

# A player connects to the server and joins the game: testo esplicativo

L'utente inserisce il proprio nome giocatore e la porta del server nell'interfaccia grafica. Cliccando conferma, queste informazioni vengono passate al `NetworkHandlerSender`. Questo cercherà di stabilire la connessione al server. Se fallisce, riporta il fallimento allo user e richiede l'inserimento delle informazioni.

Una volta stabilita la connessione possiamo seguire ciò che accade nel client o nel server. Iniziamo dal secondo: per ciascun client che si connette, il server crea due thread, uno bloccato sulla lettura dello stream associato al socket, uno bloccato nella lettura di una coda (`requestQueue`).

Nel lato client invece viene creato un thread separato, anche esso bloccato sulla lettura del socket.

A questo punto la connessione è stabilita e le classi responsabili per la comunicazione tra client e server sono inizializzate. Dunque, il client può mandare al server il proprio nome giocatore, codificato in un oggetto-evento (`NewPlayerEvent`) sotto forma di JSON. In questo diagramma la comunicazione via rete è rappresentata da una freccia marrone tra gli oggetti Socket del client e del server.

Il messaggio viene letto dal thread del server bloccato sulla lettura del socket, viene trasformato nel POJO corrispondente e inserito nella coda `requestQueue`. Poi questo thread torna a bloccarsi nella lettura del socket.

A questo punto il thread bloccato nella lettura della coda recupera l'evento appena inserito in essa. Si noti dunque come le tre classi `ClientHandlerReceiver`, `RequestQueue` e `RequestElaborator` implementino un tipico caso di Produttore Consumatore.

L'evento appena arrivato viene inviato all'event Registry che utilizza la classe di java `PropertyChangeSupport`, la quale implementa un pattern observer. A questo punto viene attivato l'handler registrato per l'evento ricevuto, il quale andrà ad effettuare le modifiche opportune al model. In questo caso specifico il giocatore viene aggiunto alla lobby che il server sta riempiendo e al client viene restituito un evento contenente lo stato della lobby. Con questa informazione il client potrà poi far visualizzare all'utente lo stato della lobby in cui si trova.

Mentre la creazione di messaggi nel server e nei client è analoga, la ricezione dei messaggi funziona in modo leggermente diverso. Il messaggio viene comunque letto dal thread bloccato sulla lettura del socket, tuttavia non viene inserito in una coda, bensì viene creato un secondo thread apposito per la gestione dell'evento. L'altro thread invece torna a bloccarsi nella lettura del socket.