Ludwig-Maximilians-Universität

SoftwareEntwicklungsPraktikum

SEP

Rummikub-Currygang

Author(s):
Cedrik Harrich
Till Kleinhans
Johannes Messner
Hyunsung Kim
Ella Mayer
Angelos Kafounis

Supervisor:
Nicolas Brauner

8. Januar 2019

Inhaltsverzeichnis

1 Introduction

In this course we need to implement a boardgame called Rummikub.

1.1 Rules

The Rules for this game are found either on the website [1] and/or on Wikipedia [2]
.

2 Tasks

In this chapter we will list all our duties that have to be fulfilled in the next 3 Weeks.

2.1 Specification

The requirements our program needs to fulfill are:

- All the rules seen on the Wikipedia page must be implemented. []
- It should be possible to start a game with 2-4 Players.
- The youngest player starts the game.
- The players proceed clockwise.
- If it is your turn you should have 2 options: Draw a card or lay one down.
- The GUI shall be implemented with JavaFX and work concurrently.
- Model-View-Controller must be used.
- It should be playable across a network with sockets.
- One of the players is the host and also holds the server.
- Information should be broadcasted as serialized Java Objects.
- Clients are sending their moves to the host.

- The host is checking for valid moves.
- If the move is legit the host updates the board and notifies all the other clients.
- If the move is not correct the player gets a notification and is able to try again.
- The host is configuring and starting the Game.
- Every Player is allowed to leave the game. (What about the host?)
- Port: 48410 needs to be used.
- When the game is over every Player gets a Notification who won the game and the host is asked if he wants to start another game.
- One can close the program with the usual x-Button. Optionally via the menu or a Shortcut.
- The whole Program needs to be confirm with Google Java Style (or at least JavaDoc and Comments are right)
- Unit Testing for Logic part
- Final game was played at least 50 times without errors.
- Edge case tested

2.2 Diagrams

We do not only need to implement the program but we also need to have a good documentation that is up-to-date all the time. The central parts of the documentation will be:

- The product requirements document (this document)
- The MVC-class diagram
- The sequence diagram(s)
- The GUI-Design Template

3 Organisation

The offical platform for our project will be our gitlab account:
https://gitlab.lrz.de/ru96vik/rummikub—currygang

3.1 Communication

Since we have to use a lot of different platforms we decided to mangage them with a central platform called Trello. Just to name a few Platforms we intend to use: Google Drive, WhattsApp, GitLab, OneNote, Moodle, Slack etc.
The informal communication will be handled with WhattsApp.

3.2 Programming Specifications

We decided to use Java SDK 11 and IntelliJ for our project. Moreover we will try to format our Code according to the Google Java Style Specification.

4 Goals and Milestones

In this chapter we will define our main goal and dismember it into smaller Milestones with a workload of approximately a week.

4.1 Definition of Done

We are done with our project, if:

1. We have a working and running program.
2. All of the rules from section 1.1 are implemented.
3. Our program fulfills all of the requirements from section 2.1.
4. The documents from section 2.2 are complete and match our end result.
5. The program was tested for its main functionalities and was played at least 50 times.

6. The executable program and all the needed documents are uploaded to our GitLab account.
7. Our program was reviewed and approved.

4.2 Milestone 1: 21.12.2018

The workload will be:

- LRZ-Gitlab Repo for the project
- invite our Tutors
- First designs of the diagrams.

4.3 Milestone 2: 04.01.2019

The workload will be:

- adjust diagrams
- Choose Design pattern
- Prototype Implementation

4.4 Milestone 3: 14.01.2019

The workload will be:

- Define our Definition of Done.
- Workplan (Trello)
- First version of our program with a list what is possible right now.

4.5 Milestone 4: 21.01.2019

The workload will be:

- Second version of our program with a list what is possible right now.

4.6 Milestone 5: 28.01.2019

The workload will be:

- Second version of our program with a list what is possible right now.
- List of planned/possible features

4.7 Milestone 6: 04.02.2019

The workload will be:

- Check all the Specifications
- Test the program
- Update the diagrams and documents
- JavaDoc and clean up code.
- Implement Features.
- Ship final working product, diagram and tests
- Prepare for Presentation.

