[2011-12] Monitor

Ci sono `numAgenti` agenti che si muovono nello 'spazio' di dimesione `x,y`. Inizialmente sono posizionati in **ordine inverso** nella prima riga:

```
.9.8.7.6.5.4.3.2.1.0.
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
```

Ogni agente si muove di una posizione (anche in diagonale) per raggiungere lo stato finale:

```
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
.0.1.2.3.4.5.6.7.8.9.
```

La gestione degli spostamenti avviene tramite un opportuno monitor `Spazio` con i seguenti metodi:

- `void register(int n, int x, int y)`: 'registra' l'agente n nella posizione x,y. Questo metodo viene invocato una sola volta quando gli agenti vengono creati. Serve per inizializzare lo spazio nella configurazione iniziale.
- `boolean move(int n, int x, int y, int dx, int dy)`: muove l'agente n dalla posizione `x,y` alla posizione `x+dx, y+dy`. Il valori `dx` e `dy` sono nel range [-1,1] in quanto gli agenti si spostano di una sola posizione. Se la posizione è occupata l'agente attende. Il metodo ritorna false nel caso l'agente n non sia nella posizione `x,y`.
- `int getAgent(int x, int y)`: ritorna l'id dell'agente nella posizione `x,y`. Se la posizione è vuota ritorna -1. Questo metodo viene usato per stampare la situazione ed eseguire test.

Scopo della prova è implementare il monitor 'Spazio' come una classe Java il cui costruttore prende in input le dimenzioni x,y dello spazio di gioco (es. `Spazio(10,10)`) e con i tre metodi sopra descritti. Utilizzare il **programma di test** riportato qui sotto.

Programma di test

```
1  import java.util.HashSet;
2  import java.util.Set;
3
4  public class Test extends Thread {
5    private static final int numAgenti =10;    // numero agent
6    private static final int x=10, y=10;       // dimensione s
7
8    private final int num;     // id dell'agente
9    private final Spazio s;    // monitor spazio di gioco
10   private int my_x,my_y;
```

```
11
12    // costruttore: salva id, monitor e posizione iniziale de
13    Test(int num, Spazio s) {
14      this.num = num;
15      this.s = s;
16      this.my_x = numAgenti -1 - num;
17      this.my_y = 0;
18    }
19
20    public void run() {
21      try {
22        code();
23      } catch (InterruptedException e) {
24        System.out.println("Agente numero "+num+" interrotto!
25      }
26    }
27
28    // codice dei thread
29    void code() throws InterruptedException {
30      int dx, dy, i, j, a;
31
32      if (num == numAgenti) {
33          // questo thread stampa solo la situazione e contro
34          // vedere nel ramo 'else' per il codice degli agent
35          boolean done = false;
36          Set <Integer>check = new HashSet <Integer>();
37
38          // attende che tutti gli agenti siano registrati
39          sleep(1000);
40
41          // controlla la registrazione
42          for (i=0;i<x;i++)
43              if (s.getAgent(i,0) != numAgenti-1-i) {
44                  System.out.println("Errore: l'agente " + i
45                  System.exit(1);
46              }
47
48          // stampa e controlla la situazione ogni secondo
49          while(!done){
50              // stampa
51              check.clear(); // svuota l'insieme di id
52              synchronized(s) {
53                  System.out.println("====");
54                  for (j=0;j<y;j++) {
55                      for (i=0;i<x;i++) {
56                          a = s.getAgent(i,j);
57                          if (a == -1)
58                              System.out.print(". ");
59                          else {
60                              System.out.print("."+a);
61                    if (check.contains(a)) {
62                      System.out.println("Errore: l'agente " + a
63                      System.exit(1);
```

```
64                  } else // lo aggiungiamo
65                                  check.add(a);
66                              }
67                          }
68                          System.out.println(".");
69                          // se tutti gli agenti sono sull'ultima
70                          if (j+2 == y && check.isEmpty())
71                              done = true;
72                      }
73                  }
74
75                  // controlla che non ci siano overlap: tutti gl
76                  if (check.size() != numAgenti) {
77                      // manca qualche agente!
78                      System.out.println("Errore: sono presenti s
79                      System.exit(1);
80                  }
81
82                  // se tutti gli agenti sono sull'ultima riga co
83                  // giusto
84                  if (done) {
85                      for (i=0;i<x;i++)
86                          if (s.getAgent(i,y-1) != i) {
87                              System.out.println("Errore: l'agent
88                              System.exit(1);
89                          }
90                  } else
91                      sleep(1000);
92              }
93              // se non siamo usciti prima il test e' superato
94              System.out.println("Tutti gli agenti sono posiziona
95
96          } else {
97              // questo sono gli agenti
98
99              // si registra
100             s.register(num,my_x,my_y);
101
102             // il giocatore e' pronto attende che tutti si regi
103             System.out.println("Agente numero "+num+" registrat
104             sleep(500);
105
106             // qui avvengono le mosse
107             while (my_y != y-1) {
108                 sleep(1000); // si muovono tutti assieme ogni s
109
110                 // calcola la mossa
111                 dy = 1; // scende sempre di una posizione
112                 if (num == my_x)
113                     dx = 0; // posizione giusta, non si muove c
114                 else {
115                     dx = (num - my_x) / Math.abs(num - my_x); /
116                 }
```

```
117
118                 // prova a fare la mossa
119                 if (!s.move(num,my_x,my_y,dx,dy)) {
120                     System.out.println("Agente numero "+num+ ":
121                     System.exit(1);
122                 }
123
124                 // aggiorna la posizione
125                 my_x += dx;
126                 my_y += dy;
127
128             }
129
130           }
131       }
132
133       public static void main(String argv[]) throws Interrupt
134           int j;
135           Spazio s = new Spazio(x,y); // crea il monitor
136
137           // crea i thread dei vari colori/numeri
138           for (j=0; j<=numAgenti; j++) {
139               (new Test(j,s)).start();
140             }
141
142       }
143   }
```

