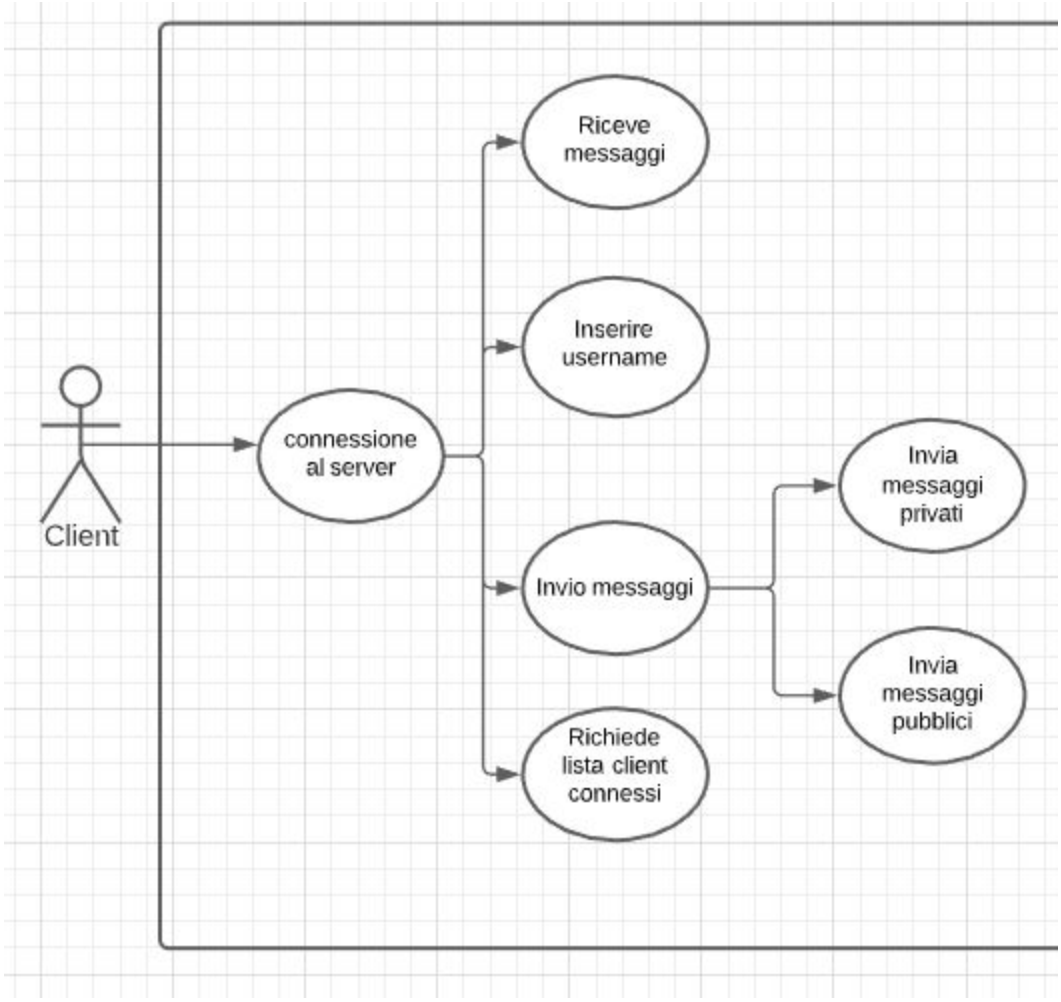
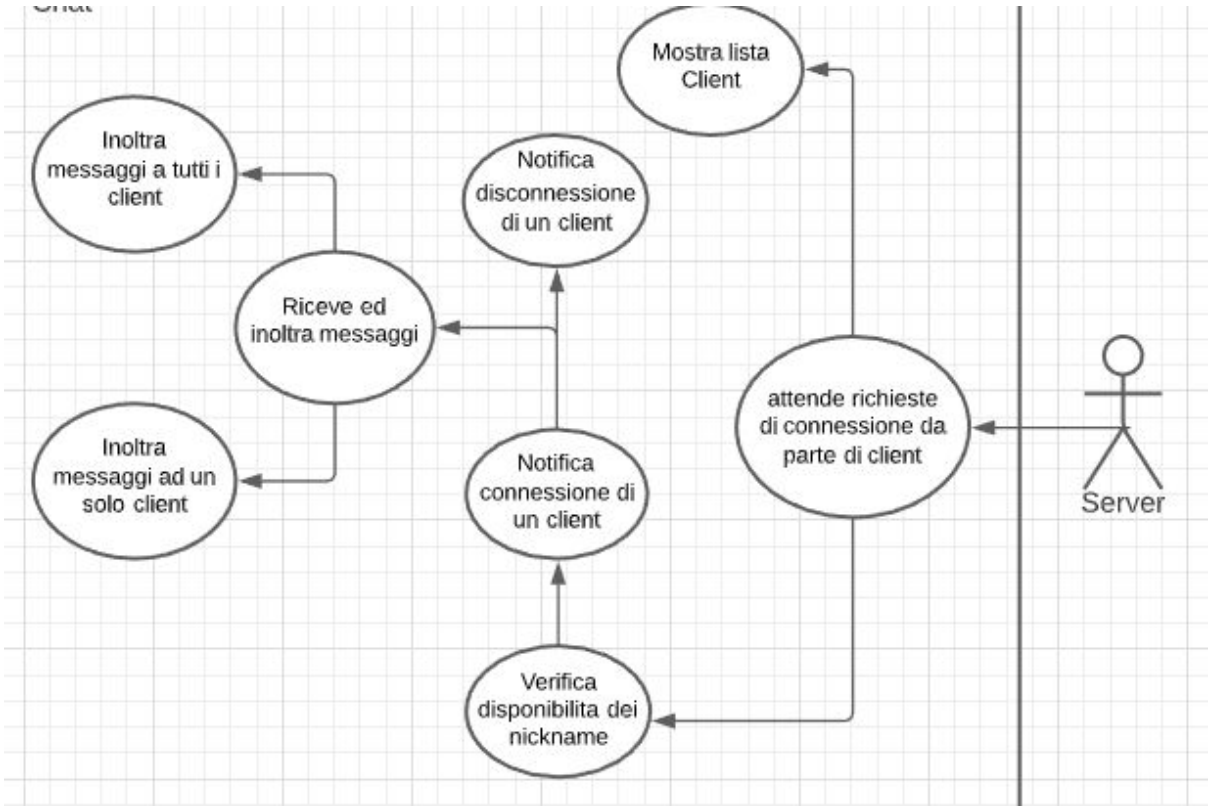
# Progettazione Chat

