

LUDWIG-MAXIMILIANS-UNIVERSITÄT

SOFTWAREENTWICKLUNGSPRAKTIKUM

SEP

---

# Rummikub-Currygang

---

*Author(s):*

Cedrik HARRICH

Till KLEINHANS

Johannes MESSNER

Hyunsung KIM

Ella MAYER

Angelos KAFOUNIS

*Supervisor:*

Nicolas BRAUNER

14. Januar 2019

# Inhaltsverzeichnis

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Rules . . . . .	3
1.2	Progress . . . . .	3
<b>2</b>	<b>Tasks</b>	<b>4</b>
2.1	Specification . . . . .	4
2.2	Diagrams . . . . .	5
<b>3</b>	<b>Organisation</b>	<b>5</b>
3.1	Communication . . . . .	6
3.2	Programming Specifications . . . . .	6
<b>4</b>	<b>Goals and Milestones</b>	<b>7</b>
4.1	Main Goal/Definition of Done . . . . .	7
4.2	Features . . . . .	7
4.3	Milestone 1: 21.12.2018 . . . . .	9
4.4	Milestone 2: 04.01.2019 . . . . .	9
4.5	Milestone 3: 14.01.2019 . . . . .	9
4.6	Milestone 4: 21.01.2019 . . . . .	10
4.7	Milestone 5: 28.01.2019 . . . . .	10
4.8	Milestone 6: 04.02.2019 . . . . .	10
<b>5</b>	<b>Functionalities</b>	<b>11</b>
5.1	UseCase Diagrams . . . . .	11
5.2	Class Diagram . . . . .	13
5.3	Sequence Diagram . . . . .	16
<b>6</b>	<b>GUI</b>	<b>18</b>
6.1	LoginPhase . . . . .	18
6.2	GamePhase . . . . .	20
<b>7</b>	<b>Verification</b>	<b>20</b>
7.1	Model Checking . . . . .	21
7.2	Testing . . . . .	21



# 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]

## 1.2 Progress

[Update: 2018.12.24]

So far we created the LRZ-Gitlab Repo for the project. We made the first designs for the UML-Class Diagram and Sequence Diagram. Moreover we also have a UseCase Diagram and a design Pattern for the view. [Update: 2019.01.04]

We adjusted the diagrams split up our workload so we are able to start programming now. [Update: 2019.01.14]

We handled our workload with Trello and it worked good so far i.e. were able to stay in schedule. We are actually a little bit ahead. This is the first time we can present a list of features implemented so far:

- The Model/Logic part of the game is as good as finished. It needs to be tested though.
- You can use the main method in Shell so you can simulate a game in the command.
- You can start the main game window. You can draw Stoneboxes and drag Stone Boxes onto the grid.
- Two clients can connect to the server. The game starts and the clients can print/ get the stones on the table and the hand.
- The connection can be tested with the named Shell.

## 2 Tasks

In this chapter we will list all our duties that have to be fulfilled until the 4th of February 2019.

### 2.1 Specification

The requirements our program needs to fulfill are:

- All the rules seen on the Wikipedia page must be implemented. [1]
- 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 official 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 (Google Drive, WhatsApp, Git-Lab ...) we decided to manage them with a central platform called Trello. If you want to have a look on our workflow click on the following link and you will be given permission to join our Trello Project:

<https://trello.com/invite/b/1U3hahCs/fb97670dcfee266a0a97b2780cb0c58a/todo-liste>

If you just want to have a slight insight how we handle our work here is a screenshot:

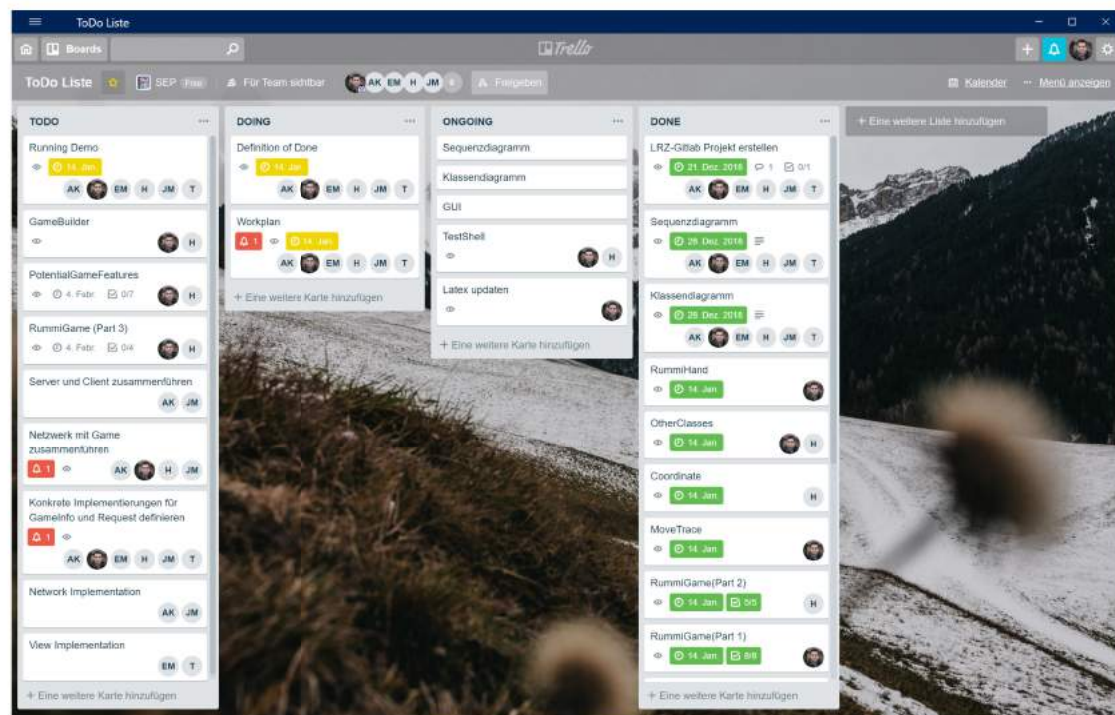


Abbildung 1: This is a small caption how we handle our workload in Trello.

### 3.2 Programming Specifications

We decided to use Java SDK 8 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 Main Goal/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. All the features from section 4.2 work.
4. Our program fulfills all of the requirements from section 2.1.
5. The documents from section 2.2 are complete and match our end result.
6. The program was tested for its main functionalities and was played at least 50 times.
7. The executable program and all the needed documents are uploaded to our GitLab account.
8. Our program was reviewed and approved.

### 4.2 Features

In this chapter we define the features we need to implement.

Model: Model Framework (Classes, Datatstructure, Interfaces and Methods)

Model: Creation of the Stones

Model: Handout Stones

Model: Laydown/Pickup Stones

Model: Move Stones on Hand



Model: Check for valid board

Model: NextTurn

Model: Undo Operation

Model: Players can join

Model: Start a game/TestShell

View: Creation of the Lobby Screen

View: Creation of the Game Screen

View: Host Game

View: Join Existing Game

View: Draw Stone

View: Move Stone on Table

View: Move Stone in Hand

View: Move Stone to Table

View: Undo

View: Quit Game

View: End Move

View: Error Handling if moves are not valid.

Server: Implementation of the UML Framework (Classes,Datastructure and Methods)

Server: Create Socket

Server: Create Listener for Client

Server: (De-)Serialiser (Server to Client)

Server: (De-)Serializer (Client to Server)

Server: GameInfoHandler

Server: RequestHandler

Server: Senders

Server: Listeners (for every Client)

### **4.3 Milestone 1: 21.12.2018**

The workload will be:

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

### **4.4 Milestone 2: 04.01.2019**

The workload will be:

- Adjust diagrams
- Choose Design pattern
- Prototype Implementation

### **4.5 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.
- The Model should be done and the Logic of the game should be implemented.

- The Framework for the server stands.
- We have the first features for our view.
- We implemented a TestShell so we can test our results without the need of the view part.
- Extra: We don't need to connect our three parts right now but we can already work on the first Interfaces.

