Protokolle aufgaben

**Protokoll Aufgabe 1:**

Kurze Aufgabenstellung: Ein Server Client Programm erstellen, indem der Client dem Server Rechnungen sendet. Der Server berechnet sie und sendet diese wieder zurück.

Das Programm wurde mit IntelliJ realisiert. Die JAR Files befinden sich in \out\artifacts

Zum Ablauf: Der Client liest die IP des Servers mittels Konsole ein und verbindet sich dorthin. Der Server gibt die Verbindung auf der Konsole aus. Der Client liest nun vom Benutzer die erste Zahl, das Berechnungszeichen und die zweite Zahl ein. Speichert diese in der Klasse *Message.* Und sendet die ganze Klasse dem Server. Dieser liest die Werte aus der Klasse aus, berechnet, gibt die Berechnung auf der Konsole aus und sendet das Ergebnis als Stream zurück. Der Client gibt dieses auf der Konsole aus. Der Client kann nun wieder von vorne Beginen. Mit dem Befehl „exit“ kann der Client das Programm beenden. Der Server kann eine Verbindung auf einmal haben, kann aber viele Verbindungen hintereinander abarbeiten. Eingabefehler können aufgefangen werden, bei anderen Fehlern, stürzt nur der Client ab, der Server läuft weiter.

Client Server

Objekt Message, mit allen Daten der Berechnung

Stream mit der Lösung

Client Server

1.Auf Verbindungen warten

1.Mit Server verbinden Exit

Antwort erhalten

Verbinden und auf Nachricht warten

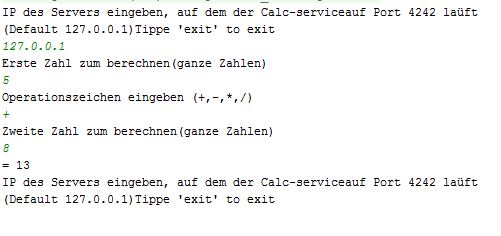
Und ausgeben Daten einlesen

Message senden

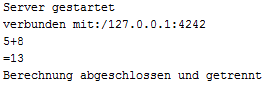
Berechnen und senden

Screenshots:

Client:



Server:



**Protokoll Aufgabe 2:**

Kurze Aufgabenstellung: Ein Server Client Programm erstellen, indem der Client sich beim Server Authentifiziert durch einen Benutzernamen um dann Berechnungen an den Server zu senden. Der Server berechnet sie und sendet diese wieder zurück. Falls er nicht Authentifiziert wird kann der Benutzer nichts berechnen lassen.

Das Programm wurde mit IntelliJ realisiert. Die JAR Files befinden sich in \out\artifacts

Zum Ablauf: Der Client liest die IP des Servers mittels Konsole ein und verbindet sich dorthin. Der Server gibt die Verbindung auf der Konsole aus. Der Benutzer wird aufgefordert einen Benutzernamen einzugeben, mit dem er sich beim Server Authentifizieren kann. Dies geschieht durch die Klasse *Auth* die übergeben wird. Falls er Authentifiziert wurde also vom Server eine Bestätigung erhält, liest das Client-Programm vom Benutzer die erste Zahl, das Berechnungszeichen und die zweite Zahl ein. Speichert diese in der Klasse *Message.* Und sendet die ganze Klasse dem Server. Dieser liest die Werte aus der Klasse aus, berechnet, gibt die Berechnung auf der Konsole aus und sendet das Ergebnis als Stream zurück. Der Client gibt dieses auf der Konsole aus. Der Client kann nun wieder von vorne Beginen mit der Authentifizierung. Mit dem Befehl „exit“ kann der Client das Programm beenden. Der Server kann eine Verbindung auf einmal haben, kann aber viele Verbindungen hintereinander abarbeiten. Eingabefehler können aufgefangen werden.

