DOCUMENTATIE

TEMA *2*

NUME STUDENT: Boitor Diana-Elena
GRUPA: 30222

CUPRINS

1. Obiectivul temei....3
2. Analiza problemei, modelare, scenarii, cazuri de utilizare 3
3. Proiectare 4
4. Implementare 5
5. Rezultate 11
6. Concluzii....113
7. Bibliografie 113

1. Obiectivul temei

Scopul principal al temei este să dezvolte o aplicație care să simuleze sisteme bazate pe cozi de așteptare. Această simulare implică sosirea și servirea unei serii de clienți, aceștia fiind plasați în cozi pentru a aștepta serviciul. După ce sunt serviți, clienții părăsesc cozile. De altfel se calculeaza și timpul mediu de așteptare al clienților, timpul mediu de servire și se identifica ora de vârf din timpul simulării.

Obiectivele secundare ale temei sunt urmatoarele:
1) Analiza problemei si identificarea cerintelor
2) Proiectarea aplicatiei de simulare
3) Implementarea aplicatiei de simulare
4) Testarea aplicatiei de simulare

2. Analiza problemei, modelare, scenarii, cazuri de utilizare

Cerințele funcționale sunt:

- Aplicatia de simulare ar trebui sa permita utilizatorilor sa introduca datele pentru simulare
- Aplicatia de simulare permite utilizatorului sa aleaga intre cele 2 strategii disponibile
- Aplicatia de simulare ar trebui sa efectueze simularea aleasa
- Aplicatia de simulare ar trebui sa afiseze cozile in timp real

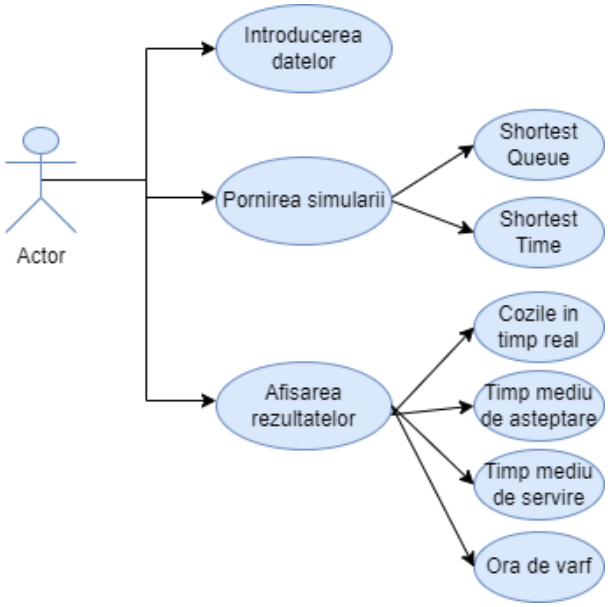
Cerințele nefuncționale sunt:

- Aplicatia de simulare este intuitiva și ușor de folosit de către utilizator
- Aplicatia de simulare are o interfață grafică bine structurată și plăcută visual

Cazuri de utilizare:

- Utilizatorul introduce de la tastatură datele necesare inceperii simularii : numarul de client, numarul de cozi, durata simularii, timpul minim si maxim de ajungere al unui client si timpul minim si maxim de servire al unui client
- Utilizatorul selectează apoi una din cele 2 strategii de simulare posibile
    - Shortest Time
    - Shortest Queue
- Calculatorul efectuează simularea si afiseaza in timp real evolutia cozilor

