

[Login >](#)

Secgroup Ca' Foscari DSI

- [Home](#)
- [Projects](#)
- [Teaching](#)
- [Competitions](#)
- [Contacts](#)
- [About](#)
- [Blog](#)



[Secgroup Ca' Foscari DSI](#) > [Teaching](#) > [Sistemi Operativi – modulo 2](#) > [Verifiche anni precedenti](#)
> [monitor] Gioco di squadra

- [Creazione di processi](#)
- [Esecuzione e terminazione](#)
- [Segnali](#)
- [Comunicazione tra processi](#)
- [Pipe](#)
- [Esercitazione sulla pipe](#)
- [Produttore e consumatore](#)
- [I Thread POSIX](#)
- [Sezione critica](#)
- [Semafori](#)
- [Programmazione con i semafori](#)
- [Semafori POSIX](#)
- [Monitor](#)
- [Thread in Java](#)
- [Programmazione con i Monitor](#)
- [Stallo](#)
- [Risultati verifiche](#)
- [Verifiche anni precedenti](#)
 - [\[2012-13\] Semafori: robots](#)
 - [\[2012-13\] Monitor: scheduler](#)
 - [\[2011-12\] Pipe](#)
 - [\[2011-12\] Semafori](#)
 - [\[2011-12\] Monitor](#)
 - [\[pipe\] Crackme](#)
 - [\[semafori\] Check-in in aeroporto](#)
 - [\[monitor\] Gioco di squadra](#)

[monitor] Gioco di squadra

Ci sono 4 squadre contrassegnate dai colori Rosso, Bianco, Verde, Blu, codificati con i numeri da 0 a 3. Ogni squadra è composta da `numComponenti` giocatori, identificati con in numeri da 0 a

numComponenti-1.

Le squadre devono cercare di entrare in una porta prima delle altre seguendo le regole seguenti:

- nessun giocatore deve entrare prima che la porta sia stata aperta dal thread principale (main) tramite l'invocazione del metodo `porta.apri()`. A tale scopo i thread invocano il metodo `porta.attendi()`;
- solo il capitano della squadra (giocatore con id 0) compete con gli altri capitani per l'accesso alla porta (metodo `porta.entra(...)`);
- quando un capitano riesce ad entrare tutti i giocatori della sua squadra possono entrare (sempre metodo `porta.entra(...)`) ma devono entrare in fila: 1,2,3, ... , numComponenti-1
- mentre una squadra entra le altre squadre attendono. Solo quando tutta la squadra è entrata i capitani possono di nuovo competere per la porta.

L'output atteso dovrebbe essere il seguente: Tutti i giocatori vengono creati e stampano a video che sono pronti (notare che una sleep 'mescola' l'ordine dei giocatori). Quando la porta viene aperta, le varie squadre entrano nella porta una alla volta. Ad esempio:

```
La porta e' aperta!!
Componente Blu numero 0 e' entrato!
Componente Blu numero 1 e' entrato!
Componente Blu numero 2 e' entrato!
Componente Blu numero 3 e' entrato!
Componente Blu numero 4 e' entrato!
Componente Blu numero 5 e' entrato!
Componente Blu numero 6 e' entrato!
Componente Blu numero 7 e' entrato!
Componente Blu numero 8 e' entrato!
Componente Blu numero 9 e' entrato!
Componente Rosso numero 0 e' entrato!
Componente Rosso numero 1 e' entrato!
Componente Rosso numero 2 e' entrato!
Componente Rosso numero 3 e' entrato!
Componente Rosso numero 4 e' entrato!
Componente Rosso numero 5 e' entrato!
Componente Rosso numero 6 e' entrato!
Componente Rosso numero 7 e' entrato!
Componente Rosso numero 8 e' entrato!
Componente Rosso numero 9 e' entrato!
Componente Verde numero 0 e' entrato!
Componente Verde numero 1 e' entrato!
...
```

L'ordine dei colori varia da un'esecuzione all'altra in quanto dipende da quale capitano è più veloce.

