Chat client-server - documentazione

Chat client-server 2.0

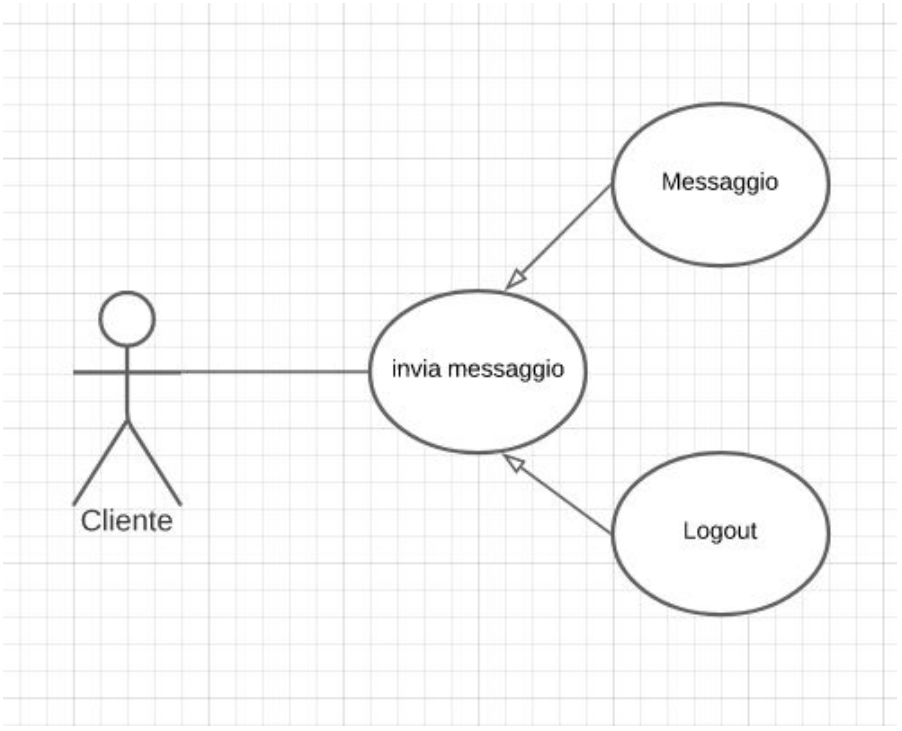
La chat tra client e server utilizza viene gestita tramite i socket. Il socket è formato dalla coppia IP e numero di porta, quando il client deve inviare un messaggio al server, questo passa dal socket poi alla pila TCP/IP dove successivamente verrà ricevuto dal server tramite il processo inverso fatto dal client. Un socket, perciò, permette di comunicare attraverso la rete utilizzando la pila TCP/IP, inoltre è progettato per utilizzare le API che mettono a disposizione del programmatore gli strumenti necessari a utilizzare il protocollo di comunicazione e a codificare la connessione.
In questa chat, la situazione è più semplice dato che server e client girano sullo stesso computer, ma il procedimento è lo stesso.

