DOCUMENTATIE

TEMA NUMARUL_II

CUPRINS

NUME STUDENT: IRIMES DAVID
GRUPA: 30227

1. Obiectivul temei

• ***Obiectiv principal:***

— *Proiectarea și implementarea unei aplicații menită să analizeze sistemele bazate pe așteptare prin (1) simularea unei serii de N clienți care vin pentru serviciu, intră în Q cozi, așteaptă, sunt serviți și în cele din urmă părăsesc cozile, și (2) calcularea timpului mediu de așteptare, timpului mediu de servire și orei de vârf.*

• ***Sub-obiective:***

— *Analiza problemei și identificarea cerințelor → cap.2*

— *Proiectarea aplicației de simulare → cap.3*

— *Implementarea aplicației de simulare → cap.4*

— *Testarea aplicației de simulare/Rezultate → cap.5*

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

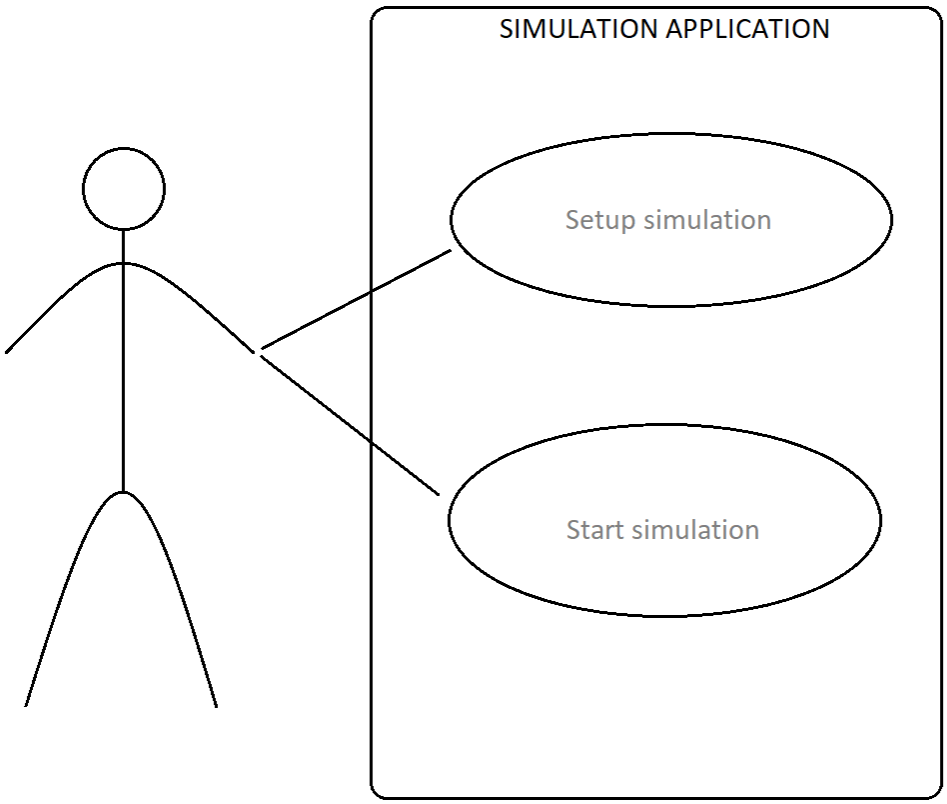


Figure 1

- **Cerințe funcționale**:

— Aplicația de simulare ar trebui să permită utilizatorilor să configureze simularea