Scopo della esercitazione è **implementare il monitor 'Porta'** con i tre metodi sopra descritti. Utilizzare il seguente programma di test:

```
import java.lang.Math;

public class Test extends Thread {
    private static final String colori[]={"Rosso","Bianco","Verde","Blu"}; // colori
    private static final int numColori = colori.length; // quanti colori/squadre ci sono
```

```
private static final int numComponenti = 10; // numero componenti
private final int col, num; // colore e numero del giocatore
private final Porta porta; // monitor utilizzato dal giocatore

// costruttore: salva colore numero e monitor
Test(int col, int num, Porta porta) {
    this.col = col;
    this.num = num;
    this.porta = porta;
}

// lancia il codice vero e proprio e cattura l'eccezione di interruzione
public void run() {
    try {
        code();
    } catch (InterruptedException e) {
        System.out.println("Componente "+colori[col]+" numero "+num+" interrotto!!");
    }
}

// codice dei thread
void code() throws InterruptedException {
    // evita che i thread siano gia' in fila
    sleep((int)(1000*Math.random()));

    // il giocatore e' pronto
    System.out.println("Componente "+colori[col]+" numero "+num+" pronto!");

    // attende che la porta venga aperta dal main
    porta.attendi();

    // cerca di entrare
    porta.entra(col,num);

    // attenzione che questo output non e' sincronizzato con entra.... quindi non e' de
    // che venga stampato nell'ordine corretto. Aggiungere un blocco synchronized(porta,
    // se necessario
    System.out.println("Componente "+colori[col]+" numero "+num+" e' entrato!");
}

public static void main(String argv []) throws InterruptedException {
    int i,j;

    Porta p = new Porta(numComponenti); // crea il monitor

    // crea i thread dei vari colori/numeri
    for (i=0; i<numColori; i++)
        for (j=0; j<numComponenti; j++) {
            (new Test(i,j,p)).start();
        }

    // attende 3 secondi prima di aprire la porta (tutti i thread devono attendere)
    sleep(3000);
    p.apri();
    System.out.println("La porta e' aperta!!");
}
}
```

Comments: 17

[Leave a reply »](#)



Davide Moro

[March 29th, 2013 at 18:16](#)

Sembra funzionare ...

```
1  class Porta {
2      // memoria condivisa
3      private boolean portaAperta; // true se è aperta
4      private int squadraCorrente;
5      private int contGiocatori;
6      private int maxGiocatori;
7
8      // all'inizio le variabili sono
9      public Porta() {
10         portaAperta = false; // è chiusa
11         squadraCorrente = -1; // numero di squadra impossibi
12     }
13
14     public Porta(int giocatori) {
15         this();
16         // così contGiocatori % maxGiocatori = 0 fa entrare
17         maxGiocatori = giocatori;
18         contGiocatori = maxGiocatori;
19     }
20
21     public synchronized void apri() {
22         portaAperta = true;
23         notifyAll(); // dopo aver aperto la porta sblocco t
24     }
25
26     // metodi in mutua esclusione
27     public synchronized void attendi() throws InterruptedException {
28         if (!portaAperta) // se la porta è chiusa il thread
29             wait();
30     }
31
32     public synchronized void entra(int squadra, int giocatore) {
33         // sincronizzo distinguendo tra giocatori capitani e
34         if (giocatore == 0) {
35             // un capitano attende se non è il suo turno oppure
36             while (giocatore != (contGiocatori % maxGiocatori))
37                 wait();
38         }
39         // altrimenti imposta come squadra corrente quella
40         squadraCorrente = squadra;
41         contGiocatori = 1;
```

```

42         // e per sbloccare i thread che non sono capitar
43         notifyAll();
44     }
45     // un giocatore attende se non è il suo turno oppure
46     else {
47         while(giocatore != contGiocatori || squadra != s
48             wait();
49         }
50         contGiocatori++;
51         // se il giocatore era l'ultimo, non ci sono più
52         if(contGiocatori == maxGiocatori) {
53             squadraCorrente = -1;
54         }
55         // devo sbloccare i giocatori della propria squa
56         // o quelli delle altre squadre se il giocatore
57         notifyAll();
58     }
59 }
60 }

