

AG Computergrafik & HCI
apl. Prof. Dr. Achim Ebert
SEP/MP 2020

Exploding Kittens

Plannungsbericht

3. Juni 2020

Gruppe 9

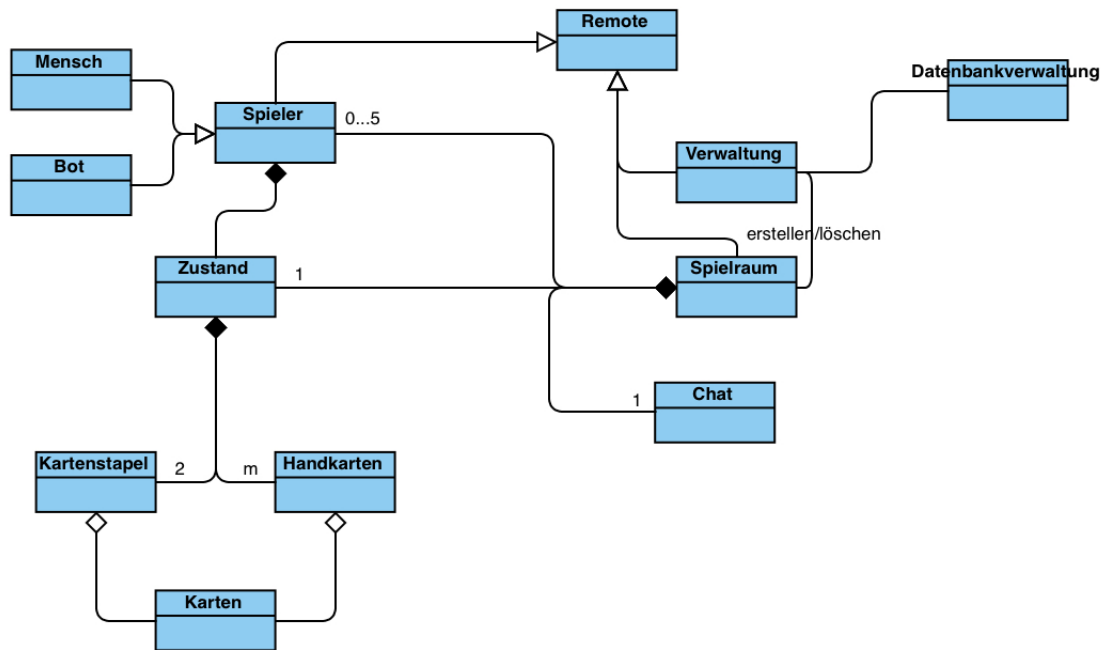
Ivana Drazovic
Merveille Kana Tsopze Mafo
Aurelle Daine Pellahe Wafo
Emilia Schreiner
Justus Will

Inhaltsverzeichnis

Inhaltsverzeichnis	2
0.1 Analysemodel	3
0.2 Klassendiagramme	4
0.3 Sequenzdiagramme	9
0.4 Exceptions	14
0.5 Bot	15
0.6 Gantt Diagramm	17

Analysemodel

Zuerst wurde mittels eines Analysemodells die benötigten Klassen und deren Verbindung ermittelt.



Klassendiagramme

Um die Implementierung zu erleichtern haben wir das Analysemodell verfeinert und alle Klassen mit ihren Methoden in Klassendiagrammen dargestellt. Später soll es für fast jede Klasse im Diagramm ein Interface geben. Bei der Implementierung ist darauf zu achten, dass manche Methoden (von Klassen die Remote implementieren) synchronized sein müssen. Dies ist nicht dargestellt, da Interfacemethoden diese Eigenschaft nicht haben können. Eine Übersicht kann mittels des Paketdiagramms [0.1](#) gewonnen werden.

