp=sharing

### Android - GameData.java

https://drive.google.com/file/d/0B5_71w6KIiZDMTc5X2d2UXkzZmc/view?usp=sharing


TicTacTooo
NOW on both
Android and PC
TicTacTooo
START
Android
PC
You are O
Waiting for first player
O has won the game
RESTART
YOU LOST