DOCUMENTATIE

Tema numarul 2

Nume Student: Faur Sergiu

Grupa: 30224

**Cuprins**

1.Obiectivul lucrarii . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

2.Analiza problemei, modelare, scenarii, cazuri de utilizare . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4

3.Proiectare . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .5

**4.Implementare . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .7**

[5.Rezultate](#_Toc95297889) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .9

6.Concluzii . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .10

7.Bibliografie . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .10

1.**OBIECTIVUL LUCRARII**

OBIECTIVUL PRINCIPAL

Obiectivul principal este de a face o aplicatie destinata sistemelor ce se bazeaza pe cozi prin simularea a n client ce sosesc pentru a fi serviti la q cozi , dupa servire parasind coada.In final vrem a generam timpul mediu de asteptare , de servire si momentul de varf cand este cel mai aglomerat.

OBIECTIVE SECUNDARE

**1.Analiza problemei si identificarea cerintelor – la aceasta se doreste intelegerea problemei si cee ace trebuie implementat – se va discuta mai detaliat in capitolul 2**

2.Designul aplicatiei de simulare a unor cozi – la aceasta dorim a ne gandim la cum ar arata aceasta aplicatie de simulare pentru a fi cat mai usor utilizata de utilizator si de asemenea ce clase trebuie implementate si care sunt relatiile dintre acestea – e va discuta mai detaliat in capitolul 3

3.Implementarea aplicatiei – pentru aceasta se va discuta implementarea claselor utilizand threaduri si implementarea interfetei grafice – se va discuta in capitolul 4

4.Testarea aplicatiei de simulare – se vor testa niste teste date , rezultatele acestora gasindu-se in fisierele Test1 , Test2, Test3. – mai multe in capitolul 5

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

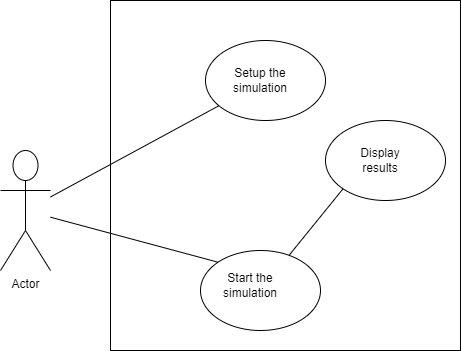
Cerinte functionale:

Aplicatia ar trebui sa lase utilizatorul sa seteze parametrii de simulare .

Aplicatia ar trebui sa lase utilizatorul sa porneasca aplicatia .

Aplicatia ar trebui sa afiseze in timp real rezultatele.

Diagram use case:



Use case: Se introduc datele

Actor principal: user

Scenariul de success:

1.Utilizatorul introduce datele corecte sub forma unor numere in campurile destinate pentru numarul de client , numarul de cozi, minimul pentru timpul de sosire , maximul pentru timpul de sosire , minimul pentru timpul de servire , maximul pentru timpul de servire si timpul de simulare.

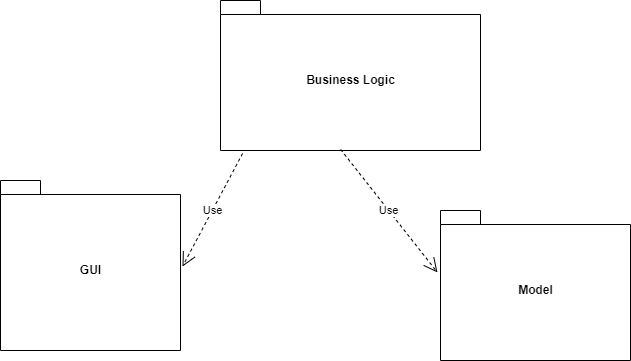
2. Utilizatorul apasa pe butonul de start si porneste simularea

3.Rezultatele sunt afisate in interfata utilizator

Caz alternativ : utilizatorul introduce date gresite in cazul in care aplicatia nu va functiona , utilizatorul trebuind sa corecteze datele de intrare.

