**TEMA 2**

**Queues simulator**

**NUME STUDENT : GLIGOR MIHAI**

**GRUPA : 302210**

**PROFESOR DE LABORATOR : DAN MITREA**

Cuprins

[Obiective 3](#_Toc75793018)

[**Analiza problemei , cazuri de utilizare** 3](#_Toc75793019)

[**Proiectare si implementare** 4](#_Toc75793020)

[**Rezultate si concluzii** 6](#_Toc75793021)

[**Bibliografie** 7](#_Toc75793022)

# **Obiective**

Obiectiv principal :

* Proiectarea si implementarea unui sistem care sa fie capabil sa simuleze cozi in timp real , sa genereze

n clienti random si sa ii distribuie intr- un mod cat mai eficient astfel incat timpul in care un client asteapta la o coada sa fie minimizat . De asemenea sa fie capabil sa genereze un fisier numit log.txt care sa contina rezultatele simularii .

Obiective secundare :

* Afisarea rezultatelor simularii la terminarea acesteia in log.txt file sau pe interfata grafica (average waiting time, average service time, peak hour s.a.) .
* O interfata grafica intuitiva si usor de folosit .
* O interfata grafica care sa permita vizualizarea in timp real a simularii .
* Analizarea problemei si identificarea cerintelor acesteia .
* Proiectarea aplicatiei .
* Implementarea aplicatiei .
* Testarea si controlul calitatii produsului final .

## **Analiza problemei , cazuri de utilizare**

**Problema :**

Cozile sunt o problema foarte des intalnita intr-o gama larga de domenii , mai ales intr-o perioada a istoriei cand „timpul inseamna bani” , este o expresie din ce in ce mai adevarata , deci eficentizarea acestora este un subiect abordat de multe companii care isi iau profitul in serios de aceea este necesara o aplicatie de simulare a mediului financiar in care se afla .

**Solutia :**

Creearea unei aplicatii care simuleaza cozile si eficientizeaza procesul de distribuire a clientlor in timp real in functie de o anumita strategie , oferind date folositoare , precum timpul mediu de asteptare , ora de varf ,timpul mediu in care un client a fost servit s.a. , date care pot fi folosite mai departe pentru imbunatatirea ulterioara a procesului . Aceasta aplicatie ofera aceste date importante fara a consuma resurse importante si fara a avea nevoie de prezenta unor clienti fizici , de asemenea fiind mai eficienta din punct de vedere a timpului , colectand date care in mod normal nu ar fi valabile decat dupa luni de observarea a pietei .

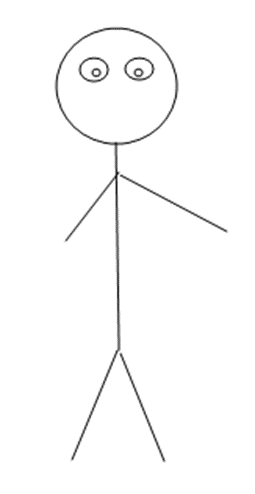
Functionalitatile sistemului Queues Simulator

* Introducerea si generarea a n clienti pentru o simularea cat mai realista .
* Selectarea unui anumit numar de cozi .
* Selectarea unui anumit timp de simulare .
* Selectarea timpului minim si maxim in care pot sa ajunga clienti .
* Selectarea timpului minim si maixim in care acesti clienti vor fi serviti .
* Afisarea in timp real a evolutiei simularii prin intermediul interfetei grafice .
* Procesarea de date folositoare pentru imbunatatirea sistemului , precum timpul mediu de asteptare , ora de varf ,timpul mediu in care un client a fost servit s.a. .

Adaugarea parametrilor simularii :

* Numar de client .
* Numar de cozi .
* Timp de simulare .
* Timpul minim si maxim de sosire .
* Timpul minim si maxim de servire .

I cazul inreoducerii de caractere invalide in casetele de text , simulatorul va afisa mesajul “ Bad input ” intr-o caseta pop -up.



