Esercizio 1. Concorrenza

Si consideri un gioco, nel quale i giocatori accedono al tavolo da gioco per leggere la situazione o per fare una mossa (la mossa consiste nel togliere una pedina da una posizione e sistemarla in un'altra). Non ci sono turni: i giocatori possono fare la loro mossa quando vogliono (infatti, nel codice dato ciascun giocatore decide autonomamente se leggere la situazione o se fare una mossa).

Si richiede di modificare il codice dato per fare in modo che

- Quando un giocatore sta facendo la propria mossa nessun altro giocatore possa accedere al tavolo, neanche per leggere (perché non avrebbe senso leggere una situazione che sta cambiando).
- Se nessun giocatore sta facendo mosse, la situazione possa essere letta da un numero qualunque di giocatori.

Si noti che i giocatori accedono al tavolo mediante un gestore, che nel codice dato non effettua alcun controllo sull'accesso al tavolo, ma si limita a rilevare le infrazioni alle regole date sopra. Occorre quindi modificare il gestore in modo che le regole indicate sopra vengano rispettate. NB: per far questo, è possibile introdurre nella classe Gestore nuovi metodi, dedicati al controllo dell'accesso al tavolo da parte dei thread.

È richiesto che non si verifichino corse critiche. NON è necessario preoccuparsi della possibilità di deadlock. Un bonus verrà riconosciuto a chi fornirà una spiegazione del motivo per cui non è necessario preoccuparsi del deadlock.

Si consegni uno zip contenente i file .java (NON i .class!).

Esercizio 2. Programmazione distribuita con socket

Si modifichi il codice realizzato come soluzione dell'esercizio di programmazione concorrente, in modo da realizzare un sistema distribuito in cui il server gestisce il gioco e i client si comportano da giocatori.

Si ricorda che bisogna caricare i file .java (NON i .class) in un unico file zip.

Esercizio 3. Programmazione distribuita con RMI

Si modifichi il codice realizzato come soluzione dell'esercizio di programmazione concorrente, in modo da realizzare un sistema distribuito in cui il server gestisce il gioco e i client si comportano da giocatori.

Il sistema deve avere le seguenti caratteristiche:

1) i client possono effettuare mosse come nell'esercizio di programmazione concorrente.
2) i client non effettuano operazioni di lettura: ogni volta che il server riceve una nuova mossa, informa tutti i client della nuova situazione.

Si realizzi il sistema usando RMI.

Si ricorda che bisogna caricare i file .java (NON i .class) in un unico file zip.