Esercitazione #4 -- Corso di Sistemi Operativi

Sincronizzazione in Java

(Monitor con lock e variabili condizione)

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

1. **CokeMachine**. Usando i *lock* e le *variabili condizione* del package java.util.concurrent come meccanismo di sincronizzazione, si realizzi in Java una soluzione al problema di prelevare lattine di coca-cola da una macchinetta e di rifornirla nel caso in cui rimanga vuota. Definire le classi per i thread utenti e il thread rifornitore. Definire, inoltre, la classe CokeMachine contenente le lattine di coca-cola (oggetti condivisi!) ed i metodi:
   - remove(…), eseguito dal generico utente per prelevare una lattina dalla macchinetta;
   - deposit(…), eseguito dal fornitore del servizio per caricare la macchinetta bel caso in cui rimane vuota.

   Si assuma che inizialmente la macchinetta è piena, e che un utente (*a scelta*: il primo a trovare la macchinetta vuota o l'utente che preleva l'ultima lattina) può segnalare al fornitore che la macchinetta è vuota.
   Lo scheletro della classe CokeMachine è riportato di seguito come piccolo aiuto.

```
import java.util.concurrent.locks.*;
class CokeMachine{
 static final int N = 50; //Capacità della macchinetta
 int count ; //Numero effettivo di lattine presenti nella macchinetta
 final Lock lock = new ReentrantLock(); //Lock per la mutua esclusione
  …. //Condition variable
  …. <COMPLETARE>….
  }
```

2. **BAR**. In uno stadio è presente un unico bar a disposizione di tutti i tifosi che assistono alle partite di calcio. I tifosi sono suddivisi in due categorie: **tifosi della squadra locale**, e **tifosi della squadra ospite**. Il bar ha una capacità massima NMAX, che esprime il numero massimo di persone (tifosi) che il bar può accogliere contemporaneamente. I tifosi si recano al bar una sola volta durante tutto il tempo della partita.

   Per motivi di sicurezza, nel bar non è consentita la presenza contemporanea di tifosi di squadre opposte.
   Il bar è gestito da un **barista** che può decidere di chiudere il bar in qualunque momento per effettuare la pulizia del locale. Al termine dell'attività di pulizia, il bar verrà riaperto.

   Durante il periodo di chiusura non è consentito l'accesso al bar a nessun cliente. Nella fase di chiusura, potrebbero essere presenti alcune persone nel bar: in questo caso il barista attenderà l'uscita delle persone presenti nel bar, prima di procedere alla pulizia.

   Ultilizzando il linguaggio Java, si modellino i clienti e il barista mediante thread concorrenti (usando le classi separate *ClienteLocale*, *ClienteOspite* e *Barista*) e si realizzi una politica di sincronizzazione tra i thread basata sul concetto di monitor (con lock e variabili condizione) che tenga conto dei vincoli dati, e che inoltre, nell'accesso al bar dia la precedenza ai tifosi della squadra ospite.

   [**Suggerimento**: Definire la classe *Bar*, che rappresenta il monitor e prevedere in essa: (i) i metodi per l'inizio della chiusura e la fine della chiusura invocati dal thread barista; (ii) i metodi per l'ingresso e l'uscita dal bar dei tifosi ospiti; (iii) i metodi per l'ingresso e l'uscita dal bar dei tifosi locali.]