

# **DOCUMENTATIE**

## **TEMA 2**

NUME STUDENT : Cojocaru Ana-Cătălina  
GRUPA: 30226

# CUPRINS

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

## 1. Obiectivul temei

Scopul temei este de a simula un sistem de management de cozi. Sunt generați  $N$  clienți unici, cu un număr random de timp de sosire la coadă și un număr random ce reprezintă timpul în care acesta se va afla la una dintre cozi.

Managementul sistemelor bazate pe cozi este interesat de minimizarea timpului de așteptare al clienților lor înainte de a fi serviți. Aplicația de gestionare a cozilor ar trebui să simuleze (prin definirea unui timp de simulare *tsimulation*) o serie de  $N$  clienți care vin pentru serviciu, intră în  $Q$  cozi, așteaptă, sunt serviți și, în cele din urmă, părăsesc cozile. Toți clienții sunt generați când simularea este pornită și sunt caracterizați prin trei parametri: ID (un număr între 1 și  $N$ ), *tarrival* (timpul de simulare când sunt pregătiți să intre în coadă) și *tService* (intervalul de timp sau durata necesară pentru a servi clientul; adică timpul de așteptare când clientul este în fața cozii). Aplicația urmărește timpul total petrecut de fiecare client în cozi și calculează timpul mediu de așteptare. Fiecare client este adăugat la coadă cu timpul minim de așteptare când timpul său de sosire este mai mare sau egal cu timpul de simulare ( $tarrival \geq tsimulation$ ). Următoarele date ar trebui considerate drept date de intrare pentru aplicație și ar trebui introduse de utilizator în interfața utilizatorului aplicației:

Numărul de clienți ( $N$ );

Numărul de cozi ( $Q$ );

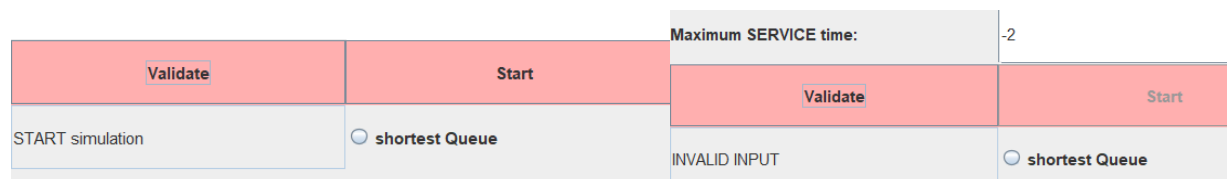
Intervalul de simulare


Timpul minim și maxim de sosire

Timpul minim și maxim de servire

IMPLEMENTAREA GUI	Cap 2
PROIECTARE+ DIAGrame UML	Cap 3
FOLOSIREA THREAD-URILOR	Cap 4 (Implementare)
TESTAREA	Cap 5

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




**QUEUES**

Number of clients:

10

Number of queues

Simulation TIME

10

Minimum ARRIVAL time:

Maximum ARRIVAL time:

10

Minimum SERVICE time:

Maximum SERVICE time:

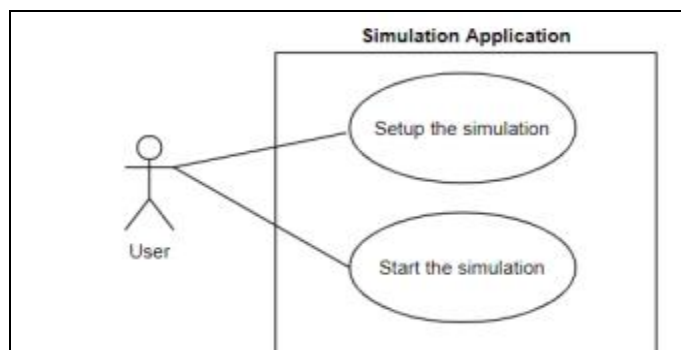
10

Validate

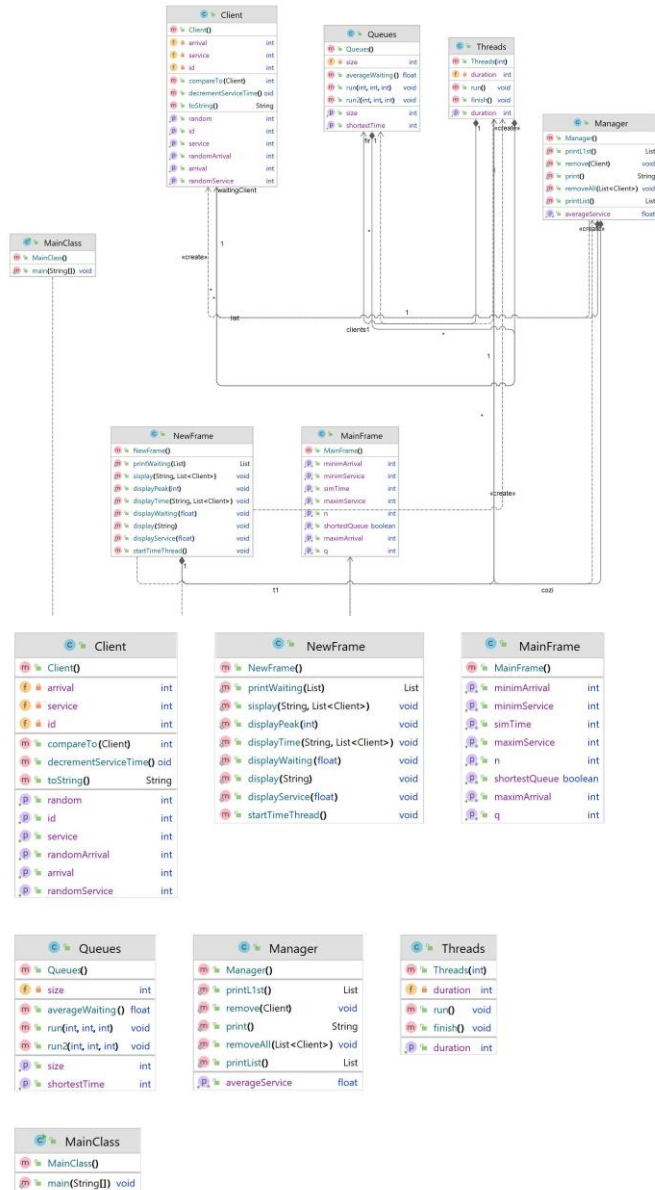
Start

☐ shortest Queue

*Sistemul este functional in toate cazurile, inclusive daca utilizatorul vrea sa aleaga ShortestQueue Strategy in loc de ShortestTime. Daca datele date nu sunt valide se va afisa un mesaj "INVALID INPUT", iar butonul de start va fi blocat. Altfel, simularea poate sa inceapa. Sistemul functioneaza pe ambele strategii, cu un numar mare de client.*



### 3. Proiectare



Programul este alcatuit din trei pachete: pachetul GUI care include clasele ce țin de interfața cu utilizatorul, pachetul de threads și cel unde se găsesc restul claselor ce țin de managementul cozilor. Clasele MainFrame și NewFrame creează cele două ferestre, clasa Main apelează constructorul unui obiect din MainFrame. Clasa Client generează clientii iar clasa Manager îi manageriază. Cozile sunt create (initializate), fiecare având un ArrayList separat cu clientii respectivi și un List sincronizat cu celalalte cozi pentru WaitingClients. Structurile de date folosite sunt sincronizate.

## 4. Implementare

Interfata cu utilizatorul e ușor de folosit, fiecare camp ce trebuie completat va fi verificat într-un bloc de try{} catch{} pentru validarea datelor. Aceste date sunt trimise la threaduri. Thread-ul de timp contorizează secunde trecute de la începerea simulării. Celelalte threaduri (ce reprezintă cozile) sunt initializate și pornite în fiecare secunda contorizată de t1 (threadul de timp). Cozile sunt sortate în funcție de size-ul sirului ce ține clienții în coadă.

Cozile sunt sincronizate și thread-safe. Dacă timpul clientului ajunge mai mic sau egal cu 0, acesta va fi scos din coadă. Cozile sunt stocate într-un arrayList de cozi. Rezultatele finale (average service, average waiting și peak hour) se vor afișa la momentul final.

Datele sunt prezentate într-un JScrollPane care permite vizualizarea în real-time a cozilor. De asemenea sunt salvate într-un .txt ce este golit la fiecare rulare a programului și în consola IntelliJ.

## 5. Rezultate

Sistemul simulează o coadă unde clienții cu id unic și timp de așteptare și de servire generat random vor fi puși la cozi în funcție de arrival time și vor sta la coadă timpul de servire necesar. Cazurile de testare sunt salvate în 3 documente .txt la finalul fiecăruia se găsesc statisticile referitoare la Average service, Average waiting și Peak Hour.

Average service is: 3.25  
Average waiting time is: 3.0  
Peak hour is: 18

Average service is: 4.02  
Average waiting time is: 21.4  
Peak hour is: 21

```
Time: 15
Waiting clients
[(34,21.6) , (49,27.5) , (16,30.5) , (18,28.5) , (10,31.3) , (39,27.3) , (12,26.7) , (17,37.2) , (23,23.1) , (30,33.2) , (5,39.5) , (28,36.4) , (15,17.1) , (20,26.3) , (29,16.7) , (31,26.2) , (41,21.7) , (13,32.1) , (9,25.2) , (36,20.5) , (45,26.7) , (42,20.3) , (7,20.6) , (2,20.6) , (35,37.4) , (38,25.7) , (14,26.3) , (43,37.2) , (32,16.1) , (3,30.5) ]

Queue 0:
[(50,9.1) , (37,15.3) ]

Queue 1:
[(6,13.2) , (8,15.6) ]

Queue 2:
[(22,13.3) , (1,15.6) ]

Queue 3:
[(11,13.5) ]

Queue 4:
[(24,12.1) , (33,14.5) ]
```

Average service is: 5.977  
Average waiting time is: 906.2  
Peak hour is: 47

## 6. Concluzii

*Am învățat să lucrez cu thread-uri, să creez fire de lucru și să le manageriez. De asemenea, am învățat să lucrez la un proiect mai mare decât cele anterioare.*

## 7. Bibliografie

*Se vor adauga referintele care au fost consultate de student pe parcursul implementarii temei .  
Exemplu:*

1. *Bruce Eckel, Thinking in Java (4th Edition), Publisher: Prentice Hall PTR Upper Saddle River, NJ United States, ISBN: 978-0-13-187248-6 Published: 01 December 2005.*
2. <https://dsrl.eu/courses/pt/>
3. [https://users.utcluj.ro/~igiosan/teaching\\_poo.html](https://users.utcluj.ro/~igiosan/teaching_poo.html)