# Developer Manual – Scrabble 13

Author: Maximilian Netzer

## Introduction

Scrabble13 is a program for playing a game of scrabble alone, with AI bots or friends. This manual is intended for developers who are interested about Scrabble13 internals. The manual will cover the functionality and structure of Scrabble13.

Scrabble is a board-and-tile game in which two to four players compete in forming words with lettered tiles on a 15x15 square board. Words spelled out by letters on the tiles interlock like words in a crossword puzzle. Players draw seven tiles from a pool at the start and replenish their supply after each turn.

Our goal was to implement a software solution, which allows the users to dive in a game of scrabble with an appealing and responsive user interface, whilst the background of the game runs smoothly without any issues.

## Structure

In order to achieve our goal, we divided our project into three major parts.

The first part is the **game logic**. It contains the components needed to play scrabble, such as the board or the tiles, as well as different types of players and a game class. Within the game class we implemented the components and the rules. Thus, when the application is running and a player makes a move, the system can check if the move was correct, and if so, save it. The different player classes are used to create players for the game and to play with AI players. Furthermore, the game logic contains other logical components such as a dictionary, a scoreboard, and an overtime watch.

Since the game should be possible to play with multiple players through a server, the second part in our project are classes to establish the **network connection**. This is structured so that there is a client and a server which are connected to the game logic and the user interface. Both have their protocol to handle incoming and outgoing messages between the server and a client. In addition, this part defines the different messages which are used.

The third part is the **user interface (UI)**. Building the graphics for the application we used JavaFx, which is a toolkit for developing client applications. The basis are the different views of the game, such as the welcome view or the game view. Each view has its own fxml-file, that contains the

graphical information of the view. Furthermore, each view has its own controller class. These controller classes are used to initiate the views, interact with the content shown in the views, switch between the views and ultimately allow the user of the application to make moves and play a game of scrabble. Also, the controllers are connected to the network classes and the game logic to fill the graphical containers with the needed information and allow the user to play scrabble.

The foundational design pattern is a "model-view-controller"-pattern. The game logic represents the model and creates the data to be displayed, the views handle the presentation of the data, and the controllers define the user interface's reaction to user input. The network connects multiple instances of the application.

Besides those three major parts of the project there are some supporting functions:

The **file transform** package contains classes to load and save data, such as local player profiles, sound files, and dictionary files.

The **client** package is used to start the application and initialize needed components, like the graphical user interface (GUI), the network client, a local player and player profiles. It marks the start point of the application and forms a basis on which the game later invokes. Finally, all packages respectively parts had to be integrated. The client creates a common foundation so that all parts can access each other.

## Outlook

During this project we developed an application that gives the user the basic means to play Scrabble. Our focus was to keep it simple but also guarantee a great user experience. In future we would like to expand this experience. A good starting point would therefore be to give users more options to customize the app according to their preferences, for example with custom colors of the app, animations, and custom sounds. Also, we do not want to settle for the performance of the application. There's always room for improvement. Therefore, we have the target to improve the responsiveness of the game.