Commenti al testo e alle soluzioni proposte dai partecipanti all'esame

Esercizio sulla concorrenza

L'esercizio proposto è un caso di lettori e scrittori. Si poteva quindi riusare lo schema di soluzione visto a lezione per tale problema.

Riguardo alle soluzioni proposte, va osservato che moltissimi hanno proposto metodi di lettura e scrittura monolitici (in cui l'attività è svolta interamente nel metodo) e synchronized. In questo modo si ottiene che inevitabilmente ci sarà sempre al più un solo giocatore che accede al tavolo, infatti il primo che arriva mette un lock sul gestore del tavolo, escludendo quindi tutti gli altri fino a quando non avrà terminato l'operazione. Questo viola chiaramente la consegna, che prevede che i lettori possano accedere concorrentemente in numero qualunque al tavolo. Va anche segnalato che all'interno dei metodi di lettura e scrittura sinchronized è del tutto inutile mettere delle wait condizionate sul numero di lettori e scrittori, perché quando un thread inizia a eseguire un metodo (dopo aver acquisito il lock) il numero di lettori e scrittori sarà sempre zero (il numero viene incrementato e decrementato durante l'esecuzione del metodo, quindi sarà di nuovo zero quando si rilascia il lock).

<u>Sulla mancanza di deadlock</u>

Si chiedeva di spiegare perché non è necessario preoccuparsi della possibilità di deadlock. La spiegazione nel caso in questione è semplice: un lettore si sospende in attesa che uno scrittore termini di scrivere, e questo evento prima o poi si verifica sempre; uno scrittore si sospende in attesa che uno scrittore termini di scrivere o che i lettori terminino di leggere, e anche questo caso si verifica sempre. In linea teorica potrebbe esserci un problema di starvation, se i lettori continuano a occupare il tavolo, estromettendo chi desidera effettuare una mossa. Tuttavia, visto che i giocatori scelgono casualmente a ogni iterazione se leggere o scrivere, esistono probabilità non trascurabili che a un certo punto non ci sia alcun giocatore che vuole leggere lo stato del tavolo, lasciando così la possibilità di effettuare una mossa.

Esercizi di programmazione distribuita

Su questi c'è poco da dire: gli esercizi proposti sono dei "classici" dato che richiedono di rendere distribuito un programma dato.

Sull'esercizio riguardante l'uso di RMI, nel quale era richiesto di comunicare la situazione ai giocatori ogni volta che viene fatta una mossa, occorre ricordare che (come visto in molti altri esempi):

1) Il client (giocatore) deve essere un oggetto remoto.

2) Il client deve comunicare il proprio riferimento remoto al server, in modo che quest'ultimo lo possa usare per comunicare ai client le conseguenze delle mosse.
3) Pertanto, il server deve avere nella propria interfaccia un metodo che consenta ai client di fornire il proprio riferimento remoto. Tipicamente il server usa una lista (o struttura dati analoga) per memorizzare i riferimenti ai clienti.
4) Il client deve fornire nella propria interfaccia un metodo che il server possa usare per comunicare le conseguenze delle mosse.

Quanto descritto sopra è il classico pattern observer, ampiamente visto a lezione e ad esercitazione.