Altri esercizi

- Gioco di squadra;
- Asta elettronica;
- Cassiere del bar.

Comments: 7

Leave a reply »

Andrea Baesso
March 25th, 2013 at 17:05

Ho provato una soluzione per l'esercizio in questo modo:
(Non ho fatto nessun controllo sul fatto che si potessero muovere per più di una posizione ma ho fatto si che se mi trovo nella posizione 9,9 con 1,1 finisco in 0,0 anche se non so quanto sia giusto

```
1  public class Spazio {
2      private int spazio[][];//matrice per le posizioni
3      private int righe;//variabile dove salvo il numero di ri
4      private int colonne;//variabile dove salvo il numero di
```

```
5       public Spazio(int x, int y){
6           int i,j;
7
8           this.spazio=new int[x][y];//inizializzo la matrice
9           this.righe=x;//salvo le righe
10          this.colonne=y;//salvo le colonne
11          for(i=0;i<x;i++){
12              for(j=0;j<y;j++){
13                  this.spazio[i][j]=-1;//pongo tutti gli eleme
14              }
15          }
16      }
17      void register(int n, int x, int y){
18       //'registra' l'agente n nella posizione x,y.
19       // Questo metodo viene invocato una sola volta quando g
20       // Serve per inizializzare lo spazio nella configuraziо
21          this.spazio[x][y]=n;
22      }
23
24      synchronized boolean  move(int n, int x, int y, int dx,
25          // Il metodo è synchronized ovvero dentro un mutex,
26          // x,y a x+dx,y+dy. Se la posizione è occupata atter
27          // ciclo while, quando è possibile muovere eseguo ur
28          // ritorna false se l'agente n non si trova verament
29          //
30
31          if( this.getAgent(x, y)!=n)
32              return false;
33          else{
34              while(this.getAgent((x+dx)%this.righe, (y+dy)%th
35                  try {
36                  wait();
37              } catch (InterruptedException ex) {
38                  System.out.println("ERRORE WAIT ");
39              }
40              notify();
41              this.spazio[x][y]=-1;
42              this.spazio[(x+dx)%this.righe][(y+dy)%this.color
43              return true;
44          }
45
46      }
47
48      int getAgent(int x, int y){
49          /*Ritorna l'id dell'agente nella posizione x,y, -1 s
50          return this.spazio[x][y];
51      }
52  }
```