5 Functionalities

The final product should be a complete Rummikub Game that is playable over a network. To model our game UML Diagrams come in handy.

5.1 UseCase Diagrams

There are basically 2 different phases in this game. First of all you have to start the program and set all the needed information like host or guest, name, age etc. as you can see in figure: 1.

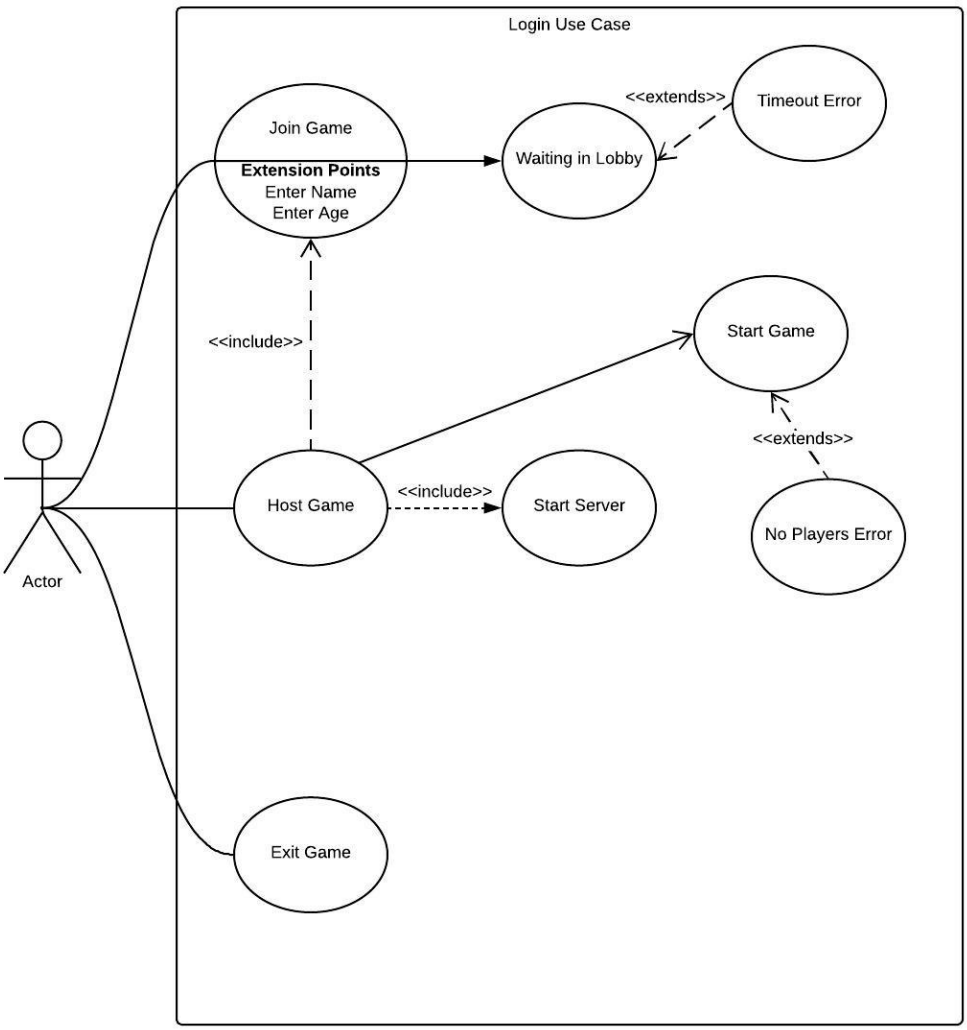

Abbildung 1: Use Case for the login screen interaction

The other standard phase is the acutual game where every player has the same options every turn as seen in figure 2.

