Brändi Dog 2.0 Architektur

Generell

Generell ist das Programm unterteilt in ein Package Client (mit Sub-Packages) und ein Package Server. Ausserdem gibt es noch das Network Package mit unserem Protokoll und ein Game Package, in welchem sich Spieldaten bezüglich Karten und Brett befinden. Das Spiel selbst im src-Verzeichnis gespeichert und die zugehörigen Tests im test-Verzeichnis. Game Assets und JavaFX fxml-Datien sowie die in-Game Version der Spielanleitung und das Intro-Video, die der Client alle benötigt, werden in einem separaten Verzeichnis resources abgelegt. Zur Erläuterung eine Darstellung der Packagestruktur, mit den Klassen, die in jedem Package zu finden sind.

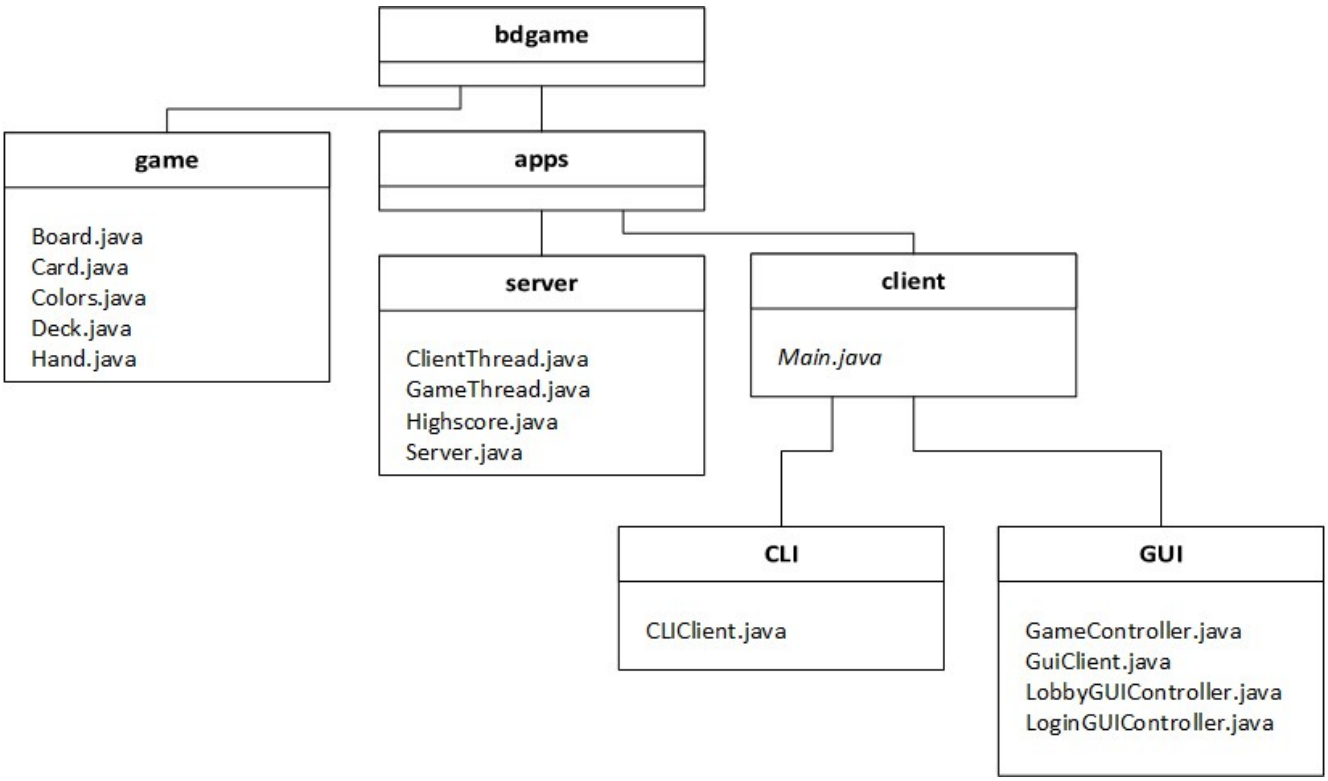

Abbildung 1: Darstellung der Packagestruktur von Brändi Dog 2.0

Im Folgenden sollen nun die Packages und deren Klassen im Detail betrachtet werden.

