

DOCUMENTAȚIE

TEMA *2*

NUME STUDENT: Vădean Adriana-Maria

GRUPA: 30221



# CUPRINS

[1. Obiectivul temei 3](#_Toc95297885)

[2. Analiza problemei, modelare, scenarii, cazuri de utilizare 3](#_Toc95297886)

[3. Proiectare 3](#_Toc95297887)

[4. Implementare 3](#_Toc95297888)

[5. Rezultate 3](#_Toc95297889)

[6. Concluzii 3](#_Toc95297890)

[7. Bibliografie 3](#_Toc95297891)



# Obiectivul temei

Obiectivul temei este de a crea un program pentru simularea unor cozi, pentru a putea eficientiza ocuparea cozilor într-un mod mult mai eficient de către oameni.

Problema așteptării la coadă este o problemă reală și des întâlnită în zilele de acum, si bineînțeles că acest lucru poate deveni foarte iritant pentru fiecare persoană în parte, iar de aceea, se caută tot felul de soluții pentru a reduce statul la coadă mai mult decât trebuie, doar că această teme fiind doar o simulare, poate doar virtual să rezolve acestă problemă, în speranța că pe mai încolo se va dezvolta această simulare într-o rezolvare reală a acestei probleme, mai ales că tehnologia pare să avanseze din ce în ce mai mult. Niște exemple foarte bune unde timpul este alocat statului la coadă ar fi: bancomatele, unde oamenii vor să efectueze diferite operațiuni, sau casele de marcat, dar orice exemplu ar fi, firea omenească este de a te uita la coada unde merge cel mai rapid, și de a te muta acolo, pentru a termina în timp util.

# Analiza problemei, modelare, scenarii, cazuri de utilizare

Cerința proiectului este aceea de a implementa o aplicație ce sistematizează niște cozi la care sunt adăugați clienții, astfel încât timpul de așteptare să fie minim.

Aplicația de gestionare a cozilor ar trebui să simuleze (prin definirea unui timp de simulare 𝑡𝑠𝑖𝑚𝑢𝑙𝑎𝑡𝑖𝑜𝑛) o serie de N clienți care sosesc pentru a fi serviți, intră în Q cozi, așteaptă, sunt serviți și în cele din urmă pleacă. Toți clienții sunt generați atunci când simularea este pornită și sunt caracterizați de trei parametri: ID (un număr între 1 și N), 𝑡𝑎𝑟𝑟𝑖𝑣𝑎𝑙 (timpul de simulare când sunt gata să intre coada) și 𝑡𝑠𝑒𝑟𝑣𝑖𝑐𝑒 (intervalul de timp sau durata necesară pentru deservirea clientului; adică timpul de așteptare când clientul este în fața cozii). Aplicația urmărește timpul total petrecut de fiecare client pentru a-l putea pune la coadă și calculează timpul mediu de așteptare. Fiecare client este adăugat la coadă cu minimum timpul de așteptare când timpul său 𝑡𝑎𝑟𝑟𝑖𝑣𝑎𝑙 este mai mare sau egal cu timpul de simulare (𝑡𝑎𝑟𝑟𝑖𝑣𝑎𝑙 ≥ 𝑡𝑠𝑖𝑚𝑢𝑙𝑎𝑡𝑖𝑜𝑛). Următoarele date ar trebui considerate date de intrare pentru aplicația care ar trebui introduse de către utilizator în interfața de utilizator a aplicației: - Număr de clienți (N); - Numărul de cozi (Q); - Interval de simulare (𝑡𝑠𝑖𝑚𝑢𝑙𝑎𝑡𝑖𝑜𝑛 𝑀𝐴𝑋 ); - Ora minimă și maximă de sosire (𝑡𝑎𝑟𝑟𝑖𝑣𝑎𝑙 𝑀𝐼𝑁 ≤ 𝑡𝑎𝑟𝑟𝑖𝑣𝑎𝑙 ≤ 𝑡𝑎𝑟𝑟𝑖𝑣𝑎𝑙 𝑀𝐴𝑋 ); - Timp minim și maxim de servire (𝑡𝑠𝑒𝑟𝑣𝑖𝑐𝑒 𝑀𝐼𝑁 ≤ 𝑡𝑠𝑒𝑟𝑣𝑖𝑐𝑒 ≤ 𝑡𝑠𝑒𝑟𝑣𝑖𝑐𝑒 𝑀𝐴𝑋 ).