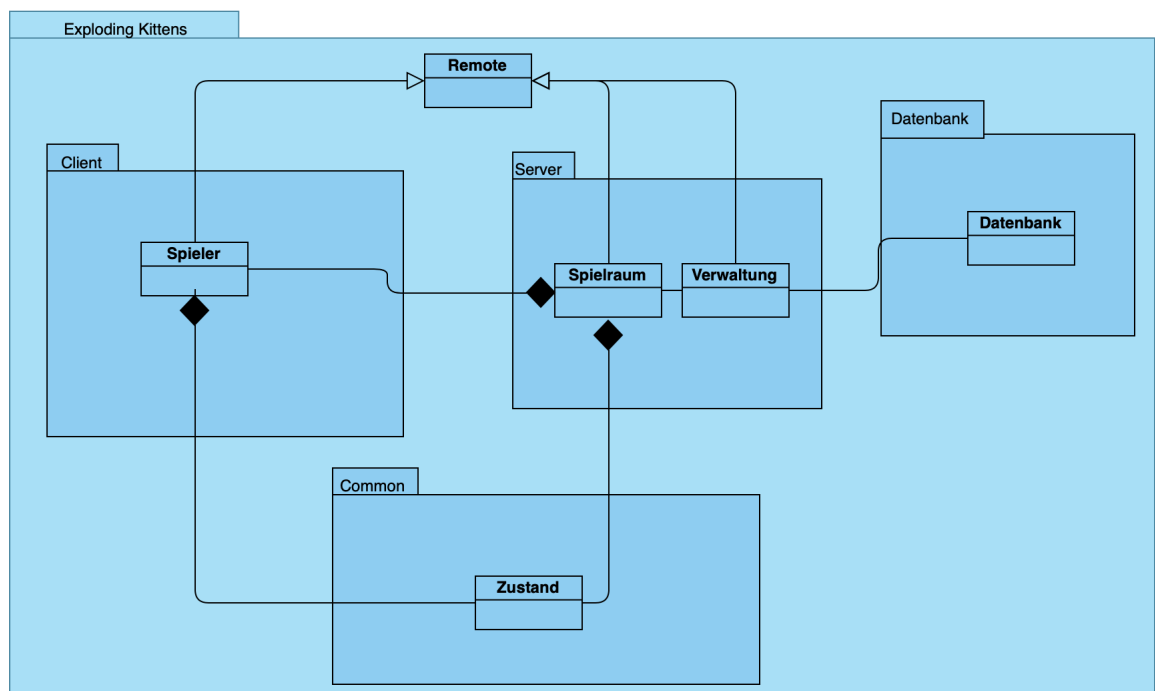


Abbildung 0.1: Neben Klassen in Client- und Serverpaketen gibt es auch Klassen, die von beiden Seiten gebraucht werden. Diese Klassen befinden sich im Paket Common.

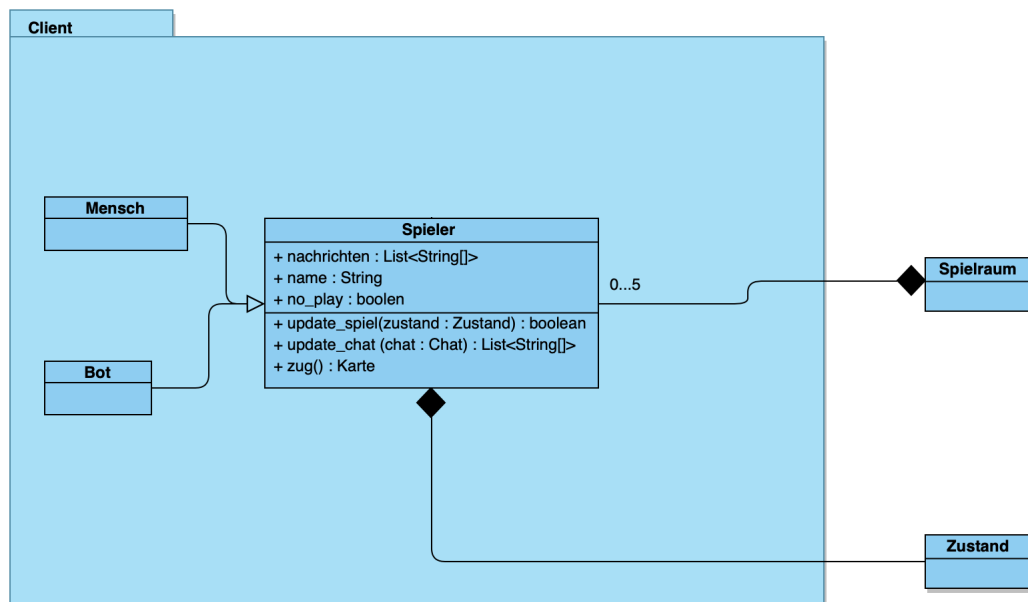


Abbildung 0.2: Clientseitig ist vor allem die abstrakte Klasse Spieler wichtig, die von Bots und Menschen implementiert wird

