## Analisi Codice pag 111-119 Tepsit ES 3 Vacanze di Natale – Andreini

Il codice riguarda un progetto di creare una chat testuale attraverso le socket, gestito con il multithreading sia lato client che lato server.

Analisi delle due strutture:

### Il Client è gestito attraverso due thread distinti

- **ThreadInvio** che si occupa di leggere in input da tastiera ed inviarlo al server grazie ad un `PrintWriter.`
  Il primo messaggio che viene inserito e che viene letto è interpretato come nome utente per la chat, mentre i successivi sono dei semplici messaggi testuali, che possiamo inviare.
  L'utilizzo di `PrintWriter con flush` permette di garantire l'invio immediato dei messaggi.
- **ThreadRicevi** che riceve i messaggi che provengono da un `BufferReader.`
  Se `readLine` restituisce `null` significa che la connessione è stata chiusa dal server e quindi il Thread stampa un messaggio di avviso e poi si chiude la socket.

### Il Server è anch'esso gestito attraverso due thread distinti fra loro

- **ListaClient** che contiene un `ArrayList` di socket e offre tre principali funzioni: aggiungere, rimuovere un client e inviare un messaggio a tutti i client.
  I metodi sono sincronizzati per evitare problemi di concorrenza.
  La funzione `sendAll` invia un messaggio a tutti i client escluso ovviamente il client che lo ha inviato.
- **ThreadConnessione** che ogni volta che un client si collega, il server crea un'istanza di questa classe e questo Thread legge i messaggi del client: il primo viene usato come nome utente come già citato, mentre i successivi vengono inoltrati agli altri client tramite `ListaClient.`
  Se il client si disconnette, il thread lo rileva e stampa un messaggio sul server.

Per il server è importante anche **MainServer**: il server apre un `ServerSocket` sulla porta 5500 e rimane in ascolto. Ogni volta che arriva una nuova connessione, crea un nuovo thread **ThreadConnessione** e lo avvia.