### **Proiectare si implementare**

**Proiectare la nivel general**

Rezultat

Date cu privire la simulare

Introducerea parametrilor

necesari simularii

**Proiectare la nivel de pachete :**

* Pachetul controller contine metoda VController si are rolul de a face legatura intre interfata grafica ( adica input -ul utilizatorului ) si aplicatie ( care primeste acel input si il proceseaza ) .
* Pachetul View contine metodele View si VSimulare care reprezinta interfetele grafice ale aplicatiei ( necesare pentru inputul utilizatorului ) .
* Pachetul Main contine clasa main care ruleaza aplicatia ( instantiaza obiectele necesare pentru rulare ) .
* Pachetul Strategy contine clasele ConcreteStrategyQueue si ConcreteStrategyTime , interfata Strategy si Enumeratia SelectPolicy ( pachetul e responsabil cu implementarea strategiei de repartizare a clientilor in cozi ).
* Pachetul model contine clasele Task ,Server Scheduler ,Simulation manager si Log care implementeaza partea functionala a simularii .

**Proiectarea la nivel de clase si metode :**

**Pachetul model :**

Clasa Task : contine constructorul pentru fiecare ‚task’ adica pentru fiecare client compus din id , timpul de sosire ( tArrival ) si timpul de servire ( tService ) , gettere si settere pentru fiecare element cat si metoda toString pentru client .

Clasa Server : contine atributele BlockingQueue < Task > tasks care reprezinta o lista de clienti , AtomicInteger waitingPeriod care reprezinta perioada de asteptare si foloseste AtomicInteger deoarece e thread safe , id pentru identificarea unica a serverului , boolean running care ne spune daca serverul e inchis sau deschis , completedTasks care numara cate taskuri sunt completate si tServerOpen .De asemenea mai contine constructorul Server si urmatoarele metode :

* metoda addTask : adauga clienti in lista si updateaza perioada de asteptare .
* metoda getConvert : introduce elementele din BlockingQueue intr- un array list pentru a facilita accesul la elementele listei pe baza unui indice si returneaza acel array list.
* metoda run : este formata din doua while uri , primul ruland pana cand se incheie threadul iar al doilea pana cand nu mai exista nici un element in lista de taskuri , dupa care se pune in variabila locala time timpul de servire a clientului si threadul de opreste pana cand trec 1000\*time milisecunde ( pana trece timpul in care e servit ), dupa care se actualizeaza waitingPeriod si se trece la urmatorul client .

Clasa Scheduler : contine o lista de servere , constructorul care creeaza un numar n de servere in functie de ce introduce utilizatorul si assigneaza fiecaruia un thread si urmatoarele metode :

* metoda change strategy care schimba strategia de repartizare a clientilor cu ajutorul unui if , verificand valoarea lui selectPolicy si creeand un nou obiect in functie de strategia selectata .
* metoda emp verifica daca mai exista clienti in servere, parcurgand lista de servere cu un for care merge pana la i < servers.size si care are in interior un if care converteste lista de clienti din fiecare server in array list si ii compara dimensiunea cu 0 in cazul in care sunt goale , returneaza true , altfel returneaza fals .
* metoda dispach task primeste ca parametru un task ( client ) care este repartizat in functie de strategie .

Clasa Simulation Manager : contine atributele necesare pentru rularea simularii ( timpul rularii , timpul minim si maxim de servire , timpul minim si maxim de ajungere la coada ,numarul de cozi , numarul de clienti ) ,instantiarea unui file writrer cu care vom scrie in fisierul log , constructorul SimulationManager su urmatoarele metode :

* metodele randomArrivingTime si randomProcessingTime care calculeaza si returneaza un timp random de sosire si de servire a clientilor folosind metoda Math.random () .
* metoda generateNRandomTasks genereaza clienti aleatoriu . Aceasta creeaza un nou array list , dupa care in interiorul unui for care merge de la 0 la i mai mic decat numarul de clienti introdusi de utilizator ,creeaza un task random cu id ul reprezentat de contorul i al for ului si arrival time si service time calculate de metodele mentionate anterior si apoi aceste taskuri sunt introduse in array lis tul creeat , care este returnat la final
* metoda run care contine un while care merge cat timp timpul curent este mai mic decat timpul simularii si cat timp exista clienti in servere si in lista de asteptare . In interiorul acelui while primul for parcurge lista de clienti in asteptare si daca vreunul din ei are timpul de sosire egal cu cel curent acesta este repartizat la unul din servere dupa care eliminat din lista de asteptare apoi un for care afiseaza clientii din lista de asteptare , urmatorul for parcurge serverele si scrie fiecare client , dupa care timpul curent este incrementat si se incheie while ul, dupa care mai avem un for pentru a parcurge inca o data serverele pentru a arata ca nu au mai ramas clienti in ele .

