Miķelis Putnieks
Student of Advanced Technology, year two
S2637846

Report Software Design – Project Scrabble

Discussion of Overall Design

First some handy information for whom it may concern:

Class **model.TileTray.java** is written according to the principles of programming by contract.

Class **model.TileTray.java** was tested using **tests.TileTrayTest.java**

Package **view** as well as classes in package **model** contain the GUI elements of the game.

The following **additional features** are present: Team-play, GUI

The MVC was harder to implement due to the GUI, yet eventually all the classes are nicely separated controlled by the **main.Controller.java** The classes in the **main** package are responsible for starting the application, the control of the game on the client side and relaying information and events between other classes. Most important of which, the **main.Controller.java** connects the networking parts with the model parts and acts as an intermediary when events from the GUI view must affect the model or trigger some network messages. Class **networking.Client.js** is very closely interconnected with the class **main.Controller.java** and together these control the non-server part of the application. This part is responsible for displaying the updates received by client, allowing interaction with GUI and translating that to messages sent by the **Client.js** to the, where they are interpreted by the **ClientHandler.js** and sent to **Server.js** for handling.

Server.js is responsible for hosting the current game, validating moves and incoming client information, updating the state of the game accordingly, and broadcasting the valid update information to all Clients. Server supports that a game may have as many clients as wanted.

Testing

The class **model.TileTray.java** was tested using **tests.TileTrayTest.java** yet when the GUI was implemented, I was unable to get the JUnit testing to work again due to the classes being too intertwined. The whole application would have to be run for a single Junit test to be completed.

Extensive system testing was done after each milestone in development. Especially during the final, networking, part of the project, many errors and bugs were found and fixed. Like when a player has placed some tiles on the board to see if he/she can arrange a word, but decides to end his turn instead. In this case tiles left on the board were often being overwritten (and thus deleted from the entire game on this players side) by the ones just received and placed by the servers broadcasted MOVE command. This particular issue was fixed by removing any non-fixed tiles left on the board after the player ends his/her turn. These tiles were not simply removed, but added to the players TileTray as, otherwise, there would be a loss of tiles, and at some point towards the end of the game this would cause off-normal behaviour at least, application crash at the worst.

Despite the endless work, some errors are still present – the ones that have been identified are:

- Double blank tile usage during a single move leads to application crash due to the pop-ups not being adjusted for such rare case.
- Blank tiles might, in rare cases, cause inaccuracies on the server side.
- If a server is started when another game server is already running from the same computer, the application might crash.

Software Metrics

Academic Skills Chapter

My plan to build the Scrabble project was:

- Start of the project – at week 8. This week will be spent by getting the basic game logic sorted out.
- Add GUI visuals to the previously implemented game logic.
- By the end of week 8 – have a working TUI version with one player playing scrabble.
- Starting week 9 – Spend the week implementing networking, plan is to have 4 clients on one server which is being hosted by one of the clients.
- By the end of week 9 there should be only minor bugs to fix or visual GUI improvements to make.
- Week 10 will be spent by writing the report, doing extensive testing and bug fixing, as well as implementing visual GUI improvements.

Now, the work on project commenced only on week 8 as I spent week 7 finishing off the programming exercises. Despite this setback I felt confident about being able to catch up with the rest of students and build the Scrabble game with a wonderful GUI in only 3 weeks' time.

When making the plan my experience from design project cautioned me to be a pessimist. I realized that when planning a project, one must assume that it will consume more time, attention, and will be more challenging than expected. This way, failures can be avoided and enough time will be scheduled for everything. After all, if something gets finished early – that is no problem.

During the design project, having been a part of a "busy bee" group where we held each other accountable for the progress we made, planning was easy to execute, and it truly felt like a teamwork. Sadly, as my Scrabble was a single person undertaking, I found it much harder to stay dedicated. Yet I still managed, on most of the days.

Starting at Week 8, I managed to code much of the basic functionality and classes in two days' time, which was way faster than expected. Sadly, this soon changed as I added the GUI functionality to the model components, so that I could use the GUI to test the game logic. GUI took the rest of week 8 to implement, and still there were some issues. This was due to it being a subject out of the scope of this course, so I had no preparation or knowledge of it. Also, I initially went with a wrong approach of distributing graphics through many classes that then needed to be tied together with awkward method calls. Soon the mess was so large, it required a rework.

Nevertheless, having done that, I went on with the game logic, which, looking back, was definitely the most challenging part of the project. The sheer difficulty of working out how to first extract move information from my GUI, then process and validate it, and then finally execute the move and again display it on GUI was mind-bending and took me much of Week 9. Again, GUI was the

problem, but I had put in too much work already to drop it now. Finally, by the end of this, it was time to code a basic loop and connect the game logic, such that it could be played by one player - no networking just yet.

Next step, however, was networking. More precisely, to make time for implementing it I ended up grinding through Week 7 exercises in one day, 12h work shift, and then proceeded to use the knowledge and parts of the code in the Scrabble project over the following 3 days. This was relatively straight forward, no major issues, yet it required a lot of system level manual testing.

As a result of the delays, this whole report was written during the last day of the project.

Note to future self when planning projects: Never be excited, never be optimistic, and for the sake of your own sanity, beautiful as it is, never decide to implement GUI where a TUI could do!

Having said all this, if I were to support next year's students, I would hand them the following advice:

- Do start the project on Week 7, even if only a small bit, as that will bring you awareness and build a mental model of how much and what is ahead, as well as give time over the weekend to think about the design choices and structure of the project.
- Try to approach every coding session with a clear set of goals in mind – make a small to-do list of which class you will complete, or which methods write and test during that session.
- Do not be shy to ask for help, and to discuss your progress with your course mates, it helps to grasp and solve problems many times faster than if one reads the book or internet.
- Never panic and give up if you get stuck on a particular issue, as that might as well be the hardest bit in the project. After that it only gets easier. For me this was the methods for checking the validity of moves, which due to my GUI implementation took me almost a week of figuring out. Who would want to give up right at the peak when climbing over a mountain...