# Haupt-Funktionalität:

Dieses Programm ist dafür da, Black Jack am Computer zu ohne echtes Risiko spielen.

Dabei verbindet sich der Client mit einem Server, der dann die Karten austeilt und als Dealer fungiert.

Es können mehrere Spieler gleichzeitig auf einem Server spielen, aber nicht miteinander Interagieren.

# Black Jack Regeln:

Blackjack ist ein Kartenspiel, bei dem das Ziel des Spiels darin besteht, eine Hand mit einem Wert von 21 oder möglichst nah an 21 zu erreichen, ohne diesen Wert zu überschreiten. Jeder Spieler spielt gegen den Dealer. Jeder Spieler erhält zu Beginn des Spiels zwei Karten, wobei eine der Karten des Dealers verdeckt bleibt. Der Wert der Karten in einer Hand besteht aus der Summe der Werte aller Karten in der Hand. Die Karten 2 bis 10 haben ihren numerischen Wert, Buben, Damen und Könige sind 10 Punkte wert, und Asse können entweder 1 oder 11 Punkte wert sein.

In Blackjack gibt es verschiedene Befehle, die der Spieler während des Spiels verwenden kann:

1. "Hit": Der Spieler möchte eine weitere Karte vom Dealer erhalten, um seine Hand zu verbessern.
2. "Stand": Der Spieler ist zufrieden mit seiner Hand und möchte keine weiteren Karten erhalten.
3. "Double": Der Spieler verdoppelt seinen Einsatz und erhält nur noch eine weitere Karte.
4. "Split": Wenn der Spieler zwei Karten mit dem gleichen Wert hat, kann er sie teilen und zwei separate Hände spielen.

# Setup & Starten des Programms:

Bei diesem Programm gibt es zwei Komponenten, den Client und den Server. Beide müssen individuell gestartet werden, funktionieren aber nur gemeinsam. Dabei ist es egal ob diese zwei Teile am selben oder auf unterschiedlichen PCs gestartet werde, sie müssen nur im selben Netzwerk sein.

**Server:**

* Der Server stellt den Dienst standardmäßig auf dem Port „6868“ zur Verfügung.
* Er hat keine grafische Oberfläche, gibt aber alle wichtigen Aktivitäten über die Konsole aus.

**Client:**

* Der Client muss sich demnach auf die IP-Adresse + Port des Servers verbinden.
* Die grafische Oberfläche wir automatisch vom Client selbst gestartet und synchronisiert sich ständig mit dem Thread, der mit dem Server kommuniziert.

Zurzeit sind alle Verbindungsdetails (IP & Port) im Code selbst enthalten.

# Aufbau & Komponenten:

**Server:**

Der Server an sich besteht aus zwei Teilen, die die Clients verwalten und die Last verteilen.

* Der Hauptteil <BlackServer> ist ein Endlos-Thread der jede Anfrage eines Clients übernimmt. Dabei öffnet er einen ServerSocket auf der localhost-Adresse am Port 6868.

Das der Server zwischen den einzelnen Anfragen weiter arbeiten kann wird ein Timeout gesetzt der den Server kurzzeitig weiterrechnen lässt.

Wenn ein Client eine Anfrage sendet startet der Server einen neuen Thread der Klasse <clientThread> dieser kümmert sich nun um die Kommunikation zum Client.

Außerdem wird die neue Verbindung in einer ArrayList gespeichert, die alle Aktiven Verbindungen beinhaltet, sowohl Socket als auch den Thread dazu.

* Der Nebenteil <clientThread> ebenfalls ein Endlos-Thread, dieser übernimmt die gesamte Kommunikation zum Client und die zugehörige Server-Logik.

Dabei wird die Kommunikation über einen BufferedReader und einen PrintWriter abgewickelt.

Falls nun eine Nachricht übertrage wird reagiert der Server entsprechend:

1. „startSession“

Der aller erste Befehl, der von einem Client gesendet wird, er bereitet den Server vor und stellt sicher das der Server Einsatzbereit ist.

1. "closeSession"

Der letzte Befehl, den der Client sendet, er beendet die Schleife und den Thread, danach wird der Eintrag (de aktiven Verbindung) im Hauptteil gelöscht.

