Leitfaden zum Chat-Server-Projekt

Du brauchst zwei Klassen: `Chatserver` und `MeinClient`.

1. **MeinClient:**
   a. Denk dran, die Engine Alpha mit import ea*.; am Anfang des Quellcodes zu importieren.
   b. Deine Klasse muss von der ea-Klasse `Client` erben. Schreibe einen Konstruktor, in welchem du den super-Konstruktor aufrufst. Als Übergabeparameter brauchst du einen `String ipAdresse` und einen `int Port`.
   c. Zum Nachrichten schreiben brauchst du eine Methode, in der du einfach die Methode `sendeString(String s)` von der Superklasse aufrufst und ihr deine Nachricht übergibst.
   d. Überschreibe die Methode `empfangeString(String s)` und lass sie den übergebenen String einfach am Bildschirm ausgeben.
2. **Chatserver**
   a. Analog zu a. und b. von MeinClient. Der Konstruktor von Server braucht allerdings nur einen `int Port` als Übergabeparameter.
   b. Da du von der Klasse ea.Server erbst, musst du folgende Methoden implementieren, kannst sie aber einfach leer lassen, wenn du sie nicht verwendest:

```
public void empfangeString(String s)
public void empfangeInt(int i)
public void empfangeByte(byte b)
public void empfangeDouble(double d)
public void empfangeChar(char c)
public void empfangeBoolean(boolean b)
public void verbindungBeendet()
```

   c. Optional kannst du in der Methode `verbindungBeendet()` eine entsprechende Nachricht an alle Clients senden, um sie entsprechend darüber zu informieren.
   d. Der Server muss grundsätzlich nur alle empfangenen Strings an alle Clients senden („broadcasten“). Dazu gibt es in der ea.Server-Klasse die Methode `sendeString`.
3. **Ausprobieren**
   Du kannst jetzt in BlueJ einen Server erstellen und anschließend(!) zwei oder mehr Clients.
   Der Port muss beim Server und allen Clients der gleiche sein.
   Rufe die Methoden `sendeNachrichtAnServer()` von den Clients auf und überprüfe, ob die Nachrichten auf der Konsole ausgegeben werden. Da jeder Client die Nachricht am Bildschirm ausgibt bekommst du so viele Bildschirmausgaben pro Nachricht, wie du Clients hast.

Zu den Extra-Aufgaben:

1. Jeder Client soll vor die Nachricht (s)einen Namen schreiben.
   a. Dem Client muss im Konstruktor ein `String name` mitgegeben werden, welcher anschließend vor jede Nachricht gehängt wird.
2. Bildschirmausgabe:
   a. 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.

b. Du musst die Methode `tasteReagieren()` überschreiben. Sie wird bei jedem Tastendruck mit dem Tasten-Code aufgerufen.
c. Führe eine Fallunterscheidung durch:
    i. 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.
    ii. 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.
d. Passe jetzt die Klasse `MeinClient` an.
    i. 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)
    ii. Füge außerdem im Konstruktor ein Argument `Game game` hinzu und speichere es in einer globalen Variablen ab.
    iii. 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.
    iv. Übergib der Methode `wurzel.add` (vom Game-Objekt aus dem Konstruktor) ein ea.Text-Objekt. Schau zwecks der Parameter in die Dokumentation.
    v. 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)
    vi. Überprüfe, ob die Zeilen außerhalb der Bildschirmhöhe liegt und verschiebe den Bildausschnitt entsprechend:
    `game.cam.verschieben(0, zeilenhöhe);`
e. 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`.