



UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

DOCUMENTAȚIE QUEUE MANAGEMENT SYSTEM



Iepure Denisa-Alexandra

grupa: 30223



CUPRINS:

1.Obiectivele proiectului

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

3.Proiectare

4.Implementare

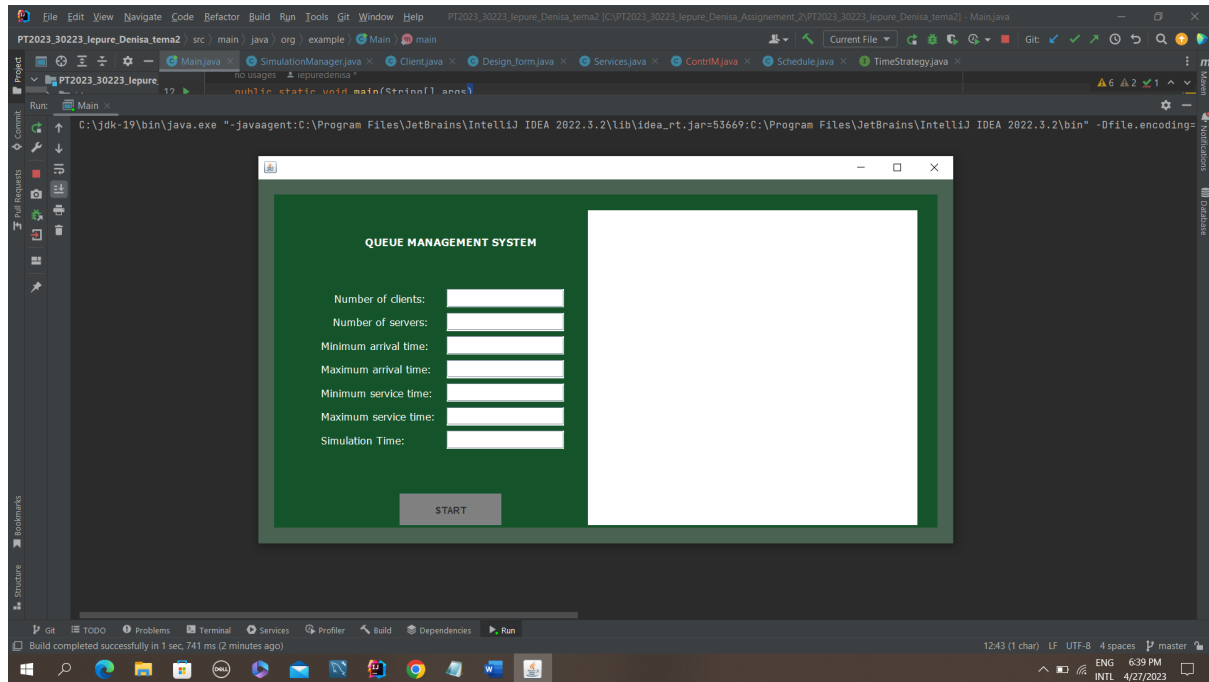
5.Rezultatele testării

6.Concluzii

7.Rezultate fisier



1. Obiectivele proiectului



Această aplicație de cozi este proiectată pentru a oferi utilizatorilor posibilitatea de a urmări în timp real sistemul de gestionarea unor cozi în funcție de timpul curent, timpul de ajungere al clientilor și cel de procesare. Pe scurt, proiectul poate fi asociat în viața reală cu casele de marcat de la un supermarket, reprezentate în proiect de lista de cozi, iar clienții programați să vină în acea zi sunt reprezentați de lista de clienți(caracterizati prin id, arrivalTime, serviceTime). Interfața grafică permite utilizatorilor să introducă timpul de functionare, cați clienți o să fie în acea zi, intervalul acestor de ajungere și intervalul de procesare a unui client.

Obiectivele secundare(Pașii urmați)

- **Analiza problemei**

Cozile funcționează după principiul FIFO(first in first out), analizand problema ne dăm seama că se încearcă eficientizarea timpului de stat la coada prin intermediul găsirii cozii cu waiting time cel mai puțin. Exemplu: mergi la un supermarket cu mai multe case de marcat, vei alege sa stai la coada care are clienții cu cele mai puține



produse pe bandă. Mai observăm faptul că atât clienții, cât și cozile lucrează în paralel, de aceea ne vom folosi de THREADURI.