Client Server

Object Auth, mit den Authentifizierung Daten

Bescheit ob Authentifiziert oder nicht

Objekt Message, mit allen Daten der Berechnung

Stream mit der Lösung

Client Server

1.Auf Verbindungen warten

1.Mit Server verbinden Exit

Antwort erhalten

Verbinden und Benutzer Authentifizieren

Und ausgeben Benutzernamen einlesen

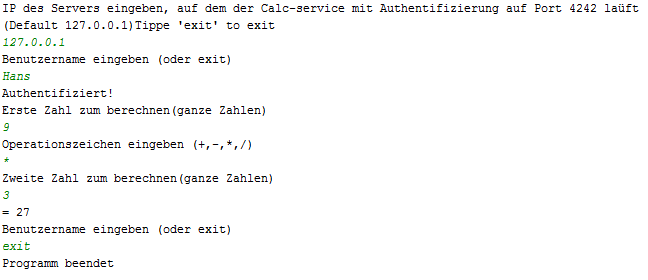
Berechnen und senden

Message senden Daten einlesen

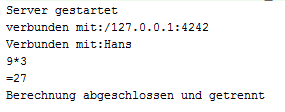
Auf Nachricht warten

Screenshots:

Client:



Server:



**Protokoll Aufgabe 3:**

Kurze Aufgabenstellung: Ein Server Client Programm erstellen, indem der Client dem Server eine URL einer Webseite sendet. Der Server lädt das HTML File herunter und sendet es dem Client. Der Client speichert diese ab.

Das Programm wurde mit IntelliJ realisiert. Die JAR Files befinden sich in \out\artifacts

Zum Ablauf: Der Client liest die IP des Servers mittels Konsole ein und verbindet sich dorthin. Der Server gibt die Verbindung auf der Konsole aus. Der Benutzer muss jetzt die URL der Webseite eingeben die er herunterladen möchte. Diese wird in die Klasse *Url* gespeichert und dann gesendet. Das http(s):// muss er auch angeben damit der Server weiß welches Protokoll er verwenden muss. Der Server lädt das HTML File lokal ab und sendet es per Stream an den Client, dieser speichert diese dann ab. Der Server kann eine Verbindung auf einmal haben, kann aber viele Verbindungen hintereinander abarbeiten. Eingabefehler können aufgefangen werden, bei anderen Fehlern, stürzt nur der Client ab, der Server läuft weiter.

Client Server

Url mit dem Link

Stream mit den Inhalt des HTML Files

Client Server

1.Auf Verbindungen warten

1.Mit Server verbinden Exit

Verbinden und URL herunterladen

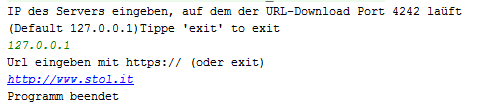
URL einlesen und senden

Daten speichern Stream empfangen

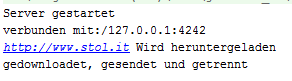
Daten per Stream senden

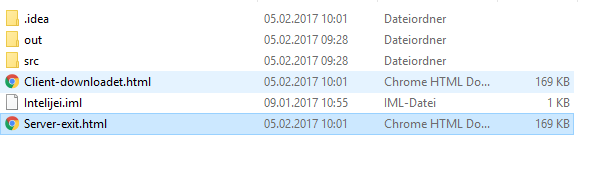
Screenshots:

Client:



Server:





**Protokoll Aufgabe 4:**

Kurze Aufgabenstellung: Ein Server Client Programm erstellen, indem zwei Clients über einen Server miteinander 4-Gewinnt spielen können.

