Design Report for TicTacToe

Fall 2016

Team

Team name:

n00bs

Team qualifications:

All students in Reykjavik University in School of Computer Science.

Team members:

- Berglind Ósk Einarsdóttir
- Drífa Örvarsdóttir
- Ísak Snær Sigmarsson
- Júlía Oddsdóttir
- Ólöf Gyða Risten Svansdóttir
- Sigurbjörg Rós Sigurðardóttir
- Tinna Þuríður Sigurðardóttir

Introduction

This document is a design report made for the course T-303 Hugbúnaðarfræði at Reykjavik University. It describes the design decisions made to implement the game TicTacToe. Testing was done using Test Driven Development (TDD). The goal of the project is to focus on the infrastructure and best coding practices. Implemented features is not a high priority in this project of team n00bs.

Design

First step was to design a console version of the game but with the option of taking it to a web-ui implementation. It was decided to have the simplest version of Player versus Player playing the game. Used Markdown for content format.

Class diagram

A class diagram (a UML) was put together in the beginning of the project to get an overview of the structure of the system. The program Creately was used to draw the diagram. There are 5 classes.

UI layer

```
class DisplayBoard
    + display()
```

Business layer

```
class GameController
    + Player _playerX
    + Player _playerO
    + startGame()
    + clearBoard()
    + printBoard()
    + getWinner()
    + getLoser()
```

Entity classess

```
class Player
    + int count
    + char sign
    + position()
    + getSign()
class Board
    + char boardArray[][]
    + getBoard()
    + initializeBoard()
    + getPosition()
    + setPositon()
    + makeMove()
    + checkStatusOfBoard()
    + hasWinner()
    + checkFull()
```

Data layer

```
class db
    + int winsX
    + int winsY
    + addWins()
```

Implementation

Team members used Google Docs to go over assignment requirements and to keep track of information and decisions throughout the design phase. GitHub was used for version control.

Testing

Testing was conducted using Travis-CI (Continous Integration server), Travis runs all gradle build and compile tasks automatically after each push and tracks the changes made to seperate branches, promting a pull request when merging said branches.