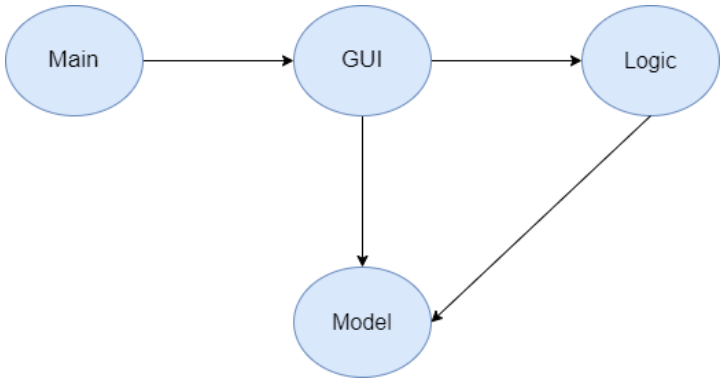
Diagrama de use-case



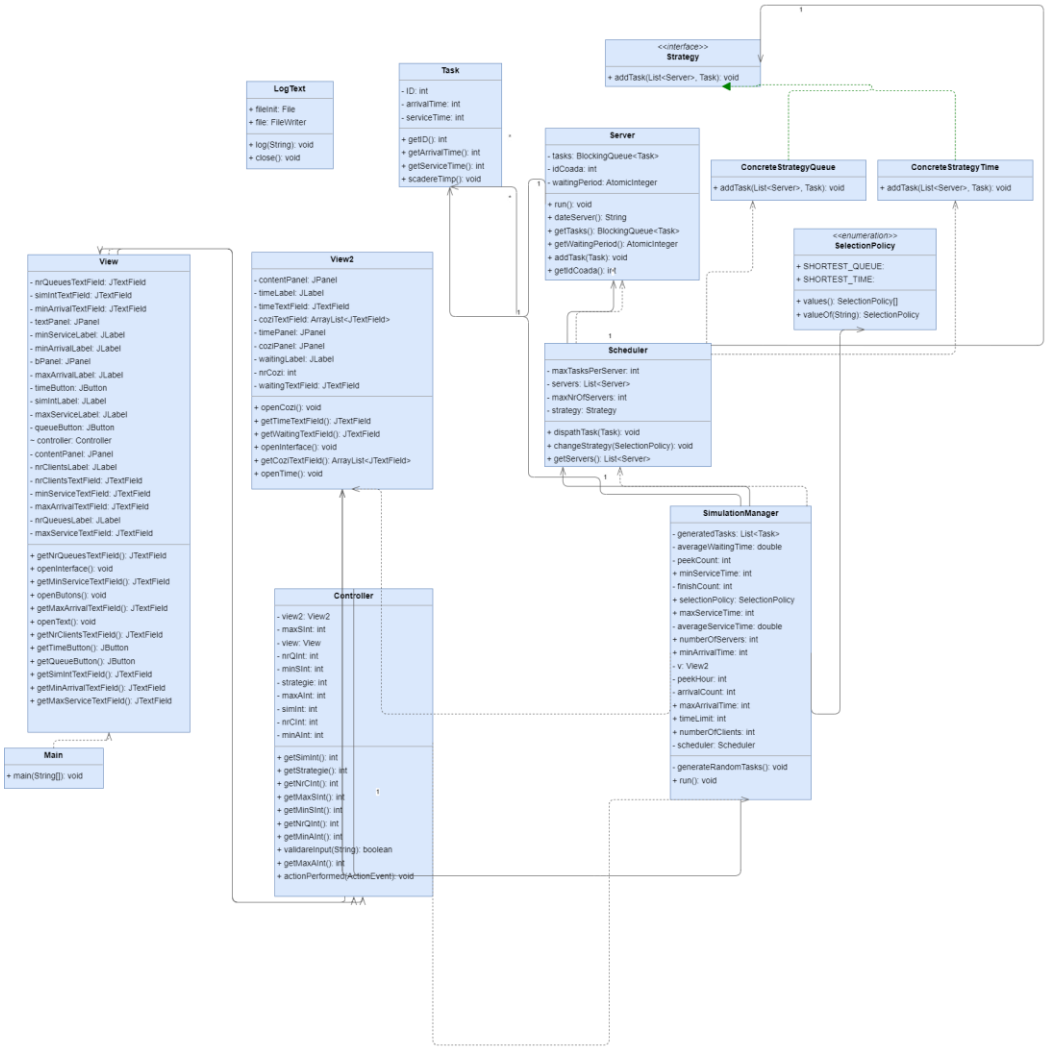
3. Proiectare

Structurile de date folosite sunt: LinkedList<>, BlockingQueue<>(), AtomicInteger.

- Diagrama pachetelor:



- Diagrama claselor:



4. Implementare

Aplicatia este realizata cu ajutorul a 11 clase, 1 interfata si 1 enum care sunt structurate in 3 pachete astfel:

- Pachetul Model: Server, Task, LogText
- Pachetul Logic: ConcreteStrategyTime, ConcreteStrategyQueue, Scheduler, SimulationManager, Strategy, SelectionPolicy
- Pachetul GUI: Controller, View, View2

➔ Clasa Server

Clasa Server este responsabilă pentru gestionarea sarcinilor primite și executarea acestora într-un mediu paralel.

