IT Project – Project Jassmend

Module:	IT Project (BIT, SS 2020)
Authors:	Florian Jäger Davide Seabra Gazmend Shefiu
Professors:	Bradley Richards Frey Lukas
Place, Date:	Basel, 24th. May 2020

Table of Contents

1. Introduction

For this year's IT project, our group has been assigned to develop an online playable Jass game. After several meetings we decided in to approach this project by making sure that the minimal requirements set by the project responsibles, will be implemented and tested.

In a further development phase, we will start with the implementation of optional requirements such as:

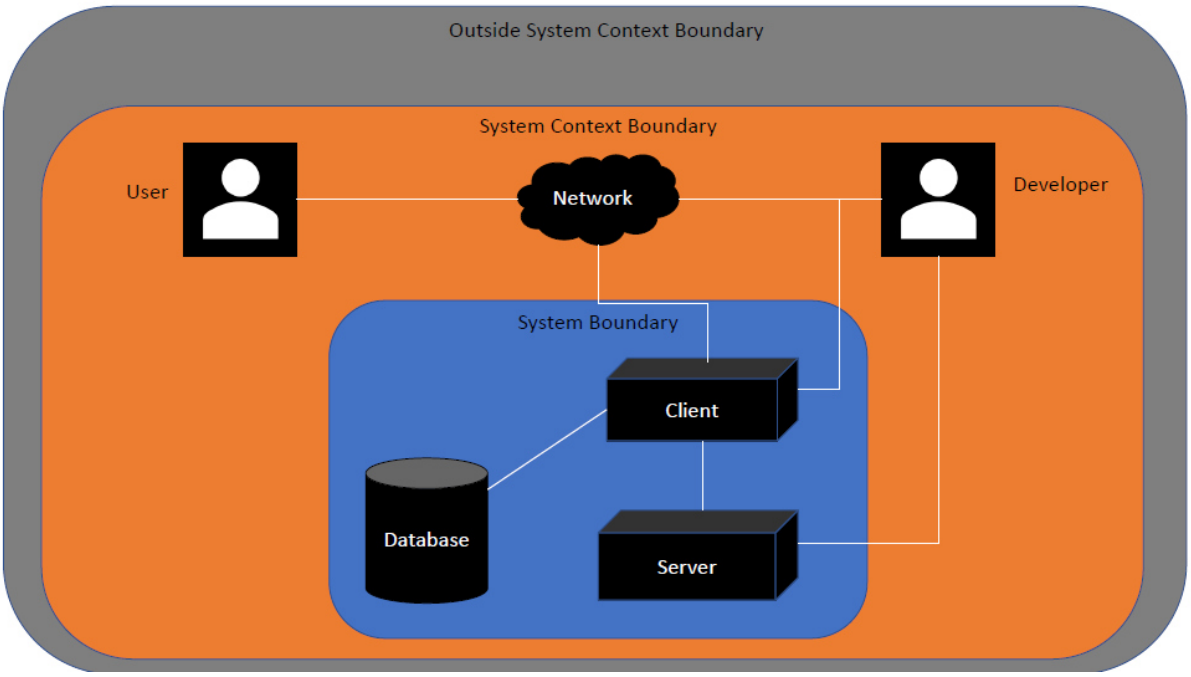
- Playing music in the background
- The possibility of an in-game chat function
- Implementing a help menu which a player can use to check the game rules

More optional requirements are possible to be implemented as the development of this project goes along.

In this document we will name and describe the requirements which will be initially implemented as well as the further optional requirements. At the end, we have our initial test plan.

2. System Boundary

This System Boundary shows the main components of our application. The system boundary consists of the server, the client and the database which is accessed by the client. In the system context boundary, we have the users and the developers who can access the system by entering the network.



3. Client & Server Communication

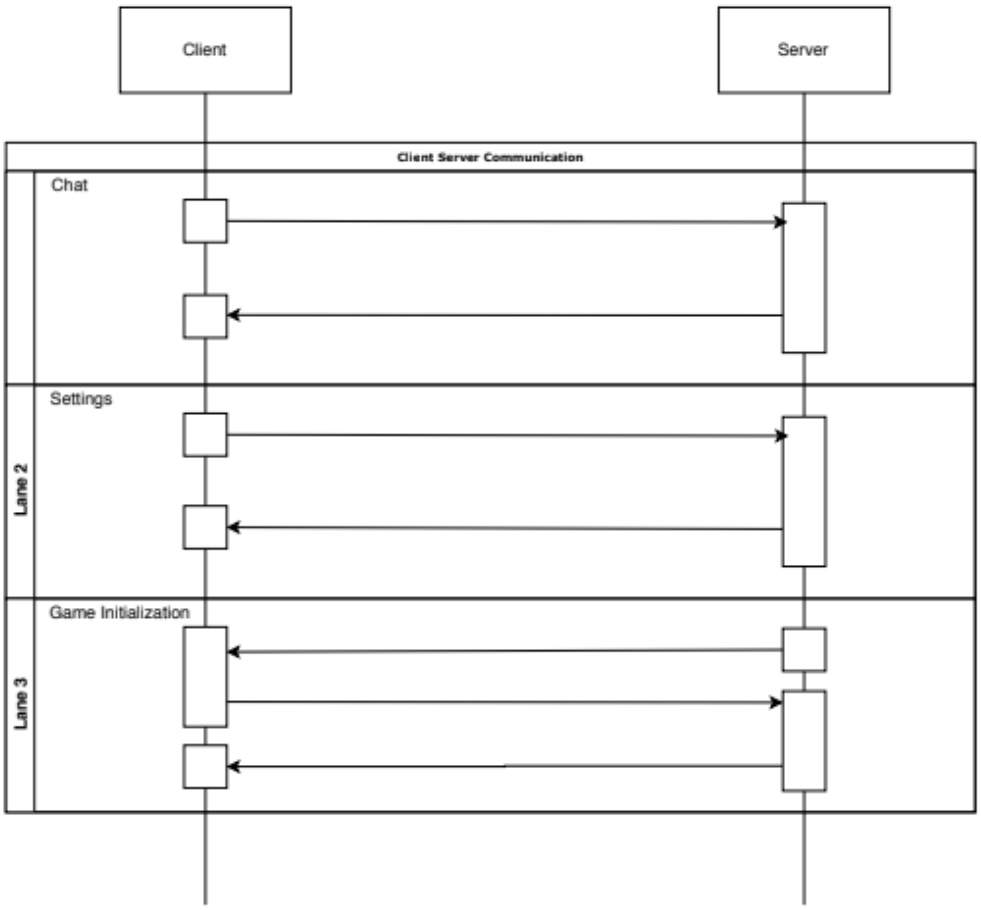
The following sequence diagram shows three different kinds of messages. Even though we are going to use many different message types, they will work in one of the three illustrated ways.

Three Message Types

Chat Messages are all the messages sent in the chat and are initiated by the user during the gamephase.

Game Initiating Messages are initiated by the server and restrict the possibilities of players and prevent illegal moves, the player is only allowed to response in a certain way.

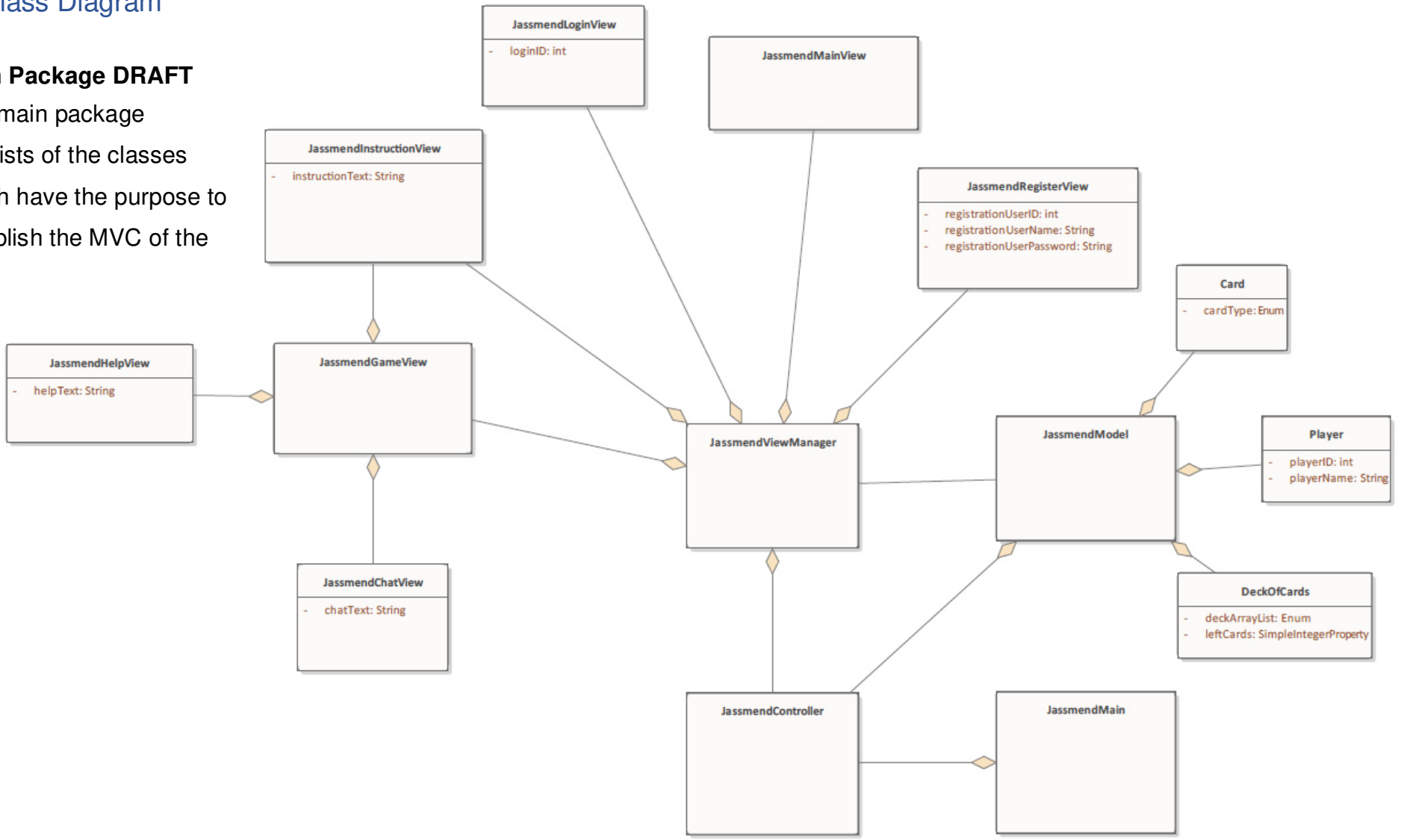
SettingsMessages are messages initiated by the user, where the user sets e.g. the gamemode, creates account, changes password or hosts a game.



