Nr. Threaduri Citire	Nr. Threaduri Workers	Timp de executie
4	2	157.34361
	4	139.68665
	12	137.13627
Secvential		347.86934

Observatii:

- Metodele care folosesc 4 threaduri de citire, si 2/ 4/ 12 threaduri workers sunt mult mai eficiente decat metoda secventiala
- Metodele aplicate in laboratorul 5 (Fine grain synchronization) sunt mai eficient decat metodele aplicate la laboratorul 4 (coarse grain synchronization), sunt mai eficiente. (La lb4 am obtinut aprox. 600-700 ms pt 1 Thr de citire, si 100-110 ms pt 2 Thr de citire).
- Cel mai bun timp l-am obtinut folosind 12 threaduri Atunci cand am folosit un singur thread pentru citire, am obtinut cel mai ineficient timp

Detalii de implementare:

MyLinkedList:

- O lista inlantuita cu santinele
- Metodele de update, add si delete se blocheaza accesul concurent la cate 3 noduri. La add, avem de refacut si legaturile, adaugarea realizandu-se la inceputul listei. Nodul de santinela primeste referinta de next la nodul adaugat, iar cel care era primul nod cu informatie inainte de adaugare primeste referinta de previous la nodul adaugat. La update se cauta nodul in lista si se actualizeaza informatia. La delete se cauta nodul in lista si se refac legaturile: nodul anterior celui sters primeste referinta de next la nodul de

dupa cel sters, iar el primeste referinta de previous la nodul anterior celui sters.

- Contine si metoda find, care cauta un nod in lista si returneaza nodul gasit
- Metoda notEmpty verifica daca lista contine noduri cu informatii (pe langa cele de santinela)

MyQueue:

- Tinem evidenta nr de cititori ramasi si avem o dimensiune maxima
- Metoda push adauga un nod in lista. Iar daca coada este deja plina, asteapta un semnal, din metoda pop, adica atunci cand se sterge un nod, si efectueaza adaugarea
- Metoda pop sterge un nod din lista. Daca coada este goala, asteapta un semnal, din metoda push, adica atunci cand a fost adaugat un nou nod, il sterge.