— Aplicația de simulare ar trebui să permită utilizatorilor să înceapă simularea

— Aplicația de simulare ar trebui să afișeze evoluția cozilor în timp real

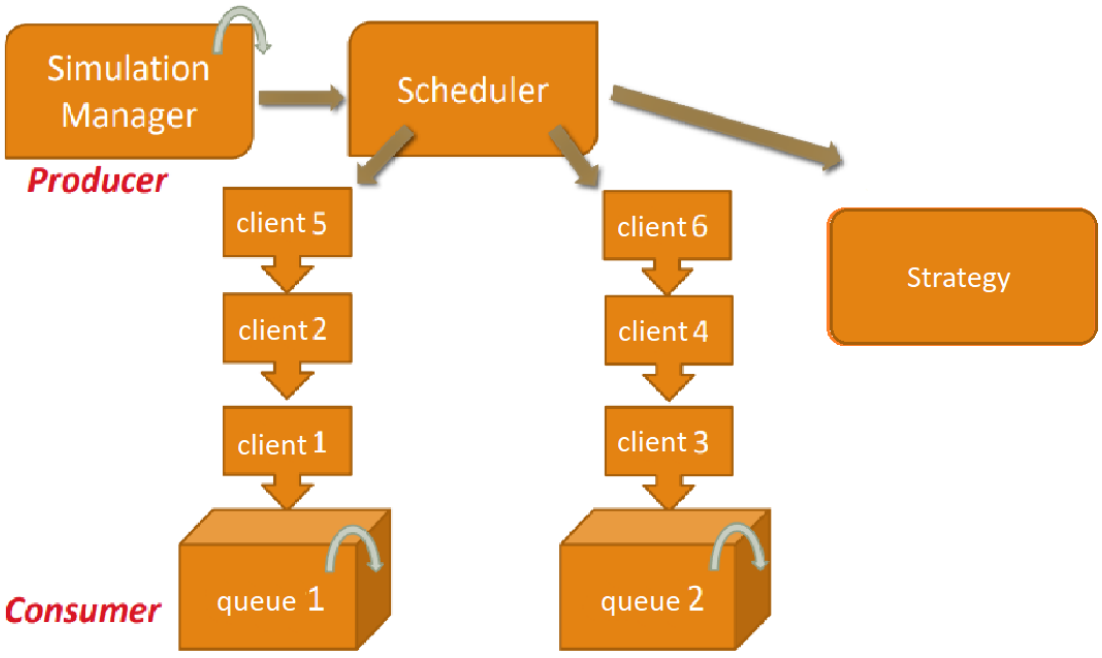
- **Cerințe non-funcționale:**

— Aplicația de simulare ar trebui să fie intuitivă și ușor de utilizat de către utilizator
— Se identifica si se instiinteaza utilizatorul cu vedere la cazurile exceptionale.
— Se pun niste limite pentru anumite inputuri (ex: numrul maxim de cozi)

- Scenariu principal de succes:
  - — Utilizatorul introduce valorile pentru: numărul de clienți, numărul de cozi, intervalul de simulare, timpul minim și maxim de sosire, și timpul minim și maxim de servire.
  - — Utilizatorul apasă pe butonul de start a datelor introduse.
  - — Aplicația nu prezinta nicio exceptie si se incepe simularea.
- Scenariu alternativ:
  - — Valori invalide pentru parametrii de configurare, cum ar fi valori negative, minime mai mari decat maxime, numar de cozi maxim depasite, etc;
  - — Aplicația prezinta utilizatorului un caz de exceptie printr-un mesaj de tip dialog box.

3. Proiectare

Principiu de functionare:

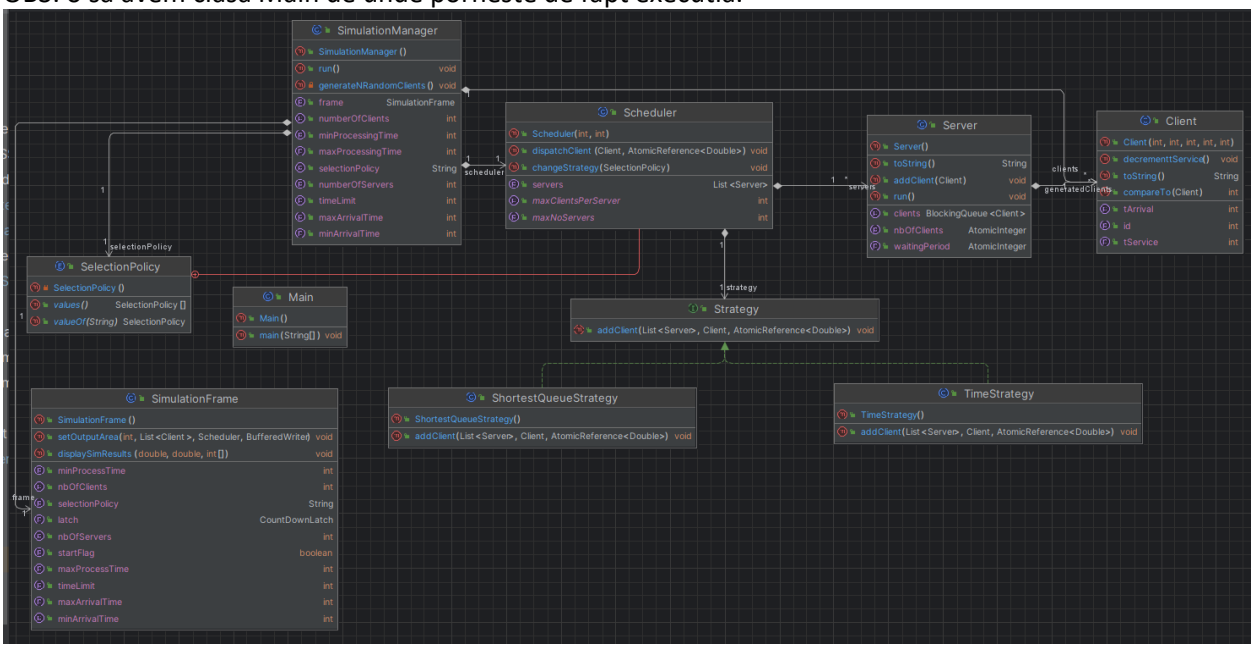


Tema se imparte in trei sectiuni, astfel definind pachetele:

— partea de interfata grafica (GUI), unde utilizatorul va avea contact cu aplicatia ➔ aceasta se va realiza in pachetul cu acelasi nume (se va lucra cu Java Swing); OBS: aceasta clasa va fi instantiata ca parte din clasa de Simulation Manager.

— partea de logica/implementare de operatii, unde se adauga de fapt functionalitatile de baza ale aplicatiei ➔ aceasta se va regasi in pachetul Model, unde se defineste o clasa Server, care reprezinta de fapt o coada, si clasa Client. OBS: se vor folosi in proiect variabile compatibile cu lucrul cu thread-uri (atomice); ex: AtomicInteger, BlockingQueue (pentru cozi), etc.
— partea de Business Logic, reprezinta punctul de convergenta al aplicatiei, unde se vor instantia si pregati toate obiectele necesare si unde se va incheia si main thread-ul impreuna cu toate celelalte. De aici se va trimite spre GUI informatia la fiecare secunda, care trebuie printata in interfata grafica si in fisierele de jurnal.

OBS: o sa avem clasa Main de unde porneste de fapt executia.



4. Implementare

Clasa Client:

— reprezintă un client în cadrul sistemului de cozi. Are atribute precum ID-ul clientului, timpul de sosire și timpul de servire. Constructorul generează aleator valorile pentru aceste atribute într-un interval specificat. Metodele permit accesul și manipularea acestor atribute, iar implementarea interfeței Comparable permite compararea clienților după timpul lor de sosire pentru ca o sa trebuiasca sa ii sortam intr-o lista.

Clasa Server:

— reprezintă un server în cadrul sistemului de cozi. Are o coadă de clienți, un contor pentru perioada de așteptare și un contor pentru numărul de clienți. Constructorul initializează aceste variabile. Metoda addClient() adaugă un client nou în coadă și actualizează contoarele. Metoda run() rulează un thread care gestionează servirea clienților în mod continuu. Aceasta verifică și procesează clienții în funcție de timpul lor de servire. Metodele getWaitingPeriod(), getNbOfClients() și getClients() permit accesul la variabilele interne ale serverului. Metoda toString() returnează o reprezentare a serverului sub formă de șir de caractere, inclusiv lista de clienți din coadă.

—

Clasa Scheduler:

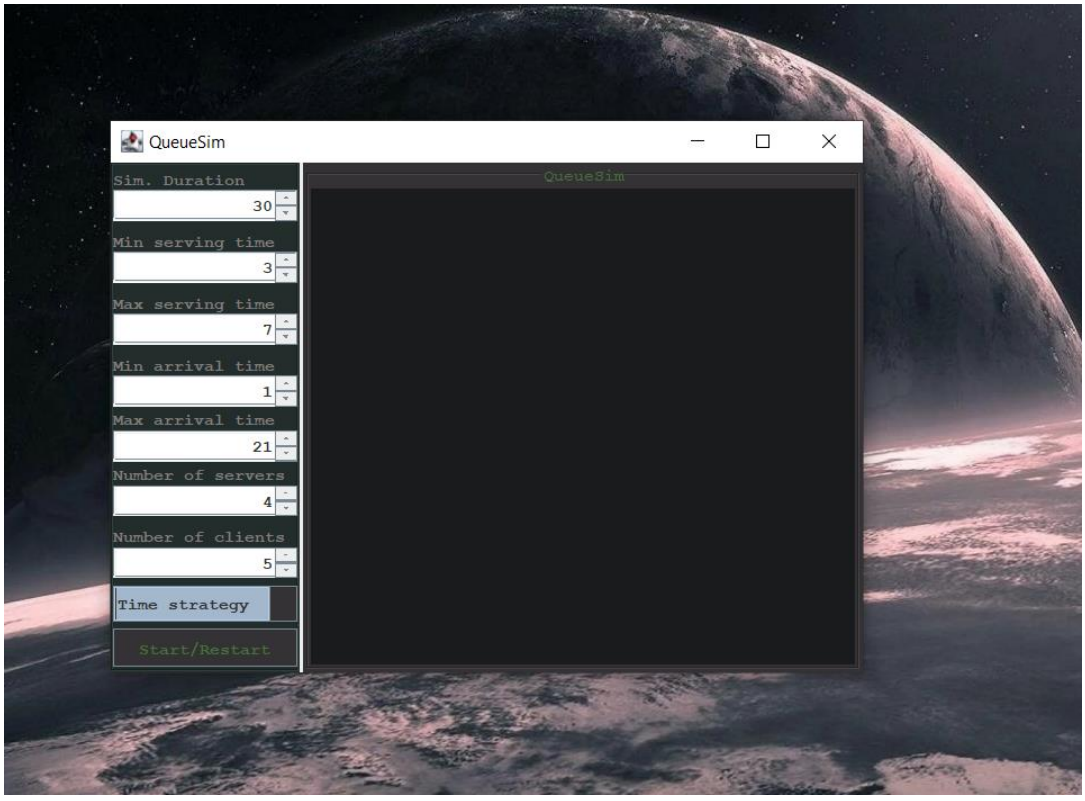
- reprezintă planificatorul în cadrul sistemului de cozi. Are o listă de servere, un număr maxim de servere și un număr maxim de clienți per server, precum și o strategie pentru gestionarea distribuției clienților. Constructorul primește numărul maxim de servere și numărul maxim de clienți per server și inițializează variabilele. Metoda changeStrategy() schimbă strategia de selecție a serverelor în funcție de politica specificată (coadă mai scurtă sau timp de așteptare mai scurt). Metoda dispatchClient() trimite un client către serverul potrivit în funcție de strategie (fie ShortestQueueStrategy fie TimeStrategy) și actualizează timpul mediu de așteptare. Enumerația SelectionPolicy definește politicile posibile pentru selectarea serverelor (coadă mai scurtă sau timp de așteptare mai scurt)

Clasa SimulationManager:

- reprezintă un manager de simulare pentru sistemul de cozi. Acesta inițializează și controlează simularea, gestionând interacțiunea cu utilizatorul prin intermediul unei interfețe grafice. În timpul simulării, generează clienți aleatori și îi distribuie către servere. Monitorizează evoluția sistemului în timp real, actualizând interfața grafică și înregistrând datele relevante într-un fișier de jurnal. După încheierea simulării, afișează rezultatele și închide fișierul de jurnal.