Utilizatorul ce interacționează cu această aplicație, ar trebui să poată introduce datele de intrare menționate mai sus, să vadă cum timpul tot crește pe parcursul simulării și se oprește când simularea se oprește însemnând că aceasta s-a terminat și toți clienții au fost serviți; să mai poată vedea cum clienții intră și ies din coadă când timpul lor de servire s-a terminat, iar la final, cum tot parcursul simulării a fost scris în 3 fișiere corespunzătoare celor 3 cazuri ce trebuie testate.

Pentru o înțelegere mai bună va fi explicat în continuare primul set de date dat.

|  |  |
| --- | --- |
| Istoricul pentru fiecare iterație a timpului | Explicație |
| 1. Timp: 0  Clienți în așteptare:  (2,2,3)  (1,7,2)  (4,15,4)  (3,30,4)  Coada 1:goală  Coada 2:goală | După cum se poate observa, clienții ce au fiecare câte un id, timpul sosirii și timpul servirii, sunt ordonați crescător după timpul sosirii. Momentan ambele cozi sunt neocupate, deoarece timpul sosirii a clienților nu corespunde cu timpul curent, pentru ca aceștia să poată să meargă la coadă |
| ... |  |
| 2.Timp: 2  Clienți în așteptare:  (1,7,2)  (4,15,4)  (3,30,4)  Coada 1:(2,2,3)  Coada 2:goală | Se poate observa cum a mers la prima coadă primul client ce aștepta mai sus, având timpul sosirii egal cu timpul curent. |
| 3.Timp: 3  Clienți în așteptare:  (1,7,2)  (4,15,4)  (3,30,4)  Coada 1:(2,2,2)  Coada 2:goală | Timpul de servire a clientului tot scade, iar când ajunge la 0, acesta poate pleca din coadă. |
| ... |  |
| 4.Timp: 30  Clienți în așteptare:  Coada 1:(3,30,4)  Coada 2:goală | Aici se poate observa cum a mers la coadă ultimul client ce trebuie servit. |
| 5. Timp: 34  Clienți în așteptare:  Coada 1:goală  Coada 2:goală | Simularea s-a terminat la 34 de secunde, deoarece atunci a plecat ultimul client ce aștepta la coadă. |



# Proiectare

La implementarea aplicație s-a folosit clasa Semafor. Un semafor controlează accesul la o resursă partajată prin utilizarea unui contor. Dacă contorul este mai mare decât zero, atunci accesul este permis. Dacă este zero, atunci accesul este refuzat. Ceea ce numără contorul sunt permise care permit accesul la resursa partajată. Astfel, pentru a accesa resursa, unui fir trebuie să i se acorde permis de la semafor.

În general, pentru a folosi un semafor, firul care dorește acces la resursa partajată încearcă să obțină un permis. Dacă numărul semaforului este mai mare decât zero, atunci firul obține un permis, ceea ce face ca numărul semaforului să fie decrementat. În caz contrar, thread-ul va fi blocat până când se va putea obține un permis. Când firul de execuție nu mai are nevoie de acces la resursa partajată, eliberează permisul, ceea ce face ca numărul semaforului să fie incrementat. Dacă există un alt thread care așteaptă un permis, atunci acel thread va obține un permis în acel moment.

Putem folosi un semafor pentru a bloca accesul la o resursă, fiecare fir care dorește să folosească acea resursă trebuie să apeleze mai întâi acquire( ) înainte de a accesa resursa pentru a obține blocarea. Când firul de execuție este terminat cu resursa, trebuie să apeleze release( ) pentru a elibera blocarea.