Package bdgame.apps.client

Das client package hat, neben den Subpackages CLI und GUI nur die Klasse **Main**.

Klasse Main

Main ist, wie der Name bereits sagt, die Main-Klasse des Spiels und wird beim starten der .jar aufgerufen. In ihr werden die Parameter beim Programmstart geparsed und entsprechend wird ein Client oder ein Server gestartet. Wird ein Client gestartet, öffnet sich das Login-Fenster des GUI Packages, wird ein Server gestartet, öffnet sich der Server und läuft im Hintergrund.

Package bdgame.apps.client.CLI

In diesem Legacy-Package ist nur die Klasse **CLIClient** zu finden.

Klasse CLICLient

CLICLient ist ein Chat Client mit Comand Line Interface. Es handelt sich um Legacy Code, der nur zu debugging Zwecken verwendet wird. Um sie zu starten, muss Code in der **Main** Klasse des bdgame.apps.client Packages geändert werden.

Metoden:

- Void send(String)
- Void showHelp()
- String buildMessage(String[], int, String)

Package bdgame.apps.client.GUI

Das GUI Package enthält einen Chat- und Spiel-Client mit graphischer Benutzeroberfläche. Es enthält die drei JavaFX Controller-Klassen **GameController**, **LobbyController** und **LoginController**, sowie die Klasse **GUIClient**. Als erstes gelangt der User zum Login-Fenster und nach erfolgreichem Login startet das Chat-Fenster. Wird schliesslich ein Spiel gestartet, öffnet sich das Spiel-Fenster.

Klasse GUIClient

Die **GUICLient** Klasse ist die Main-Klasse des graphischen Clients. Sie wird gestartet, wenn das Programm ohne Parameter oder mit den Client-spezifischen Parametern („client“ serverIP:Port) gestartet wird. Von hier aus werden die drei Hauptfenster (Login, Chat, Spiel) geöffnet und auch die Client-Server-Verbindung verwaltet.

Methoden:

- Void initGuiClient()
- Void start()
- Void stop()
- Void prepareLobby()
- Void prepareGameUI()
- Void startLogin(String[])
- String loginOnServer(String)
- Void StartRecieving()
- Stage getLoginStage()
- Stage getLobbyStage()
- Stage getGameStage()
- Void messageServer(String)
- Void processProtocol(String)
- String getServerInfo()
- Void requestClientList()
- Void requestLobbyList()
- Void requestScoreBoard()
- Void requestNicknameChange(String)
- Void enterLobby(String)

Klasse GameController

Die Klasse **GameController** ist das graphische JavaFX-Interface, mit welchem der User das Spiel spielt. Alle Interaktionen mit dem Spiel geschehen hier.

Methoden:

- Void initCards()
- Void initHomes()
- Void initPits()
- Void initFields()
- Void initTokens()
- Void initPlayerCards()
- Void showError(String), showInfo(String)
- Void sendActionToServer(String)
- Void setGameEnd(String)
- Void openDialogForCardAction()
- Void setTurn(String, String)
- Void clearHand()
- Void setOwnColor(String)
- Void serTokenPlayable(String, Boolean)
- Void playCard(Button)
- Void setLastPlayedCard()
- Void MoveToken(ImageView, Pane)
- Void setTokenPosition(String, String)
- Void assignCard(Button, Image)
- Void addCardToHand(String)
- Void removeCardFromHand(String)
- Void CreateErrorAlert()
- Void openManual()
- Void exitGame()

Klasse LobbyGUIController

Die **LobbyGUIController** Klasse ist das graphische JavaFX-Interface mit dem der User mit dem Chat interagiert (Nachrichten sendet/empfängt, Lobbies beitritt, Spiele startet). Sie enthält auch die private Klasse **ScoreEntry**, die die Highscores aus einem einfachen String in eine menschenlesbare Tabelle überträgt.

