Esercizio su thread java e synchronized

6 Maggio 2016

Problema

In un ristorante vi sono 2 bagni: uno per gli uomini e l'altro per le donne.

Utilizzando Java, si realizzi un'applicazione concorrente nella quale ogni utente dei bagni(uomo o donna) è rappresentato da un thread ed ogni bagno come una risorsa comune.

Poiché ogni toilette può accogliere una persona alla volta, la soluzione dovrà garantire che ogni bagno possa essere acceduto da **un solo thread alla volta**.

Impostazione

1. Quali thread?
2. Quali risorse?
3. Qual è la struttura di ogni thread?
4. Definizione delle risorse (dati e metodi)
5. Definizione del programma concorrente

Impostazione

1. Quali thread?
   - uomini
   - Donne
2. Quali risorse?

I 2 Bagni:

- Toilet per gli Uomini: **TU**
- Toilet per le Donne: **TD**

Soluzione

3. Qual è la struttura di ogni thread?

Uomo:

```
…
TU.usa_bagno();
…
```

Donna:

```
…
TD.usa_bagno();
…
```

5

Thread Uomo

```
import java.util.Random;
public class Uomo extends Thread{
      Toilet t;
      Random r; //durata uso bagno
      public  Uomo(Toilet T, Random R){
            this.t=T;
            this.r=R;
      }
      public void run(){
            try{

      Thread.sleep(r.nextInt(5)*1000);
                        t.usa_bagno(getName(), r);

                  }catch(InterruptedException e){}
      } //chiude run
}
```

Thread Donna

```
import java.util.Random;
public class Donna extends Thread{
      Toilet t;
      Random r;
      public  Donna(Toilet T, Random R){
            this.t=T;
            this.r=R;
      }
      public void run(){

            try{

      Thread.sleep(r.nextInt(5)*1000);
                        t.usa_bagno(getName(), r);
            }catch(InterruptedException e){}
      } //chiude run
}
```

Risorse

4. Definizione delle risorse :

```
import java.util.Random;
public class Toilet {
      int visitatori;
      String tipo; //donna o uomo

      public  Toilet(String t)
      {     this.visitatori=0;
            this.tipo=t;

      }
```

```
public synchronized void usa_bagno(String TID, Random r)
{  visitatori++;
   System.out.print("\nIl Thread "+TID+" è entrato nel
  bagno per "+tipo+"  [visitatore n. "+ visitatori +"]
  \n");
     try{
         Thread.sleep(r.nextInt(5)*1000);
     }catch(InterruptedException e){}
     System.out.print("\nIl Thread "+TID+" sta uscendo dal
  bagno per "+tipo+" \n");
  }

}// fine classe Toilet
```

Definizione main

```
public class bagno_synchronized {

  public static void main(String[] args) {
      final int NT=10;
      int i;
      Random r=new Random(System.currentTimeMillis());
      Toilet tu=new Toilet("UOMO"); // bagno uomini
      Toilet td=new Toilet("DONNA"); // bagno donne
      Uomo []U= new Uomo[NT];
      Donna []D= new Donna[NT];
      for (i=0; i<NT; i++)
      {     U[i]=new Uomo(tu, r);
            D[i]=new Donna(td, r);
      }
      for (i=0; i<NT; i++)
      {     U[i].start();
            D[i].start();
      }
  }   }
```

Soluzione alternativa

- la soluzione vista usa 2 istanze della classe Toilet.
- in alternativa, si potrebbe pensare a una soluzione con un unico oggetto condiviso che rappresenta la toilette, suddiviso internamente in bagno uomo e bagno donna:

```
public class Toilet {
        int visitatoriU;
        int visitatoriD;

public void usa_bagnoU(String TID, Random r)
{...}

public void usa_bagnoD(String TID, Random r)
{ ... } }
```

I due metodi devono essere realizzati in modo tale da:

- impedire chiamate concorrenti dello stesso metodo.
- consentire chiamate concorrenti di metodi diversi

COME FARE?