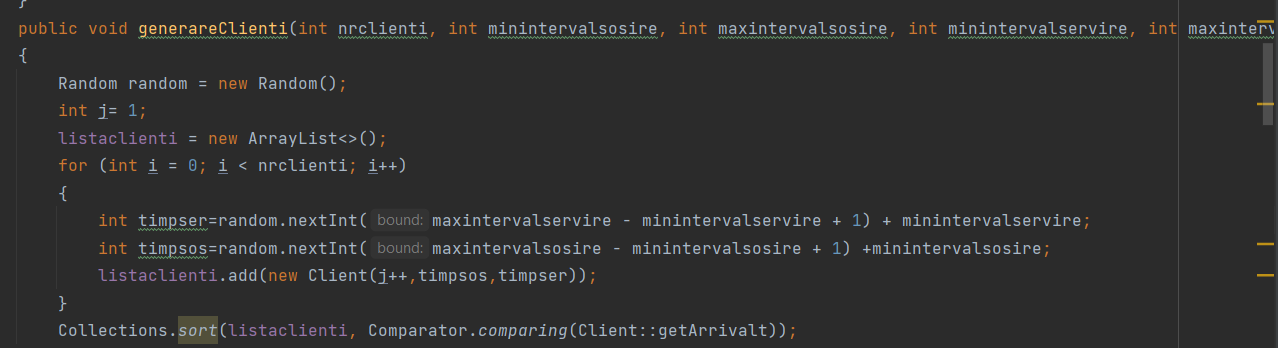
Structurile de date cele mai folosite au fost: List<> pentru lista de clienti și de consumatori din clasa Producător, ArrayList<> folosită în interfața pentru simulare pentru a contura coada și clienții, iar BlockingOueue<> tot așa pentru clienți, doar că în clasa Consumatori

# Implementare

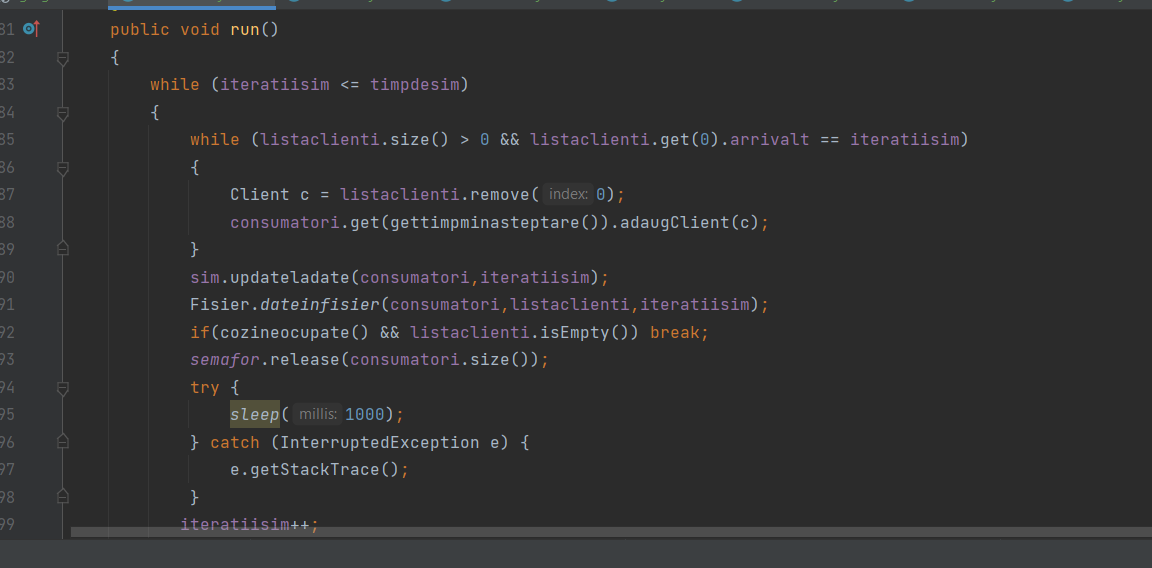
Pentru început, am folosit arhitectura MVC: Model-View-Controller pentru a împărți aplicația în mai multe pachete, pentru a face să fie cât mai ușoară folosirea acesteia de către oricine si pentru o organizare cât mai bună.

În pachetul **model** : se află cele 3 clase principale și anume Producător, Client și Consumator.

* Clasa **Producător** conține cele mai multe metode din acest program.



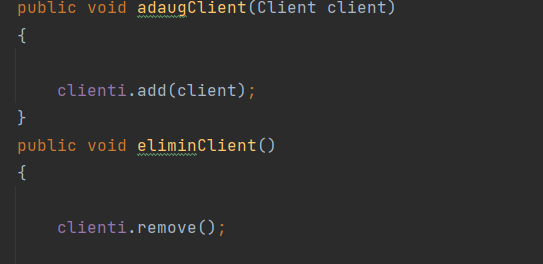
-această metodă generează random, un timp random de sosire și un timp random de servire din intervalul dat de către utilizator în interfața grafică și îi tot adaugă pe clienți în lista listaclienti, iar apoi după închiderea buclei, sortează lista după timpul de sosire al acestora.