Obiettivo: Gestione della disconnessione del client

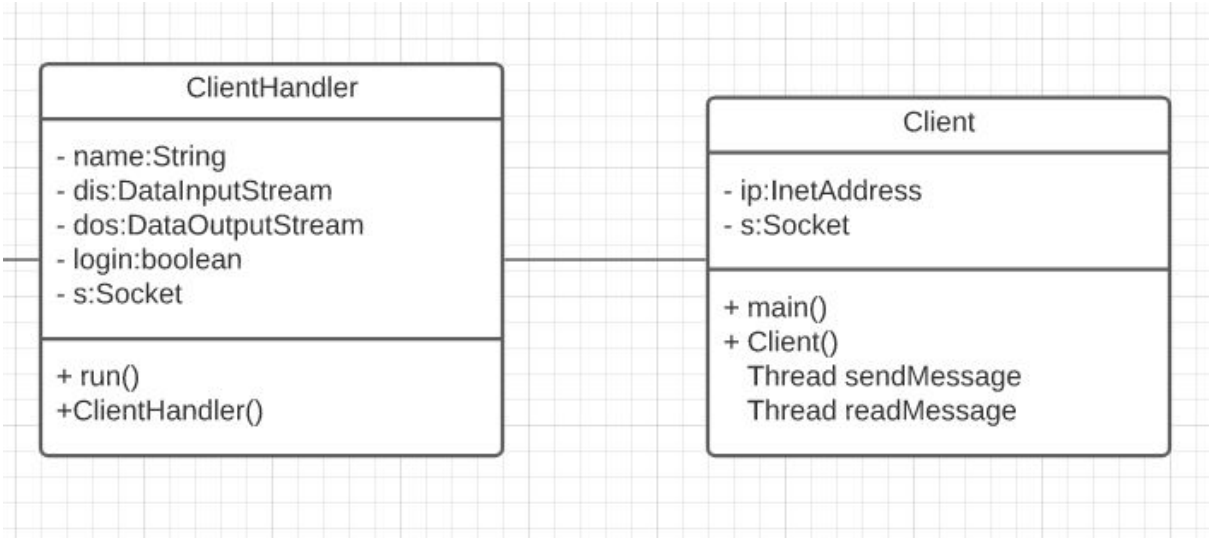
Descrizione della soluzione adoperata: Per eseguire l'uscita manuale dalla chat, l'utente deve inserire la stringa "Logout", subito dopo viene inviato un messaggio al server in cui viene detto che il client è uscito dalla chat e lo rimuove dal vettore.
Se un altro client prova a inviare il messaggio al client fuori dalla chat, il server restituirà una stringa in cui spiega che quel determinato destinatario non è più presente nella chat e non è raggiungibile.
Per poter gestire il logout utilizzo un controllo con un if in cui verifico che, se la stringa scritta dall'utente è uguale a Logout, allora imposta la variabile login a falsa, in modo tale da rimuoverlo anche nel vettore, e chiude il socket s.

```
while (true){
        try{
           ricevuta = dis.readUTF();
           System.out.println(ricevuta);
           if(ricevuta.equals("Logout")){
              this.login=false;
              this.s.close();
              break;
           }
```

Casi d'uso:



Classi utilizzate: Il controllo riferito al comando del logout è scritto all'interno della classe ClientHandler dato che questa gestisce il vettore **lc** contenente ogni client connesso alla chat, perciò è proprio questa classe che deve controllare chi si disconnette.



Messaggi tra client e server: Il client che vuole uscire dalla chat deve, innanzitutto essere connesso alla chat, poi scrivere il messaggio "Logout" (senza virgolette) e inviare il messaggio al server. Una volta che il server ha ricevuto il messaggio, questo invia una

stringa nel suo output in cui dice quale client si è disconnesso dalla chat. Infine, al client viene bloccata la possibilità di scrivere altri messaggi proprio perché è uscito dalla chat.

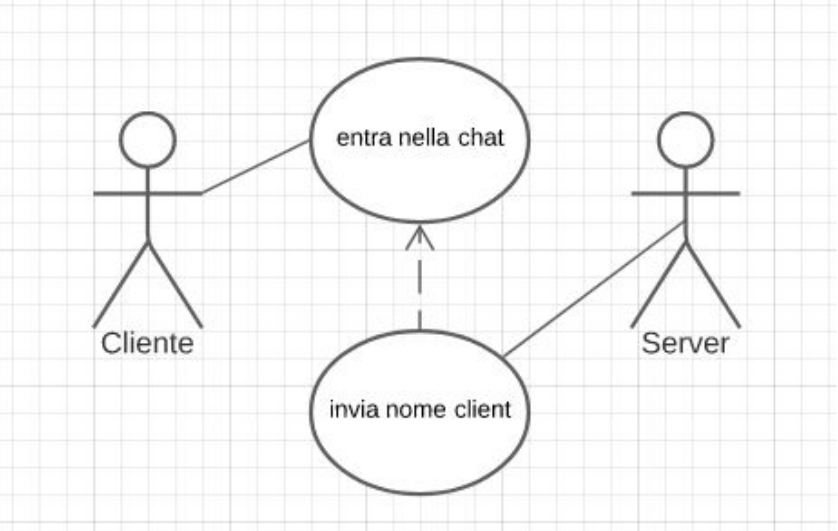
Messaggio	**Mittente**	**Utilizzo**	**Scrittura messaggio**
Disconnessione	Client	invio messaggio per disconnessione	Logout

