SISTEMI OPERATIVI IIN/IEL/IDT
INFORMATICA INDUSTRIALE E SISTEMI OPERATIVI IDI
SISTEMI DI ELABORAZIONE P.O.
prova scritta del 23.09.2005

Nome: _____ Cognome: _____

Un processo industriale utilizza una cisterna in cui possono aggiungere e prelevare liquido 2N pompe (N di ingresso ed N di uscita di uguale portata). Ogni pompa di ingresso è attivata ad intervalli di tempo casuali e rimane in funzione per il tempo necessario ad immettere un volume di Q m^3.
Al termine di ogni trasferimento è attivata una pompa di uscita che preleva un volume di Q m^3. Supponendo che la cisterna abbia capacità infinita, si vuole tenere traccia della quantità totale di liquido immesso nella cisterna e sincronizzare il funzionamento tra le pompe di ingresso e di uscita.

Supponendo che l'azione di ogni pompa possa essere rappresentata da un thread, si definiscano:

- una classe Java che riproduca il comportamento di una pompa di ingresso;
- una classe Java che riproduca il comportamento di una pompa di uscita;
- una classe Java che ad intervalli di tempo prefissati legga e visualizzi la quantità di liquido prodotto;
- un programma che istanzi ed avvii il thread che visualizza la quantità di liquido prodotto ed i thread corrispondenti alle N pompe di ingresso e N di uscita.

Soluzione

```
public class Contatore {
   private int totale = 0;
   private int incremento;
   public Contatore (int q) {
        incremento = q;
   }

   public void incrementa() {
        totale += incremento;
   }

   public int totale() {
        return totale;
   }
}

public class PompaIn extends Thread {
   private Contatore contatore;

   public PompaIn ( Contatore c ) {
        contatore = c;
   }

   public void run() {
        while ( true ) {
             synchronized( contatore ) {
                  contatore.incrementa();
             }

             /*
              * ... inserisci liquido ...
              */
             synchronized( contatore ) {
                  notify();
             }
        }
   }
}

public class PompaOut extends Thread {
   private Contatore contatore;

   public PompaOut ( Contatore c ) {
        contatore = c;
   }

   public void run() {
        while ( true ) {
             synchronized( contatore ) {
                  contatore.wait();
             }

             /*
              * ... rimuovi liquido ...
              */
        }
   }
}
```

```
public class Visualizzatore extends Thread {
   private Contatore contatore;

   public Visualizzatore( Contatore c ) {
        contatore = c;
   }

   public void run() {
        while ( true ) {
             try {
                  Thread.sleep( 1000 );
             }
             catch ( InterruptedException ie ) {
             }
             int v;
             synchronized( contatore ) {
                  v = contatore.totale();
             }
             System.out.println( "totale: " + v );
        }
   }
}

public class Main {
   public static final int N = 10;
   public static final int Q = 15;

   public static void main( String[] args ) {
        Contatore c = new Contatore(Q);

        for (int i=0; i<N; i++) {
             PompaIn pi = new PompaIn( c );
             pi.start();
        }

        for (int i=0; i<N; i++) {
             PompaOut po = new PompaOut( c );
             po.start();
        }

        Visualizzatore v = new Visualizzatore( c );
        v.start();
   }
}
```