**Pachetul main**

Clasa main : contine doar instantierea obiectelor necesare pentru a rula programul (obiect de tip view si controller) si setarea vizibilitatii pentru interfata cu utilizatorul .

**Pachetul Strategy**

Interfata Strategy : contine metoda abstracta addTask pe care o vor implementa clasele ConcreteStrategyQueue si ConcreteStrategyTime .

Enumeratia SelectPolicy : contine cele doua optiuni cand vine vorba de strategie , adica SHORTEST\_QUEUE si SHORTEST\_TIME .

Clasa ConcreteStrategyQueue : nu a mai fost implementata .

Clasa ConcreteStrategyTime : Contine metoda addTask care cu ajutorului unui for parcurge serverele si cauta timpul minim de asteptare , dupa care compara timpul minim de asteptare cu timpul de asteptare curent al serverelor si adauga clientul la coada cu timpul cel mai scurt .

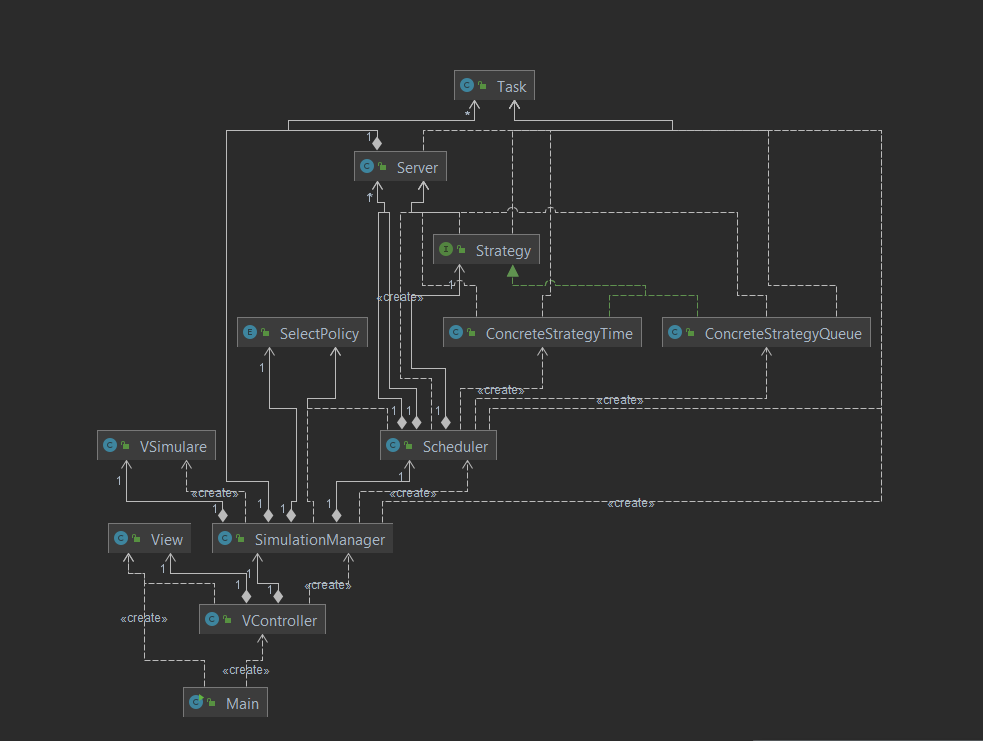
**Pachetul view**

Clasa View : contine interfata de setup pentru rularea simularii , cu textField urile necesare ( timpul rularii , timpul minim si maxim de servire , timpul minim si maxim de ajungere la coada ,numarul de cozi , numarul de clienti ) , label – uri pentru a face interfata cat mai usor de folosit de care utilizator si metode prin intermediul carora obtinem textul introdus in textfield uri .

