Übungsblatt 5

Aufgabe 1

2,3,4,5,6,7,8,9,10 zählen ihre aufgedruckten Punkte.

Dame, Bube und König zählen 10 Punkte.

Ass kann sowohl 1 Punkt als auch 11 Punkte zählen, immer zum Vorteil des Spielers. (Wird aber nicht vom Spieler entschieden)

Beim Black Jack spielt man immer gegen den Croupier und sich selbst, nie gegen andere Mitspieler.

Sobald man seine erste Karte erhält, können die Wetteinsetze nicht mehr verändert werden.

Man versucht steht so nah an 21 ranzukommen wie möglich. Falls der Spieler oder der Croupier mehr als 21 haben, hat dieser automatisch verloren, dies nennt man auch „Busting/Überkauft“

Bei Begin des Spiels erhält man 2 Karten.

Einen „Black Jack“ hat man, wenn die ersten beiden Karten gleich eine 21 ergeben.

Beim Black Jack gibt es folgende Speilzüge:

„Stay“: Keine Weitere Karte nehmen

„Hitting“: Weitere Karte nehmen

„Doubling Down“: Wenn man mit den ersten beiden Karten 9,10 oder 11 Punkte erhält, kann man seinen Wetteinsatz verdoppeln und bekommt dann nur eine Karte.

„Splitting“: Wenn die ersten beiden Karten den gleichen Zahlenwert haben, kann man „splitten“, dabei kann man die beiden Karten auf 2 Spiele aufteilen und den Wetteinsatz verdoppeln

„Surrender“: Wenn man seine ersten beiden Karten erhalten hat, kann man „surrender“, dann hat man seine Wette automatisch verloren bekommt aber die Hälfte des Einsatzes zurück. Dies geht nur solange der Croupier kein „Black Jack“ bekommt.

Aufgabe 2

Daten:

Um Probleme bei der Kommunikation zu vermeiden, müsste man genau festlegen in welcher Form man die Daten versendet. Ich würde hier zu JSON formatierten Strings tendieren.

Ein schnell überlegtes Beispiel könnte folgendermaßen aussehen:

```
{
        “sender“: “player“
        “receiver“: “croupier“
         “type“: “data“  // Oder “command”
        “value”: [

        ]
}
```

Kommunikation:

Da die Kommunikation über UDP läuft sollten man mit Timeouts arbeiten, um Pakete die eventuell verloren gehen wieder neu anzufragen und gegebenenfalls das Spiel zu beenden.

Weiter sollten man mit den Kommilitonen feste Namen für die einzelnen Befehle festlegen und die Möglichkeiten mitgelieferter Parameter klar definieren. Sollte eines der Programme ungültige Befehle erhalten, sollte es mit Error Nachrichten und einer Aufforderung der korrekten Syntax antworten.

Man könnte auch darüber nachdenken beim erstmaligen Verbinden, eine Session ID zu übertragen, so könnte man das andere Programm nicht nur über IP und Port identifizieren.\

Aufgabe 3

Mir fällt der Croupier zu.

- Erstellt Karten
    - 1-8 Decks
- Hat Karten in seiner Hand
- Berechnet, ob ein Spieler gewinnt
- Berechnet, wie viel ein Spieler gewinnt
- Spielt selbst auch (ohne Einsatz)
- Kommunikation AUSGEHEND
    - Anfrage Statistik vom Zähler
        - Für Ausschlussentscheidung über Spieler
    - Angabe der Gewinnsumme an Spieler
    - Sendet
- Kommunikation EINGEHEND
    - Angabe des Wetteinsatzes von Spieler
    - Anfrage vom Zähler über die Anzahl der Karten und welche ausgegeben wurden
    -

Aufgabe 4

Nachdem ich mich bei der Erstellung meines Programms nicht an meine anfänglich Formulierten Vorsätze gehalten habe und erstmal mit meinem treuen Begleiter ChatGPT drauflosgearbeitet habe, kam es dann, wie es kommen musste. Als ich mich mit Kommilitonen zusammengesetzt habe, um unsere Programme zu testen, konnten wir nur einige wenige „Funktionen“ austesten. Dies lag weniger an der Art der Kommunikation, sondern viel mehr daran, dass wir unterschiedliche Auffassungen davon hatten welcher Programmteil welche Aufgaben übernehmen soll. Dies führte dazu, dass wir einigen Funktionen gar nicht implementiert hatten und andere doppelt.

Wenn ich mir eine Anmerkung zu dieser Aufgabe erlauben dürfte. Käme für eine solche Aufgabe nicht vielleicht ein Spiel wie beispielsweise Poker viel gelegener? Hier könnte ein Kartengeber/Dealer (Server) vom Übungsleiter zur Verfügung gestellt werden und alle Studenten müssten dann einen eigenen Spieler (Client) entwickeln. Oder einige Studenten machen den Client und andere einen Server. 😊