

Esercizio sui Semafori

11 maggio 2010

Esercizio di programmazione concorrente in Java

- Si vuole realizzare un sistema per la gestione delle transazioni commerciali relative a un mercato ortofrutticolo all'ingrosso.
- Il sistema deve permettere a un numero arbitrario di thread concorrenti, ciascuno rappresentante un utente, di effettuare acquisti (nel caso di grossisti) e vendite (nel caso di produttori).
- Per semplicità si supponga che:
 - il mercato tratti solo un tipo di merce (es. ciliegie), che può essere acquistato/venduto a multipli interi del quintale
 - il prezzo di vendita al quintale (PV) e di acquisto al quintale (PA) siano costanti.
- Il mercato ha una capacità massima M che esprime il numero massimo di quintali di merce che possono essere immagazzinati.
- Gli utenti sono quindi di due tipi:
 - **Produttore**: vende al mercato un quantitativo Q_p di merce al prezzo PV; ottiene quindi la somma $Q_p * PV$
 - **Grossista**: acquista dal mercato un quantitativo Q_g di merce al prezzo PA; ottiene quindi la somma $Q_g * PA$
- Si sviluppi un'applicazione java che, rappresentando ogni utente come un thread concorrente, e utilizzando i semafori come strumenti di sincronizzazione tra thread, realizzi il sistema.

Esempio di soluzione

- Il mercato è una risorsa condivisa dai thread concorrenti -> classe risorsa
- Le variabili che caratterizzano lo stato della risorsa sono:
 - **Merce**: la quantità di merce in quintali depositata nel mercato
 - **Cassa**: la somma (in euro) disponibile nella cassa del mercato
- Due tipi di thread:
 - Produttore: immette merce e estrae euro
 - Grossista: immette soldi e estrae merce
- Sincronizzazione: 3 semafori
 - **sM**: mutua esclusione nell'accesso alla risorsa
 - **sP**: sospensione dei produttori
 - **sG**: sospensione dei grossisti

Thread produttore

```
import java.util.Random;

public class threadP extends Thread{ // thread Produttore
    risorsa r; // risorsa condivisa: mercato

    public threadP(risorsa R)// costruttore
    {
        this.r=R;
    }

    public void run()
    {
        int Qp;
        Qp=(int )(Math.random()*10+1);
        System.out.print("\n[P]: il mio ID è: "+getName()+" e
        voglio vendere "+ Qp + " quintali..\n");
        r.vendi(Qp); // accesso al mercato per vendere

    } //chiude run
}
```

Thread grossista

```
import java.util.concurrent.*;

public class threadG extends Thread{ // thread grossista
    risorsa r;

    public threadG(risorsa R)
    {    this.r=R;
    }

    public void run()
    {    int Qg;
        Qg=(int) (Math.random()*10+1) ;
        System.out.print("\n[G]: il mio ID è: "+getName()+"e
        voglio acquistare "+ Qg + " quintali..\n");
        r.compra(Qg); // accesso al mercato per acquistare
    } //chiude run
}
```

Mercato: risorsa condivisa

```
public class risorsa { // rappresenta il mercato
    final int PV=10; // prezzo di vendita per i produttori
    final int PA=12; // prezzo di acquisto per i grossisti
    final int MAX= 100; // capacità massima mercato
    int merce, cassa;
    semaforo sP; /* per la sospensione dei produttori; */
    semaforo sG; /* per la sospensione dei grossisti */
    int sospesiP; // numero thread produttori sospesi su Sp
                // in attesa di soldi o di spazio
    int sospesiG; // numero thread grossisti sospesi su sG
                // in attesa della merce richiesta

    semaforo sM; // semaforo di mutua esclusione

    // continua..
```

```
// continua classe risorsa...
```

```
public  risorsa(int M, int C) // costruttore
{
    sP=new semaforo(0);
    sG=new semaforo(0);
    sM=new semaforo(1); // sem. di mutua esclusione
    merce=M;
    cassa=C;
    sospesiG=0;
    sospesiP=0;

}
```

```
// continua..
```

```

//.. continua
public void compra(int quintali) // eseguita da grossisti
{
    int i;
    int prezzo; // prezzo totale
    prezzo=quintali*PA;
    try{ sM.p(); // inizio sezione critica
        while (quintali>merce)
        {
            sospesiG++;
            sM.v();
            sG.p(); // attesa
            sM.p();
        }
        merce=merce-quintali;
        cassa=cassa+prezzo;
        while (sospesiP>0) // risveglio i produttori
        {
            sP.v();
            sospesiP--;
        }
        sM.v(); //fine sez critica
    }catch(InterruptedException e){}
}
// continua..

```

```

//... continua
public void vendi(int quintali) // eseguita da produttori
{
    int i;
    int ricavo; // ricavo dalla vendita

    ricavo=quintali*PV;
    try{ sM.p(); // inizio sezione critica
        while ((ricavo>cassa) || (quintali+merce>MAX))
        {
            sospesiP++;
            sM.v();
            sP.p(); // attesa
            sM.p();
        }
        merce=merce+quintali;
        cassa=cassa-ricavo;
        while (sospesiG>0) // risveglio i grossisti
        {
            sG.v();
            sospesiG--;
        }
        sM.v(); //fine sez critica
    }catch(InterruptedException e){}
} }// fine classe risorsa

```

Programma di test

```
import java.util.concurrent.*;
public class mercato{
    public static void main(String args[]) {
        int i;
        final int NP=10;//produttori
        final int NG=10;    //grossisti
        final int MERCE=9; // quintali inizialmente nel mercato
        final int CASSA=50; //soldi disponibili in cassa
        risorsa R=new risorsa(MERCE, CASSA);
        System.out.print("\nSISTEMA AVVIATO: ci sono "+ MERCE +"
quintali e "+ CASSA + " EURO....\n");
        threadP []TP=new threadP[NP];        //thread produttori
        threadG []TG=new threadG[NG];        //thread grossisti
        for (i=0; i<NP; i++)
            TP[i]=new threadP(R);
        for (i=0; i<NG; i++)
            TG[i]=new threadG(R);
        for (i=0;i<NP; i++)
            TP[i].start();
        for (i=0;i<NG; i++)
            TG[i].start(); }}
}
```