```



Gabriele Volpato

[April 4th, 2013 at 23:19](#)

Sembra funzionare perfettamente 😊

```

1  public class Porta {
2      private boolean aperta; // per vedere se la porta è aper
3      private int giocatore; // numero del giocatore corrente
4      private int numgiocatori; // totale giocatori per squad
5      private int colore; // colore della squadra corrente
6
7      public Porta(int n) {
8          aperta = false; //inizialmente la porta è chiusa
9          colore = -1; // non c'è nessuna squadra all'inizio
10         giocatore = -1; // non c'è nessun giocatore all'iniz
11         numgiocatori = n;
12     }
13
14     synchronized void apri() {
15         aperta = true; //apro la porta
16         notifyAll(); // quando è aperta lo notifico a tutti
17     }
18
19     synchronized void attendi() throws InterruptedException
20         while (!aperta) { //finché non è aperta, aspetto
21             wait();
22         }
23     }
24
25     synchronized void entra(int col, int num) throws Interru
26         if (num == 0) {

```

```

27         if (giocatore == -1){ //sono all'inizio
28             colore = col; // ora la mia squadra sta entr
29             giocatore = num;
30             notifyAll();
31         }
32         else while (giocatore != numgiocatori - 1) { //
33             wait();
34         }
35         colore = col; // ora la mia squadra sta entrand
36         giocatore = num;
37         notifyAll();
38     }
39     else {
40         while (num != giocatore + 1 || col != colore) {
41             wait();
42         }
43         giocatore = num;
44         notifyAll();
45     }
46 }
47 }

```



Claudio

[April 7th, 2013 at 13:03](#)

Posto la mia soluzione, che è molto simile a quella di Davide, ma usa una matrice. Per rendere più “generico” il codice ho messo a public la stringa dei colori nella classe Test. Tanto è comunque final quindi è impossibile modificarla.

Ho anche commentato la riga di output nella classe Test, perché tanto avevo messo vari output sulla classe Porta e veniva un po’ strano leggere due volte “è entrato”.

Classe Test:

```

1  public class Test extends Thread {
2      public static final String colori[]={"Rosso","Bianco","Ver
3      private static final int numColori = colori.length; // qua
4      private static final int numComponenti =10; // numero comp
5      private final int col, num; // colore e numero del giocat
6      private final Porta porta; // monitor utilizzato dal giocat
7
8      // costruttore: salva colore numero e monitor
9      Test(int col,int num, Porta porta) {
10         this.col = col;
11         this.num = num;
12         this.porta = porta;
13     }
14
15     // lancia il codice vero e proprio e cattura l'eccezione c
16     @Override
17     public void run() {
18         try {

```

```

19         code();
20     } catch (InterruptedException e) {
21         System.out.println("Componente "+colori[col]+" numero
22     }
23 }
24
25 // codice dei thread
26 void code() throws InterruptedException {
27     // evita che i thread siano gia' in fila
28     sleep((int) (1000*Math.random()));
29
30     // il giocatore e' pronto
31     System.out.println("Componente "+colori[col]+" numero "+
32
33     // attende che la porta venga aperta dal main
34     porta.attendi();
35
36     // cerca di entrare
37     porta.entra(col,num);
38
39     // attenzione che questo output non e' sincronizzato con
40     // che venga stampato nell'ordine corretto. Aggiungere u
41     // se necessario
42
43     //synchronized(porta) { System.out.println("Componente '
44
45 }
46
47 public static void main(String argv []) throws Interruptec
48     int i,j;
49
50     Porta p = new Porta(numComponenti); // crea il monitor
51
52     // crea i thread dei vari colori/numeri
53     for (i=0; i<numColori; i++)
54         for (j=0; j<numComponenti; j++) {
55             (new Test(i,j,p)).start();
56         }
57
58     // attende 3 secondi prima di aprire la porta (tutti i t
59     sleep(3000);
60     p.apri();
61     System.out.println("La porta e' aperta!!");
62 }
63
64 }

