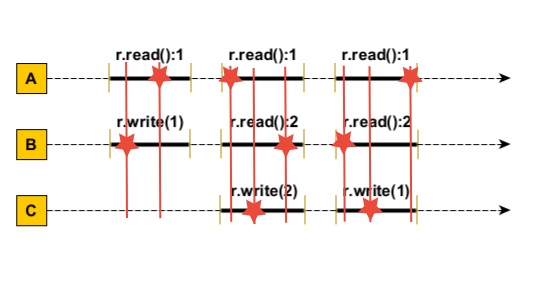
Ex1.



In figura de mai sus putem observa o secventa de puncte de liniarizare care sa asigure executia corecta a programului, astfel secventa este liniarizabila. Prin definitie, secventa este si consistent secventiala, daca este deja liniarizabila.

3a.

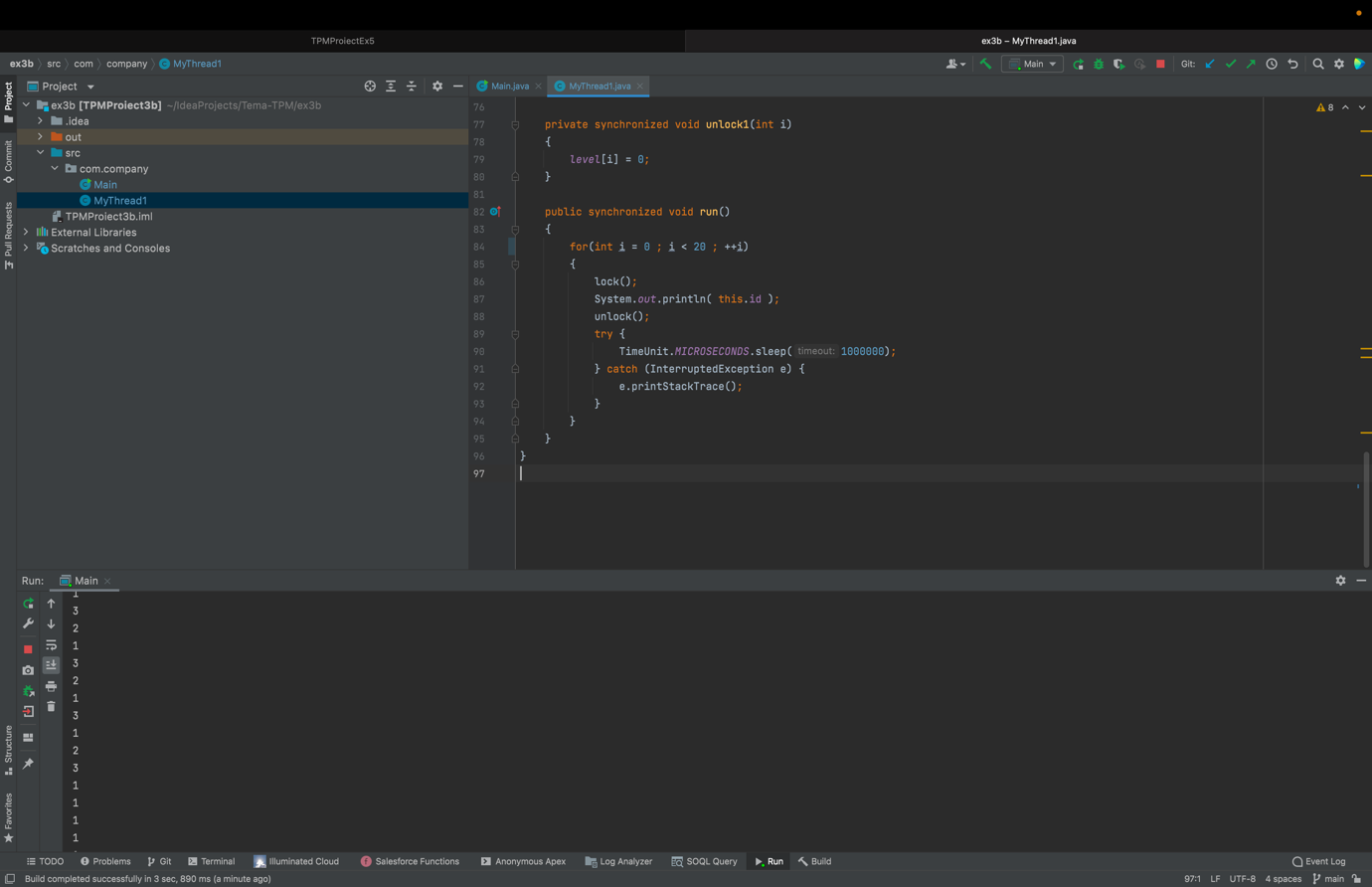
Acest algoritm de implementare a Lock-ului nu este deadlock-free (adică exista cel putin un scenariu în care toate thread-urile implicate rămân blocate în așteptarea de a accesa zona partajata).