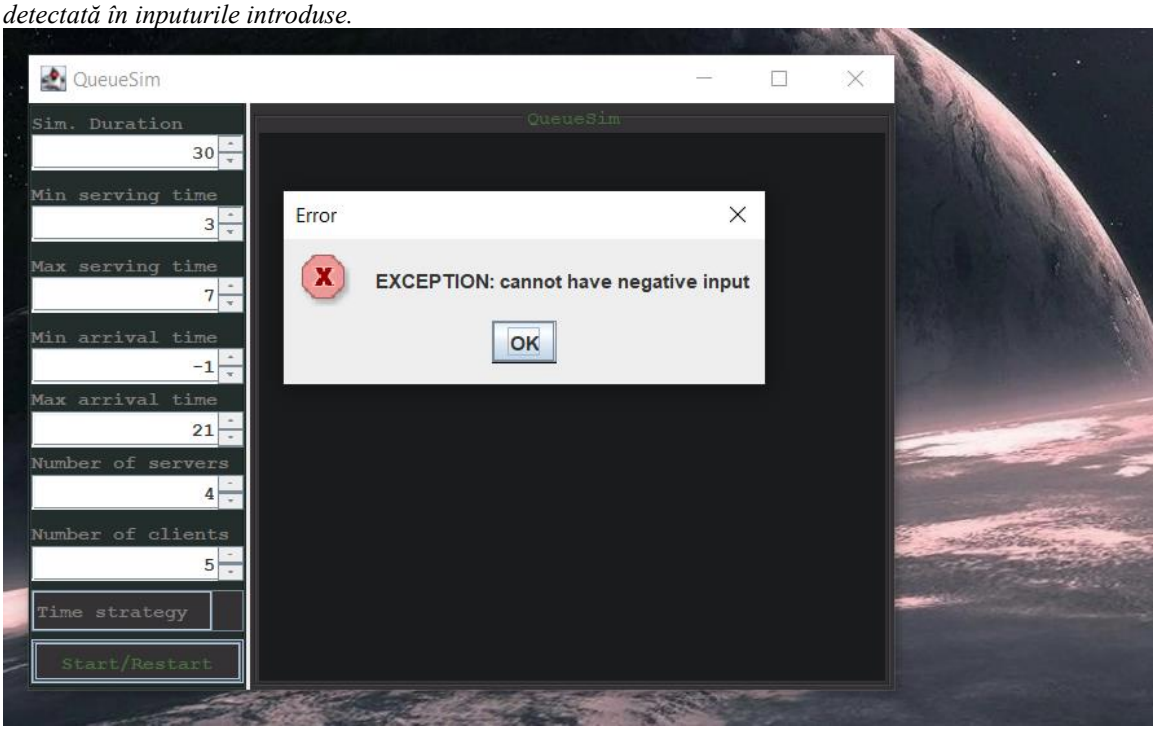
Clasa SimulationFrame:

- compune interfata grafica, trimite variabilele de input inspre SimulationManager si primeste de la acesta output-ul pe care il afiseaza in inapoi in interfata.
- are un camp special sub forma unui flag, care permite verificarea din main daca aplicatia trebuie restartata (la a doua apasare a butonului), in cazul in care utilizatorul doreste sa ruleze o simulare noua.



5. Rezultate

— *Având în vedere specificațiile cerinței, se va sublinia faptul că aplicația are o limită maximă de cozi pe care le poate suporta, aceasta fiind de 20. Dacă utilizatorul introduce un număr mai mare de cozi, aplicația nu va porni și va afișa un mesaj corespunzător. În plus, se va remarca că aplicația gestionează în mod corespunzător inputurile negative sau situațiile în care timpurile maxime sunt mai mici decât timpurile minime. În aceste cazuri, utilizatorul va fi informat printr-un dialog box cu privire la eroarea detectată în inputurile introduse.*



— *In cazul in care totul este introdus corect, simularea incepe si se termina fie atunci cand s-a scurs tot timpul, fie cand s-au servit toti clientii. In acest caz, la final se vor afisa si niste statistici despre simulare: timpul mediu de servire, timpul mediu de asteptare, "ora" de varf.*

6. Concluzii

— *În urma dezvoltării acestei teme, am învățat importanța planificării și implementării eficiente a sistemelor bazate pe cozi, precum și modul în care acestea pot fi simulate și analizate într-un mediu de laborator.*

— *În ceea ce privește dezvoltările ulterioare, există oportunități de extindere a funcționalității pentru a permite analize mai detaliate sau pentru a integra alte aspecte importante ale sistemelor de cozi, cum ar fi desenarea de grafice care sa prezinte mai clar rezultatele simularii. O alta functionalitate posibila ar fi ca utilizatorul sa aiva un mod direct de a compara diferite strategii de aderare la cozi a clientilor, pentru a gasi usor cea mai buna strategie, dupa caz. De asemenea, o interfata mai user friendly ar fi potrivita in unele situatii, cum ar fi folosirea de animatii pentru a reprezenta cozile.*

7. Bibliografie

— *Java Concurrency Utilities (JCU) - Javadoc:* *https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/package-summary.html*

— *BufferedWriter pentru adaugarea output-ului la fisierele jurnal:* *https://docs.oracle.com/javase/8/docs/api/java/io/BufferedWriter.html*

— *CountDownLatch, folosit pentru confirmarea inputului corect inainte de pornirea simularii:* *https://www.baeldung.com/java-countdown-latch*