Abbildung 2: Use Case that shows what options every player has in every turn

5.2 Class Diagram

Since we know the different Use Cases we can now derive our needed classes, methods and relationships and put them all together in a Class Diagram (fig: 3)

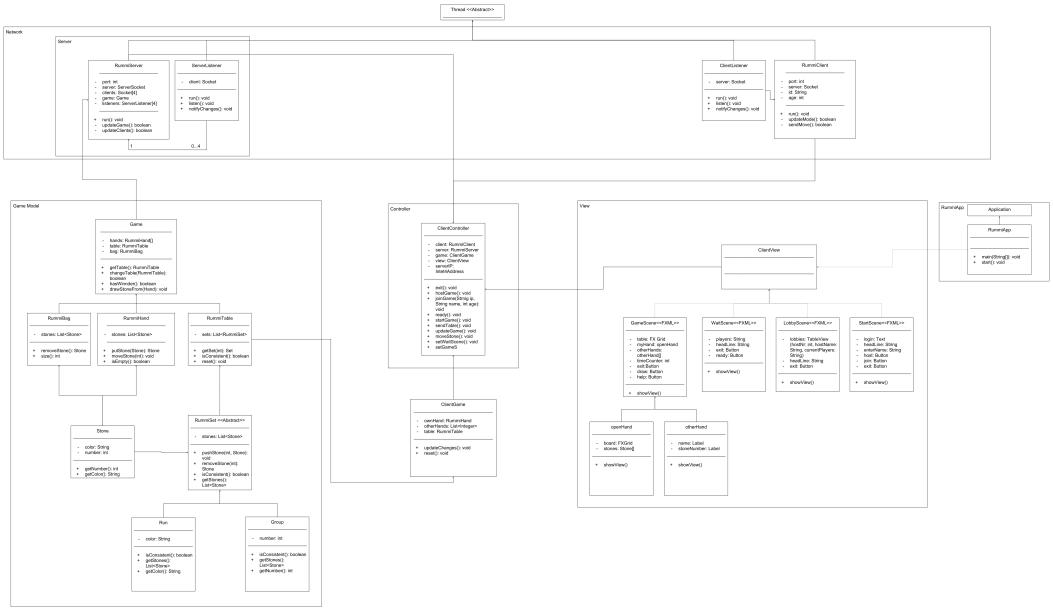

Abbildung 3: This class diagram describes all the classes, methods and relatonships we have to implement.

5.3 Sequence Diagram

To make the relationships more clearly especially how classes interact with each other in a time perspective we also made a sequence diagram. (fig: 4)