Methoden:

- Void mouseDragsMenuBar(MouseEvent)
- Void mousePressedOnMenuBar(MouseEvent)
- Void handleEnterPressed(KeyEvent)
- Void toggleReady(ActionEvent)
- Void clickedConnectItem(ActionEvent)
- Void clickedCreateLobbyItem(ActionEvent)
- Void clickedNicknameItem(ActionEvent)
- Void clickedCloseItem(ActionEvent)
- Void clickedScoreboardItem(ActionEvent)
- Void clickedInfoItem(ActionEvent)
- Void clickedOnServerChatTab(Event)
- Void lobbiesSelected(Event)
- Void usersSelected(Event)
- Void initialize(URL, ResourceBundle)
- Void appendServerMessageArea(String)
- Void privateMessageRecieved(String)
- Void lobbyMessageRecieved(String)
- Void updateClientList(HashSet<String>)
- Void toggleReadyLock()
- Void startPrivateChat(String)
- Void addLineToTextArea(TextArea, String)
- Void updateNicknames(String, String)
- Void updateScoreboard(String)
- Void updateLobbyList(String)
- Void createErrorAlert(String)
- Void enterLobby(String)
- Void openGameWindow()
- Void clear()

Klasse LoginGUIController

Diese Klasse stellt dem User das Login-Fenster zur Verfügung, wenn dieser das Programm startet oder den Server wechselt.

Methoden:

- Void initialize(URL, ResourceBundle)
- Void clickedLogin(ActionEvent)
- Void clickedExit(ActionEvent)
- Void closeLogin()
- Void createErrorAlert(String)
- Void setServerAddressField(String)
- Void setServerPortField(String)

Package bdgame.apps.server

Das Server Package enthält die nötigen Klassen um Client-Verbindungen, Chat und Spiele zu verwalten. Neben den Klassen **ClientThread**, **GameState**, **GameThread** und Server hat das Package auch die Klasse **Highscore**, die sich, wenig überraschend, um den Highscore kümmert.

Für jeden neuen Client wird ein **ClientThread** geschaffen und für jedes Spiel ein **GameThread** mit **GameState**.

Klasse Server

Die Klasse **Server** wird gestartet, wenn das Programm mit den nötigen Server Parametern („server“ Port) gestartet wird. In ihr wartet eine Endlosschlaufe auf neue Clients, die sich verbinden wollen und schafft ihnen einen **ClientThread**.

Klasse ClientThread

Die ClientThread Klasse ist der Manager eines einzelnen Clients. In ihr werden die nötigen Informationen zum Client (Adresse, Name, Lobby...) gespeichert sowie die nötigen Informationen über die anderen Clients, die zur Kommunikation notwendig sind.

Methoden:

- Void processProtocol(String)
- String enterLobby(String)
- Void leaveLobby(String)
- Void messageLobby(String)
- Void updateClients()
- Void updateClient(String, ClientThread)
- Void register(String)
- Void changeNickname(String)
- Void addClient(String, String)
- Void removeClient(String)
- String getClientLobby()
- String getClientName()
- String checkName(String)
- String checkNameSymbols(String)
- String getAllClients()
- String getAllLobbies()
- Boolean isLobbyReady(String)
- Void toggleReady()
- Void closeConnection()
- Boolean isReady()
- Void recieveMessage(String)
- Void broadcast(String)
- Void privateMessage(String, String)
- Void startGame()
- Void setGameThread(GameThread)

Klasse GameThread

In der Klasse GameThread wird ein laufendes Spiel verwaltet. Der Server nimmt die Spielzüge der Clients entgegen, überprüft die Legalität der Spielzüge und, falls sie legal sind, führt er diese aus und informiert die Clients über den geänderten Status. Zum Spielen wird ausserdem das Package bdgame.game benötigt, welches sich um Karten, Deck, Hand und Spielbrett kümmert.