#### **4.6 Milestone 4: 21.01.2019**

The workload will be:

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

#### **4.7 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

#### **4.8 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: 2.

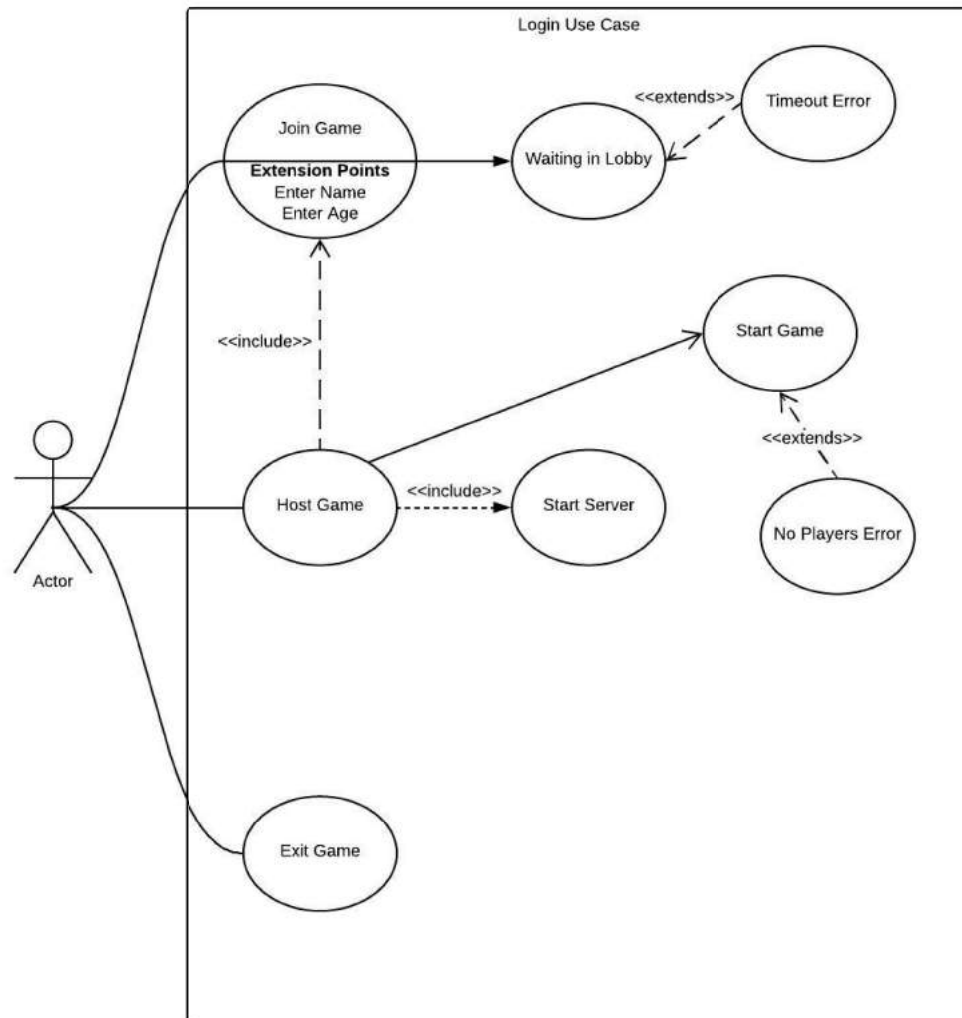


Abbildung 2: Use Case for the login screen interaction

The other standard phase is the actual game where every player has the same options every turn as seen in figure 3.



Abbildung 3: 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: 4)