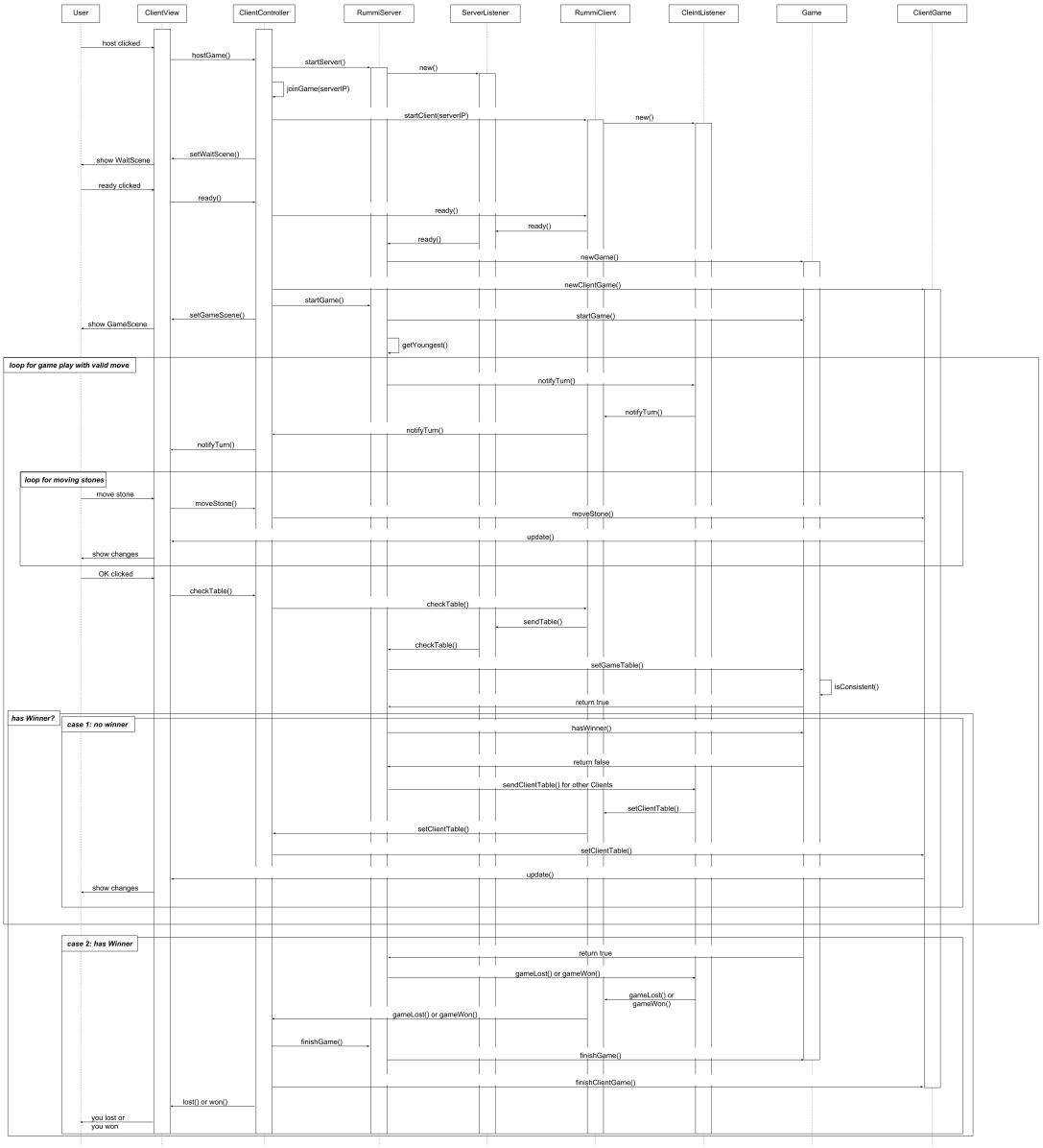

Abbildung 4: This sequence diagram should give insight how the actions are performed over time and how they interact with each other.

6 GUI

In this section we present the user interface. In the following figures you can see the different screens in the different phases of the game.

6.1 LoginPhase

The only purpose for this phase is to connect the players and asks for every information needed to start the game.

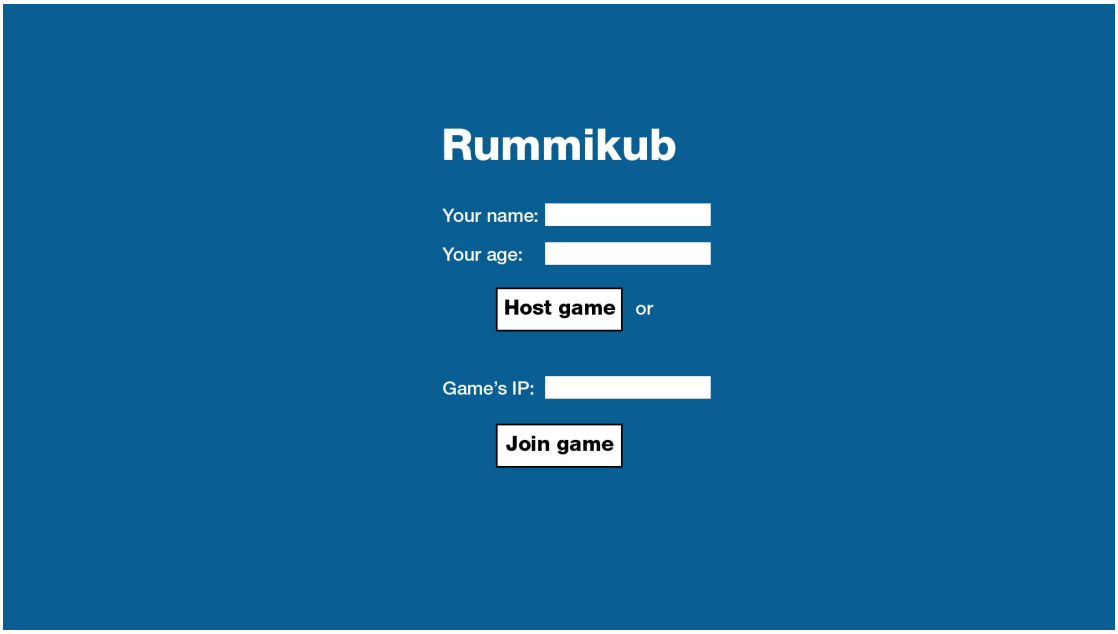

Abbildung 5: First Screen you will see after starting the program.