Methoden:

- Boolean processAction()
- Boolean isValidMoveThenExecute()
- Boolean isAbleToPlay(String)
- Boolean retrievePlayerAction(String)
- Void updateScoreboard()
- Void messagePlayerByColor(String, String)
- Void messageAllPlayers(String)

Klasse Highscore

Die Klasse **Highscore** generiert oder lädt (je nach Vorhandensein) eine Highscore-Datei. Ausserdem werden die Highscores auf dem Laufenden gehalten und mit den Ergebnissen von neuen Spielen aktualisiert.

Methoden:

- Void incrementScore(String)
- String getScoreboard()
- Void checkForFile()
- Map<K,V> sortByValue(Map)

Package bdgame.game

In diesem Package findet sich das wieder, was auch im der Physikalischen Version des Spiels zu finden ist: Das Spielbrett, die Karten und die Murmeln. Die Karten werden in den Klassen **Card** (für die einzelne Karte), **Deck** (für das gesamte Kartendeck) und **Hand** (für die Karten, die ein Spieler auf der Hand hat) verwaltet. In der Klasse **Board** wird das Spielbrett und die Murmeln darauf verwaltet. Die Klasse **Colors** ist ein Enumerator mit den gültigen Spielerfarben.

Klasse Card

In der Klasse **Card** werden einzelne Spielkarten erstellt. Jede Karte hat einen Wert von Zwei bis Ass und eine Blattfarbe.

Methoden:

- String getValue()
- String toString()

Klasse Deck

In der Klasse **Deck** wird ein Kartendeck erstellt, die erstellten Karten werden an die Spieler ausgegeben und das Deck kann natürlich auch gemischt werden.

Methoden:

- Void shuffleDeck()
- Card dealCard()

Klasse Hand

Die Klasse **Hand** verwaltet die Karten die ein Spieler auf der Hand hat. Jede Hand kann Karten erhalten, Karten spielen und Karten wieder verlieren.

Methoden:

- Void addCard(Card)
- Void removeCard(Card)
- Void getSize()
- ArrayList<Card> getCardsInHand()
- Void emptyHand()

Klasse Colors

Der Enumerator in der Klasse **Colors** enthält die vier Farben der vier möglichen Spieler: Rot, Grün, Gelb, Blau (RED, GREEN, YELLOW, BLUE).

Klasse Board

Die Klasse **Board** behält die Übersicht über das Spielbrett. In ihr werden die einzelnen Felder und Murmeln aufbewahrt. Jede Murmel ist in einem Feld, entweder zuhause, im Spiel oder im Zwinger (Ziel). **Board** verwaltet auch den winstate, sprich, die Klasse weiss, wann wer gewonnen hat. Ausserdem ist zu beachten, dass die Klasse **Board** nur tut, was ihr gesagt wird. Um die Regeln kümmert sich der **GameThread** im bdgame.apps.server Package.

Methoden:

- String getTokenPosition(String)
- String getFieldEntry(String)
- Boolean isPitEntryField(String)
- Boolean isStartingFieldEmpty(String)
- Int getPitEntryFieldForColor(String)
- Void setTokenToStart(String)
- Void moveToken(String, String)
- String getWinner()
- Void switchTokens(String, String)

Package bdgame.network

In diesem Package ist unser Netzwerk-Protokoll zu finden. **Protocol** ist sie einzige Klasse. Natürlich greifen sowohl Client als auch Server auf dieses Package zu.

Klasse Protocol

Die Klasse Protocol enthält zwei Enumeratoren: ChatProtocol und GameProtocol. In ihnen sind sämtliche Befehle für Chat- resp. Spiel-Kommunikation enthalten.

Nähere Informationen zum Protokoll können der entsprechenden Dokumentation entnommen werden.