MINISTERUL EDUCAȚIEI ȘI CERCETĂRII ȘTIINȚIFICE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
CATEDRA CALCULATOARE

DOCUMENTAȚIE

TEMA *2*

GESTIONAREA COZILOR

CIOBAN FABIAN-REMUS
30223

CUPRINS

1. Obiectivul temei 3
2. Analiza problemei, modelare, scenarii, cazuri de utilizare 3
3. Proiectare 4
4. Implementare 5
5. Rezultate 7
6. Concluzii 7
7. Bibliografie 8

1. Obiectivul temei

Această temă implică implementarea unei aplicații de gestionare a coziilor, cu scopul de a adăuga clienții în coada cu cel mai mic timp de așteptare. Programul va fi realizat într-o interfață grafică, în care datele de intrare, cum ar fi numărul de cozi, numărul de clienți, timpul minim și maxim de așteptare, timpul minim și maxim de procesare și timpul maxim de simulare, pot fi introduse. După introducerea datelor de intrare, programul va genera informații despre fiecare client cu parametrii respectivi.

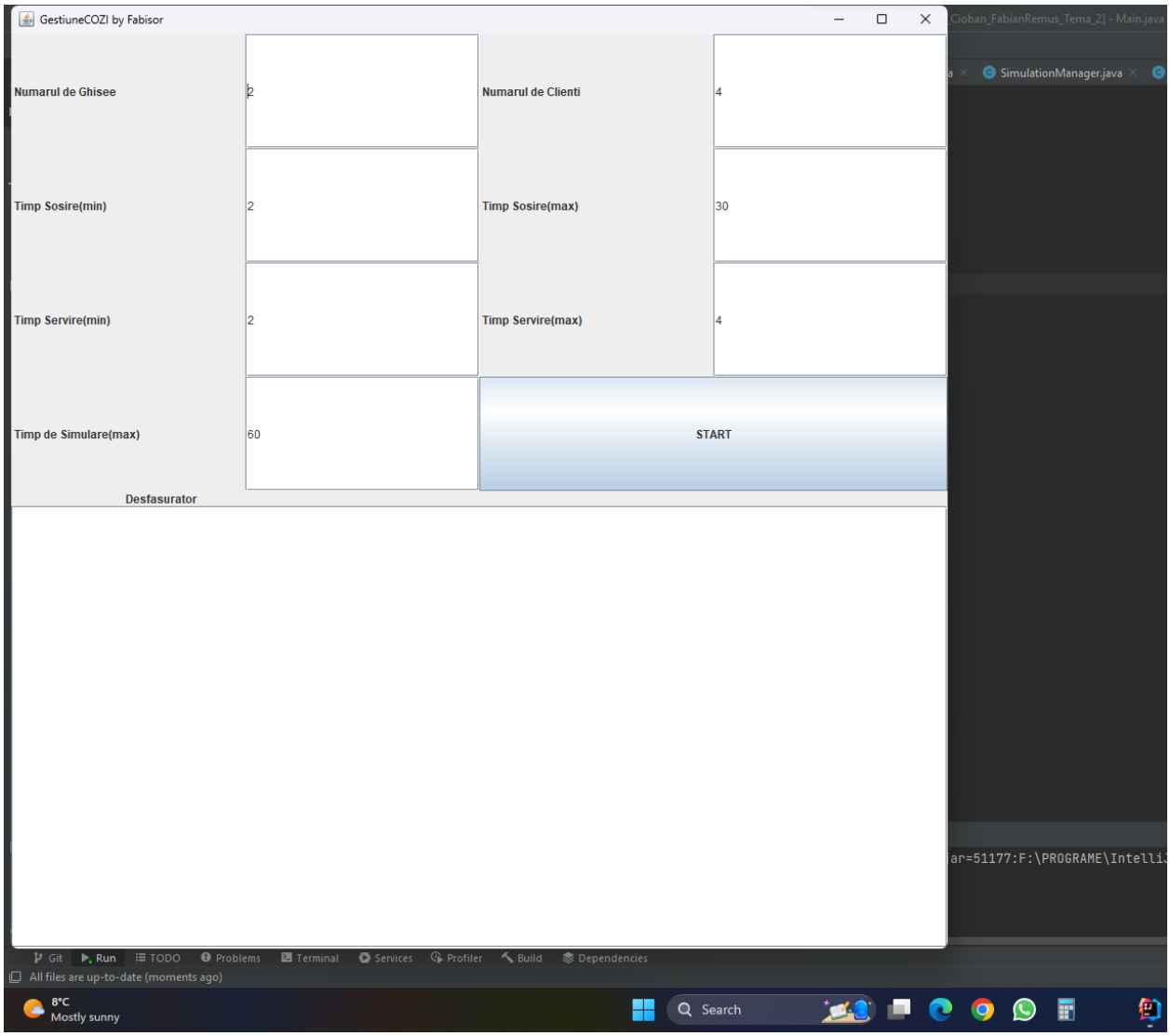
2. Analiza problemei, modelare, scenarii, cazuri de utilizare

Analizând cerințele programului, se observă că acesta trebuie să adauge clienți în coada cu cel mai mic timp de așteptare și să scoată clienții care au terminat din cozi, în fiecare secundă de simulare.