# Dieses Programm wurde auf einen anderes selbst programmiertes Programm aufgebaut.

Das Programm wurde mit IntelliJ realisiert. Die JAR Files befinden sich in \out\artifacts

Zum Ablauf: Der Client kann sich mittels IP zu einem Server Verbinden, oder einen Server erstellen. Alles wird mittels Grafischer Oberfläche eingegeben. Der Client sendet dem Server, mittels der Klasse Nachricht, den Punkt an dem er seinen Stein einfüge will. Der Server speichert dies und sendet es dem anderen Client. Beide Clients haben *Listener* die auf solche Nachrichten Warten. Der Server nutzt zudem die Klasse *writer* um Nachrichten zu senden. Falls jeman gewinnt erkennen dies die Clients und geben dies aus.

#Das Programm ist nicht ganz fertig, weil ein Fehler alles aufhält. Die Spielregeln und die Gewinn-Erkennung müssen noch implementiert werden und auch der Fehler muss gefunden werden damit alles funktioniert.

Client Server

Objekt Nachricht, mit den Infos des getätigten Zuges

Nachricht mit dem Zug des Anderen

Falls jemand gewonnen hat mitteilen

Client Server

1.Auf 2 Verbindungen warten

1.Mit Server verbinden

Felder entsprechend

Auf Nachricht warten

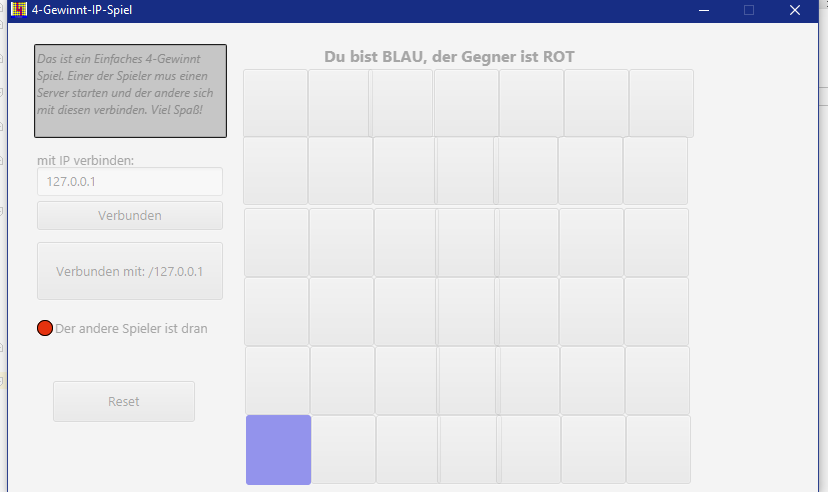
anzeigen Benutzer Eingabe warten

Nachricht erhalten Nachricht senden

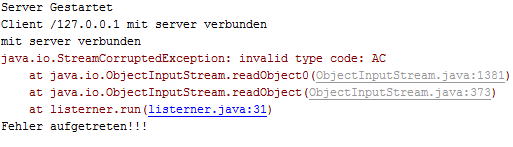
Nachricht an anderen senden

Screenshots:

Client:



Server:



# Der Fehler

**Protokoll Aufgabe 5:**

Kurze Aufgabenstellung: Ein Server Client Programm erstellen, indem der Client dem Server einen Pfad zu einer, auf dem Server liegenden, Datei angibt. Der Server sendet dies dem Client. Der Client speichert es ab. Mehrere Clients sollen parallel dieses dienst Nutzen können.

Das Programm wurde mit IntelliJ realisiert. Die JAR Files befinden sich in \out\artifacts

Zum Ablauf: Der Client liest die IP des Servers mittels Konsole ein und verbindet sich dorthin. Der Server gibt die Verbindung auf der Konsole aus. Der Benutzer muss jetzt den Pfad der Datei eingeben die er herunterladen möchte. Diese wird in die Klasse *Message* gespeichert und dann gesendet. Der Server öffnet das File und sendet den Inhalt per Stream an den Client, dieser speichert dies dann ab. Der Server kann mehrere Verbindungen auf einmal haben, dies wurde mit einem Executor realisiert. Eingabefehler können aufgefangen werden, bei anderen Fehlern, stürzt nur der Client ab, der Server läuft weiter.

