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

L'applicazione (ClientApp) viene avviata. All'utente viene chiesto di inserire l'IP del server nell'interfaccia grafica. Cliccando conferma, questa informazione viene restituita alla ClientApp. Questa creerà un NetworkAdapter che cercherà di stabilire la connessione al server. Se fallisce, riporta il fallimento allo user e richiede nuovamente l'inserimento delle informazioni. Inoltre, il NetworkAdapter instanzierà anche gli oggetti responsabili per la lettura e scrittura del socket. Il primo, inoltre, crea un'istanza di EventRegister, classe responsabile di registrare per ogni evento una funzione che lo gestisca e di chiamarle al momento opportuno. Infine, NetworkAdapter registra per ogni evento una funzione che lo gestisca (per chiarezza, ne mostriamo solo una nel diagramma).

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 soket, uno bloccato nella lettura di una coda (requestQueue).

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

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, dopo averlo chiesto all'utente, 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 oggetto 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 soket.

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

L'evento appena arrivato viene inviato all'EventRegistry del server. A questo punto viene attivato l'handler registrato per l'evento ricevuto, il quale andrà ad effetuare le modifiche opportune al model. In questo caso specifico il giocatore viene aggiunto alla lobby che il server sta riempendo 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 immediatamente gestito. Poi il thread torna a bloccarsi sulla lettura del socket.