

1 Esercizio

In questo esercizio, vi chiediamo di usare la classe `Semaphore` di JAVA che propone una implementazione dei semafori. Potete guardare online la documentazione di questa classe e in particolari le tre funzione seguente che rappresentano l'uso dei semafori visto a lezione:

- il costruttore `Semaphore(int permits)` per creare un nuovo semaforo inizializzato a `permits`;
- la funzione `void acquire()` che corrisponde al `wait` visto a lezione;
- la funzione `void release()` che corrisponde al `signal` visto a lezione.

Consideriamo una piscina pubblica con N_S spogliatoi individuali e N_C armadietti per lasciare i suoi vestiti. Un cliente che si presenta alla piscina realizza le tappe seguente:

- Prende la chiave di uno spogliatoio;
- Prende la chiave di un armadietto;
- Si cambia nello spogliatoio;
- Libera lo spogliatoio;
- Mette i suoi vestiti nel armadietto;
- Rida la chiave dello spogliatoio;
- Nuota (tenendosi la chiave del armadietto);
- Prende la chiave di un spogliatoio;
- Ricupera i suoi vestiti nel armadietto;
- Si riveste nello spogliatoio;
- Libera lo spogliatoio;
- Rida le chiave dello spogliatoio e del armadietto.

Lo scopo del esercizio è di proporre un implementazione in JAVA di un programma che simula il comportamento dei clienti in questa piscina.

- Creare un programma JAVA che simula tanti clienti concorrenti. La disponibilità degli spogliatoi e degli armadietti sarà rappresentata da due semafori che saranno condivisi fra gli diversi thread che rappresentarono i clienti.
- Testare il vostro programma con diversi numeri di clienti, di spogliatoi e di armadietti. In particolare, provate a trovare casi in cui c'è un deadlock (i.e. tutti i clienti sono bloccati).
- Proponete un cambio minore nel comportamento dei clienti per evitare questo deadlock.

2 Consegna

Per la consegna, creare uno `zip` con tutti vostri file. Lo `zip` dovrà anche contenere un file `partecipanti.txt` dove gli nomi di chi ha partecipato alla consegna (questo anche se siete da solo a farla).