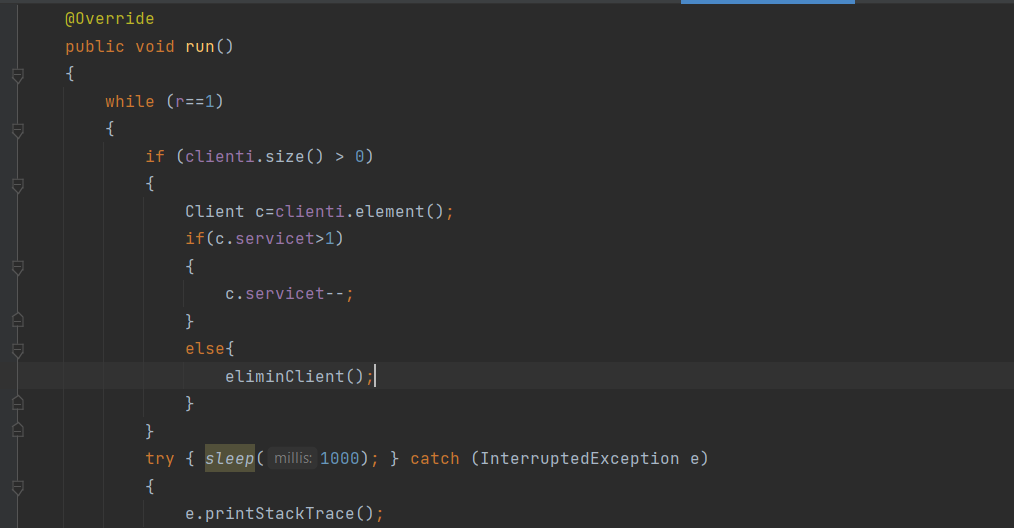
-această metodă scrie în fișier întreg conținutul istoricului, care a fost explicat mai sus, după fiecare din cele 3 teste, controlează în interfață cum clienții vin la coada respectivă și după pleacă.



* Clasa **Client :** unde sunt doar definite variabilele id, timpul de sosire al clientului și timpul de servire, și setter și getteri pentru fiecare fiind declarate de tipul private.
* Clasa **Consumator** :



-metodele pentru adăugarea, inclusiv ștergerea unui client



-metoda run() unde dacă timpul de servire e mai mare ca 1, atunci tot scade până ajunge la 0, adică până clientul este gata servit, iar când acesta ajunge la 0, clientul este șters, adică pleacă din coadă.



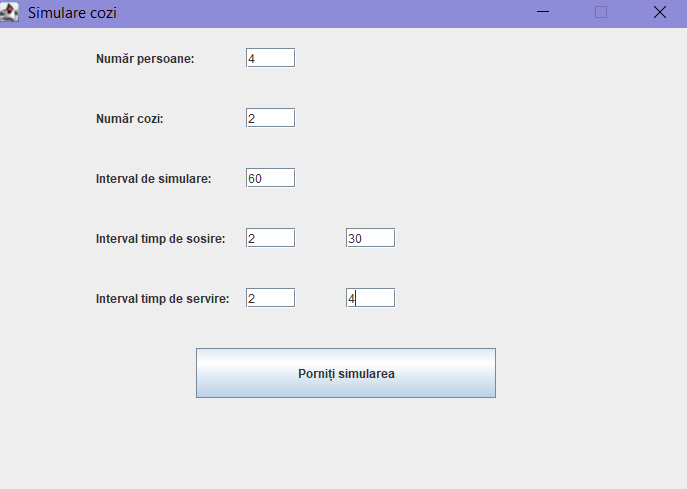
În pachetul **view** se află clasele View, Simulare și ViewJPanel

* Clasele **View** și **ViewJPanel** sunt pentru interfața grafică unde se introduc numărul de clienți, numărul de cozi, intervalul minim de sosire și maxim, intervalul minim de servire și maxim și durata simulării.
* Clasa **Simulare** este tot pentru interfață, doar că este interfața de simulare unde se arată cum clienții vin și pleacă, iar timpul curent crește și se oprește când ultimul client este gata servit.

