Esercitazione #5 -- Corso di Sistemi Operativi

Sincronizzazione in Java

(Monitor con lock e variabili condizione)

Luca Gherardi e Patrizia Scandurra – a.a. 2012-13

1. Si consideri il seguente problema di sincronizzazione. In un *circolo enoculturale*, np persone appassionate di vino si riuniscono per una degustazione. E' presente una botte di vino pregiato di cui si desidera valutarne la qualità. La botte ha nr=3 rubinetti e contiene inizialmente nl=20 litri di vino; di conseguenza possono bere solo nr persone alla volta. Si supponga che ogni bevuta (da un singolo rubinetto) faccia diminuire il vino in botte di una quantità q variabile, ma inferiore a 0.3 litri; se la quantità q è maggiore della disponibilità di vino presente nella botte, viene prelevata solo la quantità di vino effettivamente presente (svuotando così la botte).

Si utilizzino i meccanismi di sincronizzazione dei *lock* e delle *variabili condizione* per realizzare un *monitor* (classe `BotteSinc`). La classe `BotteSinc` deve definire i metodi `occupaRubinetto` e bevi, i quali vengono invocati dal generico thread "bevitore" (la classe `Bevitore`). Tali metodi consentano ad un bevitore, rispettivamente, di accedere ad uno dei rubinetti e di prelevare una certa quantità di vino.

Il comportamento di un bevitore è il seguente:

- chiacchiera un po';
- decide di bere e si mette in attesa se non ci sono rubinetti liberi;
- una volta acquisito l'accesso esclusivo ad un rubinetto attende un periodo di tempo casuale e decide una certa quantità q di vino da bere (anch'essa casuale)
- preleva dalla botte la quantità q
- libera il rubinetto e ricomincia di nuovo.

Qualora non ci fosse più vino, il bevitore termina la sua esecuzione.

```
//Classe "monitor" dell'oggetto "botte"
import java.util.concurrent.locks.*;
public class BotteSinc {

  // variabili private necessarie per l'implementazione

  // lock e condizioni

  // costruttore

  // metodo che restitiuscie I litri restanti
  public double getLitri()

  // metodo per occupare un rubinetto
  public void occupaRubinetto(String id)

  // metodo per ber
  public void bevi(String id, double quantity)


 }
```

2. Un ristorante ha una toilette unica per uomini e donne ed ha una capacità limitata a N persone. Si assuma che ogni utente della toilette (uomo o donna) sia rappresentato in Java da un thread e che il bagno (un oggetto Java) sia la risorsa condivisa.

Utilizzando il linguaggio Java, si modellino gli utenti della toilette come thread e si definisca la classe `Toilette` (l'oggetto condiviso!) realizzando così uno schema di sincronizzazione tra i thread basato sul concetto di monitor che garantisce i seguenti vincoli:

- nella toilette non possono esserci contemporaneamente uomini e donne
- nell'accesso alla toilette, le donne hanno la priorità sugli uomini.

Tali condizioni di sincronizzazione devono riflettersi nel codice della classe `Toilette` nel modo seguente.

```
public class Toilette {
private int N; //capacità del bagno
private int donne_in; /* numero di donne in toilette*/
… <DA COMPLETARE>

public Toilette(int N){ //Inizializzazione
  this.N=N;
 … <DA COMPLETARE> }

public void donna_esce()  { /*Provo ad acquisire il lock sulla risorsa Toilette per uscire.
Acquisito il lock, aggiorno lo stato del bagno per indicare che ci sarà una donna in meno, e poi
prima di uscire dal bagno e rilasciare il lock: se ci sono donne che aspettano risveglio una
donna sospesa, altrimenti se non ci sono più donne risveglio tutti gli uomini. */
    <DA COMPLETARE> }

public void uomo_esce() {
/*Similmente alla donna, ma prima di uscire controlla: se ci sono donne in attesa di entrare
e se nel bagno non ci sono uomini, allora risveglio tutte le donne sospese; altrimenti, se non
ci sono donne che aspettano e ci sono, invece, uomini che aspettano, allora risveglio un uomo.*/
<DA COMPLETARE> }

public void donna_entra(){
/*Fintanto che  la toilette è piena o ci sono uomini in bagno, il thread donna si sospende*/
<DA COMPLETARE> }

public void uomo_entra() {
/*Fintanto che la toilette è piena o ci sono donne in bagno o ci sono donne in attesa di
entrare, il thread uomo si sospende */
<DA COMPLETARE> }
}//End class
```