- Variabile:
    - tasks: O coadă de tipul BlockingQueue<Task> care stochează sarcinile care trebuie procesate de către server.
    - waitingPeriod: Un contor de tipul AtomicInteger ce stocheaza timpul total de așteptare al sarcinilor în coadă.
    - idCoada: Un câmp static care reprezintă un identificator unic pentru server
- Metode:
    - Server(): Constructorul clasei care inițializează coada de sarcini și contorul de așteptare.
    - addTask(Task newTask): Adaugă o nouă sarcină în coada serverului și actualizează timpul total de așteptare.
    - run(): Metoda care rulează într-un fir de execuție separat și preia sarcinile din coadă pentru a le procesa. Cat timp avem clienti neprocesati ii luam pe rand si ii procesam pana cand le indeplinim serviceTime ul fiecaruia.
    - dateServer(): Returnam sub forma de text starea serverului si toti clientii ce sunt in asteptare (daca sunt).

➔ Clasa Task

Clasa Task reprezintă o sarcină care trebuie să fie procesată de către un server.

- Variabile:
    - arrivalTime: Reprezintă momentul de timp la care sarcina sosește pentru procesare.
    - serviceTime: Durata necesară pentru a finaliza sarcina.
    - ID: Identificatorul unic al sarcinii.
- Metode:
    - Task(int ID, int arrivalTime, int serviceTime): Constructorul clasei care inițializează variabilele de bază ale sarcinii.
    - scadereTimp(): Scade timpul de serviciu al sarcinii cu o unitate de timp. Este folosită pentru simularea progresului în timp al unei sarcini

➔ Clasa LogText

Clasa LogText oferă funcționalitatea de a înregistra mesaje într-un fișier de jurnal (log).

- Variabile:
  - fileInit: Reprezintă obiectul care indică către fișierul de jurnal. Este un fișier cu numele "log.txt".
  - file: Reprezintă obiectul FileWriter care este utilizat pentru a scrie în fișierul de jurnal.

- Blocul static: Inițializează fișierul de jurnal și obiectul FileWriter în momentul încărcării clasei. În cazul în care apare o excepție la crearea fișierului, aceasta va fi afișată.

- Metode:
  - log(String message): Este utilizată pentru a înregistra un mesaj în fișierul de jurnal. Mesajul este urmat de un caracter newline pentru a separa mesajele. Se utilizează blocuri de cod sincronizate pentru a asigura accesul concurent corect la fișier.
  - close(): Este utilizată pentru a închide fișierul de jurnal atunci când este necesar. De asemenea, este sincronizată pentru a evita problemele de concurență.

➔ Clasa ConcreteStrategyTime

Clasa ConcreteStrategyTime implementează interfața Strategy și reprezintă o strategie de adăugare a unei sarcini în coada de așteptare a unui server, luând în considerare timpul minim de așteptare al serverelor disponibile.

Metoda addTask(List<Server> servers, Task newTask): Această metodă implementează logica de adăugare a unei sarcini (newTask) în coada de așteptare a unui server din lista de servere (servers). Ea parcurge fiecare server din listă și determină serverul cu cel mai mic timp de așteptare. Sarcina este adăugată în coada de așteptare a serverului identificat.

➔ Clasa ConcreteStrategyQueue

Clasa ConcreteStrategyQueue implementează interfața Strategy și reprezintă o strategie de adăugare a unei sarcini în coada de așteptare a unui server, luând în considerare dimensiunea minimă a cozii de așteptare a serverelor disponibile.

Metoda addTask(List<Server> servers, Task newTask): Această metodă implementează logica de adăugare a unei sarcini (newTask) în coada de așteptare a unui server din lista de servere (servers). Ea parcurge fiecare server din listă și determină serverul cu cea mai mică dimensiune a cozii de așteptare. Sarcina este adăugată în coada de așteptare a serverului identificat.

➔ Clasa Scheduler

Clasa Scheduler coordonează distribuirea sarcinilor către servere utilizând o anumită strategie, definită de obiectul strategy.

- Variabile:
  - servers: O listă de servere disponibile pentru distribuirea sarcinilor.
  - maxNrOfServers: Numărul maxim de servere care pot fi create în cadrul planificatorului.
  - maxTasksPerServer: Numărul maxim de sarcini pe care un server îl poate gestiona simultan.
  - strategy: Obiectul care definește strategia de distribuire a sarcinilor către servere.