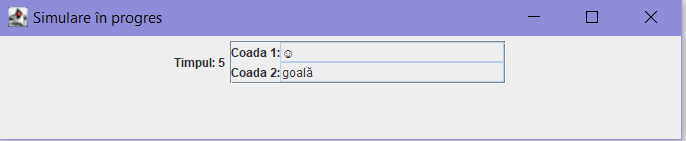
În pachetul **controller** este o singură clasă și anume clasa **Controller** unde se face legătura cu apăsarea butonului ,,Porniți simularea”.

În pachetul **fisier** se află clasa **Fisier** unde se află afișarea în diferite fișiere a evenimentelor pentru fiecare din cele 3 teste care sunt date.

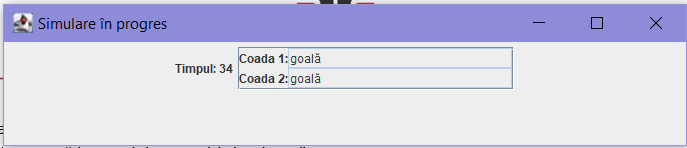
# Rezultate



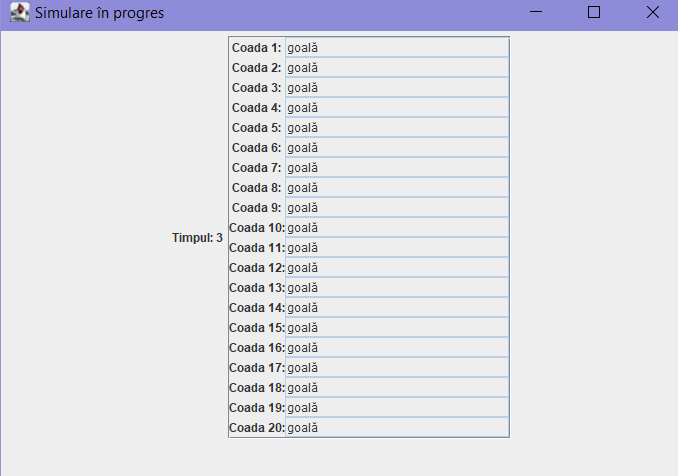
-aceasta este interfața pentru când aplicația este pornită unde sunt introduse primele date a primul test, iar după se apasă butonul de ,,Porniți simularea”.



-clientul este reprezentat în această simulare cu un emoticon, iar așa arată când intră în coadă, iar când iese, dispare emoticonul.



-acesta este sfârșitul simulării pentru primul test.



-interfața simulării când sunt mai multe cazuri.



# Concluzii

Odată cu implementarea acestei aplicații, au fost lămurite mult mai bine conceptele legate de Thread-uri și sincronizarea lor.

Ca și dezvoltare ulterioară, s-ar putea îmbunătăți interfața grafică sau să se schimbe programul astfel încât să calculez, iar apoi să afișeze statistici precum ora de vârf, timpul mediu de așteptare, etc.

# Bibliografie

<https://dsrl.eu/courses/pt/materials/PT2021-2022_Assignment_2.pdf>

<https://dsrl.eu/courses/pt/materials/A2_Support_Presentation.pdf>

<https://www.geeksforgeeks.org/semaphore-in-java/>

<https://www.w3schools.com/java/java_threads.asp>

<https://www.tutorialspoint.com/java/java_multithreading.htm>

**8.Rezultatul din fișier a primului test +finalul celor 2 teste**

N = 4 Q = 2 𝑡𝑠𝑖𝑚𝑢𝑙𝑎𝑡𝑖𝑜𝑛 𝑀𝐴𝑋 = 60 seconds [𝑡𝑎𝑟𝑟𝑖𝑣𝑎𝑙 𝑀𝐼𝑁 ,𝑡𝑎𝑟𝑟𝑖𝑣𝑎𝑙 𝑀𝐴𝑋 ] = [2, 30] [𝑡𝑠𝑒𝑟𝑣𝑖𝑐𝑒 𝑀𝐼𝑁 ,𝑡𝑠𝑒𝑟𝑣𝑖𝑐𝑒 𝑀𝐴𝑋]= [2, 4]