```

Classe Porta:

```

1 public class Porta {
2
3     //Variabili globali
4     // aperta -> flag che identifica se la porta è aperta c

```

```
5     private boolean aperta;
6
7     // matrix -> una matrice di booleani che mi identifica c
8     //           per ogni squadra sono passati o no
9     private boolean[][] matrix;
10
11    // tot -> totale dei giocatori per squadra
12    // lastColor -> colore della squadra corrente. All'inizi
13    //           squadra, quindi è settato a -1
14    private int tot, lastColor = -1;
15
16    public Porta(int tot){
17
18        // Setto i dati di default.
19        aperta = false;
20        matrix = new boolean[Test.colori.length][tot];
21        this.tot = tot;
22        init();
23
24    }
25
26    // Inizializza la matrice
27    private void init(){
28
29        for(int i=0; i<4; i++)
30            for(int j=0; j<tot; j++)
31                matrix[i][j]=false;
32    }
33
34    //Quando la porta viene aperta, avviso tutti dell'evento
35    public synchronized void apri(){
36
37        aperta = true;
38        notifyAll();
39    }
40
41    //Finché la porta è chiusa, tutte le squadre attendono
42    public synchronized void attendi() throws InterruptedExc
43
44        while(aperta==false) wait();
45
46    }
47
48    //Ho diviso l'entrata in 3 casi:
49    // Caso 1: È il capitano ad entrare
50    // Caso 2: È l'ultimo giocatore di una squadra ad entra
51    // Caso 3: È uno dei giocatori ad entrare
52    public synchronized void entra(int col, int num) throws
53
54        //Se è il capitano entra. Esso non può entrare se la
55        //perché questo significa che qualche squadra sta gi
56        if(num==0) {
57            while(lastColor!=-1) wait();
```



```

58         lastColor=col;
59         System.out.println("Il primo componente "+Test.c
60     }
61     //Se è l'ultimo giocatore ad entrare. Esso deve atte
62     //non è ancora entrato o se non è la sua squadra que
63     //entrato setta a -1 lastColor così da indicare alle
64     //di entrare.
65     else if(num==tot-1) {
66         while(lastColor!=col || matrix[col][num-1]==false
67             lastColor = -1;
68         System.out.println("L'ultimo componente "+Test.c
69     }
70     //Come per l'ultimo giocatore, ma senza il setting c
71     else {
72         while(lastColor!=col || matrix[col][num-1]==false
73             System.out.println("Componente "+Test.colori[col
74     }
75     //Dopo essersi entrati setto la posizione nella matr
76     //tutti dell'evento.
77     matrix[col][num]=true;
78     notifyAll();
79
80
81     }
82
83 }

```



Roberta Prendin

[April 7th, 2013 at 18:42](#)

Ecco la mia soluzione, sembra (e sottolineo sembra) funzionare. Ho risolto creando una coda solo per i capitani. Ho anche dovuto sincronizzare la stampa dei componenti entrati, com'era stato suggerito nei commenti della classe Test.

(Ammetto d'aver trovato la soluzione – s'è giusta – dopo due giorni di tentativi... è stata un'esperienza! : D)

```

1  import java.util.ArrayList;
2
3  public class Porta {
4      // variabili globali
5      private boolean porta; // la porta da aprire o chiudere
6      private int numComponenti; // il numero di componenti di
7      private ArrayList coda; // una coda da usare solo per i
8      private int indice; // l'indice dei giocatori che possor
9                          // porta
10     private int passati; // il numero di giocatori già passa
11
12     // costruttore
13     public Porta(int numComponenti) {
14         porta = false; // la porta è chiusa