- Metodele:
  - Constructorul Scheduler(int maxNrOfServers, int maxTasksPerServer): Inițializează planificatorul cu numărul maxim de servere și numărul maxim de sarcini per server. În constructor, sunt create și pornite firele de execuție pentru fiecare server.
  - Metoda changeStrategy(SelectionPolicy policy): Schimbă strategia planificatorului în funcție de politica specificată. Dacă politica este SHORTEST_QUEUE, se utilizează o strategie bazată pe cea mai mică coadă de așteptare, iar dacă politica este SHORTEST_TIME, se utilizează o strategie bazată pe cel mai scurt timp.
  - Metoda dispatchTask(Task task): Distribuie o sarcină către unul dintre servere utilizând strategia curentă.
  - Metoda getServers(): Returnează lista de servere disponibile în cadrul planificatorului.

➔ Clasa SimulationManager

Clasa SimulationManager coordonează simularea și gestionarea sistemului bazat pe cozi de așteptare. Această clasă reprezintă componenta centrală a simulării, care coordonează interacțiunea între obiectele Scheduler, Server, Task și interfața grafică.

- Variabilele:
  - timeLimit: Limita de timp pentru simulare.
  - maxServiceTime: Timpul maxim de serviciu al unei sarcini.
  - minServiceTime: Timpul minim de serviciu al unei sarcini.
  - maxArrivalTime: Timpul maxim de sosire al unei sarcini.
  - minArrivalTime: Timpul minim de sosire al unei sarcini.
  - numberOfServers: Numărul total de servere disponibile.
  - numberOfClients: Numărul total de clienți.
  - selectionPolicy: Politica de selecție a serverelor.

- scheduler: Obiectul Scheduler responsabil pentru gestionarea distribuirii sarcinilor.
- generatedTasks: Lista sarcinilor generate aleatoriu.

- Metodele:
  - Constructorul SimulationManager(Controller c): Inițializează simularea cu parametrii furnizați de către obiectul Controller. Generează sarcini aleatorii și începe simularea prin crearea unui fir de execuție
  - Metoda generateRandomTasks(): Generează sarcini aleatorii cu parametrii specificați și le sortează crescator în funcție de timpul de sosire.
  - Metoda run(): Realizează simularea efectivă, gestionând sosirea și servirea sarcinilor, actualizând interfața grafică și înregistrând diverse statistici în fișierul de jurnal (log.txt). Metoda începe prin inițializarea variabilei currentTime la zero, reprezentând momentul inițial al simulării. Apoi, intră într-o buclă while care rulează până când timpul simulării atinge limita specificată în timeLimit. Se analizează lista de sarcini generate și se identifică sarcinile care trebuie încă servite. Sarcinile sunt distribuite către servere atunci când acestea ajung la momentul de sosire specificat. Metoda parcurge fiecare server și actualizează starea sarcinilor aflate în coada sa, reducând timpul de serviciu rămas pentru fiecare sarcină. De asemenea, înregistrează momentul de terminare al sarcinilor servite și elimină sarcinile finalizate din cozi. În timpul simulării, sunt calculate diverse statistici, cum ar fi timpul mediu de așteptare, timpul mediu de servire și ora de vârf. Aceste statistici sunt înregistrate și afișate în fișierul de jurnal. La finalul fiecărei iterații a buclei, metoda avansează timpul simulării cu o unitate de timp și apoi așteaptă o scurtă perioadă de timp (1 secundă) pentru a simula trecerea timpului în mod realist.

➔ Interfata Strategy

Interfața Strategy definește un contract pentru strategiile utilizate în gestionarea sarcinilor în cadrul simulatorului bazat pe cozi de așteptare.

Metoda addTask: Această metodă este responsabilă pentru adăugarea unei sarcini într-unul dintre serverele disponibile. Ea primește ca parametri o listă de servere și o sarcină care trebuie adăugată. Implementarea acestei metode variază în funcție de strategia specifică utilizată

➔ Clasa Controller

Acesta este Controller-ul din aplicație, care gestionează interacțiunea între interfața grafică și logica de simulare a sistemului de cozi.

