Esercizio 1.

Si desidera realizzare un programma che simula un gioco di carte in cui, per ogni mano

- i giocatori fanno la loro mossa
- quando tutti i giocatori hanno giocato, il gestore del gioco conclude la mano e calcola il punteggio parziale
- i giocatori leggono il punteggio ottenuto nella mano appena conclusa

Nel codice dato il comportamento dei giocatori è simulato dal metodo exec della classe MainGioco in modo rigidamente sequenziale. Si modifichi il codice dato, in modo da ottenere un programma concorrente, in cui ciascun giocatore è simulato da un thread. In questo gioco non ci sono turni precisi, cioè il giocatore A può sempre giocare prima o dopo il giocatore B (si pensi ad esempio a un gioco in cui i giocatori mettono una carta coperta sul tavolo, e le carte si scoprono solo dopo che hanno giocato tutti). Nel codice modificato i giocatori devono poter giocare in un ordine qualunque.

Attenzione: occorre assicurarsi che 1) la lettura del punteggio avvenga dopo che tutti i giocatori hanno giocato, e 2) che una nuova mano cominci solo quando tutti i giocatori sono pronti, avendo letto l'esito della mano precedente.

Come sempre, occorre evitare i problemi tipici della programmazione concorrente (corse, critiche, deadlock, starvation, ecc.).

Si ricorda che bisogna caricare i file .java, NON i .class.

Esercizio 2.

Si consideri il codice dato, che simula attraverso thread alcuni giocatori che giocano al lotto (per semplicità i giocatori possono puntare su un solo numero, cioè non possono giocare ambi, terni, ecc.).

Il server opera ciclicamente come segue:

1. Apre le scommesse. Da questo momento le eventuali giocate da parte dei giocatori vengono accettate.
2. Aspetta per un po', permettendo così ai giocatori di fare le loro scommesse.
3. Effettua l'estrazione. Da questo momento eventuali giocate da parte dei giocatori vengono rifiutate.
4. Aspetta per un po'. Durante questo tempo eventuali richieste da parte dei giocatori di conoscere l'esito della scommessa ricevono risposta.

Un client esegue un ciclo in cui:

1) prova a piazzare una puntata presso il server. Se il tentativo non riesce aspetta un po' e riprova;
2) dopo aver fatto la puntata richiede l'esito. Questa è una operazione sospensiva, che termina quando l'esito diventa disponibile.

Si modifichi il codice, in modo da ottenere un programma distribuito, in cui ciascun giocatore si comporta da client, e il gestore della partita si comporta da server.

Si realizzi il sistema mediante socket.

Come sempre, occorre evitare i problemi tipici della programmazione concorrente e distribuita (corse, critiche, deadlock, starvation, ecc.).

Si ricorda che bisogna caricare i file .java, NON i .class.

Esercizio 3.

Si consideri il codice dato, che simula attraverso thread alcuni giocatori che giocano al lotto (per semplicità i giocatori possono puntare su un solo numero, cioè non possono giocare ambi, terni, ecc.).

Il server opera ciclicamente come segue:

1. Apre le scommesse. Da questo momento le eventuali giocate da parte dei giocatori vengono accettate.
2. Aspetta per un po', permettendo così ai giocatori di fare le loro scommesse.
3. Effettua l'estrazione. Da questo momento eventuali giocate da parte dei giocatori vengono rifiutate.
4. Aspetta per un po'. Durante questo tempo eventuali richieste da parte dei giocatori di conoscere l'esito della scommessa ricevono risposta.

Un client esegue un ciclo in cui:

1) prova a piazzare una puntata presso il server. Se il tentativo non riesce aspetta un po' e riprova;
2) dopo aver fatto la puntata richiede l'esito. Questa è una operazione sospensiva, che termina quando l'esito diventa disponibile.

Si modifichi il codice, in modo da ottenere un programma distribuito, in cui ciascun giocatore si comporta da client, e il gestore della partita si comporta da server.

Domanda 3.A.

Si realizzi il sistema mediante RMI.

Come sempre, occorre evitare i problemi tipici della programmazione concorrente e distribuita (corse, critiche, deadlock, starvation, ecc.).

Domanda 3.B.

Si modifichi il programma in modo che il server mandi ai client (giocatori) il numero estratto.

I client dopo aver effettuato una giocata possono fare altro in attesa che si apra la scommessa successiva. Ricevono notifica asincrona del numero estratto.

Si ricorda che bisogna caricare i file .java, NON i .class.