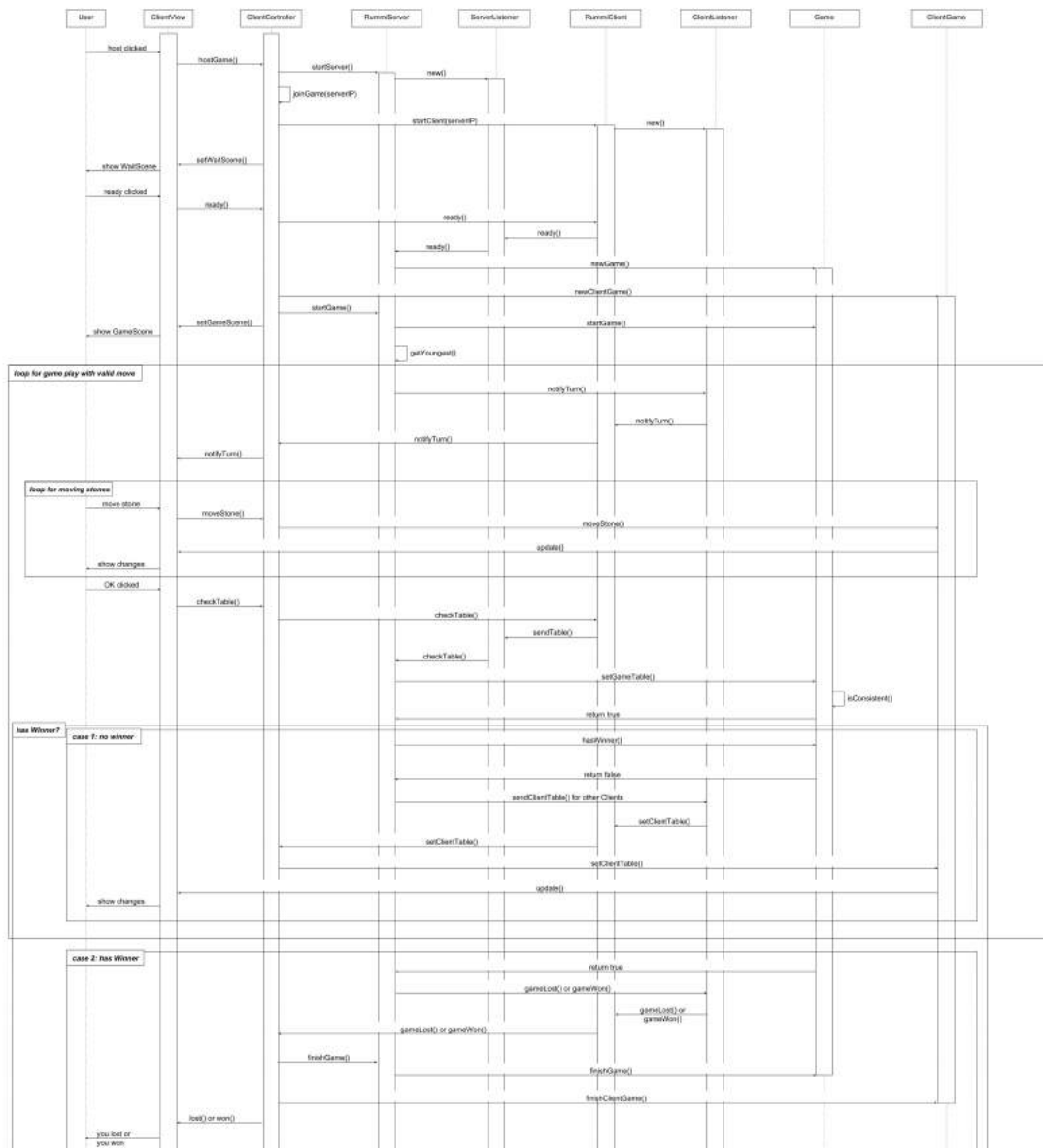






### 5.3 Sequence Diagram

To make the relationships more clearly especially how classes interact with each other in a time perspective we also made a sequence diagram. (fig: 6)