Obiettivo: I client devono conoscere i nomi dei client connessi

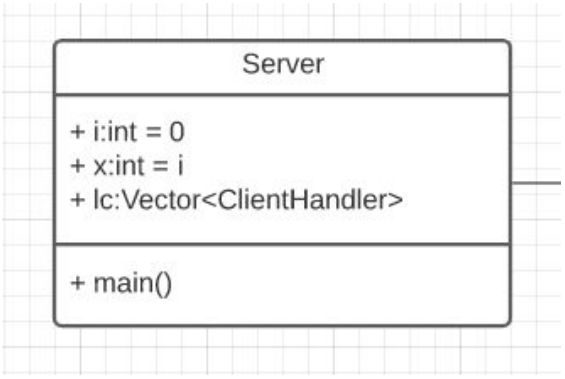
Descrizione della soluzione adoperata: Per inviare il nome sia del client stesso che degli altri creo un controllo sul numero del client stesso ovvero la variabile i.
Quando il client entra nella chat viene eseguito un controllo sulla variabile i, se questa è uguale a 0 significa che il client è il primo è essere entrato nella chat e perciò, se questo invia un messaggio gli verrà restituita una stringa in cui gli viene detto il nome e il fatto che è il primo ad essere entrato nella chat.
Quando entra un secondo client, viene controllata la variabile i come nel primo ma i sarà aumentata e perciò passa al controllo successivo in cui gli viene mostrato il suo nome.
Subito dopo si trova un while che mostra tutte le i precedenti finché questa non arriva a 0
Aggiungo prima del while una variabile x che pongo uguale ad i in modo tale da usare x solo per diminuirla e mostrare al client tutti quelli già connessi nella chat, mentre così i rimane con lo stesso valore senza modificarla. Dopo ogni fine del ciclo while la x viene posta nuovamente uguale ad i, facendo così non si intacca il contatore i e non si verificano errori con il numero del client.
Per far mostrare al client i vari messaggi del server riguardante chi è connesso nella chat, utilizzo dos.writeBytes() che consente di inviare la stringa del messaggio al client. Nella classe client creo un metodo per poter mostrare all'utente il messaggio con un System.out.println().

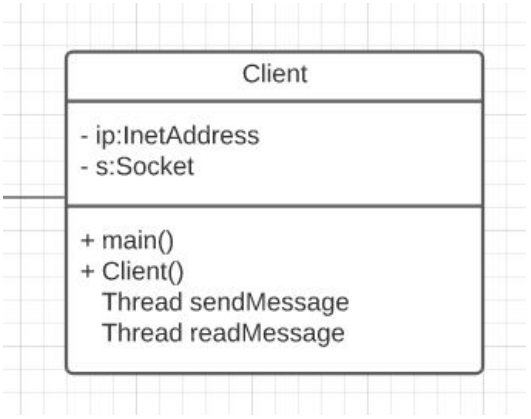
```
System.out.println("Client " + i + " è entrato nella chat");
        if(i == 0){
           dos.writeBytes("Sei il Client 0 e sei il primo nella chat" + '\n');
        }
        else if(i > 0){
           dos.writeBytes("Sei il Client " + i + '\n');
           int x = i;
           do {
             dos.writeBytes("Nella chat è presente: Client " + x-- + '\n');
           }while(x != 0);
        }
{
… //codice
}
i++;
```

Casi d'uso:



Classi utilizzate: Per l'invio del messaggio al client, il controllo avviene nella classe server in cui viene eseguito un controllo ogni volta che un client entra nella chat. Una volta che questo è entrato nella chat, gli viene inviato da parte del server un messaggio e ricevuto dalla classe client dal metodo "str" che, una volta ricevuta la stringa dal server, la mostra allo stesso client mediante un system.out.println.





