Soluzione Esercitazione n.9

10 maggio 2010

Esercizio di programmazione concorrente in Java

Si realizzi un programma Java che simuli la gestione di una cucina di un ristorante.

In particolare:

- due thread **AiutoCuoco** si incaricano di portare sul tavolo di preparazione (risorsa condivisa) gli ingredienti necessari (di due tipi diversi)

- un thread **Chef** sovrintende alla preparazione del piatto, attende che siano disponibili le quantità necessarie di ingredienti; quando disponibili, le preleva dal tavolo e prepara il piatto

Esercizio 2- dettagli

- Tavolo ha **capacità limitata** (diversa) per ingrediente1 e ingrediente2:
  - M1 per ingrediente1
  - M2 per ingrediente2

- Ciascun ***AiutoCuoco*** porta sempre una unità di ingrediente

- Per preparare il piatto, lo **Chef** ha bisogno di quantità prestabilite per i 2 ingredienti:
  - Q1 per ingrediente1
  - Q2 per ingrediente2

Esercizio 2 - Sincronizzazione

- **Tavolo è una risorsa condivisa** con due buffer, uno per ciascun ingrediente:
  - gestione della capacità: non è possibile altre unità di ingredienteX se il tavolo ha già saturato la capacità per X
- necessità di mutua esclusione tra thread che accedono al tavolo (aggiornamento dello stato)
- thread **Chef** deve attendere che siano disponibili gli ingredienti nelle quantità stabilite; soltanto allora può iniziare la preparazione del piatto (ordinamento)

Esempio di soluzione

- Due tipi di thread:
  - AiutoCuoco: prepara un tipo di ingrediente (un'unità alla volta) e lo deposita sul tavolo . Una istanza per ingrediente: 2 thread.
  - Chef: usa gli ingredienti prodotti dagli aiutanti per realizzare un piatto

Risorsa condivisa: Tavolo

Il Tavolo è una risorsa condivisa dai thread concorrenti -> classe Tavolo

- Le variabili che caratterizzano lo stato della risorsa sono:
  - **quantita_ingrediente1: quantità del primo ingrediente sul tavolo ;**
  - **quantita_ingrediente2: quantità del secondo ingrediente sul tavolo ;**
  - **i1disp: quantità di ingr.1 disponibile per nuovi piatti**
  - **i2disp: quantità di ingr.2 disponibile per nuovi piatti**
- Sincronizzazione: 4 semafori

  s_ingrediente1: sospensione del primo aiuto cuoco (se lo spazio a disposizione per il primo ingrediente è esaurito) [v.i. M1]

  s_ingrediente2: sospensione del secondo aiuto cuoco (se lo spazio a disposizione per il secondo ingrediente è esaurito); [v.i. M2]

  s_preparazione: sospensione dello chef (se non sono pronti gli ingredienti nelle quantità richieste Q1 e Q2); [v.i. 0]

  sM: semaforo di mutua esclusione [v.i. 1]

Risorsa condivisa: Tavolo

Operazioni:

portaIngrediente1: deposito sul tavolo di una unità di ingrediente 1 (invocata dal primo aiutocuoco)

portaIngrediente2: deposito sul tavolo di una unità di ingrediente 2 (invocata dal secondo aiutocuoco)

preparazione: preparazione di un piatto (invocata dallo chef)

Thread chef

```
public class Chef extends Thread{
   int i,piattiPreparati = 0;
   Tavolo t;
   public Chef(Tavolo t){
        this.t=t;
   }
   public void run(){
        try{   System.out.println("Preparazione... ");
               while(true){
                      t.preparazione();
                      sleep(100);
                      piattiPreparati++;
               }
        }
        catch (InterruptedException e ){}
   }
}
```

Thread Aiuto Cuoco

```
public class AiutoCuoco extends Thread{
   int i = 0;
   Tavolo t;
   int tipo_ingrediente;
   public AiutoCuoco(Tavolo t, int ingrediente ){
        this.t=t;
        this.tipo_ingrediente=ingrediente;
   }
   public void run(){
        try{          while(i<30){
                      sleep(100);
                      if (tipo_ingrediente==1)
                            t.portaIngrediente1();
                      else if (tipo_ingrediente==2)
                            t.portaIngrediente2();
                      i++;
               }}catch (InterruptedException e ){}
   }}
```

Tavolo: risorsa condivisa

```
public class Tavolo {
   final int Q1 = 5; // quantità rich. Ingr. 1
   final int Q2 = 16; // quantità rich. Ingr. 2
   final int M1 = 10; //capienza ingr.1
   final int M2 = 20; //capienza ingr.2
   Semaforo s_ingrediente1, s_ingrediente2,s_preparazione;
   Semaforo sM; //mutua esclusione

   private int quantita_ingrediente1 = 0; //i1 sul tavolo
   private int quantita_ingrediente2 = 0; //i2 sul tavolo
   private int i1disp=0; //i1 non ancora allocato
   private int i1disp=0; //i2 non ancora allocato
```



```
//continua..
```

```
// continua classe Tavolo…

public Tavolo (){
        s_ingrediente1 = new Semaforo(M1);
        s_ingrediente2 = new Semaforo(M2);
        sM = new Semaforo(1);
        s_preparazione = new Semaforo(0);
   }


// continua..
```

```
//.. continua
public void portaIngrediente1(){
        s_ingrediente1.p();
        sM.p();
        quantita_ingrediente1 ++;
        i1disp++;
        if (i1disp == Q1 && i2disp >= Q2 )
        {       s_preparazione.v();
                i1disp-=Q1;
                i2disp-=Q2;
        }
        sM.v();

   }
```

```
//.. continua
public void portaIngrediente2()
   {
       s_ingrediente2.p();
       sM.p();
       quantita_ingrediente2 ++;
       i2disp++;
       if (i1disp1 >= Q1 && i2disp == Q2 )
       {     s_preparazione.v();
             i1disp-=Q1; //alloc. Q1 per nuovo piatto
             i2disp-=Q2; //alloc. Q2 per nuovo piatto
       }
       sM.v();
   }
```

```
public void preparazione()
{       s_preparazione.p();
        sM.p();
        quantita_ingrediente1-=Q1;
        quantita_ingrediente2-=Q2;
        for (int i=0; i< Q1; i++)
                s_ingrediente1.v();
        for (int i=0; i<Q2; i++)
                s_ingrediente2.v();
        sM.v();
        }
}// FINE TAVOLO
```

```
public class semaforo {
   private int value;
   public semaforo (int initial){
      this.value = initial;
   }
   synchronized public void v(){
       ++value;
      notify();
   }
   synchronized public void p() throws
     InterruptedException
   { while(value == 0) {
      try { wait();
      } catch (InterruptedException e) {}
   }
      value--;
   }}
```

Programma di test

```
public class Cucina {

   public static void main (String args[]){
       Tavolo t = new Tavolo();
       AiutoCuoco aiutoCuoco1 = new AiutoCuoco(t,1);
       AiutoCuoco aiutoCuoco2 = new AiutoCuoco(t,2);
       Chef chef = new Chef(t);

       aiutoCuoco1.start();
       aiutoCuoco2.start();
       chef.start();
   }
}
```