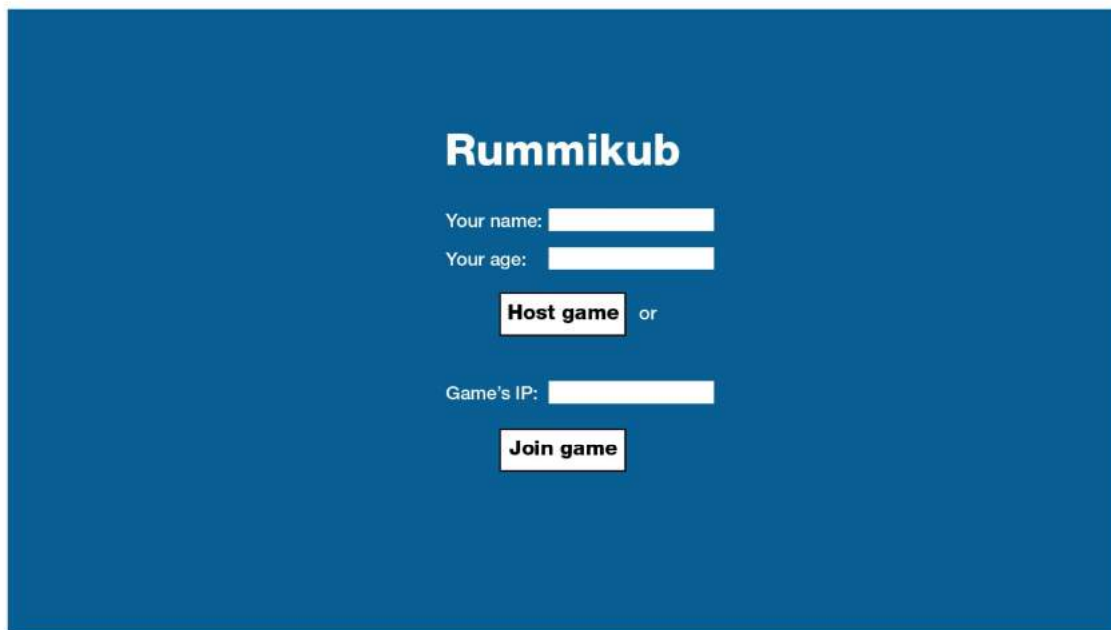


## 6 GUI

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.



**Rummikub**

Your name:

Your age:

**Host game** or

Game's IP:

**Join game**

Abbildung 7: First Screen you will see after starting the program.

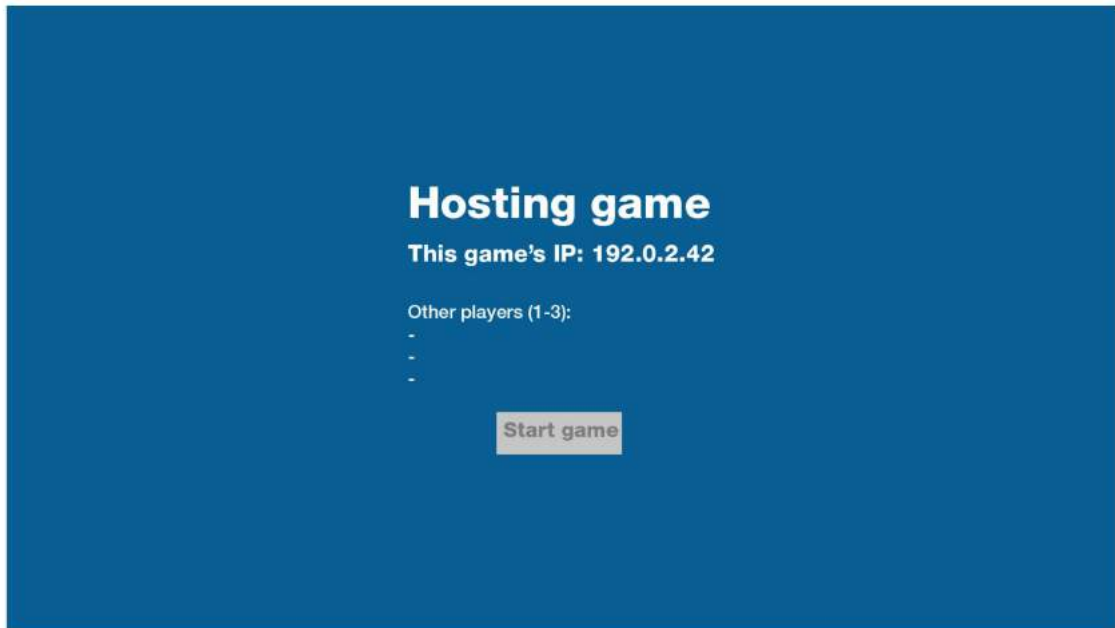


Abbildung 8: The screen you see right after clicking on host.

