Esercizio 1. Concorrenza

In un gioco da tavolo i giocatori possono trovarsi in diversi stati (gli stati sono numerati 1, 2, 3, ...).

I giocatori effettuano iterativamente una delle azioni seguenti:

- cambiare stato,
- fare una progressione.

Il cambio di stato non è soggetto a condizioni e viene eseguito sempre immediatamente (richiede solo il tempo strettamente necessario per l'esecuzione). Viceversa, la progressione avviene solo quando ci sono un certo numero di giocatori nello stesso stato: se viene richiesta quando non ci sono abbastanza giocatori nello stesso stato, il giocatore richiedente viene messo in attesa. Un giocatore in attesa di progressione può uscire dall'attesa in due casi:

a) quando viene soddisfatta la condizione per la progressione (cioè quando abbastanza giocatori sono nello stato richiesto)

b) quando scade il tempo massimo di attesa.

Ad es. il giocatore X è l'unico nello stato 2 e prova a fare una progressione. Poiché le regole richiedono che ci siano due giocatori nello stato 2 per la progressione, il giocatore X viene messo in attesa. Ad un certo punto, il giocatore Y entra nello stato 2: di conseguenza, il giocatore X viene sbloccato e fa la sua progressione. Se invece nessun giocatore entra nello stato 2 entro il tempo massimo di attesa, il giocatore X viene sbloccato senza progressione.

Si richiede di modificare il codice dato per fare in modo da implementare il gioco descritto, senza che si verifichino i problemi tipici della programmazione concorrente (corse critiche, deadlock, ecc.).

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 realizzi il sistema usando socket.

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.

Si realizzi il sistema usando RMI.

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