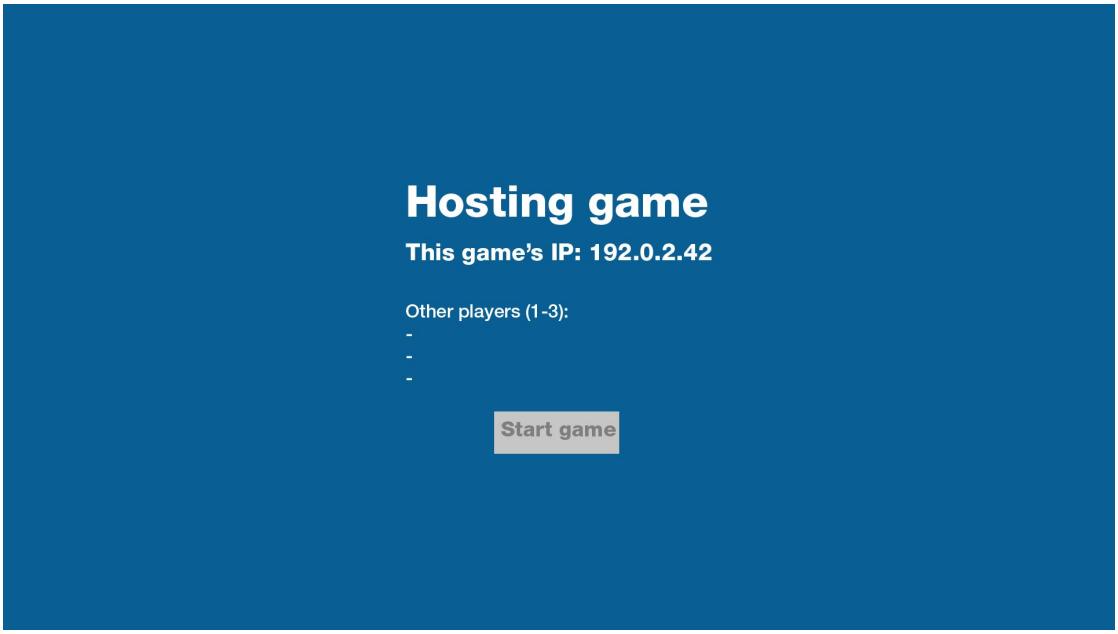

Abbildung 6: The screen you see right after clicking on host.

