

Software Development (202001064)
Programming (202001066)

Programming Report – Scrabble

Hanna Hamielec, s-2617730, h.hamielec@student.utwente.nl

January/2022

Overall Design

1.1. Requirements Implementation Map

Designed system presents a simple game of Scrabble, allowing exactly two people to play with each other. Each player after starting a game is present with textual representation of a board and is able to start playing this game. Each player is able to connect to server on chosen host number and port, where after joining using unique usernames they are added to queue waiting for other player to join before the game is started. In presented unfinished final version of this project not all requirements are met. Players after being connect to a server and starting new game, are presented with board but are not able to play together over a server. Game is playable only locally.

Handling possible errors that could occur, like incorrect input or lack of server response was done by creating new exception classes. All exceptions like that are located in `game.exceptions` package.

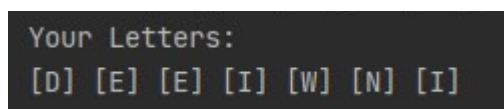
1.2. Model-View-Controller Implementation

More complex entities like 'Player', 'Board' or 'Game' impellent interfaces of corresponding names, while simpler ones like 'TileBag' or 'Tile' doesn't. View of this Board is implemented in `src/game/BoardView`. `boardPrint.java`, that allows to print a board based on given parameters. Implemented in the class methods allow to build a textual view of the board using frame elements/lines. Method `formatBoard()` is the one responsible for putting those elements and return the finished board as a single string. Method `printBoard()` in `Board.java` class connects this view.

Each part of the project is separated in different packages in `src/game`. All interfaces needed for entities like Board or Player are in package `game/interfaces`, while classes implementing them are in the `game` package directly. Package `game/protocol` is made for networking part of the project and includes 3 separate packages: `protocol` (containing interfaces), `server`, `client`, and each of them included related to their names classes. Both server and client have their views implemented in respective 'clientTUI' and 'serverTUI' classes.

1.3. User Interface

User interface for the project is in a form of text-based user interface allowing client to clearly interact with system. Besides visible in form of pictures presented below elements, other interface parts like error prompts, messages from server, announcing winner are represented by a single line printed to console including all needed for the situation information.



```
Your Letters:
[D] [E] [E] [I] [W] [N] [I]
```

Figure 1: TUI representation of player's letter rack

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	
1	null	null	null	null	null	null	null	null	null	null	null	null	null	null	null	1
2	null	null	null	null	null	null	null	null	null	null	null	null	null	null	null	2
3	null	null	null	null	null	null	null	null	null	null	null	null	null	null	null	3
4	null	null	null	null	null	null	null	null	null	null	null	null	null	null	null	4
5	null	null	null	null	null	null	null	null	null	null	null	null	null	null	null	5
6	null	null	null	null	null	null	null	null	null	null	null	null	null	null	null	6
7	null	null	null	null	null	null	null	null	null	null	null	null	null	null	null	7
8	null	null	null	null	null	null	null	null	null	null	null	null	null	null	null	8
9	null	null	null	null	null	null	null	null	null	null	null	null	null	null	null	9
10	null	null	null	null	null	null	null	null	null	null	null	null	null	null	null	10
11	null	null	null	null	null	null	null	null	null	null	null	null	null	null	null	11
12	null	null	null	null	null	null	null	null	null	null	null	null	null	null	null	12
13	null	null	null	null	null	null	null	null	null	null	null	null	null	null	null	13
14	null	null	null	null	null	null	null	null	null	null	null	null	null	null	null	14
15	null	null	null	null	null	null	null	null	null	null	null	null	null	null	null	15
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	