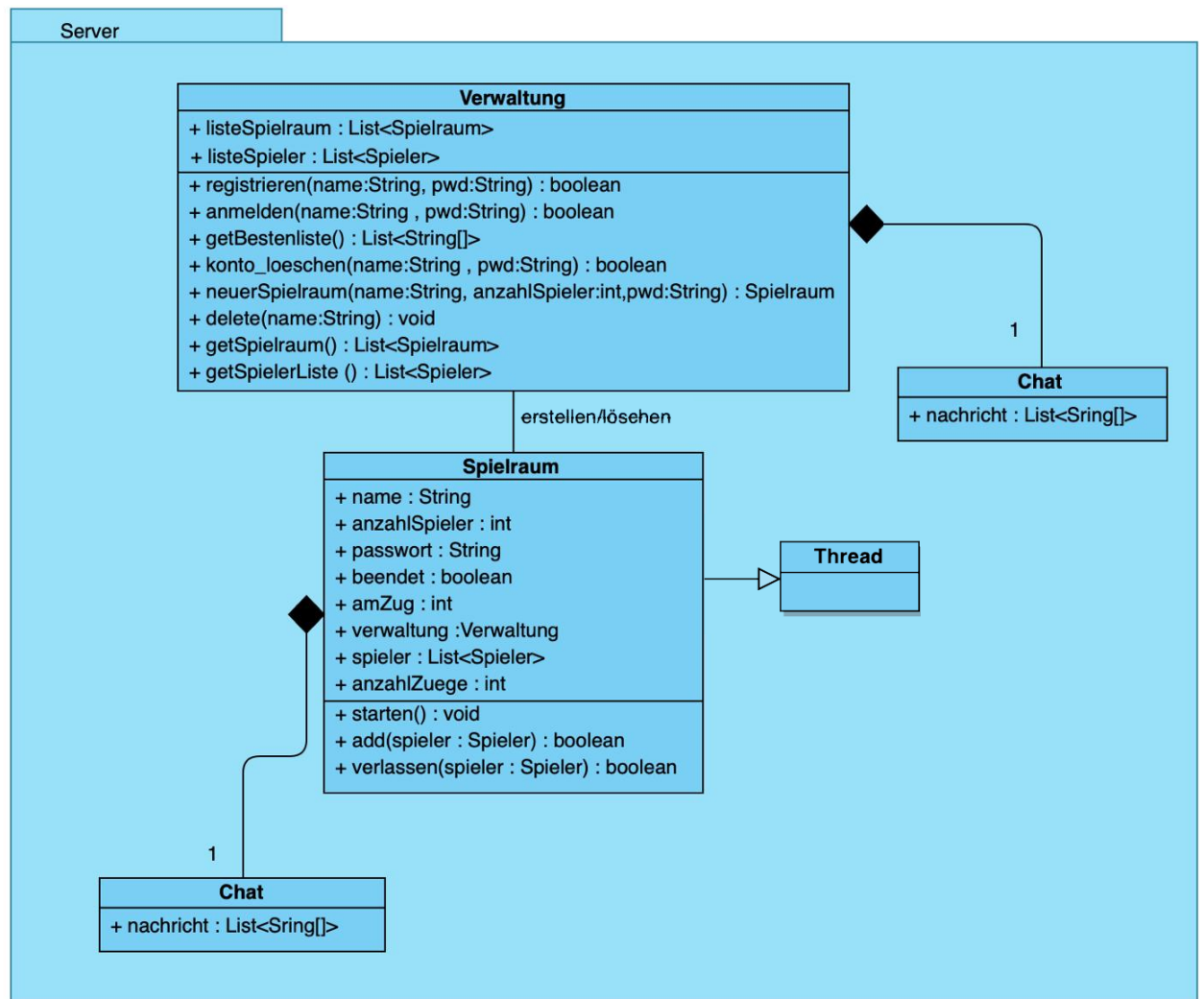


Abbildung 0.3: Neben einer Verwaltung, die allgemeine Aufgabe wie Login übernimmt, ist der Spielraum als Leiter des Spiels wichtig. Mehrere Spielräume können parallel laufen.

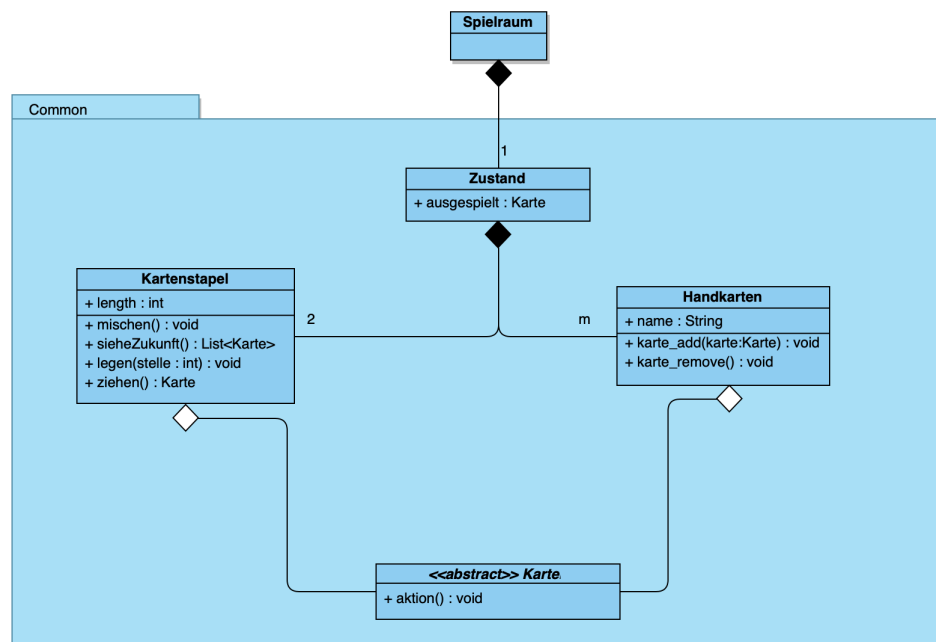


Abbildung 0.4: Die Spiellogik wird von beiden Seiten benötigt. Es gibt keine direkten Instanzen von Karte, siehe 0.5

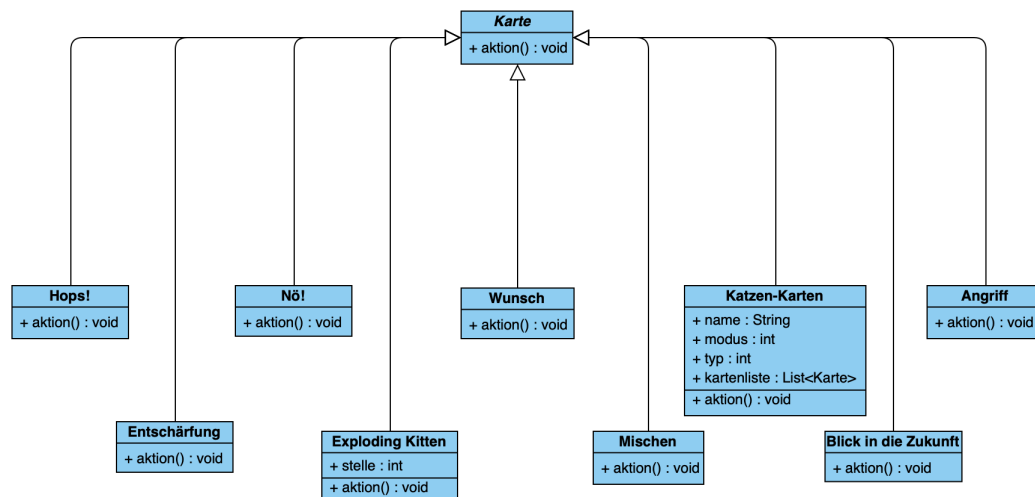


Abbildung 0.5: Alle Kartenarten erben von Karte. Da die Methode *zug()* von *Spieler* eine *Karte* zurückgibt, wir aber auch Kombinationen und das Zurücklegen einer Exploding Kitten modellieren, haben die jeweiligen Karten Attribute die für das Ausführen der jeweiligen Aktion relevant sind.

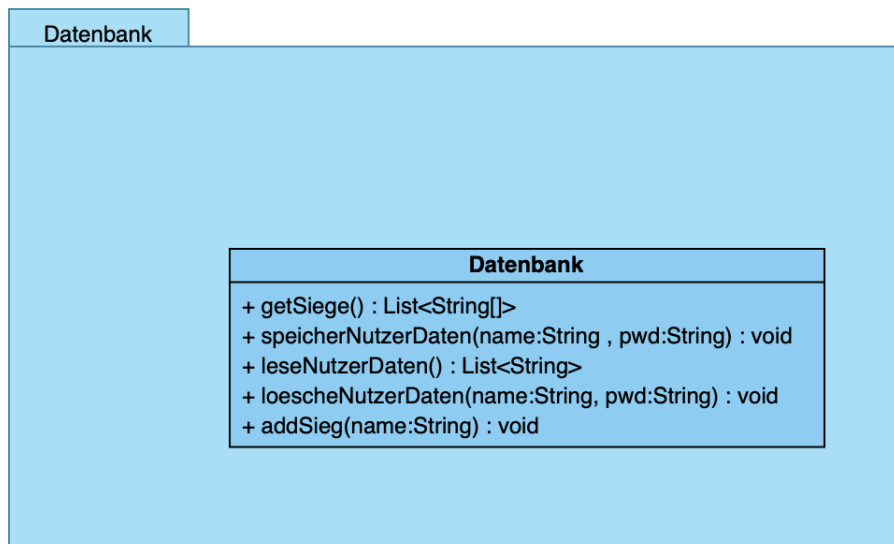


Abbildung 0.6: Die Verwaltung von Nutzerdaten und Bestenlisten wird von einem Paket geregelt, das mit JDBC zwei SQL Tabellen verwaltet.

Sequenzdiagramme

Um einen besseren Überblick über den Ablauf des Spiels, die Funktionsweise von *Spielraum* und die Kommunikation von Client und Server sind der Ablauf eiener Runde, sowie beispielhaft einige Interaktion mit der *Verwaltung* als Sequenzdiagramm dargestellt. Für genauere Angabe über die Fehler siehe [0.4](#)

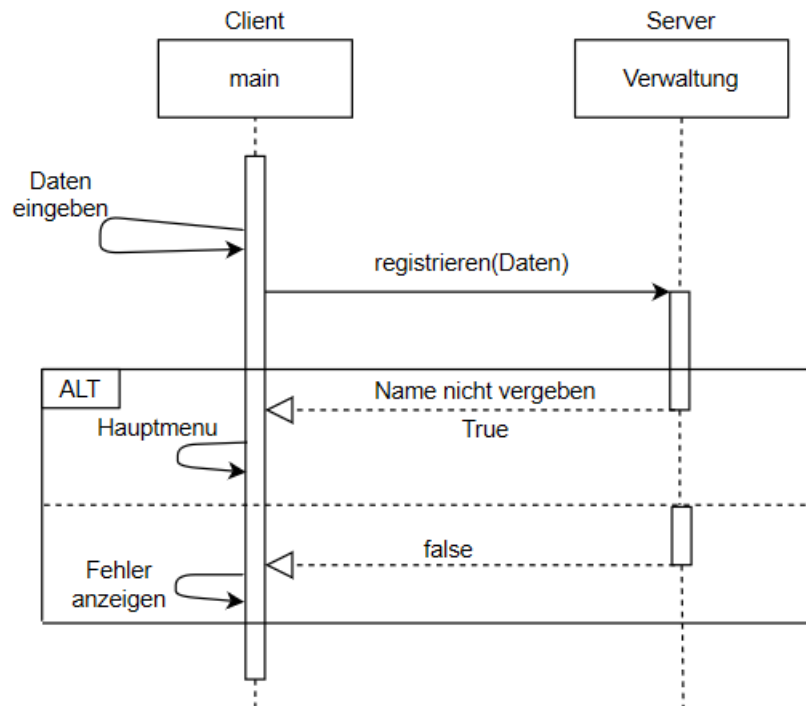


Abbildung 0.7: Falls bei der Interaktion keine Antwort kommt wird ein *NameFalsch* Fehler geworfen

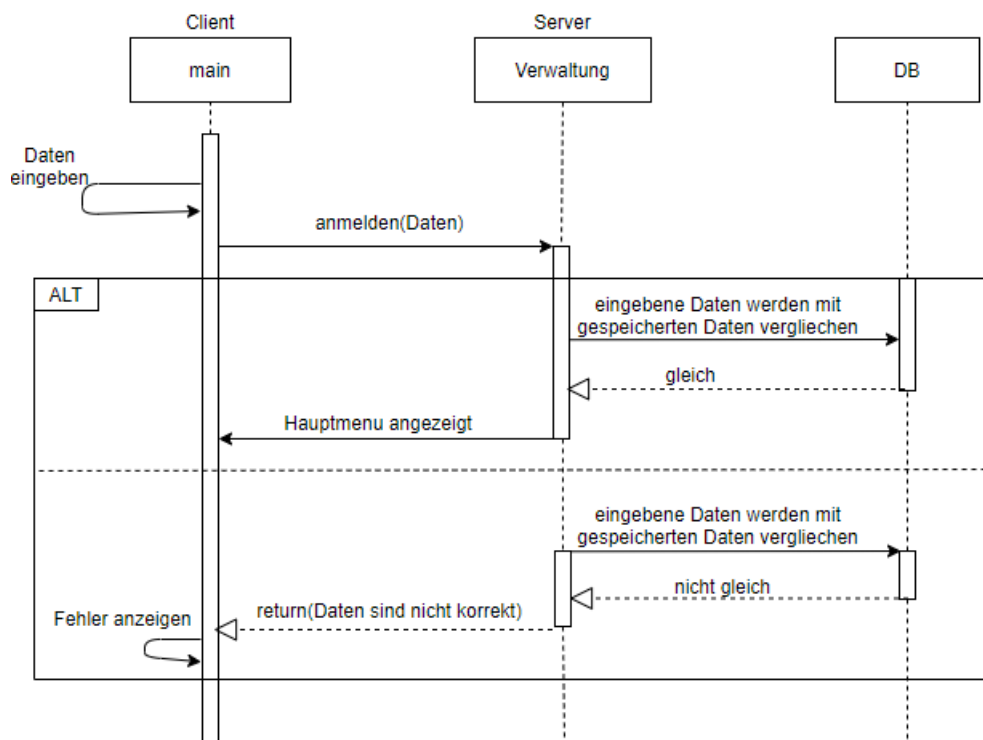


Abbildung 0.8: Anmeldeprozess, man kommt nur mit den richtigen Daten ins Hauptmenu, sonst gibt es einen *PasswordFalsch* Fehler