Client Server

Pfad zur Datei auf dem Server

Stream mit den Inhalt des Files

Client Server

1.Auf Verbindungen warten

1.Mit Server verbinden Exit

Datei öffnen

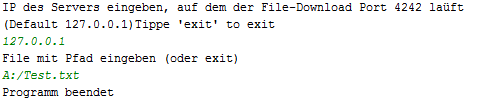
Pfad einlesen und senden

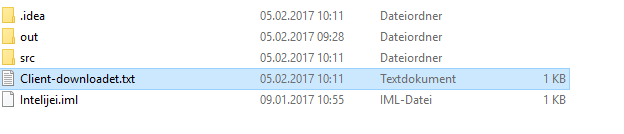
Daten speichern Stream empfangen

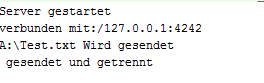
Inhalt per Stream senden

Screenshots:

Client:



Server:



**Protokoll Aufgabe 6:**

Kurze Aufgabenstellung: Ein Server Client Programm erstellen, indem der Client dem Server eine Anfrage zu Aktuelle Zeit oder ein zufälliger Wert sendet. Der Server soll dafür länger brauchen und es zurücksenden. Eine Version soll mit einem Thread und eine mit mehreren realisiert werden.

Das Programm wurde mit IntelliJ realisiert. Die JAR Files befinden sich in \out\artifacts

Zum Ablauf: Der Client liest die IP des Servers mittels Konsole ein und verbindet sich dorthin. Der Server gibt die Verbindung auf der Konsole aus. Der Benutzer muss jetzt 0 oder 1 eingeben, zufälliger Wert oder Aktuelle Zeit. Die Nachricht wird mit der Klasse Message übertragen, und mit einer gleichen Klasse wieder zurück gesendet. Eine Version arbeitet immer einen Client nach einander ab, die andere kann mehrere gleichzeitig bedienen. Mit Thread.sleep() soll eine längere Berechnung simuliert werden. Beobachtung: Die Clients bekommen die Antwort ca. gleich schnell, nur es kann sich nur ein einziger Client zu der Version mit einem Thread verbinden.

Client Server

Klasse Message mit 0 oder 1

Klasse Message mit zufälligem Wert oder Aktueller Zeit

Client Server

1.Auf Verbindungen warten

1.Mit Server verbinden Exit

Wert berechnen

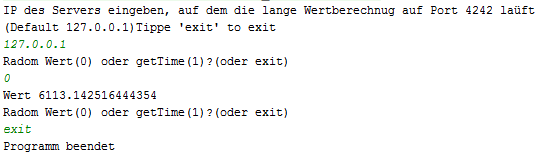
Wert einlesen und senden

Wert empfangen

Wert senden

Screenshots:

Client:



Server:



**Protokoll Aufgabe 7:**

Kurze Aufgabenstellung: Ein Server Client Programm erstellen, indem der Client dem Server eine Anfrage zu Aktuelle Zeit oder ein zufälliger Wert sendet. Der Server soll dafür länger brauchen und es zurücksenden. Der Server und der Client sollen zudem alle Fehler abfangen, und der Server soll einen sauberen Shutdown durchführen können.

Das Programm wurde mit IntelliJ realisiert. Die JAR Files befinden sich in \out\artifacts

Zum Ablauf: Der Client liest die IP des Servers mittels Konsole ein und verbindet sich dorthin. Der Server gibt die Verbindung auf der Konsole aus. Der Benutzer muss jetzt 0 oder 1 eingeben, zufälliger Wert oder Aktuelle Zeit. Die Nachricht wird mit der Klasse Message übertragen, und mit einer gleichen Klasse wieder zurück gesendet. Mit Thread.sleep() soll eine längere Berechnung simuliert werden. Der Client kann zudem den Server auffordern sich sicher abzuschalten, alle anderen Threads werden sicher abgeschaltet mit dem ExecutorService. .