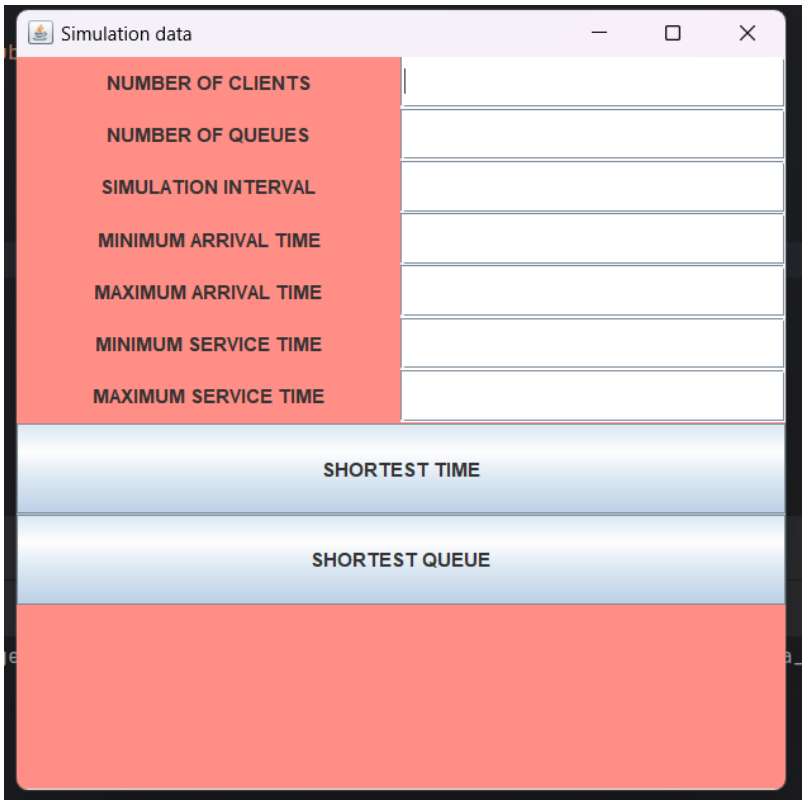
- Variabile:
  - view și view2: Referințe către interfețele grafice ale aplicației.

- nrCInt, nrQInt, simInt, minAInt, maxAInt, minSInt, maxSInt, strategie: Variabile pentru a stoca valorile introduse de utilizator pentru configurarea simulării.

- Metode:
  - validareInput(String text): O metodă pentru a valida dacă textul introdus este format din cifre.
  - actionPerformed(ActionEvent e): Implementarea metodei din interfața ActionListener. Această metodă gestionează evenimentele generate de apăsarea butoanelor din interfața utilizator și inițiază simularea în funcție de butonul apăsat și datele introduse de utilizator.

➔ Clasa View

Această clasă reprezintă interfața grafică a aplicației și include o serie de componente Swing pentru a permite utilizatorului să introducă parametrii simulării și să inițieze simularea.



➔ Clasa View2

Această clasă reprezintă interfața grafică pentru a afișa starea simulării in timp real, inclusiv timpul curent și cozile de așteptare asociate fiecărui server.

➔ Clasa Main

Aceasta clasă conține o metodă statică “main” care reprezintă punctul de intrare al aplicației. În interiorul metodei "main" se creează și deschide fereastra principală pentru aplicația Java.

5. Rezultate

Am rulat cele 3 teste si le am stocat rezultatele in fisierele test1.txt, test2.txt si test3.txt.

- Testul 1 :

  N = 4, Q = 2, $tsimulation\ MAX$ = 60 seconds

  [$tarrival\ MIN$,$tarrival\ MAX$] = [2, 30]

  [$tservice\ MIN$,$tservice\ MAX$]= [2, 4]

```
Clientii:

id: 2 arrival time: 5 service time: 3
id: 0 arrival time: 15 service time: 3
id: 3 arrival time: 21 service time: 2
id: 1 arrival time: 25 service time: 2

Time: 0
Waiting clients:
(2 5 3)
(0 15 3)
(3 21 2)
(1 25 2)
Queue 1: closed
Queue 2: closed
```

```
Time: 60
Waiting clients:
Queue 1: closed
Queue 2: closed
Average service time : 2.5
Average waiting time : 2.5
Peek hour: 5
```

- Testul 2:

  N = 50, Q = 5, $tsimulation\ MAX$ = 60 seconds

  [$tarrival\ MIN$,$tarrival\ MAX$] = [2, 40]

  [$tservice\ MIN$,$tservice\ MAX$]= [1, 7]

```
Clientii:

id: 34 arrival time: 5 service time: 6
id: 33 arrival time: 6 service time: 3
id: 8 arrival time: 8 service time: 2
id: 10 arrival time: 8 service time: 4
id: 29 arrival time: 8 service time: 4
id: 3 arrival time: 9 service time: 3
id: 48 arrival time: 9 service time: 4
id: 15 arrival time: 10 service time: 6
id: 41 arrival time: 10 service time: 2
id: 6 arrival time: 11 service time: 1
id: 19 arrival time: 11 service time: 3
id: 12 arrival time: 12 service time: 1
id: 44 arrival time: 12 service time: 4
id: 35 arrival time: 13 service time: 4
id: 32 arrival time: 14 service time: 4
id: 40 arrival time: 14 service time: 5
id: 49 arrival time: 15 service time: 4
id: 20 arrival time: 16 service time: 2
id: 7 arrival time: 17 service time: 3
```