Figure 2: TUI view of empty board

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	
1	null	null	null	null	null	null	null	null	null	null	null	null	null	null	null	1
2	null	null	null	null	null	null	null	null	null	null	null	null	null	null	null	2
3	null	null	null	null	null	null	null	null	null	null	null	null	null	null	null	3
4	null	null	null	null	null	null	null	null	null	null	null	null	null	null	null	4
5	null	null	null	null	null	null	null	null	null	null	null	null	null	null	null	5
6	null	null	null	null	null	null	null	null	null	null	null	null	null	null	null	6
7	null	null	null	null	null	null	null	null	null	null	null	null	null	null	null	7
8	null	null	null	null	null	null	null	D	R	A	I	N	null	null	null	8
9	null	null	null	null	null	null	null	0	null	null	null	null	null	null	null	9
10	null	null	null	null	null	null	null	6	null	null	null	null	null	null	null	10
11	null	null	null	null	null	null	null	null	null	null	null	null	null	null	null	11
12	null	null	null	null	null	null	null	null	null	null	null	null	null	null	null	12
13	null	null	null	null	null	null	null	null	null	null	null	null	null	null	null	13
14	null	null	null	null	null	null	null	null	null	null	null	null	null	null	null	14
15	null	null	null	null	null	null	null	null	null	null	null	null	null	null	null	15
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	

Figure 3: board with placed letters

Testing

2.1 Unit Testing

Classes Board.java, Player.java and Scoring.java were all partially tested using JUNIT in specially designed test classes (included in src/test package) during implementation process.

Class Player.java deemed as the most complex and important one was the only class with most of its methods tested separately in unit test. This class was tested alongside TileBag.java, as proper functioning of TileBag.java class was needed for methods of Player.java to work correctly.

Board.java and Scoring.java were tested individually in created for the purpose of testing classes. Both Board.java and Scoring.java were tested in isolation using JUNIT methods.

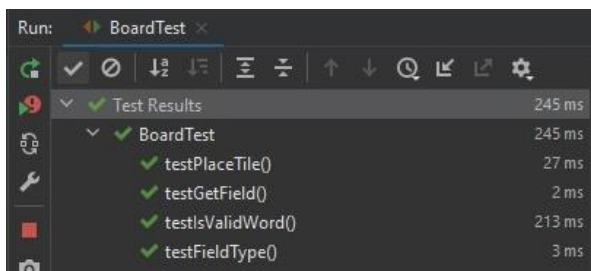


Figure 4: results of test for Board.java

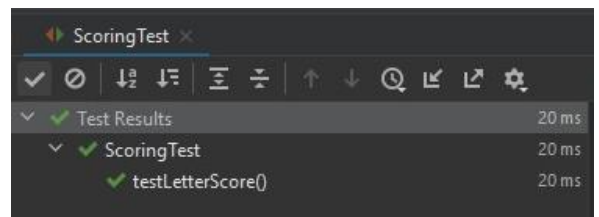


Figure 6: results of test for Scoring.java

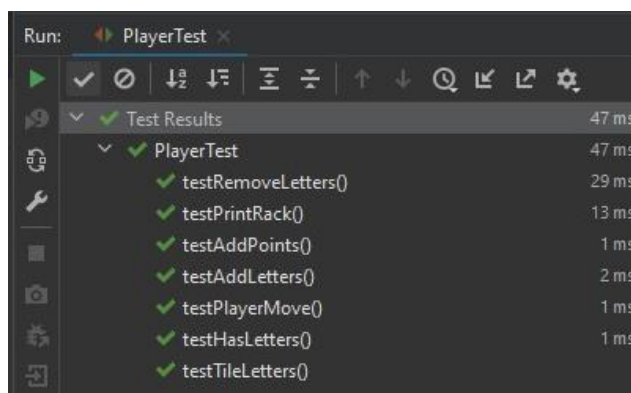


Figure 5: results of test for Player.java

90% classes, 81% lines covered in 'all classes in scope'

Element	Class, %	Method, %	Line, %
test	100% (4/4)	100% (14/14)	100% (92/92)
game.Board	100% (1/1)	35% (6/17)	41% (23/55)
game.Player	100% (1/1)	75% (9/12)	84% (32/38)
game.Scoring	50% (1/2)	16% (1/6)	8% (5/56)
game.TileBag	100% (1/1)	50% (1/2)	97% (129/1...
game.TileBag	100% (1/1)	50% (1/2)	97% (129/1...

