Inhaltsverzeichnis

[1. Abstract 2](#_Toc526510825)

[2. Zielsetzung 2](#_Toc526510826)

[4. Theorie 3](#_Toc526510827)

[4.1 Client-Server-Kommunikation 3](#_Toc526510828)

[4.2 TCP/IP 3](#_Toc526510829)

[4.3 Delegations-Eventmodell 3](#_Toc526510830)

[5. Tichu 4](#_Toc526510831)

[5.1 Die Sonderkarten 4](#_Toc526510832)

[5.1.1 Der Mah Jong 4](#_Toc526510833)

[5.1.2 Der Hund 4](#_Toc526510834)

[5.1.3 Der Drache 4](#_Toc526510835)

[5.1.4 Der Phönix 4](#_Toc526510836)

[5.2 Die Kombinationen 5](#_Toc526510837)

[5.3 Ablauf 5](#_Toc526510838)

[5.4 Spielende 5](#_Toc526510839)

[6. Design 6](#_Toc526510840)

[7. Implementierung 6](#_Toc526510841)

[7.1 Kommunikation 6](#_Toc526510842)

[7.2 Spiellogik 7](#_Toc526510843)

[7.3 Grafik 7](#_Toc526510844)

[8. Testing 8](#_Toc526510845)

[9. Fazit 8](#_Toc526510846)

[10. Anhang 8](#_Toc526510847)

1. Abstract

2. Zielsetzung

4. Theorie

4.1 Client-Server-Kommunikation

Ein Bild, das Text enthält.

Mit sehr hoher Zuverlässigkeit generierte BeschreibungWann immer zwei Geräte miteinander Daten austauschen, spielt die Client-Server-Kommunikation eine Rolle. Wie der Name schon sagt, geht es bei der Client-Server-Kommunikation um die Kommunikation zwischen Client und Server. Der Client ist dabei das Programm, das Anfragen an den Server sendet und mit welchem der Nutzer interagiert. Wogegen der Server das Programm ist, welches auf Anfragen des Cli-ents reagiert und antwortet. Beim Beispiel des Tichuspiels sendet der Client eine Anfrage an den Server, ob die vom Spieler gewählten Karten eine gültige Kombination ergeben, worauf der Server es überprüft und die Antwort an den Client zurückschickt. Der Server wartet immer auf einem vordefinierten Port auf eine Anfrage des Clients, wobei allen Clients, die mit dem Server interagieren wollen, der Port bekannt sein muss. Die Portnummer gehen von 0 bis 65535, wovon die Ports 0 bis 1023 für Systemdienste reserviert sind und nicht anders gebraucht werden dürfen.

4.2 TCP/IP

4.3 Delegations-Eventmodell

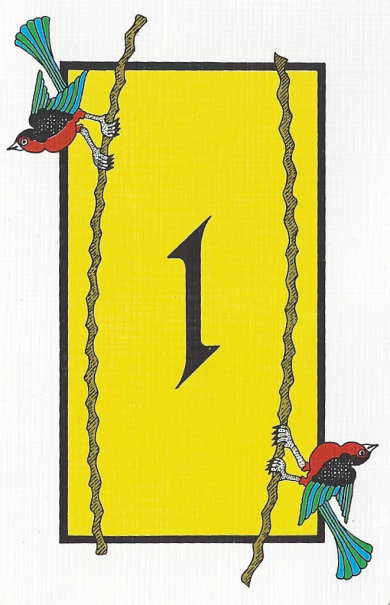
Sobald ein Programm nicht mehr sequentiell abläuft, da es auf Ereignisse, wie Tastatur- oder Mauseingaben, wartet, wird es als ereignisgesteuertes Programm bezeichnet. Ereignisgesteuerte Programme haben Methoden, die inaktiv sind, bis ein bestimmtes Event sie aktiviert. Solche Methoden nennt man Callbackmethoden. Ein Programm, welches Callbackmethoden verwenden will, muss einen Eventlistener implementieren, welcher beim Auftreten des erwarteten Events die zugehörige Callbackmethode aufruft. Die Callbackmethode wird in der Klasse, welche den Eventlistener implementiert überschrieben, wodurch selbst bestimmt werden kann, was passieren soll, sobald die Callbackmethode aufgerufen wird. Als Eventmodell bezeichnet man das Verfahren, mit welchem dieses Konzept mit Callbackmethoden programmiertechnisch umgesetzt wird. Da das klassische Eventmodell nicht objektorientiert ist und daher nicht wirklich in eine Klassenstruktur passt, entwickelten die Java-Entwickler in der Java Version 1.1 das Delegations-Eventmodell, welches viel besser zum objektorientierten Programmieren passte.

5. Tichu

Tichu ist ein Kartenspiel für vier Spieler, wobei jeweils die zwei einander gegenübersitzenden Spieler ein Team bilden. Gespielt wird Tichu mit 56 Karten: von Zwei bis Ass in vier Farben plus vier Sonderkarten.

5.1 Die Sonderkarten

5.1.1 Der Mah Jong

Der Mah Jong wird häufig auch Eins genannt und hat auch diese Höhe. Der Spieler, der beim effektiven Spielstart den Mah Jong auf der Hand hat, darf ausspielen. Dabei ist es egal ob er den Mah Jong oder eine andere Kombination spielt. Wird der Mah Jong als Einzelkarte gespielt - was neben der Strasse die einzige Kombination ist, in welcher der Mah Jong gespielt werden kann - so kann der Spieler, der ihn gespielt hat eine Karte Wünschen, die gespielt werden muss. Sonderkarten dürfen nicht gewünscht werden. Falls der nächste Spieler die gewünschte Karte hat muss er sie spielen. Falls er die gewünschte Karte nicht hat kann er eine andere Karte spielen und der nächste Spieler, der kann, muss die gewünschte Karte spielen. Zum Teil wird Tichu auch so gespielt, dass jeder, der die gewünschte Karte nicht hat, passen muss, bis jemand die gewünschte Karte spielen kann.

5.1.2 Der Hund

Der Hund ist die tiefste Karte im Spiel. Er kann nur ausgespielt werden, da er über keine andere Karte gespielt werden kann. Wird der Hund ausgespielt, so geht das Ausspielrecht an den Teampartner über. Falls der Teampartner bereits fertig ist, wird das Ausspielrecht nach rechts weitergegeben.

5.1.3 Der Drache

Der Drache ist die höchste Karte im Spiel, kann nur einzeln gespielt werden und ist 25 Punkte wert. Wird ein Stich mit dem Drachen gemacht, so muss der ganze Stich einem der Gegner gegeben werden.

5.1.4 Der Phönix

Der Phönix funktioniert wie ein Joker und kann folglich als beliebige andere Karte in Kombinationen eingebaut werden, nur in Bomben darf der Phönix nicht eingebaut werden. Wird der Phönix einzeln gespielt, so ist er immer ein halbes hoher als die zuvor gespielte Karte. Wird er Beispielsweise über eine Zwei gespielt, so hat er eine Wert von zweieinhalb. So kann er auch über ein Ass gespielt werden und ist somit nach dem Drachen die höchste Einzelkarte. Jedoch hat der Phönix einen Wert von -25 Punkten, damit er nicht zu stark ist.

5.2 Die Kombinationen

Beim Tichu gibt es viele verschiedene Kombinationen. Und zwar:

|  |  |
| --- | --- |
| Einzelkarte | Eine einzelne Karte |
| Paar | Zwei Karten von derselben Höhe |
| Drilling | Drei Karten von derselben Höhe |
| Full House | Ein Drilling und ein Paar, wobei die Höhe des Drillings die Höhe des Full House bestimmt |
| Treppe | Zwei oder mehr direkt auf einander folgende Paare |
| Strasse | Fünf oder mehr direkt auf einander folgende Karten. Nur eine gleichlange höhere Strasse kann eine andere Strasse schlagen |
| Bombe | Alle vier Karten derselben Höhe |
| Strassenbombe | Eine Strasse in nur einer Farbe. Eine längere Strassenbombe schlägt eine kürzere egal wie hoch sie ist. |

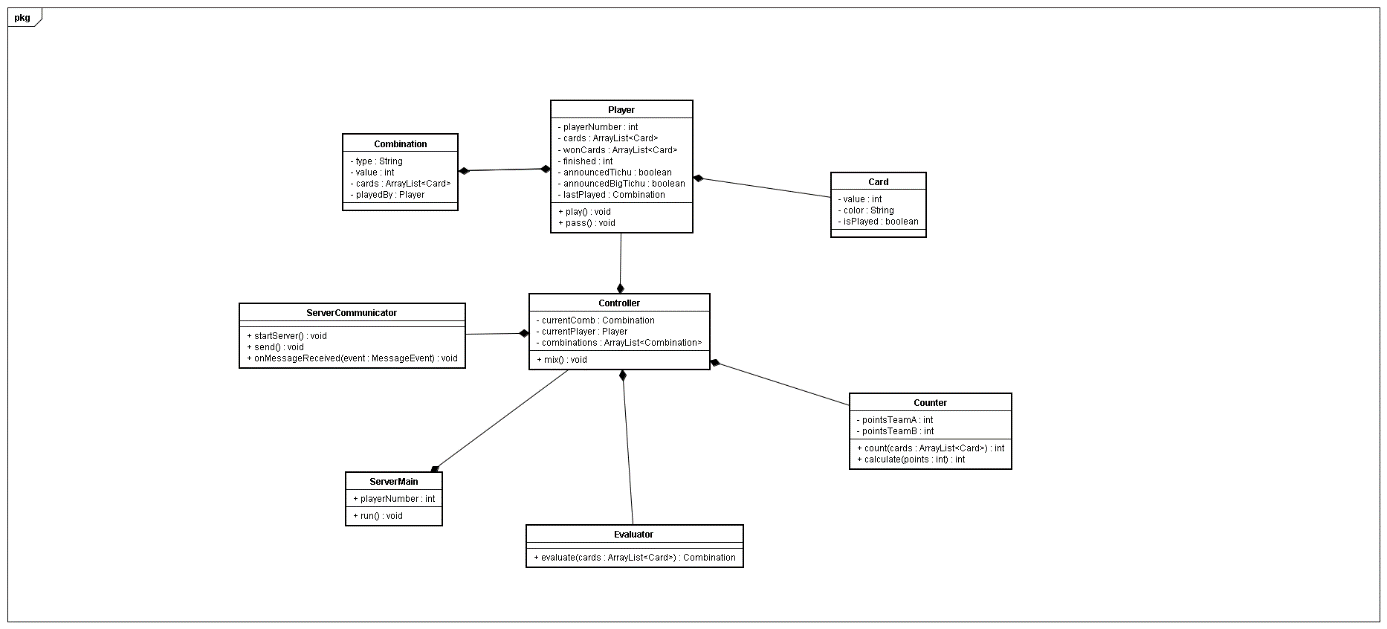
5.3 Ablauf

Wenn alle Spieler ihre Karten bekommen haben, beginnt das so genannte Schupfen, bei dem jeder Spieler jedem anderen je eine Karte gibt. Nach dem Schupfen beginnt der Spieler, der das Mah Jong in seinen Karten hat, mit dem Spielen, wobei es egal ist, ob er als erstes das Mah Jong oder eine andere Karte spielt. Auf eine gespielte Kombination kann nur eine höhere gleichartige Kombination oder eine Bombe bzw. Strassenbombe gespielt werden. Falls der Spieler, der an der Reihe ist nicht in der Lage ist über eine Kombination zu spielen oder es nicht möchte, so kann er passen. Wenn alle anderen Spieler gepasst haben, geht der Stich an den Spieler mit der zuletzt gespielten Kombination. Der Spieler, der den Stich gewonnen hat, hat danach das Ausspielrecht. Wenn ein Spieler keine Karten mehr auf der Hand hat, ist er fertig.

5.4 Spielende

Das Spiel endet, sobald beide Spieler eines Teams fertig sind. Falls noch keiner aus dem anderen Team fertig ist, haben sie einen Doppelsieg gemacht und bekommen 200 Punkte. Falls schon einer der Gegner fertig ist, gehen die Handkanten des Spielers, der nicht fertig wurde an das Team, das ganz fertig ist, und seine Stiche gehen an den Spieler, der als erstes fertig war. Dann werden die Punkte zusammengezählt und das Team mit mehr Punkten gewinnt die Partie. Gezählt wird so, dass die Fünfer 5, die Zehner und Könige 10, der Drache 25 und der Phönix -25 Punkte geben. Insgesamt gibt es also 100 Punkte.

6. Design

Nachdem Idee und Betreuer gefunden waren, war der nächste Schritt die Planung des Projektes. Zuerst wurde grob überlegt, welche Funktionen das Programm haben muss. Als nächstes wurde eine Programmiersprache gewählt, in der das Programm später implementiert werden sollte. Da der Autor sowohl im Ergänzungsfach Informatik, wie auch früher im Fakultativfach Programmieren, mit Java arbeitete, fiel die Wahl auf die Verwendung von Java, mit der Library Jeda, die bereits im Fakultativfach verwendet wurde, da diese Klassen zur grafischen Darstellung und zu Client-Server-Kommunikation beinhaltet. Die nächsten Schritte bestanden aus dem einlesen in die Client-Server-Kommunikation von Jeda und dem Erstellen eines Klassendiagramm.

Da beim Erstellen des Klassendiagramms immer wieder neue Dinge eingebaut werden mussten, dauerte es eine Weile bis es so weit ausgearbeitet war, dass mit der Implementation begonnen werden konnte. Vorher wurde aber noch eine Skizze der grafischen Oberfläche erstellt, die bei der Implementation als Orientierung diente.

7. Implementierung

7.1 Kommunikation

Die Client-Server-Kommunikation wurde mit den Klassen Connection und Server der Jeda Library umgesetzt. Als erstes muss im Server Programmteil ein neues Server-objekt erstellt und an einem angegebenen Port gestartet werden. Danach kann im Client Programmteil ein neues Connection-objekt erstellt werden und mit der open()-Methode eine Verbindung zum Server auf angegebenem Port hergestellt werden. Wenn eine Verbindung zum Server hergestellt wurde, erhält der Server ein ConnectionEvent aus dem er ein Connection-Objekt, welches der Verbindung zum Client entspricht, erstellen kann, über welches er mit dem Client kommunizieren kann. Client und Server kommunizieren mithilfe von Strings miteinander. In der untenstehenden Tabelle sind die Strings, die zur Kommunikation gebraucht werden, und deren Bedeutung abgebildet.

|  |  |  |  |
| --- | --- | --- | --- |
| Übermittelt | Notation | Erklärung | Sender |
| Karten des Spielers | «Cards:k1,k2,…,k14» | k1,…,k14 Karten-IDs | Server |
| Karten, die Spieler schupft | «x:SchupfCards:k1,k2,k3» | k1,k2,k3: Karten-IDs, x: Spielernummer | Client |
| Karten, die Spieler spielen will | «x:Play:k1,…,kn» | 0<n<15, k1,…,kn: Karten-IDs, x: Spielernummer | Client |
| Spieler passt | «x:Pass» | x: Spielernummer | Client |
| Problem beim Spielen der Karten | «Error:Text» | Text: Grund für Problem | Server |
| Neu gespielte Karten | «Played:k1,…,kn» | 0<n<15, k1,…,kn: Karten-IDs | Server |
| Karten, die Spieler geschupft werden | «SchupfedCards:k1,k2,k3» | k1,k2,k3: KartenIDs | Server |
| Spieler ist an der Reihe | «YourTurn:true» |  | Server |
| Spieler ist nicht mehr an der Reihe | «YourTurn:false» |  | Server |
| Spieler hat gepasst | «Passed:x» | x: Spielernummer | Server |
| Nachricht | «Message:Text» | Text: Nachricht | Server |
| Spieler passt | «x:Pass: » | x: Spielernummer | Client |
| Spieler gewinnt Runde | «x:Won:Round» | x: Spielernummer | Client |

Da Strings zur Kommunikation verschickt werden, werden also beim Spielen nicht die Kartenobjekte zwischen Client und Server verschoben, sondern nur ihre ID und die Anweisung, was damit gemacht werden soll.

7.2 Spiellogik

Das wichtigste Element für das Implementieren der Spiellogik ist die Evaluator-Klasse, welche überprüft, ob die vom Spieler ausgewählten Karten eine gültige Kombination ergeben. Das funktioniert so, dass zuerst überprüft wird, wie viele Karten ausgewählt wurden und danach die für diese Anzahl möglichen Kombinationen geprüft werden und danach noch geprüft wird, ob die Kombination hoch genug ist.

7.3 Grafik

Die grafische Oberfläche wurde mithilfe der Klassen View und Canvas aus der Jeda Library implementiert. Nach jedem Spielzug wird die Methode draw() aufgerufen, welche das ganze Spielfeld neu zeichnet. Also die eigenen Karten, die zuletzt gespielten Kombinationen der anderen Spieler und die Knöpfe zum Spielen und Passen oder zum Schupfen.

Beispielcode 1: Die Methode draw()

In der Methode draw() wird zuerst das ganze Fenster von einem weissen Rechteck bedeckt, was als Löschvorgang dient, beidem die alten Sachen von vorher überdeckt werden. Dies ist nötig, da die bereits gespielten Karten sonst teilweise weiterhin angezeigt werden. Nachdem das weisse Rechteck gezeichnet wurde wird mit einer for-Schleife durch die Karten des Spielers gewandert, und die Karten gezeichnet, wobei bei jeder Karte geprüft wird, ob sie angewählt ist oder nicht, und je nach dem die Grafik mit oder ohne gelbe Markierung gezeichnet. Danach wird die Textgrösse und -farbe gesetzt und bei jedem Spieler getestet, ob er gepasst hat, und gegebenen falls an der richtigen Stelle «Pass» geschrieben, und andernfalls die zuletzt gespielte Kombination gezeichnet.

8. Testing

9. Fazit

10. Anhang