În ceea ce privește modelarea, se utilizează structura de date "Task" pentru a reprezenta clienții, care conține atributele specifice, cum ar fi id-ul, timpul de sosire și timpul de procesare. Pe de altă parte, structura de date "Server" reprezintă cozile, care conțin coada de clienți și timpul total de așteptare.

În privința scenariilor, se poate observa că dacă timpul de simulare se termină și încă există clienți în cozi, programul se oprește. Este important ca timpul minim de procesare să fie mai mic sau egal cu timpul maxim de procesare, la fel ca și în cazul timpului de sosire. În cazul în care datele introduse nu sunt valide, utilizatorul va primi o atenționare.

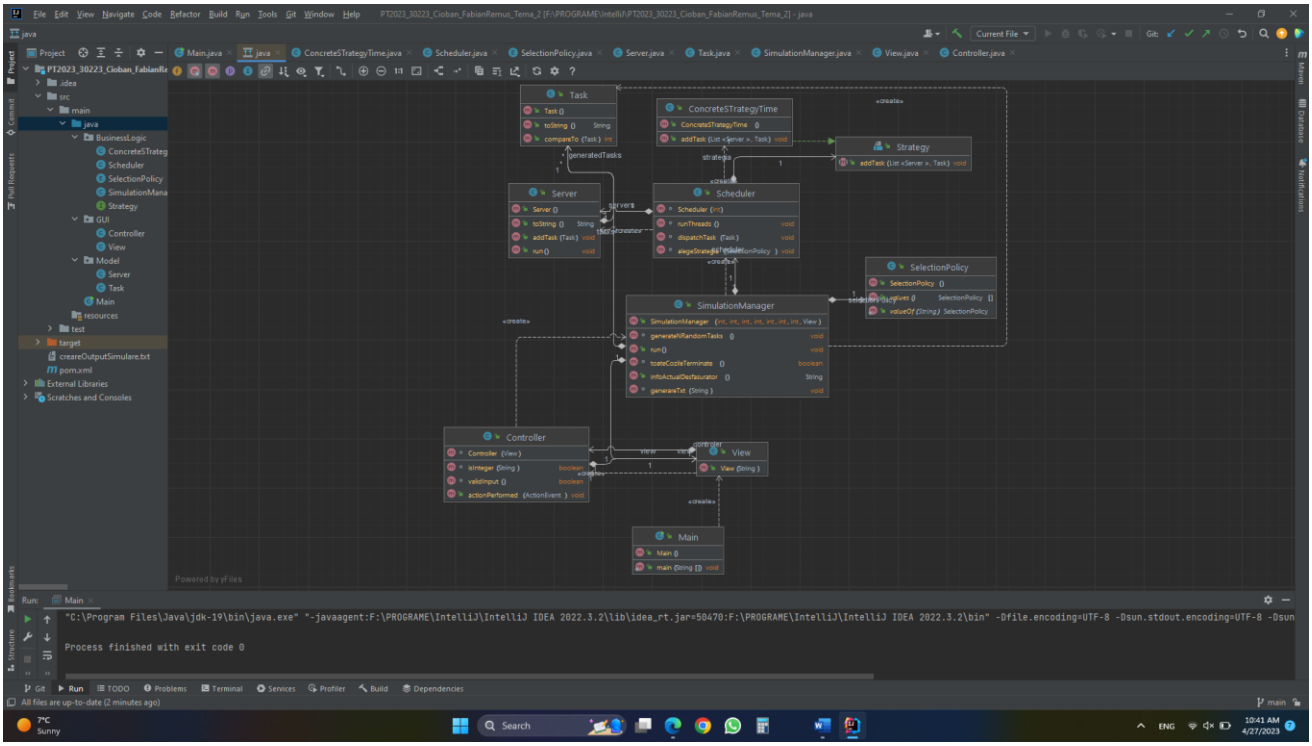
În interfața grafică, datele pot fi introduse în partea stângă a ferestrei, iar după inserarea lor, utilizatorul poate apăsa butonul "Run", care va porni simularea cu parametrii respectivi. Informațiile despre simulare vor fi afișate în desfășurător-ul din partea dreaptă a ferestrei.



3. Proiectare

Diagrama UML

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.



La ce revine analiza problemei? Este foarte simplu, avem nevoie să gestionăm cozile(de exemplu când stai la un ghișeu la coadă) pentru a eficentiza timpul.

4. Implementare

Am decis sa implementez clasele cat mai intuitiv posibil, cu un cod cat mai lizibil si cat mai usor de inteles deoarece in viitor va fi nevoie ca o ehipa intreaga sa inteleaga codul pentru a putea lucra toti la acel proiect. O sa descriu fiecare clasa pe rand:

Clasa Main este punctul de intrare în programul nostru. În cazul nostru, clasa Main creează o nouă instanță a clasei View. Aceasta este o practică comună în programarea orientată pe obiect, unde crearea obiectelor se realizează de obicei în clasa principală a programului. Clasa Main poate fi considerată punctul central din care se pornește întregul program.

Clasa View este responsabilă pentru crearea și afișarea interfeței grafice a aplicației noastre. Această clasă este importantă, deoarece interacțiunea utilizatorului cu aplicația noastră se realizează prin intermediul interfeței grafice. Prin intermediul clasei View, putem crea toate componentele de interfață necesare, cum ar fi panourile, butoanele, casetele de text. Aceste componente sunt atributele clasei View și sunt definite în constructorul acesteia.

Clasa Controller are rolul de a prelua datele introduse de utilizator prin intermediul interfeței grafice și de a le transmite mai departe către clasa de simulare.

Clasa Task este o clasă ce reprezintă un client într-un sistem de cozi. Această clasă conține atribute specifice, cum ar fi un identificator unic pentru fiecare client, momentul la care acesta ajunge la coadă și timpul necesar pentru a fi procesat. Clasa Task este un element important al modelării sistemului, deoarece permite crearea unui obiect ce conține toate informațiile necesare pentru a putea fi plasat într-o coadă și prelucrat în mod corespunzător.

Clasa Server este o componentă importantă a sistemului de cozi. Această clasă este responsabilă de gestionarea cozilor și a clienților care așteaptă în acestea. Clasa Server implementează interfata Runnable, ceea ce permite executarea în paralel a mai multor instanțe de server, astfel încât să poată fi gestionate mai multe cozi simultan. Clasa Server conține metode specifice pentru adăugarea, ștergerea și prelucrarea clienților din coadă, precum și metode pentru a determina starea curentă a cozilor.

Interfața Strategy defineste o metodă de adaugare a unui client în cadrul sistemului de cozi. Această interfață este apelată din clasa Scheduler utilizând o anumită strategie, care poate fi determinată fie în funcție de timpul de așteptare al cozilor. Utilizarea interfeței Strategy permite sistemului de cozi să fie flexibil și să adapteze dinamic strategia de adăugare a clienților, în funcție de nevoile și condițiile de utilizare.

Clasa ConcreteSTrategyTime implementează interfața Strategy definită de programator. Această clasă conține doar o metodă care inserează un client în coada cu timpul de așteptare minim. Această strategie se bazează pe calculul timpului de așteptare pentru fiecare coadă și plasarea clientului în coada cu cel mai mic timp de așteptare.