4. Class Diagram

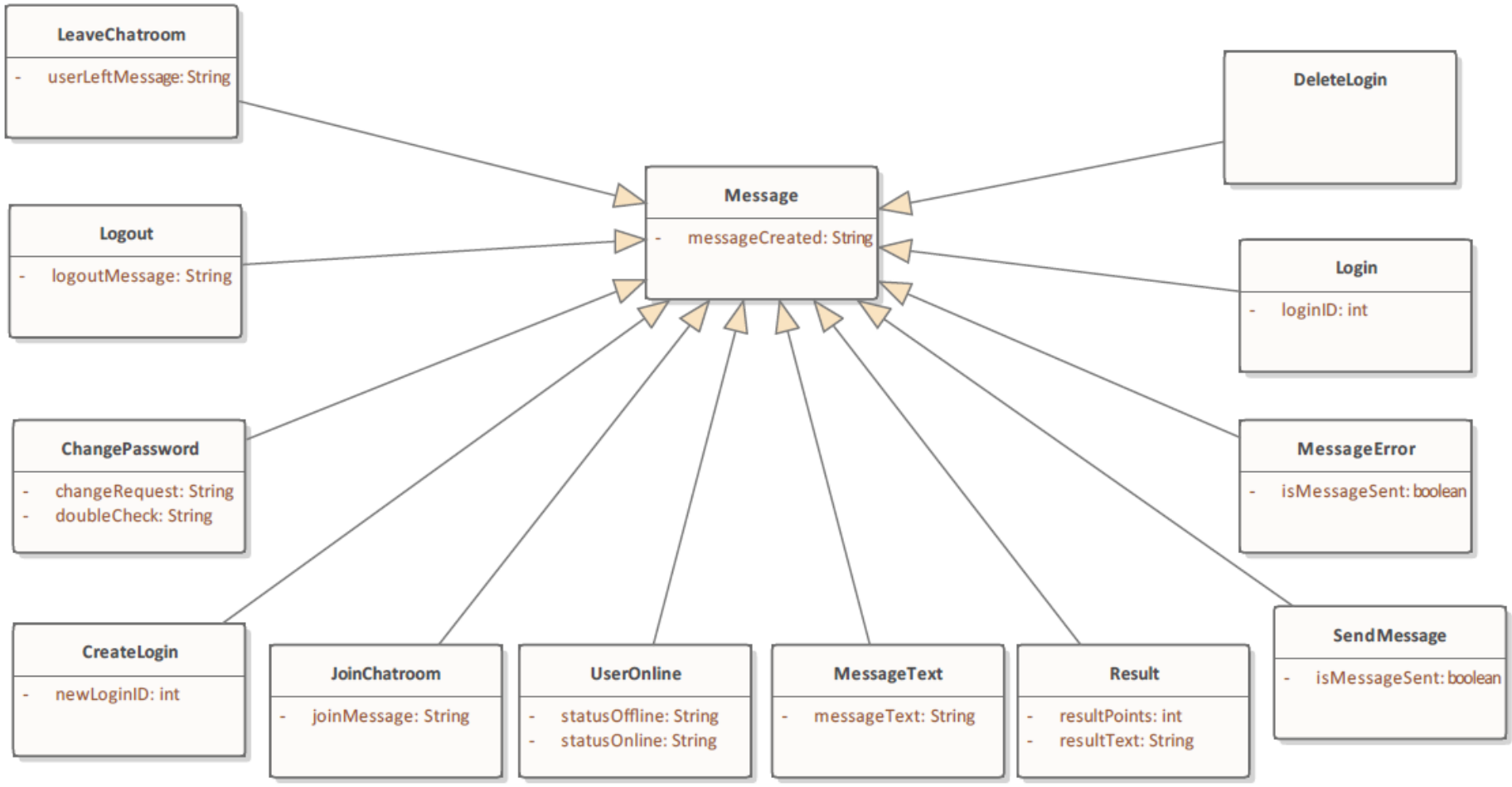
Main Package DRAFT

The main package consists of the classes which have the purpose to establish the MVC of the



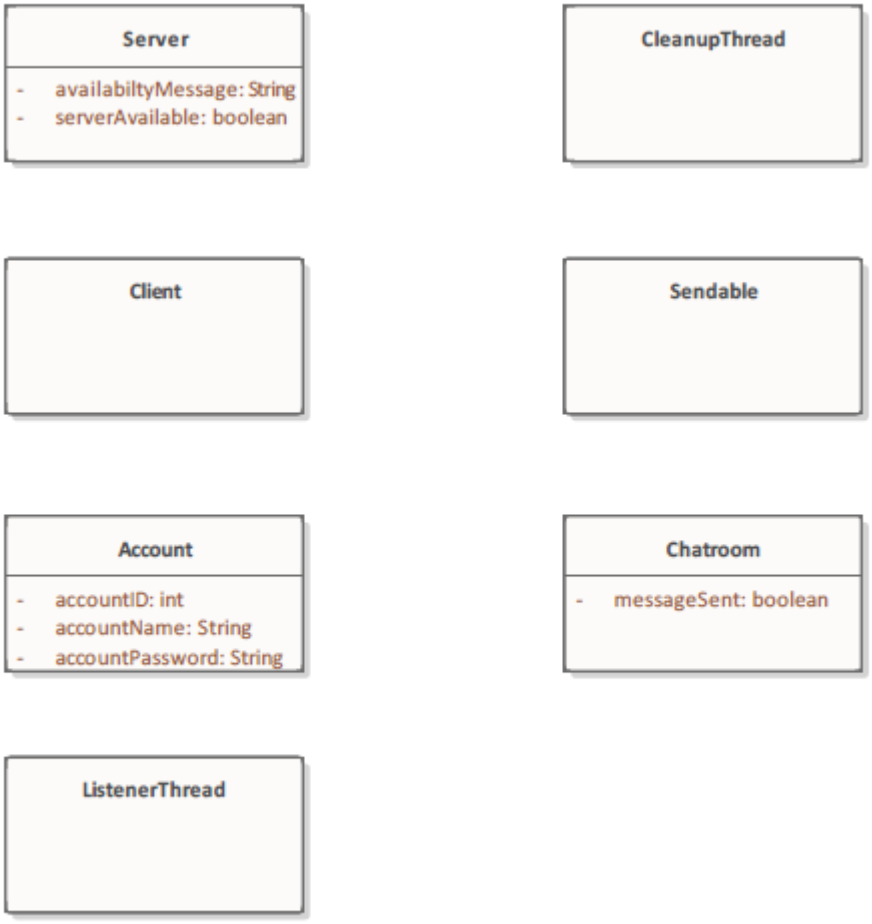
Message Package DRAFT

The message package consists of the classes which only have the purpose of sending and receiving texts etc.



Server Package DRAFT

The server package consists of the classes which have the purpose to establish a connection to the server and show if the server is available or not etc.

