DOCUMENTATIE

TEMA *2*

NUME STUDENT: Airinei Daniel-Razvan
GRUPA: 30227

CUPRINS

1. Obiectivul temei

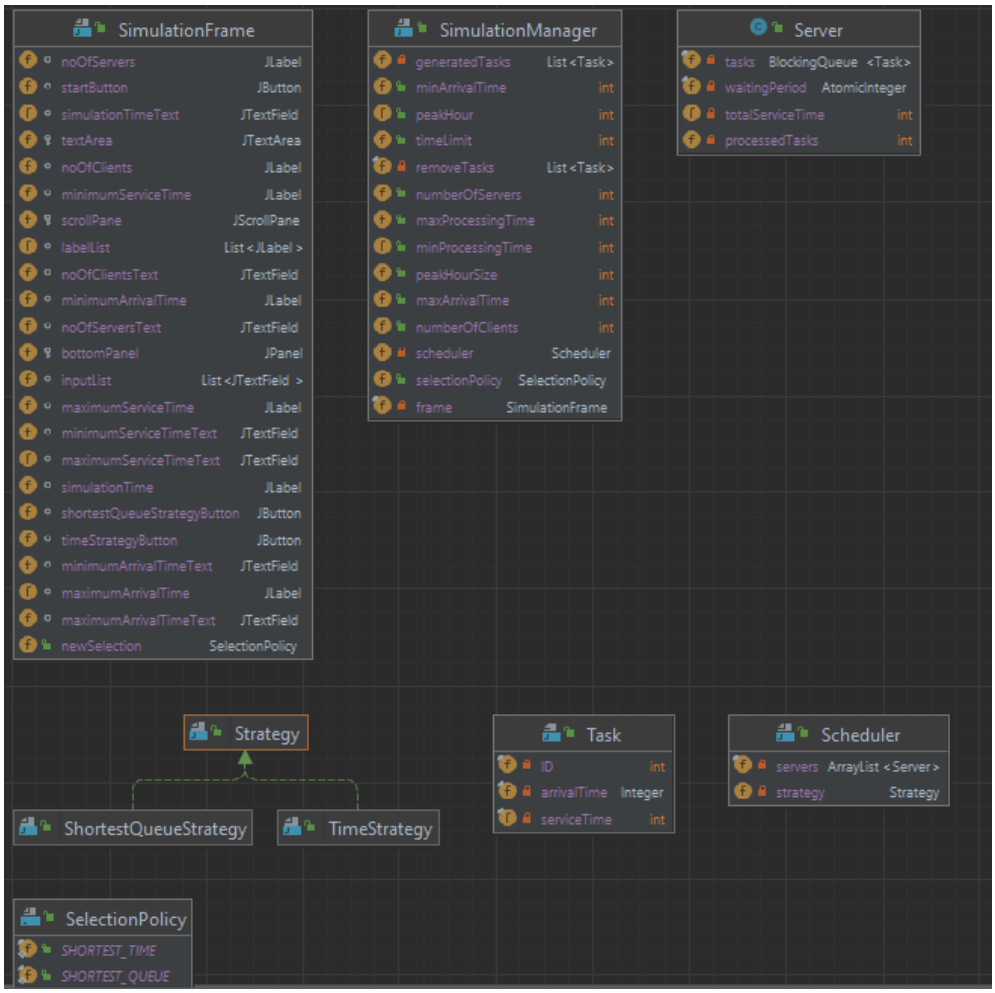
In aceasta tema obiectivul aplicatie pe care o vom crea este de a minimiza timpul de asteptare si a eficientiza cozile. Cozile sunt o problema de care dam zilnic in viata de zi cu zi fie ca stam la coada la magazine sau in complet alt loc, principiul cozilor este unul foarte simplu ci anume FIFO, primul venit , primul servit. Pentru ca ne intalnim cu cozile atat de des, zi de zi, este important ca acestea sa fie cat mai eficiente deoarece timpul inseamna bani si cu cat se sta mai mult la coada devine si mai frustrant sa stai la coada dar pierzi si timp valoros.