```

```
15         this.numComponenti = numComponenti;
16         indice = 0; // inizialmente passa solo il capitano
17         coda = new ArrayList(); // inizialmente la coda è vuota
18         passati = 0; // inizialmente non è passato nessuno
19     }
20
21     /* Apri: il metodo apre la porta alle squadre */
22     public synchronized void apri() {
23         porta = true; // ora la porta è aperta
24         notifyAll(); // notifico tutti i thread che è possibile
25                     // passare
26     }
27
28     /* Attendi: se la porta non è stata aperta dal main blocco
29     public synchronized void attendi() throws InterruptedException {
30         while (porta == false) {
31             wait();
32         }
33     }
34
35     /*
36     * Entra: il metodo riceve tutti i giocatori, quindi fa
37     * quattro capitani quindi fa passare i restanti giocatori
38     * capitano.
39     */
40     public synchronized void entra(int colore, int id)
41         throws InterruptedException {
42         /* se arriva il capitano, lo aggiungo alla coda segreta
43         if (id == 0) {
44             coda.add(colore);
45         }
46         /*
47         * tutti i thread vogliono passare: se però il loro
48         * valore (0, 1, 2... incremento dopo incremento) oppure
49         * colore non è quello del primo capitano nella coda
50         */
51
52         while (id != indice || colore != coda.get(0)) {
53             wait();
54         }
55         /*
56         * ogni volta che esco dal ciclo while vuol dire che ho
57         * un nuovo capitano, quindi incremento l'indice e il valore
58         * passati per la porta
59         */
60         indice = (indice + 1) % numComponenti;
61         passati++;
62         /*quando sono passati tutti i giocatori, rimuovo il
63         if (passati == numComponenti) {
64             coda.remove(0);
65             passati = 0;
66         }
67         notifyAll(); //notifico gli altri thread
```

```
68     }  
69 }
```



[riccardo](#)

[April 8th, 2013 at 16:04](#)

@Davide, Gabriele, Claudio: tutti avete duplicato eccessivamente il codice o messo condizioni ridondanti. Provate a semplificare e rendere più leggibile. Ricordatevi che è facile fare errori che magari saltano fuori raramente quindi è estremamente importante che il codice sia il più semplice possibile.

@Roberta: la tua soluzione è interessante e molto leggibile. Hai però introdotto una forzatura: i capitani si accodano a priori e non competono tutte le volte che la porta è libera. Prova a modificare il codice in modo che non sia deciso a priori quale capitano entra al turno successivo.



Mattia Corami, Diego Cornello

[April 9th, 2013 at 12:50](#)

```
1  class Porta{  
2      private int numComponenti;  
3      private boolean portaAperta;  
4      private int proxGiocatore;  
5      private int coloreEntrato;  
6  
7      public Porta(int num) {  
8          numComponenti=num;  
9          portaAperta=false; //porta chiusa  
10         proxGiocatore=0; //primo giocatore deve essere un cap  
11         coloreEntrato=-1; //nessun colore assegnato  
12     }  
13  
14     public synchronized void apri() {  
15         portaAperta=true; //porta aperta  
16         notifyAll(); //libera tutti  
17     }  
18  
19     // attende che la porta venga aperta dal main  
20     public synchronized void attendi() {  
21         while(!(portaAperta)) {  
22             try{  
23                 wait(); //se la porta è chiusa attendo  
24             }  
25             catch (InterruptedException e) {  
26                 System.out.println(Thread.currentThread().ge  
27             }  
28         } //while  
29     }  
30  
31     // cerca di entrare
```

```

32     public synchronized void entra(int col,int num){
33         if(num==0 && proxGiocatore==0)//è capitano
34             coloreEntrato=col;//setto il colore della squadra
35
36         while(num!=proxGiocatore || col!=coloreEntrato){//no
37             try{
38                 wait();
39                 if(num==0 && proxGiocatore==0)//se è capitano
40                     coloreEntrato=col;
41             }
42             catch(InterruptedException e){
43                 System.out.println(Thread.currentThread().getName());
44             }
45         }//while
46
47         proxGiocatore=(proxGiocatore+1)%numComponenti;//calcolo
48
49         notifyAll();//libero tutti
50     }
51
52 }//Monitor Porta

```



Rocco

[April 9th, 2013 at 17:58](#)