Loris
March 26th, 2013 at 23:37

La mia soluzione.
Speriamo che anche l' esame sia così!

```
1  public class Spazio{
2
3      int[][] tabella;
4
5      public Spazio(int x, int y){
6          int i,j;
7          tabella=new int[x][y];
8          /* Inizializzo le celle a -1*/
9          for(i=0;i<x;i++)
10             for(j=0;j<y;j++)
11                 tabella[i][j] = -1;
12     }
13
14     synchronized void register(int n, int x, int y){
15         tabella[x][y] = n;
16     }
17
18     synchronized boolean move(int n, int x, int y, int dx, i
19         if(n != tabella[x][y])
20             return false; // l' agente non è nella poszione
21         while(tabella[x+dx][y+dy] != -1)
22             wait();
23         tabella[x][y] = -1;
24         tabella[x+dx][y+dy] = n;
25         notifyAll();
26         return true;
27     }
28
29     synchronized int getAgent(int x, int y){
30         return tabella[x][y]; // Posso restituire direttamer
31     }
32 }
```

riccardo
March 31st, 2013 at 20:22

@Andrea: OK non serviva che chiudessi il gioco modulo righe,colonne. Gli agenti non vanno mai fuori dai bordi

@Loris: OK ma commenta! Prendete l'abitudine di commentare sempre il codice via via che lo scrivete.

Gabriele Volpato
April 4th, 2013 at 22:12

Dovrebbe essere tutto ok, sembra funzionare bene

```
1  public class Spazio {
2      int matrice[][]; //matrice delle posizioni
3
4      public Spazio(int x, int y){
5          matrice = new int[x][y];
6          for (int i = 0; i < x; i++) {
7              for (int j = 0; j < y; j++) { //inizializzo tutt
8                  matrice[i][j] = -1;
9              }
10         }
11     }
12     /** 'Registra' l'agente n nella posizione x,y.
13      *   Questo metodo viene invocato una sola volta quando
14      *   Serve per inizializzare lo spazio nella configurazi
15      */
16     void register(int n, int x, int y){
17         this.matrice[x][y] = n;
18     }
19     /** Muove l'agente n dalla posizione x,y alla posizione
20      *  Il valori dx e dy sono nel range [-1,1] in quanto gl
21      *  Se la posizione è occupata l'agente attende. Il metc
22      * @throws InterruptedException
23      */
24     synchronized boolean move(int n, int x, int y, int dx, i
25         if(dx < -1 || dx > 1 || dy < -1 || dy > 1 || matrice
26             return false;  //se il passo è troppo lungo o l'
27         }
28         while(matrice[x+dx][y+dy] != -1){ //finché la casell
29             wait();
30         }
31         matrice[x+dx][y+dy] = n; //mi sposto sulla nuova cas
32         matrice[x][y] = -1; //svuoto la casella precedente
33         notify();
34         return true;
35     }
36
37     /** Ritorna l'id dell'agente nella posizione x,y.
38      *  Se la posizione è vuota ritorna -1.
39      *   Questo metodo viene usato per stampare la situazior
40      */
41
42     int getAgent(int x, int y){
43         return this.matrice[x][y]; //mi basta ritornare la c
44     }
45
46 }
```