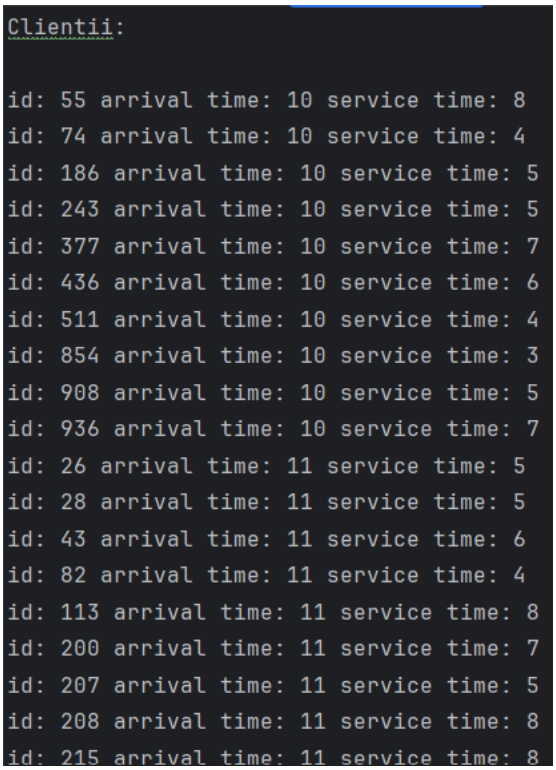
```
Time: 60
Waiting clients:
Queue 1: closed
Queue 2: closed
Queue 3: closed
Queue 4: closed
Queue 5: closed
Average service time : 3.36
Average waiting time : 4.52
Peek hour: 11
```

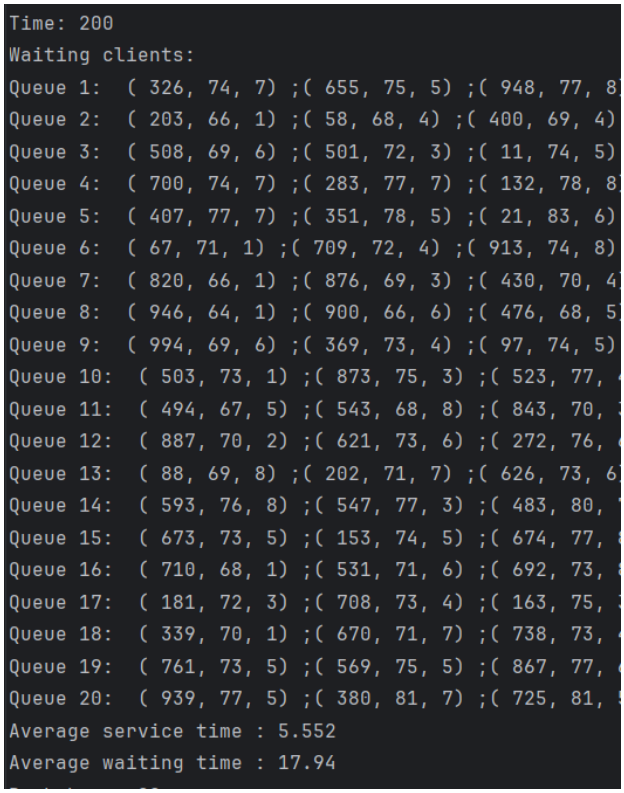
- Testul 3:

  N = 1000, Q = 20, $tsimulation\ MAX$ = 200 seconds

  [$tarrival\ MIN$,$tarrival\ MAX$] = [10, 100]

[$tservice\ MIN$,$tservice\ MAX$]= [3, 9]





6. Concluzii

În concluzie, implementarea acestei aplicații m-a ajutat sa imi aprofundez cunostintele legate de Thread-uri si dezvoltarea aptitudinii de a lucra cu acestea .

7. Bibliografie

1. *https://dsrl.eu/courses/pt/materials/PT_2024_A2_S1.pdf*
2. *https://dsrl.eu/courses/pt/materials/PT_2024_A2_S2.pdf*
3. *https://www.geeksforgeeks.org/java-threads/*