# Leitfaden zum Chat-Server-Projekt

1. Zunächst brauchst du die zwei Klassen Chatserver und MeinClient.
2. Kümmern wir uns erst um den Client:  
   Seine Aufgabe ist es, Nachrichten an den Server zu schicken, Nachrichten vom Server zu empfangen und auszugeben.
   1. Deine Klasse muss von der ea-Klasse Client erben. Schreibe einen Konstruktor, in welchem du den super-Konstruktor mit IP-Adresse und Port aufrufst.
   2. Außerdem benötigst du die Methoden sendeNachrichtAnServer, die als Argument einen String annimmt. Rufe in dieser Methode die entsprechende geerbte Methode zum Senden eines Strings auf und übergib ihr als Argument den String.
   3. Überschreibe die Methode empfangeString. Schreibe die empfangene Nachricht zunächst auf die Konsole (system.out).
3. Die Aufgabe des Servers ist es grundsätzlich, die Nachricht eines Clients an alle Clients zurückzuschicken.
   1. Die Klasse muss von der ea-Klasse Server erben. Im Konstruktor rufst du wie beim Client den super-Konstruktor auf und übergibst ihm eine Port-Nummer (denke daran, es muss die gleiche wie beim Client sein!).
   2. Da du von der Klasse ea.Server erbst, musst du folgende Methoden implementieren:
      1. public void empfangeString(String s)
      2. public void empfangeInt(int i)
      3. public void empfangeByte(byte b)
      4. public void empfangeDouble(double d)
      5. public void empfangeChar(char c)
      6. public void empfangeBoolean(boolean b)
      7. public void verbindungBeendet()
   3. Der Server muss grundsätzlich nur alle empfangenen Strings an alle Clients senden („broadcasten“). Dazu gibt es in der ea.Server-Klasse die Methode sendeString.
   4. Optional kannst du in der Methode verbindungBeendet(), die aufgerufen wird, falls ein Client die Verbindung schließt, eine entsprechende Nachricht an alle Clients senden, um sie darüber zu informieren.
4. Du kannst jetzt in BlueJ einen Server erstellen und anschließend(!) zwei oder mehr Clients.  
   Rufe die Methoden sendeNachrichtAnServer() von den Clients auf und überprüfe, ob die Nachrichten auf der Konsole ausgegeben werden. Warum erscheinen sie mehrmals?

Extra-Aufgaben:

1. Jeder Client soll vor die Nachricht (s)einen Namen schreiben.
2. Du kannst mit der Engine Alpha leicht eine Ausgabe für den Text bauen:
   1. Erstelle eine Klasse Ausgabe und lasse sie von ea.Game erben. Rufe im Konstruktor den super-Konstruktor auf:  
      super(800, Ausgabe.FENSTERHOEHE, "Client-Server-Test", false, false);  
      Deklariere die public static final Variable FENSTERHOEHE und setze sie auf einen beliebigen Wert (empfehlenswert ist ca. 300).  
      Initialisiere im Konstruktor einen Server und einen oder mehr Clients.
   2. Du musst die Methode tasteReagieren() überschreiben. Sie wird bei jedem Tastendruck mit dem Tasten-Code aufgerufen.
   3. Führe eine Fallunterscheidung durch:
      1. Ist der Tastencode 32 (->ESC), soll das Fenster geschlossen werden:  
         Rufe der Reihe nach bei all deinen Clients und danach bei deinem Server die Methode beendeVerbindung() und schließlich super.beenden() auf.
      2. In den anderen Fällen soll eine Nachricht an den Server gesendet werden.  
         Um den Nutzer eine Nachricht eingeben zu lassen, stellt die Klasse ea.Game (die Superklasse) die Methode eingabeFordern(String aufforderung) zur Verfügung. Sie öffnet ein Eingabe-Fenster und gibt anschließend die Eingabe zurück.
   4. Passe jetzt die Klasse MeinClient an.
      1. Es wird wieder eine public static final Variable benötigt: ZEILENHOEHE (die in einem Schritt deklariert und mit einem Wert von ca. 30 initialisiert werden soll)
      2. Füge außerdem im Konstruktor ein Argument Game game hinzu und speichere es in einer globalen Variablen ab.
      3. In der Methode empfangeString soll der Text jetzt nicht mehr auf der Konsole, sondern im Ausgabe-Fenster angezeigt werden, aber nur, solange dem Konstruktor ein Game-Objekt und nicht null übergeben wurde.
      4. Übergib der Methode wurzel.add (vom Game-Objekt aus dem Konstruktor) ein ea.Text-Objekt. Schau zwecks der Parameter in die Dokumentation.
      5. Denke daran, dass, um die Nachrichten untereinander anzuzeigen, du die empfangenen Nachrichten mitzählen musst und die Höhe des nächsten Textes dementsprechend anpassen (Tipp: Zeilenhöhe\*Anzahl der Nachrichten)
      6. Überprüfe, ob die Zeilen außerhalb der Bildschirmhöhe liegt und verschiebe den Bildausschnitt entsprechend:  
         game.cam.verschieben(0, zeilenhöhe);
   5. Passe den Aufruf der Konstruktoren für die Clients in der Anzeige-Klasse an und übergib beim ersten Client this (als Argument für den Game-Parameter) und bei den weiteren null.