UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

TEHNICI DE PROGRAMARE
TEMA 2: QUEUES SIMULATOR

STUDENT: PELE CARMEN-IOANA

GRUPA 30222

Cuprins

1.OBIECTIVUL TEMEI

Obiectivul principal al acestei teme este de a implementa o aplicatie care simuleaza cozile de asteptare, care analizeaza sistemul de cozi pentru a determina si minimiza timpul de asteptare minim al clientilor. Aceasta simulare ii va permite utilizatorului sa introduca datele dorite pentru a simula evolutia unui anumit numar de cozi si clienti in timp, dar acesta va avea si posibilitatea sa vada evolutia lor prin intermediul unei interfete in raport cu timpul, iar la final acesta va putea cateva date rezultate in urma simularii cozilor, cum ar fi timpul mediu de asteptat la coada, timpul mediu de lucru pe cozi si ora de varf.

Alte sub-obiective ar fi: analizarea problemei si identificarea necesitatiilor, design-ul simulatorului de cozi, implementarea simulatorului si testarea functionalitatii acestuia.

2.ANALIZA PROBLEMEI, MODELARE, SCENARII, CAZURI DE UTILIZARE

Necesitatile functionale ale programului sunt:

- Programul ar trebui sa ii permita utilizatorului sa introduca numarul de clienti pentru care se doreste simularea
- Programul ar trebui sa ii permita utilizatorului sa introduca numarul de cozi pentru care se doreste simularea
- Programul ar trebui sa ii permita utilizatorului sa introduca intervalul de simulare
- Programul ar trebui sa ii permita utilizatorului sa introduca timpul minim si maxim la care clientii pot ajunge la coada
- Programul ar trebui sa ii permita utilizatorului sa introduca timpul minim si maxim de asteptare la coada a clientilor
- Programul ar trebui sa poata simula evolutia cozilor in functie de datele introduse
- Programul ar trebui sa ii permita utilizatorului sa vada in evolutia cozilor in functie de timp prin intermediul interfetei
- Programul ar trebui sa ii permita utilizatorului sa vada timpul mediu de asteptare, timpul mediu de lucru pe cozi si ora de varf, rezultate in urma simularii

Necesitatile nefunctionale ale programului sunt:

- Simulatorul ar trebui sa fie intuitiv si usor de utilizat de catre utilizator
- Programul ar trebui sa atentioneze utilizatorii in caz ca acestia nu au introdus corect datele de intrare
- Programul ar trebui sa ilustreze simularea cozilor intr-o forma usor de inteles de catre utilizator
- Programul ar trebui sa efectueze simularea in timpul dorit de utilizatori, daca datele sunt introduse corect
- Programul ar trebui ca in urma simularii sa ii afiseze rezultatele utilizatorului sub o forma usor de inteles de catre acesta

Un caz de utilizare este:

Use Case: introducerea datelor simularii

Actorul principal: utilizatorul

Scenariul principal de succes:

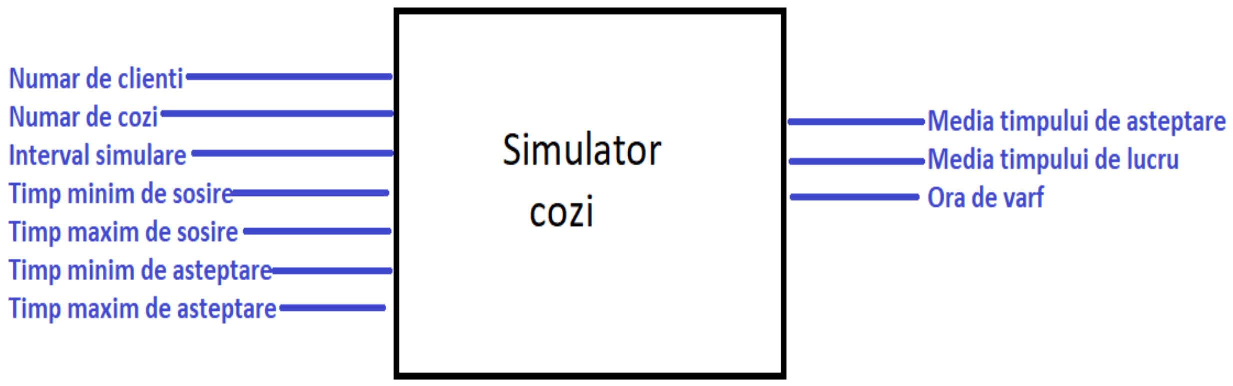
1. Utilizatorul introduce numarul de clienti, numarul de cozi, intervalul de simulare, timpul maxim si minim de sosire la coada si timpul maxim si minim de asteptare la cozi
2. Utilizatorul apasa butonul "start" pentru a verifica datele introduse, iar daca acestea sunt corecte se va deschide o noua fereastra care ii afiseaza simularea evolutiei cozilor in funtie de datele introduse
3. Utilizatorul asteapta terminarea simularii pentru a i se deschide o noua fereastra in care ii sunt trecute rezultatele in urma simularii

Un scenariu alternativ in cazul in care datele sunt incorecte:

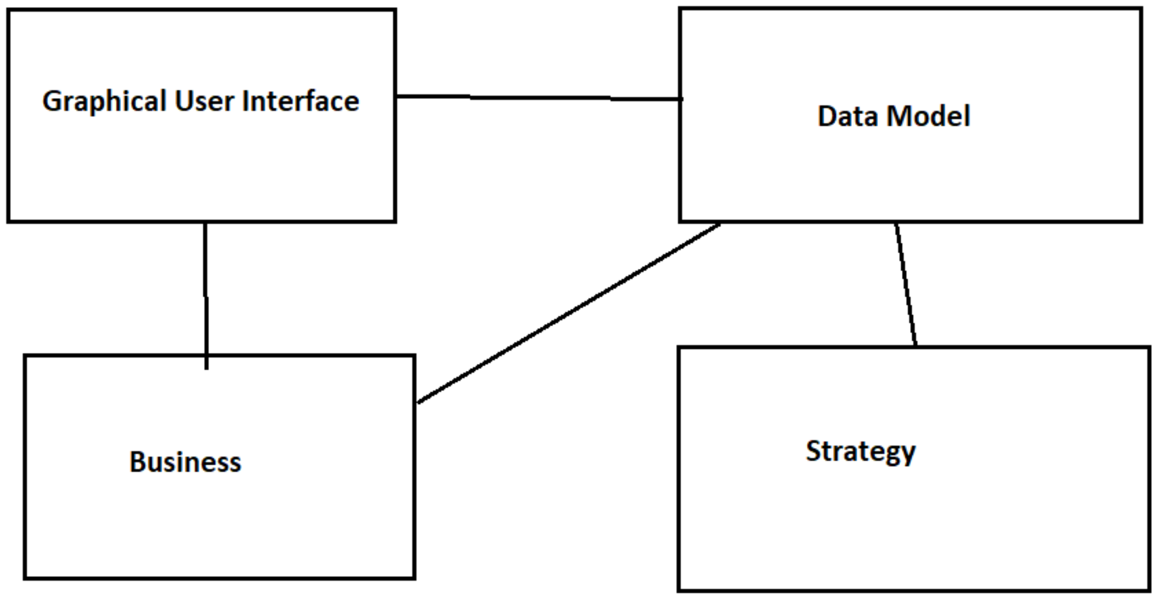
- Utilizatorul insereaza datele
- Acesta apasa "start" pentru a incepe
- Programul il atentioneaza ca datele introduse nu sunt corect scrise sau nu permit o functionare corecta a simularii
- Scenariul se reintoarce la pasul 1, adica cel de introducere a datelor

3.PROIECTARE (DECIZII DE PROIECTARE, DIAGRAME UML, STRUCTURI DE DATE, PROIECTARE CLASE, INTERFETE, RELATII, PACKAGES, ALGORITMI, INTERFATA UTILIZATOR)

3.1. Design-ul overall al sistemului:



3.2 Divizarea in sub-sisteme/pachete

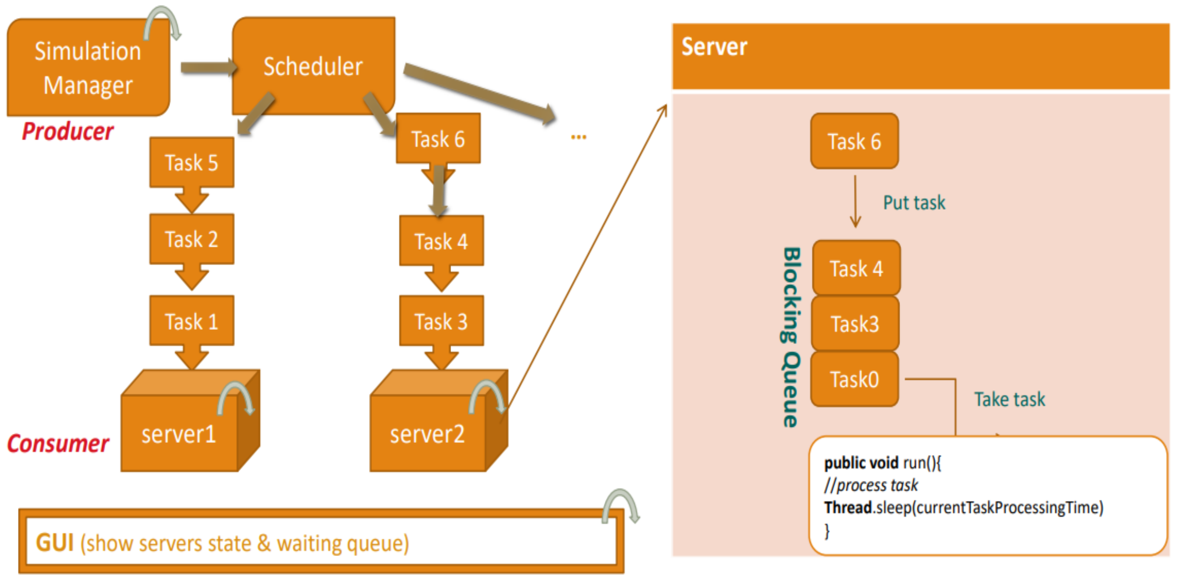


“Graphical User Interface” contine clasele care implementeaza interfata grafica a utilizatorului.

“Business Logic” contine clasele care implementeaza functionalitatea cozilor, prin thread-uri, exceptii si scrierea in fisier a datelor obtinute.

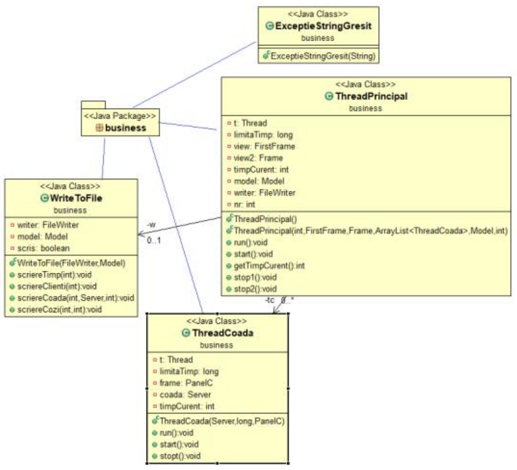
“Data models” contine clasele care care modeleza datele aplicatiei(Task si Serve).

“Strategy” contine metodele care ajuta la generarea aleatoare a clientilor si adaugarea acestora la cozi.



```
public void run(){
//process task
Thread.sleep(currentTaskProcessingTime)
}
```

3.3.Divizare in clase

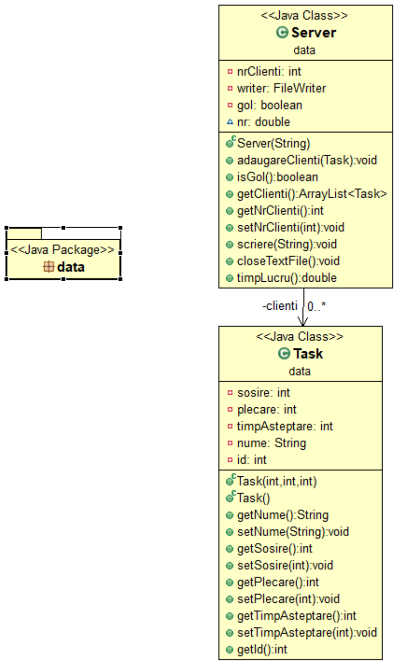


Pachetul business are clasele:

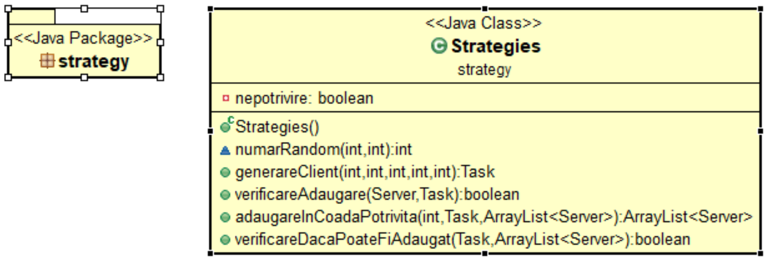
- ExceptieStringGresit
- WriteToFile
- ThreadPrincipal
- ThreadCoada

Pachetul data are clasele:

- Task
- Server



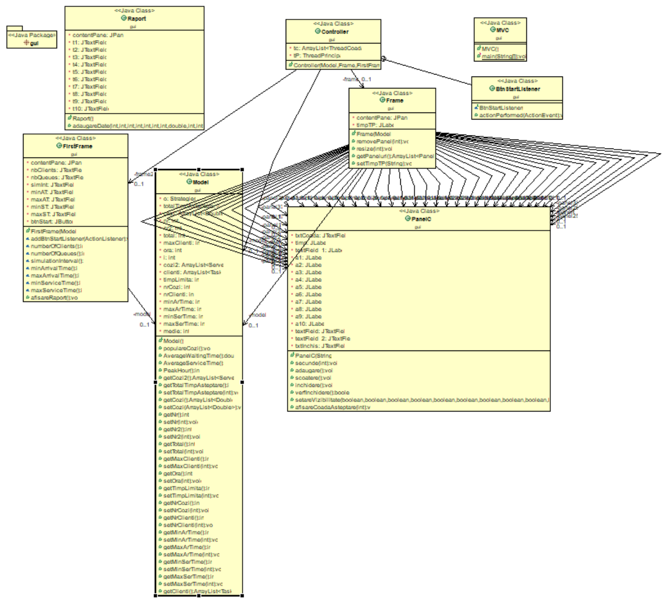
Pachetul strategy contine clasa Strategies.



<<Java Package>>
strategy
<<Java Class>>
Strategies
strategy
nepotrivire: boolean
Strategies()
numarRandom(int,int):int
generareClient(int,int,int,int,int):Task
verificareAdaugare(Server,Task):boolean
adaugareInCoadaPotrivita(int,Task,ArrayList<Server>):ArrayList<Server>
verificareDacaPoateFiAdaugat(Task,ArrayList<Server>):boolean

Pachetul gui contine clasele:

- Controller
- FirstFrame
- Frame
- Model
- MVC
- PanelC
- Raport



3.4. Divizarea pe metode:

Clasa ThreadCoada are urmatoarele metode:

- run()- efectueaza ilustrarea evolutiei unei coade, prin adaugarea si scoaterea clientilor de pe interfata, in functie de timp
- start()-initializeaza thread-ul si il porneste
- stopt()-folosita pentru oprirea thread-ului din exterior

Clasa ThreadPrincipal are urmatoarele metode:

- run()-scrie datele in fisierul “simulation.txt” si porneste thread-urile cozilor
- start()-initializeaza thread-ul si il porneste
- stop1()- opreste thread-urilor cozilor si thread-ul principal
- stop2()-opreste thread-urile cozilor, inchide fereastra de simulare, afiseaza fereastra de raport si opreste thread-ul principal
- getTimpCurent()- returneaza timpul curent din thread

Clasa WriteToFIle are urmatoarele metode:

- scriereTimp()- scrie in fisier sub forma “Time:”+valoare timp curent+”\n”
- scriereClienti()- scrie in fisier clientii care asteapta la cozi, adica cai care au timpul de sosire mai mare decat timpul curent
- scriereCoada()- scrie clientii dintr-o coada transmisa ca parametru
- scriereCozi()- scrie toate cozile cu clientii lor in fisier

Clasa Server are urmatoarele clase:

- adaugareClienti()
- isGol()
- getClienti()
- getNrClienti()
- setNrClienti()
- scriere()- scrie in fisierul cozi respective un string care contine datele in legatura cu clientii
- closeTextFile() – inchide fisierul de cozi
- timpLucru()- returneaza suma timpului de servire a fiecarui client din coada respectiva

Clasa Task are urmatoarele clase:

- getNume()
- setNume()
- getSosire()
- setSosire()
- getPlecare()
- setPlecare()
- getTimpAsteptare()
- setTimpAsteptare()
- getId()

Clasa Controller are subclasa BtnStartListener care implementeaza verificarea datelor introduse de utilizator, ascunde primul frame si afiseaza frame-ul cu simularea, initializeaza 40 de thread-uri, care ar fi numarul maxim de cozi care pot fi simulate in program, ascunde panel-uri corespunzatoare cozilor goale, initializaza si porneste thread-ul principal.

Clasa FirstFrame are urmatoarele clase:

- addBtnStartListener()
- numberOfClients()- returneaza valoarea introdusa de utilizator sau arunca o exceptie in caz ca datele introduse sunt gresite
- numberOfQueues()
- simulationInterval()
- minArrivalTime()
- maxArrivalTime()
- minServiceTime()
- maxServiceTime()
- afisareRaport()- inchide frame-ul curent si afiseaza frame-ul raportului

Clasa Frame are urmatoarele clase:

- removePanel()- deoarece am asezat un numar de 40 de panel-uri care ar corespunde cozilor, aceasta metoda le ascunde pe cele care nu sunt folosite
- resize()- aceasta metoda seteaza dimensiunea frame-ului in functie de un nr transmis ca parametru, care reprezinta numarul de cozi
- getPaneluri() – returneaza ArrayList-ul de paneluri

Clasa Model are urmatoarele clase:

- populareCozi()-populeaza coziile cu clientii, in caz ca un client nu se poate repartiza la cozile introduse de utilizator acesta va fi adaugat intr-o alta coada suplimentara, daca nu se poate adauga la nici una din cele 40 de cozi, care reprezinta maximul, atunci se genereaza alte date la acel client
- averageWaitingTime()- returneaza timpul de asteptare mediu
- averageServiceTime()- returneaza timpul de lucru mediu pe fiecare coada
- peakHour()- returneaza ora de varf
- getCozi2()
- getTotalAsteptare()
- getTotalTimpAsteptare()
- setTotalTimpAsteptare()
- getCozi()
- setCozi()
- getOra()
- setOra()
- getTimpLimita()
- setTimpLimita()
- getNrCozi()
- setNrCozi()
- getNrClienti()

- setNrClienti()
- getMinArTime()
- setMinArTime()
- getMaxArTime()
- setMaxArTime()
- getMinSerTime()
- setMinSerTime()
- getMaxSerTime()
- setMaxSerTime()
- getClienti()

Clasa PanelC are urmatoarele clase:

- secunde()- seteaza timpul curent al panel-ului
- adaugare()- afiseaza un client ca fiind in locul de servire
- scoatere()- scoate clientul din zona de servire
- inchidere()- afiseaza semnul de "inchis" in dreptul cozii respective
- verfInchidere()- verifica daca coada e inchisa sau nu
- setareVizibilitate()- seteaza vizibilitatea label-urilor cu imaginile clientilor, maxim 10 clienti per coada
- afisareCoadaAsteptare()- afiseaza coada in functie de numarul de clienti

Clasa Raport are clasa:

- adaugareDate()- care adauga rezultatele in urma simularii in textField-urile din raport

Clasa Strategies are urmatoarele clase:

- numarRandom()- genereaza un numar random cuprins intre un minim si maxim
- generareClient()- genereaza aleatoriu un client in functie de date introduse
- verificareAdaugare()- verifica daca un client poate fi adaugat intr-o coada, adica daca intervalul lui de timp nu se suprapune cu al unui alt client deja existent in coada
- adaugareInCoadaPotrivita() – adauga clientul in coada care are liber intervalul de timp pe care-l ocupa clientul respectiv
- verificareDacaPoateFiAdaugat()- returneaza daca clientul poate fi adaugat in una din cozile disponibile

3.5. Decizii de proiectare- *Model View Controller Architectural Pattern:*

Context:

Multe sisteme se ocupa cu cautarea datelor dintr-un „depozit" si sa afiseze datele utlizatorilor prin intermediul unei interfete grafice.

-Utilizatorul poate modifica datele si modificarile vor fi salvate in depozit
-Fluxul de informatii circula continuu intre GUI si depozit -> poate fi tentant sa implementezi totul intr-o singura clasa
Dezavantaje:

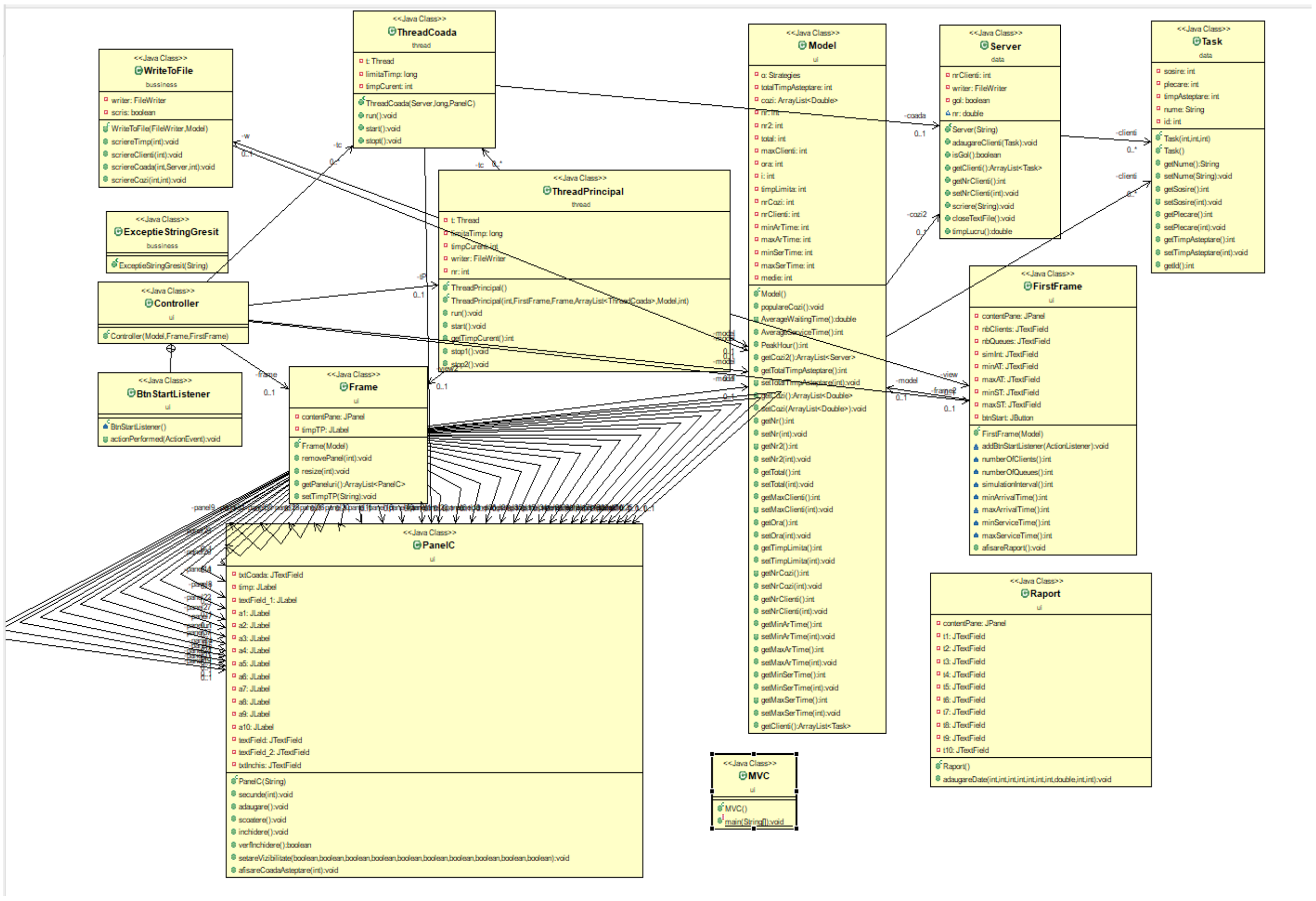
- GUI se schimba mai mult decat implementarea business logic-ului-> daca sunt implementate in aceeasi clasa de fiecare data cand GUI se schimba si business logic se schimba
- Business logic nu poate fi reutilizat
- Codul e complex si greu de mentinut

Solutie:

Utilizarea Model View Controller pattern care imparte aplicatia in 3 zone: procesare, output si input.

- Componentele modelului: incapsuleaza nucleul datelor si functionalitatilor
- Componenta view: Afiseaza informatiile catre utilizator, obtinand datele afisate din model
- Controller: Fiecare view are un controller asociat. Acesta controleaza input-urile, de obicei evenimente obtinute din miscarile cu mouse-ul. Evenimentele sunt traduse ca necesitati ale sistemului, care sunt transmise fie modelului, fie view-ului.

3.6 Diagrame UML



3.7. Structuri de date

Pentru implementarea unui Task avem structura:

```
public class Task {
	private int sosire;
	private int plecare;
	private int timpAsteptare;
	private String nume;
	private int id;

…}
```

Variabilele “sosire”, “plecare”, “timpAsteptare”, “nume”, “id” repezinta ora la care clientul soseste la coada, ora la care acesta pleaca, timpul de asteptare/servire, numele sub forma “clientul ”+id,si id-ul clientului.

Pentru implementarea unui Server avem structura:

```
public class Server {
	private ArrayList<Task> clienti;
	private int nrClienti;
	private  FileWriter writer;
	private boolean gol;

…}
```

ArrayList-ul “clienti” contine clientii care asteapta la coada respectiva, aduca cei care vor fi serviti la un moment dat in coada respectiva. Variabila “nrClienti” contine numarul de clienti din lista, si aceasta se va decrementa cand unul dintre acestia este servit, “writer” este fisierul in care vom scrie datele cozii si “gol” e adevarat daca coada e goala de la inceput si fals daca au fost introdusi clienti in ea.

3.8.Algoritmi

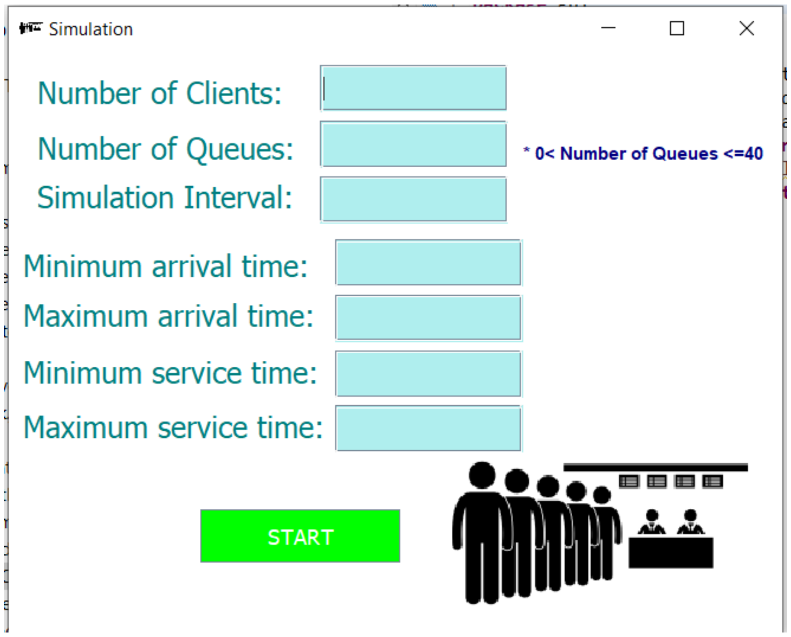
Pentru generarea unui Client: generam un numar random intre valorile de minim si maxim de sosire introduse de utilizator, la fel si un pentru valoarea timpului de asteptare al clientului, si cream un client cu datele obtinute.

Pentru adaugarea clientilor in coada am folosit urmatorul algoritm: pentru fiecare client care trebuie introdus in cozi, generam random datele acestuia si verificam daca acesta poate fi introdus in cozi, cat timp acesta nu poate fi introdus in nici una din coziile din sistem, se tot genereaza alte date si se reverifica. Deoarece este posibil ca intervalul de asteptare introdus de utilizator sa faca ca nu toti clientii sa poata fi introdusi in cozi, am verificam inainte de generare daca minSerTime permite introducerea tuturor clientilor in maxim 40 de cozi, altfel genereaza eroare, in caz ca maxSerTime nu permite introducerea tuturor clientilor, acesta se decrementeaza pana cand ajunge la o valoare care permite acest lucru, cel putin egala cu minSerTime.Astfel fiecare client poate poate fi introdus in cel mult 40 de cozi. La adaugarea lui in coada, se parcurg toate coziile, mai intai cele care ar trebui care corespund numarului de cozi introduse de utilizator, in caz ca nu poate fi introdus in nici una din acestea, se parcurg si celelalte cozi, in limita a 40, astfel ca indiferent de datele lui, clientul sa poata fi introdus in una din cozi.

Pentru generarea mediei timpului de asteptare, am adunat timpul de asteptare de la fiecare client si am impartit la numarul de clienti. Pentru generarea mediei timpului de servire, am adunat timpul de asteptare a tuturor clientilor dintr-o coada, pastrand valorile acestea intr-un arraylist si dupa am facut media aritmetica a valorilor din acesta. Pentru ora de varf am parcurs fiecare coada si am determinat cati clienti sunt serviti la toate cozile la un anumit moment de timp, retinand valoarea maxima a acestui numar si ora corespunzatoare, pe care am returnat-o la final.

3.9.Interfata utilizatorului

Primul frame care se deschide la rularea programului, pentru a permite utilizatorului sa introduca datele


Simulation

Number of Clients:

Number of Queues: * 0< Number of Queues <=40

Simulation Interval:

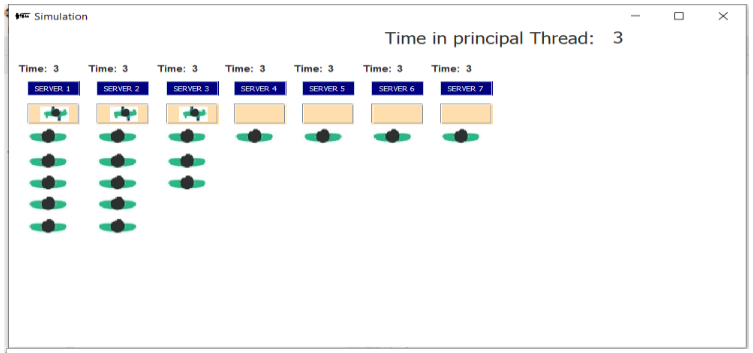
Minimum arrival time:

Maximum arrival time:

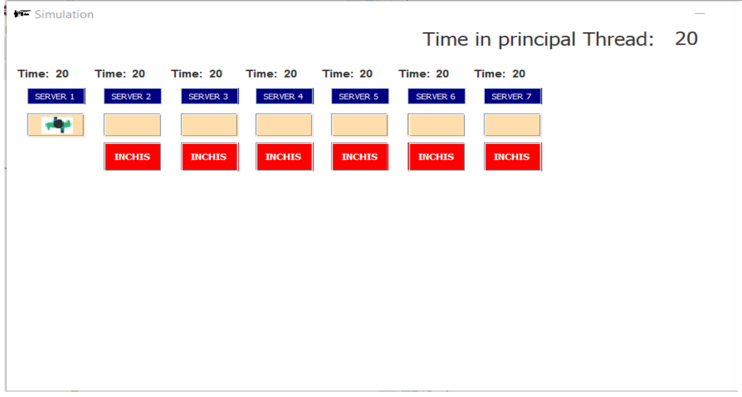
Minimum service time:

Maximum service time:

Al doilea frame care se deschide daca au fost introduse corect datele si s-a apasat „start”:

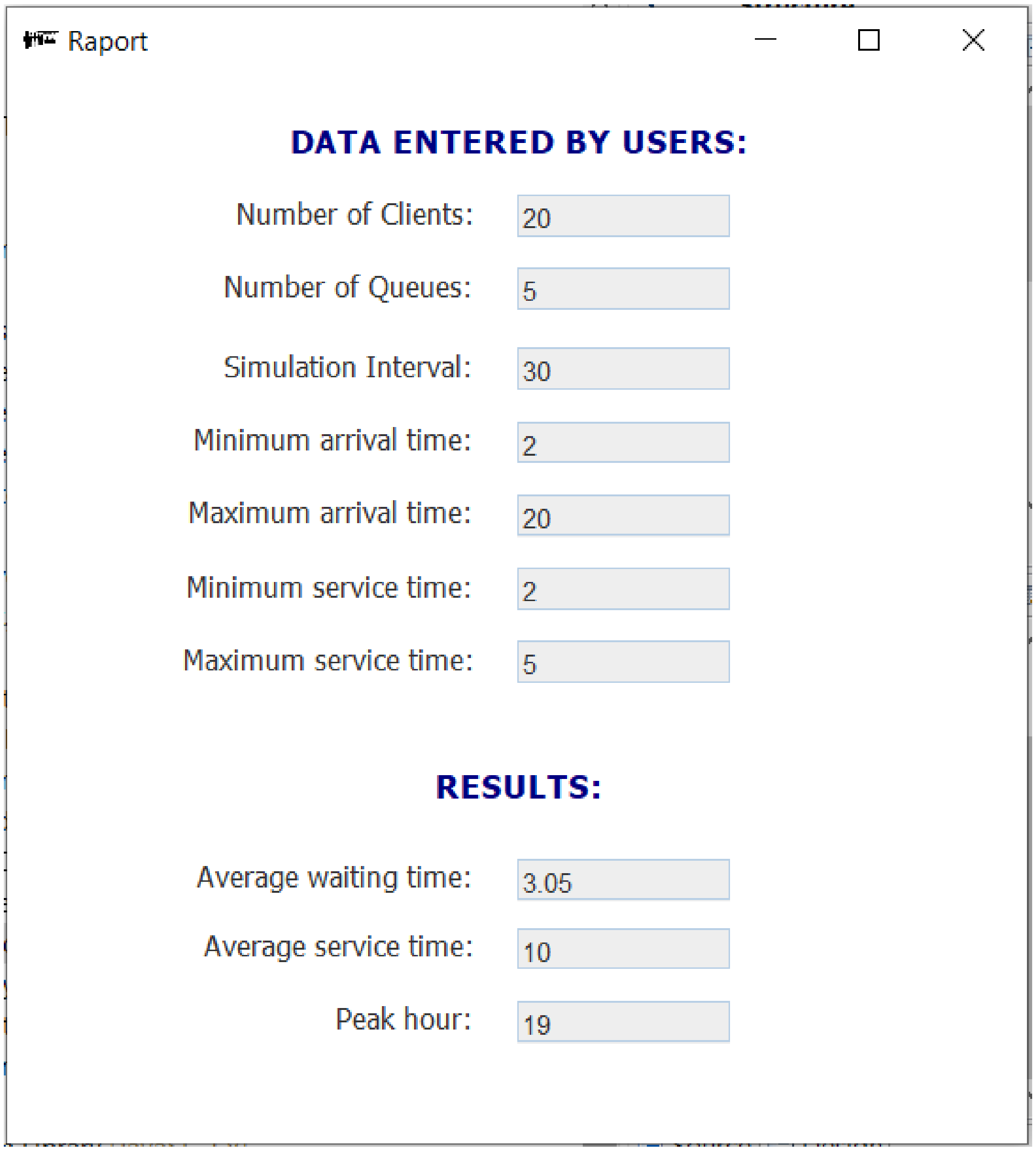

Simulation

Time in principal Thread: 3


Simulation

Time in principal Thread: 20

Ultimul frame care reprezinta raport-ul in urma simularii, acesta se deschide dupa 2 secunde dupa terminarea timpului simularii:


Raport

DATA ENTERED BY USERS:

Number of Clients:	20
Number of Queues:	5
Simulation Interval:	30
Minimum arrival time:	2
Maximum arrival time:	20
Minimum service time:	2
Maximum service time:	5

RESULTS:

Average waiting time:	3.05
Average service time:	10
Peak hour:	19

4.IMPLEMENTARE

Utilizatorul introduce datele in primul frame, iar dupa apasarea butonului de start se verifica acuratetea datelor introduse, mai intai daca au fost scrise bine si mai apoi daca datele introduse permit functionarea corecta a simularii. In caz ca sunt incorecte se afiseaza mesajul corespunzator. Daca toate datele au fost introduse corect se inchide prima fereastra si se deschide una care ilustreaza evolutia cozilor in timp in functie de datele introduse. In acelasi timp se scrie atat pentru fiecare coada in parte intr-un fisier propiu ce clienti au fost, ora de sosire, timpul de asteptare si ora de plecare, dar si intr-un fisier care contine toate datele legate de cozi, scrise sub forma data in cerinta temei. Dupa terminarea timpului dat ca Simulation Interval, thread-urile de cozi se opresc, insa thread-ul principal mai ruleaza inca 2 secunde, pe urma fereastra simularii se inchide si se deschide fereastra raportului.

Am limitat in implementarea mea numarul de cozi la 40, pentru ca sa pot rula si exemplul din cerinta cu 1000 de clienti, deoarece aceste cozi care sunt afisate corespund unor panel-uri, care functioneaza pe baza thread-urilor, aceste panel-uri sunt deja adaugate in fereastra doar ca daca nu sunt folosite sunt setate sa fie invizibile, si fereastra sa isi modifice singura dimensiunile, pentru ca acestea sa fie aranjate pe fereastra in momentul rularii. Am utilizat trei frame-uri care se deschid unul dupa celalalt, pentru a nu incarca un singur frame si pentru a face o utilizare mai simpla pentru utilizator a programului. Iar pentru popularea cozilor, am implementat un algoritm care verifica intervalele clientilor existenti si a clientului pentru care se face verificarea, astfel ca daca un client are timpul de sosire si plecare mai mici decat cele a celorlalti clienti acesta poate fi adaugat in fata altui client deja existent, nu adaugam doar daca e dupa clientii deja existenti.

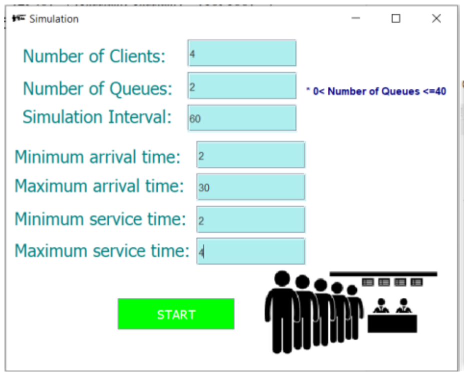
5.REZULTATE

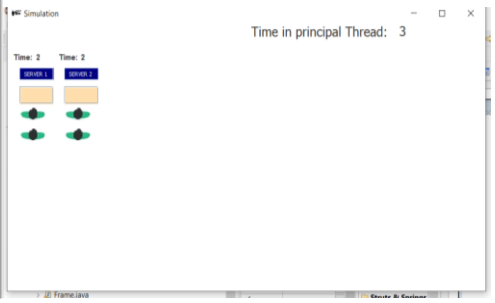
Pentru testele:

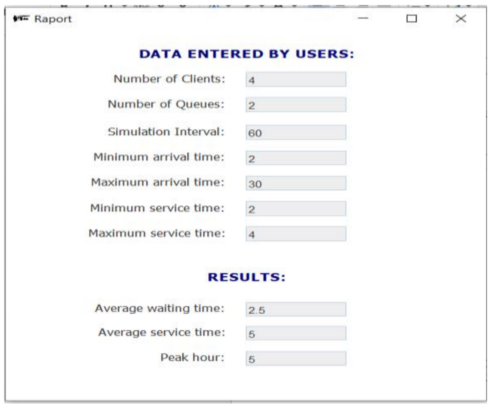
Test 1	Test 2	Test 3
N = 4 Q = 2 $t_{simulation}^{MAX}$ = 60 seconds $[t_{arrival}^{MIN}, t_{arrival}^{MAX}] = [2, 30]$ $[t_{service}^{MIN}, t_{service}^{MAX}] = [2, 4]$	N = 50 Q = 5 $t_{simulation}^{MAX}$ = 60 seconds $[t_{arrival}^{MIN}, t_{arrival}^{MAX}] = [2, 40]$ $[t_{service}^{MIN}, t_{service}^{MAX}] = [1, 7]$	N = 1000 Q = 20 $t_{simulation}^{MAX}$ = 200 seconds $[t_{arrival}^{MIN}, t_{arrival}^{MAX}] = [10, 100]$ $[t_{service}^{MIN}, t_{service}^{MAX}] = [3, 9]$

Am obtinut urmatoarele rezultate:

- Test 1


Number of Clients: 4
Number of Queues: 2
Simulation Interval: 60
Minimum arrival time: 2
Maximum arrival time: 30
Minimum service time: 2
Maximum service time: 4


Time in principal Thread: 3

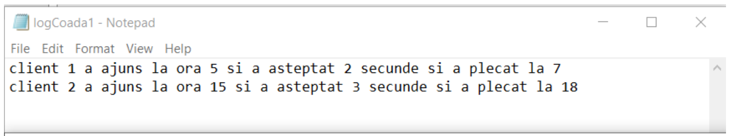


DATA ENTERED BY USERS:

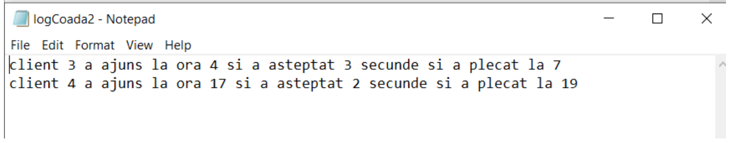
Number of Clients:	4
Number of Queues:	2
Simulation Interval:	60
Minimum arrival time:	2
Maximum arrival time:	30
Minimum service time:	2
Maximum service time:	4

RESULTS:

Average waiting time:	2.5
Average service time:	5
Peak hour:	5



```
client 1 a ajuns la ora 5 si a asteptat 2 secunde si a plecat la 7
client 2 a ajuns la ora 15 si a asteptat 3 secunde si a plecat la 18
```

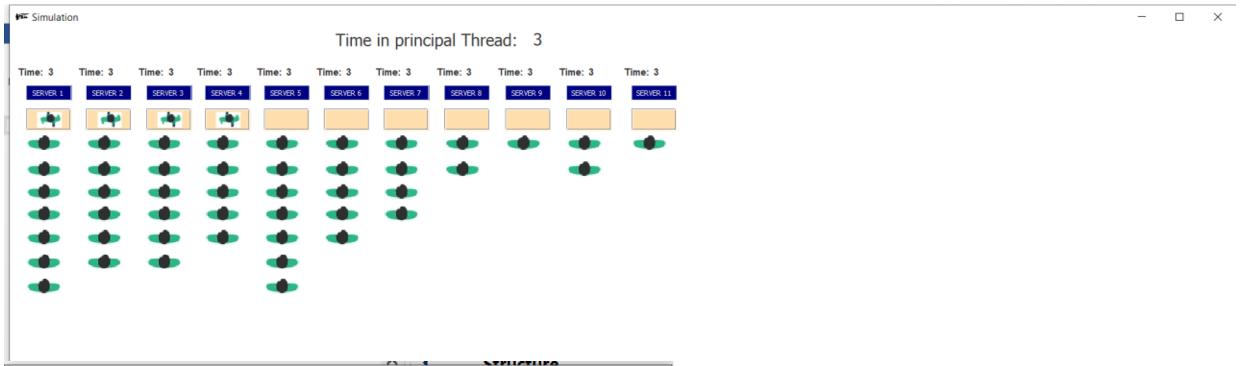


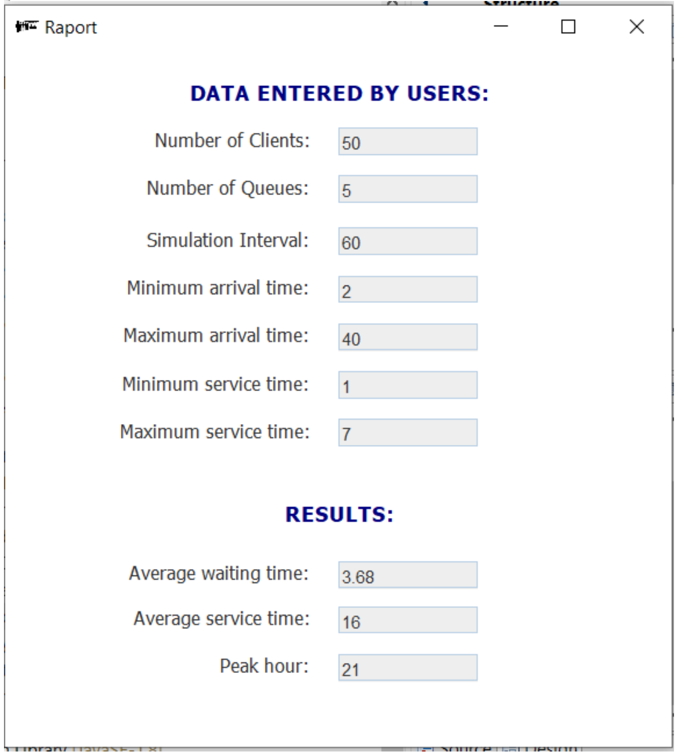
```
client 3 a ajuns la ora 4 si a asteptat 3 secunde si a plecat la 7
client 4 a ajuns la ora 17 si a asteptat 2 secunde si a plecat la 19
```

```
Time:0
Waiting
Clients:(1,5,2)(2,15,3)(3,4,3)(
4,17,2)
Queue 1: closed
Queue 2: closed
Time:1
Waiting
Clients:(1,5,2)(2,15,3)(3,4,3)(
4,17,2)
Queue 1: closed
Queue 2: closed
Time:2
Waiting
Clients:(1,5,2)(2,15,3)(3,4,3)(
4,17,2)
Queue 1: closed
Queue 2: closed
Time:3
Waiting
Clients:(1,5,2)(2,15,3)(3,4,3)(
4,17,2)
Queue 1: closed
Queue 2: closed
Time:4
Waiting
Clients:(1,5,2)(2,15,3)(4,17,2
)
Queue 1: closed
Queue 2: (3,4,3)
Time:5
Waiting
Clients:(2,15,3)(4,17,2)
Queue 1: (1,5,2)
Queue 2: (3,4,2)
Time:6
Waiting
Clients:(2,15,3)(4,17,2)
Queue 1: (1,5,1)
Queue 2: (3,4,1)
Time:7
Waiting
Clients:(2,15,3)(4,17,2)
Queue 1: closed
Queue 2: closed
Time:8
Waiting
Clients:(2,15,3)(4,17,2)
Queue 1: closed
Queue 2: closed
Time:9
Waiting
Clients:(2,15,3)(4,17,2)
Queue 1: closed
Queue 2: closed
Time:10
Waiting
Clients:(2,15,3)(4,17,2)
Queue 1: closed
Queue 2: closed
Time:11
Waiting
Clients:(2,15,3)(4,17,2)
Queue 1: closed
Queue 2: closed
Time:12
Waiting
Clients:(2,15,3)(4,17,2)
Queue 1: closed
Queue 2: closed
Time:13
Waiting
Clients:(2,15,3)(4,17,2)
Queue 1: closed
Queue 2: closed
Time:14
Waiting
Clients:(2,15,3)(4,17,2)
Queue 1: closed
Queue 2: closed
Time:15
Waiting Clients:(4,17,2)
Queue 1: (2,15,3)
Queue 2: closed
Time:16
Waiting Clients:(4,17,2)
Queue 1: (2,15,2)
Queue 2: closed
Time:17
Waiting Clients:
Queue 1: (2,15,1)
Queue 2: (4,17,2)
Time:18
Waiting Clients:
Queue 1: closed
Queue 2: (4,17,1)
Time:19
Waiting Clients:
Queue 1: closed
Queue 2: closed
Time:20
Waiting Clients:
Queue 1: closed
Queue 2: closed
Time:21
Waiting Clients:
Queue 1: closed
Queue 2: closed
Time:22
Waiting Clients:
Queue 1: closed
Queue 2: closed
Time:23
Waiting Clients:
```

```
Queue 1: closed
Queue 2: closed
Time:24
Waiting Clients:
Queue 1: closed
Queue 2: closed
Time:25
Waiting Clients:
Queue 1: closed
Queue 2: closed
Time:26
Waiting Clients:
Queue 1: closed
Queue 2: closed
Time:27
Waiting Clients:
Queue 1: closed
Queue 2: closed
Time:28
Waiting Clients:
Queue 1: closed
Queue 2: closed
Time:29
Waiting Clients:
Queue 1: closed
Queue 2: closed
Time:30
Waiting Clients:
Queue 1: closed
Queue 2: closed
Time:31
Waiting Clients:
Queue 1: closed
Queue 2: closed
Time:32
Waiting Clients:
Queue 1: closed
Queue 2: closed
Time:33
Waiting Clients:
Queue 1: closed
Queue 2: closed
Time:34
Waiting Clients:
Queue 1: closed
Queue 2: closed
Time:35
Waiting Clients:
Queue 1: closed
Queue 2: closed
Time:36
Waiting Clients:
Queue 1: closed
Queue 2: closed
Time:37
Waiting Clients:
Queue 1: closed
Queue 2: closed
Time:38
Waiting Clients:
Queue 1: closed
Queue 2: closed
Time:39
Waiting Clients:
Queue 1: closed
Queue 2: closed
Time:40
Waiting Clients:
Queue 1: closed
Queue 2: closed
Time:41
Waiting Clients:
Queue 1: closed
Queue 2: closed
Time:42
Waiting Clients:
Queue 1: closed
Queue 2: closed
Time:43
Waiting Clients:
Queue 1: closed
Queue 2: closed
Time:44
Waiting Clients:
Queue 1: closed
Queue 2: closed
Time:45
Waiting Clients:
Queue 1: closed
Queue 2: closed
Time:46
Waiting Clients:
Queue 1: closed
Queue 2: closed
Time:47
Waiting Clients:
Queue 1: closed
Queue 2: closed
Time:48
Waiting Clients:
Queue 1: closed
Queue 2: closed
Time:49
Waiting Clients:
Queue 1: closed
Queue 2: closed
Time:50
Waiting Clients:
Queue 1: closed
Queue 2: closed
Time:51
Waiting Clients:
Queue 1: closed
Queue 2: closed
Time:52
Waiting Clients:
Queue 1: closed
Queue 2: closed
Time:53
Waiting Clients:
Queue 1: closed
Queue 2: closed
Time:54
Waiting Clients:
Queue 1: closed
Queue 2: closed
Time:55
Waiting Clients:
Queue 1: closed
Queue 2: closed
Time:56
Waiting Clients:
Queue 1: closed
Queue 2: closed
Time:57
Waiting Clients:
Queue 1: closed
Queue 2: closed
Time:58
Waiting Clients:
Queue 1: closed
Queue 2: closed
Time:59
Waiting Clients:
Queue 1: closed
Queue 2: closed
Time:60
Waiting Clients:
Queue 1: closed
Queue 2: closed
```

➢ Test 2




Raport

DATA ENTERED BY USERS:

Number of Clients:	50
Number of Queues:	5
Simulation Interval:	60
Minimum arrival time:	2
Maximum arrival time:	40
Minimum service time:	1
Maximum service time:	7

RESULTS:

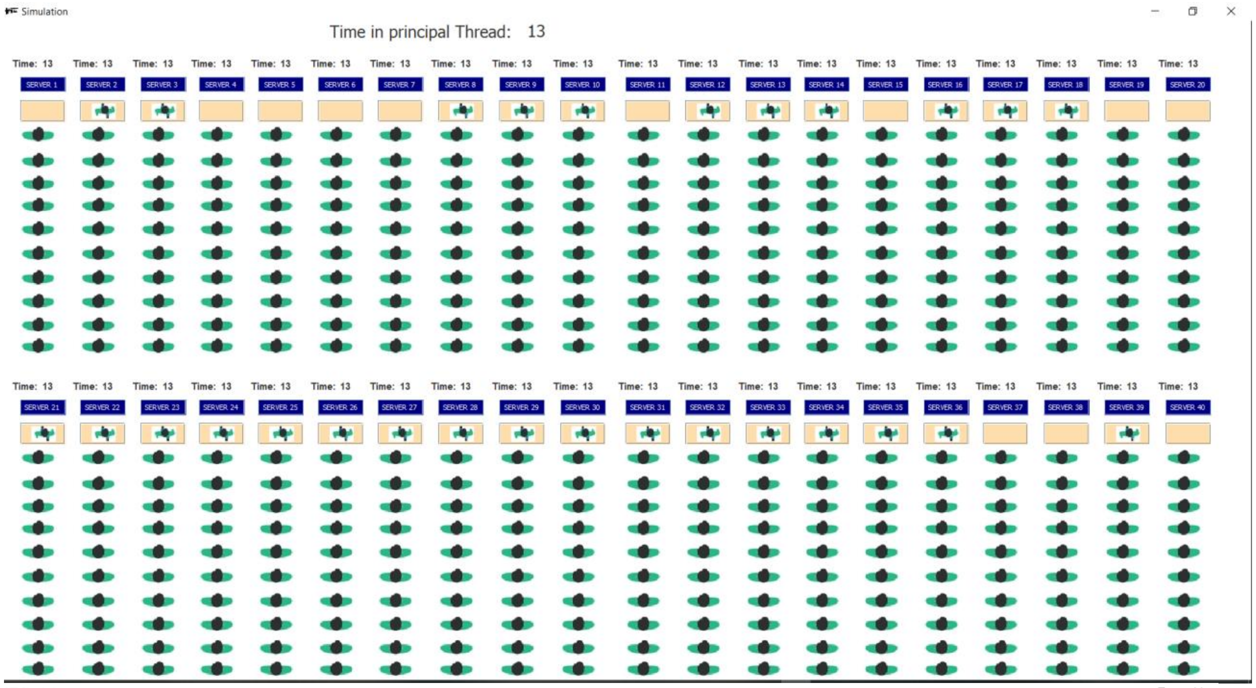
Average waiting time:	3.68
Average service time:	16
Peak hour:	21

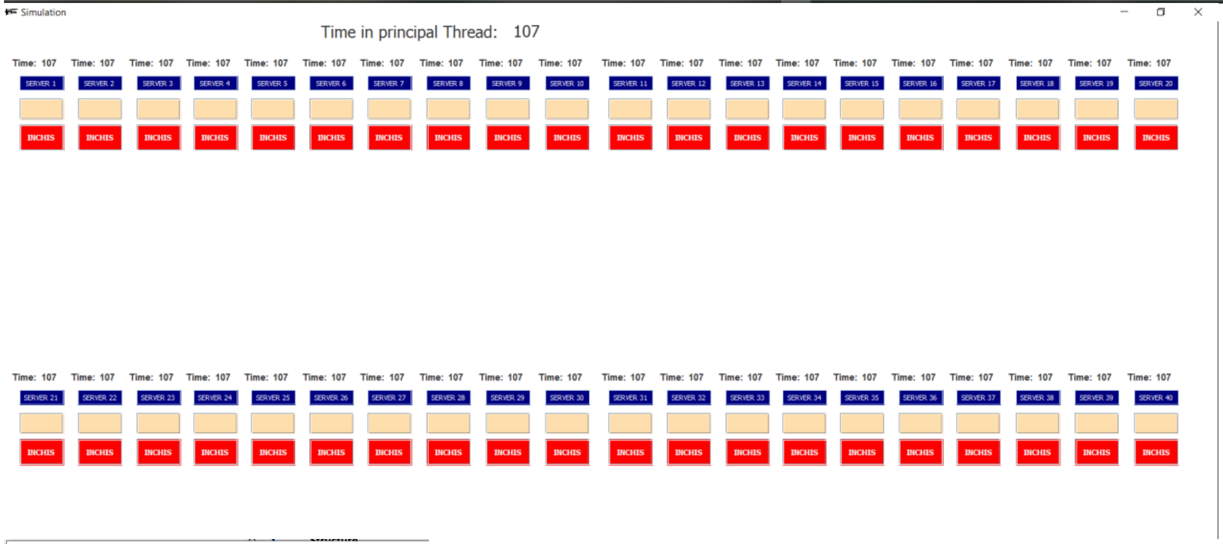
Log coada 1:
client 10 a ajuns la ora 2 si a asteptat 4 secunde si a plecat la 6
client 13 a ajuns la ora 9 si a asteptat 6 secunde si a plecat la 15
client 26 a ajuns la ora 16 si a asteptat 2 secunde si a plecat la 18
client 2 a ajuns la ora 18 si a asteptat 5 secunde si a plecat la 23
client 1 a ajuns la ora 26 si a asteptat 3 secunde si a plecat la 29
client 6 a ajuns la ora 31 si a asteptat 6 secunde si a plecat la 37
client 3 a ajuns la ora 37 si a asteptat 2 secunde si a plecat la 39
client 7 a ajuns la ora 39 si a asteptat 3 secunde si a plecat la 42
Log coada 2:
client 36 a ajuns la ora 3 si a asteptat 2 secunde si a plecat la 5
client 14 a ajuns la ora 5 si a asteptat 2 secunde si a plecat la 7
client 20 a ajuns la ora 7 si a asteptat 6 secunde si a plecat la 13
client 24 a ajuns la ora 14 si a asteptat 3 secunde si a plecat la 17
client 4 a ajuns la ora 19 si a asteptat 4 secunde si a plecat la 23
client 35 a ajuns la ora 27 si a asteptat 2 secunde si a plecat la 29
client 8 a ajuns la ora 31 si a asteptat 6 secunde si a plecat la 37

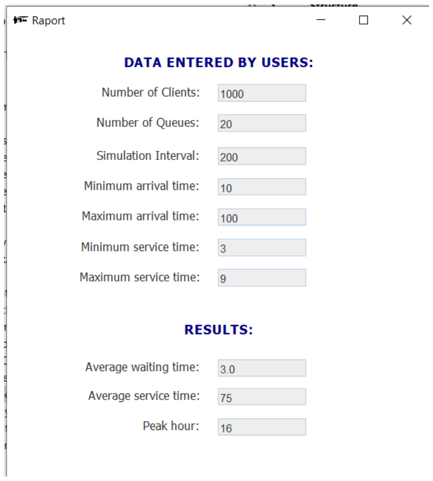
```
Log coada 3:
client 29 a ajuns la ora 2 si a asteptat 6 secunde si a plecat la 8
client 37 a ajuns la ora 8 si a asteptat 2 secunde si a plecat la 10
client 22 a ajuns la ora 12 si a asteptat 1 secunde si a plecat la 13
client 43 a ajuns la ora 14 si a asteptat 3 secunde si a plecat la 17
client 5 a ajuns la ora 19 si a asteptat 5 secunde si a plecat la 24
client 12 a ajuns la ora 28 si a asteptat 5 secunde si a plecat la 33
client 17 a ajuns la ora 34 si a asteptat 3 secunde si a plecat la 37
Log coada 4:
client 41 a ajuns la ora 2 si a asteptat 2 secunde si a plecat la 4
client 25 a ajuns la ora 8 si a asteptat 5 secunde si a plecat la 13
client 9 a ajuns la ora 18 si a asteptat 4 secunde si a plecat la 22
client 16 a ajuns la ora 22 si a asteptat 5 secunde si a plecat la 27
client 44 a ajuns la ora 29 si a asteptat 4 secunde si a plecat la 33
client 21 a ajuns la ora 34 si a asteptat 1 secunde si a plecat la 35
Log coada 5:
client 31 a ajuns la ora 4 si a asteptat 6 secunde si a plecat la 10
client 39 a ajuns la ora 12 si a asteptat 1 secunde si a plecat la 13
client 11 a ajuns la ora 19 si a asteptat 2 secunde si a plecat la 21
client 18 a ajuns la ora 21 si a asteptat 1 secunde si a plecat la 22
client 23 a ajuns la ora 22 si a asteptat 6 secunde si a plecat la 28
client 46 a ajuns la ora 30 si a asteptat 2 secunde si a plecat la 32
client 28 a ajuns la ora 33 si a asteptat 4 secunde si a plecat la 37
Log coada 6:
client 40 a ajuns la ora 7 si a asteptat 4 secunde si a plecat la 11
client 48 a ajuns la ora 11 si a asteptat 6 secunde si a plecat la 17
client 15 a ajuns la ora 19 si a asteptat 2 secunde si a plecat la 21
client 27 a ajuns la ora 22 si a asteptat 5 secunde si a plecat la 27
client 33 a ajuns la ora 33 si a asteptat 4 secunde si a plecat la 37
Log coada 7:
client 45 a ajuns la ora 6 si a asteptat 4 secunde si a plecat la 10
client 49 a ajuns la ora 10 si a asteptat 3 secunde si a plecat la 13
client 19 a ajuns la ora 19 si a asteptat 2 secunde si a plecat la 21
client 30 a ajuns la ora 21 si a asteptat 5 secunde si a plecat la 26
Log coada 8:
client 50 a ajuns la ora 9 si a asteptat 4 secunde si a plecat la 13
client 32 a ajuns la ora 18 si a asteptat 3 secunde si a plecat la 21
Log coada 9:
client 34 a ajuns la ora 16 si a asteptat 5 secunde si a plecat la 21
Log coada 10:
client 47 a ajuns la ora 18 si a asteptat 2 secunde si a plecat la 20
client 38 a ajuns la ora 20 si a asteptat 6 secunde si a plecat la 26
Log coada 11:
client 42 a ajuns la ora 17 si a asteptat 5 secunde si a plecat la 22
```

Fisierul „simulation.txt” contine toate datele de mai sus, dar sub forma data in cerinta.

➢ Test 3






Raport

DATA ENTERED BY USERS:

Number of Clients:	1000
Number of Queues:	20
Simulation Interval:	200
Minimum arrival time:	10
Maximum arrival time:	100
Minimum service time:	3
Maximum service time:	9

RESULTS:

Average waiting time:	3.0
Average service time:	75
Peak hour:	16

Pentru a nu incarca prea mult documentatia, o secventa din ce s-a afisat in fisierul simularii e:

```
...
Time:16
Waiting
Clients:(1,17,3)(2,71,3)(4,44,3)(5,39,3)(7,77,3)(8,34,3)(9,28,3)(10,80,3)(11,46,3)(12,22,3)(13,45,3)(14,69,3)(15,61,3)(16,92,3)(1
7,58,3)(18,98,3)(19,37,3)(20,82,3)(21,91,3)(22,49,3)(23,66,3)(24,96,3)(25,63,3)(26,62,3)(27,57,3)(28,73,3)(29,57,3)(30,62,3)(32
,76,3)(33,92,3)(34,80,3)(35,39,3)(36,24,3)(37,97,3)(39,33,3)(40,48,3)(41,32,3)(42,83,3)(43,38,3)(45,67,3)(46,69,3)(47,92,3)(49,
58,3)(50,92,3)(51,26,3)(52,78,3)(53,74,3)(54,69,3)(55,94,3)(56,32,3)(57,42,3)(58,75,3)(59,24,3)(60,91,3)(61,91,3)(62,69,3)(63,3
6,3)(64,79,3)(65,59,3)(66,37,3)(67,60,3)(68,76,3)(69,52,3)(70,25,3)(72,91,3)(73,20,3)(74,24,3)(75,21,3)(76,96,3)(77,80,3)(78,59
,3)(79,68,3)(80,40,3)(81,34,3)(82,55,3)(84,97,3)(85,44,3)(86,19,3)(87,72,3)(90,79,3)(91,44,3)(92,91,3)(93,84,3)(94,64,3)(95,57,
3)(96,40,3)(97,76,3)(98,31,3)(99,48,3)(100,21,3)(101,44,3)(102,27,3)(104,66,3)(105,98,3)(106,25,3)(107,31,3)(108,72,3)(109,95
,3)(110,75,3)(111,71,3)(113,85,3)(114,99,3)(115,41,3)(117,53,3)(118,90,3)(119,57,3)(120,52,3)(121,74,3)(122,99,3)(123,47,3)(1
24,86,3)(126,52,3)(127,52,3)(128,97,3)(129,60,3)(130,64,3)(131,28,3)(132,81,3)(133,64,3)(134,39,3)(135,46,3)(136,43,3)(138,6
4,3)(139,77,3)(141,51,3)(142,45,3)(143,41,3)(144,35,3)(145,81,3)(146,91,3)(147,96,3)(148,70,3)(149,35,3)(150,56,3)(151,93,3)(
152,30,3)(153,36,3)(154,45,3)(155,42,3)(156,36,3)(157,23,3)(158,76,3)(159,38,3)(160,72,3)(161,28,3)(162,22,3)(163,21,3)(164,
68,3)(165,33,3)(166,62,3)(167,19,3)(168,85,3)(169,30,3)(170,82,3)(171,66,3)(172,72,3)(173,75,3)(174,82,3)(175,48,3)(176,92,3
)(177,75,3)(178,74,3)(179,95,3)(180,79,3)(181,79,3)(182,23,3)(183,64,3)(184,68,3)(185,89,3)(186,61,3)(187,29,3)(188,77,3)(18
9,33,3)(190,52,3)(191,18,3)(192,53,3)(193,70,3)(194,45,3)(195,30,3)(196,28,3)(197,27,3)(198,75,3)(199,28,3)(200,87,3)(201,92,
3)(202,70,3)(203,31,3)(204,70,3)(205,93,3)(206,57,3)(207,65,3)(208,21,3)(209,83,3)(210,18,3)(211,64,3)(212,48,3)(214,46,3)(2
16,50,3)(218,81,3)(219,61,3)(220,25,3)(221,71,3)(222,97,3)(223,22,3)(225,34,3)(226,81,3)(227,39,3)(228,23,3)(229,39,3)(230,9
2,3)(232,45,3)(233,80,3)(234,66,3)(235,38,3)(236,93,3)(237,67,3)(238,93,3)(239,20,3)(240,62,3)(241,63,3)(242,92,3)(243,63,3)(
244,26,3)(245,85,3)(246,63,3)(247,52,3)(248,87,3)(249,21,3)(250,38,3)(251,29,3)(252,37,3)(253,96,3)(254,87,3)(255,87,3)(256,
50,3)(257,48,3)(258,68,3)(260,31,3)(261,91,3)(263,88,3)(264,77,3)(265,32,3)(266,70,3)(267,52,3)(268,39,3)(269,17,3)(270,30,3
)(271,54,3)(272,23,3)(273,48,3)(274,91,3)(275,56,3)(276,84,3)(277,56,3)(280,53,3)(281,34,3)(282,66,3)(283,54,3)(284,32,3)(28
5,68,3)(286,98,3)(287,68,3)(289,89,3)(291,82,3)(292,17,3)(293,61,3)(294,41,3)(295,38,3)(296,98,3)(298,77,3)(300,72,3)(301,95,
3)(302,48,3)(303,69,3)(304,64,3)(305,73,3)(306,97,3)(307,38,3)(309,39,3)(310,82,3)(311,46,3)(312,42,3)(313,45,3)(314,29,3)(3
15,88,3)(316,40,3)(317,57,3)(318,18,3)(319,47,3)(320,29,3)(321,27,3)(322,25,3)(323,69,3)(324,94,3)(325,23,3)(326,65,3)(327,3
4,3)(328,59,3)(329,59,3)(330,47,3)(331,97,3)(333,97,3)(334,59,3)(335,59,3)(336,60,3)(337,90,3)(338,69,3)(339,34,3)(340,46,3)(
341,49,3)(342,81,3)(343,58,3)(344,22,3)(345,99,3)(346,24,3)(347,37,3)(348,55,3)(349,30,3)(350,23,3)(351,68,3)(352,90,3)(353,
21,3)(354,88,3)(355,44,3)(356,40,3)(357,37,3)(358,53,3)(359,85,3)(360,49,3)(361,92,3)(362,86,3)(363,21,3)(365,74,3)(366,23,3
)(367,64,3)(368,84,3)(369,34,3)(370,87,3)(371,68,3)(372,26,3)(373,74,3)(374,34,3)(375,23,3)(376,42,3)(377,28,3)(378,33,3)(37
9,71,3)(380,91,3)(381,26,3)(382,98,3)(383,75,3)(385,41,3)(386,62,3)(387,32,3)(388,52,3)(389,60,3)(392,36,3)(394,53,3)(395,60,
3)(396,21,3)(397,91,3)(398,34,3)(399,20,3)(400,17,3)(401,90,3)(402,60,3)(403,19,3)(404,77,3)(405,39,3)(406,31,3)(407,21,3)(4
08,36,3)(409,27,3)(411,38,3)(412,50,3)(413,76,3)(414,39,3)(415,61,3)(416,22,3)(417,39,3)(418,78,3)(419,52,3)(420,73,3)(421,3
4,3)(423,32,3)(424,43,3)(425,65,3)(426,56,3)(427,18,3)(429,42,3)(430,69,3)(431,94,3)(432,83,3)(433,80,3)(434,44,3)(435,39,3)(
436,91,3)(437,44,3)(438,79,3)(439,55,3)(440,20,3)(441,92,3)(442,58,3)(443,76,3)(444,17,3)(445,41,3)(446,61,3)(447,28,3)(448,
53,3)(449,32,3)(450,81,3)(451,38,3)(452,78,3)(453,75,3)(454,98,3)(455,24,3)(457,36,3)(458,54,3)(459,21,3)(460,84,3)(461,88,3
)(462,90,3)(463,53,3)(464,76,3)(465,91,3)(466,80,3)(467,85,3)(468,39,3)(469,43,3)(470,86,3)(471,36,3)(472,99,3)(473,33,3)(47
4,59,3)(475,62,3)(476,73,3)(477,31,3)(478,54,3)(479,36,3)(480,31,3)(481,99,3)(482,39,3)(483,89,3)(484,51,3)(485,68,3)(486,60,
3)(487,78,3)(488,21,3)(489,93,3)(490,37,3)(491,71,3)(492,96,3)(493,42,3)(495,69,3)(496,40,3)(497,89,3)(499,49,3)(500,83,3)(5
01,46,3)(502,99,3)(503,62,3)(504,52,3)(505,41,3)(506,23,3)(507,62,3)(508,35,3)(509,34,3)(511,97,3)(512,34,3)(513,19,3)(514,9
4,3)(515,67,3)(516,59,3)(517,51,3)(518,68,3)(520,86,3)(521,49,3)(522,75,3)(523,17,3)(524,42,3)(525,79,3)(526,29,3)(527,34,3)(
528,42,3)(529,28,3)(530,30,3)(531,55,3)(532,59,3)(533,50,3)(534,57,3)(535,40,3)(536,72,3)(537,22,3)(538,51,3)(539,85,3)(540,
74,3)(541,62,3)(542,88,3)(543,26,3)(545,40,3)(546,80,3)(547,48,3)(549,54,3)(550,57,3)(551,19,3)(552,92,3)(553,50,3)(554,75,3
)(555,46,3)(556,89,3)(557,38,3)(558,17,3)(559,88,3)(561,46,3)(562,43,3)(564,86,3)(566,58,3)(567,39,3)(568,62,3)(569,57,3)(57
0,51,3)(571,82,3)(572,70,3)(573,69,3)(574,71,3)(575,25,3)(577,83,3)(579,97,3)(580,51,3)(581,67,3)(583,76,3)(584,67,3)(585,31,
3)(586,86,3)(587,53,3)(588,33,3)(589,63,3)(590,48,3)(591,30,3)(592,59,3)(595,29,3)(596,66,3)(597,45,3)(598,30,3)(599,98,3)(6
00,91,3)(601,52,3)(602,82,3)(603,60,3)(604,96,3)(606,96,3)(607,87,3)(608,43,3)(609,49,3)(610,93,3)(611,27,3)(612,30,3)(613,3
8,3)(615,73,3)(616,96,3)(617,94,3)(618,65,3)(619,87,3)(620,51,3)(621,91,3)(622,81,3)(624,24,3)(625,93,3)(626,23,3)(627,63,3)(
628,70,3)(629,18,3)(630,95,3)(631,29,3)(633,72,3)(634,78,3)(635,26,3)(636,64,3)(637,67,3)(638,65,3)(639,98,3)(640,37,3)(641,
47,3)(642,31,3)(643,50,3)(644,75,3)(645,66,3)(646,23,3)(647,99,3)(648,84,3)(649,56,3)(650,52,3)(651,61,3)(652,58,3)(653,47,3
)(654,35,3)(656,97,3)(657,75,3)(658,50,3)(659,57,3)(660,75,3)(661,94,3)(662,78,3)(663,92,3)(664,63,3)(665,44,3)(666,96,3)(66
7,97,3)(668,35,3)(669,76,3)(670,98,3)(671,44,3)(672,19,3)(673,48,3)(674,25,3)(675,17,3)(676,46,3)(677,88,3)(678,61,3)(679,39,
3)(680,17,3)(681,54,3)(682,41,3)(683,64,3)(684,25,3)(685,76,3)(686,38,3)(687,29,3)(688,80,3)(689,71,3)(690,23,3)(691,85,3)(6
92,36,3)(693,42,3)(694,28,3)(695,58,3)(697,26,3)(698,49,3)(699,62,3)(700,53,3)(701,64,3)(702,92,3)(703,84,3)(704,35,3)(705,7
9,3)(706,64,3)(707,54,3)(708,32,3)(709,88,3)(710,59,3)(711,58,3)(712,84,3)(713,78,3)(714,40,3)(715,37,3)(716,17,3)(717,21,3)(
718,64,3)(719,46,3)(720,60,3)(721,50,3)(722,86,3)(723,61,3)(726,72,3)(727,61,3)(728,99,3)(729,31,3)(730,82,3)(731,45,3)(732,
41,3)(733,99,3)(734,95,3)(735,99,3)(736,92,3)(737,80,3)(738,84,3)(741,70,3)(744,24,3)(745,42,3)(746,22,3)(747,41,3)(748,78,3
)(749,67,3)(750,25,3)(751,54,3)(752,62,3)(753,90,3)(754,49,3)(755,71,3)(756,62,3)(758,89,3)(759,26,3)(760,90,3)(761,91,3)(76
2,74,3)(763,37,3)(764,38,3)(765,58,3)(766,28,3)(767,37,3)(769,36,3)(770,63,3)(771,68,3)(772,72,3)(773,34,3)(774,57,3)(775,28,
3)(776,89,3)(777,71,3)(778,18,3)(779,21,3)(780,67,3)(781,34,3)(782,62,3)(783,33,3)(784,75,3)(785,36,3)(787,86,3)(789,20,3)(7
90,66,3)(791,23,3)(792,59,3)(793,88,3)(794,34,3)(795,97,3)(796,32,3)(797,70,3)(798,74,3)(799,40,3)(800,28,3)(801,71,3)(802,8
7,3)(803,89,3)(804,31,3)(805,88,3)(806,94,3)(807,97,3)(808,95,3)(809,28,3)(810,64,3)(811,40,3)(812,46,3)(813,83,3)(814,75,3)(
815,72,3)(816,54,3)(817,41,3)(818,81,3)(820,93,3)(821,77,3)(822,17,3)(823,18,3)(824,58,3)(825,73,3)(827,70,3)(828,34,3)(829,
95,3)(830,33,3)(831,88,3)(832,46,3)(833,34,3)(834,26,3)(835,93,3)(836,75,3)(837,24,3)(838,21,3)(839,29,3)(840,46,3)(841,27,3
)(842,27,3)(845,97,3)(846,24,3)(847,55,3)(848,33,3)(849,34,3)(850,66,3)(852,65,3)(853,55,3)(854,83,3)(856,84,3)(857,81,3)(85
8,53,3)(859,21,3)(861,53,3)(862,95,3)(863,59,3)(864,97,3)(865,67,3)(866,19,3)(867,17,3)(868,47,3)(869,87,3)(870,82,3)(871,68,
3)(872,74,3)(873,21,3)(874,54,3)(875,72,3)(876,42,3)(877,29,3)(878,92,3)(879,60,3)(880,46,3)(881,29,3)(882,83,3)(883,70,3)(8
84,69,3)(885,91,3)(886,22,3)(887,69,3)(888,66,3)(889,95,3)(890,85,3)(891,70,3)(892,50,3)(893,49,3)(894,31,3)(895,55,3)(896,4
2,3)(897,53,3)(898,67,3)(899,74,3)(900,87,3)(901,82,3)(902,84,3)(903,70,3)(904,74,3)(905,71,3)(906,26,3)(907,43,3)(909,89,3)(
910,56,3)(911,67,3)(912,82,3)(913,86,3)(914,86,3)(915,90,3)(916,49,3)(917,19,3)(918,65,3)(919,67,3)(920,19,3)(921,30,3)(922,
21,3)(923,50,3)(924,47,3)(925,86,3)(926,74,3)(927,54,3)(928,85,3)(929,43,3)(931,90,3)(932,24,3)(933,55,3)(934,30,3)(935,17,3
)(936,82,3)(937,52,3)(938,31,3)(939,87,3)(940,57,3)(941,51,3)(942,86,3)(943,51,3)(945,75,3)(946,25,3)(947,54,3)(948,87,3)(94
9,57,3)(952,53,3)(953,76,3)(954,45,3)(955,27,3)(956,18,3)(957,24,3)(958,41,3)(959,25,3)(960,78,3)(961,46,3)(962,99,3)(963,20,
3)(964,25,3)(965,20,3)(966,98,3)(967,81,3)(968,77,3)(969,50,3)(970,59,3)(971,20,3)(972,47,3)(973,19,3)(974,96,3)(975,44,3)(9
76,81,3)(977,81,3)(978,28,3)(979,28,3)(980,99,3)(981,84,3)(982,50,3)(983,43,3)(984,19,3)(985,45,3)(986,84,3)(987,79,3)(988,9
8,3)(989,49,3)(990,44,3)(991,81,3)(992,44,3)(993,77,3)(994,99,3)(995,78,3)(996,77,3)(997,44,3)(998,99,3)(999,83,3)(1000,56,3
)
Queue 1: closed
Queue 2: (31,16,3)
Queue 3: (38,15,2)
Queue 4: (48,14,1)
Queue 5: (71,16,3)
Queue 6: (88,16,3)
Queue 7: (103,16,3)
Queue 8: (112,14,1)
Queue 9: (125,16,3)
Queue 10: (137,14,1)
Queue 11: (140,16,3)
Queue 12: (213,15,2)
Queue 13: (217,15,2)
Queue 14: (259,14,1)
Queue 15: (391,16,3)
Queue 16: (297,14,1)
Queue 17: (308,14,1)
Queue 18: (384,15,2)
Queue 19: (390,14,1)
Queue 20: (393,16,3)
Queue 21: (410,14,1)
Queue 22: (456,15,2)
Queue 23: (498,15,2)
Queue 24: (510,15,2)
Queue 25: (548,15,2)
Queue 26: (560,14,1)
Queue 27: (563,15,2)
Queue 28: (576,16,3)
Queue 29: (578,16,3)
Queue 30: (593,14,1)
Queue 31: (614,15,2)
Queue 32: closed
Queue 33: (740,16,3)
Queue 34: (742,15,2)
Queue 35: (743,16,3)
Queue 36: (788,16,3)
Queue 37: (819,14,1)
Queue 38: (843,14,1)
Queue 39: (851,16,3)
Queue 40: (855,14,1)
....
Time:101
Waiting Clients:
Queue 1: closed
Queue 2: (114,99,1)
Queue 3: closed
Queue 4: (122,99,1)
Queue 5: closed
Queue 6: closed
Queue 7: closed
Queue 8: (345,99,1)
Queue 9: closed
Queue 10: closed
Queue 11: (472,99,1)
Queue 12: closed
Queue 13: closed
Queue 14: closed
Queue 15: closed
Queue 16: closed
Queue 17: closed
Queue 18: (481,99,1)
Queue 19: (502,99,1)
Queue 20: closed
Queue 21: closed
Queue 22: closed
Queue 23: (647,99,1)
Queue 24: (728,99,1)
Queue 25: (733,99,1)
Queue 26: closed
Queue 27: closed
Queue 28: closed
Queue 29: closed
Queue 30: (735,99,1)
Queue 31: (962,99,1)
Queue 32: closed
Queue 33: closed
Queue 34: closed
Queue 35: (980,99,1)
Queue 36: closed
Queue 37: closed
Queue 38: (994,99,1)
Queue 39: closed
Queue 40: (998,99,1)
```