Ecco la mia soluzione, in principio era simile a quella di Roberta, poi ho provato a fare la modifica richiesta per far competere nuovamente i capitani.

```

1  import java.util.LinkedList;
2  class Porta {
3      Boolean porta;
4      LinkedList l;
5      int numComponenti;
6      int coloreattuale;
7      int numeroatteso;
8      Porta(int n){
9          porta=false;
10         l=new LinkedList();
11         coloreattuale=-1;
12         numeroatteso=-1;
13         numComponenti=n;
14     }
15     synchronized void attendi(){
16         while(porta==false){//se la porta non è aperta attendi
17             try {
18                 wait();
19             }
20             catch(InterruptedException ie) {
21                 System.out.println("Interrotto!");
22             }
23         }

```

```

24     }
25     synchronized void apri() {
26         porta=true; //apro la porta
27         notifyAll(); //comunico a tutti dell'apertura
28     }
29     synchronized void entra(int col,int num) {
30         if (num==0) { //se sei il capitano competi
31             l.add(col); //ti aggiungo per competere
32             while (l.peek()!=col) { //se non sei il primo aspe
33                 try {
34                     wait();
35                     if (l.isEmpty()) //se la lista è vuota, a
36                         entra(col,num); //richiamo entra
37                 }
38                 catch (InterruptedException ie) {
39                     System.out.println("Interrotto!");
40                 }
41             }
42             coloreattuale=col; //il colore della squadra che
43             numeroatteso=num; //il capitano è il primo a pass
44         }
45         while (col!=coloreattuale || num!=numeroatteso) { //a
46             try {
47                 wait();
48             }
49             catch (InterruptedException ie) {
50                 System.out.println("Interrotto!");
51             }
52         }
53         numeroatteso++; //prossimo compagno di squadra che de
54         if (numeroatteso==numComponenti) { //se è passata tut
55             coloreattuale=-1;
56             numeroatteso=-1;
57             l.clear(); //pulisco da lista, in modo da poter a
58         }
59         notifyAll(); //avviso tutti, i capitani in questo moc
60     }
61 }

```



Loris

[April 9th, 2013 at 19:05](#)

Posto anche io la mia soluzione. Ho usato un booleano per sincronizzare i 4 capitani (segnala agli altri 3 che devono attendere).

Per far passare il resto dei giocatori in ordine ho utilizzato una matrice di booleani (solo se il giocatore n-1 e' passato allora posso passare anche io)

Ho dovuto mettere in monitor la stampa per far visualizzare l' output correttamente. (E' corretto dire che e' necessario metterla in mutex perchè potrei essere

bloccato dallo scheduler prima di fare la stampa, e qualche altro processo mi 'soffia' il posto e stampa prima di me?)

```
1  public class Porta {
2      //Costruttore del monitor
3      boolean isaperta = false;
4      boolean[][] giocatori = new boolean[4][10]; //Matrice di
5      boolean isfree; //Mi serve per avvisare gli altri capitani
6      int numComponenti;
7
8      public Porta(int numComponenti) {
9          int i,j;
10         for (i=0;i<4;i++)
11             for (j=0;j<numComponenti;j++)
12                 giocatori[i][j]=false; //All' inizio la
13         isfree=true;
14         this.numComponenti=numComponenti;
15     }
16
17     synchronized void apri() {
18         isaperta=true;
19         notifyAll(); //Notifica che la porta e' aperta a tutti
20     }
21
22     synchronized void attendi() throws InterruptedException {
23         while(!isaperta) //Se la porta non e' aperta mette
24             wait();
25     }
26
27     synchronized void entra(int colore,int numero) throws InterruptedException {
28         int i;
29         if(numero!=0) { //Se non sono il capitano
30             while(!giocatori[colore][0]) //aspetto che sia
31                 wait();
32         }
33         else { //Se sono il capitano
34             while(!isfree) //Se c'e' gia' qualche capitano c'
35                 wait();
36             isfree=false; //Prendo la porta
37             giocatori[colore][numero] = true; //Avviso quelli
38             notifyAll();
39             return; //Il capitano ha fatto il suo dovere
40         }
41         //Ora gli altri membri della mia squadra possono passare
42
43         while(giocatori[colore][numero-1] == false) //Finche'
44             wait();
45         giocatori[colore][numero] = true; //Sono passato
46         if(numero==numComponenti-1) //Se sono l' ultimo
47             isfree=true;
48         notifyAll(); //notifico gli altri giocatori, possono
49     }
50 }
```

[riccardo](#)

[April 10th, 2013 at 10:13](#)

@Mattia e Diego: Avete duplicato il codice:

```

33 |   if (num==0 && proxGiocatore==0) //è capitano
34 |       coloreEntrato=col; //setto il colore della squadra

```

Non mi pare necessario.

