MINISTERUL EDUCAȚIEI ȘI CERCETĂRII ȘTIINȚIFICE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICA SI CALCULATOARE

Tema nr 2

Documentatie

Costea Paul
Grupa 30229

Cuprins

1.Obiectiv

Cerinta temei de laborator este implementarea unei aplicatii care simuleaza “un magazin” care are niste cozi cu clienti, tinand cont de unele aspecte.

Scop principal : crearea unei aplicatii care simuleaza ce se inampla cu cozile la fiecare moment dat.

Scop secundar: crearea unei interfete grafice care permite utilizatorului sa isi faca propria lui simulare. Aceasta interfata ar trebui sa ii permita utilizatorului sa introduca anumite

date cum ar fi : numarul de cozi, numarul de clienti, timpul de sosire al fiecarui client sau timpul de servire.

2.Studiul Problemei

Pentru aceasta aplicatie implica intelegerea modului in care un magazin cu cozi de clienti functioneaza, precum si identificarea aspectelor cheie care trebuie luate in considerare pentru a crea o simulare adecvata.

Primul aspect important este gestionarea cozilor. In aplicatia noastra, trebuie sa putem gestiona mai multe cozi de clienti simultan. Fiecare coada trebuie sa aiba propriul ei timp de asteptare si timp de servire, iar clientii trebuie asezati in cozi in functie de numarul lor de produse sau de alte criterii.

Al doilea aspect important este generarea de clienti. Pentru a simula un magazin realist, trebuie sa putem genera clienti care vin la cozi in mod aleatoriu. Fiecare client trebuie sa aiba propriul timp de sosire si, eventual, un timp maxim de asteptare in coada.

Al treilea aspect important este gestiunea interactiunilor dintre clienti si cozi. De asemenea, trebuie sa putem urmari interactiunile dintre clienti si cozi pentru a putea genera informatii relevante despre experienta clientilor.

In final, trebuie sa putem crea o interfata grafica intuitiva care sa permita utilizatorilor sa isi configureze propria lor simulare. Interfata trebuie sa permita utilizatorilor sa introduca anumite date cum ar fi numarul de cozi, numarul de clienti, timpul de sosire al fiecarui client sau timpul de servire.

Pentru a implementa aceasta aplicatie, vom utiliza algoritmi si structuri de date adecvate. Vom folosi, de exemplu, cozi si liste pentru a gestiona clientii si cozile, si vom utiliza algoritmi de simulare pentru a genera evenimente aleatorii.

3.Implemare:

3.1 Diagramele UML

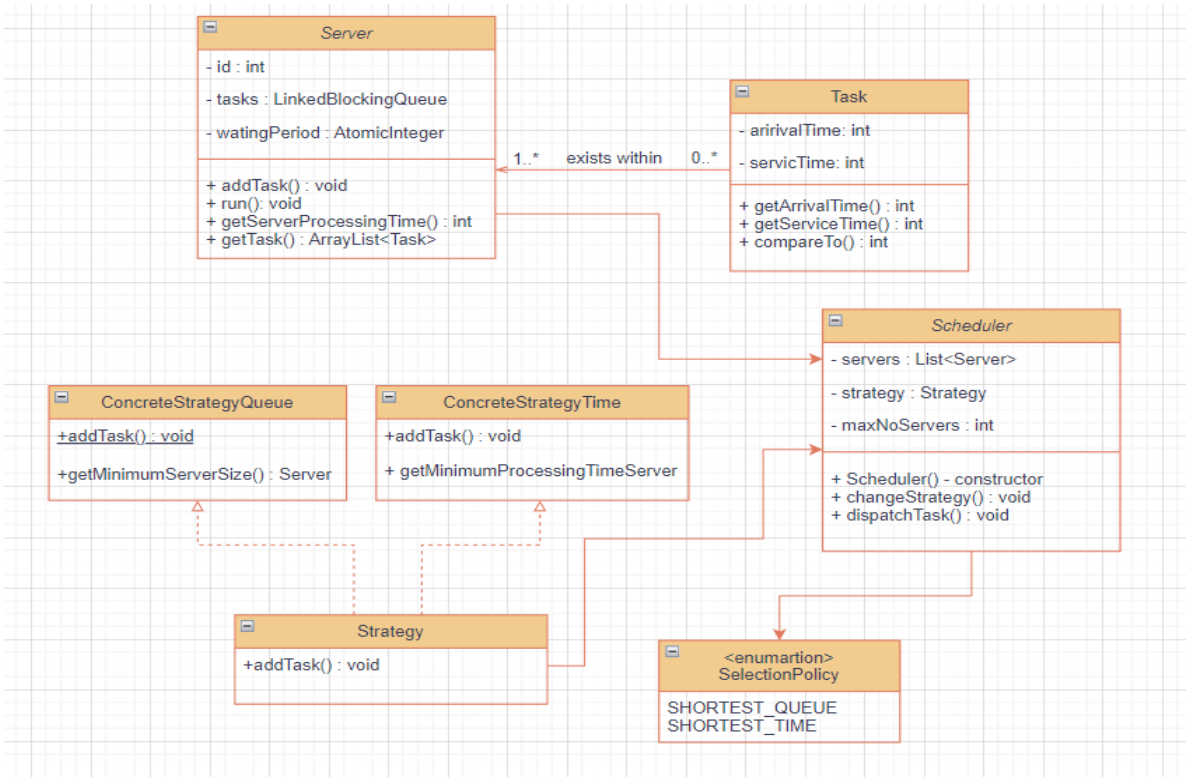

Diagrama UML care reprizinta logica aplicatiei

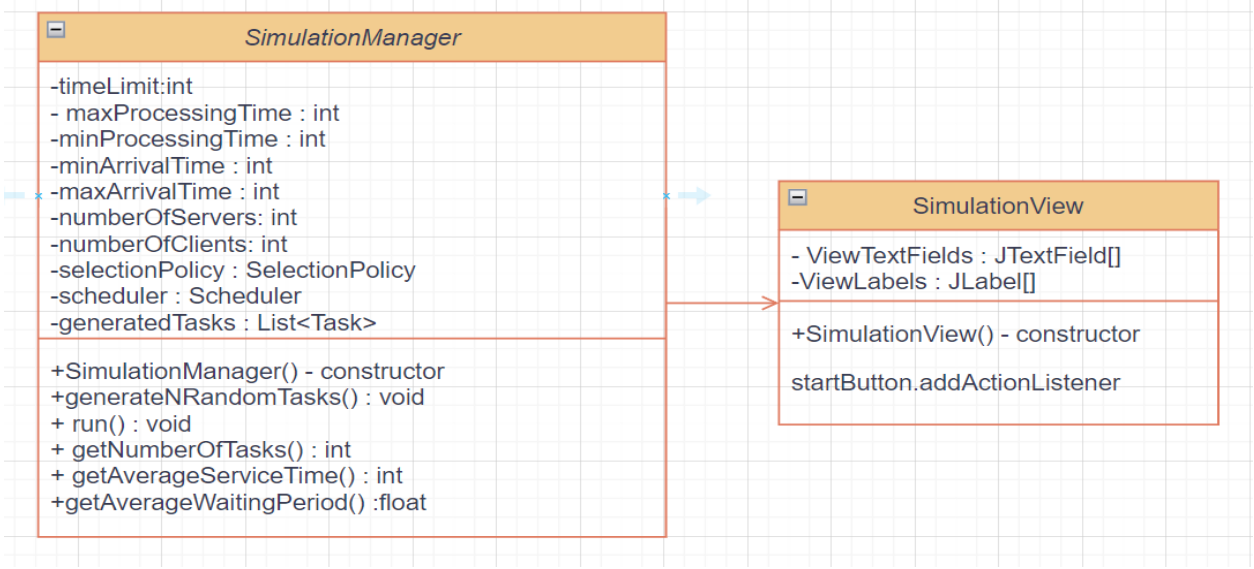

Diagrama UML care reprizinta flow-ul simularii

Unified Modeling Language sau UML pe scurt este un limbaj standard pentru descrierea de modele si specificatii pentru software. UML a fost la bază dezvoltat pentru reprezentarea complexității programelor orientate pe obiect, al căror fundament este structurarea programelor pe clase, și instanțele acestora (numite și obiecte). Cu toate acestea, datorită eficienței și clarității în reprezentarea unor elemente abstracte, UML este utilizat dincolo de domeniul IT. Așa se face că există aplicații ale UML-ului pentru management de proiecte, pentru business.