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

Cozile pot fi eficientizate intr-o multitudine de moduri , cel mai usor dintre acestea e sa avem cate mai multe cozi, deoarece daca impartim o coada foarte mare in doua cozi deja timpul de asteptare s-a impartit la 2, pe acest principiu putem imparti o coada in cat mai multe cozi de exemplu daca imparti o coada in 50 de cozi timpul de asteptare va fi considerabil mai mic dar daca suntem realisti intr-un scenariu din viata de zi cu zi acest lucru nu este foarte intalnit deoarece s-ar folosi extrem de multe resurse, existand si alte metode care nu folosesc atat de multe resurse si ofera rezulate promitatoare. Una din cele mai bune metode ar fi sa impartim o coada in cateva cozi mai mici in care sa prioritizam timpul, adica daca intr-o coada sunt 3 clienti care vor sta cate 5 minute in coada si in alta sunt 10 clienti care vor sta 1 min fiecare , este evident ca e spre avantajul clientului care se alatura cozii ca coada cu 10 clienti este mai eficienta din punct de vedere al timpului care asteapta la coada. Deci unul din cele mai bune approach-uri ale problemei ar fi impartim clientii in functie de timpul pe care il vor petrece in coada si sa redirectionam un nou client catre coada cu cel mai mic timp de asteptare.

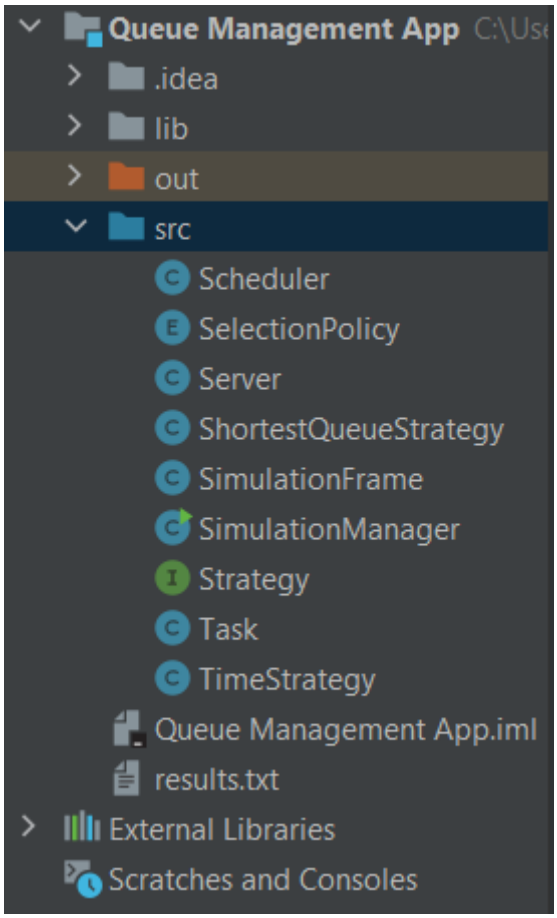
3. Proiectare

Diagrama UML :



t

Pachetul de clase :



4. Implementare

Aplicatia este structurata in noua clase, una din ele fiind o interfata iar alta o clasa de tipul enum, clasa SimulationManger este “main-ul” acestui program , in aceasta clasa se intampla cele mai importante operatii si este clasa care leaga toate celelalte clase intre ele, Operatiile care se executa in aceasta clasa sunt : generarea task-urilor, parsarea datelor din interfata, scrierea rezultatelor in fisier dar si in interfata si executia propriu-zisa a threadului principal care initializeaza cozile.

Clasa “Scheduler” se ocupa de crearea serverelor si a threadurilor, serverele fiind cozi, dar in aceasta clasa se selecteaza si strategia care se va ocupa pentru managementul cozilor iar din aceasta clasa se adauga task-uri cozilor mai pe scurt clasa selecteaza strategia care se va aplica iar mai apoi se apeleaza methoda addTask referindu-se la obiectul de tip Strategy.