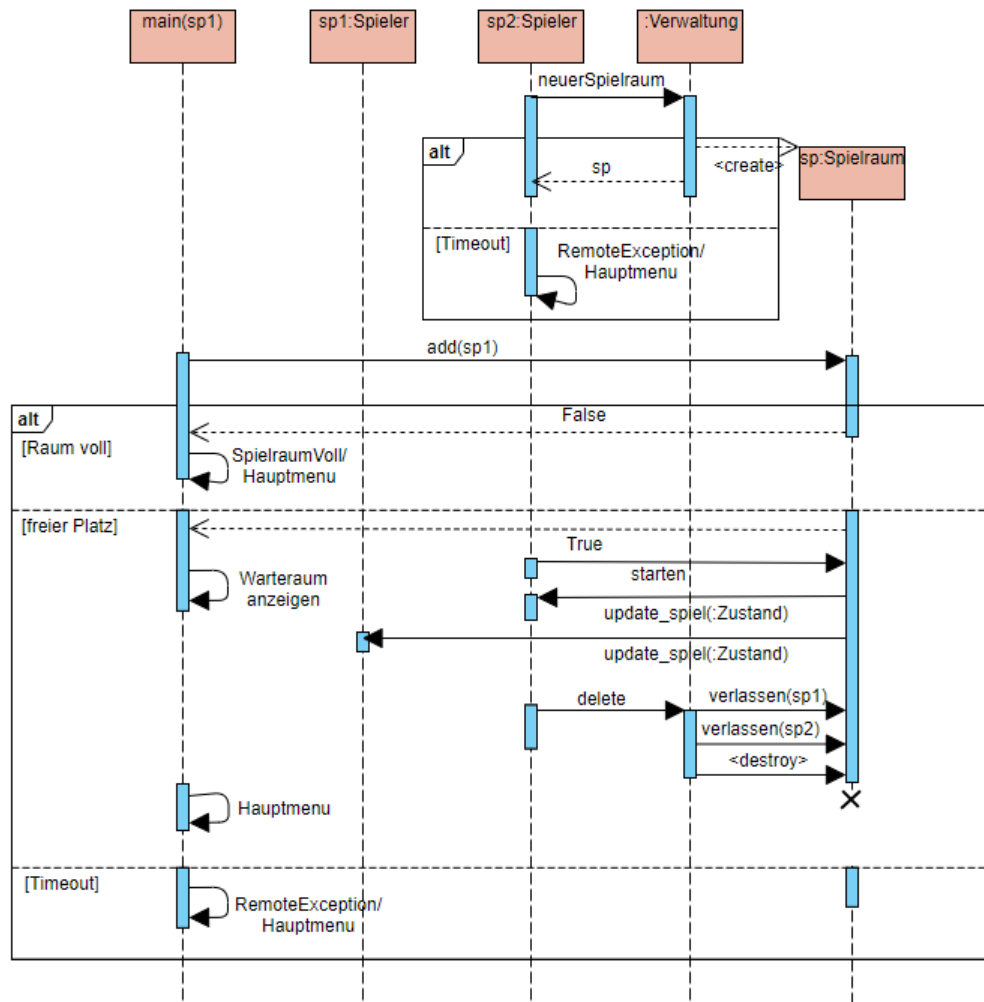


Abbildung 0.9: Erstellen, beitreten und Löschen eines Spielraums, sowie starten und beenden des Spiels.

Grundsätzlich gibt es bei jeder Kommunikation zwischen Server und Client die Möglichkeit einer *RemoteException*, der Übersicht halber aber nur für das Erstellen und Beitreten eingezeichnet (auch bei folgenden Diagrammen wurde das weggelassen).

Streng genommen kommen die Aufrufe von z.B. *starten* aus der *main/GUI*, das darzustellen wäre aber unnötig unübersichtlich.

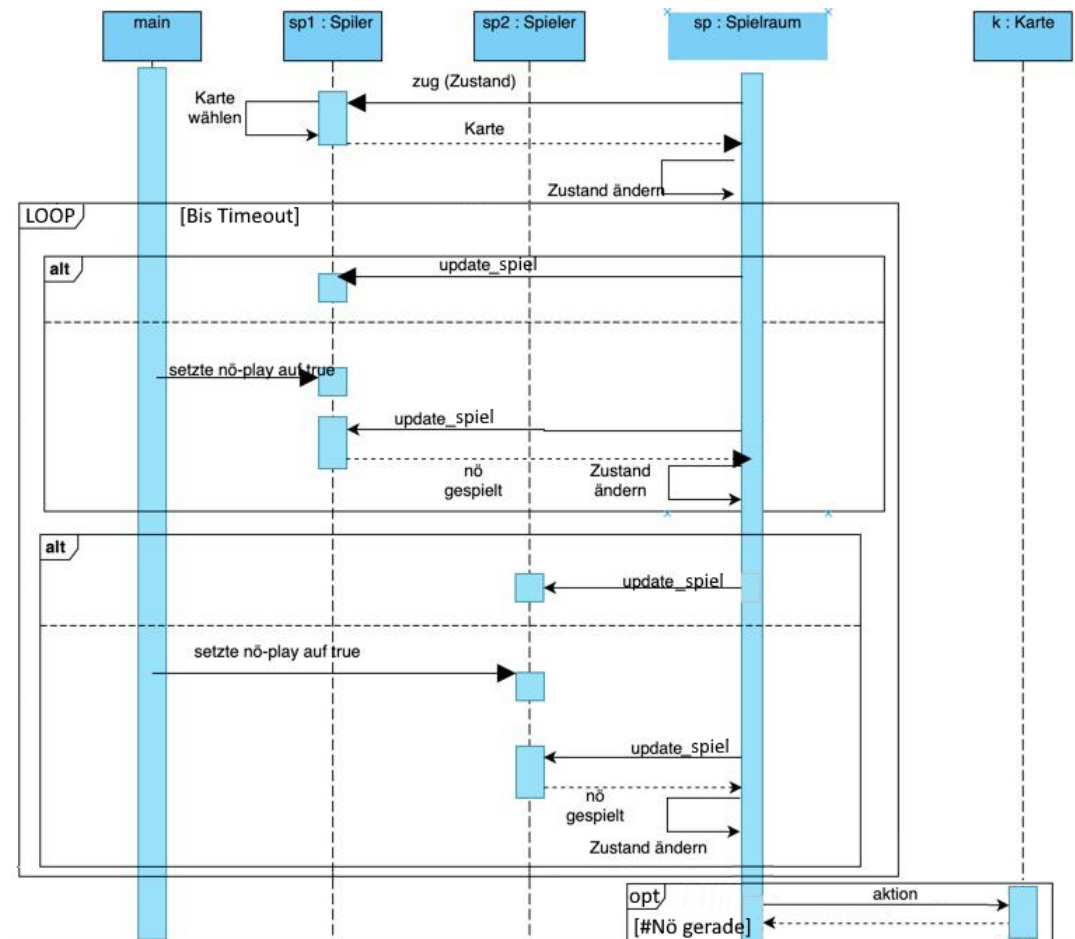


Abbildung 0.10: Grundbaustein des Spiels - Das Legen einer Karte.

Dies geht nur als Antwort auf *zug*, dann wird der Zug bei anderen Spieler per *update_spiel* angezeigt. Nun wird gewartet, ob ein anderer Spieler optional ein Nö spielen will. Die jeweilige GUI setzt dazu ein Attribut in *sp1/sp2*.

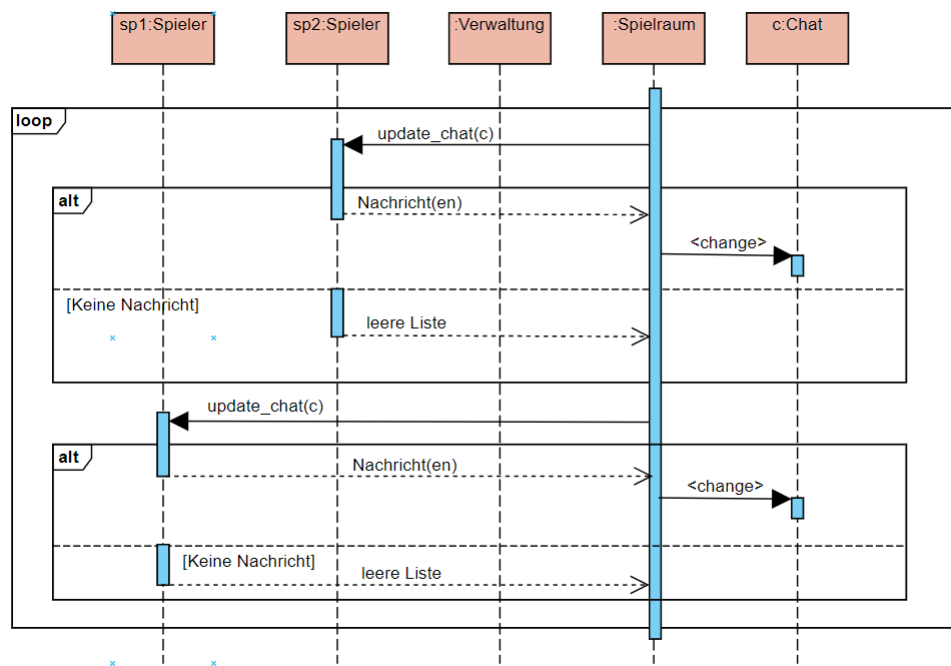


Abbildung 0.11: Chatfunktionen

Spieler wird nicht aktiv sondern gibt nach dem Aufruf von *update_chat* alle im Puffer liegenden Nachrichten zurück.

Exceptions

Die in 0.3 erwähnten Fehler/*Exceptions* sind hier nochmal in einem Klassendiagramm verdeutlicht:

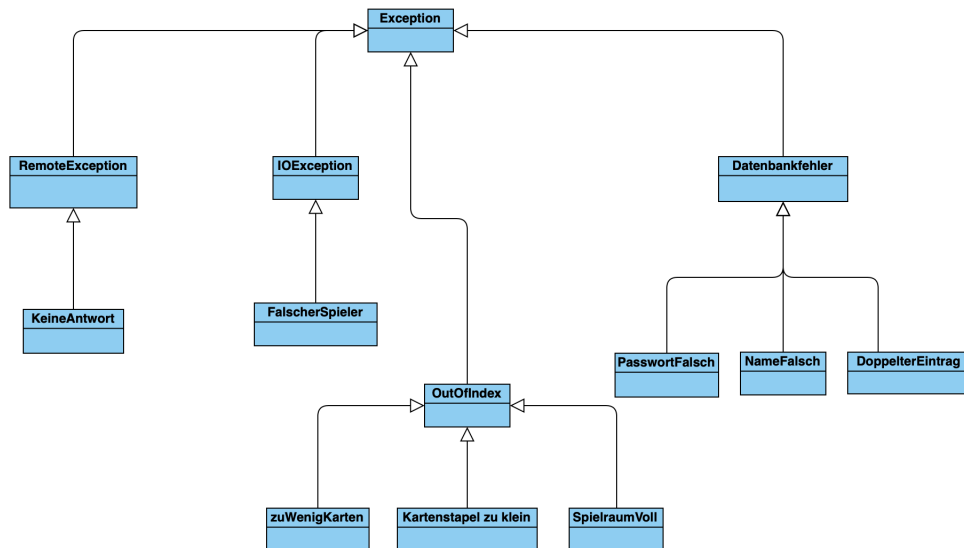


Abbildung 0.12: Wir unterteilen unter anderem zwischen Verbindungsfehlern (*RemoteException*), Fehlern in der Spiellogik (*OutOfIndex*) und *Datenbankfehlern*

Zeit *einfach*: alle 2 Runden

komplex: In Abhängigkeit von $\frac{\#X}{\#KartenimStapel}$, also der Chance auf eine Exploding Kitten.