Un thread este o secvență separată de execuție în cadrul unui program care poate rula în paralel cu alte thread-uri. Mai multe thread-uri pot rula în paralel, partajând o zonă comună de memorie

- **Proiectare**

După identificarea cerințelor, se proiectează interfața grafică a aplicației, astfel încât utilizatorul să poată introduce și gestiona această aplicație. Această etapă include designul, testFieldurile, label-urile, tot ce vede utilizatorul.

- **Implementarea**

Implementăm o coadă, pe care încercăm să o particularizăm, aceasta va avea înăuntrul ei o listă de clienți, un ID, iar în funcție de câți clienți sunt în ea, va avea un Waiting Time, acest lucru ajutându-ne să aflăm la un moment dat care coadă are Waiting Time-ul minim, astfel gestionând eficientizarea programului.

Implementarea unor threaduri pentru operațiile care se desfășoară în coadă.

Implementarea unei interfețe cu fielduri și textFielduri pentru completarea de către utilizator a acestora.

După ce am proiectat interfața grafică și clasele noastre, putem începe să implementăm codul pentru a face funcțional acest sistem de gestionare al cozilor.

Pentru a da viață sistemului, implementăm interfața ActionListener și vom suprascris metoda actionPerformed(), pentru butonul de start.

- **Testarea**

Această etapă presupune introducerea unor valori, și verificarea datelor de ieșire. În acest proiect este important să vizualizăm următoarele:

- > în momentul în care arrivalTime-ul clientului este egal cu currentTime trebuie să ștergem clientul din WaitingListClients și să îl introducem în coada corespunzătoare timpului de așteptare minim

- > serviceTime-ul scade odată ce crește currentTime

- > în momentul în care serviceTime ajunge pe 0, clientul trebuie scos din coada respectivă

2. ANALIZA PROBLEMEI, MODELARE, SCENARIU, CAZURI DE UTILIZARE



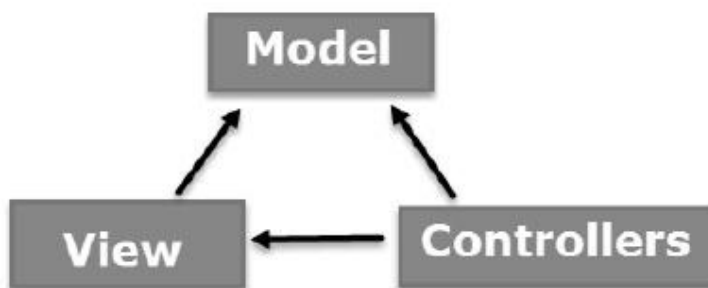
Cerințe funcționale

- ☐ Este nevoie de o aplicație care să simuleze un sistem bazat pe coadă pentru a analiza și minimiza timpul de așteptare al clienților. Aplicația are un număr de clienți și va urmări timpul petrecut de fiecare client în coadă pentru a calcula timpul mediu de așteptare. Fiecare client va fi caracterizat prin ID-ul său, timpul de simulare când este gata să meargă în coadă și timpul necesar pentru a fi servit. Clienții vor fi adăugați la coadă cu timp minim de așteptare. Scopul aplicației este de a minimiza timpul de așteptare al clienților prin gestionarea eficientă a sistemului bazat pe coadă.

3.PROIECARE

Unified Modeling Language (UML) este un limbaj standard pentru descrierea de modele și specificații pentru software. UML a fost la bază dezvoltat pentru reprezentarea complexității programelor orientate pe obiect, al căror fundament este structurarea 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.

Model-View-Controller este un model arhitectural care separă o aplicație (în cazul nostru calculatorul de polinoame) în trei componente principale: **model**, **view**, **controller**. Fiecare componentă este construită să gestioneze diferite părți ale aplicației.





4.IMPLEMENTARE

În pachetul Model avem următoarele clase : Client, Schedule , Service, SimulationManager și interfața TimeStrategy.