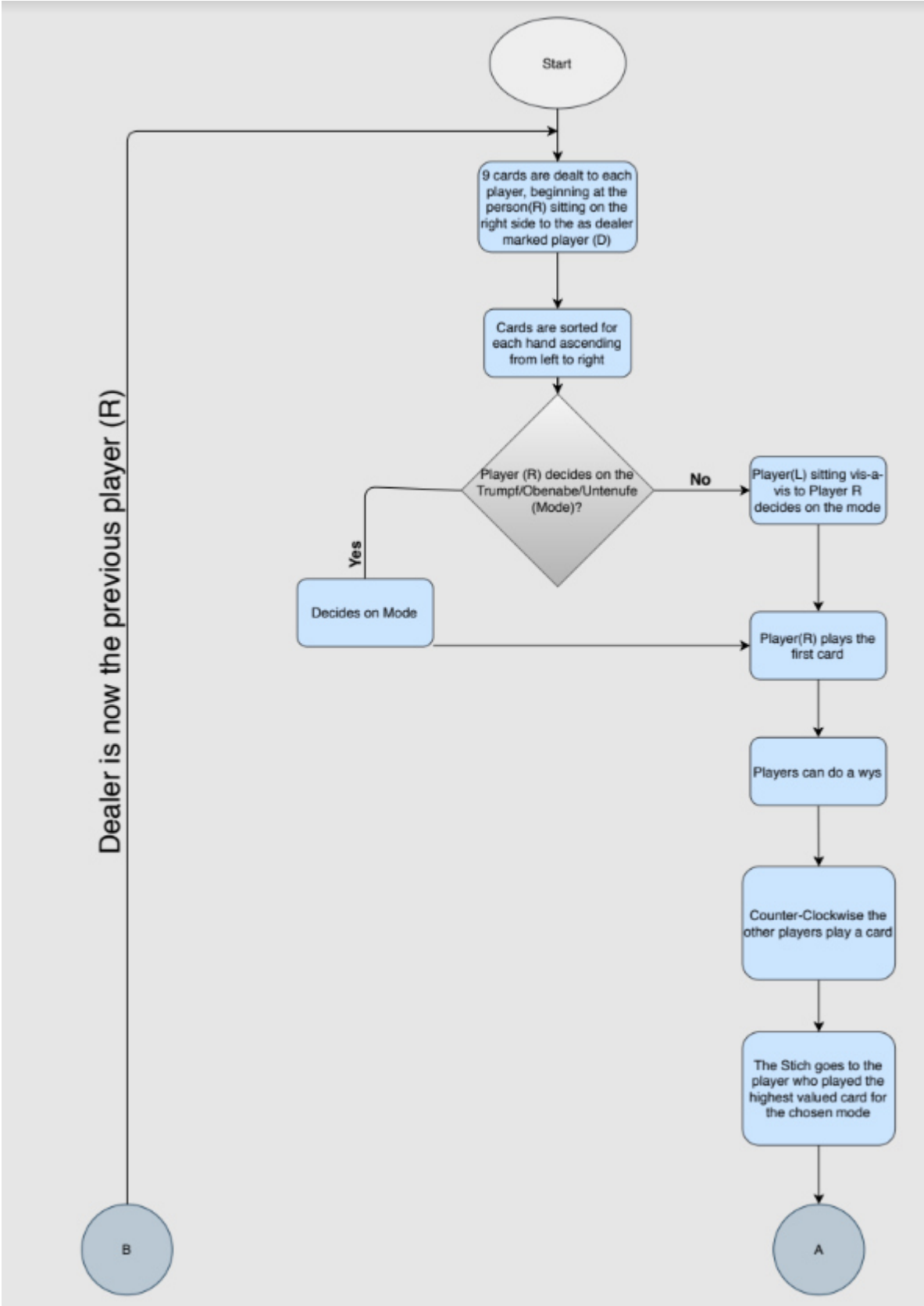


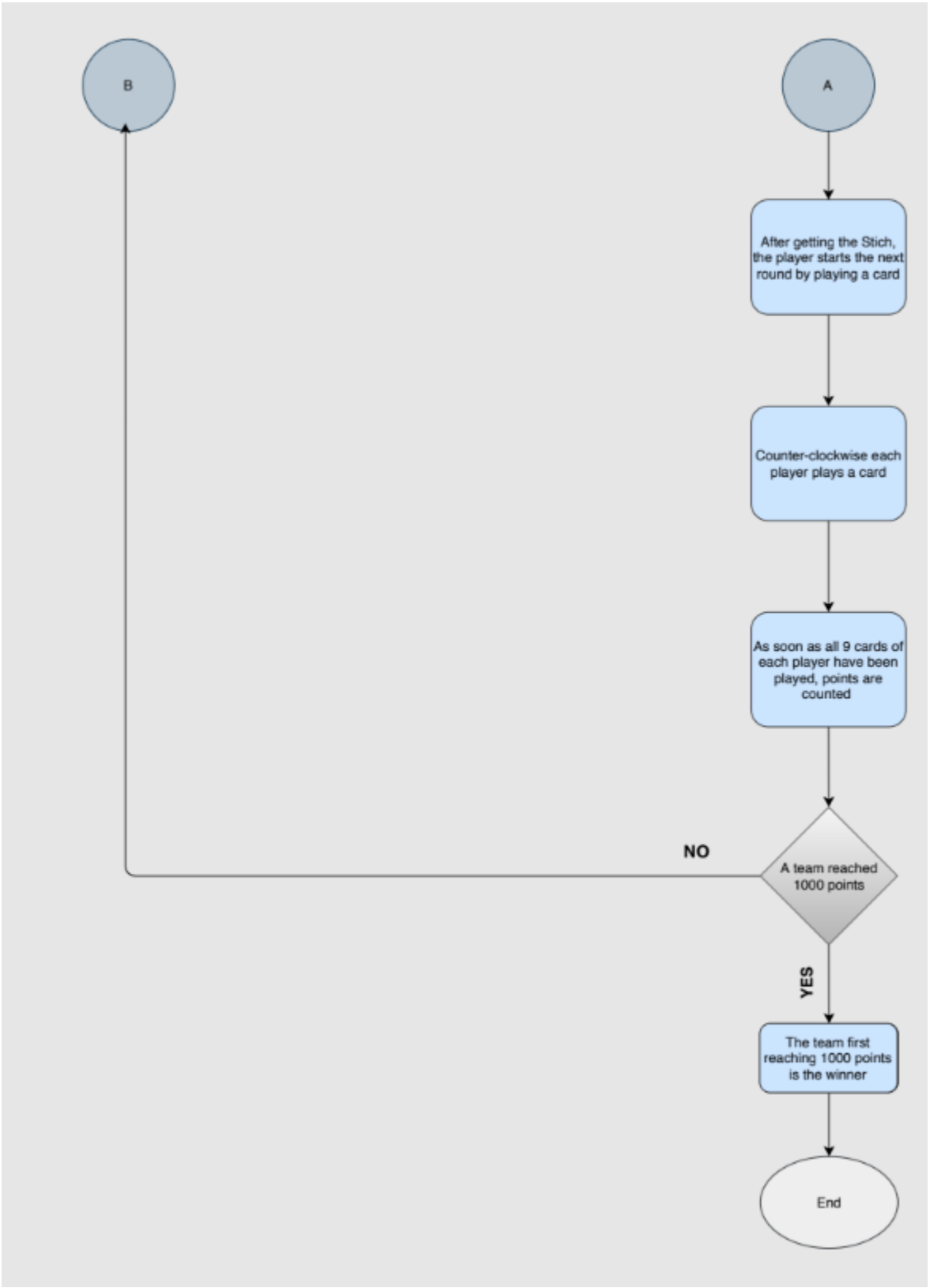
With the draft of the class diagrams, we could start with the organization of the classes. After we implemented the classes, we could start to implement the GUI and work with the code implementation. In the course of time we changed classes, created new classes and of course deleted classes which were unnecessary. Through the implementation of new classes and modifying classes, our class diagram changed with the design and more importantly with the size. We tried to add the class diagram in this documentation, but the diagrams were to large and not readable. So, we decided to save the class diagrams in a separate file, which is more readable than to put it in between the documentation.

We decided to not delete the draft class diagrams in our documentation, so it could be compared with the actual class diagram.

5. Flowchart Jass

This flowchart shows all the steps to play a game of Jass.




B
A
After getting the Stich, the player starts the next round by playing a card
Counter-clockwise each player plays a card
As soon as all 9 cards of each player have been played, points are counted
A team reached 1000 points
NO
YES
The team first reaching 1000 points is the winner
End

6. Gaming Instructions

The gaming instructions are easy to understand and described as follows:

1. Create an Account
2. Log in with your credentials
3. Create a game lobby
4. Join with three other players
5. Game starts automatically
6. Select Trumpf
7. Have fun!

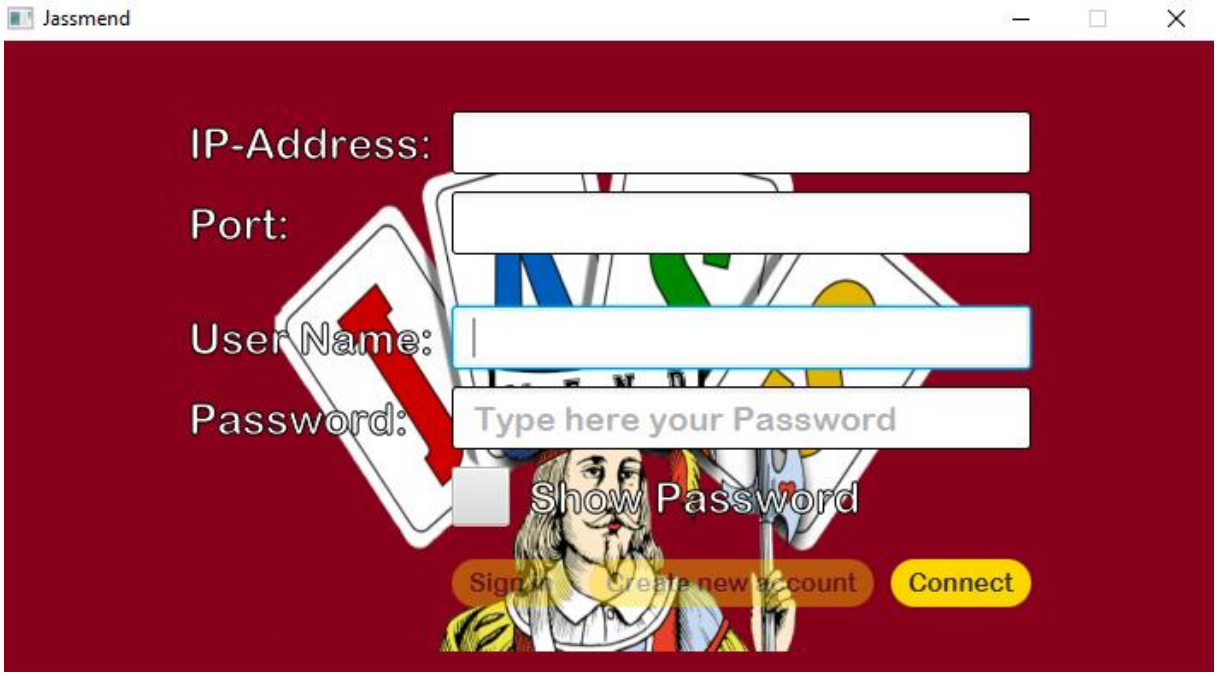
If the database cannot establish a connection, you have to add the jar file in the dblib in the library. This has only to be done once, if the database connection doesn't work.

To access the database, you will find an attachment which serves as an instruction on how to access the database.

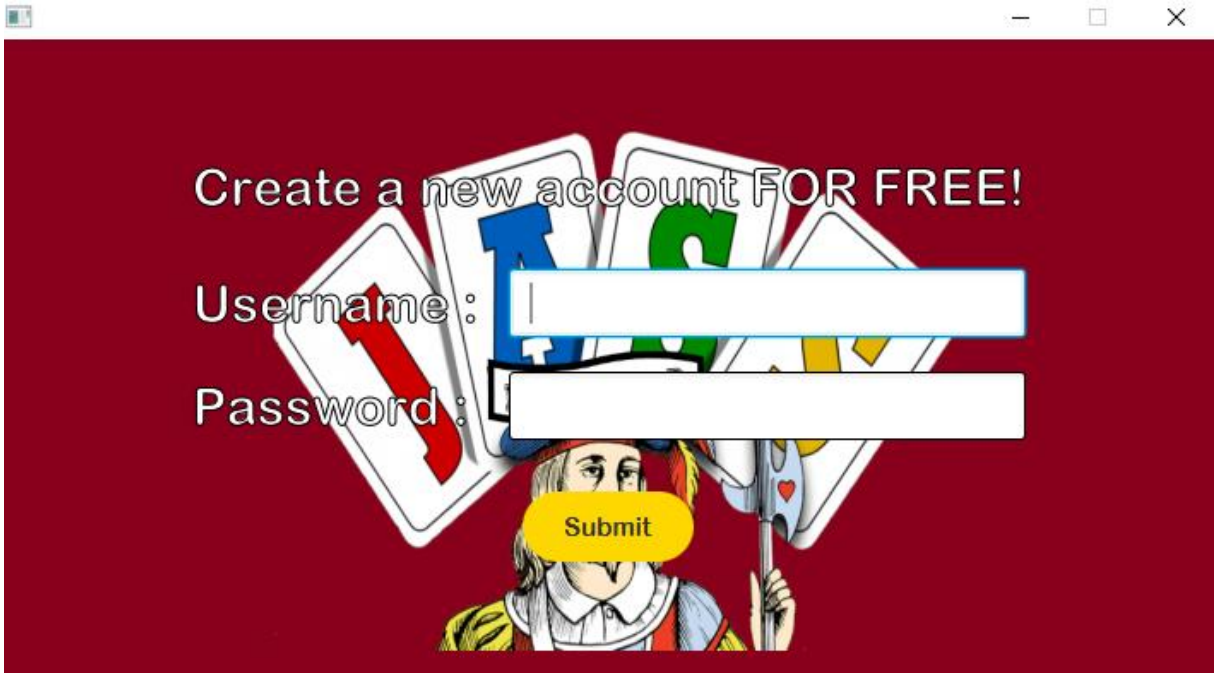
7. Mock-up

The following mock-ups are intended to show some design elements of our application.