# 3.Proiectare

Diagrama de pachete:

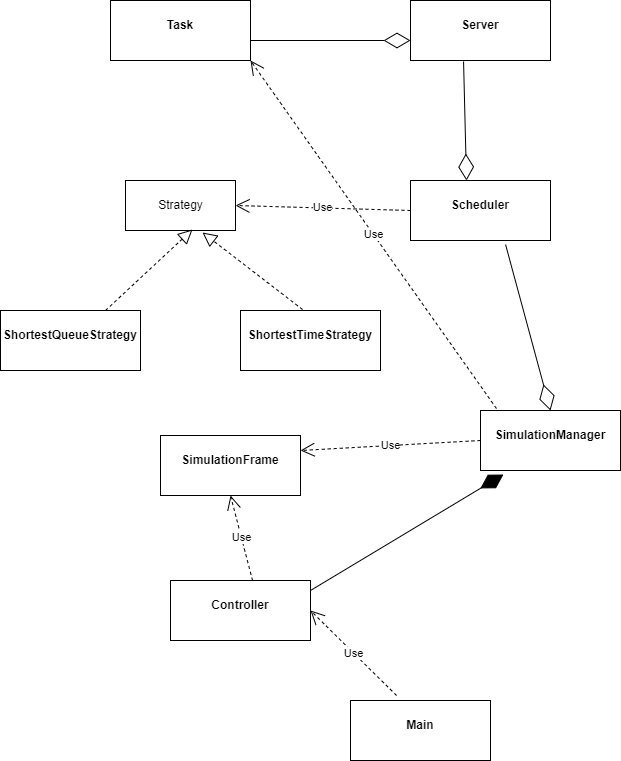


Pachetul Model contine clasele Task si Server.

Pachetul Business Logic contine clasele SimulationManager, Scheduler, Interfata Strategy si calsele care o implementeaza ShortestQueueStrategy si ShortestTimeStrategy. Pe langa asta mai este si enumeratia StrategyPolicy care ajuta la alegerea strategiei.

Pachetul GUI contine clasa SimulationFrame care este de fapt interfata cu utilizatorul.

Diagrama Claselor:



MODEL

Clasele Task si Server din pachetul Model sunt echivalentul unui client si a unei cozi din viata reala , relatia de agregare fiind una naturala din cauza ca o coada este formata din mai multi client.

BUSINESS LOGIC

Clasa Scheduler din acest pachet este echivalentul unui planificator care allege coada optima in functie de strategie , de aceea acesta contine toate cozile si imparte taskuri in functie de enumeratia Strategie.

Clasele ce implementeaza interfata Strategy sunt ShortestQueueStrategy si ShortestTimeStrategy fiecare avand o implementare diferita a functiei addTask in functie de cea mai scurta coada sic cel mai scurt timp.

Clasa SimulationManager are rolul de a menegeria celelate clase , acesta fiind un Thread , in el intamplandu-se firul principal al executiei .

GUI

Clasa SimulationFrame reprezinta interfata grafica cu utilizatorul .

Clasa Controller are rolul de a porni aplictia la apasarea pe buton , atunci creeandu-se un nou SimulationManager

Clasa Main foloseste Controllerul , ea ruland aplicatia.

# 4.Implementare

Fiind un proiect OOP , se bazeaza pec lase sip e relatiile dintre acestea , in capitolul acesta vom descrie fiecare clasa cu functionalitatea sa si cee ace reprezinta .Deasemenea , fiind un proiect cu Threaduri , vom descrie cum functioneaza acestea.

Task

Aceasta clasa reprezinta un client care trebuie sa fie servit .

Atributele acestei clase sunt id-ul clientului , timpul de sosire al clientului si timpul de servire .

Server

Aceatsa clasa reprezinta o coada ce contine clienti .

Pe langa array-ul de client , un alt atribut este cel de waitingPeriod ce reprezinta timpul de asteptare la coada.

Metodele din aceasta clasa sunt pentru a adauga un client la coada, a sterge un client din coada si fiindca un server reprezinta un thread , aceasta clasa implementeaza interfata runnable , de aceea este si metoda de run . In aceasta metoda practice se ia primul client din coada ( cel ce urmeaza sa fie servit ) si in timpul de servire al clientului , threadul este oprit ( thread.sleep ) , decrementandu-se si timpul de servire pentru acest client sic el de asteptare la coada.

Interfata Strategy

Aceasta interfata este doar cu o metoda cea , addTask pentru a adauga client la cozi in functie de o strategie data de enumeratia StrategyPolicy , in functei de cel mai scurt timp si cea mai scurta coada.

Clasele ShortestQueueStrategy si ShortestTimeStrategy

Aceste clase implementeaza interfata Strategy , fiecare avand metoda addtask , dar in functie de cel mai curt timp I cea mai curta coada.

Clasa Scheduler

Aceasta clasa reprezinta planificatorul de coz ice are ca attribute toate cozile si strategia pentru care se alege unde se va adauga noul client. Acest lucru se intampla in metoda dispatchTask unde se apeleaza metoda de addTask din strategie.