- Clasa Client

❖ atribute:

```
private int idClient; // ID-ul , unic
private int arrivalTime; timpul la care acesta trebuie să intre într-o coadă
private int serviceTime; //timpul de servire
```

-> cel mai important lucru care se desfășoară în această clasă este compararea a doi clienți pe baza timpului în care aceștia ajung , astfel va fi mai ușor să îi adăugăm în coadă la momentul corespunzător.

- Clasa Service

❖ atribute:

```
private BlockingQueue<Client> queueClients;
private AtomicInteger waitingPeriod;
private int nb; // numarul cozii
private boolean isEmpty;
private boolean exist=true;
```

-> una dintre clasele de bază ale acestui proiect , locul unde clienții așteaptă să fie serviți . Utilizând threadurile trebuie să implementăm Runnable, astfel în metoda run se întâmplă întreaga acțiune. Clientul care se află în față, ceea ce înseamnă că la fiecare secundă el este tot mai aproape să iasă din coadă , deci decrementăm serviceTime-ul , dar înainte adormim threadul o secundă , în timp ce alți clienți pot fi adăugați . Clientul va fi eliminat din coadă atunci când i se termina timpul de procesare.

- Clasa Schedule

->Clasa "Schedule" este utilizată pentru a programa și organiza cozile, în vederea minimizării timpului de așteptare al clienților. Aceasta conține o listă de servere care oferă servicii și numărul acestora. Când un client trebuie să fie adăugat într-o coadă, programul găsește coada cu cel mai mic timp de așteptare și adaugă clientul acolo. De asemenea, programul poate calcula timpul mediu de așteptare și poate verifica dacă toate cozile sunt goale. Clasa utilizează interfața



"TimeStrategy" pentru a implementa strategia de timp. Constructorul clasei inițializează serverele și pornește firele de execuție asociate lor.

- Clasa "SimulationManager
->" implementează logica simulării unui sistem de cozi cu ajutorul thread-urilor și a interfeței grafice Swing. Aceasta gestionează evenimentele generate de sosirea și eliberarea clienților din cozi, determinând timpii de așteptare și de servire, respectând restricțiile impuse de timpul de lucru al serviciilor și de timpul limită al simulării. De asemenea, clasa se ocupă și de afișarea și scrierea în fișier a rezultatelor simulării.

In pachetul Controller avem legatura dintre Interfata grafica și Model
In pachetul View avem strict partea legată de interfață.

6.Concluzii

Crearea unui sistem de distribuție a clienților, astfel încât timpul petrecut în așteptarea la coadă să fie redus. Pentru a simula modul real de servire a clienților într-un spațiu comercial cu mai multe case de marcat, aplicația utilizează thread-uri. Folosind tehnica de multi-threading, adăugarea și ștergerea simultană a clienților din mai multe cozi poate fi realizată eficient.

7.Rezultate teste



test1.txt - Notepad

File Edit Format View Help

```
|-----Time limit:60
Current time:0
Waiting clients:(3-4-3) (1-11-2) (4-18-3) (2-30-3)
Queue 1: goala
Queue 2: goala
-----Time limit:60
Current time:1
Waiting clients:(3-4-3) (1-11-2) (4-18-3) (2-30-3)
Queue 1: goala
Queue 2: goala
-----Time limit:60
Current time:2
Waiting clients:(3-4-3) (1-11-2) (4-18-3) (2-30-3)
Queue 1: goala
Queue 2: goala
-----Time limit:60
Current time:3
Waiting clients:(3-4-3) (1-11-2) (4-18-3) (2-30-3)
Queue 1: goala
Queue 2: goala
-----Time limit:60
Current time:4
Waiting clients:(1-11-2) (4-18-3) (2-30-3)
Queue 1: (3-4-3)
Queue 2: goala
-----Time limit:60
Current time:5
Waiting clients:(1-11-2) (4-18-3) (2-30-3)
Queue 1: (3-4-2)
Queue 2: goala
-----Time limit:60
Current time:6
Waiting clients:(1-11-2) (4-18-3) (2-30-3)
Queue 1: (3-4-1)
Queue 2: goala
-----Time limit:60
Current time:7
Waiting clients:(1-11-2) (4-18-3) (2-30-3)
```




test1.txt - Notepad

File Edit Format View Help

Queue 2: goala

-----Time limit:60

Current time:8

Waiting clients:(1-11-2) (4-18-3) (2-30-3)

