Programmazione Concorrente e Distribuita
Appello d'esame – 28 giugno 2017

Nome..... Cognome.....

Matricola.....

Non si possono consultare appunti e libri.

Esercizio 1

Si consideri il seguente gioco a due giocatori, basato su una scacchiera fatta di 8x8 caselle alternate bianche e nere.

- Nella prima fase, i due giocatori giocano in parallelo (cioè senza per forza alternare le proprie mosse) posizionando 30 pedine, una dopo l'altra, in una casella *casuale* della scacchiera. Ad ogni mossa, se la casella su cui il giocatore mette la pedina è bianca, allora può accumulare la sua pedina ad altre eventualmente già presenti in quella casella. Se invece la casella su cui il giocatore g vuole fare la sua mossa è nera, allora eventuali pedine già presenti in quella casella vengono "mangiate", cioè prelevate dalla scacchiera, e nella casella nera rimane solo l'ultima pedina aggiunta da g. Le pedine "mangiate" da un giocatore determineranno il suo punteggio finale.
- Quando entrambi i giocatori hanno completato le loro 30 mosse, la scacchiera inverte i colori delle caselle: mantenendo inalterato il numero di pedine su ogni casella, ogni casella bianca diventa nera e viceversa ogni casella nera diventa bianca. Dopodich/'e si ripete la fase precedente, cioè ogni giocatore fa altre 30 mosse secondo le stesse regole descritte nel punto precedente.
- Il punteggio finale di ogni giocatore è dato dal numero totale di pedine "mangiate" con le proprie 30+30 mosse.

Il programma indicato nella pagina seguente implementa il gioco descritto. Si chiede di completare il codice nel modo seguente:

- definire il metodo `int add()` della classe `Casella` che corrisponde all'aggiunta di una pedina nella casella `this` e restituisce il numero di pedine "mangiate" con questa mossa.
- definire il metodo `int add(int riga, int colonna)` della classe `Scacchiera` che corrisponde all'aggiunta di una pedina nella casella di posizione `riga` x `colonna` nella scacchiera.
- completare il metodo `run` della classe `Giocatore` in modo tale che il giocatore faccia altre 30 mosse solo dopo che la scacchiera ha cambiato colore.
- completare il codice del metodo `main` della classe `Gioco` in modo tale che avvi il gioco e visualizzi a video i punteggi finali dei due giocatori ed il numero totale di pedine rimaste alla fine sulla scacchiera.
- Il programma deve allo stesso tempo evitare errori di conocorrenza e sincronizzazioni inutili.
- È possibile aggiungere (pochi) campi **privati** e metodi (qualsiasi) alle classi date.

Esercizio 2

Considerando il programma completo scritto nel punto precedente, rispondere alle seguenti domande:

1. Quale thread cambia il colore della scacchiera?
2. Qual è l'ultimo thread che termina in questo programma? La risposta è la stessa per tutte le esecuzioni possibili?

Esercizio 3

Scrivere la versione distribuita del programma scritto nel punto precedente. In particolare:

- I due giocatori devono essere avviati su due host distinti e la scacchiera deve essere avviata su un terzo host.
- Indicare con precisione quale codice deve essere presente su ognuno dei tre host.
- Indicare su quali host compaiono le stampe prodotte dall'applicazione.
- Indicare quale thread cambia il colore della scacchiera.

```
class Casella {
    private boolean bianco;
    private int contenuto=0; //numero pedine su casella

    Casella(boolean b){bianco =b;}
    void swapColor(){ bianco= !bianco; }
    int size(){return contenuto;}

    .....add().....
}

class Scacchiera {
    public static final int NRIGHE = 8;
    public static final int NCOLONNE = 8;
    private Casella[] board = new Casella[NRIGHE*NCOLONNE];

    public Scacchiera() {
        boolean bianco = true;
        for (int i=0; i<NRIGHE*NCOLONNE; i++){
            board[i] = new Casella(bianco); bianco = !bianco;
        }
    }

    public void swap(){for(Casella c: board) c.swapColor(); }

    public int tot(){
        int t=0;
        for(Casella c:board){ t += c.size();}
        return t;
    }
    ... add(int riga, int colonna)...
}

class Giocatore extends Thread {
    private String name;
    private Scacchiera scacchiera;
    private int score=0;
    Giocatore(String n, Scacchiera s){name=n; scacchiera=s;}

    public void run(){
        for(int i=0; i<30; i++){
            int r=((int)(Math.random()*10))%Scacchiera.NRIGHE;
            int c=((int)(Math.random()*10))%Scacchiera.NCOLONNE;
            score +=scacchiera.add(r,c);
        }
               .....
    }
}

public class Gioco {
    public static void main(String[] a){
      Scacchiera s=new Scacchiera();
      Giocatore g1=new Gioco("pippo",s);
      Giocatore g2=new Gioco("mario",s);
      ....
    }
}
```