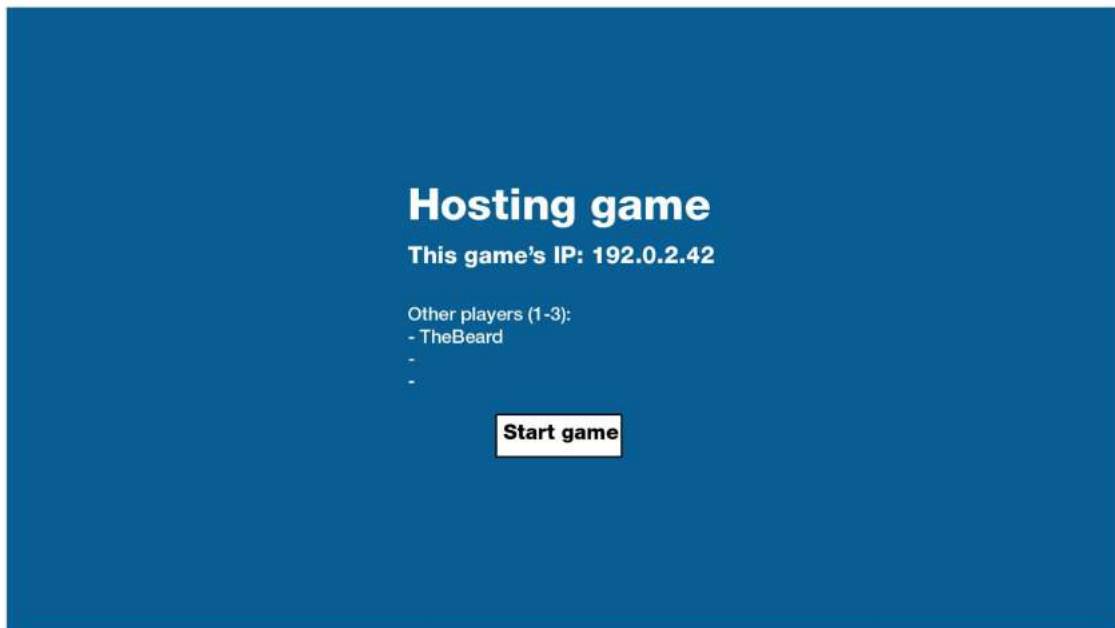


Abbildung 9: The screen the host sees when someone joined the game.

## 6.2 GamePhase

This phase displays the actual game.

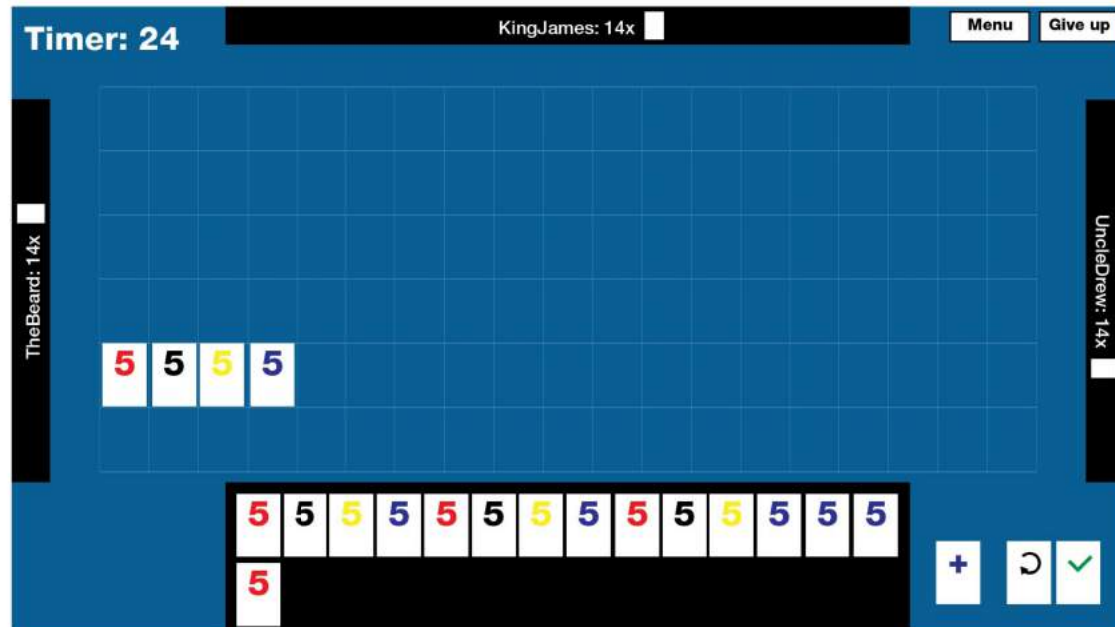


Abbildung 10: The screen you see when the host starts the game.

## 7 Verification

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 sep: 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.