

DOCUMENTATIE

TEMA 2

NUME STUDENT: Jarda Adina-Ionela
GRUPA:30222

CUPRINS

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

1. Obiectivul temei

Obiectivul principal al proiectului este să dezvoltăm un simulator pentru gestionarea cozilor de clienți folosind thread-uri și mecanisme de sincronizare. Printre obiectivele secundare se numără:

- Implementarea a două strategii de distribuție:
 1. Selectarea cozii cu cel mai scurt timp de așteptare.
 2. Selectarea cozii cu cei mai puțini clienți.
- Realizarea unei interfețe grafice pentru a afișa în timp real întreaga simulare.

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

Cerințe Funcționale ale acestui proiect sunt:

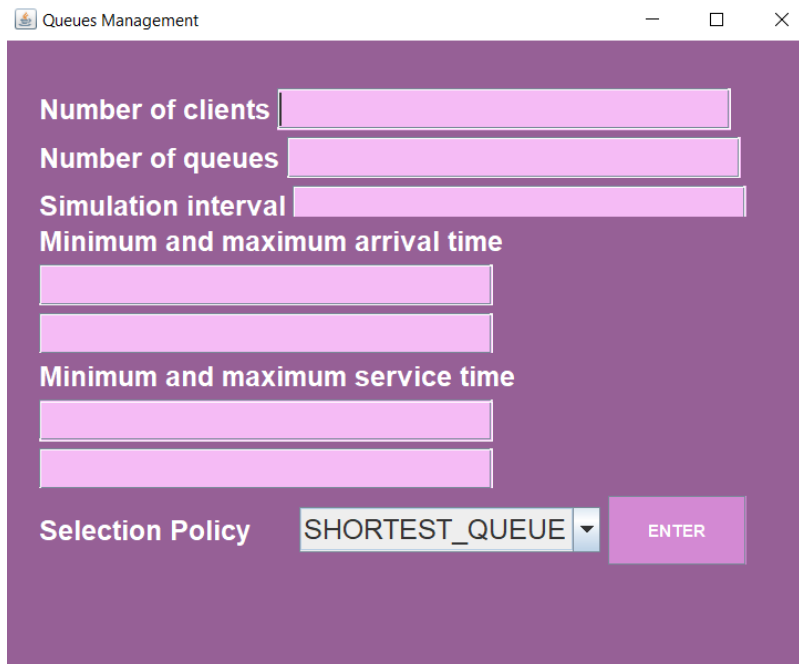
- Generarea unui thread distinct pentru fiecare server.
- Simularea distribuției și procesării serviciilor pentru fiecare client.

Cerințe Non-Funcționale ale acestui proiect sunt:

- Implementarea unei interfețe grafice pentru afișarea în timp real a simulării

Mod de utilizare

1. După rularea simulatorului de către utilizator pe ecran va apărea interfața grafică



Queues Management

Number of clients

Number of queues

Simulation interval

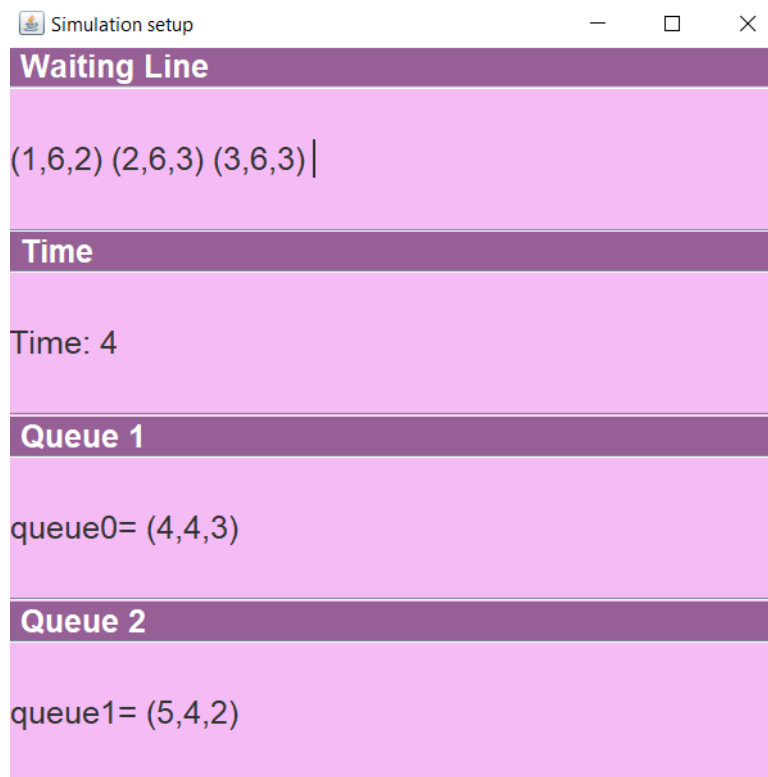
Minimum and maximum arrival time

Minimum and maximum service time

Selection Policy

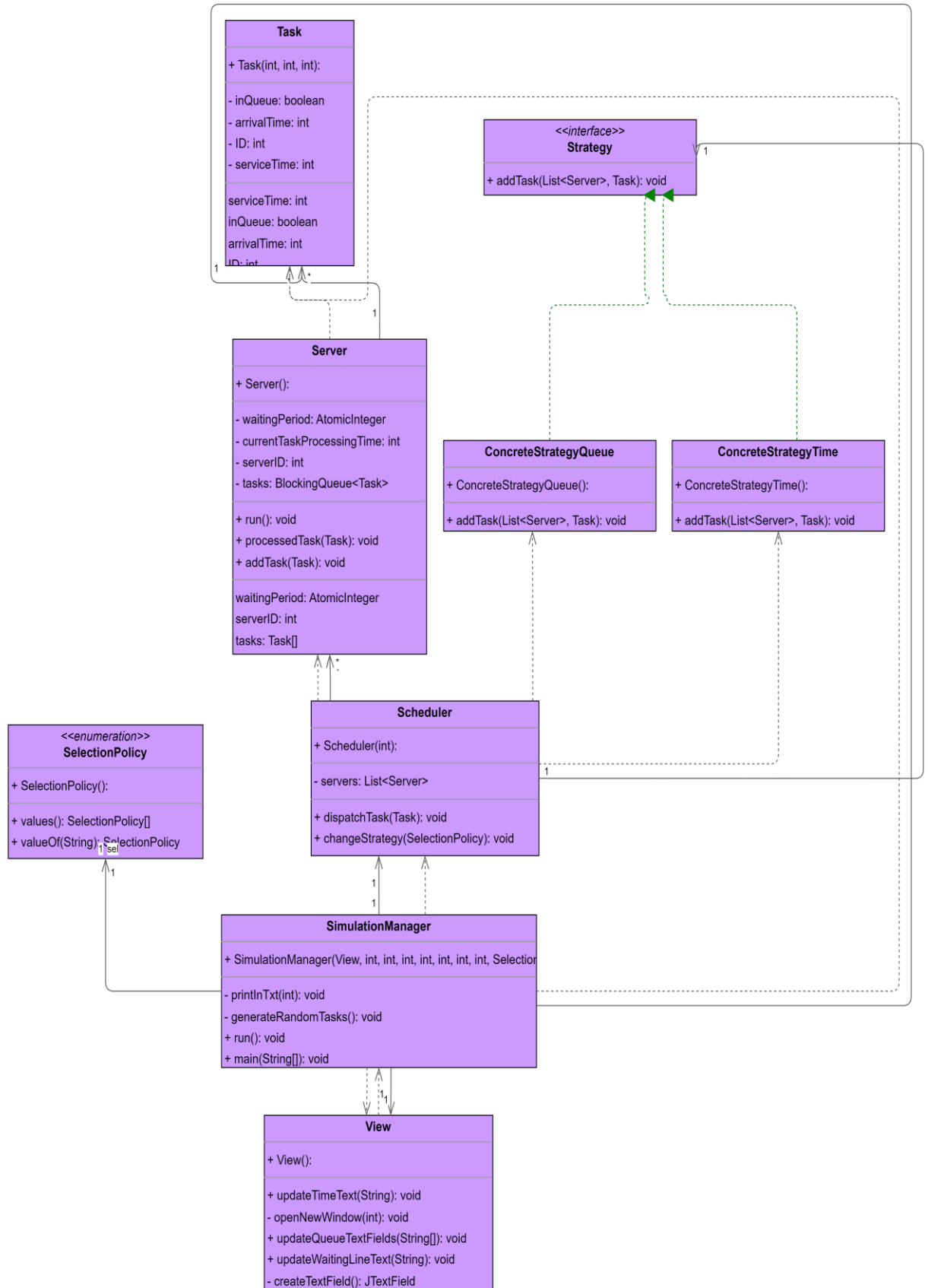
2. In fiecare camp liber se introduc valorile dorite, in caz de introducere unor valori neconforme, pe ecran va apare un mesaj de eroare caracteristic.
3. Din combo-box-ul de jos, se alege strategia de distribuire dorita
4. Se apasa enter pentru inceperea simularii.

Odata ce ai apasat enter se deschide o alta interfata grafica care arata in timp real waiting list-ul, timpul si statusul fiecărei cozi la timpul respectiv.



3. Proiectare

Diagrama UML a claselor, interfetelor si a enum-ului, fara includerea si pachetelor din care acestea fac parte.



4. Implementare

1. **SimulationManager**: Această clasă servește ca epicentrul simulatorului, coordonând fluxul general al execuției. Aici, se inițializează obiectul View, iar metoda run, ulterior apelată din cadrul View, orchestrează afișările și desfășurarea simulării.
2. **View**: În această clasă are loc inițializarea interfeței grafice primare. De aici, se preiau datele introduse, iar cu ajutorul lor se inițializează o a doua interfață grafică. De asemenea, se declanșează procesul simulatorului.
3. **Scheduler**: Această clasă se ocupă de gestionarea cozilor și distribuirea clienților. Practic, pentru fiecare coadă este generat un obiect de tip Server, iar clienții sunt distribuiți către acestea.
4. **Server**: Fiecare coadă este asociată unei liste de clienți de tip Task. Această clasă gestionează procesul de așteptare și procesare a clienților. Fiecare fir de execuție pornit primește ca parametru primul client din listă și este întrerupt pentru perioada de procesare a acestuia. La fiecare secundă de așteptare, timpul de procesare al clientului în curs este scăzut.
5. **Task**: Această clasă reprezintă un client în cadrul simulatorului. Aici este construit obiectul clientului, iar timpul de procesare poate fi modificat.
6. **Clasele ConcreteStrategyTime și ConcreteStrategyQueue**: Aceste două clase implementează interfața Strategy. Ele se ocupă de adăugarea clienților în cozi, în funcție de strategia de distribuire a clienților.
7. **Interfața Strategy**: Această interfață conține definiția metodei addTask, utilizată în clasele ConcreteStrategyTime și ConcreteStrategyQueue.
8. **Enum SelectionPolicy**: Acest enum reprezintă cele două strategii de distribuire a clienților. El furnizează numele în baza cărora se face alegerea între metodele de distribuire.

5. Rezultate

Pentru a valida corectitudinea simulării, am efectuat șase rulări distincte, iar rezultatele au fost înregistrate în șase fișiere de tip .txt.

6. Concluzii

Prin intermediul acestui proiect, am dobândit cunoștințe esențiale în gestionarea thread-urilor și sincronizarea lor, prin utilizarea DeblockingDeque pentru stocarea și gestionarea cozilor. O evoluție potențială în viitor ar putea consta în exploatarea atributului waitingPeriod din clasa Server, care ar permite monitorizarea timpului de așteptare actual. Această funcționalitate ar fi deosebit de valoroasă în calculul timpului mediu de așteptare și în procesul de distribuire a clienților din cadrul clasei ConcreteStrategyTime.

7. Bibliografie

https://dsrl.eu/courses/pt/materials/PT2024_A2.pdf

https://dsrl.eu/courses/pt/materials/PT_2024_A2_S1.pdf

https://dsrl.eu/courses/pt/materials/PT_2024_A2_S2.pdf