Clasa “Task” este una din cele mai simple clase din acest program, aceasta clasa are rolul de a crea un task ce are 3 parametri ci anume ID, ServiceTime si ArrivalTime, in aceasta clasa se realizeaza si compararea tuturor taskurilor din lista generatedTask care se afla in SimulationManger, aceasta comparare fiind necesara pentru sortarea taskurilor in functie de ArrivalTime. Avem si cateva gettere in aceasta clasa pentru a ne folosi de ServiceTime si ArrivalTime.

Clasa “Server” este clasa unde practic facem operatii pe coada si se simuleaza executia cozii, atunci cand threadul este executat vom lua element cu element din coada, si vom pune threadul pe sleep echivalent cand cate secunde are

valoarea ServiceTime pentru fiecare task. Dupa ce am asteptat numarul de secunde vom face exact acelasi lucru pana cand coada este goala.

Clasele “TimeStrategy si ShortestQueueStrategy” sunt clasele care adauga task-uri in coada sau in server in functie de strategia pe care fiecare din ele o aplica. De exemplu in clasa TimeStrategy luam fiecare server si waiting time-ul sau si comparam cu o valoare minima care la inceput este initializat cu o valoare foarte mare, facem asta pana parcurgem toate serverele si adaugam taskul in serverul cu cel mai putin timp de asteptare. Pasii sunt aceeasi si pentru shortest queue strategy doar ca in loc sa comparam timpul de asteptare comparam dimensiunea fiecarei cozi si adaugam in coada care are cele mai putine task-uri active.

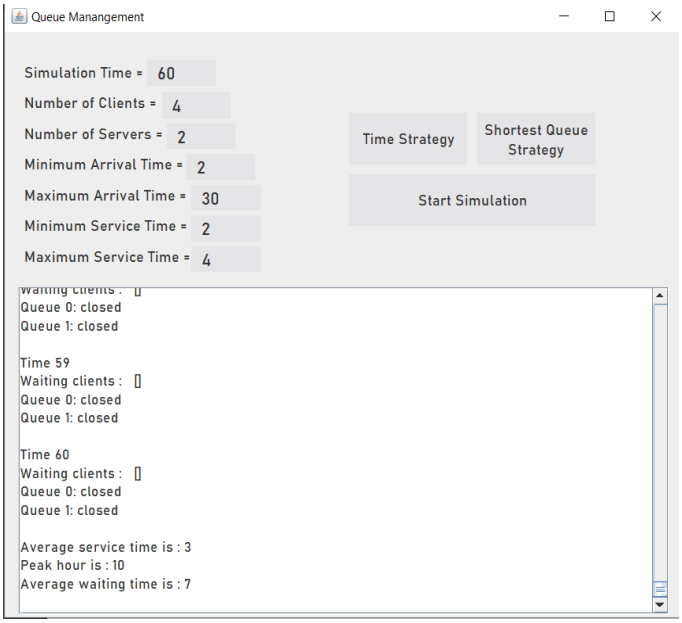
5. Rezultate

Rezultatele de mai jos sunt pentru testele din cerinta respectiv pentru :

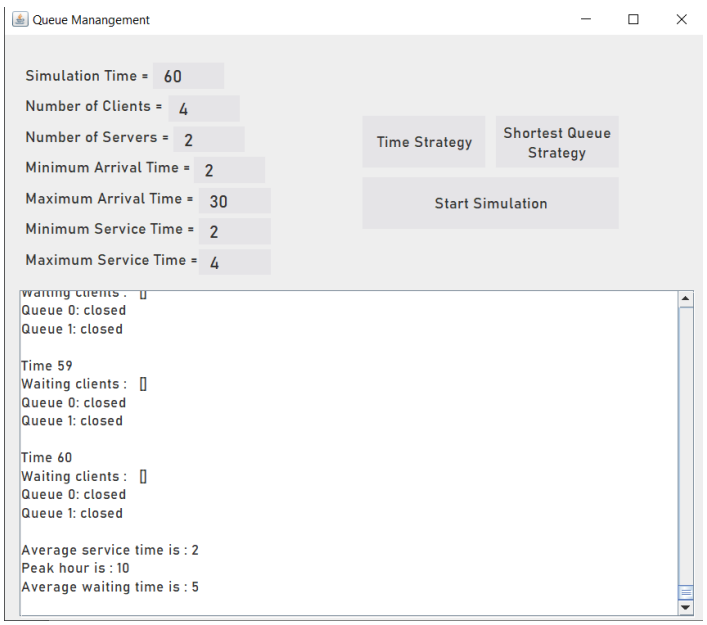
Test 1	Test 2	Test 3
N = 4 Q = 2 $t^{MAX}_{simulation}$ = 60 seconds $[t^{MIN}_{arrival}, t^{MAX}_{arrival}] = [2, 30]$ $[t^{MIN}_{service}, t^{MAX}_{service}] = [2, 4]$	N = 50 Q = 5 $t^{MAX}_{simulation}$ = 60 seconds $[t^{MIN}_{arrival}, t^{MAX}_{arrival}] = [2, 40]$ $[t^{MIN}_{service}, t^{MAX}_{service}]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

