Corso sul linguaggio Java

Modulo JAVA9

B3.2 – Produttore-consumatore

Prerequisiti

- Elementi di programmazione concorrente
- Mutua esclusione
- Produttore-consumatore

Introduzione

In questa Unità vediamo il modo in cui Java semplicemente riesce a trattare problemi che si rifanno al modello produttore-consumatore.

Produttore-consumatore

Abbiamo visto che tramite la sincronizzazione un *thread1* può modificare in modo sicuro il valore di una variabile che potrà essere letta successivamente da un altro *thread2*.

Resta una questione: come fa thread2 a sapere che thread1 ha modificato i valori?

Produttore-consumatore

Ovviamente, soltanto se c'è un meccanismo mediante il quale i due *thread* possono comunicare. A questo tipo di concorrenza, si dà il nome espressivo di **produttore-consumatore** (che in seguito abbreviamo con la locuzione **modello PC**)

Il **produttore** è il *thread* che genera l'informazione, il **consumatore** è il *thread* che la utilizza.

Vediamo le caratteristiche di questo tipo di concorrenza.

Vincoli del modello PC

La comunicazione può avvenire in base a queste caratteristiche:

1. non si può leggere quando la memoria è vuota
2. non si può scrivere quando la memoria è piena

A tale scopo, usare solo parti di codice **synchronized** non è sufficiente. Ad esempio:

- il produttore acquisisce il **lock** sulla memoria
- la memoria è piena, il produttore si pone in attesa che si svuoti
- il consumatore deve accedere alla memoria per svuotarla, ma questa è bloccata dal produttore.
- si crea un possibile stallo (*deadlock*):

Vincoli del modello PC

Inoltre:

3. non si devono leggere cose già lette
4. non si devono scrivere cose non ancora lette

A tale scopo, usare solo parti di codice **synchronized** non è sufficiente. Ad esempio:

- il produttore acquisisce il lock
- scrive una nuova informazione
- il produttore riacquisisce il **lock** nuovamente, prima del consumatore
- il produttore sovrascrive un'informazione non ancora recuperata dal consumatore.

Modello PC (esempio errato)

```
class CellaCondivisa
{
   int valore = 10;
   // altri metodi di default
}
```

Iniziamo a definitre una variabile che sarà condivisa dai vari *client*.

Modello PC (esempio errato)

```
class Produttore extends Thread
{  CellaCondivisa cella;
   public Produttore (CellaCondivisa cella)
   {   this.cella = cella;   }
   public void run()
   {   for (int i = 1; i <= 10; ++i)
       {   synchronized (cella)
           {   ++(cella.valore);
               System.out.println ("Prodotto: " + cella.valore);
           }
       }
   }
}
```

Sembra corretto, ma non è in grado di avvisare il *thread Consumatore* che il dato è stato prodotto.

Modello PC (esempio errato)

```
class Consumatore extends Thread
{  CellaCondivisa cella;
   public Consumatore (CellaCondivisa cella)
   {   this.cella = cella;   }
   public void run()
   {   int valore_letto;
       for (int i = 1; i <= 10; ++i)
       {   synchronized (cella)
           {   valore_letto = cella.valore;
               System.out.println ("Consumato: " + valore_letto);
           }
       }
   }
}
```

Sembra corretto, ma non è in grado di avvisare il *thread Produttore* che il dato è stato consumato..

Modello PC (esempio errato)

Questo è il **main**() che lancia i due *thread*. Ma i risultati non sono quelli desiderati in base ai vincoli posti.

```
class ProduttoreConsumatore
{
   public static void main (String args[])
   {
         CellaCondivisa d = new CellaCondivisa();
         Produttore p = new Produttore(d);
         Consumatore c = new Consumatore(d);
         p.start();
         c.start();
   }
}
```