Client Server

Klasse Message mit 0 oder 1

Klasse Message mit zufälligem Wert oder Aktueller Zeit

Client Server

Sicherer Shutdown

1.Mit Server verbinden Exit

1.Auf Verbindungen warten

Wert berechnen

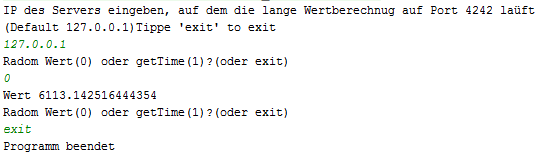
Wert einlesen und senden

Wert empfangen

Wert senden

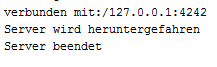
Screenshots:

Client:



Server:





**Protokoll Aufgabe 8:**

Kurze Aufgabenstellung: Ein Server Client Programm erstellen, indem mehrere Clients in einem Netz mit einem Chat sich austauschen können. Dabei wird UPD verwendet.

Das Programm wurde mit IntelliJ realisiert. Die JAR Files befinden sich in \out\artifacts

Zum Ablauf: Der Client liest die IP des Servers mittels Konsole ein und sendet alle Eingaben des Benutzers zu dieser Adresse. Der Server empfängt diese und sendet die Nachricht mittels Broadcast an alle anderen Clients die dieses Programm verwenden. Der Chat funktioniert nur wenn alle Clients und der Server im gleichen Netz hängen. Jeder Client hat einen Listener-Thread der alle erhaltenen Nachrichten vom Server auf der Konsole ausgibt. Da der Server aber an alle die Nachricht weiterleitet, erhält der Client der es gesendet hat auch die Nachricht und gibt sie aus.

Client Server

Nachricht als byte-Array

Nachricht als byte-Array, von alle anderen Nutzern

Client Server

1.Warten auf Eingabe

1. Server wartet

Erhält Nachricht von einem Client

Eingabe an Server senden

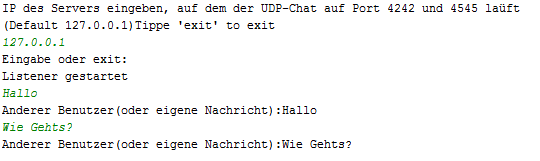
1.auf Nachricht warten

Nachricht Auf Konsole ausgeben

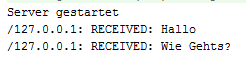
Sendet Nachricht an alle weiter

Screenshots:

Client:



Server:



**Protokoll Aufgabe 9:**

Kurze Aufgabenstellung: Ein Server Client Programm erstellen, indem der Client dem Server eine Ortschaft sendet und der Server antwortet mit der Jeweiligen Lawinengefahr. Hierbei wird C in Linux verwendet

Zum Ablauf: Der Client liest die IP des Servers mittels Konsole ein und verbindet sich dorthin. Der Client liest nun vom Benutzer die Ortschaft ein und sendet diese an den Server. Der Server berechnet die Gefahr(Zufällige Zahl in dieser Übung) und sendet es dem Client. Der Server gibt dabei auf diese Infos auf der Konsole aus.

Client Server

Buffer mit Ortsangabe

Buffer mit der jeweiligen Gefahr

Client Server

1.Auf Verbindungen warten

1.Mit Server verbinden

Antwort erhalten

Ortschaft erhalten und berechnen

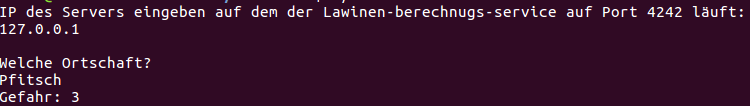
Und ausgeben Ortschaft einlesen

Buffer senden

Antwort mittels Buffer senden

Screenshots:

Client:



Server:

