SISTEMI OPERATIVI IIN/IEL/IDT
INFORMATICA INDUSTRIALE E SISTEMI OPERATIVI IDI
SISTEMI DI ELABORAZIONE P.O.
prova scritta del 20.7.2004

Nome: _____ Cognome: _____

Un gioco prevede la partecipazione di *n* giocatori, seduti intorno ad un tavolo circolare che utilizzano un mazzo di *2k+1* carte, accoppiate a due a due salvo una. Il numero *2k+1* è divisibile per *n* (2k+1=m*n) e sufficientemente grande. Gli indici dei giocatori sono compresi nell'intervallo *0 …n-1*.
Inizialmente le carte sono distribuite in numero uguale tra i giocatori; quindi, ogni giocatore scarta tutte le coppie in suo possesso e, se il suo indice è diverso da zero, si sospende. In una qualsiasi fase del gioco, il giocatore che esaurisce le sue carte viene escluso dal gioco. Il gioco è iniziato dal giocatore di indice 0 e quindi procede con una successione di giocate eseguite a turno dai giocatori non esclusi. Il giocatore di indice i al quale spetta il turno:

- individua l'indice *j* del giocatore non escluso che lo precede nell'ordinamento ciclico degli indici e preleva una carta scelta casualmente tra quelle in possesso del giocatore di indice *j*. Se questo giocatore esaurisce le carte viene escluso dal gioco;
- scarta la coppia eventualmente formatasi. Se ha esaurito le carte viene escluso dal gioco;
- in ogni caso individua l'indice *k* del giocatore non escluso che lo segue nell'ordinamento ciclico degli indici e lo riattiva; quindi si sospende.

Il gioco termina quando sono stati esclusi tutti i giocatori eccetto uno.

Risolvere il problema, scrivendo il programma del generico processo "*giocatore*", realizzando la sincronizzazione per mezzo di:

- Java (anno accademico 2003/04);
- semafori (anni accademici precedenti).

Schema di soluzione

```
/* classe utilizzata per gestire il mazzo di carte e la mano dei singoli giocatori */
public class InsiemeDiCarte {
  public InsiemeDiCarte pesca( int q ) {/* pesca q carte dall'insieme, e le inserisce in un nuovo insieme */ }
  public void aggiungi( InsiemeDiCarte c )  { /* aggiunge l'insieme c all'insieme */ }
  public int dimensione() { /* restituisce la dimensione dell'insieme */ }
}

public class Tavolo {
  public Tavolo( int k, int n, int m ) {
    mazzo = mazzo( k );
    giocatori = new Giocatore[ n ];
    blocchi = new Object[ n ];
    for( int t = 0; t < n; t++ ) {
      InsiemeDiCarte mano = mazzo.pesca( m );
      giocatori[ t ] = new Giocatore( t, mano, this );
      giocatori[ t ].start();
    }
    attuale = 0;
    synchronized( blocchi[ attuale ] ) blocchi[ attuale ].notify();
  }
  public Giocatore precedente() { /* resituisce il riferimento al giocatore precedente */ }
  public Giocatore successivo() { /* resituisce il riferimento al giocatore successivo */ }
  public InsiemeDiCarte mazzo( int k ) { /* restituisce il riferimento ad un nuovo mazzo di dimensione 2k+1 */ }
  public synchronized void attendi() {
    int id = ( (Giocatore) Thread.currentThread() ).id();
    try { blocchi[ id ].wait(); } catch ( InterruptedException ie ) {}
  }
  public synchronized void passa() {
    int a = attuale;
    Giocatore successivo = successivo();
    attuale = successivo.id();
    synchronized( blocchi[ attuale ] ) blocchi[ attuale ].notify();
    synchronized( blocchi[ a ] ) try { blocchi[ a ].wait(); } catch ( InterruptedException ie ) {}
  }
  private InsiemeDiCarte mazzo;
  private Giocatore[] giocatori;
  private Object[] blocchi;
  private int attuale;
  private int id;
}
```

```
public class Giocatore extends Thread {
  public Giocatore( int id, InsiemeDiCarte mano, Tavolo tavolo ) {
    this.id = id;
    this.mano = mano;
    this.tavolo = tavolo;
  }
  private boolean scarta() { /* scarta le coppie di carte uguali; restituisce TRUE se ha terminato */ }
  public InsiemeDiCarte pesca() { return mano.pesca( 1 ); }
  public void run() {
    terminato = scarta(); // analizza la mano inziale, e scarta se possibile
    while( !terminato ) {
      tavolo.attendi(); // attende il proprio turno
      Giocatore precedente = tavolo.precedente();
      if ( null != precedente ) {
        System.out.println( “giocatore ” + id + “ ha perso” );
        terminato = true;
      } else {
        mano.aggiungi( precedente.pesca() );
        terminato = scarta();
      }
      if ( !terminato ) tavolo.passa();
    }
  }
  public int id() { return id; }
  private int id;
  private InsiemeDiCarte mano;
  private Tavolo tavolo;
  private boolean terminato = false;
}
```