**METODO 0 (usato attualmente):**

Su server

* Model (Match ecc)
* Controller
* RemoteView

La RemoteView quando riceve una update dal Model semplicemente invia il MatchView.toString(); quando deve richiedere una scelta invia la Choice al Client che la processa e la restituisce con i dati, la RemoteView controlla che i dati ricevuti siano corretti e in caso richiede di nuovo di inserire una scelta. (ultima parte da fare)

Su client:

* Gestione connessione
* Stampa e gestione input
* Gestione invio Choice processata

**METODO 1 (migliore per adesso):**

Su server

* Model (Match ecc)
* Controller
* RemoteView

Su client:

* Gestione connessione
* Stampa String ricevuta e Invio input letto da tastiera

In questo caso il Client è molto più leggero e semplicemente in loop aspetta un messaggio da stampare dalla connessione e immediatamente dopo chiede un input, il quale viene inviato alla RemoteView senza controlli (possibile controllo su stringa per lasciare la partita). La RemoteView quando deve richiedere una scelta all’utente, subito dopo aver ricevuto una richiesta dal controller attraverso il metodo specifico, semplicemente in base al tipo dinamico di Choice invia una Stringa al client, il quale la stampa e risponde con l’input, quindi la RemoteView fa il controllo sulla correttezza dell’imput e in caso procede facendo la notify() al controller.

**METODO 2:**

Su server

* Model (Match ecc)
* Controller

Su client:

* Gestione connessione
* View

In questo caso non ci basiamo su una RemoteView ma gli oggetti Choice delle richieste e la gestione delle update() richiamate dal Model (Match) vengono gestite direttamente sul Client, il quale scambierà gli oggetti necessari sia con il Controller che con il Match. Possibile caso di utilizzo dell’RMI per invocare i metodi update() in remoto attraverso le classi Observer e Observable. Caso peggiore per mantenere un Client leggero e facilmente adattabile a cambiamenti del model/toString/stampa su view ( tentativo di implementazione di un fit Client).