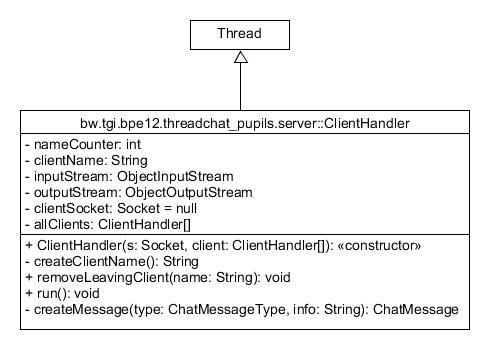
## Erweitern des Servers

Der Server hat die Aufgabe, die Nachrichten zu empfangen und an alle angeschlossenen, empfangsbereiten Clients weiterzuleiten. Gesteuert wird die Verteilung durch den sendenden Client, der auswählen kann, wer die Nachricht empfangen soll.

Im zur Verfügung gestellten Code ist die Server-Gui und der Empfänger wie in bisherigen Beispielen gegeben. Neu ist nun, dass wir mehrere Clients synchronisieren müssen. Dazu wird zunächst ein ClientHandler erzeugt (siehe Diagramm unten), von denen jedes Objekt einen Client ‚betreut‘.



Im Hauptprogramm des Servers werden alle Clients in einem Array gespeichert (alternative Lösung: Änderung der Speicherstruktur in eine Liste), sodass das Hauptprogramm weiter auf Empfang für neue Clients bleibt.

Die Klasse ClientHandler ist von der Klasse Thread abgeleitet, sodass der wesentliche Teil in der run()-Methode passieren muss. Der ClientHandler hält Verbindung zu ‚seinem‘ Client über den Socket und kommuniziert mit ihm über den input- und outputStream. Gleichzeitig ‚kennt‘ er aber auch die anderen Clients, da er die Referenz auf alle über den Konstruktor mitbekommt (Achtung: Da ist er auch selbst mit drin).

Was passiert nun in den einzelnen Methoden:

* In createClientName wird das Klassenattribut nameCounter hochgezählt und zusammen mit einem weiteren Namenselement (beliebig) ein eindeutiger Name erzeugt.
* Im Konstruktor werden wie üblich mithilfe der Parameter die entsprechenden Objektattribute initialisiert.
* Für die run-Methode liegt die nachfolgende Beschreibung vor:

Initialisiere den Input-und Outputstream mit Hilfe des sockets

Erzeuge den Namen des Clients

// Synchronisierung des Zugriffs auf die anderen ClientHandler

ChatMessage erzeugen für eine Verbindungsbestätigung

Die Message an den eigenen Client schicken

ChatMessage erzeugen, neuer Client hinzugefügt

An alle anderen Clients

Message neuer Client über outstream verschicken

Endlosschleife

ChatMessage empfangen

Wenn der Client sich abmeldet

**break**;

// Synchronisierung des Zugriffs auf die anderen ClientHandler

Für alle vorgesehenen Empfänger

Die empfangene Nachricht an alle Empfänger verschicken

// Synchronisierung des Zugriffs auf die anderen ClientHandler

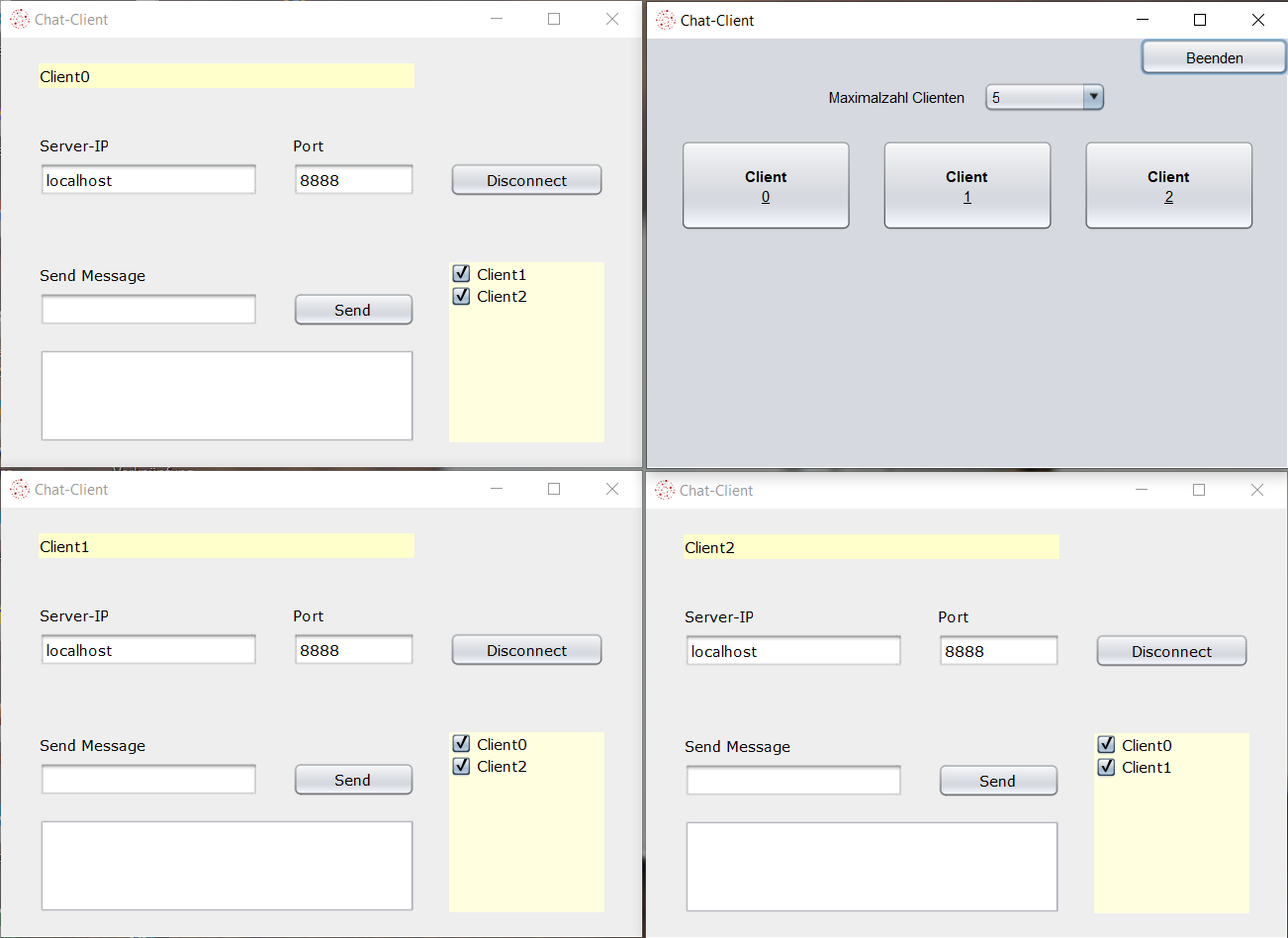
An alle anderen Clients

Abmeldung des Clients verschicken

// Synchronisierung des Zugriffs auf die anderen ClientHandler

Den eigenen Client aus der Liste löschen

inputStream, outputStream und Socket schließen

Wie man der Ablaufbeschreibung entnehmen kann, sind die Zugriffe auf die gemeinsame Liste (im Ursprung Arrays, aber auch als Liste möglich) jeweils zu synchronisieren, da ja jederzeit möglich ist, dass gerade ein anderer Client über seinen ClientConnector eine Nachricht verschickt.

Das Bild zeigt drei gestartete Clients und den Server. Jeder Client kennt die anderen und kann über die Checkboxen bestimmen, wer eine Nachricht erhalten soll.