The first window for Login



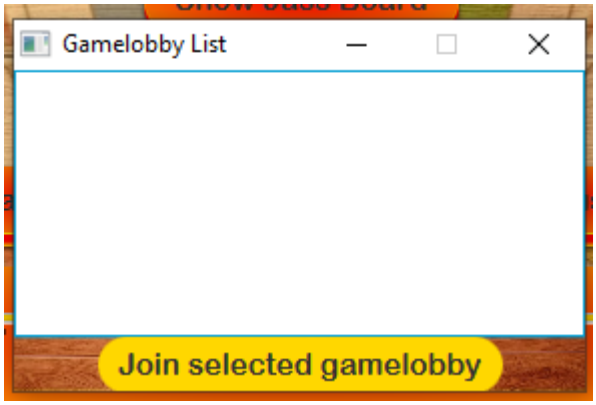
The Sign-up window



The Main Menu

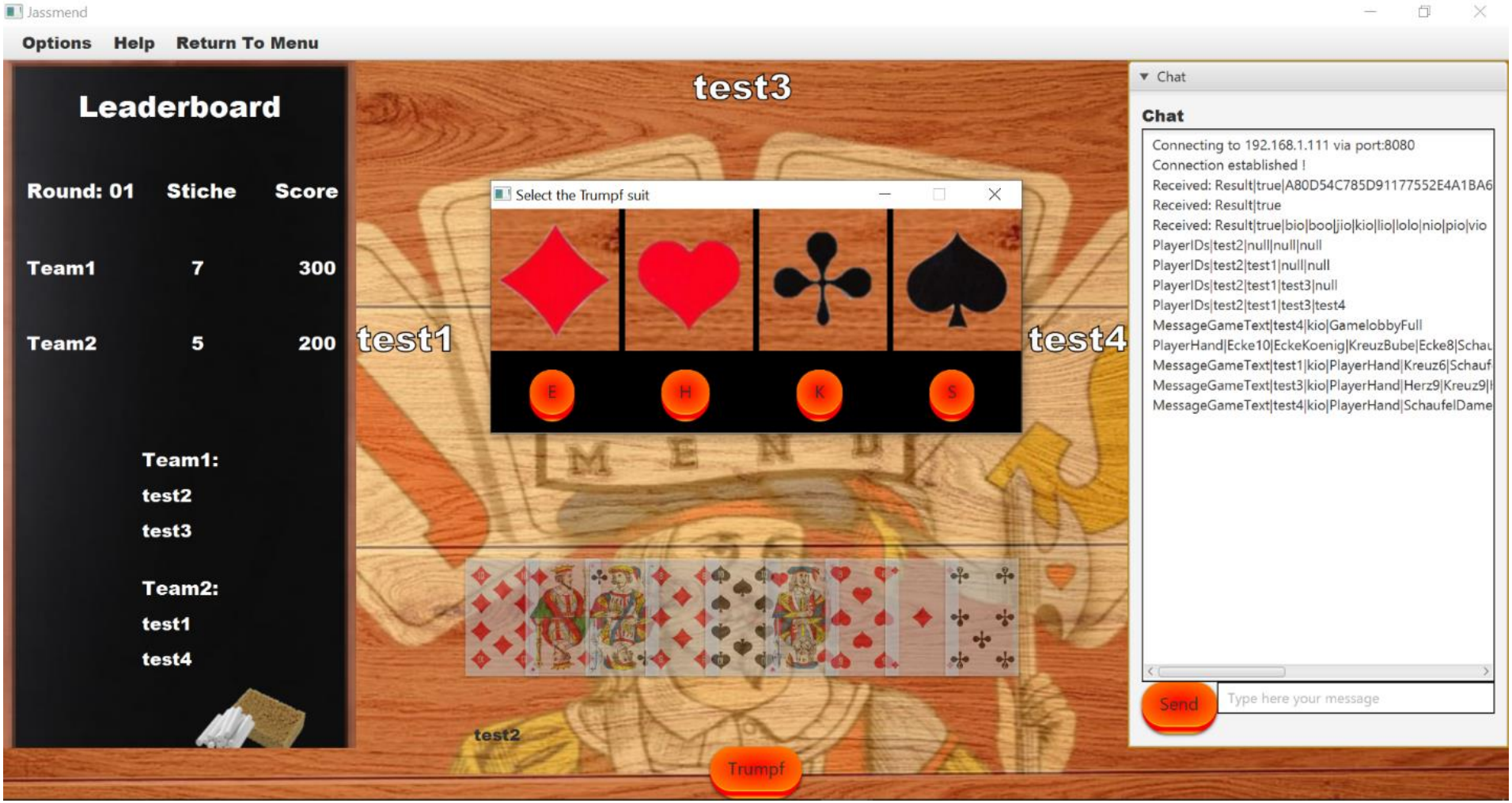
The Create Lobby Window

The Join Lobby Window



A Mock-Up of the gameplay window with the Trumpf setter

This mock-up serves as the gameplay window with the select Trumpf window. When all four players have joined the gamelobby, the message appears that a trumpf suit has to be chosen which can be accesses via the trumpf selection button.



A Mock-up of the gameplay window

This mock-up serves as the definitive gameplay window of our jass project. The chosen Trumpf suit is displayed in the middle of the board. The chat window on the right side is needed to communicate with the other players. The leaderboard on the left serves as an overview about the current game information. Every selected card is displayed in the middle and visible for all players.

8. Programming Distribution

This programming distribution table is provisional and will be extended as well as adapted in the course of the module.

Requirements	Responsible Person	Additional Information	Deadline	Done? (Yes / No)
1. GitHub Establishment	Gazmend Shefiu	-	16.03.2020	YES
2. Establish MVC	Davide Seabra	-	23.03.2020	YES
3. Client, Server & Messaging	Florian Jäger	-	10.04.2020	YES
4. Advanced GUI	Gazmend Shefiu	-	14.04.2020	YES
5. Create Model	Florian Jäger	-	16.04.2020	YES
6. Create View	Davide Seabra	-	19.04.2020	YES
7. Create Controller	Florian Jäger	-	19.04.2020	YES
8. Music Implementation	Gazmend Shefiu	-	25.04.2020	YES
9. Create Game Logic	Florian Jäger	-	26.04.2020	YES
10. Create Help Window	Davide Seabra	-	14.05.2020	YES
11. Create Chat Window	Florian Jäger	-	16.05.2020	YES
12. Final testing	Davide Seabra / Florian Jäger / Gazmend Shefiu	-	23.05.2020	YES

9. List of Requirements

On the table below we have written a few keywords which we are going to use for the list of requirements. Furthermore, we have chosen to use the kano categories to categorize the requirements as we did in requirements engineering.

Abbreviation	Kano Category Name	Abbreviation	General
DS	Dissatisfier	UC ID	Use Case ID
S	Satisfier	KC	Kano Category
D	Delighter	NP	Networking Package
		GLP	Game Logic Package
		CP	Chat Package

UC ID	KC	Requirements	Description	Priority
NP_1	DS	Client-/Server implementation of the game	The game can be played online.	HIGH
NP_2	S	Create account	The user can create an account.	HIGH

UC ID	KC	Requirements	Description	Priority
NP_3	S	Login	The user can login to the server.	HIGH
NP_4	S	Logout	The user can logout from the server.	HIGH
GLP_1	DS	Fixed number of players supported	The default Jass game mode will allow four players to play the game.	HIGH
GLP_2	DS	Computer chooses play mode, start player, etc.	Menu screen will show different options to choose from. There are going to be multiple buttons with the corresponding options to let the player choose from.	HIGH
GLP_3	DS	Play mode “Trumpf” has to be supported	One suit will be chosen to be the “Trumpf”.	HIGH
GLP_4	DS	Make moves (only legal moves)	The moves of the users have to be according to the rules.	HIGH
GLP_5	DS	Game can end after one round	The players choose how many rounds they are going to play. If they choose to play one round, the game will end after one round.	HIGH

UC ID	KC	Requirements	Description	Priority
GLP_6	DS	Calculate the winner/Message about the winner	The winner of the round will be shown at the end of the round. Additionally, the high score of one player will be shown if there are multiple rounds.	HIGH
GLP_7	DS	Basic design	General viewable design of the start screen, play screen, options screen and chat window	HIGH
GLP_8	S	Advanced design	Basic design enhanced with CSS elements and animations for a better game experience.	HIGH
GLP_9	S	Music Implementation	Implementation of music which can be played, paused and stopped. Music volume should be adjustable with volume slider.	MEDIUM
GLP_10	D	Help window	A Help icon will always be visible. When the user clicks on it, it will show an information page with all the rules of the game and tips.	MEDIUM
CP_1	S	Chat function	Chat window is always visible in the play screen.	HIGH

UC ID	KC	Requirements	Description	Priority
CP_2	S	Send message	The user can send messages.	HIGH
CP_3	S	Receive message	The user receives messages.	HIGH