Il modello PC in Java

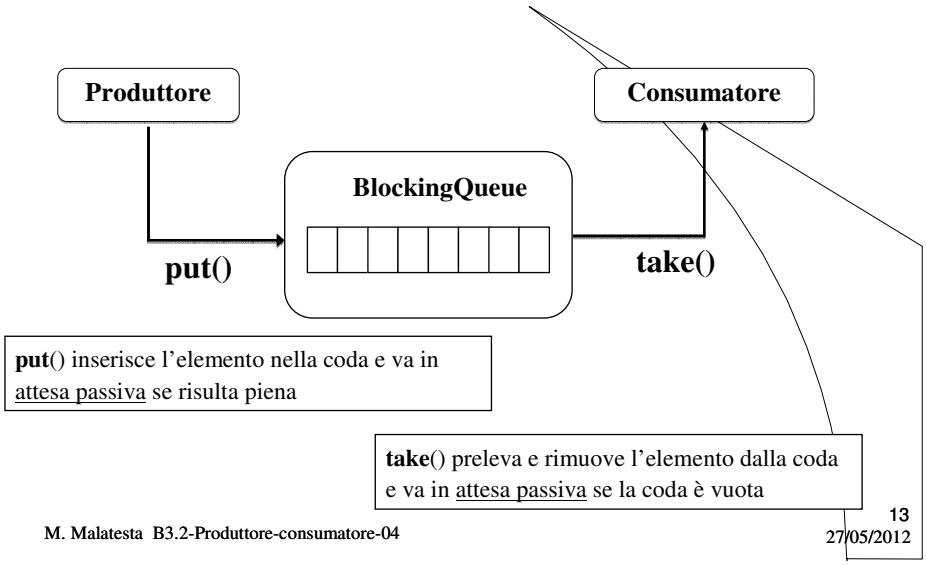
In altre parole, la cella condivisa è acceduta in mutua esclusione dal *Consumatore* (quando la legge) e dal *Produttore* (quando ci scrive) ma i due processi non si comunicano l'avvenuta scrittura e lettura.

Occorre allora introdurre uno strumento per sincronizzare *Produttore* e *Consumatore* in modo che rispettino i vincoli descritti in precedenza.

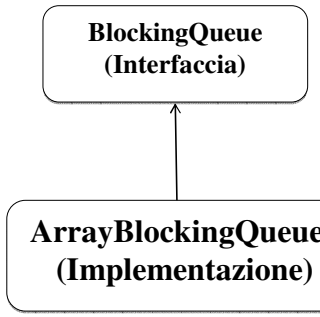
In teoria il modello è descritto dall'uso di 3 semafori, **mutex**, **Prelevato** e **Depositato**.

Come risolve Java il modello Produttore-Consumatore?

Il modello PC in Java



Java ArrayBlockingQueue



java.util.concurrent.ArrayBlockingQueue

(Introdotta a partire da java 1.5)

Esempio di BlockingQueue

```
import java.util.concurrent.ArrayBlockingQueue;
import java.util.concurrent.BlockingQueue;

public class EsempioBlockingQueue
{   public static void main (String args[ ]) throws Exception
    {     BlockingQueue queue = new ArrayBlockingQueue(20);

          Produttore produttore = new Produttore(queue);
          Consumatore consumatore = new Consumatore(queue);
          new Thread (produttore).start();
          new Thread (consumatore).start();
     }
}
```

Coda di 20 elementi

Produttore

```
public class Produttore extends Thread
{  protected BlockingQueue queue = null;
   public Produttore (BlockingQueue queue)    {  this.queue = queue;}
   public void run()
       {  try
          { int i=0;
            while (true)
            {   queue.put(i);
                i++;
                Thread.sleep(1000);
             }
       } catch (InterruptedException e)    {   e.printStackTrace();}
   }
}
```

Il *Produttore* avanza con ciclo infinito e mediante **put**() inserisce elementi nella coda.

put() va automaticamente in attesa passiva se la coda risulta piena

Consumatore

```
import java.util.concurrent.BlockingQueue;
public class Consumatore extends Thread
{   protected BlockingQueue queue = null;
    public Consumatore (BlockingQueue queue)  {     this.queue = queue;   }
    public void run()
  {       try
          {   while (true)
                  System.out.println(queue.take());
          }   catch (InterruptedException e)     {  e.printStackTrace();       }
  }
}
```

Il *Consumatore* avanza con ciclo infinito e mediante **take**() e preleva elementi dalla coda. **take**() va automaticamente in attesa passiva se la coda risulta vuota.

Argomenti

- Produttore-consumatore
- Vincoli del modello PC
- Modello PC (esempio errato)
- Il modello PC in Java
- Java ArrayBlockingQueue
- Esempio di BlockingQueue
- Produttore
- Consumatore

Altre fonti di informazione

- Camagni, Nicolassy–Java Interfacce grafiche e programmazione concorrente, ed. Hoepli
- Google: stallo in Java c'era parecchia roba
- Linguaggi\Java\Altro materiale\Threads.htm
- Yield() join()