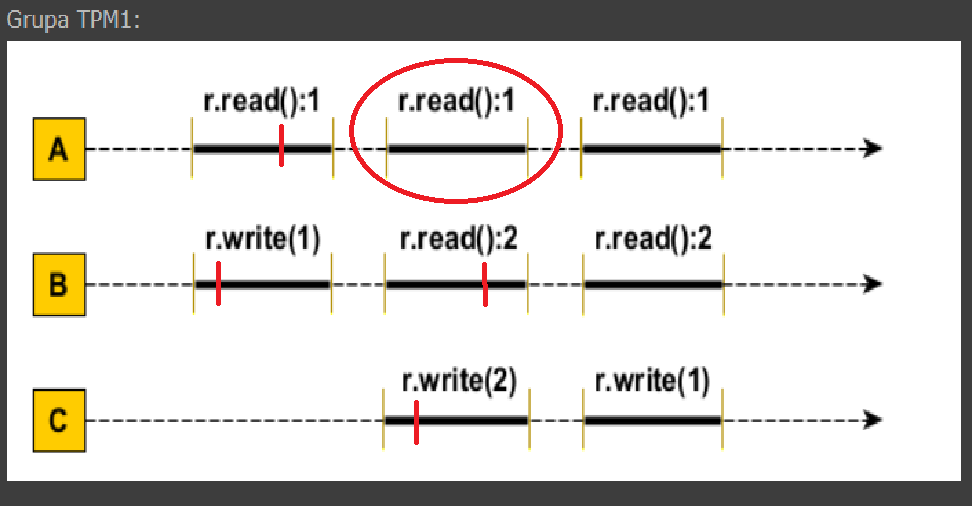
TEMA 1

**ECHIPA:** HORODINCĂ MIHAI (TPM1)

MIRILĂ VASILE DĂNUȚ (TPM1)

1. 

Pentru operatia incercuita nu mai este nici o valoare in variabila r, deci nu are cum sa citeasca valoarea 1. In concluzie nu este linearizabila.

Aceasta secventa nu este nici consistent secventiala deoarece sunt operatii de la thread-uri diferite care se suprapun pe axa temporala.

2. a.

Generalizarea cozii asignată grupei noastre este:

public class LockBasedQueue {

int head = 0, tail = 0;

int items [] = new int[ QSIZE ];

ReentrantLock lock = new ReentrantLock();

public void enq(int x) {

while ( tail - head == QSIZE ) {};

lock.lock();

try {

items [ tail % QSIZE ] = x;

tail ++;

} finally {

lock.unlock();

}

}

public int deq () {

while ( tail == head ) {};

lock.lock();

try {

int item = items [ head % QSIZE ];

head ++;

return item;

} finally {

lock.unlock();

}

}

}

Noi am observat o problemă care poate să apară în acestă abordare. Să presupunem că nu avem niciun element în coadă iar două thread-uri vor să citească. Ele așteaptă în block-ul while până când un alt thread adaugă un elemnt, moment în care thread-urile care așteptau își vor continua rularea. Unul dintre thread-uri primește lock-ul, ia singurul element din coadă și deblochează lock-ul. Acum, deși nu mai avem niciun element in coadă, celălalt thread de citire va primi lock-ul și va citi dintr-o locație invalidă. Același lucru se poate întâmpla și în cazul scrierii (două thread-uri vor să scrie când coada este plină, cineva ia un singur element, ambele thread-uri cred că mai e loc și adaugă prea multe elemente).

Pentru a remedia această situație putem să punem toate comenzile din block-ul try într-un if care verifică faptul că situația încă îmi permite să adaug/scot elemente din coadă.

2. b.

Algoritmul Bakery foloseste o ordine lexicografica a thread-urilor. Aceasta ordine este creata prin compararea tuplelor de forma (label[i], i). Doua thread-uri pot avea aceeasi eticheta (label). Astfel, daca in compararea a doua thread-uri ar fi fost utilizate doar label[i] si label[k], atunci algoritmul nu ar fi capabil sa dea precedenta unui thread.

Exemplu:

Presupunem ca avem vectorul labels: [C, B, O, A, B]. In acest exemplu labels[0] = labels[4], deci nu se pot ordona lexicografic. De aceea algoritmul Bakery foloseste tuplele (labels[0], 0) si (labels[4], 4) pentru a realiza ordonarea. Pentru ca labels[0] = labels[4] se verifica si ordinea indecsilor. 0 < 4, deci (labels[0], 0) < (labels[4], 4). Astfel este imposibil sa nu acordam precedenta unui thread.

2. c.

Este recomandat ca metoda “lock” sa fie apelata inainte de block-ul try-catch pentru ca aceasta metoda poate arunca o exceptie de tip unchecked. Daca s-ar apela in blocul try si ar arunca o exceptie, atunci ar fi rulat codul din finally care ar face unlock, dar nu s-a facut lock pe “someLock”.

3.

|  |  |  |  |
| --- | --- | --- | --- |
| 3 a) Statistici pentru cazul în care fiecare mănâncă doar o dată  (măsurăm timpul în care se termină execuția) | | | |
|  | Nr thread-uri = 4 | Nr thread-uri = 8 | Nr thread-uri = 16 |
| Nr porții = 4 | 3 ms  (aprox. 13333 porții / 10 sec) | 4 ms  (aprox. 10000 porții / 10 sec) | 7 ms  (aprox. 5714 porții / 10 sec) |
| Nr porții = 8 | 2 ms  (aprox. 40000 porții / 10 sec) | 3 ms  (aprox. 13333 porții / 10 sec) | 5 ms  (aprox. 16000 porții / 10 sec) |
| Nr porții = 16 | 3 ms  (aprox. 53333 porții / 10 sec) | 4 ms  (aprox. 40000 porții / 10 sec) | 4 ms  (aprox. 40000 porții / 10 sec) |

|  |  |  |  |
| --- | --- | --- | --- |
| 3 b) Statistici pentru cazul în care fiecare mănâncă de oricâte ori  (în tabel avem notat numarul de porții mancate de thread-uri în 10 secunde) | | | |
|  | Nr thread-uri = 4 | Nr thread-uri = 8 | Nr thread-uri = 16 |
| Nr porții = 4 | 645237  (Min: 213577, Max: 216467) | 131089  (Min: 12609, Max: 25423) | 7112  (Min: 212, Max: 1230) |
| Nr porții = 8 | 876560  (Min: 287220, Max: 296439) | 180312  (Min: 18702, Max: 35266) | 15810  (Min: 417, Max: 1770) |
| Nr porții = 16 | 1010692  (Min: 333802, Max: 341979) | 222541  (Min: 26784, Max: 36358) | 14128  (Min: 363, Max: 1497) |