O secventa dintr-un log-urile cozilor (al cozii 40):

```
client 855 a ajuns la ora 14 si a asteptat 3 secunde si a plecat la 17
client 935 a ajuns la ora 17 si a asteptat 3 secunde si a plecat la 20
client 922 a ajuns la ora 21 si a asteptat 3 secunde si a plecat la 24
client 964 a ajuns la ora 25 si a asteptat 3 secunde si a plecat la 28
client 979 a ajuns la ora 28 si a asteptat 3 secunde si a plecat la 31
client 938 a ajuns la ora 31 si a asteptat 3 secunde si a plecat la 34
client 849 a ajuns la ora 34 si a asteptat 3 secunde si a plecat la 37
client 767 a ajuns la ora 37 si a asteptat 3 secunde si a plecat la 40
client 811 a ajuns la ora 40 si a asteptat 3 secunde si a plecat la 43
client 997 a ajuns la ora 44 si a asteptat 3 secunde si a plecat la 47
client 989 a ajuns la ora 49 si a asteptat 3 secunde si a plecat la 52
client 952 a ajuns la ora 53 si a asteptat 3 secunde si a plecat la 56
client 1000 a ajuns la ora 56 si a asteptat 3 secunde si a plecat la 59
client 970 a ajuns la ora 59 si a asteptat 3 secunde si a plecat la 62
client 782 a ajuns la ora 62 si a asteptat 3 secunde si a plecat la 65
client 919 a ajuns la ora 67 si a asteptat 3 secunde si a plecat la 70
client 903 a ajuns la ora 70 si a asteptat 3 secunde si a plecat la 73
client 953 a ajuns la ora 76 si a asteptat 3 secunde si a plecat la 79
client 986 a ajuns la ora 84 si a asteptat 3 secunde si a plecat la 87
client 948 a ajuns la ora 87 si a asteptat 3 secunde si a plecat la 90
client 885 a ajuns la ora 91 si a asteptat 3 secunde si a plecat la 94
client 889 a ajuns la ora 95 si a asteptat 3 secunde si a plecat la 98
client 998 a ajuns la ora 99 si a asteptat 3 secunde si a plecat la 102
```

6.CONCLUZII

Prin urmatorii pasi:

- Stabilirea problemei
- Desenare schema defalcata pe structura MVC
- Implementare clase si metode cat mai divizat, in pachetele business, data, gui si strategy.
- Trasare interfata grafica
- Schitarea diagramei UML
- Testare

Am reusit sa implemntez tema ceruta.

Pe viitor acest program ar putea fi dezvoltat prin adaugarea mai multor cozi disponibile, pentru a permite o disponibilitate mai mare a valorilor de introdus, poate fi dezvoltat pentru o altfel de repartizare a clientilor si pentru a afisa mai multe date ca rezultat in urma simularii.

7.BIBLIOGRAFIE

- ASSIGNMENT_2_SUPPORT_PRESENTATION
- https://www.codejava.net/java-se/file-io/how-to-read-and-write-text-file-in-java
- https://www.tutorialspoint.com/java/java_multithreading.htm