@Rocco: non compila! Inoltre mi pare difficile possa funzionare con una chiamata ricorsiva a entra ...

@Loris: perché serve il codice qui sotto?

```

29 |   if (numero!=0) {           //Se non sono il capitano
30 |       while (!giocatori[colore][0]) //aspetto che sia entrato
31 |           wait();
32 |   }

```



Francesco

[April 10th, 2013 at 12:06](#)

```

1 |   public class Porta{
2 |       //aperta mi serve per indicare quando il mutex della porta
3 |       private boolean aperta;
4 |       //numComponenti indica quanti componenti della squadra ci
5 |       private int numComponenti;
6 |       //indica il colore della squadra che sta passando; a -1
7 |       private int colore_squadra=-1;
8 |       //ultimo_entrato indica il numero del giocatore entrato prima
9 |       private int ultimo_entrato=-1;
10 |
11 |       public Porta(int numComponenti){
12 |           this.numComponenti=numComponenti;
13 |           aperta=false;
14 |       }
15 |       //il main apre la porta e quindi sblocca tutti i thread in
16 |       public synchronized void apri(){
17 |           aperta=true;
18 |           notifyAll();
19 |       }
20 |       //se la porta non è stata aperta dal main faccio attendere
21 |       public synchronized void attendi() throws InterruptedException{
22 |           while (!aperta){
23 |               wait();
24 |           }
25 |       }
26 |
27 |
28 |   }
29 |
30 |       //metodo che gestisce l'ingresso corretto delle squadre in

```

```

31     public synchronized void entra(int col,int num) throws Int
32         //in questo while controllo che il giocatore abbia il
33         // e che sia del colore della squadra che sto facendo
34         while( (ultimo_entrato!=num-1) || (colore_squadra!=col) )
35             //finchè non setto il colore_squadra tutti i giocatori
36             //se però entra un capitano e mi accorgo che il colore è sbagliato
37             if((num==0) && (colore_squadra==col))
38                 colore_squadra=col;
39             else
40                 //altrimenti se il giocatore non rispetta l'ordine
41                 wait();
42
43         }
44         //il blocco di istruzioni che segue viene eseguito solo se il colore è quello della
45         //per la squadra che sto attualmente facendo passare
46
47         //devo controllare quando arriva l'ultimo giocatore e se è il capitano
48         if(ultimo_entrato+1==numComponenti-1){
49             colore_squadra=-1; //con questa impostazione resta il colore della squadra
50             ultimo_entrato=-1; //con questa impostazione reinizializzo il colore della squadra
51         }
52         else
53             //se il giocatore non è l'ultimo incremento ultimo_entrato
54             ultimo_entrato++;
55         //ogni volta che mi passa il giocatore con il colore sbagliato
56         //tra cui vi sarà presente il prossimo giocatore da far entrare
57         notifyAll();
58     }
59 }
60 }

```



Loris

[April 10th, 2013 at 13:34](#)

Mi serve per separare i giocatori dai capitani, così nell' else gestisco solo i capitani. Poi i capitani competeranno per la porta. Mi sembrava piu' chiaro così'.

Aveva qualcos' altro in mente?



[riccardo](#)

[April 10th, 2013 at 13:39](#)

@Francesco: la tua soluzione è molto compatta ma ti suggerirei di spostare la condizione di bloccaggio dei capitani dentro il while in modo da preservare lo schema standard while(.....) wait(); Io farei qualcosa di questo genere:

```

34     while( (num>0 && (ultimo_entrato!=num-1 || colore_squadra!=col) )
35         wait();
36         if (num==0)
37             colore_squadra=col;

```


Direi che con questa modifica la tua soluzione è quella minimale, non si può fare molto meglio 😊



[riccardo](#)

[April 10th, 2013 at 16:43](#)

@Loris: Qualcos'altro in mente? Direi di sì: toglierlo non serve a nulla!