3.2 Diagrama Use-Case

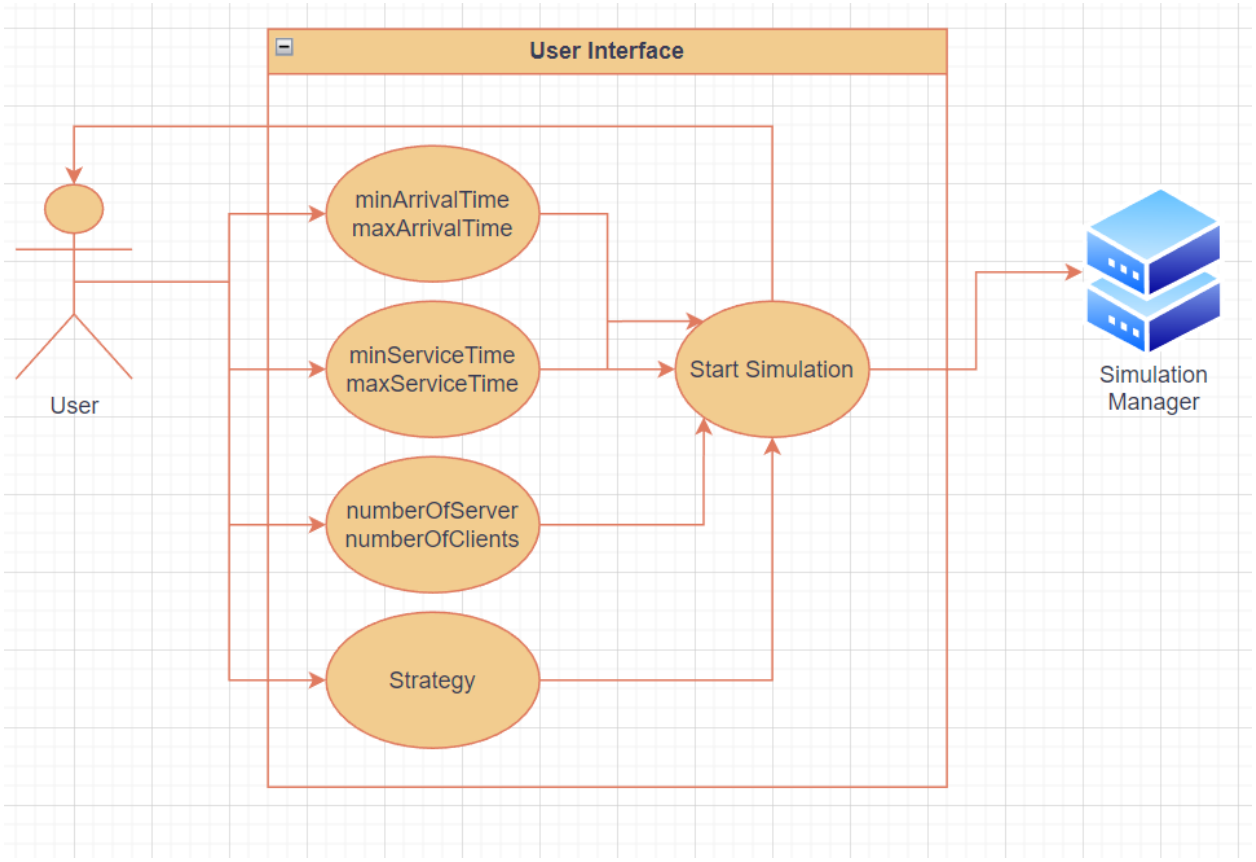

Diagrama use case a aplicatiei

Aceasta diagrama reprezinta cum s-ar astepta utilizatorul sa foloseasca apliciatia. Acesta poate introduce mai multe date, intr-un panel care sunt trimise la Simulation Manager cand acesta apasa pe Start Simulation.

3.3 Clase

Aplicatia este impartita in doua pachete : Model (reprezentarea logicii si manipularea datelor), Simulation(reprezentarea grafica a datelor pentru utilizator, cat si updatarea real time a datelor).

In pachetul **model** avem 6 clase si 1 interfata:

Clasa **TASK** - este clasa care reprezinta clientul care este identificat prin timpul de sosire la coada si timpul de asteptare al acestuia la coada.O metoda mai importanta din aceasta clasa este compareTo() deoarece sortam task-urile crescator.

Clasa **Server** - este reprezentarea practic a cozii, care este identificata printr-un Queue de taks-uri si un waitingPeriod() si de asemenea fiecare server are id-ul propriu.Este foarte importat de mentionat ca aceasta clasa implementeaza interfata Runnable deoarece si intr-un magazin real putem avea mai mult decat o singura coada si de aceea este foarte important sa putem lucra in paralel cu ajutotul thread-urilor.Cea mai importanta metoda din aceasta clasa este run() deoarece aceasta metoda defineste o mare parte din intreaga logica a aplicatiei(descrie comportamenul unei adevarate cozi).

Interfata **Strategy** - aceata interfata defineste comportamentul dupa care vrem sa adaugam clienti in cozi, in functie de marimea cozii, adica cati clienti sunt inaintea noastra, sau in functie de timpul de finalizare ar tuturor celorlate persoane pana la randul nostru.

Clasele **ConcreteStrategyTim**e si **ConcreteStrategyQueue** sunt clasele care implementeaza interfata Strategy. Prima clasa adauga clientii in functie de timp iar a doua in functie de marimea cozii.

Clasa **Scheduler** - este clasa care imbina toata logica din model la un loc acesta clasa da dispatch la task-ul nostru, folosind strategia aleasa de noi la serverul corespunzator.

De mentionat ca mai avem si enumeratia **SelectionPolicy** care reprezinta cele 2 strategii.

In pachetul Simulation avem 2 clase :

Clasa **SimulationManage**r : aceasta clasa ia datele introduse de utilizator si le foloseste pentru a creea aplicatia propriu zisa. Reusim sa facem acest lucru cu ajutorul metodei updateServers() care are rolul de a updata continutul tuturor serveror la orcie moment de timp, la fel si cu lista de clienti.

Clasa **SimulationView :** este defapat GUI-ul aplicatiei, utilizatorul are un panou prin care introduce datele pentru simulare si cand apasa pe start se creeaza un al panou care afiseaza simularea in timp real.

4. Concluzii:

In urma implementării aplicatiei de simulare a cozilor intr-un magazin, am invatat cat de importante sunt algoritmii si structurile de date pentru a gestiona clientii si cozile. Am folosit cozi si liste pentru a gestiona clientii si cozile si am utilizat algoritmi de simulare pentru a genera evenimente aleatorii.

Un aspect important al aplicatiei este generarea aleatoare a clientilor pentru a simula un magazin realist. Clientii trebuie sa aiba propriul lor timp de sosire si, eventual, un timp maxim de asteptare in coada. Gestionarea cozilor este, de asemenea, importanta pentru a putea gestiona mai multe cozi de clienti simultan. Fiecare coada trebuie sa aiba propriul ei timp de asteptare si timp de servire, iar clientii trebuie asezati in cozi in functie de numarul lor de produse sau de alte criterii.

Am creat o interfata grafica intuitiva care permite utilizatorilor sa isi configureze propria lor simulare. Interfata permite utilizatorilor sa introduca anumite date cum ar fi numarul de cozi, numarul de clienti, timpul de sosire al fiecarui client sau timpul de servire.

5.Bibliografie

1. "Unified Modeling Language." Wikipedia, The Free Encyclopedia. Wikimedia Foundation, Inc., 15 Jul. 2021.
2. "Simulating Queues in Java." Java Code Geeks, 13 May. 2020, www.javacodegeeks.com/2019/05/simulating-queues-in-java.html.