Legen *einfach*: Immer an die erste Stelle

komplex: Mit bestimmten Wahrscheinlichkeiten an Stelle 1,2,3 oder zufällig.

Kombi *einfach*: nur 2er Kombinationen

komplex: Sparen auf 3er/5er Kombinationen, in Abhängigkeit von eigenen $\#Handkarten$ und der Chance auf eine Exploding Kitten. Die Wahl des Opfers hängt ebenfalls von den jeweiligen $\#Handkarten$ ab.

Die komplexe Variante entscheidet in Abhängigkeit von $\#Handkarten$ und Position auf des Spielers bei welchem Spieler ein *Nö* gespielt wird.

Gantt Diagramm

Für die Planung des weiteren Vorgehens wurden alle Aufgaben der Implementierung und Optimierung benannt und in einem Gantt Diagramm visualisiert.

	Task Name	Duration	Start	Finish	Predec	Resource Names
1	Implementierungsphase	21 days	Mon 25.05.20	Mon 22.06.20		
2						
3	Umsetzung der methoden Signaturen und Beziehungen der Klassen	0,5 days	Tue 26.05.20	Tue 26.05.20		Aurelle;Merveille
4	Datenbanksystem erstellen	0,5 days	Tue 26.05.20	Tue 26.05.20		Justus;Aurelle
5	JavaDoc für öffentliche Methoden	0,5 days	Tue 26.05.20	Tue 26.05.20	3	Emilia;Ivana
6	JavaDoc vervollständigen	3 days	Wed 27.05.20	Fri 29.05.20	5	Ivana;Emilia
7	Unit-Test Implementieren	3 days	Wed 27.05.20	Fri 29.05.20	5	Merveille;Aurelle;Justus
8						
9	Graphische User Interface	5 days	Wed 03.06.20	Tue 09.06.20	7	Aurelle;Emilia;Ivana
10	GUI Funktionen Implementieren	6 days	Wed 10.06.20	Wed 17.06.20	9	Aurelle;Emilia;Ivana
11	Chat Funktion implementieren	3 days	Wed 03.06.20	Fri 05.06.20		Justus;Merveille
12	Login Funktionen Implementieren	4 days	Mon 08.06.20	Thu 11.06.20	11	Justus;Merveille
13						
14	Optimierungsphase	16 days	Tue 23.06.20	Tue 14.07.20		
15						
16	Besprechung der Spiellogik	0,5 days	Tue 23.06.20	Tue 23.06.20		Aurelle;Emilia;Ivana;Justus;Merveille
17	Entwicklung der verbleibenden Klassen	4 days	Wed 24.06.20	Mon 29.06.20	16	Aurelle;Emilia;Ivana;Justus;Merveille
18	Synchronisation des Spielvorgangs unter mehrere Clients	4 days	Wed 01.07.20	Sun 05.07.20	17	Justus;Aurelle
19	Einfacher Bot	6 days	Mon 06.07.20	Mon 13.07.20	18	Emilia;Merveille;Aurelle;Justus;Ivana
20	Komplexerer Bot	7 days	Thu 09.07.20	Fri 17.07.20		Justus;Ivana;Merveille

Abbildung 0.14: Alle zu erledigende Aufgaben

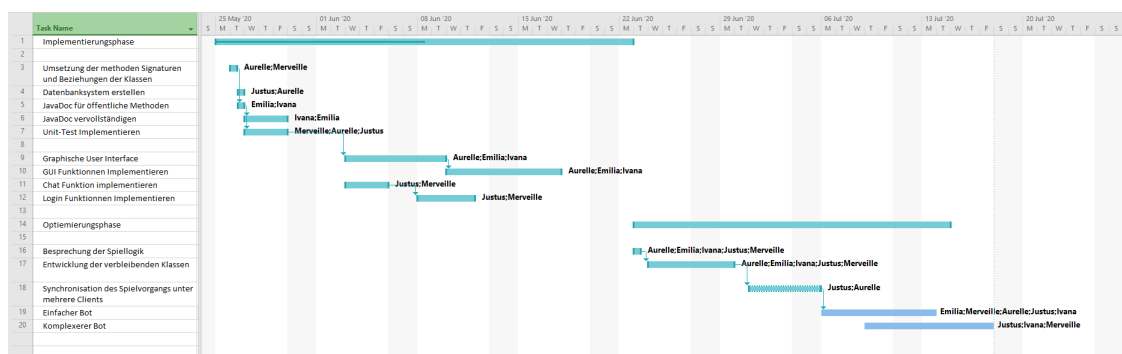


Abbildung 0.15: Visualisierung der Aufgaben in 0.14. Besser lesbar in 0.6

Gantt Diagramm