Messaggi tra client e server:

Evento	**Mittente**	**Conseguenza**	**Messaggio inviato dal server**
Entrata nella chat per primo	Client	Server invia messaggio con nome del client	Sei il Client 0 e sei il primo nella chat
Entrata nella chat dopo il primo	Client	Server invia messaggio con nome del client e client precedenti	Sei il Client " + i

Obiettivo: Possibilità di avviare chat one to one oppure one to all

Descrizione della soluzione adoperata: Il codice server per inviare il messaggio, contenuto nella variabile chiamata ricevuta, al client destinatario posto dopo # che viene inserito nella variabile recipient.

```
StringTokenizer st = new StringTokenizer(ricevuta, "#");
String MsgToSend = st.nextToken();
String recipient = st.nextToken();
```

Dopo che il messaggio viene inviato viene fatto un controllo nel vettore di client e se trova il destinatario online nella chat allora invia normalmente il messaggio, se il destinatario non è più online, mostra una stringa in cui afferma che il destinatario è uscito dalla chat, infine se il numero dei client presenti nel vettore è minore di 1, invia una stringa che dice che il client mittente è l'unico presente nella chat.

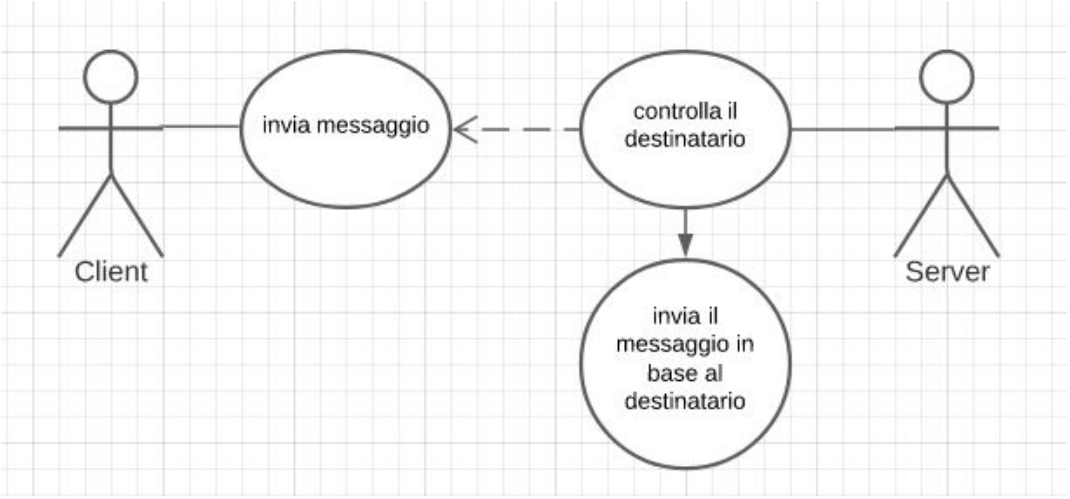
```
for (ClientHandler mc : Server.lc){
        if (mc.name.equals(recipient) && mc.login==true){
          mc.dos.writeUTF(this.name+ " : " +MsgToSend);
          break;
        } else if(mc.login==false){
          System.out.println("Il client destinatario è uscito dalla
chat");
        } else if(Server.i <= 1){
          System.out.println("Sei l'unico partecipante alla chat");
        }
      }
```

Per l'invio del messaggio a tutti i client, viene utilizzata solo una variabile StringTokenizer denominata st, e dopo un controllo sullo stato di login dei client all'interno del vettore lc, se i client sono online il messaggio viene inviato a tutti.

