Universität Trier

Fachbereich IV / Informatik

Prof. Dr. Peter Sturm
D-54286 Trier
Telefon 0651/201-3313
Fax 0651/201-3842
Email sturm@uni-trier.de

Philipp Müller
Email s4pimuel@uni-trier.de

Rechnernetze, Übungsblatt 5, Sommer 2025

Abgabe

Die Abgabe in Form eines Pull-Requests in Ihren Branch des Repositories https://github.com/syssoft-ds/blackjack-2025S muss bis zum 25.06.2025 um 23:59 Uhr geschehen.

Aufgabe 1: Blackjack

Informieren Sie sich über das Glücksspiel „Blackjack“ und darüber, wie Kartenzählen bei diesem Spiel funktioniert. Überlegen Sie sich, wie Sie ein Programm aufbauen würden, das aus 3 Klassen „Spieler“, „Croupier“ (Kartengeber), „Kartenzähler“ besteht. Alle 3 beteiligten Klassen sollten Ihre jeweilige Funktion mathematisch optimal bezüglich Blackjack ausführen (Tipp: „harte“ bzw. „weiche“ 17).

a) Der Croupier erstellt die Karten, bestehend aus 1-8 Kartendecks mit jeweils Rängen {2-10, Bube, Dame, König, Ass} sowie Typen {Pik, Kreuz, Herz, Karo}. Er hat in seiner Hand Karten und gibt Karten aus. Er berechnet, ob ein Spieler gewonnen hat und ob und wie viel Geld der Spieler bekommt. Ansonsten macht er dasselbe wie der Spieler, nur dass er keinen Einsatz berechnen muss. Der Croupier fragt außerdem vom Kartenzähler Statistiken an, um zu entscheiden, wann er den Spieler aus dem Casino wirft, weil er zu oft gewonnen hat. Der Croupier muss selbst entscheiden, welche Spielaktionen er tätigt, er bekommt keine Empfehlungen dazu vom Kartenzähler.

b) Der Spieler hat ein Startkapital, Karten in seiner Hand und kann Spielaktionen ausführen („Hit“, „Stand“, „Split“, „Double Down“, „Surrender“). Er muss einen Wetteinsatz an den Croupier geben und bekommt nach dem Spiel eine Gewinnausschüttung, je nachdem, wie gut er sich geschlagen hat. Die optimale Höhe des Einsatzes muss der Spieler berechnen. Er kann dazu Informationen vom Kartenzähler anfragen.

c) Der Kartenzähler fragt beim Croupier an, mit wie vielen Kartensets das Spiel gespielt wird, und welche Karten ausgegeben werden. Er berechnet aus diesen Informationen die Aktion mit dem höchsten Erwartungswert für den Spieler, und sendet diesem diese Empfehlung. Außerdem speichert er eine Statistik für jedes Spiel, wie oft der Spieler gewinnt, wie viele Blackjacks er punktet, und eine Liste der Karten, die insgesamt verwendet wurden. Diese Statistik gibt er bei Bedarf an Spieler oder Croupier.

Aufgabe 2: UDP

Überlegen Sie sich nun, was Sie beachten müssen, wenn die 3 Klassen aus Aufgabe 1 unterschiedliche Programme sind, die per UDP miteinander kommunizieren.

Auf welche Dinge müssen Sie achten, um sicherzustellen, dass die gesendeten Nachrichten beim Empfänger ankommen? Wie können Sie Fehlkommunikation vermeiden (wenn z.B. ein Paket falsch interpretiert wird). Wie können Sie sicherstellen, dass Ihr Programm mit den 2 anderen Programmen kommunizieren kann und die Pakete so interpretiert, wie Sie das geplant haben?

Nutzen Sie dafür die Erkenntnisse vergangener Aufgaben, in denen Sie UDP untersucht haben, und speziell die letzte Aufgabe des vorigen Übungsblattes, bei dem Sie gemeinsame Richtlinien für Ihre Programme spezifizieren mussten.

Aufgabe 3: Implementierung

Implementieren Sie nun eine der drei Klassen des Blackjack-Programms als eigenständiges Programm, das durch UDP mit den anderen beiden Programmen kommuniziert. Nutzen Sie die Überlegungen der Aufgaben 1 & 2.

a) Das „Spieler“-Programm soll von allen implementiert werden, deren Nachnamen mit N-Z beginnt.

b) Das „Croupier“-Programm soll von allen implementiert werden, deren Nachnamen mit D-M beginnt.

c) Das „Kartenzähler“-Programm soll von allen implementiert werden, deren Nachnamen mit A-C beginnt.

Jeder soll hier nur eines der drei Programme konstruieren, wobei es hilfreich sein wird, eng mit den anderen Gruppen zusammen zu arbeiten, um zu sehen, welche Informationen und in welcher Form von den anderen Programmen benötigt bzw. bereitgestellt wird.

Aufgabe 4: Kommunikation

Kommunizieren Sie mit den anderen Gruppen, z.B. über das bereitgestellte Google Docs, oder über beliebige andere Wege. Einigen Sie sich darauf, wie Sie die Elemente Ihrer Programme gestalten, die von den anderen Studis genutzt werden.

Probieren Sie schließlich aus, ob die Programme zusammen funktionieren. Wenn etwas nicht funktioniert, tauschen Sie sich mit den anderen Gruppen aus, warum dieser Fehler auftaucht, und wie man das Problem zusammen lösen kann. Reflektieren Sie diese Erfahrungen in einem Textdokument.

Aufgabe 1

Siehe google docs und Diagramme im branch unter *diagrams/* bzw. *out/*

Aufgabe 2

Es muss sichergestellt werden, dass nachrichten, die verschickt werden, ankommen. Das kann durch ein Quittungssystem sichergestellt werden, das für jede versendete spielbezogene Nachricht auf eine Quittung wartet. Damit das jeweilige Programm (Croupier, Kartenzähler oder Spieler) nicht zu lange wartet, sollte weiterhin ein Timeout für die Antwort festgelegt werden. Ferner kann die Kommunikation zwischen den 3 Programmen robuster gemacht werden, indem Nachrichten, die die zulässige Zeit (timeout) überschritten haben, erneut gesendet werden. Dabei sollte einerseits eine maximale Anzahl an *resends* festgelegt werden, z.B. 5 und andererseits sollte die Länge des Timeouts für jeden *resend* erhöht werden.
Um Fehlinterpretationen von Paketen zu vermeiden, sollte eine einheitliche Syntax für den Aufbau der Nachrichten sowie ein einheitliches Format z.B. Json festgelegt werden. ()Siehe google Docs.)
Außerdem soillte ein allgemeiner Programmablauf und die Punkte an denen Nachrichten ausgetauscht sowie welche Nachrichten an diesen Punkten ausgetauscht werden, festgelegt werden. Das verhindert, dass der Programmfluss stecken bleibt, da ein oder mehrere Teilnehmer auf Nachrichten warten, die nicht gesendet werden.

Aufgabe 3

Leider hat sich niemand, der/die einen Kartenzähler implementieren sollten an unserer Diskussion im google Docs beteiligt, entsprechend fehlt die Logik für die Kommunikation mit dieser Klasse / ist nur in Ansätzen in meiner Implementierung der Croupier-Klasse vorhanden.

Aufgabe 4

Haben wir gemacht. Waren gefühlt aber auch die einzigen 3 Teilnehmer, die das google Docs genutzt haben. Schade, weil die Aufgabe eigentlich echt Spaß gemacht hat

.