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Timp: 0  Clienți în așteptare:  (4,2,4)  (3,11,3)  (1,19,3)  (2,30,4)  Coada 1:goală  Coada 2:goală | Timp: 1  Clienți în așteptare:  (4,2,4)  (3,11,3)  (1,19,3)  (2,30,4)  Coada 1:goală  Coada 2:goală | Timp: 2  Clienți în așteptare:  (3,11,3)  (1,19,3)  (2,30,4)  Coada 1:(4,2,4)  Coada 2:goală | Timp: 3  Clienți în așteptare:  (3,11,3)  (1,19,3)  (2,30,4)  Coada 1:(4,2,3)  Coada 2:goală | Timp: 4  Clienți în așteptare:  (3,11,3)  (1,19,3)  (2,30,4)  Coada 1:(4,2,2)  Coada 2:goală | Timp: 5  Clienți în așteptare:  (3,11,3)  (1,19,3)  (2,30,4)  Coada 1:(4,2,1)  Coada 2:goală | Timp: 6  Clienți în așteptare:  (3,11,3)  (1,19,3)  (2,30,4)  Coada 1:goală  Coada 2:goală | Timp: 7  Clienți în așteptare:  (3,11,3)  (1,19,3)  (2,30,4)  Coada 1:goală  Coada 2:goală | Timp: 8  Clienți în așteptare:  (3,11,3)  (1,19,3)  (2,30,4)  Coada 1:goală  Coada 2:goală |  |
| Timp: 9  Clienți în așteptare:  (3,11,3)  (1,19,3)  (2,30,4)  Coada 1:goală  Coada 2:goală | Timp: 10  Clienți în așteptare:  (3,11,3)  (1,19,3)  (2,30,4)  Coada 1:goală  Coada 2:goală | Timp: 11  Clienți în așteptare:  (1,19,3)  (2,30,4)  Coada 1:(3,11,3)  Coada 2:goală | Timp: 12  Clienți în așteptare:  (1,19,3)  (2,30,4)  Coada 1:(3,11,2)  Coada 2:goală | Timp: 13  Clienți în așteptare:  (1,19,3)  (2,30,4)  Coada 1:(3,11,1)  Coada 2:goală | Timp: 14  Clienți în așteptare:  (1,19,3)  (2,30,4)  Coada 1:goală  Coada 2:goală | Timp: 15  Clienți în așteptare:  (1,19,3)  (2,30,4)  Coada 1:goală  Coada 2:goală | Timp: 16  Clienți în așteptare:  (1,19,3)  (2,30,4)  Coada 1:goală  Coada 2:goală | Timp: 17  Clienți în așteptare:  (1,19,3)  (2,30,4)  Coada 1:goală  Coada 2:goală | Timp: 18  Clienți în așteptare:  (1,19,3)  (2,30,4)  Coada 1:goală  Coada 2:goală |
| Timp: 19  Clienți în așteptare:  (2,30,4)  Coada 1:(1,19,3)  Coada 2:goală | Timp: 20  Clienți în așteptare:  (2,30,4)  Coada 1:(1,19,2)  Coada 2:goală | Timp: 21  Clienți în așteptare:  (2,30,4)  Coada 1:(1,19,1)  Coada 2:goală | Timp: 22  Clienți în așteptare:  (2,30,4)  Coada 1:goală  Coada 2:goală | Timp: 23  Clienți în așteptare:  (2,30,4)  Coada 1:goală  Coada 2:goală | Timp: 24  Clienți în așteptare:  (2,30,4)  Coada 1:goală  Coada 2:goală | Timp: 25  Clienți în așteptare:  (2,30,4)  Coada 1:goală  Coada 2:goală | Timp: 26  Clienți în așteptare:  (2,30,4)  Coada 1:goală  Coada 2:goală | Timp: 27  Clienți în așteptare:  (2,30,4)  Coada 1:goală  Coada 2:goală | Timp: 28  Clienți în așteptare:  (2,30,4)  Coada 1:goală  Coada 2:goală |
| Timp: 29  Clienți în așteptare:  (2,30,4)  Coada 1:goală  Coada 2:goală | Timp: 30  Clienți în așteptare:  Coada 1:(2,30,4)  Coada 2:goală | Timp: 31  Clienți în așteptare:  Coada 1:(2,30,3)  Coada 2:goală | Timp: 32  Clienți în așteptare:  Coada 1:(2,30,2)  Coada 2:goală | Timp: 33  Clienți în așteptare:  Coada 1:(2,30,1) | Timp: 34  Clienți în așteptare:  Coada 1:goală  Coada 2:goală |  |  |  |  |



N = 50 Q = 5 𝑡𝑠𝑖𝑚𝑢𝑙𝑎𝑡𝑖𝑜𝑛 𝑀𝐴𝑋 = 60 seconds [𝑡𝑎𝑟𝑟𝑖𝑣𝑎𝑙 𝑀𝐼𝑁 ,𝑡𝑎𝑟𝑟𝑖𝑣𝑎𝑙 𝑀𝐴𝑋 ] = [2, 40] [𝑡𝑠𝑒𝑟𝑣𝑖𝑐𝑒 𝑀𝐼𝑁 ,𝑡𝑠𝑒𝑟𝑣𝑖𝑐𝑒 𝑀𝐴𝑋 ]= [1, 7]

Timp: 46

Clienți în așteptare:

Coada 1:goală

Coada 2:goală

Coada 3:goală

Coada 4:goală

Coada 5:goală

N = 1000 Q = 20 𝑡𝑠𝑖𝑚𝑢𝑙𝑎𝑡𝑖𝑜𝑛 𝑀𝐴𝑋 = 200 seconds [𝑡𝑎𝑟𝑟𝑖𝑣𝑎𝑙 𝑀𝐼𝑁 ,𝑡𝑎𝑟𝑟𝑖𝑣𝑎𝑙 𝑀𝐴𝑋 ] = [10, 100] [𝑡𝑠𝑒𝑟𝑣𝑖𝑐𝑒 𝑀𝐼𝑁 ,𝑡𝑠𝑒𝑟𝑣𝑖𝑐𝑒 𝑀𝐴𝑋 ]= [3, 9]

Timp: 200

Clienți în așteptare:

Coada 1:(20,70,5) (525,72,6) (245,74,4) (714,75,7) (672,77,8) (213,80,4) (999,80,3) (818,81,3) (741,82,3) (464,83,9) (683,86,4) (51,88,4) (890,88,8) (603,90,5) (191,92,7) (243,94,9) (587,96,8) (308,98,9)

Coada 2:(148,71,5) (574,72,9) (410,75,3) (227,76,8) (380,78,4) (50,80,9) (928,81,8) (54,85,8) (299,87,7) (46,89,3) (504,89,9) (266,92,4) (249,93,9) (595,95,6) (910,96,3) (761,97,4) (668,98,8)

Coada 3:(491,69,3) (950,71,6) (322,73,3) (645,74,6) (586,76,6) (283,78,5) (94,80,6) (441,81,9) (252,84,7) (994,85,4) (851,87,9) (635,89,3) (199,90,9) (917,92,4) (332,94,4) (869,94,7) (829,96,8) (752,98,7)



Coada 4:(162,71,9) (409,73,3) (129,75,9) (325,77,4) (513,78,3) (215,79,8) (545,81,8) (392,84,6) (716,85,6) (161,88,7) (294,89,7) (69,91,7) (445,93,4) (670,94,3) (152,95,6) (793,96,9) (894,98,3) (47,100,9)

Coada 5:(776,69,3) (35,72,4) (100,73,4) (506,74,6) (338,76,8) (616,78,3) (288,79,8) (588,81,9) (611,84,6) (83,86,4) (864,87,9) (654,89,4) (738,90,6) (465,92,5) (652,93,7) (757,95,7) (43,97,6) (956,98,5) (440,100,3)

Coada 6:(113,69,1) (276,71,5) (755,72,4) (935,73,6) (90,76,5) (400,77,8) (132,80,4) (847,80,5) (970,81,3) (875,82,9) (836,85,9) (658,88,8) (298,90,3) (110,91,9) (698,93,5) (201,95,4) (139,96,5) (97,97,3) (29,98,7) (165,100,9)

Coada 7:(164,71,3) (108,72,3) (942,72,3) (429,73,6) (723,75,9) (312,78,8) (450,80,6) (71,82,4) (271,83,8) (841,85,8) (398,88,8) (86,90,8) (628,92,5) (795,93,4) (775,94,5) (253,96,8) (53,98,3) (273,99,6)

Coada 8:(902,69,3) (187,72,8) (566,74,8) (814,76,9) (413,79,7) (516,81,4) (688,82,8) (169,85,7) (367,87,6) (925,88,4) (769,89,8) (720,91,4) (124,93,3) (913,93,8) (810,95,7) (383,97,6) (339,99,4) (553,100,7)

Coada 9:(166,71,7) (126,73,5) (285,75,6) (604,76,8) (936,78,8) (56,81,8) (415,83,9) (92,86,8) (689,88,6) (3,90,6) (307,91,7) (561,93,9) (6,96,8) (767,97,3) (547,98,6) (941,100,4)

Coada 10:(973,69,3) (275,72,3) (947,72,4) (75,74,5) (857,75,5) (264,77,8) (470,79,5) (957,80,5) (120,82,5) (978,83,5) (397,85,5) (190,87,3) (335,88,6) (140,89,3) (80,90,5) (112,91,9) (24,94,6) (481,95,9) (394,97,4) (581,98,3) (598,99,3)

Coada 11:(909,70,4) (444,72,9) (331,75,7) (281,77,3) (196,78,9) (536,80,9) (931,82,6) (313,85,8) (985,87,7) (217,89,5) (306,90,4) (575,91,3) (659,92,5) (26,94,6) (518,95,4) (599,96,4) (156,97,4) (382,98,7)

Coada 12:(237,71,5) (577,72,9) (640,75,5) (819,76,3) (700,77,3) (660,78,4) (157,80,9) (507,82,8) (84,85,3) (438,85,8) (347,88,6) (354,89,9) (736,91,5) (530,93,3) (366,94,8) (255,96,8) (172,98,3) (311,99,9)



Coada 13:(517,70,2) (794,71,5) (231,73,6) (363,75,9) (713,77,6) (486,79,4) (608,80,4) (621,81,6) (461,83,9) (236,86,8) (703,88,5) (804,89,6) (226,91,9) (40,94,5) (246,95,5) (625,96,8) (442,98,3) (419,99,3) (209,100,9)

Coada 14:(67,71,6) (967,72,4) (118,74,7) (443,76,8) (696,78,9) (207,81,7) (934,82,4) (768,84,7) (878,86,7) (832,88,4) (421,89,9) (918,91,8) (495,94,8) (293,96,3) (907,96,7) (582,98,9)

Coada 15:(580,69,1) (567,71,7) (309,73,6) (666,75,4) (675,76,3) (448,77,9) (303,80,7) (860,81,8) (779,84,6) (268,86,3) (396,87,3) (349,88,3) (21,89,3) (477,89,6) (845,90,3) (49,92,8) (523,94,8) (356,96,9) (447,98,7)

Coada 16:(774,70,3) (318,72,7) (208,74,6) (222,76,6) (949,77,6) (682,79,6) (418,81,3) (886,81,9) (144,85,9) (13,88,4) (859,88,5) (946,89,3) (406,90,7) (827,92,5) (154,94,7) (801,95,6) (912,96,3) (904,97,6) (602,99,6)

Coada 17:(19,70,1) (673,71,8) (449,73,3) (300,75,3) (988,75,4) (219,77,3) (989,77,3) (731,78,3) (726,79,9) (908,81,8) (839,84,9) (646,87,7) (109,89,8) (966,90,6) (868,92,7) (748,94,7) (627,96,6) (983,97,6) (709,99,4)

Coada 18:(744,69,3) (405,72,3) (31,73,9) (4,76,7) (28,78,6) (834,79,5) (959,80,8) (32,83,9) (858,85,8) (572,88,9) (520,90,5) (146,92,4) (218,93,9) (579,95,3) (538,96,6) (515,97,3) (494,98,4) (805,99,4)

Coada 19:(72,71,5) (614,72,8) (403,75,5) (710,76,9) (104,79,4) (351,80,9) (874,82,5) (846,84,4) (674,85,5) (742,87,6) (33,89,9) (65,91,4) (333,92,7) (612,94,5) (590,95,5) (809,96,3) (263,97,7) (472,99,6)

Coada 20:(377,70,4) (498,72,5) (760,73,4) (408,75,5) (753,76,7) (927,78,7) (739,80,6) (559,82,3) (150,83,4) (997,84,7) (147,87,7) (877,88,4) (503,89,4) (554,90,5) (185,92,9) (684,94,6) (78,96,9) (280,98,6) (107,100,5)