Figure 7: test coverage for all mentioned above classes

Additionally class `additionalTests.java`, included in the same package, was used to test if TUI view of the board and players rack is printed correctly. More over few other methods (like parsing user input of board's positions used to make move) were also tested. Mentioned methods were testes by adding a main method that creates instances of the class under test, and then invoked serval methods on these objects, while TUI was tested by visual inspection of printed elements.

Test results for visual representation of user interface and methods described above were both successful.

2.2 System Testing

Since final game that was supposed to meet all requirements isn't finished only system testing that was conducted was testing separate methods that were being developed at the time, by creating a new connection between 2 clients and server and later creating a game.

First tested functionality was checking if client is able to connect to a server on chosen port and host number, later on functionality was expanded and tested if user can connect to server and join using unique name.

Beside mentioned above cases, playability of the game (not over server) and cooperation of different classes was tested by creating a separate class `Scrabble.java` and having one person playing the game. During this test bug surrounding validation of words and player being bale to use tiles they don't own were found.

Academic Skills Chapter (Hanna Hamielec-s2617730)

5.1 Time Management and Procrastination Avoidance

At the very beginning of the module I was able to efficiently manage my time without falling behind on any part the material. In the first 2 weeks I was able to finish all my tasks in time without procrastinating. Unfortunately later on during weeks 3-4 I started to fall beginning on programming lab exercises. Although I had a plan and tried to stick to it, due to circumstances like quite uncooperative design lab pair which caused me to work alone on design related exercises and in process spending less time than I planned on programming.

Later on my work load became steady again, after getting my request about transferring grades for design part of module and no longer having to follow it, I was able to catch up on lab exercises, and during winter break catch up with math part of this module. I was quite motivated to continue working and was successfully progressing with studying without falling too much behind.

I focused more on programming exercises and tried to fix my group work with my programming pair, although the communication was quite good, some important parts like my pair falling behind on lab exercises were revealed only in later weeks. Although this lack of group work didn't affect my progress with programming labs parts of course, but it did affect project related work later on.

My time management for project wasn't as good as that for previous week of the module. After being left alone without any response from my programming partner I started to feel overwhelmed by upcoming exams while having to work on project and finish lab exercises, which caused me to be less concentrated on my tasks and procrastinate in middle of work. I was able to develop a project alone, but due to procrastination at the beginning and having to work alone I was not able to finish and meet all needed requirements. Thing that contributed the was poorly splitting my time into certain tasks, spending too much time on creating the user interface and too little on making server and client, was not a good thing to do.

5.2 Strengths and Weaknesses

During the module I was able to practice and boarded my java knowledge. Having less or more of previous knowledge about topics of each week was a big help for me and was main reason lack of collaboration didn't further negatively affect my progress. Topics of unit testing, handling exceptions, inheritance and object oriented programming are my strength. I feel quite confident having to create a program using parts relate to any of these topics and doing so won't take me way too much time.

On the other hand parts like collections or networking are my weaknesses and I still have to spend more time trying to practice them. Although I understand the theory behind those two topics and remember how to use them, having to implement more in depth related programs to those two topics takes me more time than other tasks. My final version of the project reflects this, creating server – client connection is a big part of this project, and is the only functionality that does not fully work, because I left it as very last thing to implement (which was a mistake to do). Although those few days during which I started to implement server-client connection for the project helped me a lot to understand networking better. And I believe if I started working on this part earlier I would be able to finish it on time.

To better improve I believe I should not only plan further before but also learn how to manage my time in case of unexpected situation that could possibly negatively affect my plans. By doing so I would be able to focus better on topics that are harder for me to grasp, by distributing my workload according to given situation rather than trying to follow previously made plans that now won't fully work.

5.3 Checkpoint Meetings

During this course I was able so self-asses my progress which let me better track and organize what was happening and what did I learn during the certain period of time. Although I don't feel like it had any noticeable impact on my progress. Reflecting and looking back on made progress with self-assessments didn't influence my learning much beside highlighting what I should focus more on and what I already can do, but even without this I was able to tell which parts of the learning goals are my weaknesses and on which should I spend more time.