Roberta Prendin
April 4th, 2013 at 22:24

Ecco anche la mia soluzione!

```
1  public class Spazio {
2
3      /** Campi:
4       * - board è lo spazio dove si muovono gli agenti
5       * - sx e sy sono le dimensioni della board
6       */
7      private int sx;
8      private int sy;
9      private int[][] board;
10
11     /** Costruttore:
12      * - La board è di dimensioni date, x e y;
13      * - la board è inizialmente tutta a -1 (non c'è nessun
14      */
15     public Spazio(int x, int y) {
16         this.sx = x;
17         this.sy = y;
18         board = new int[x][y];
19         for (int i = 0; i < sx; i++) {
20             for (int j = 0; j < sy; j++) {
21                 board[i][j] = -1;
22             }
23         }
24     }
25
26     // Metodi */
27
28     /** Register: per registrare l'esistenza degli agenti su
29     public synchronized void register(int n, int x, int y) {
30         board[x][y] = n;
31     }
32
33     /**
34      * Move: muove l'agente n dalla posizione x,y alla posiz
35      * - I valori dx e dy sono nel range [-1,1] in quanto gl
36      * una sola posizione. - Se la posizione è occupata l'ag
37      * - Il metodo ritorna false nel caso l'agente n non sia
38      */
39
40     public synchronized boolean move(int n, int x, int y, in
41         /**
42          * Condizione di blocco:
43          * - Se la posizione d'arrivo non è libera (c'è un a
44          */
45         while (board[x + dx][y + dy] >= 0) {
46             wait();
47         }
48         // Quando la posizione d'arrivo si libera...
49         board[x + dx][y + dy] = n; // ...riposiziono l'agent
50         if (board[x][y] == n) { //se l'agente era presente n
51             board[x][y] = -1; //...lo "cancello" dalla boarc
52             notifyAll(); //..e notifico gli altri thread
53             return true;
```

```
54              } else {
55                  return false; //ritorno false se l'agente era ne
56              }
57          }
58
59          /**
60           * getAgent() ritorna l'id dell'agente nella posizione x
61           * è vuota ritorna -1. Questo metodo viene usato per sta
62           * ed eseguire test.
63           */
64          public synchronized int getAgent(int x, int y) {
65              int val;
66              if (board[x][y] < 0) {
67                  val = -1;
68              } else
69                  val = board[x][y];
70              return val;
71          }
72      }
```

Lorenzo Bordoni
April 5th, 2013 at 11:43

La mia soluzione è molto simile alle vostre, ho usato però notify() anziché notifyAll() come ha fatto anche Andrea. In questo caso quale dei due è più corretto?

```
1   public class Spazio {
2       private int[][] matrix; // matrice che rappresenta lo sp
3
4       public Spazio(int w, int h) {
5           this.matrix = new int[w][h]; // crea una matrice w x
6           for(int i = 0; i < w; i ++)
7               for(int j = 0; j < h; j ++)
8                   matrix[i][j] = -1; // inizializza tutte le c
9       }
10
11      /*
12       * 'registra' l'agente n nella posizione x,y. Questo met
13       * invocato una sola volta quando gli agenti vengono cre
14       * per inizializzare lo spazio nella configurazione iniz
15       */
16      public synchronized void register(int n, int x, int y) {
17          this.matrix[x][y] = n; // l'agente n occupa la cella
18      }
19
20      /*
21       * muove l'agente n dalla posizione x,y alla posizione x
22       * Il valori dx e dy sono nel range [-1,1] in quanto gli
23       * spostano di una sola posizione. Se la posizione è occ
24       * l'agente attende. Il metodo ritorna false nel caso l'
25       * non sia nella posizione x,y.
```

```
26          */
27         public synchronized boolean move(int n, int x, int y, ir
28             if(this.matrix[x][y] != n) // se l'agente non si trc
29                 return false; // operazione fallita
30             else {
31                 while(this.matrix[x + dx][y + dy] != -1) { // fi
32                     try {
33                         this.wait(); // l'agente aspetta
34                     }
35                     catch(InterruptedException e) {
36                         System.err.println("InterruptedExceptior
37                     }
38                 }
39                 this.matrix[x + dx][y + dy] = n; // sposta l'age
40                 this.matrix[x][y] = -1; // libera la posizione p
41                 this.notify(); // avvisa un'altro agente che puc
42                 return true; // operazione riuscita
43             }
44         }
45 
46         /*
47          * ritorna l'id dell'agente nella posizione x,y. Se la p
48          * vuota ritorna -1. Questo metodo viene usato per stamp
49          * situazione ed eseguire test.
50          */
51         public synchronized int getAgent(int x, int y) {
52             return this.matrix[x][y]; // restituisce il valore c
53         }
54     }
```

riccardo
April 8th, 2013 at 00:55

@Roberta: la verifica della presenza dell'agente è meglio farla prima del wait. Inutile attendere per poi ritornare false

@Gabriele: OK

@Lorenzo: meglio usare notifyAll quando hai thread che attendono eventi differenti. notify potrebbe sbloccare un thread che si riblocca e magari altri che potrebbero procedere attendono inutilmente. Di fatto in java usi spesso notifyAll avendo un'unica coda di attesa. Usi notify quando in effetti è indifferente quale thread viene sbloccato (coda in cui i thread attendono tutti lo stesso evento).

Leave a Reply

Name *

Mail * (will not be published)

Website

Comment

Submit Comment