Clasa VSimulare : contine interfata grafica care arata in timp real desfasurarea simularii prin intermediul unui JScrollPanel si a unui JTextArea care este actualizat in interiorul metodei run a clasei SimulationManager prin intermediul metodei setText care primeste ca parametru un string s pe care il scrie in text area.

**Pachetul controllers**

Clasa VController : contine doar un constructor care contine la randul lui un action listener pentru butonul care porneste simularea . In interior se intantiaza un nou obiect de tipul SimulationManager care primeste ca parametrii valorile scrise in textfield -urile din interfata grafica pentru setup -ul simularii dupa care se porneste un thread separat pentru rularea acestuia si se seteaza vizibilitatea interfetei pentru vizualizarea in timp real a simularii .



O imagine care conține text, captură de ecran, negru

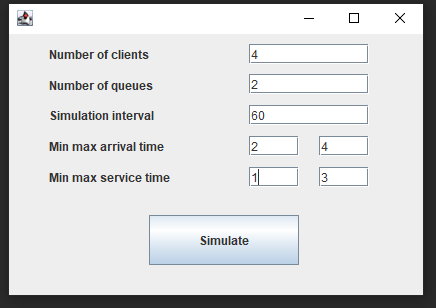
Descriere generată automat

#### **Rezultate si concluzii**

Rezultatul obtinut consta intr -o aplicatie usor de folosit de catre orice utilizator , cu o interfata grafica intuitiva care permite o utilizare in care experienta nu este necesara . Aceasta aplicatie este capabila sa simuleze n cozi la care sa ajunga m clienti in functie de preferintele clientului . De asemenea clientul poate seta paramentrii precum timpul in care ruleaza simularea , timpul minim si timpul maxim in care pot sa ajunga clientii in coada si timpul maxim si timpul minim in care clientii sunt serviti , creeand o simulare care sa se apropie cat mai mult de realitate si care sa satisfaca pe deplin nevoile utilizatorului .

Modalitate de utilizare :

Odata ce aplicatia este rulata va aparea un ecran care ajuta la introducerea parametrilor necesari simularii precum in imaginea urmatoare :



In primul textbox se va introduce numarul de clienti ,

In al doilea textbox se va introduce numarul de cozi dorite

In al treilea textbox se va introduce durata simularii masurata in secunde

In urmatoarele doua textbox- uri se va introduce timpul minim de sosire a unui client respectiv timpul maxim de sosire a unui client

Iar in ultimele doua textbox -uri se va introduce numarul timpul minim de servire a unui client , respectiv timpul maxim de servire a unui client

Dupa care tot ce trebuie sa faca utilizatorul apese cu cursorul pe butonul de simulate si va aparea o noua interfata pe langa cea de setup in interiorul careia se va afisa simularea in timp real .

O imagine care conține text

Descriere generată automat

Daca utilizatorul doreste sa introduca alti parametrii tot ce trebuie sa faca e sa apese pe butonul de X din coltul noii interfete aparute si sa schimbe parametrii in cadrul primei interfete de setup , si sa apese din nou butonul de simulare .

Daca utilizatorul doreste fisierul log.txt generat , acesta poate fii gasit in folderul proiectului si contine acelas text care apare pe interfata grafica in timp real .

O imagine care conține text

Descriere generată automat

##### **Bibliografie**

Thread start and management :

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

<http://tutorials.jenkov.com/java-concurrency/creating-and-starting-threads.html>

<https://www.youtube.com/watch?v=eQk5AWcTS8w&t=206s>

Thread safe elements :

<https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/atomic/AtomicInteger.html>

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

Accesing elements form the list :

<https://www.geeksforgeeks.org/stack-peek-method-in-java/>

https://www.geeksforgeeks.org/queue-poll-method-in-java/

GUI elements :

<https://docs.oracle.com/javase/tutorial/uiswing/components/textarea.html>

<https://www.youtube.com/watch?v=PARFguh3ck8&t=175s>

Provided by UTCN :

- <http://docs.oracle.com/javase/tutorial/essential/concurrency/index.html>

- <http://www.tutorialspoint.com/java/util/timer_schedule_period.htm>

- <https://www.javacodegeeks.com/2013/01/java-thread-pool-example-using-executors-and-threadpoolexecutor.html>