**-Casi d'uso Client e Server**




-Descrizione Client:
Nella soluzione proposta il Client effettua una richiesta di connessione al Server, che verrà accettata se ci sono posti disponibili, altrimenti la connessione non avverrà.
Successivamente il Client dovrà inserire un Username UNIVOCO con cui farsi riconoscere tra tutti gli altri client.
Il Client può inviare messaggi in BROADCAST, ovvero a tutti gli altri client, oppure ad un client specifico tramite comando.
Il Client può anche richiedere la lista di tutti i Client connessi alla chat ed è dotato di un thread che permette la continua ricezione di messaggi dal canale di comunicazione.

**-Casi d'uso Server**




-Descrizione Server:
Il numero massimo di client viene richiesto in input dopo l'avvio del server.
Il server si mette in attesa di eventuali richieste da possibili client.
Quando ciò avviene notifica a tutti gli altri client connessi l'avvenuta connessione con il nuovo client, stessa notifica avviene alla disconnessione di quest'ultimi.
Tramite dei Thread si occupa di inoltrare i messaggi ai vari client.

## Connessione:

Il client instaura una connessione creando un socket connettendosi al server. Per riuscire a connettersi, deve conoscere l'indirizzo IP e la porta del server. Il client, una volta connesso, deve gestire i canali di input e di output. Il client, per gestire questi due canali contemporaneamente, deve utilizzare i Thread. Può capitare l'errore in cui il server non sia raggiungibile o che non sia in ascolto.

## Classe Client

La classe client utilizza 2 metodi che sono: main(String[],args) di tipo void e connessione() di tipo void. Il primo metodo serve per dichiarare un oggetto di tipo server e richiamare il metodo connessione(), che serve per connettersi al server.

## Classe Lettura

Il Thread lettura è implementato per permettere una continua ricezione dei messaggi dal server.
Nella classe lettura sono presenti 3 metodi. Il primo metodo è Lettura() il costruttore. L'override del metodo run() della classe Thread() permette la continua ricezione dei messaggi provenienti dal server, e poi c'è il metodo chiusura() che serve per chiudere il Thread che riceve e legge i messaggi.

## Classe Scrittura

Il Thread scrittura è implementato per permettere di inviare quanti messaggi vuole il client al server. Nella classe scrittura sono presenti 2 metodi. Il primo metodo è Scrittura(), che sarebbe il Costruttore. Poi c'è l'override del metodo run() della classe Thread dove gestisce i messaggi che vengono inviati dal client.