1. Json-Message

Alle anderen Nachrichten sind als JSON codiert, sie beinhalten die Daten und Anfragen vom Client. Der JSON String besteht aus zwei Teilen „Type“ und „Value“.

Der Client schickt zwei Befehle über die JSON-Nachricht:

1. „draw“

Hier fordert der Client eine neue Karte an, der Server zieht aus dem Kartendeck eine neue Karte, codiert diese als JSON-String und sendet ihn zurück. Dabei benutzen alle Threads des Hauptservers dasselbe Kartendeck. Wird eine Karte gezogen wird sie logischerweise aus dem Deck entfernt, bis das Kartendeck neu gemischt wird.

Bsp. - Anfrage: „{"name":"draw","value":0}“

Bsp. - Antwort: „{"name":"10 of Clover","value":10}“

1. „stand“

Mit dieser Nachricht beendet der User seinen Zug, und sendet den vollen Wert seiner Kartenhand mit. Der Server generiert nun auf Basis des Wertes der Spieler-Kartenhand seine eigene Dealer-Kartenhand. Mit beiden Werten wird nun der Spielausgang errechnet, der Server sendet also den Status {„draw“, „win“, „loose“} mit dem Wert der Dealer-Kartenhand zurück.

Bsp. - Anfrage: „{"name":"stand","value":0}“

Bsp. - Antwort: „{"name":"win","value":18}“

**Client:**

Der Client besteht ebenfalls aus zwei Teilen:

* Der <BlackClient>, ein Endlos-Thread, der als erstes die Kommunikation mit dem Server aufbaut und dann die Eingaben des Users weiterleitet. Er startet auch direkt den zweiten Teil die GUI. Auch hier wird die Kommunikation über einen BufferedReader und einen PrintWriter abgewickelt.

Der Client fungiert nur als Übersetzer, er codiert alle Nachrichten zu und von JSON und gibt alle Daten an die GUI weiter. Beide Befehle {„draw“, „stand“} müssen zwei Mal behandelt werden, da der User auch die Möglichkeit hat seine Karten zu splitten.

Beim Ziehen einer Karte muss der GUI-Thread und der Client-Thread synchronisiert werden, damit alles reibungslos abläuft.

* Die <GUI> ist natürlich ihr eigener Thread und stellt das User-Interface zur Verfügung.

Darin werden alle bekannten Möglichkeiten eines Black Jack-Spiels implementiert. Es gibt vier Knöpfe für die Funktionen „Draw“, „Stand“, „Double“ und „Split“. Darüber hinaus gibt es ein Eingabefeld für den Wetteinsatz der nach jeder Runde neu berechnet und Angezeigt wird.

**Miscellaneous:**

Das sind alle Teile die für die Logik der Karten wichtig sind. Dazu gehören das Kartendeck, die Kartenhand und die Karte an sich. Prinzipiell sind alle drei nur einzelne Java-Objekte die in den anderen als ArrayList benutzt werden.

* Kartendeck

Das Kartendeck besteht aus einer ArrayList, die mit allen Arten von Karten initialisiert wird, also Ass bis König und von Herz bis Kreuz. Bei jeder neuen Ziehung einer Karte wird über einen zufälligen Wert die Karte bestimmt. Wenn das Deck leer ist wird es automatisch neu befüllt.

* Kartenhand

Die Kartenhand besteht ebenfalls aus einer ArrayList von Karten, die gezogen wurden. Dabei gibt es aber die Methoden, die die ganze Hand als JSON oder den gesamten Wert der Hand zurückgibt.

* Karte

Die Karte ist ein einfache Objekt, mit dem Karten Namen und dem Karten Wert.

Es gibt auch noch ein Java-Objekt <JsonObj> das für die Umwandlung zum JSON-String benötigt wird.

# Externe Ressourcen:

Es gibt nur eine Library die über Maven zusätzlich geladen wird und zwar GSON von Google.

Weitere Ressourcen die genutzt werden sind die Grafiken der Karten von „<https://www.kenney.nl/assets/playing-cards-pack>“