In metoda chagePolicy se alege strategia in functie de enumeratia ce contine shortestTime si shortestQueue.

De asemenea in contructorul acestui planificator , se pornesc threadurile pentru fiacre coada .

SimulationManager

In constructorul acestei clase se initializeaza valorile parametrilor pentru simulare si de asemenea se intantiatiaza Schedulerul si se apeleaza metoda de generateNRandomTasks . Aceasta metoda are rolul de a genera random venirea clientilor si timpul lor de servire.

Fiind o clasa ce implementeaza interfata Runnable , in metoda de run se produce fluxul principal al executiei programului. Pentru inceput se ia un currentTime ce tine contorul timpului pan ace acesta ajunge la timpul de simulare , de asemenea se iau si variabilele de averageWaitingTime , averageServiceTime si peakHour . Primele 2 dintre acestea e calculeaza in spate pentru fiecare coada , incrementandu-se de fiecare data cand apare un client nou , in run doar facandu-se media aritmetica pentru cati client au fost in total.Pentru a calcula peakHour am folosit o metoda in Scheduler pentru a afla cati client sunt la cozi intr-un anumit moment de timp , in metoda de run doar facand un maxim dintre ecestea.

Mai departe e iau Taskurile generate random si e aleg cele al caror moment de sosire este egal cu currentTime , apelandu-se metoda de dispatchTask din Sceduler , eliminandu-se din Takurile generate random si adaugandu-se intr-un ir de Taskuri ce trebuie ulterior sterse din coada. Mai departe e verifica ce Taskuri trebuie eliminate din cozi prin verificareatimpului de ervire a acestora cand este egal cu 0 ( acesta decrementandu-se la fiecare moment de timp ). Dupa aceasta se apeleaza functia de display pentru a afisa in timp real clientii din cozi , iar cand timpul ajunge la final se afiseaza si averageWaitingTime ,averageServiceTime si peakHour.In final se pune pe thread.sleep pentru 1 secunda.

SimulationFrame

In aceasta clasa e creeaza interfata pentru utilizator

Controller

ACeasta clasa a fost facuta pentru a porni simularea doar cand butonul de start a fost apasat, adica se creeaza un nou SimulationManager cand acesta este apasat.

Main

In main e intantiaza un Controller si un nou frame , creeand astfel o noua fereastra.

# 5.Rezultate

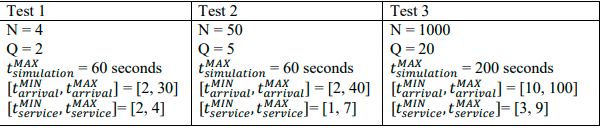
Pentru afisarea rezultatelor am folosit o metoda de display care afiseaza in timp real coziile si clientii din ele

public StringBuilder display(int currTime) {  
 StringBuilder stringBuilder = new StringBuilder();  
 stringBuilder.append("\nTime " + currTime + "\n");  
 stringBuilder.append("Waiting clients: ");  
 for (Task task : generatedTasks  
 ) {  
 stringBuilder.append(task.toString() + ",");  
 }  
 for (int i = 1; i <= numberOfServers; i++) {  
 stringBuilder.append("\n" + "Queue " + i + ":");  
 if (scheduler.getServers().get(i - 1).getTasks().isEmpty())  
 stringBuilder.append("closed");  
 else  
 for (Task task : scheduler.getServers().get(i - 1).getTasks()  
 ) {  
 stringBuilder.append(task.toString() + ",");  
 }  
 }  
 return stringBuilder;  
}

In interfata utilizator se modifica textArea prin punerea aplearea functiei de append facuta in SimulationFrame

public void appendTextArea(String s) {  
 results.append(s);  
}

Testele au fost facute pentru urmatorii parametrii si au fost scrise in fisierele Test1 Tes2 si Tes3 atasate.



# 6.Concluzii

Acest proiect a fost benefic pentru experienta de lucru cu Threaduri si pentru intelegerea functionarii lor.

Niste dezvoltari ulterioare sunt:

* Imbunatatirea interfetei utilizator pentru recunoasterea introducerii unor date gresite
* Infrumusetarea interfetei utilizator
* Optiunea de a schimba strategia din interfata

# 7.Bibliografie

<https://app.diagrams.net>

<https://dsrl.eu/courses/pt/materials/A1_Support_Presentation.pdf>