Loris

[April 10th, 2013 at 17:58](#)

Wooops, era ridondante. Ecco la versione modificata

```
1  synchronized void entra(int colore,int numero) throws Interru
2      int i;
3      //Se sono il capitano
4      if(numero==0)
5          while(!isfree) //Se c'e' gia' qualche capitano c
6              wait();
7          isfree=false; //Prendo la porta
8          giocatori[colore][numero] = true; //Avviso quell
9          notifyAll();
10         return; //Il capitano ha fatto il suo dovere
11     }
12     //Ora gli altri membri della mia squadra possono pas
13
14     while(giocatori[colore][numero-1] == false)
15         wait();
16     giocatori[colore][numero] = true; //Sono passato
17     if(numero==numComponenti-1) //Se sono l' ultimo
18         isfree=true;
19     notifyAll(); //notifico gli altri giocatori, pas
20 }
```

Mi sono accorto pero' che va solo 9 volte su 10

Componente Bianco numero 0 e' entrato!

Componente Bianco numero 1 e' entrato!

Componente Bianco numero 2 e' entrato!

Componente Blu numero 9 e' entrato!

Componente Bianco numero 3 e' entrato!

Puo' essere perche' notifico prima di uscire dal metodo 'entra' e mi rubano il mutex quando cerco di ottenerlo per stampare?



[riccardo](#)

[April 10th, 2013 at 19:19](#)

C'è un commento a proposito di questo problema nel test

```
1 // attenzione che questo output non e' sincronizzato con enti
2 // che venga stampato nell'ordine corretto. Aggiungere un bloc
3 // se necessario
4 System.out.println("Componente "+colori[col]+" numero "+num+" "
```



Giulio Lazzaro

[April 10th, 2013 at 23:15](#)

Posto la mia soluzione anche se in notevole ritardo (sono entrato a conoscenza dell'esercizio solo ora).

È sostanzialmente simile a tutte quelle postate fino ad ora tranne nel controllo “entra”.

```
1 public class Porta
2 {
3     private boolean aperta = false;;           // Valore dell'a
4     private int numero = 0;                     // Contatore del
5     private int colore = -1;                    // Colore della
6     private int numComponenti;                  // NumComponenti
7
8     // Inizializzo costruttore con numComponenti base per t
9     public Porta (int numComponenti)
10    {
11        this.numComponenti = numComponenti;
12    }
13
14    // Apro la porta e notifico tutti (potrebbero esserci gi
15    synchronized void apri()
16    {
17        aperta = true;
18        notifyAll();
19    }
20
21    // Attendo che la porta venga aperta (dal metodo apri ch
22    synchronized void attendi() throws InterruptedException
23    {
24        while (!aperta)
25            wait();
26    }
27
28    // Metodo entra
29    synchronized void entra(int col,int num) throws Interrup
30    {
31        // Solo se sono il giocatore n-esimo della squadra '
32        // Siccome il numero (ticket) iniziale è 0, solo un
33        while (num != numero || (colore != -1 && colore != c
34            wait();
35
36        /* Se entro: 1) vuol dire che sono il capitano (comi
37                2) Sono il giocatore n-esimo della squa
```

```
38         colore = col;    // Imposto il colore (capitano lo fa
39         numero++;        // Sono passato, il ticket aumenta e
40         System.out.print( "Giocatore" + "["+col+"]"+num + "e
41
42         // Quando io sono l'ultimo giocatore della squadra ]
43         if (numero == numComponenti)
44         {
45             colore = -1;
46             numero = 0;
47         }
48         // Notifico tutti
49         notifyAll();
50     }
51 }
```



Loris

[April 11th, 2013 at 00:24](#)

Ho gia' messo la stampa in mutex!

Leave a Reply

Name *

Mail *

(will not be published)

Website

Comment

Submit Comment

© 2014 Secgroup Ca' Foscari DSI