Fie următorul scenariu pentru n = 2 fire de execuție concurente.

Thread 1 actualizează variabila volatilă turn = 1 (id unic pozitiv) și iese din bucla do-while nested intrucat used != true, iar după următoarea instrucțiune actualizează used la valoarea true. Între timp Thread 2 intra în bucla nested do-while și actualizează turn = 2. În aceasta situație thread 2 rămâne blocat în bucla nested întrucât used = true și thread 1 rămâne blocat în bucla exterioara do-while întrucât turn != 1 și nu poate astfel apela metoda unlock pentru a elibera Thread 2 din bucla nested.

EX3b

Algoritmul nu este fair, cum este exemplificat in exemplul de mai jos



(accesul threadurilor la sectiunea critica)

Algoritmul nu este fair deoarece un thread poate sa nu intre in if-ul de la linia 10, pe cand alte thread uri asteapta ca lacatul sa fie deblocat. Acest caz are loc in momentul in care acel thread este singurul care trece consecutiv prin liniile 7, 8, 9, 10.

Ex3c

White de maxim n-1 ori: "white" nu poate fi returnat de n ori deoarece pentru a returna "white", este necesara o actualizare a variabilei "getWhite". Threadul care face aceasta modificare va returna fie "black", fie "red", deoarece actualizarea variabilei se face dupa verificarea conditiei pentru ca threadul sa returneze "white". Astfel, cel putin un thread va vedea variabila "getWhite" ca fiind false, deci cel putin un thread nu va returna "white", de aici rezulta maximul de n-1

Red maxim o data: Pentru ca un thread sa returneze "red", trebui mai intai sa treaca de verificarea din if-ul cu "getWhite". Presupunem ca un numar de m threaduri au avut "getWhite" = false si urmeaza sa evalueze conditia "last==me". Dintre aceste m threaduri, conditia va vi adevarata doar pentru maxim 1 thread, deoarece maxim 1 thread poate avea acelasi id ("me") ca variabila "last". Conditia poate fi falsa pentru toate threadurile in cazul in care alt thread intre timp a modificat valoarea variabilei "last". In acel caz, 0 threaduri vor returna "red". Deci fie 0, fie 1 thread va returna "red".

Ex5

Fie un program ce implementeaza algoritmul lui Peterson generalizat, pentru 9 threaduri, si urmatorul exemplu concurent de executie:

Presupunem ca functia lock() a fost apelata de threadul numarul 1 inainte de threadul numarul 2, iar celelalte 7 threaduri inca nu au apelat aceasta functie.

Pentru urmatorul exemplu exemplificat in figura de mai jos, thread-ul 1 a ajuns pe nivelul 5, iar threadul 2 pe nivelul 1.

Level

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 4 | 1 |  |  |  |  |  |  |

Victim

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 2 |  |  | 1 |  |  |  |  |

Dupa algoritmul lui Peterson generalizat, threadul 2 va fi “starved”, pana cand threadul 1 va apela functia unlock(). O posibila solutie pentru a opri fenomenul de “starving” este de a lasa threadurile sa inainteze pe urmatorul nivel, daca nu este ocupat de nici un thread, astfel, pentru exemplul de mai sus, threadul 2 putand ajunge pana maxim pe nivelul 7, daca threadul 8 se afla pe ultimul nivel si inca nu a iesit din sectiunea critica.

Level

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 8 | 7 |  |  |  |  |  |  |

Victim

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|  |  |  |  |  |  | 2 | 1 |

We can observer that starvation is prevented by looking at the significant difference in execution time for only 10 threads.