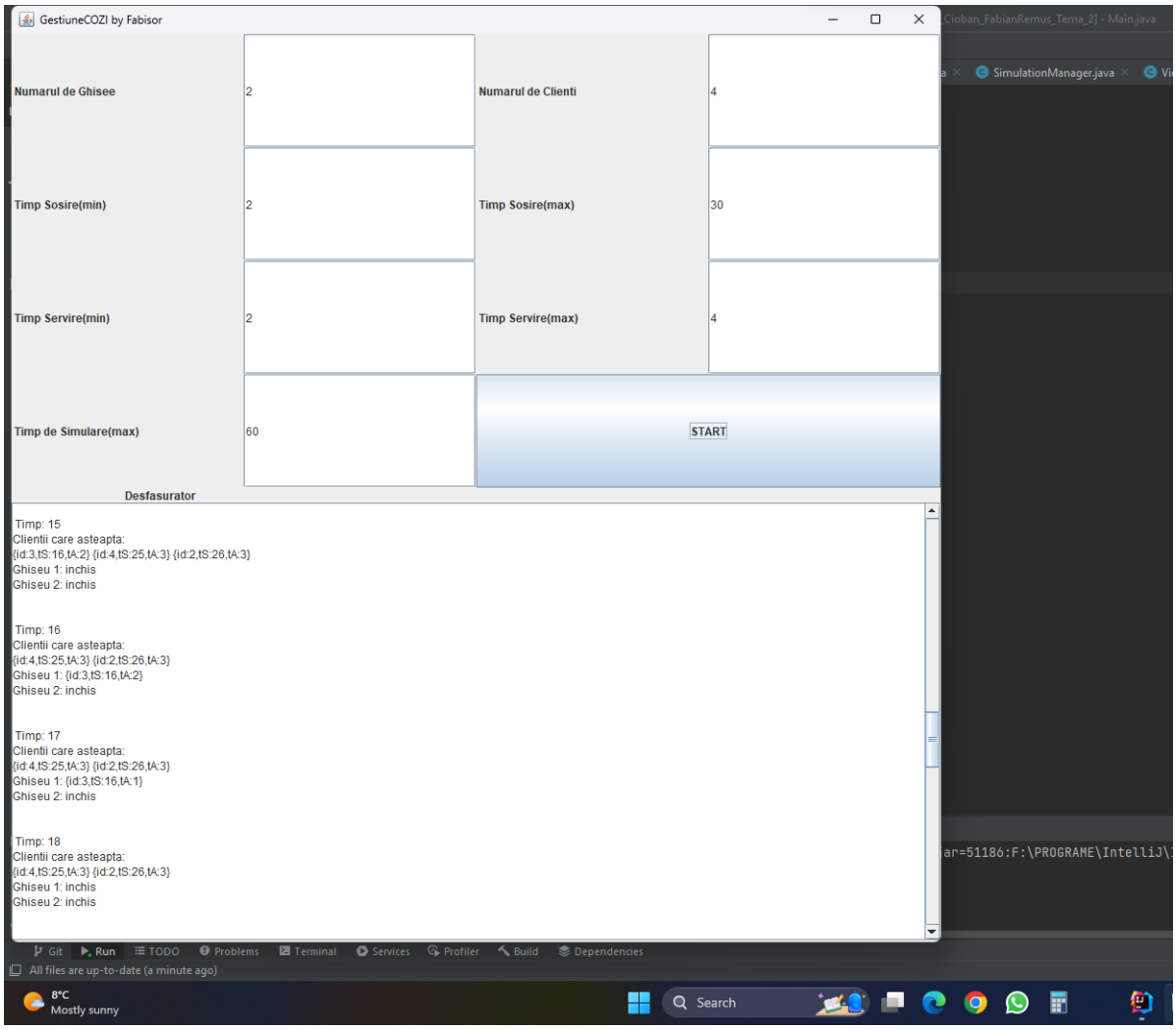
Clasa Scheduler reprezintă un element important al sistemului de cozi, deoarece este responsabilă cu organizarea și gestionarea fluxului de clienți și a cozilor aferente. Această clasă se ocupă de asocierea fiecărei cozi cu un thread, cu pornirea thread-urilor și cu adăugarea clienților în cozi în funcție de strategie. Prin intermediul constructorului clasei, se realizează asocierea cozilor cu thread-urile, prin instantierea acestora și asignarea lor fiecărei cozi. De asemenea, în constructor se instantiază și strategia de adăugare a clienților în cozi, prin crearea unui obiect de tip ConcreteStrategyTime.

Clasa SimulationManager reprezintă o componentă importantă a sistemului de cozi, deoarece are rolul de a gestiona și controla fluxul de clienți ce urmează să fie procesați în cadrul simulării. Această clasă este responsabilă cu stocarea și gestionarea datelor introduse de utilizator, precum numărul de cozi și numărul de clienți generați pentru simulare. De asemenea, clasa este responsabilă cu generarea celor N clienți, folosind datele stocate, și trimiterea acestora în cozi la momentul potrivit.În afara de aceste responsabilități, clasa SimulationManager se ocupă și de stocarea log-ului, care conține informații importante despre procesul de simulare, cum ar fi timpul de așteptare

și de procesare a fiecărui client sau informații despre erorile apărute. Log-ul poate fi afișat în interfața grafică sau poate fi stocat în fișiere, pentru a fi analizat ulterior.

5. Rezultate

Rezultatele testărilor pot fi găsite în fișierul creat " creareOutputSimulare.txt", sau pot fi văzute direct în aplicație prin rularea simulării din interfața grafică.



6. Concluzii

Această temă mi-a oferit oportunitatea de a-mi aprofunda cunoștințele despre thread-uri și despre modul în care acestea pot fi utilizate pentru a gestiona și a organiza operațiunile într-o aplicație. De asemenea, am învățat cum să creez și să implementez o aplicație de management de tip client-server, care poate prelua date de la utilizatori și să le proceseze, oferind în același timp

o experiență interactivă prin intermediul unei interfețe grafice. Prin această experiență, am dobândit o mai bună înțelegere a modului în care se pot dezvolta și gestiona aplicații complexe și am dezvoltat abilități utile în domeniul programării și al dezvoltării de aplicații.

7. Bibliografie

1.YouTube
2.Wikipedia
3.W3School
4.Oracle
5. Geeksforgeeks