PCAD 23/24 - Laboratorio 3

Negli esercizi seguenti proponiamo di programmare il problema dei lettori/scrittori in Java. Per ciascuno degli esercizi, provare ad osservare il comportamento del programma ottenendo diversi set di esecuzione (potete addormentare i thread per un attimo grazie alla funzione statica `sleep` della classe `Thread`).

1 Esercizio 1

In una prima versione, ci interessiamo solo alla concorrenza senza sincronizzazione ne mutua esclusione.

1. Scrivere una classe `RWbasic` contenendo un campo intero privato `data` inizializzato a 0 al quale l'accesso è fatto senza mutua esclusione da due funzione:
   - `read` chi ritorna il valore di `data`, e,
   - `write` chi aumenta di 1 il valore di `data` in vari fasi:
     (a) mette il valore di `data` in una variabile temporanea `tmp`;
     (b) aumenta di 1 il valore di `tmp`;
     (c) mette il valore di `tmp` in `data`.
2. Scrivere una classe `Reader` che implementa l'interfaccia `Runnable`. Il costruttore di questa classe riceve come argomento un oggetto della classe `RWbasic`. Nel suo codice eseguibile (funzione `run`), chiama la funzione `read` associata e stampa il valore della data.
3. Scrivere una classe `Writer` che implementa l'interfaccia `Runnable`. Il costruttore di questa classe riceve come argomento un oggetto della classe `RWbasic`. Nel suo codice eseguibile (funzione `run`), chiama la funzione `write` associata.
4. Scrivere un programma principale che crea un oggetto di classe `RWbasic` e crea un certo numero di thread reader e writer (ad esempio 50 di ciascuno). Osservare, ad esempio mettendo i thread in `sleep` in mezzo alla scrittura, che al termine dell'esecuzione il valore del data non è uguale al numero di writers (non esitate ad utilizzare la funzione `join` della classe Thread per aspettare la fine di un thread). Non esitare a dare un identità ai thread per potere capire meglio l'esecuzione ottenuta.

2 Esercizio 2

In una seconda versione, ci interessiamo alla mutua esclusione.

1. Scrivere una classe `RWexclusive` che estende `RWbasic` rendendo esclusivo l'accesso alla data.
2. Modificare il codice gia scritto delle classe `Reader`, `Writer` e del programma principale perché usano la classe `RWexclusive` al posto di `RWbasic`.
3. Osservare il comportamento del programma come nel esercizio precedente. Il problema sollevato dovrebbe essere risolto.

3 Esercizio 3

La terza versione che vi chiediamo è una versione dove più lettori possono accedere alla data allo stesso tempo ma invece i scrittori devono essere in mutua esclusione e se dei lettori stanno legendo la data, i scrittori non possono accederci.

1. Scrivere una classe `RW` che estende `RWbasic` e verifica le proprietà date qui sopra. Per fare questo, potete usare un contattore che conta i lettori e dividere la funzione `read` in varie sotto funzione. In più è consigliata di usare `wait` e `notify` per svegliare gli scrittori che aspettano la fine della lettura dei lettori.
2. Osservare il comportamento del programma modificato.

4 Esercizio 4

Modificare il codice precedente con una classe `RWext` che garantisce che ogni valore scritta e letta da al meno un lettore.

5 Consegna

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