Testul 1 :

Time Strategy

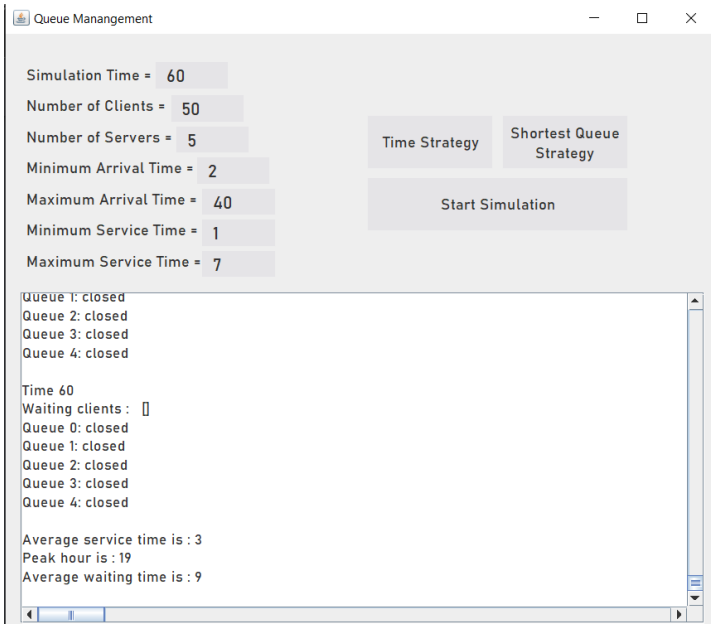


Shortest Queue Strategy

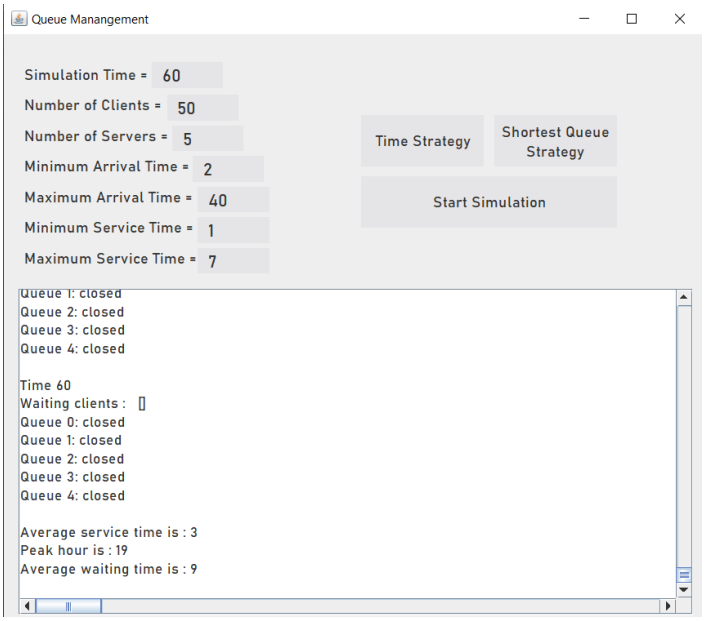


Testul 2 :

