L'algoritmo di Peterson

L'**algoritmo di Peterson** combina le idee di *ContaPostiLiberi* e *MutuaEsclusione2*, discussi nei Paragrafi 2 e 3 del Capitolo 1. L'*algoritmo di Peterson* è un algoritmo basato sulla "gentilezza", perché, in caso di accesso contemporaneo alla sezione critica, entrambi i processi invitano l'altro a farsi avanti.

```
/*-------------------------------------------------------------------
 * L'algoritmo di Peterson
 *-------------------------------------------------------------------*/
class Peterson
{
   static boolean voglioEntrare[] = new boolean[2];
   static int turno;

   public static void main(String args[])
   {
      voglioEntrare[0] = false;
      voglioEntrare[1] = false;

      /*-----------------------------------------------------------------
       * P0 e P1 sono eseguiti in parallelo
       *-----------------------------------------------------------------*/
       P0();
       P1();
   }
/*-------------------------------------------------------------------
 * P0 esegue                                  P1 esegue
 *-----------                                ----------------------------------*/
   void P0()                                  void P1()
   {                                          {
      . . .                                      . . .
      voglioEntrare[0] = true;                   voglioEntrare[1] = true;
      turno = 1;                                 turno = 0;
      while (voglioEntrare[1] and                while (VoglioEntrare[0] and
            turno == 1) {}                             turno == 0) {}
      Sezione Critica 0                          Sezione Critica 1
      voglioEntrare[0] = false;                  voglioEntrare[1] = false;
      . . .                                      . . .
   }                                          }
}
```

Un processo prima di entrare nella propria sezione critica lo dichiara settando a *true* il proprio flag *voglioEntrare[x]*

Ciclo di attesa attiva

Per dimostrare il corretto funzionamento dell'*algoritmo di Peterson*, consideriamo il processo **P0** e osserviamo che, se **P0** è entrato nella sezione critica, vale l'uguaglianza:

```
voglioEntrare[0] = true
```

Di conseguenza **P1** non potrà entrare nella propria sezione critica sino a quando **P0** non la lascerà. Con analoghe argomentazioni si comprende che, se **P1** è nella *Sezione Critica 1*, **P0** non potrà entrare nella *Sezione Critica 0*. Inoltre, se **P0** e **P1** vogliono entrare contemporaneamente nella propria sezione critica, valgono le uguaglianze:

```
voglioEntrare[0] = voglioEntrare[1] = true
```

e il valore di *turno* sarà 0 o 1 in base all'ordine di esecuzione dell'istruzione *turno = valore*. L'ultimo processo che ha assegnato un valore a *turno* è quello forzato ad attendere.
L'*algoritmo di Peterson*, pur proponendo una soluzione software semplice, è comunque caratterizzato dal difetto di fare uso dell'attesa attiva.