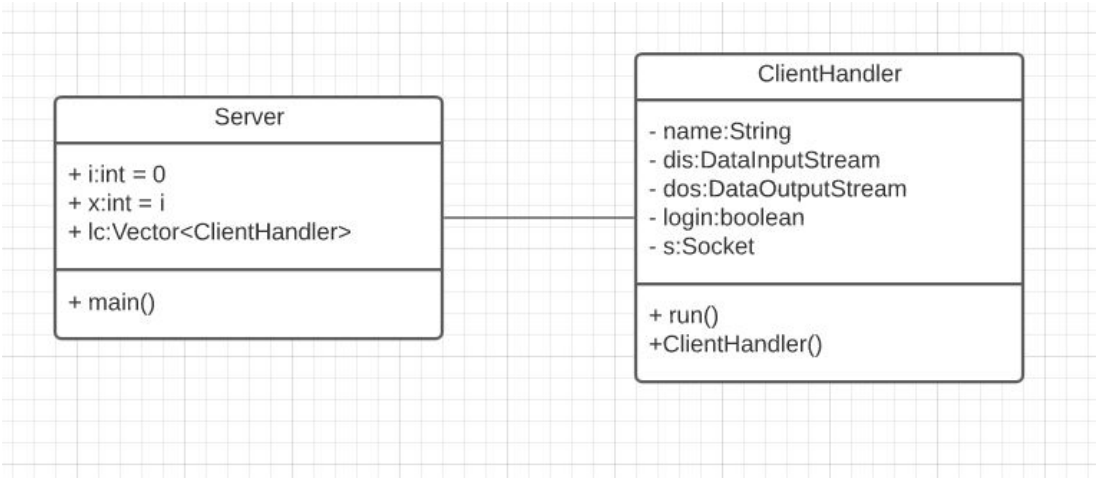
```
StringTokenizer st;

for (ClientHandler mc : Server.lc){
        if (mc.login==true){
          mc.dos.writeUTF(ricevuta);
          System.out.println("Messaggio: " + ricevuta + "\n");
        }
      }
```

N.B. Le funzioni di chat one-to-one e one-to-all sono separate in due diversi file dato che unirli in un unico file provocava errori che non permettevano di ricevere i messaggi.

Casi d'uso:



Classi utilizzate: L'invio del messaggio e i controlli su quest'ultimo, vengono tutti gestiti dalla classe ClientHandler dato che lo scopo di questa classe è proprio quello di gestire tutti i client all'interno del vettore lc e, di conseguenza, tutti i messaggi inviati da parte di ogni client.



Messaggi tra client e server:

Messaggio	**Mittente**	**Utilizzo**	**Scrittura messaggio**
Messaggio 1 (in chat one-to-one)	Client	invio messaggio per un destinatario	messaggio#Client(numero destinatario)
Messaggio 2 (in chat one-to-all)	Client	invio messaggio per tutti	messaggio

Chat client-server 1.0

Passaggi per la connessione:
Creazione del server
Creazione del client che si dovrà collegare al server
Dopo aver instaurato la connessione il client invia il nome al server
Il client invia il messaggio al server, questo lo invia all'altro client
Viceversa il secondo client deve inviarlo al primo
per uscire dalla chat scrivere "Logout"

Classe Server: è usata per creare il server, ricevere e accettare le varie richieste di comunicazione da parte dei client

Classe ClientHandler: viene usata per inserire i nomi dei client all'interno del vettore per usarli in seguito nella ricerca del destinatario.
Qui utilizzo Stringtokenizer per separare il messaggio dal destinatario tramite il formato: messaggio#destinatario

Classe Client: La classe contiene 2 thread, uno per l'invio del messaggio, l'altro per la lettura del messaggio.
WriteUTF e ReadUTF utilizzano la codifica UTF-8 ovvero una codifica usata per rappresentare i caratteri Unicode come **#** o **:**
Il **#** viene usato come separatore tra il messaggio e il destinatario.