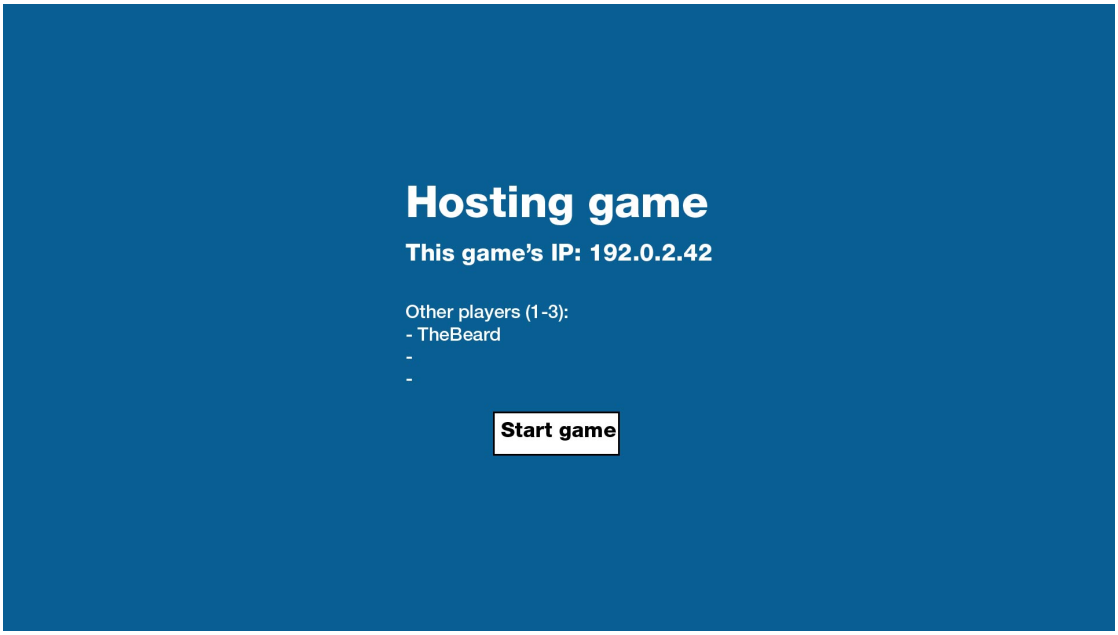

Abbildung 7: The screen the host sees when someone joined the game.

6.2 GamePhase

This phase displays the actual game.

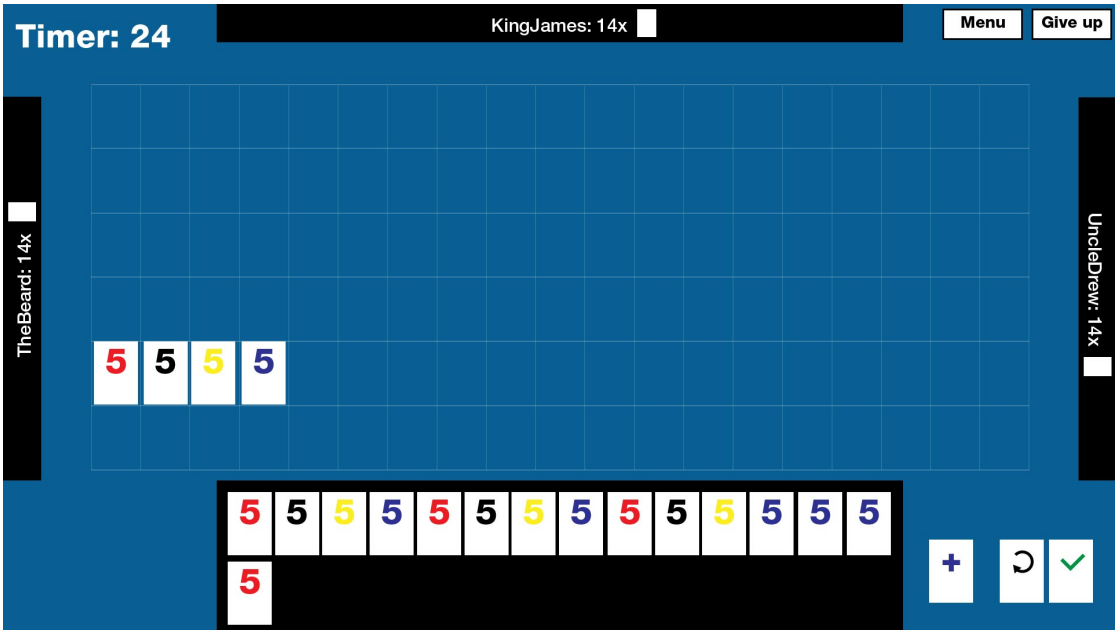

Abbildung 8: The screen you see when the host starts the game.

7 Verfication

In this section we will list all the methods used to check if our program works as planned. In each sub-chapter we briefly describe the used approach followed by a detailed documentation of our results.

7.1 Model Checking

7.2 Testing

8 Conclusion

Literatur

[1] D. Beyer. Praktikum ßep: Java-programmierung. https://www.sosy-lab.org/Teaching/2018-WS-SEP/. Accessed: 2019-01-08.

[2] Wikipedia. Rummikub. https://de.wikipedia.org/wiki/Rummikub. Accessed: 2019-01-08.