10. Diagrams & Packages Overview

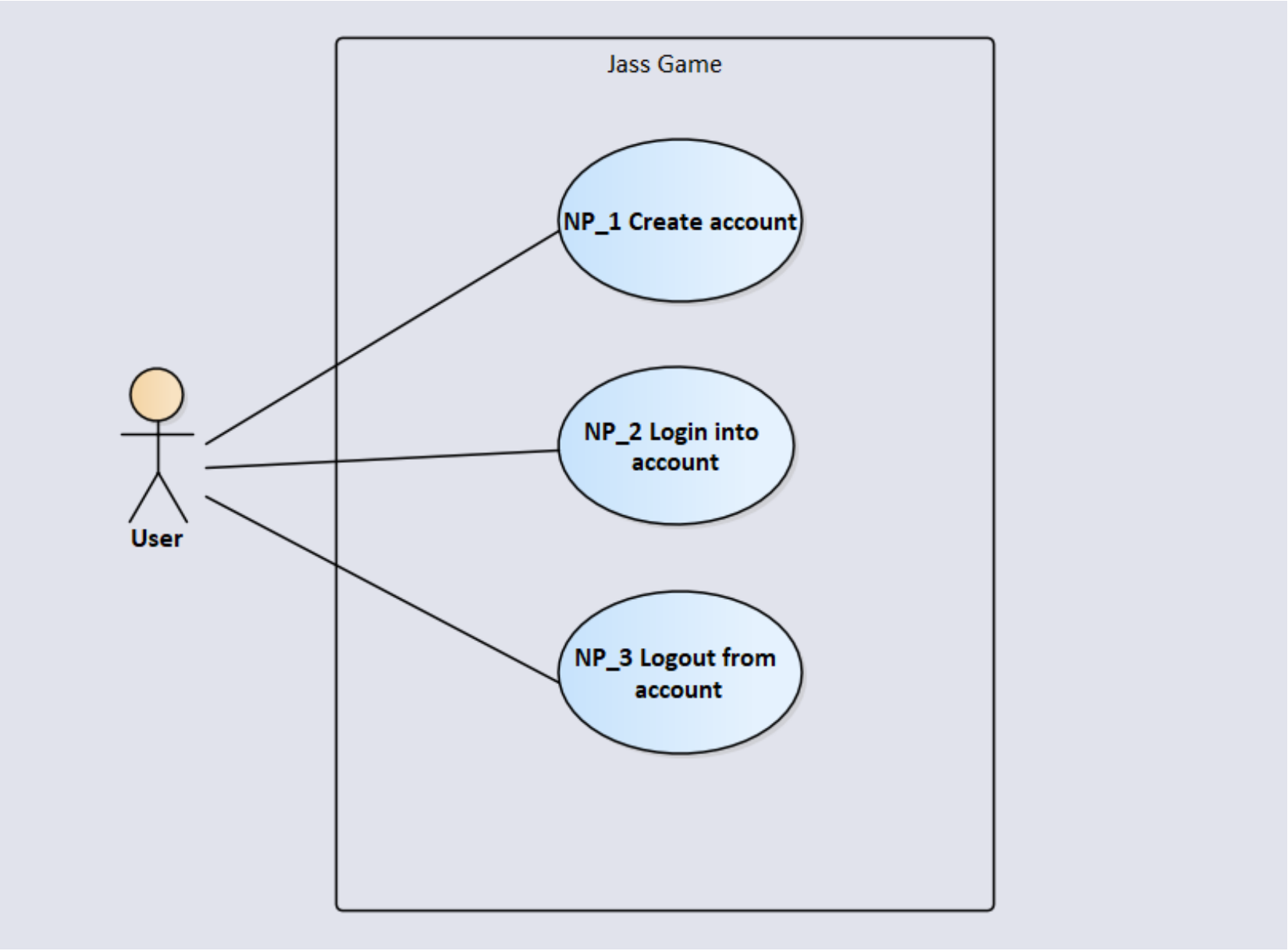
This overview shows the different requirements of our IT-Project. We have chosen 15 requirements which we have divided in three different packages.

ID	Requirement	Diagram Type	Packages
1	Client-/Server implementation of the game	Use Case	Networking Package
2	Create account	Use Case	
3	Login	Use Case	
4	Logout	Use Case	
5	Fixed number of players supported	Use Case	Game Logic Package
6	Computer chooses play mode, start player, etc.	Use Case	
7	Play mode “Trumpf” has to be supported	Use Case	
8	Make moves (only legal moves)	Use Case	
9	Game can end after one round	Use Case	
10	Calculate the winner/Message about the winner	Use Case	
11	Help window	Use Case	
12	Implementation of Music	Use Case	
13	Chat window	Use Case	Chat Package
14	Send message	Use Case	
15	Receive message	Use Case	

Use Case ID	NP_2	
Description	The User registers into the game by defining a username and a password	
Responsible Actor	User	
Participating Actor	Server	
Trigger	User wants to create a new account.	
Preconditions	None	
Input	Input Data: Username, Password.	
Scenario		
	Main	Alternative
	1. User enters username	
	2. User enters password	
	3. System checks the entered data	
	4. Data is accepted - user has succesfuly registered a new account	
	5. System creates new user account.	
Results	A new account has been made.	
Postconditions	User has registered a new account.	

Use Case ID	NP_3	
Description	The User logs in with the Username and Password he used for the registration.	
Responsible Actor	User	
Participating Actor	Server	
Trigger	User wants to login into the game in order to join an existing game or create a new one.	
Preconditions	UC - Create account	
Input	Input Data: Username, Password	
Scenario		
	Main	Alternative
	1. User enters Username & Password	
	2. System checks data	
	4. Data is correct	4a Data is not correct 4a1 User proceeds with step 1 until data is correct.
Results	-	
Postconditions	User is logged in to system	

Use Case ID	NP_4	
Description	The User logs out from the game.	
Responsible Actor	User	
Participating Actor	Server	
Trigger	User wants to logout from the game	
Preconditions	UC - Login into account	
Input	-	
Scenario		
	Main	Alternative
	1. User clicks on the logout button	
	2. User is logged out	
Results	-	
Postconditions	User is logged out from the system	


Jass Game
NP_1 Create account
NP_2 Login into account
NP_3 Logout from account
User

11. Test Plan

Test ID	Test Description	Pre-Condition	Expected Result	Actual Result
1	Create user with name and password	Established connection to the server	New user is created on the server with the password and name inserted by the user.	
2	Login / Logout	Established connection to the server	User is logged out / in from/to the server	
3	Change password	Established connection to the server	User's password is changed	
4	Establish network connection		Connection to server is established	
5	Send messages to server	Established connection to the server	Messages can be sent to the server and are readable for the server	
6	Receive messages from server	Established connection to the server	Messages from the server can be read and processed	
7	Create game and launch on server	User is logged in.	New game is created on server and has an ID	

8	Join open game	User is logged in, game is launched on the server	Join existing game by using the game ID	
9	Card deck with correct cards for the game		Set of jass cards with pictures.	
10	Cards are distributed to each player in the right amount	Game has started	Each player has seven random cards at the start of each round	
11	Recognize value of cards (Color, number, trumpf)		The difference in value of two cards is recognized	
12	Fixed number of players of four		The game starts not with less than four players. There cannot be more than four players in a game.	

13	Fixed play mode		There is one play mode which cannot be changed by the user	
14	Set number of rounds		The host of the game, can decide on the number of rounds to be played	
15	Rules for legal moves		Moves which are not legal and not compliant with the game rules, cannot be made	
16	Recognize end of round with user alert		Recognition of round end and visual user feedback	
17	Count rounds		Rounds are counted and displayed in the GUI	
18	Count points for each user in the game		Points are calculated and displayed in the GUI	
19	Find the winner of each round		Winner is calculated and displayed in the GUI	

12. References

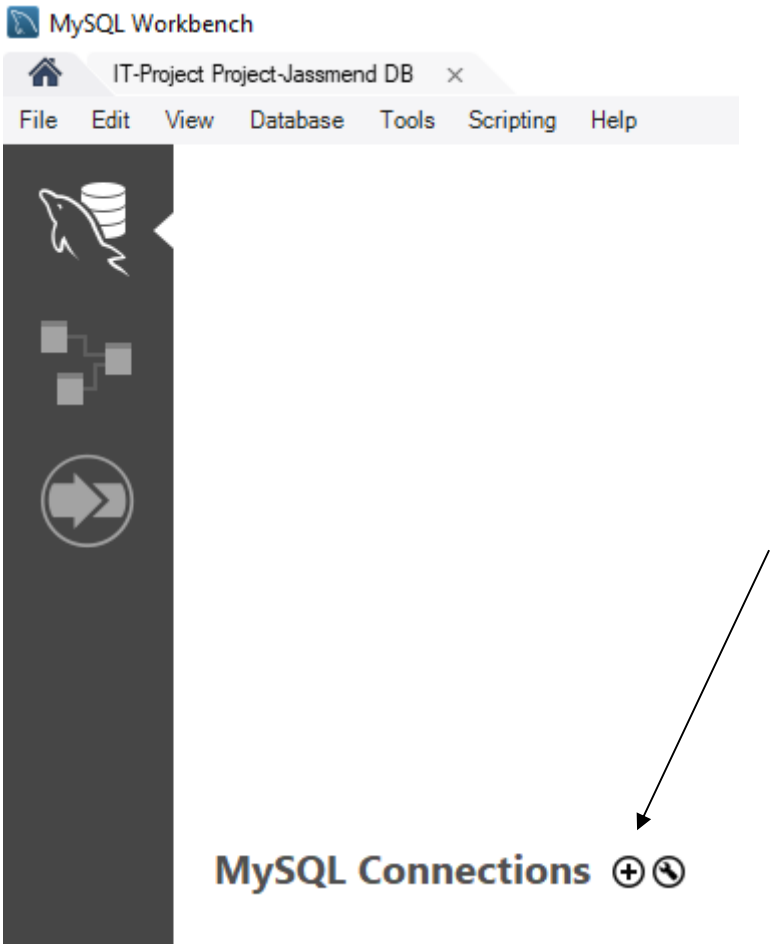
Title Page was edited by our group. The original photo is referenced below.

Title Page picture: https://agenda.bielertagblatt.ch/de/veranstaltung/club-de-jass-romand-bienne

Attached is a pdf file which serves as an instruction on how to access the database.

Attached is a pdf file which shows the actual class diagram.

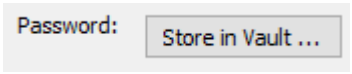
Access to github: https://github.com/Gazi22/IT-Project-Jassmend



Hostname: eu-cdbr-west-03.cleardb.net

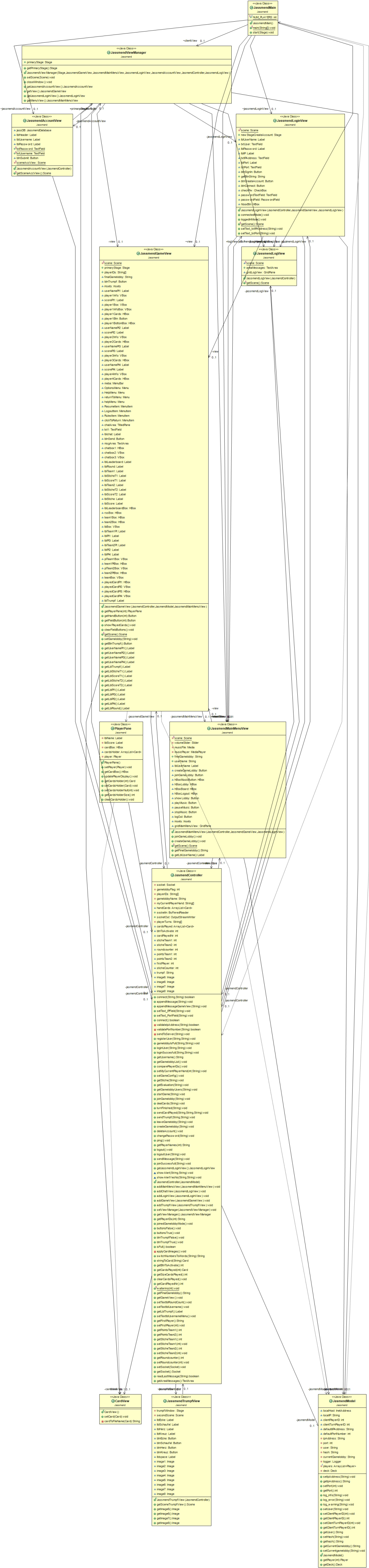
Port: 3306

Username: b3ab4e3875375e



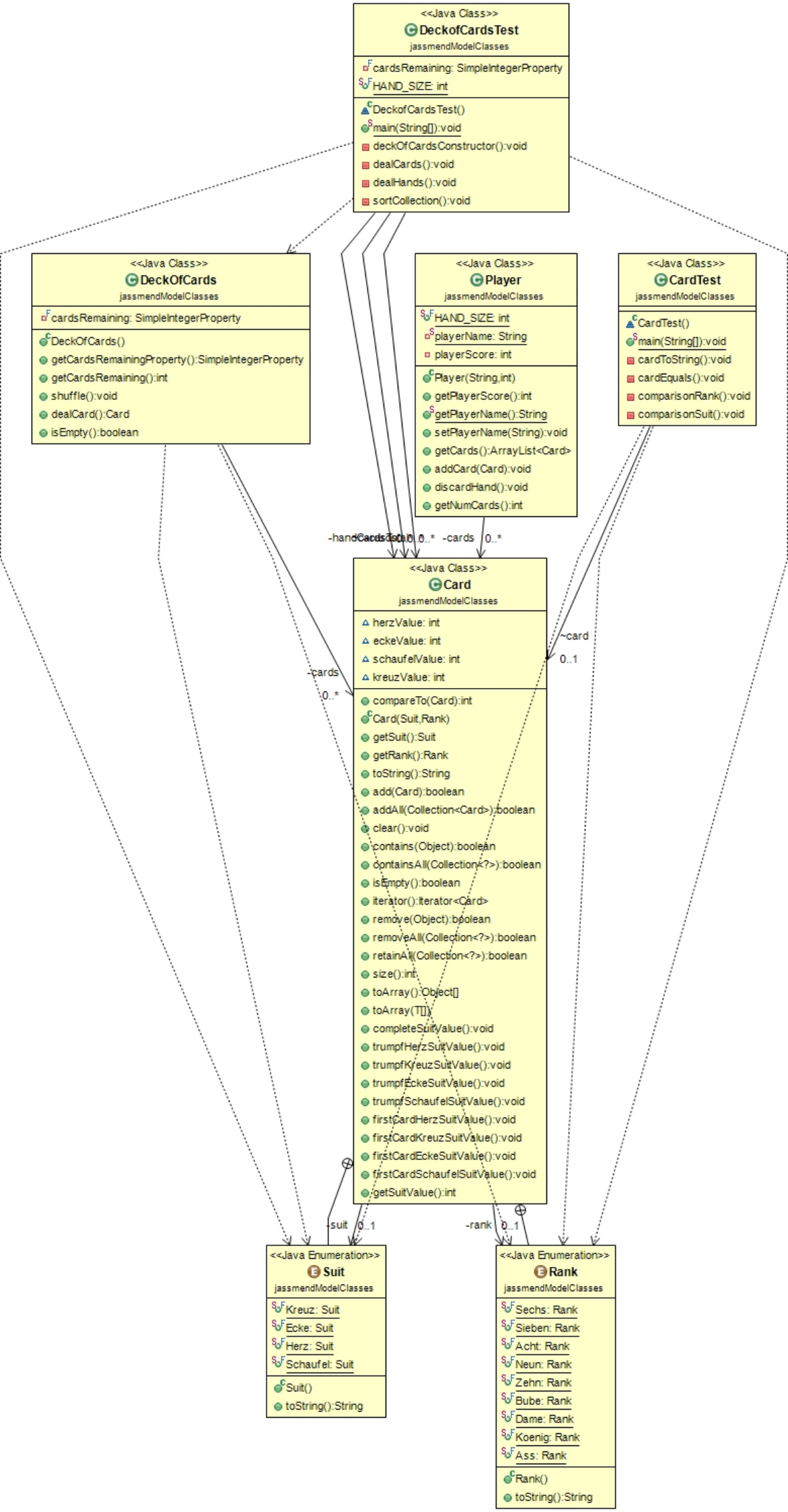
Password: 10ff5735

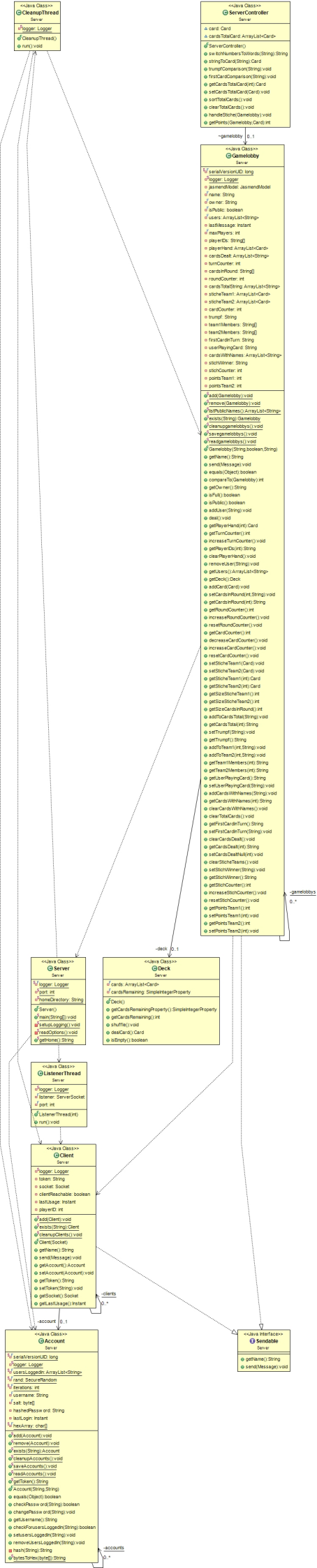
mysql://b3ab4e3875375e:10ff5735@eu-cdbr-west-03.cleardb.net/heroku_ffd397e5d72f806?reconnect=true