## Classe Server

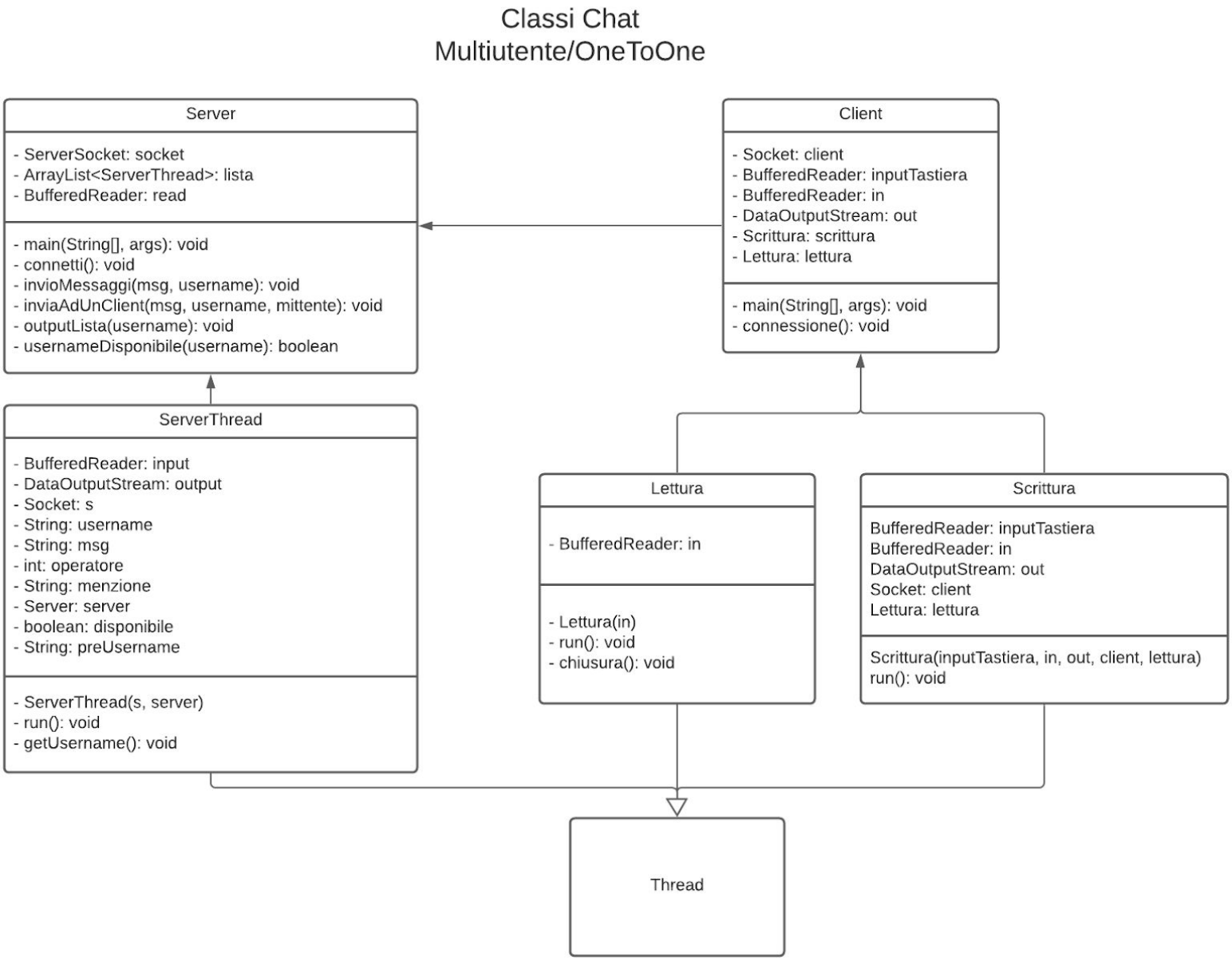
La classe Server è la classe che si occupa dell'instaurazione delle connessioni con i client e l'assegnazione dei Thread per ognuno di loro e la loro memorizzazione in un vettore.
Le richieste vengono accettate in un ciclo for(), all'interno del metodo connetti().
Altri metodi presenti all'interno della classe Server sono implementati per permettere l'inoltro di messaggi sia privati che rivolti a tutti i client, la visualizzazione dei client connessi e un per controllare l'username in ServerThread.

## Server Thread

La classe ServerThread estende thread e serve per gestire il data socket assegnato per lo scambio di dati con il client.
Viene creato un ServerThread per ogni client che si connette.

# *Messaggi scambiati tra client e server*

| **Messaggio** | **Mittente** | **Contenuto** | **Formato** |
|---|---|---|---|
| username | Client | nome del client | *username*lorenzo |
| messaggio_1 | Client | messaggio per tutti i client | *msg1*messaggio |
| messaggio_2 | Client | messaggio per un client | @nome_dest + *msg2* messaggio |
| disconnessione | Client | disconnessione del client | *stop* |




*Team:*

- *Cibecchini Lorenzo*
- *De Viti Lorenzo*
- *Fappani Niccolo'*
- *Camigliano Lorenzo*