Time Strategy

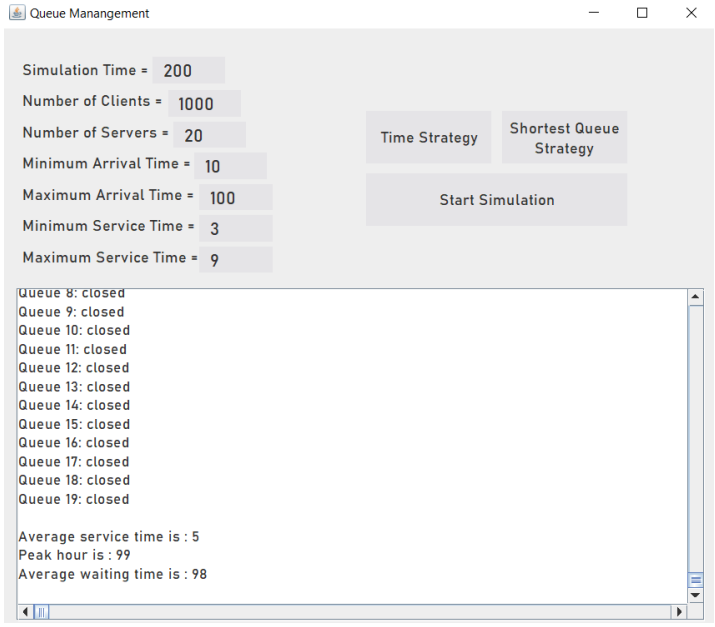


Shortest Queue Strategy

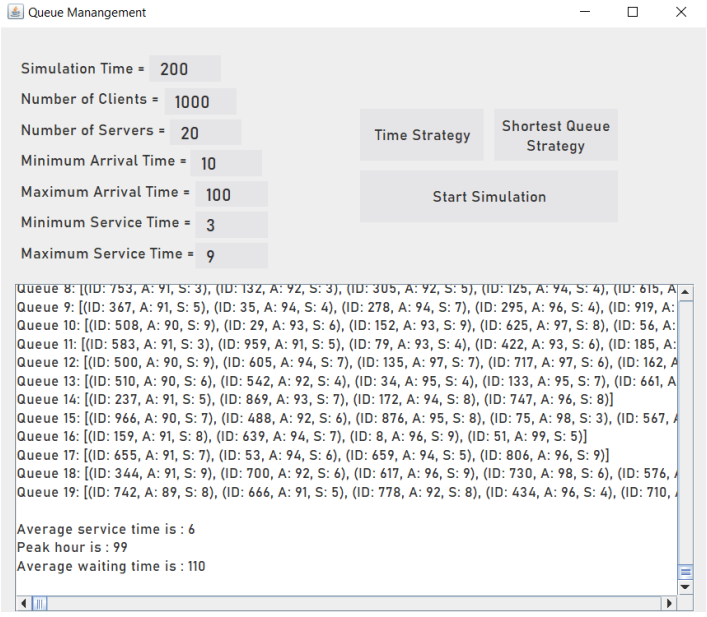


Testul 3 :

Time Strategy



Shortest Queue Strategy



Putem observa din pozele cu teste ca pe cat datele introduse cresc pe atat metoda in care luam cea mai scurta coada este mai ineficienta , in cazul in care am avut 1000 de clienti si 20 de cozi iar timpul simularii a fost 200 de secunde

cozile nici macar nu erau terminate fiind multe taskuri in continuare in coada, pe cand la strategia in care introducem in coada cu cel mai putin timp de asteptare cozile erau de mult terminate si inchise.

6. Concluzii

In final aceasta aplicatie m-a invatat destul de multe lucruri, lucrul cu threaduri si cozi au fost principalele doua lucruri ce le-am invatat din acest program dar mi-am si dezvoltat abilitatile de intelegere si cunoastere in POO deoarece a fost primul proiect in care am lucrat cu atat de multe clase dar si prima oara cand am folosit interfete si enumeratii.

7. Bibliografie

code-it.ro
w3schools.com
tutorialspoint.com