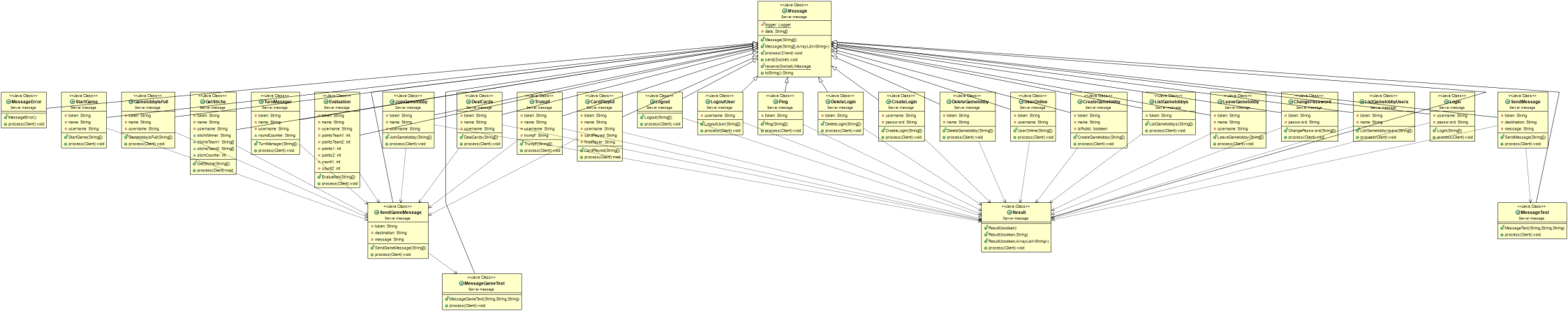


<<Java Class>>
JassmendMain
jassmend
NUM_PLAYERS: int
JassmendMain()
main(String[]):void
start(Stage):void
<<Java Class>>
JassmendViewManager
jassmend
primaryStage: Stage
getPrimaryStage():Stage
JassmendViewManager(Stage,JassmendGameView,JassmendMainMenuView,JassmendLoginView,JassmendAccountView,JassmendController,JassmendLogView)
setScene(Scene):void
closeWindow():void
getJassmendAccountView():JassmendAccountView
getView():JassmendGameView
getJassmendLoginView():JassmendLoginView
getMenuView():JassmendMainMenuView
<<Java Class>>
JassmendAccountView
jassmend
jassDB: JassmendDatabase
lblHeader: Label
lblUsername: Label
lblPassword: Label
txtPassword: TextField
txtUsername: TextField
btnSubmit: Button
sceneAccView: Scene
JassmendAccountView(JassmendController)
getSceneAccView():Scene
<<Java Class>>
JassmendLoginView
jassmend
scene: Scene
newStageCreateAccount: Stage
lblUserName: Label
txtUser: TextField
lblPassword: Label
lblIP: Label
txtIPAddress: TextField
lblPort: Label
txtPort: TextField
btnSignIn: Button
getBtnString: String
btnCreateAccount: Button
btnConnect: Button
checkPw: CheckBox
passwordTextField: TextField
passwordField: PasswordField
hboxBtn: HBox
JassmendLoginView(JassmendController,JassmendGameView,JassmendLogView)
connectedMode():void
loggedInMode():void
getScene():Scene
setText_txtIPAddress(String):void
setText_txtPort(String):void
<<Java Class>>
JassmendLogView
jassmend
scene: Scene
sceneMessages: TextArea
gridLogView: GridPane
JassmendLogView(JassmendController)
getScene():Scene
<<Java Class>>
JassmendGameView
jassmend
scene: Scene
primaryStage: Stage
playerIDs: String[]
finalGamelobby: String
btnTrumpf: Button
insets: Insets
userNamePl1: Label
player1Info: VBox
scorePl1: Label
player1Box: VBox
player1InfoBox: VBox
player1Cards: HBox
player1Btn: Button
player1BottomBox: HBox
userNamePl2: Label
scorePl2: Label
player2Info: VBox
player2Cards: HBox
userNamePl3: Label
scorePl3: Label
player3Info: VBox
player3Cards: HBox
userNamePl4: Label
scorePl4: Label
player4Info: VBox
player4Cards: HBox
menu: MenuBar
OptionsMenu: Menu
HelpMenu: Menu
returnToMenu: Menu
helpMenu: Menu
ResumeItem: MenuItem
LogoutItem: MenuItem
RulesItem: MenuItem
clickToReturn: MenuItem
chatArea: TitledPane
txt1: TextField
lblChat: Label
btnSend: Button
msgArea: TextArea
chatbox1: HBox
chatbox2: VBox
chatbox3: VBox
lblLeaderboard: Label
lblRound: Label
lblTeam1: Label
lblStichsT1: Label
lblScoreT1: Label
lblTeam2: Label
lblStichsT2: Label
lblScoreT2: Label
lblStiche: Label
lblScore: Label
lblLeaderboardBox: HBox
roundBox: HBox
team1Box: HBox
team2Box: HBox
lblBox: VBox
lblTeam1Pl: Label
lblPl1: Label
lblPl3: Label
lblTeam2Pl: Label
lblPl2: Label
lblPl4: Label
plTeam1Box: VBox
team1PlBox: HBox
plTeam2Box: VBox
team2PlBox: HBox
teamBox: VBox
playedCardPl1: HBox
playedCardPl2: VBox
playedCardPl3: HBox
playedCardPl4: VBox
lblTrumpf: Label
JassmendGameView(JassmendController,JassmendModel,JassmendMainMenuView)
getPlayerPane(int):PlayerPane
getHandButton(int):Button
getPlayButton(int):Button
showPlayedCards():void
clearFieldButtons():void
getScene():Scene
setGamelobby(String):void
getBtnTrumpf():Button
getUserNamePl1():Label
getUserNamePl2():Label
getUserNamePl3():Label
getUserNamePl4():Label
getLblTrumpf():Label
getLblStichsT1():Label
getLblScoreT1():Label
getLblStichsT2():Label
getLblScoreT2():Label
getLblPl1():Label
getLblPl3():Label
getLblPl2():Label
getLblPl4():Label
getLblRound():Label
<<Java Class>>
PlayerPane
jassmend
lblName: Label
lblScore: Label
cardBox: HBox
cardsHolder: ArrayList<Card>
player: Player
PlayerPane()
setPlayer(Player):void
getCardBox():HBox
updatePlayerDisplay():void
getCardsHolder(int):Card
setCardsHolder(Card):void
setCardsHolderNull(int):void
getCardsHolderSize():int
clearCardsHolder():void
<<Java Class>>
JassmendMainMenuView
jassmend
scene: Scene
volumeSlider: Slider
musicFile: Media
musicPlayer: MediaPlayer
finalGamelobby: String
userName: String
lblUserName: Label
createGameLobby: Button
joinGameLobby: Button
hBoxMusicButton: HBox
hBoxLobby: HBox
hBoxBoard: HBox
hBoxLogout: HBox
showLobby: Button
playMusic: Button
pauseMusic: Button
stopMusic: Button
logOut: Button
insets: Insets
gridMainMenuView: GridPane
JassmendMainMenuView(JassmendController,JassmendGameView,JassmendLoginView)
joinGameLobby():void
createGameLobby():void
getScene():Scene
getFinalGamelobby():String
getLblUserName():Label
<<Java Class>>
JassmendController
jassmend
socket: Socket
gamelobbyFlag: int
playerIDs: String[]
gamelobbyName: String
myCurrentPlayerHand: String[]
handCards: ArrayList<Card>
socketIn: BufferedReader
socketOut: OutputStreamWriter
playerTurns: String[]
cardsPlayed: ArrayList<Card>
btnToActivate: int
cardPlayedNr: int
stichsTeam1: int
stichsTeam2: int
roundcounter: int
pointsTeam1: int
pointsTeam2: int
firstPlayer: int
stichsCounter: int
trumpf: String
image5: Image
image6: Image
image7: Image
image8: Image
connect(String,String):boolean
appendMessage(String):void
appendMessageGameView(String):void
setText_IPField(String):void
setText_PortField(String):void
connect():boolean
validateIpAddress(String):boolean
validatePortNumber(String):boolean
sendToServer(String):void
registerUser(String,String):void
gamelobbyIsFull(String,String):void
loginUser(String,String):void
loginSuccessfull(String,String):void
getUsername():String
getGamelobbyList():void
comparePlayerIDs():void
setMyCurrentPlayerHand(int,String):void
setGameConfig():void
getStiche(String):void
getEvaluation(String):void
getGamelobbyUsers(String):void
startGame(String):void
joinGamelobby(String):void
dealCards(String):void
turnFinished(String):void
sendCardPlayed(String,String,String):void
sendTrumpf(String,String):void
leaveGamelobby(String):void
createGamelobby(String):void
deleteAccount():void
changePassword(String):void
ping():void
getPlayerNames(int):String
logout():void
logoutUser(String):void
sendMessage(String):void
joinSuccessfull(String):void
getJassmendLoginView():JassmendLoginView
showAlert(String,String):void
showAlertYesNo(String,String):void
JassmendController(JassmendModel)
addMainMenuView(JassmendMainMenuView):void
addChatView(JassmendLogView):void
addLoginView(JassmendLoginView):void
addGameView(JassmendGameView):void
addTrumpfView(JassmendTrumpfView):void
setViewManager(JassmendViewManager):void
getViewManager():JassmendViewManager
getPlayerIDs(int):String
joinedGamelobbyMode():void
buttonsFalse():void
buttonsTrue():void
btnTrumpfFalse():void
btnTrumpfTrue():void
isFull():boolean
applyCardImages():void
switchNumbersToWords(String):String
stringToCard(String):Card
getBtnToActivate():int
getCardsPlayed(int):Card
getSizeCardsPlayed():int
clearCardsPlayed():void
getCardPlayedNr():int
whatismin(int):void
getFinalGamelobby():String
getGameView():void
setTextlblRoundCount():void
setTextlblUsername():void
getLblTrumpf():Label
setTextlblUsernameMenu():void
getFirstPlayer():String
setFirstPlayer(int):void
getPointsTeam1():int
getPointsTeam2():int
getStichsTeam1():int
setStichsTeam1(int):void
getStichsTeam2():int
setStichsTeam2(int):void
getRoundcounter():int
setRoundcounter(int):void
setSocket(Socket):void
getSocket():Socket
readLastMessage(String):boolean
getAreaMessages():TextArea
<<Java Class>>
CardView
jassmend
CardView()
setCard(Card):void
cardToFileName(Card):String
<<Java Class>>
JassmendTrumpfView
jassmend
trumpfWindow: Stage
secondScene: Scene
lblEckle: Label
lblSchaufel: Label
lblHerz: Label
lblKreuz: Label
btnEcke: Button
btnSchaufel: Button
btnHerz: Button
btnKreuz: Button
lblSpace: Label
image1: Image
image2: Image
image3: Image
image4: Image
image5: Image
image6: Image
image7: Image
image8: Image
JassmendTrumpfView(JassmendController)
getSceneTrumpfView():Scene
getImage5():Image
getImage6():Image
getImage7():Image
getImage8():Image
<<Java Class>>
JassmendModel
jassmend
localHost: InetAddress
localIP: String
clientPlayerID: int
clientTurnPlayerID: int
defaultIPAddress: String
defaultPortNumber: int
ipAddress: String
port: int
user: String
hash: String
currentGamelobby: String
logger: Logger
players: ArrayList<Player>
deck: Deck
setIpAddress(String):void
getIpAddress():String
setPort(int):void
getPort():int
log_info(String):void
log_error(String):void
log_warning(String):void
setUser(String):void
setClientPlayerID(int):void
getClientPlayerID():int
setClientTurnPlayerID(int):void
getClientTurnPlayerID():int
getUser():String
setHash(String):void
getHash():String
getCurrentGamelobby():String
setCurrentGamelobby(String):void
JassmendModel()
getPlayer(int):Player
getDeck():Deck
-clientView 0..1
-primaryStage
-jassmendAccountView 0..1
-jassmendLoginView 0..1
-view 0..1
-logView
-jassmendLogView
-jassmendGameView 0..1
-jassmendMainMenuView 0..1
-jassmendController 0..1
-jassmendModel
-cardView
-jassmendTrumpfView


<<Java Class>>
JassmendDatabase
jassmendDatabase
sqlUrl: String
sqlUser: String
sqlPw : String
insertQuery: String
checkQuery: String
userName: String
userPassword: String
jassmendController: JasmendController
dateLastRequest: Date
createAccView : JassmendAccountView
myConnection: Connection
rs: ResultSet
stmt: PreparedStatement
stmt2: PreparedStatement
accCreationOk: boolean
userNameLogin: String
userPasswordLogin: String
localIP: String
errorCode: String
JassmendDatabase()
getInstance():JassmendDatabase
connectDB():boolean
clearResource():void
checkRegistratedAccDB(String):boolean
registerAccDB(String,String):void
checkLogin(String,String):void
getIpAdress():String
generateStrongPasswordHash(String):String
getSalt():byte[]
toHex(byte[]):String
validatePassword(String,String):boolean
fromHex(String):byte[]
infoBox(String,String):void
infoBox(String,String,String):void
getLocalIp():String
setLocalIp(String):void
getDataBaseDriver():JassmendDatabase
setDataBaseDriver(JassmendDatabase):void
getpWord():String
setpWord(String):void
getuName():String
setuName(String):void
getCn():Connection
setCn(Connection):void
getStmt():Statement
setStmt(PreparedStatement):void
getRs():ResultSet
setRs(ResultSet):void
getDateLastRequested():Date
setDateLastRequested(Date):void
getErrorCode():String
setErrorCode(String):void
getAccUser():String
setAccUser(String):void
getAccPw ():String
setAccPw (String):void
isAccCreationOk():boolean
setAccCreationOk(boolean):void
-databaseDriver
0..1


<<Java Class>>
DeckofCardsTest
jassmendModelClasses
cardsRemaining: SimpleIntegerProperty
HAND_SIZE: int
DeckofCardsTest()
main(String[]):void
deckOfCardsConstructor():void
dealCards():void
dealHands():void
sortCollection():void
<<Java Class>>
DeckOfCards
jassmendModelClasses
cardsRemaining: SimpleIntegerProperty
DeckOfCards()
getCardsRemainingProperty():SimpleIntegerProperty
getCardsRemaining():int
shuffle():void
dealCard():Card
isEmpty():boolean
<<Java Class>>
Player
jassmendModelClasses
HAND_SIZE: int
playerName: String
playerScore: int
Player(String,int)
getPlayerScore():int
getPlayerName():String
setPlayerName(String):void
getCards():ArrayList<Card>
addCard(Card):void
discardHand():void
getNumCards():int
<<Java Class>>
CardTest
jassmendModelClasses
CardTest()
main(String[]):void
cardToString():void
cardEquals():void
comparisonRank():void
comparisonSuit():void
-handCards
0..*
-cards
0..*
<<Java Class>>
Card
jassmendModelClasses
herzValue: int
eckeValue: int
schaufelValue: int
kreuzValue: int
compareTo(Card):int
Card(Suit,Rank)
getSuit():Suit
getRank():Rank
toString():String
add(Card):boolean
addAll(Collection<Card>):boolean
clear():void
contains(Object):boolean
containsAll(Collection<?>):boolean
isEmpty():boolean
iterator():Iterator<Card>
remove(Object):boolean
removeAll(Collection<?>):boolean
retainAll(Collection<?>):boolean
size():int
toArray():Object[]
toArray(T[])
completeSuitValue():void
trumpfHerzSuitValue():void
trumpfKreuzSuitValue():void
trumpfEckeSuitValue():void
trumpfSchaufelSuitValue():void
firstCardHerzSuitValue():void
firstCardKreuzSuitValue():void
firstCardEckeSuitValue():void
firstCardSchaufelSuitValue():void
getSuitValue():int
~card
0..1
-cards
0..*
-suit
0..1
-rank
0..1
<<Java Enumeration>>
Suit
jassmendModelClasses
Kreuz: Suit
Ecke: Suit
Herz: Suit
Schaufel: Suit
Suit()
toString():String
<<Java Enumeration>>
Rank
jassmendModelClasses
Sechs: Rank
Sieben: Rank
Acht: Rank
Neun: Rank
Zehn: Rank
Bube: Rank
Dame: Rank
Koenig: Rank
Ass: Rank
Rank()
toString():String


«Java Class»
CleanupThread
Server
logger: Logger
CleanupThread()
run():void
«Java Class»
ServerController
Server
card: Card
cardsTotalCard: ArrayList<Card>
ServerController()
switchNumbersToWords(String):String
stringToCard(String):Card
trumpfComparison(String):void
firstCardComparison(String):void
getCardsTotalCard(int):Card
setCardsTotalCard(Card):void
sortTotalCards():void
clearTotalCards():void
handleStiche(Gamelobby):void
getPoints(Gamelobby,Card):int
~gamelobby
0..1
«Java Class»
Gamelobby
Server
serialVersionUID: long
logger: Logger
jasmendModel: JasmendModel
name: String
owner: String
isPublic: boolean
users: ArrayList<String>
lastMessage: Instant
maxPlayers: int
playerIDs: String[]
playerHand: ArrayList<Card>
cardsDealt: ArrayList<String>
turnCounter: int
cardsInRound: String[]
roundCounter: int
cardsTotalString: ArrayList<String>
stricheTeam1: ArrayList<Card>
stricheTeam2: ArrayList<Card>
cardCounter: int
trumpf: String
team1Members: String[]
team2Members: String[]
firstCardInTurn: String
userPlayingCard: String
cardsWithNames: ArrayList<String>
stichWinner: String
stichCounter: int
pointsTeam1: int
pointsTeam2: int
add(Gamelobby):void
remove(Gamelobby):void
listPublicNames():ArrayList<String>
exists(String):Gamelobby
cleanupgamelobbys():void
savegamelobbys():void
readgamelobbys():void
Gamelobby(String,boolean,String)
getName():String
send(Message):void
equals(Object):boolean
compareTo(Gamelobby):int
getOwner():String
isFull():boolean
isPublic():boolean
addUser(String):void
deal():void
getPlayerHand(int):Card
getTurnCounter():int
increaseTurnCounter():void
getPlayerIDs(int):String
clearPlayerHand():void
removeUser(String):void
getUsers():ArrayList<String>
getDeck():Deck
addCard(Card):void
setCardsInRound(int,String):void
getCardsInRound(int):String
getRoundCounter():int
increaseRoundCounter():void
resetRoundCounter():void
getCardCounter():int
decreaseCardCounter():void
increaseCardCounter():void
resetCardCounter():void
setStricheTeam1(Card):void
setStricheTeam2(Card):void
getStricheTeam1(int):Card
getStricheTeam2(int):Card
getSizeStricheTeam1():int
getSizeStricheTeam2():int
getSizeCardsInRound():int
addToCardsTotal(String):void
getCardsTotal(int):String
setTrumpf(String):void
getTrumpf():String
addToTeam1(int,String):void
addToTeam2(int,String):void
getTeam1Members(int):String
getTeam2Members(int):String
getUserPlayingCard():String
setUserPlayingCard(String):void
addCardsWithNames(String):void
getCardsWithNames(int):String
clearCardsWithNames():void
clearTotalCards():void
getFirstCardInTurn():String
setFirstCardInTurn(String):void
clearCardsDealt():void
getCardsDealt(int):String
setCardsDealtNull(int):void
clearStricheTeams():void
setStichWinner(String):void
getStichWinner():String
getStichCounter():int
increaseStichCounter():void
resetStichCounter():void
getPointsTeam1():int
setPointsTeam1(int):void
getPointsTeam2():int
setPointsTeam2(int):void
~gamelobbys
0..*
~deck
0..1
«Java Class»
Server
Server
logger: Logger
port: int
homeDirectory: String
Server()
main(String[]):void
setupLogging():void
readOptions():void
getHome():String
«Java Class»
Deck
Server
cards: ArrayList<Card>
cardsRemaining: SimpleIntegerProperty
Deck()
getCardsRemainingProperty():SimpleIntegerProperty
getCardsRemaining():int
shuffle():void
dealCard():Card
isEmpty():boolean
«Java Class»
ListenerThread
Server
logger: Logger
listener: ServerSocket
port: int
ListenerThread(int)
run():void
«Java Class»
Client
Server
logger: Logger
token: String
socket: Socket
clientReachable: boolean
lastUsage: Instant
playerID: int
add(Client):void
exists(String):Client
cleanupClients():void
Client(Socket)
getName():String
send(Message):void
getAccount():Account
setAccount(Account):void
getToken():String
setToken(String):void
getSocket():Socket
getLastUsage():Instant
~clients
0..*
~account
0..1
«Java Class»
Account
Server
serialVersionUID: long
logger: Logger
usersLoggedIn: ArrayList<String>
rand: SecureRandom
iterations: int
username: String
salt: byte[]
hashedPassword: String
lastLogin: Instant
hexArray: char[]
add(Account):void
remove(Account):void
exists(String):Account
cleanupAccounts():void
saveAccounts():void
readAccounts():void
getToken():String
Account(String,String)
equals(Object):boolean
checkPassword(String):boolean
changePassword(String):void
getUsername():String
checkForUsersLoggedIn(String):boolean
setUsersLoggedIn(String):void
removeUsersLoggedIn(String):void
hash(String):String
bytesToHex(byte[]):String
~accounts
0..*
«Java Interface»
Sendable
Server
getName():String
send(Message):void


«Java Class»
Message
Server message
logger: Logger
data: String[]
Message(String[])
Message(String[], ArrayList<String>)
process(Client): void
send(Socket): void
receive(Socket): Message
toString(): String
«Java Class»
MessageError
Server message
MessageError()
process(Client): void
«Java Class»
StartGame
Server message
token: String
name: String
username: String
StartGame(String[])
process(Client): void
«Java Class»
GameLobbyIsFull
Server message
token: String
name: String
username: String
GameLobbyIsFull(String[])
process(Client): void
«Java Class»
GetStiche
Server message
token: String
name: String
username: String
stichWinner: String
sticheTeam1: String[]
sticheTeam2: String[]
stichCounter: int
GetStiche(String[])
process(Client): void
«Java Class»
TurnManager
Server message
token: String
name: String
username: String
roundCounter: String
TurnManager(String[])
process(Client): void
«Java Class»
Evaluation
Server message
token: String
name: String
username: String
pointsTeam1: int
pointsTeam2: int
points1: int
points2: int
count1: int
count2: int
Evaluate(String[])
process(Client): void
«Java Class»
JoinGameLobby
Server message
token: String
name: String
lobbyName: String
JoinGameLobby(String[])
process(Client): void
«Java Class»
DealCards
Server message
token: String
name: String
username: String
DealCards(String[])
process(Client): void
«Java Class»
Trumpf
Server message
token: String
name: String
username: String
trumpf: String
Trumpf(String[])
process(Client): void
«Java Class»
CardPlayed
Server message
token: String
name: String
username: String
CardPlayed: String
firstPlayer: String
CardPlayed(String[])
process(Client): void
«Java Class»
LogOut
Server message
LogOut(String[])
process(Client): void
«Java Class»
LogoutUser
Server message
username: String
LogoutUser(String[])
processM(Client): void
«Java Class»
Ping
Server message
token: String
Ping(String[])
process(Client): void
«Java Class»
DeleteLogin
Server message
token: String
DeleteLogin(String[])
process(Client): void
«Java Class»
CreateLogin
Server message
username: String
passw ord: String
CreateLogin(String[])
process(Client): void
«Java Class»
DeleteGamelobby
Server message
token: String
name: String
DeleteGamelobby(String[])
process(Client): void
«Java Class»
IfNotOnline
Server message
token: String
username: String
UserOnline(String[])
process(Client): void
«Java Class»
CreateGamelobby
Server message
token: String
name: String
isPublic: boolean
CreateGamelobby(String[])
process(Client): void
«Java Class»
ListGamelobbys
Server message
token: String
ListGamelobbys(String[])
process(Client): void
«Java Class»
LeaveGamelobby
Server message
token: String
name: String
username: String
LeaveGamelobby(String[])
process(Client): void
«Java Class»
ChangePassword
Server message
token: String
passw ord: String
ChangePassw ord(String[])
process(Client): void
«Java Class»
ListGamelobbyUsers
Server message
token: String
name: String
ListGamelobbyUsers(String[])
process(Client): void
«Java Class»
Login
Server message
username: String
passw ord: String
Login(String[])
process(Client): void
«Java Class»
SendMessage
Server message
token: String
destination: String
message: String
SendMessage(String[])
process(Client): void
«Java Class»
SendGameMessage
Server message
token: String
destination: String
message: String
SendGameMessage(String[])
process(Client): void
«Java Class»
MessageGameText
Server message
MessageGameText(String,String,String)
process(Client): void
«Java Class»
Result
Server message
Result(boolean)
Result(boolean,String)
Result(boolean,ArrayList<String>)
process(Client): void
«Java Class»
MessageText
Server message
MessageText(String,String,String)
process(Client): void