Queue 1: goala

Queue 2: goala

-----Time limit:60

Current time:9

Waiting clients:(1-11-2) (4-18-3) (2-30-3)

Queue 1: goala

Queue 2: goala

-----Time limit:60

Current time:10

Waiting clients:(1-11-2) (4-18-3) (2-30-3)

Queue 1: goala

Queue 2: goala

-----Time limit:60

Current time:11

Waiting clients:(4-18-3) (2-30-3)

Queue 1: (1-11-2)

Queue 2: goala

-----Time limit:60

Current time:12

Waiting clients:(4-18-3) (2-30-3)

Queue 1: (1-11-1)

Queue 2: goala

-----Time limit:60

Current time:13

Waiting clients:(4-18-3) (2-30-3)

Queue 1: goala

Queue 2: goala

-----Time limit:60

Current time:14

Waiting clients:(4-18-3) (2-30-3)

Queue 1: goala

Queue 2: goala

.



test1.txt - Notepad

File Edit Format View Help

-----Time limit:60

Current time:15

Waiting clients:(4-18-3) (2-30-3)

Queue 1: goala

Queue 2: goala

-----Time limit:60

Current time:16

Waiting clients:(4-18-3) (2-30-3)

Queue 1: goala

Queue 2: goala

-----Time limit:60

Current time:17

Waiting clients:(4-18-3) (2-30-3)

Queue 1: goala

Queue 2: goala

-----Time limit:60

Current time:18

Waiting clients:(2-30-3)

Queue 1: (4-18-3)

Queue 2: goala

-----Time limit:60

Current time:19

Waiting clients:(2-30-3)

Queue 1: (4-18-2)

Queue 2: goala

-----Time limit:60

Current time:20

Waiting clients:(2-30-3)

Queue 1: (4-18-1)

Queue 2: goala

-----Time limit:60

Current time:21

Waiting clients:(2-30-3)

Queue 1: goala


Queue 2: goala

-----Time limit:60

Current time:22

Waiting clients:(2-30-3)



 test1.txt - Notepad

File Edit Format View Help

Queue 2: goala

-----Time limit:60

Current time:23

Waiting clients:(2-30-3)

Queue 1: goala

Queue 2: goala

-----Time limit:60

Current time:24

Waiting clients:(2-30-3)

Queue 1: goala

Queue 2: goala

-----Time limit:60

Current time:25

Waiting clients:(2-30-3)

Queue 1: goala

Queue 2: goala

-----Time limit:60

Current time:26

Waiting clients:(2-30-3)

Queue 1: goala

Queue 2: goala

-----Time limit:60

Current time:27

Waiting clients:(2-30-3)

Queue 1: goala

Queue 2: goala

-----Time limit:60

Current time:28

Waiting clients:(2-30-3)

Queue 1: goala

Queue 2: goala

-----Time limit:60

Current time:29

Waiting clients:(2-30-3)

Queue 1: goala

Queue 2: goala



test1.txt - Notepad

File Edit Format View Help

Current time:25

Waiting clients:(2-30-3)

Queue 1: goala

Queue 2: goala

-----Time limit:60

Current time:26

Waiting clients:(2-30-3)

Queue 1: goala

Queue 2: goala

-----Time limit:60

Current time:27

Waiting clients:(2-30-3)

Queue 1: goala

Queue 2: goala

-----Time limit:60

Current time:28

Waiting clients:(2-30-3)

Queue 1: goala

Queue 2: goala

-----Time limit:60

Current time:29

Waiting clients:(2-30-3)

Queue 1: goala

Queue 2: goala

-----Time limit:60

Current time:30

Waiting clients:

Queue 1: (2-30-3)

Queue 2: goala

-----Time limit:60

Current time:31

Waiting clients:

Queue 1: (2-30-2)

Queue 2: goala

-----Time limit:60

Current time:32

Waiting clients:

Queue 1: (2-30-1)

Queue 2: goala



UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA
